कक्षा 11

कक्षा 11

सूचना प्रौद्योगिकी और प्रोग्रामिंग–I

सूचना प्रौद्योगिकी और प्रोग्रामिंग–I

सूचना प्रौद्योगिकी और प्रोग्रामिंग–I

कक्षा – 11

माध्यमिक शिक्षा बोर्ड राजस्थान, अजमेर

पाठ्यपुस्तक निर्माण समिति

पुस्तक : सूचना प्रौद्योगिकी और प्रोग्रामिंग–I

कक्षा – 11

संयोजक :–

डॉ. विष्णु गोयल, निदेशक
सेन्टर फॉर इलेक्ट्रोनिक गवर्नेन्स, जयपुर

लेखकगण :–

1. **डॉ. अनिल गुप्ता,** आचार्य
एम.बी.एम. इंजीनियरिंग कॉलेज,
जयनारायण व्यास विश्वविद्यालय, जोधपुर

2. **डॉ. प्रियदर्शी पाटनी,** निदेशक
लाचू मैमोरियल कॉलेज, जोधपुर

3. **प्रफुल्ल मेहता,** टी.ए.
एम.बी.एम. इंजीनियरिंग कॉलेज,
जयनारायण व्यास विश्वविद्यालय, जोधपुर

4. **अरविन्द कुमार शर्मा,** विभागाध्यक्ष कम्प्यूटर साईंस
साइन इन्टरनेशनल इंस्टीट्यूट ऑफ टैक्नोलॉजी, जयपुर

5. **मनिन्दर सिंह भुई,** सहायक आचार्य
सेन्टर फॉर इलेक्ट्रोनिक गवर्नेन्स, जयपुर

6. **योगेश शर्मा,** सहायक आचार्य
सेन्टर फॉर इलेक्ट्रोनिक गवर्नेन्स, जयपुर

पाठ्यक्रम समिति

सूचना प्रौद्योगिकी और प्रोग्रामिंग –I

कक्षा XI

संयोजक

डॉ. विष्णु गोयल
निदेशक
सेन्टर फॉर ई–गवर्नेस,
राजकीय खेतान पॉलिटेक्निक कॉलेज, जयपुर

लेखकगण :

डॉ. अनिल गुप्ता
सहायक आचार्य
कम्प्यूटर विज्ञान व अभियांत्रिकी विभाग
एम.बी.एम. अभियांत्रिकी महाविद्यालय
जोधपुर

हरजीराम चौधरी
सहायक आचार्य
राजकीय अभियांत्रिकी महाविद्यालय
अजमेर, राज.

दलपत सिंह सोनगरा
सहायक आचार्य
राजकीय महिला अभियांत्रिकी महाविद्यालय
अजमेर, राज.

अमरजीत पूनियां
सहायक आचार्य
राजकीय महिला अभियांत्रिकी महाविद्यालय
अजमेर, राज.

विष्णु प्रकाश शर्मा
सहायक आचार्य
राजकीय अभियांत्रिकी महाविद्यालय
अजमेर, राज़.

राजेश कुमार तिवारी
प्रधानाचार्य
राजकीय उच्च माध्यमिक विद्यालय
जोताया, वाया–सरवाड़ (अजमेर)

प प्रस्तावना**स्तावना**

सूचना एवं संचार प्रौद्योगिकी हमारे जीवन में एक जबरदस्त क्रांति लेकर आई है। सूचना प्रौद्योगिकी के सहारे हम निरंतर आर्थिक संपन्नता की ओर अग्रसर हो रहे हैं। इलेक्ट्रॉनिक वाणिज्य के रूप में ई–कॉमर्स, इंटरनेट द्वारा डाक भेजने के लिए ई–मेल, ऑनलाईन सरकारी कामकाज विषयक ई–शासन, ई–बैंकिंग द्वारा बैंक व्यवहार, ऑनलाईन, शिक्षा सामग्री के लिए ई–एज्यूकेशन आदि माध्यम से सूचना प्रौद्योगिकी का निरंतर विकास हो रहा है। सूचना प्रौद्योगिकी के बहु आयामी उपयोग से विकास के नये द्वार खुल रहे हैं। आज सूचना एवं संचार प्रौद्योगिकी अनुसन्धान, व्यापार, उद्योग, कृषि, चिकित्सा, शिक्षा, मनोरंजन, संचार, यातायात, पर्यावरण, मौसम विज्ञान, अन्तरिक्ष विज्ञान आदि अनगिनत क्षेत्रों में अपनी प्रभावी उपयोगिता सिद्ध कर रही है। अतः इसका मूलभूत ज्ञान आज हम सबके लिए आवश्यक हो गया है।

इसी अपरिहार्यता को दृष्टिगत रखते हुए माध्यमिक शिक्षा बोर्ड राजस्थान ने इसे अपने पाठ्यक्रम में स्थान दिया है। प्रस्तुत पुस्तक कक्षा 11 के विद्यार्थियों के लिए बोर्ड के नवीन पाठ्यक्रमानुसार लिखी गयी है। पुस्तक में तकनीकी शब्दों को अंग्रेजी में रखते हुए भी भाषा को सरल, स्पष्ट एवं बोधगम्य बनाने का प्रयास किया गया है। तकनीकी शब्द भारत सरकार द्वारा स्वीकृत शब्दावली के अनुसार दिए गए हैं। आवश्यकतानुसार अंग्रेजी शब्द कोष्ठक में भी दिए गए हैं। इस किताब का शीर्षक सूचना प्रौद्योगिकी और प्रोग्रामिंग–I रखा गया है।

इस पुस्तक में सी प्रोग्रामिंग की सम्पूर्ण जानकारी दी गयी हैं। ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग, कंप्यूटर नेटवर्किंग, एक्सएमएल (XML), डेटाबेस प्रबंधन प्रणाली (DBMS) और पीएचपी(PhP) को भी पाठ्यक्रम के अनुसार सम्मिलित किया गया है। इस पुस्तक के द्वारा छात्रों को प्रोग्रामिंग की सम्पूर्ण जानकारी मिलेगी। इस पुस्तक में परिभाषा एवं मुख्य बिन्दुओं को उदहारण के द्वारा समझाया गया हैं। लेखकों ने प्रोग्रामिंग की सूक्ष्म बातों को साधारण भाषा में समझाया है। प्रोग्रामों को उदाहरण के द्वारा विस्तृत रूप से समझाया गया है। हर अध्याय के अंत में छात्रों के लिए मुख्य बिन्दु दिए गए हैं। प्रश्न प्रत्र में पूछे जाने वाले सभी प्रकार के प्रश्न भी अध्याय के अन्त में दिए गये हैं।

इस पुस्तक में सुधार के लिए आपके सुझाव का स्वागत है।

–संयोजक एवं लेखकगण

सूचना प्रौद्योगिकी और प्रोग्रामिंग–I

विषय कोड– 03

इस विषय के दो प्रश्न पत्रों की सैद्धान्तिक एवं प्रायोगिक की परीक्षा होगी। परीक्षार्थी को दोनों में पृथक–पृथक उत्तीर्ण होना अनिवार्य है। विषय की परीक्षा योजना निम्नानुसार है –

प्रश्नपत्र	समय (घंटे)	प्रश्नपत्र के लिए अंक परीक्षा	पूर्णांक	
सैद्धान्तिक	3.15	70	70	
प्रायोगिक	3.00	30	30	100

पूर्णांक– 70

क्र सं. विषय वस्तु

यूनिट–1 : 'सी' भाषा का परिचय **06**

'सी' भाषा का परिचय, इतिहास, 'सी' प्रोग्राम की संरचना, अक्षर समूह, स्थिरांक, चर, अभिज्ञारक, डेटा प्रकार–घोषणा, कीवर्ड, इनपुट–आउटपुट फलन, ऑपरेटर, व्यंजक का प्रकार, रूपान्तरण, औपरेटरों की पूर्वता, नियंत्रक कथन, **if** कथन, **if-else** कथन, नेस्टेड **(nested) if-else** कथन, लूपिंग **(looping) for** लूप, **while** लूप और **do while** लूप, नेस्टेड लूप **(nested** लूप**)** , ब्रेककथन, **continue** कथन।

यूनिट–2 : 'सी' भाषा की प्रोग्रामिंग **08**

ऐरे : घोषणा और आरंभिकरण, एक आयामी ऐरे, द्विआयामी ऐरे और बहु आयामी ऐरे फलनः पूर्व निर्धारित फलन, उपयोगकर्ता परिभाषित फलन– घोषणा और परिभाषा, वैश्विक और स्थानीय चर, भंडारण कक्षाएं, वास्तविक और औपचारिक पैरामीटर, आव्हान, संदर्भ द्वारा आव्हान, मान द्वारा आव्हान, एक ऐरे को फलन में भेजना, प्रत्यावर्तन, संकेत का परिचय, संरचना **(structure)** संरचना के साथ ऐरे, संघ **(union)** का परिचय।

यूनिट–3 ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग **08**

ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग की अवधारणा, संरचित प्रोग्रामिंग, प्रक्रियात्मक प्रोग्रामिंग एवं ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग–अंतर, फायदे और सीमाएँ। OOPs विशेषताएं, डेटा एनेकॅप्सुलेशन (encapsulation) , डेटा सम्पुटीकरण, बहुरूपता और वंशानुक्रम।

C^{++} प्रोग्राम की संरचना – टोकन, कीवर्ड, स्थिरांक, मूल डेटा प्रकार। **cout** निर्देशोंके साथ आउटपुट और **cin** निर्देशों के साथ इनपुट।

यूनिट–4 कम्प्यूटर नेटवर्किंग **14**

नेटवर्किंग का परिचय– विशेषतायें, अवयव, नेटवर्क स्थान, डाटा संचार मॉडल, डाटा संचार प्रणाली के प्रदर्शन के अवयव, संगति, विश्वसनीयता। रिकवरी, सुरक्षा, **नेटवर्क टोपोलॉजी–** स्टार, बस, रिंग, मेश, ट्री, हाइब्रि, **मानकीकरण और प्रोटोकॉल–** मानक की आवश्यकता, मानकीकरण समितियाँ,

इंटरनेट मानक, **संचरण मोड और डाटा ट्रांसमिशन–** सिम्प्लेक्स, अर्ध डुप्लेक्स, पूर्ण डुप्लेक्स, समान्तर प्रसारण, धारावाहिक संचरण, नेटवर्क की श्रेणियां – लैन, मैन, वैन, पैन, internet works **ओ.एस.आई. मॉडल –** भौतिक, डाटा लिंक, नेटवर्क, परिवहन, सत्र, प्रस्तुति, आवेदन परतों **टीसीपी/आईपी मॉडल :नेटवर्किंग में सिगनल–** एनालॉग, डिजिटल, आवधिक, गैर आवधिक संकेतों, **नेटवर्किंग में ट्रांसमिशन मीडिया परिचय–** ट्विस्टेडके बल, कोऑक्सिअल केबल, ऑप्टिकल फाइबर केबल, रेडियो तरंग संचरण, उपग्रह संचार।

नेटवर्किंग उपकरण के बारे में परिचय– ब्रिज, हब, स्विच, राऊटर, रिपीटर्स आदि **नेटवर्किंग (मैक और आईपी) में पतों का परिचय–** क्लासेज, आईपी एड्रेस अवयब, सबनेट मास्क, आईपी एड्रेस योजना, मैक और आईपीवी 6 के बारे में परिचय, **वायरलेस के बारे में परिचय–** वायरलेय लैन टेक्नालॉजीज, **WLAN के मानकों और सुरक्षा–** आरएफ बैंड, 802.11 WLAN के ग्राहकों को नेटवर्क, प्रवेश मोड, कवरेज क्षेत्रों, डेटा दरों, WLAN के उपकरण, **कंजक्शन नियंत्रण और सेवाओं की गुणवत्ता–** डेटा यातायात, कन्जेक्शन, कन्जेक्शन नियंत्रण, लोड शेडिंग, जिटर, क्यूओएस, क्यूओएस तकनीक में सुधार, एकीकृत सेवा, डिफरेंशियल सेवाऐं : **डीएनएस और ईमेल–** डीएनएस नेमस्पेस, इंटरनेट में डीएनएस, डीएनएस संदेश, डीएनएस में संकल्प प्रक्रिया, डीएनएस विषाक्तता, ई–मेल, ई–मेल सेवाएं, ई–मेल आर्किटेक्चर, मेल सर्वर।

यूनिट–5 : एक्सएमएल 10

एक्सएमएल (XML) परिचय– एक्सएमएल (XML) परिचय और अवलोकन, मार्कअप लैंगवेज का परिचय, डेटा संरचना, एक्सएमएल दस्तावेज बनाने की प्रक्रिया, मान्य एक्सएमएल दस्तावेज बनाने की प्रक्रिया, एक्सएमएल की विशेषताएं।

एक्सएमएल दस्तावेज बनाना : एक्सएमएल घोषणा (declaration) , टैग और एलिमेंट, गुण (attributes), संदर्भ, टेक्स्ट।

DTD की मूल बातें: एक्सएमएल दस्तावेज स्कीमा की आवश्यकता, एक्सएमएल दस्तावेज स्कीमा के प्रकार, DTD स्कीमा परिभाषा : एक्सएमएल दस्तावेज, एंटिटी डिक्लेरेशन एक्सएमएल स्कीमा मूल बातें : एक्सएमएल स्कीमा, एक्सएमएल स्कीमा और DTD परिभाषाएं के बीच अंतर, एक्सएमएल स्कीमा संरचना।

एक्सएमएल नेमस्पेस: एक्सएमएल नेमस्पेस का परिचय, नेमस्पेज डिक्लेरेशन, डिफॉल्ट नेमस्पेस

यूनिट–6 : डेटाबेस प्रबंधन प्रणाली 10

डेटाबेस अवधारणा, परिचय और डेटाबेस की परिभाषा, डेटाबेस का उपयोग, डेटाबेस प्रबंधन प्रणाली (DBMS), डीबीएमएस के फायदे। डोमेन, टपल, संबंध, कुंजी, प्राथमिक कुंजी, वैकल्पिक कुंजी, उम्मीदवार कुंजी है, डेटाबेस प्रबंधन प्रणाली के उदाहरण , आरबीडीएमएस का परिचय, संरचित क्वेरी भाषा (SQL) : परिचय, एसक्यूएल के लक्षण, एसक्यूएल के प्रयोगके लाभ, डेटा परिभाषा, भाषा (DDL) डेटा मैनीपुलेशन भाषा (DML). MySQL कमांडस : CREATE TABLE, DROP TABLE, ALTER TABLE,UPDATE SET, INSERT, DELETE, SELECT, DISTINCT, FROM, WHERE, IN, BETWEEN, GROUP BY, HAVING, ORDER BY ; functions :Number, string, date.

यूनिट–7 : पीएचपी 144

पीएचपी के द्वारा वेब निर्माण– स्क्रिप्ट की मूल बातें, क्लाइंट बनाम सर्वर साइड पटकथा, पीएचपी द्वारा वेब निर्माण की आवश्यकता, स्क्रिप्ट टैग, चर की घोषणा और एक्सेस डेटाप्रकार : कोड में टिप्पणी (commenting) शैली, primitive डेटा के प्रकार, अचर घोषणा, ऐक्सेस फलन।

लूप– प्रकार for, while, do while , ऐरे एक आयाम ऐरे, बहु आयाम ऐरे , ऐर डेटा निकालने के लिए for each लूप इनबिल्ट ऐरे : $_ GET, $_ POST, $_ REQUEST आदि में PHP सशर्त Constructs : सशर्त Constructs निर्माण के विभिन्न प्रकार if - else – और else –if , switch-case कथन।

ऑपरेटर्स : गणितीय तुलनात्मक, असाइनमेंट, तार्किक, सशर्त तार्किक। ऑपरेटर की पूर्वता।

MySQL के साथ PHP के लिए कनेक्टिविटी : MySQL डाटाबेस के लिए php में कनेक्टिविटी कोड, MySQL के संचालन के लिए विभिन्न PHP स्क्रिप्ट का उपयोग जैसे : डेटा क्वेरी लैग्वेज (DQL), डेटा परिभाषा भाषा (DDL), डेटा मैनीपुलेशन भाषा (DML), डेटा नियंत्रण भाषा डेटा परिभाषा भाषा (DCL) ।

सूचना प्रौद्योगिकी और प्रोग्रामिंग–I

प्रायोगिक योजना

समय – 3 घण्टा **अंक : 30**

क्र.सं.	विषय	अंक भार
1	'C' भाषा प्रोग्रामिंग	8
2	C^{++} भाषा प्रोग्राम	4
3	XML प्रोग्राम	3
4	XQL प्रोग्राम	3
5	PHP प्रोग्राम	2
6	Practical Record	5
7	Viva	5

नोट : प्रायोगिक परीक्षा हेतु 'C' भाषा एवं 'C $^{++}$' भाषा में से एक प्रोग्राम अंक 12 का तथा XML, SQL एवं PHP में से एक प्रोग्राम अंक8 का दिया जाना चाहिये।

निर्धारित पुस्तक –

सूचना प्रौद्योगिकी और प्रोग्रामिंग–I – माध्यमिक शिक्षा बोर्ड, राजस्थान, अजमेर।

विषय सूची
INDEX

पाठ 1 'सी' भाषा का परिचय **(Introduction to 'C' Language)**	**1—38**
पाठ–2 'सी' भाषा की प्रोग्रामिंग **(Programming in 'C' Language)**	**39—85**
पाठ–3 ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग **(Object Oriented Programming)**	86—98
पाठ–4 कम्प्यूटर नेटवर्किंग (**Computer Networking**)	99—137
पाठ–5 एक्सएमएल **(XML)**	138—149
पाठ–6 डेटाबेस मैनेजमेंट सिस्टम **(DBMS)**	150—178
पाठ–7 पीएचपी **(Php)**	179—211

पाठ–1

'सी' भाषा का परिचय
(Introduction to 'C' Language)

कम्प्यूटर के लिये विभिन्न भाषाओ का विकास हुआ है। ये सभी भाषा किसी विशेष कार्य हेतु डिजाईन की गई है। मशीन भाषा (Machine Language) प्रोग्रामिंग भाषा के रूप में विकसित हुई थी। इसमे प्रोग्राम लिखना बड़ा ही कठिन होता था। क्योंकि इसमें बहुत ही विस्तारित निर्देश समूह (Instruction set) होता था। यह निर्देश समूह कम्प्यूटर के हार्डवेयर (Hardware) को सीधे संचालित करता था।

फोरट्रान (FORTRAN) का विकास वैज्ञानिक तथा गणितीय समस्याओं के समाधान के लिये तथा कोबोल (COBOL) का विकास वाणिज्य (Commercial) समस्याओं में उपयोग के लिये हुआ था। उस समय आवश्यकता के अनुसार C भाषा का विकास हुआ। इस भाषा का प्रयोग सभी प्रकार के कार्यो में किया जा सकता है।

मुख्य रूप से 'सी' भाषा सन् 1972 में डेनिस रिची द्वारा बैल टेलिफोन लेबोरट्रीज इण्डस्ट्रीज (अब यह AT & T Bell Laboratories) में BCPL (बेसिक कम्बाइण्ड प्रोग्रामिंग लैंग्वेज) तथा B की अग्रणी भाषा के रूप में विकसित हुई। इस भाषा को यूनिक्स (UNIX) तथा डॉस (DOS) ऑपरेटिंग सिस्टम दोनों पर प्रयोग किया जा सकता है, इसमें कम्पाइलर का अन्तर होता है। ANSI 'C' और TURBO 'C' भाषा का नया वर्जन C++ सन् 1980 में विकसित हुआ था।

1.1 'C' भाषा के गुण (Property of 'C' language)

1. 'C' भाषा एक लचीली भाषा है, जिसमें विभिन्न प्रकार के प्रोग्राम आसानी से बनाये जा सकते हैं।
2. इस भाषा में लिखे गए प्रोग्राम की गति तीव्र होती है।
3. यह एक उच्च स्तरीय भाषा है, लेकिन इसमें निम्नस्तरीय भाषा के गुण भी विद्यमान होते हैं।
4. इसमें लिखे हुए प्रोग्राम कम स्थान घेरते हैं।
5. इस भाषा में प्रोग्रामों को खण्डों (Blocks) के रूप में व्यवस्थित कर सकते हैं जिससे बड़े प्रोग्रामों को आसानी से बनाया जा सकता है।
6. इस भाषा में पॉइन्टर का उपयोग कर प्रोग्राम बनाये जा सकते हैं।
7. इस भाषा में बिट संकारक (0 या 1) होने के कारण बिट्स द्वारा कार्य करना सम्भव है।

1.2 प्रोग्राम की संरचना

'C' भाषा का प्रोग्राम विभिन्न फलनों का समूह है। प्रत्येक फलन वाक्यो का समूह है जो कोई एक विशेष कार्य करता है। कम्पाइलर (Compiler) main() फलन को सबसे पहले कॉल (call) करता है। अतः main( )फलन प्रत्येक प्रोग्राम में होना चाहिये। 'C' भाषा के प्रोग्राम की संरचना में हमें यह जानना अति आवश्यक है कि प्रोग्राम के कौन से वाक्यो को किस स्थान पर लिखना होगा। प्रोग्राम की संरचना निम्न प्रकार होती है।

1. प्री प्रोसेसर डायरेक्टिव तथा हैडर फाइल
   (Pre-Processor directive and Header file)
2. ग्लोबल चर और फलन प्रोटोटाइप
   (Global declarations of variables and Function Prototype)
3. *main( )* फलन
4. *{*
5. लोकल चर और फलन प्रोटोटाइप मेन फलन के लिये
   (Local declarations of variables and Function Prototype of main function)
6. एकल या मिश्रित वाक्य
   (Single or compound statements)
7. *}*
8. अन्य फलन का हैडर आरगुमेन्ट के साथ
   (Header of other function with its arguments)
9. *{*
10. फलन के लोकल चर
    (Local variables of function)
11. एकल या मिश्रित वाक्य
    (Single or compound statements)
12. *}*

वर्णन

लाइन नं. 1: हम जो लाइब्रेरी फलन अपने प्रोग्राम में सम्मिलित करते है। उसके अनुसार हम हैडर फाइल को लिखते है। जिससे हैडर फाइल के सभी फलनो की परिभाषा प्रोग्राम के साथ जुड़ जाती है। अगर हमें अपने प्रोग्राम में गणितीय फलनो को सम्मिलित (जैसे– sin, cos और sqrt) करना हो तो हम निम्न लाइन लिखते है।

#include<math.h>

एक प्रोग्राम में एक या एक से ज्यादा हैडर फाइल हो सकती है। तथा इनको किसी भी क्रम में लिख सकते है।

प्री–प्रोसेसर डायरेक्टिव **(Pre-Processor Directive)**

कम्पाइलर द्वारा कम्पाइल (compile) करने से पूर्व निष्पादित किये जाते है। ये कम्पाइलर को कुछ निर्देश प्रदान करते है ।कुछ प्री–प्रोसेसर डायरेक्टिव #include, #define आदि है।

#include प्री–प्रोसेसर हैडर फाइलो को प्रोग्राम में सम्मिलित करने हेतु उपयोग में लाया जाता है तथा #define सिम्बोलिक अचर (symbolic constant) घोषित करने हेतु उपयोग में लाया जाता है।

उदाहरण :

#define A 10

A एक अचर होगा तथा इसका मान 10 होगा।

लाइन नं. 2: इस लाइन में चरों को घोषित (declare) किया जाता है। यहाँ पर घोषित चर प्रोग्राम के किसी भी भाग में उपयोग किये जा सकते है। यह फलनों में तब ही उपयोग किये जा सकते है जबकि वहाँ उसी

नाम (identifier) का चर घोषित नहीं होता है।

उदाहरण :

int A;

float B, C;

यहाँ पर फलन के प्रोटोटाइप भी घोषित होते है। तथा यह फलन प्रोग्राम के किसी भी भाग में कॉल हो सकते है।

उदाहरण :

int sum(int a, int b);

float max(flaot a, float b, float c);

लाइन नं. 3 : यह main ( ) फलन होता है। सभी 'C' प्रोग्रामो में यह होना अति आवश्यक है। क्योंकि सर्वप्रथम कम्पाइलर main( ) फलन को ही कॉल करता है।

लाइन नं. 4 : बाँया मझंला कोष्ठक से main( ) फलन की सीमा की शुरुआत होती है।

लाइन नं. 5 : मझंला कोष्ठक के बाद घोषित चर केवल main( ) फलन में ही उपयोग कर सकते है।

इन्हें लोकल चर भी कह सकते है। यह ग्लोबल चरों (लाइन नं. 2) की तरह घोषित होते है। यहाँ पर फलन के प्रोटोटाइप भी घोषित होते है तथा यह फलन प्रोग्राम में केवल मेन फलन में ही कॉल हो सकते है।

लाइन नं. 6 : यहाँ पर main( ) फलन की परिभाषा लिखते है। जिसमें एकल या मिश्रित वाक्य हो सकते है।

लाइन नं. 7: दाँया मझंला कोष्ठक से main( ) फलन की सीमा को बन्द करते है।

लाइन नं. 8, 9, 10, 11, और 12 : यह लाइनें, लाइन नं. 3, 4, 5, 6, 7 के जैसी है। इनमें लाइन नं. 8 पर फलन का शीर्ष (Header) लिखते है और सभी लाइनें main( ) फलन के बाद वाली लाइनो जैसी है।

इन लाइनों को प्रोग्राम में जब लिखते है जब कोई उपयोग कर्ता घोषित फलन (user define function) प्रोग्राम में लिखते है।

'C' प्रोग्राम में टिप्पणी का उपयोग दस्तावेज बनाने के लिये किया जाता है। टिप्पणियों को कम्पाइलर अनदेखा कर देता है। इसे /* तथा */ के बीच में लिखा जाता है।

प्रोग्राम 1 : किसी आयत का क्षेत्रफल ज्ञात करने हेतु 'C' भाषा में प्रोग्राम लिखिए।

```
#include<stdio.h>
/* To calculate the area of Rectangle. */
main()
{
        float a,b,area;
        printf("Enter Side of the Rectangle :\n");
        scanf("%f%f",&a,&b);
        area = a * b;
        printf("\nThe area of the Rectangle = %f",area);
}
```

Input/output of the above program:

Enter Side of the Rectangle : 12 14

The area of the Rectangle = 168

प्रोग्राम 2: दिए गए सूत्र की गणना करने हेतु 'C' भाषा में प्रोग्राम लिखिए।

$$c = \frac{1}{2}mv^2$$

```
#include<stdio.h>
main()
{
        float m,v,c;
        printf("Enter The value of m & v :\n");
        scanf("%f%f",&m,&v);
        c =0.5 * m * v * v;
        printf("\nThe value of c = %f",c);
}
```

Input/output of the above program:

Enter The value of m & v : 2.5 1.7

The value of c = 3.612500

1.3 'C' का करैक्टर सैट (Character set of 'C') :

'C' भाषा में निम्न तथा उच्च वर्णमाला (Lower and upper case letters), अंक (digits) तथा विशेष चिन्ह आते हैं।

अक्षर	A, B, C, ... Z
	a, b, c, ... z
अंक	0, 1, 2, 3, ... 9

विशेष चिन्ह नीचे सारणी–1.1 में दिये गये हैं।

सारणी–1.1

चिन्ह	अर्थ	चिन्ह	अर्थ
#	हैश का चिन्ह	?	प्रश्नवाचक चिन्ह
<	लैस देन	:	कॉलन
>	ग्रेटर देन	;	सेमी कॉलन
=	इक्वल टू	^	कैरेट
+	जोड़ का चिन्ह	&	एम्परसैन्ट
-	घटाने का चिन्ह	~	टाईल्ड
*	एस्ट्रिक	$	डॉलर
%	प्रतिशत	,	कोमा
/	फ्रंट स्लेश	\	बैक स्लैश
@	एट दा रेट	' '	सिंगल कोट
" "	डबल कोट	!	विस्मय बोधक
|	लम्बवत् बार		

1.4 अभिज्ञानक **(Identifier)**

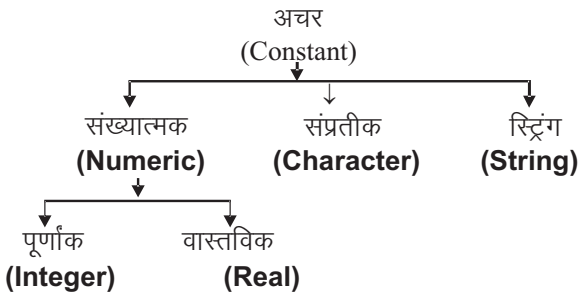
प्रोग्राम के विभिन्न तत्वों जैसे चर का नाम (variable name), फलन का नाम (Function name) तथा सिम्बोलिक अचर का नाम (Symbolic Constant Name) इत्यादि को यूजर द्वारा दिये गए नाम को अभिज्ञानक (Identifier) कहा जाता है।

इसको लिखने के लिए निम्न नियम होते हैं :

1. अभिज्ञानक का प्रथम अक्षर अंग्रेजी वर्णमाला का अक्षर होना चाहिए।
2. प्रथम अक्षर के बाद अभिज्ञानक में अंग्रेजी वर्णमाला के अक्षर (A..Z और a..z), अंक (0 से 9) तथा रेखांकन चिन्ह (_) आ सकते हैं।
3. अभिज्ञानक में अधिकतम 31 अक्षर हो सकते हैं। (कम्पाईलर पर निर्भरता)
4. आरक्षित शब्द अभिज्ञानक के रूप में उपयोग नहीं कर सकते हैं।
5. सी भाषा के केस संवेदनशील होने के कारण इसमें उच्च व निम्न वर्णमाला अक्षर अलग–अलग माने जाते हैं। अतः अधिकतर निम्न वर्णमाला अक्षरों का ही उपयोग करना चाहिये।

1.5 अचर **(Constant)**

ऐसे अभिज्ञानक जिनका मान पूरे प्रोग्राम में स्थिर होता है अचर कहलाते हैं। अचर तीन प्रकार के होते हैं।



चित्र 1.1: विभिन्न प्रकार के अचर

1. संख्यात्मक अचर **(Numeric Constant)**

   ये दो प्रकार के होते हैं।

(a) पूर्णांक स्थिरांक **(Integer Constant)**

ये केवल अंकों से मिलकर बनते है और इसमें दशमलव नहीं होता है। इनको तीन विभिन्न अंक प्रणालियों में प्रदर्शित किया जा सकता है :

(i) दशमलव (Base 10)

(ii) ऑक्टल (Base 8)

(iii) हैक्सा डेसीमल (Base 16)

विभिन्न अंक प्रणालियों में काम आने वाले अंक तथा वर्णमाला के अक्षर सारणी–1.2 में दर्शाये गए है।

सारणी – 1.2

Type	Digits / alphabet	Example
Decimal	0, 1, 2, ... 9	0, 279, 972, 32767

Octal	0, 1, 3, ... 7	02, 027, 07777 ऑक्टल नम्बर 0 (शून्य) से शुरू होता है।
Hexa decimal	0, 1, 2, ... 9 and a, b, c .... (upper and lower case)	0x1, 0xab, 0xffff हैक्सा डेसीमल नम्बर 0x या 0X से शुरू होता है।

(b) वास्तविक स्थिरांक **(Real Constant)**

जिन संख्यात्मक अक्षरों में दशमलव होता है, वे वास्तविक स्थिरांक कहलाते हैं।

उदाहरण :

0.96, 872.127, 2.0 E-7

0.0693, 1.7676E + 10

2. करैक्टर स्थिरांक **(Character Constant)**

जब एक ही अक्षर एकल कोट (Single Quotes) में लिखते हैं तो वह एकल करैक्टर स्थिरांक कहलाता है। इसमें अंग्रेजी वर्णमाला के अक्षर, अंक तथा विशेष चिन्ह आ सकते हैं।

उदाहरण : 'A' '5' 'b' '$'

3. स्ट्रिंग स्थिरांक **(String Constants)**

जब शून्य, एक या, एक से अधिक करैक्टर डबल कोट में लिखते है तो वह स्ट्रिंग स्थिरांक कहलाता है।

उदाहरण :

"Home Loan" "ALWAR"

" " "2 * 6 * 7 - 5 + 2"

4. बैक स्लैश स्थिरांक **(Back Slash Character Constant)**

यह भी एक प्रकार के करैक्टर स्थिरांक है जो कि प्रिन्ट नहीं होते है। ये करैक्टर स्थिरांक कन्ट्रोल करने के उपयोग में आते हैं। इसको बैक स्लैश (\) के साथ लिखते है। कुछ बैक स्लैश स्थिरांक व उनके अर्थ सारणी–1.3 में दिये गये हैं।

सारणी – 1.3

Escape Sequence	Meaning
\a	bell (alert)
\t	horizontal tab
\v	vertical tab
\n	new line
\v	carriage return
\'	quotation mark
\b	back space
\0	NULL

1.6 चर (variable)

चर वे अभिज्ञानक होते हैं जिनमें मानों (values) का संचय होता हैं तथा प्रोग्राम निष्पादन में इनका

मान परिवर्तित हो सकता है। इसको लिखने के लिए अभिज्ञानक लिखने वाले नियम ही लागू होते हैं कुछ वैद्य चर निम्न है :

employee, result123, roll_num, min

तथा कुछ अवैध चर निम्न है :

short	आरक्षित शब्द
b's	विशेष संप्रतीक
5ort	प्रथम संप्रतीक वर्णमाला अक्षर होना चाहिए।
int eger	खाली स्थान

1.7 आंकड़ों का प्रकार (Data Type)

'C' प्रोग्रामिंग भाषा विभिन्न प्रकार के आंकड़ों का समर्थन करती हैं। आंकड़ों के इन प्रकारों का मेमोरी में साइज (size) भी भिन्न–भिन्न होता है। 'सी' भाषा में सामान्यतया चार प्रकार के आँकड़े काम में लिये जाते हैं।

1. int	पूर्णांक मान संचय के लिए
2. char	एक संप्रतीक को संचय करने के लिए
3. float	इकहरी शुद्धता वाली फ्लोटिंग पाईंट संख्या
4. double	दोहरी शुद्धता वाली फ्लोटिंग पाईंट संख्या

कुछ आंकड़ों के प्रकार, आकार, की–शब्द व अंक सीमाएँ सारणी–1.4 दी गई है।

सारणी – 1.4

Data type	key word	Size (in bits)	Range
Character	char	8	-128 to +127
	unsigned char	8	0 to 255
Integer	int	16	-32768 to +32767
	long int	32	-2, 147, 483, 648 to +2,147, 483, 647
	unsigned int	16	0 to 65535
	unsigned long int	32	0 to 4, 294, 967, 295
Float	float	32	3.4E-38 to 3.4E+38
Double	double	64	1.7E-308 to 1.7E+308
	long double	80	3.4E-4932 to 1.7E +4932

1.8 घोषणा (Declaration)

सभी चरों का निष्पादन होने से पहले उन्हें प्रोग्राम में घोषित करना आवश्यक होता है। चरों को प्रोग्राम में सबसे ऊपर (जैसा की प्रोग्राम की संरचना में दिखाया गया है) घोषित करना होता है। इसका रुप निम्न प्रकार है :

<data type> <variable name>;

उदाहरण :

```
int a, b, c;
char cl;
```

1.9 की–शब्द (key word)

ऐसे शब्द जिनका अर्थ तथा उपयोग 'सी' भाषा के लिए पहले से ही निश्चित किया गया हो की–शब्द (या आरक्षित शब्द) कहलाते है। की–शब्द का उपयोग 'सी' भाषा में निश्चित अर्थ के लिए होता है। की–शब्द किसी अभिज्ञानक के रूप में उपयोग नहीं हो सकता है। 'सी' भाषा में कुल 32 की–शब्द होते हैं। इनको सारणी– 1.5 में दर्शाया गया है।

सारणी – 1.5

auto	float	const	struct
break	for	continue	switch
case	goto	default	typedef
char	if	do	union
double	int	short	unsinged
else	long	singned	void
enum	register	sizeof	volatile
extern	return	static	while

1.10 ऑपरेटर्स (Operators)

कम्प्यूटर द्वारा गणनायें कराने के लिये प्रोग्राम में कुछ प्रतीकों (symbols) का उपयोग किया जाता है। इन प्रतीकों को संकारक (operator) कहते है। संकारक यह बताता है कि कौन सी संक्रिया करनी है। चरों, अचरों तथा संकारकों की सहायता से व्यंजक (expression) बनाया जाता है। 'C' भाषा में एकल (Unary), द्विआधारी (Binary) तथा त्रिआधारी (Ternary) संकारक होते है।

'C' भाषा में उपयोग आने वाले ऑपरेटर निम्न वर्गों में बाँटे गये हैं –

1. अंकगणितीय संकारक (Arithmatic Operators)
2. तुलनात्मक संकारक (Relational Operators)
3. तार्किक संकारक (Logical Operators)
4. कन्डीशनल संकारक (Conditional Operators)
5. वृद्धि व ह्रास संकारक (Increment and Decrement Operator)
6. बिटवार संकारक (Bitwise Operators)
7. निर्धारण संकारक (Assignment Operators)

1.10.1 अंकगणितीय संकारक (Arithmatic Operator)

अंकगणितीय संकारकों का उपयोग संख्यात्मक गणनायें करने के लिए किया जाता है। 'सी' भाषा में उपयोग में आने वाले अंकगणितीय ऑपरेटर निम्न हैं।

सारणी – 1.6

ऑपरेटर	उद्देश्य
+	जोड़ना
-	घटाना
*	गुणा
/	भाग
%	शेषफल (भाग करने के बाद बचा शेष)

'C' भाषा में कोई घातीय (exponential) संकारक नहीं है घात के लिए pow(x, y) फलन होता है जो math.h हैडर फाइल में उपलब्ध होता है।

1.10.2 अंकगणितीय व्यंजक **(Arithmatic Expression)**

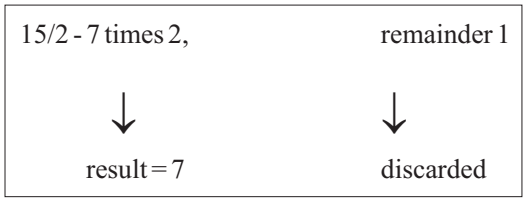
चरों तथा अचरों को संकारको की सहायता से जोड़कर व्यंजक बनाये जाते हैं। यदि दोनो ऑपरेंड (operand) पूर्णांक चर या अचर होते हैं तो वह व्यंजक पूर्णांक व्यंजक (Integer Expression) कहलाते हैं।

उदाहरण : यदि a = 15 और b = 2 हो तो निम्न व्यंजकों का परिणाम होगा।

व्यंजक	परिणाम
a + b	17
a - b	13
a * b	30
a / b	7
a % b	1

a/b का वास्तविक परिणाम 7.5 होता है। लेकिन यदि दोनों ऑपरेंड पूर्णांक चर (या अचर) होते हैं तो परिणाम भी पूर्णांक संख्या में ही आता है अतः 15 / 2 का परिणाम 7 आयेगा।

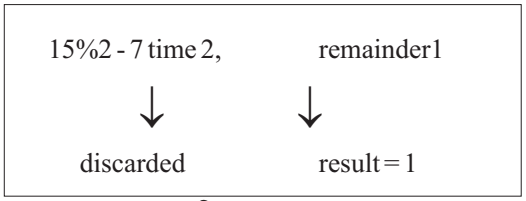
Integer division



चित्र – 1.2

मॉड्यूल्स ऑपरेटर (%), भाग ऑपरेटर का उल्टा होता है। इसमें पूर्णांक भाग (Integer division) होने के बाद शेषफल परिणाम होता है।

Modulus Operation



चित्र – 1.3

मॉडूल्स ऑपरेटर (%) के अन्य उदाहरण निम्न है :

-15%2 =-1

-15%-2 =-1

15%-2 = 1

यदि दोनों ऑपरेंड वास्तविक संख्याएँ हो तो उनका परिणाम भी वास्तविक सँख्या होगी। इस तरह के व्यंजक को वास्तविक व्यंजक (real expression) कहते हैं। यदि इसमें एक संख्या वास्तविक तथा दूसरी संख्या पूर्णांक संख्या हो तो परिणाम वास्तविक संख्या होगी तथा इस व्यंजक को मिक्स मोड़ व्यंजक कहते हैं।

उदाहरण :

यदि a=15.5, b=4.2 तो

a+b = 15.5 + 4.2 = 19.700000

a*b = 15.5 * 4.2 = 65.100000

उपरोक्त दोनों ऑपरेंड वास्तविक (real) संख्याएं है।

उदाहरण :

यदि a=15.5 (real), b = 4 (integer) हो तो

a+b=15.5 (real)+4(int)=19.500000 (real)

a*b =15.5 (real) * 4 (int)=62.000000 (real)

उपरोक्त दोनों व्यंजक मिक्स मोड़ व्यंजक है।

1.10.3 तुलनात्मक संकारक **(Relational Operators)**

तुलनात्मक संकारकों का उपयोग दो मानों की तुलना करने में होता है। इन सभी संकारकों को दो ऑपरेंड की आवश्यकता होती है। यदि तुलना करने पर सम्बन्ध सही (True) है तो यह 1 रिटर्न करता है। अन्यथा गलत (False) होने पर 0 रिटर्न करता है। इसमें दोनों ऑपरेड एक ही प्रकार के होने आवश्यक है। सी भाषा में उपयोग में आने वाले तुलनात्मक ऑपरेटर निम्न है।

सारणी 1.7

संकारक **(operater)**	अर्थ **(Meaning)**
<	less than
<=	less than equal to
==	equal to
!=	not equal to
>	greater than
>=	greater than equal to

उदाहरण :

यदि p, q और r पूर्णांक चर है तथा इनका मान 1, 2 और 3 क्रमशः है। तो नीचे दी गई सारणी में तुलनात्मक व्यंजकों का मान निम्न होगा।

सारणी 1.8

Relational expression	**Result**	**value**
p<q	true	1

(p+q)>=r	true	1
(p+5)==r+4	false	0
r!=5	true	1
p*q>q*r	false	0

1.10.4 तार्किक संकारक **(Logical Operator)**

तार्किक संकारक दो ऑपरेंड पर कार्य करते हैं। ये ऑपरेंड तार्किक या तुलनात्मक व्यंजक भी हो सकते हैं। तार्किक व्यंजकों का परिणाम True या False में आता है। C भाषा में तीन तार्किक ऑपरेटर होते हैं।

सारणी 1.9

संकारक	अर्थ
& &	and
||	or
!	not

तार्किक **And (&&)**

तार्किक and (&&) का परिणाम True तभी होगा जब दोनों ऑपरेड True हो अन्यथा परिणाम False होगा।

उदाहरण :

यदि i=9 तथा j=8.5 है तब निम्न व्यंजक का मान होगा

(i > 6) && (j < 9.5)

इसका मान True होगा क्योंकि दोनों व्यंजकों का मान True है।

तार्किक **OR (||)**

तार्किक OR का परिणाम True होगा यदि इसके दोनों ऑपरेंड्स में से एक भी ऑपरेंड का मान True हो। अन्यथा परिणाम False होगा।

तार्किक **Not (!)**

NOT एक एकल संकारक है। अतः इसमें केवल एक ही ऑपरेन्ड की आवश्यकता होती है। यदि ऑपरेंड True है तो यह परिणाम False देता है और ऑपरेंड False होने पर परिणाम True देता है।

1.10.5 तार्किक व्यंजक **(Boolean Expression)**

तार्किक व्यंजक का मान सही (True) या गलत (False) होता है। बूलियन व्यंजक एक बूलियन चर, अचर या अचर और चर तुलनात्मक संकारको द्वारा जुड़े हो सकते है। दो बूलियन व्यंजक को तार्किक संकारको के द्वारा जोड़ा जा सकता है।

तार्किक व्यंजको के उदाहरण निम्न है–

(i) age>55 && salary<1000

(ii) number<0 && number>100

(iii) !(status==1)

उदाहरण :

माना i=7, f=35 और c = 'W' है तो। अनेक जटिल तार्किक व्यंजक उनके परिणाम के साथ दर्शाये गयें हैं।

Boolean Expression	Result	Value
(i >=6) && (c ==‘W’)	True	1
(f >11) && (i >100)	False	0
c = ! (‘a’ | | (1 + f) <=10)	True	1
c = = i | | f > 20	False	0

1.10.6 कन्डीशनल संकारक (?और :) **(conditional operator)**

यह एक त्रिआधारी (ternary) संकारक है अतः इस संकारक में तीन ऑपरेंड होते हैं। इनको निम्न प्रकार से लिखा जाता है–

variable = exp1 ? exp2 : exp3 ;

exp1 एक तार्किक व्यंजक (Logical Expression) होता है। सबसे पहले व्यंजक exp1 के मान की गणना होती है यदि यह मान True होता है तो exp2 व्यंजक का मान variable में स्टोर होता है अन्यथा इसमें exp3 व्यंजक का मान स्टोर होगा।

उदाहरण :

यदि a = 5, b = 9

c = a > b ? a : b ;

इसमें c का मान 9 होगा। क्योंकि a > b False होने के कारण व्यंजक exp3 का मान c में स्टोर करेगा जो कि b है अतः b का मान 9, c में स्टोर होगा।

1.10.7 वृद्धि व ह्रास संकारक (++ और ––) **(Increment and decrement operators)**

‘C’ भाषा में दो अत्यन्त उपयोगी संकारक दिये गये हैं :

(i) वृद्धि संकारक (Increment Operator) - ++

(ii) ह्रास संकारक (Decrement Operator) - -

ये दोनों ऑपरेटर एकल (unary) है अतः इन्हें केवल एक ही ऑपरेंड की आवश्यकता होती है। इसमें वृद्धि संकारक (++) चर के मान को 1 बढ़ा देता है और ह्रास (—) संकारक चर के मान को 1 घटा देता है। इन संकारकों को निम्न प्रकार से लिखा जाता है।

++A या A++

तथा

--A या A--

यदि चर के बायें इस ऑपरेटर का प्रयोग करते हैं तो यह ऑपरेटर पहले चर में एक जोड़ता है अथवा घटाता है। यदि यह ऑपरेटर दायें ओर प्रयोग करते हैं तो यह ऑपरेटर बाद में 1 घटाता है या जोड़ता है पहले यह संक्रिया में भाग लेता है।

उदाहरण :

```
int A = 10;
A++;
```

1.10.8 निर्धारण संकारक **(Assignment Operators)**

किसी भी चर का कोई मान निर्धारित करने के लिए निर्धारण संकारक का उपयोग किया जाता है। इसको “=“ चिन्ह द्वारा प्रदर्शित किया जाता है। यह संकारक अपने दांये भाग के व्यंजक का मूल्यांकन

करके परिणाम को बांयी ओर लिखे चर को निर्धारित कर देता है। इसका सामान्य रूप निम्न है –

चर = व्यंजक

उदाहरण :

निम्न कथन निर्धारण संकारक के उदाहरण है :

area = 2 * P1 * r1

x = y

a = 7.52

Total = A[1] + A[2]

उपरोक्त कथनों में बायें भाग में चर (अचर व व्यंजक नहीं हो सकता है) होना चाहिए।

'C' भाषा में विशेष निर्धारण संकारक होते हैं वह निम्न है–

+=, *=, %=, -=, /=

उदाहरण :

x = x + 4 को x += 4 लिखते हैं।

x = x * 5 को x *= 5 लिखते हैं।

x = x % 4 को x %= 4 लिखते हैं।

x = x - 5 को x -= 5 लिखते हैं।

x = x / 4 को x /= 4 लिखते हैं।

1.10.9 व्यंजक का प्रकार बदलना **(Type Conversion of expression)**

इसका उपयोग किसी भी व्यंजक के परिणाम को आंकड़ों के प्रकार (Data type) में बदलने के लिए किया जाता है। इसे निम्न प्रकार से लिखा जाता है:

(आंकड़ों का प्रकार) व्यंजक

(data type) expression, व्यंजक का मान कोष्ठक में दिये गये आंकड़ों के प्रकार में बदल देता है।

यदि किसी व्यंजक में एक से अधिक ऑपरेंड हो तो व्यंजक का मान उसमें उपस्थित ऑपरेडो में से सबसे ऊंचे स्तर के ओपरेन्ड के बराबर होगा। इस प्रकार को इम्पलिसिट (implicit type casting) प्रकार बदलना कहते हैं।

उदाहरण :

यदि a = 7 (पूर्णांक) b = 9.5 (फ्लोटिंक पाईंट) हो तो व्यंजक 2a + b का मान फ्लोटिंग पाईंट में मान 23.500000 होगा।

अगर किसी ऑपरेड का मान प्रबलता से बदलना हो तो उसे एक्सप्लिसिट (Explicit Type Casting) प्रकार बदलना कहते हैं।

उदाहरण :

यदि a = 7 (पूर्णांक) b = 3 (पूर्णांक) हो तो व्यंजक a/b का मान पूर्णांक में 2 होगा। यदि इसका परिणाम फ्लोटिंग पाईंट में लाना है तो इस व्यंजक को निम्न प्रकार लिखना होगा तथा इसका परिणाम 2.333333 होगा।

(float)a/b;

प्रोग्राम 3 :

```
/* Program to convert a centimeter into meter-
 centimeter with integer Arithmatic*/
 #include<stdio.h>
 main()
 {
        int cent,meter;
        printf("Enter the value = ");
        scanf("%d",&cent);
        meter = cent / 100;
        cent = cent % 100;
        printf("\nMeters = %d and Centimeter = %d",meter,cent);
 }
```

उपरोक्त प्रोग्राम का आउटपुट निम्न होगा।

```
Enter the value = 1056
Meters = 10 and Centimeter = 56
```

प्रोग्राम 4 :

```
/* Program to explain explicit and implicit conversion */
#include<stdio.h>
main()
{
        int A=7,B=8;
        float X,Y=7.5;
        X = A + Y; /*Implicit Type Conversion*/
        printf("X = %f\n",X);
        X = (float)A/B; /*Explicit Type Convesion*/
        printf("X = %f\n",X);
}
```

उपरोक्त प्रोग्राम का आउटपुट निम्न होगा।

```
X = 14.500000
X = 0.875000
```

1.11 संकारकों की प्रायिकता **(Precedence of Operators):**

यदि किसी व्यंजक में एक से अधिक संकारक होते हैं तो यह आवश्यक हो जाता है कि उनको किस क्रम में निष्पादित करना है। यह क्रम निश्चित होता है कि कौनसा संकारक पहले हल किया जायेगा। इसके लिए इन संकारकों की प्राथमिकता निश्चित कर दी गयी है। किसी व्यंजक में दो संकारक की प्रायिकता समान (equal) होने पर उनका साहचर्य नियम के अनुसार सरलीकरण किया जाता है :

सारणी : 1.10 संकारकों की प्रायिकता व साहचर्य नियम

(Precedence of Operators and their associativity rules)

प्रायिकता समूह	संकारक	साहचर्यता
फलन, एरे, struct सदस्य तथा संकेतक	() [ ] . →	L→R
एकल संकारक	- ++ -- ! ~ * & sizeof(type)	R→L
अंकगणितीय गुणन, भाजक तथा शेषफल	* / %	L→R
अंकगणितीय योग तथा घटा	+ -	L→R
बिटवार शिफ्ट संकारक	<< >>	L→R
तुलनात्मक संकारक	< <= > >=	L→R
बराबरता संकारक	== !=	L→R
बिटवार AND	&	L→R
बिटवार XOR	^	LàR
बिटवार OR	|	L→R
तार्किक AND	&&	L→R
तार्किक OR	||	L→R
कन्डीशनल संकारक	? :	R→L
निर्धारण संकारक	= += -= *= /= %=&= ^= != <<= >>=	R→L
कौमा संकारक	,	L→R

उदाहरण :

यदि X = 7 y = 3.0 Z = 2 A = 2.5 B = 7 हो तो निम्न व्यंजक को हल करो।

X + Y / (Z * A + B / Z)

हल –

X + Y / (Z * A + B/Z)
= 7 + 3.0 / (2 * 2.5 + 7/2)
= 7 + 3.0 / (5.0 + 7/2)
= 7 + 3.0 / (5.0 + 3)
= 7 + 3.0 / 8.0
= 7 + 0.375
= 7.375

प्रोग्राम 5 :

```
/* Program to calculate a expression */
#include<stdio.h>
main()
{
        float A,B=2.5,D=9.25;
        int i = 5,j=10;
        A = (float)i/j + (B*2)/(i + j) + D;/* Explicit Type Conversion*/
        printf("Result = %f\n",A);
}
```

उपरोक्त प्रोग्राम का आउटपुट निम्न होगा।

A = 10.083333

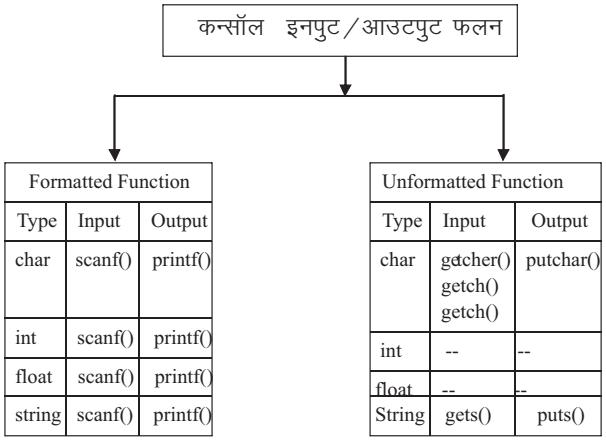
1.12 इनपुट–आउटपुट फलन (Input-Output Function) :

साधारणतया किसी भी प्रोग्राम में आंकड़ों को पढ़ना, उस पर प्रक्रिया करना तथा आउटपुट देना, यह तीन कार्य किये जाते हैं। प्रोग्राम को इनपुट और आउटपुट देने के लिए इनपुट–आउटपुट फलनों का उपयोग किया जाता है। यह फलन है scanf(), printf(), getchar(), putchar()। इनपुट/आउटपुट दो प्रकार के होते हैं। संरूपित (formatted) तथा असंरूपित (unformatted)। printf तथा scanf संरुपित तथा putchar तथा getchar असंरूपित I/O की श्रेणी में आते हैं।

1.12.1 कन्सॉल इनपुट/आउटपुट (Console I/O) :

कन्सॉल इनपुट/आउटपुट फलन को भी दो वर्गो में विभक्त किया जाता है–

1. अनफारमेटेड कन्सॉल इनपुट/आउटपुट फलन
   (Unformatted console input/ output functions)
2. फारमेटेड कन्सॉल इनपुट/आउटपुट फलन
   (Formatted console input/ output functions)



चित्र 1.4

इन दोनो में मुख्य अन्तर यह है कि फारमेटेड कन्सॉल इनपुट कीबोर्ड से डाटा लेता है तथा आउटपुट VDU पर डाटा यूजर की आवश्यकता के अनुसार फारमेटेड रुप में भेजता है जबकि अनफारमेटेड में एक ही प्रकार से डाटा प्राप्त कर सकते है और भेज भी सकते है। जैसे हमें Total_item तथा sale_item को VDU पर प्रिन्ट करना है तो वह VDU पर कहाँ पर प्रिन्ट होगा तथा इनके मध्य कितनी जगह होगी यह हम केवल फारमेटेड कन्सॉल इनपुट/आउटपुट के द्वारा ही कर सकते है। नीचे चित्र में दोनो वर्गो के फलनो को दिखाया गया है।

1.12.2 आउटपुट फलन **printf (Output Function printf)**

आउटपुट फलन के द्वारा कम्प्यूटर मेमोरी से मानक आउटपुट युक्ति (Output device) या आउटपुट फाइल में भेजा जा सकता है। 'सी' भाषा में printf() फलन होता है जो आंकड़ों को कम्प्यूटर की स्मृति (Memory) से आउटपुट युक्ति पर भेजता है। printf() फलन निम्न प्रकार से लिखा जाता है।

printf ("नियन्त्रण स्ट्रिंग", चर1, चर2, चर3) ;

चर1, चर2 तथा चर3 वे चर है जिनके मान आउटपुट युक्ति पर भेजना है। नियन्त्रण स्ट्रिंग को डबल कोट्स (" ") के अन्दर लिखा जाता है। नियन्त्रण स्ट्रिंग चरों को आंकड़ों के प्रकार के अनुसार लिखा जाता है। जैसे integer आंकड़े के लिए %d लिखा जाता है। नीचे दी गयी सारणी 1.11 में प्रयोग किये जाने वाले नियन्त्रण स्ट्रिंग के प्रारुप को दर्शाया गया है।

सारणी – 1.11

प्रारूप (कन्ट्रोल स्ट्रिंग)	चर के मान का प्रकार
%d	int (दशमलव)
%x	int (हैक्सा डेसीमल)
%o	int (ऑक्टल)
%u	unsigned int
%ld	long int
%f	float
%e	float (मान घातांक)
%lf	double float
%Lf	long double
%c	character
%s	string (स्ट्रिंग)

उदाहरण :

A = 20, B = 30

printf("%d %d",A,B)

Output : 20 30

A और B पूर्णांक प्रकार के चर है तथा कन्ट्रोल स्ट्रिंग के अनुसार A व B के मान प्रिन्ट होते हैं। printf फलन के द्वारा किसी भी स्ट्रिंग को ज्यो का त्यो प्रिन्ट करता है।

उदाहरण :

printf ("C is a good programming language.")

Output : C is a good programming language.

1.12.3 संरूपित **printf** फलन **(Formatted printf function)**

जब printf ( ) फलन में लिखे चरों के मानों को संरूपित रूप में लिखना होता है तो कन्ट्रोल स्ट्रिंग में इसके लिए अलग से कुछ करेक्टर लिखने होते हैं।

पूर्णांक नम्बरों के लिए आउटपुट

Integer नम्बरों को प्रिन्ट करने के लिए कन्ट्रोल स्ट्रिंग में निम्न प्रारूप लिखना होता है।

%wd

जहाँ पर w एक पूर्णांक नम्बर है जो यह दर्शाता है कि आउटपुट कितने कॉलम में लिखना है।

उदाहरण :

```
int A=1476
main()
{
    printf("%7d",A);
{
```

उपरोक्त प्रोग्राम का आउटपुट होगा।

			1	4	7	6

यह आउटपुट सात कॉलम में लिखा जायेगा। लेकिन नम्बर चार कॉलम में आ जाता है। अतः शुरू के तीन कॉलम खाली रहेगें।

वास्तविक नम्बरों के लिए आउटपुट

Real नम्बरों को प्रिन्ट करने के लिए कन्ट्रोल स्ट्रिंग में निम्न प्रारूप लिखना होता है।

% w.pf

जहां w व p दो पूर्णांक नम्बर है। w यह दर्शाता है कि आउटपुट कितने कॉलम में लिखना है तथा p यह दर्शाता है कि दशमलव बिन्दु के बाद कितने कॉलम होने चाहिए।

उदाहरण :

```
float x = 17.7927 ;
main()
{
   printf("7.3f", x) ;
}
```

उपरोक्त प्रोग्राम का आउटपुट होगा :

	1	7	.	7	9	2

यह आउटपुट 7 कॉलम में लिखा जाएगा। दशमलव बिन्दु के बाद तीन कॉलम का उपयोग होगा, एक कॉलम दशमलव बिन्दु के लिए तथा दो कॉलम दशमलव बिन्दु के पहले वाले नम्बर के लिए उपयोग होगा। क्योंकि इसके लिए 6 कॉलम की आवश्यकता होगी अतः एक कॉलम खाली रहेगा। जैसा कि आउटपुट में दर्शाया गया है।

स्ट्रिंग के लिए आउटपुट :

स्ट्रिंग को प्रिन्ट करने के लिए कन्ट्रोल स्ट्रिंग में निम्न प्रारूप लिखना होता है।

% w.ps

जहां w व p दो पूर्णांक नम्बर है। w यह दर्शाता है कि आउटपुट कितने कॉलम में लिखना है तथा p

यह दर्शाता कि स्ट्रिंग में से कितने करेक्टर प्रिन्ट करने हैं।

उदाहरण :

```
char str[9] = "computer" :
main()
{
        printf("\n%12.5s", str) ;
        printf("\n%.5s", str) ;
        printfl("\n%12s", str) ;
}
```

उपरोक्त प्रोग्राम का आउटपुट होगा :

							c	o	m	p	u

c	o	m	p	u

				c	o	m	p	u	t	e	r

प्रथम printf() कथन कुल 12 कॉलम में आउटपुट लिखेगा तथा स्ट्रिंग के केवल पाँच करेक्टर्स को ही प्रिन्ट करेगा। द्वितीय printf( ) कथन केवल पाँच करेक्टर्स, प्रथम पाँच कॉलम में प्रिन्ट करेगा तथा तृतीय printf() कथन 12 कॉलम में से 8 कॉलम में computer स्ट्रिंग को प्रिन्ट करेगा एवं 4 कॉलम खाली रहेंगे। जैसा कि आउटपुट में दर्शाया गया हैं।

एकल करैक्टर को उपयुक्त स्थान पर रखने के लिए निम्न प्रारूप होता है।

%wc

यह करेक्टर (w-1) कॉलम खाली छोड़ कर w वें कॉलम में प्रिन्ट हो जाएगा।

1.12.4 इनपुट फलन **scanf(input function scanf)**

scanf( ) फलन के द्वारा आंकड़ों को मानक इनपुट युक्ति (Input device-keyboard) से लेकर प्रोग्राम के विभिन्न चरों में भंडारित (store) किया जाता है। scanf( ) फलन निम्न प्रकार से लिखा जाता है –

scanf("कन्ट्रोल स्ट्रिंग",&चर1,&चर2,&चर3)

scanf में &चर1, &चर2 तथा &चर3 उन चरों के पते (address) है जहाँ पर की–बोर्ड द्वारा प्राप्त आंकड़ों को भंडारित करना है। इसमें चर से पूर्व & का चिन्ह यह दर्शाता है कि यह उस चर का पता है न कि उसका मान। कन्ट्रोल स्ट्रिंग में % चिन्ह के साथ printf फलन की भाँति अंग्रेजी वर्णमाला का अक्षर लिखा जाता है। जैसा सारणी – 1.11 में दर्शाया गया है।

उदाहरण :

scanf("%d%d%d", &NUM1, &NUM2, &NUM3) ;

NUM1, NUM2 और NUM3 पूर्णांक प्रकार के चर है। सभी चरों के लिए अलग–अलग कन्ट्रोल स्ट्रिंग लिखना होता है।

प्रोग्राम 6 : तीन वास्तविक नम्बर पढ़कर उनमें से सबसे छोटा नम्बर ज्ञात करने हेतु 'C' भाषा में प्रोग्राम लिखिए।.

```
#include<stdio.h>
/* C program to read three float number and find smallest number.*/
main()
{
        float a,b,c,small;/*Declaration of float numbers*/
        printf("Enter three float Numbers :\n");
        scanf("%f %f %f",&a,&b,&c);/*read three float numbers*/
        if(a < b)
                if(a < c)
                        small = a;
                else
                        small = c;
         else
                if(b < c)
                        small = b;
                else
                  small = c;
        printf("\nThe Smallest Number = %f",small);
}
```

Input :Enter three float Numbers :

78.23 78.96 85.52

Output:The Smallest Numbers : 85.52

Input :Enter three float Numbers :

23.23 23.15 23.16

Output:The Smallest Numbers : 23.15

getchar( ) फलन

यह फलन इनपुट युक्ति से एक करेक्टर लेता है और कम्प्यूटर को देता है। इसका सामान्य स्वरूप है :

चर–नाम = getchar( );

उदाहरण

```
char c ;
. . . . . .
. . . . . .
c = getchar ( ) ;
```

उपरोक्त उदाहरण में पहली पंक्ति में c नामक चर char प्रकार का घोषित किया गया है। जैसे ही दूसरा कथन c=getchar( ) निष्पादित किया जाता है कम्प्यूटर की–बोर्ड से कुंजी के दबने की प्रतीक्षा करता है (यदि डाटा बफर में नहीं है तो) तथा कुंजी दबने पर उस संप्रतीक को c में भंडारित कर देता है। अतः c = getchar( ) तथा scanf("%c", &c) दोनो कथन समतुल्य है।

putchar() फलन

इस फलन के द्वारा कम्प्यूटर की मेमोरी से एक करेक्टर लेकर उसे मोनीटर पर दिखाता है। इसका सामान्य स्वरूप निम्न है :

putchar (संप्रतीक चर का नाम) ;

इसमें संप्रतीक चर पूर्व में घोषित चर है जिसका मान मोनीटर पर डिस्प्ले होगा।

उदाहरण :

```
char a;
. . . . .
. . . . .
putchar(a)
```

पहले कथन से a संप्रतीक प्रकार का चर घोषित होता है। दूसरा कथन a चर के मान को आउटपुट युक्ति पर भेज देता है। putchar में आवश्यक है कि a संप्रतीक प्रकार का चर है। अतः putchar(a) तथा printf ("%c",a) दोनों कथन समतुल्य हैं।

प्रोग्राम 7 : 'C' भाषा में लाइन पढ़कर उसको अपर केस में लिखने के लिए प्रोग्राम लिखिए।

```
#include<stdio.h>
/* C program to read a line and print it into upper case.*/
main()
{
    char c[80];/*Declaration of char array or string*/
    int i;
    printf("Enter A line:\n");
    for(i=0;(c[i]=getchar()) != '\n';i++);/*This statement read a line*/
    c[i]='\0'; /* Store NULL char at end*/
    printf("The UpperCase line is:\n");
    for(i=0;c[i]!='\0';i++) {
      c[i] = toupper(c[i]);/*Convert into the UpperCase Letter*/
      putchar(c[i]);  /*To print a Character on the screen*/
    } }
```

Input :Enter a line :

The string is a combination of character.

Output:The UpperCase line is:

THE STRING IS A COMBINATION OF CHARACTER.

1.13 कन्ट्रोल कथन (Control Statements)

'C' भाषा से कथनों का निष्पादन उसी क्रम में होता है जिस क्रम में कथन प्रोग्राम में लिखे गये हैं। प्रत्येक कथन केवल एक बार ही निष्पादित होता है। परन्तु कुछ प्रोग्रामों में एक तार्किक कन्डीशन से निर्णय होता है कि किन कथनों का निष्पादन होना है और किन कथनों का निष्पादन नहीं होना है। यह if तथा switch कथनों के द्वारा पूरा किया जा सकता है।

जब तक तार्किक कण्डीशन (Logical Condition) true रहती है जब तक कुछ कथन निष्पादित होते रहते हैं। इसको लूपिंग कहते हैं।

कन्ट्रोल कथन प्रोग्राम के कथनों को निश्चित क्रम में निष्पादित करती हैं। 'C' भाषा में निम्न प्रकार के कन्ट्रोल कथन होते है –

1. निर्णय सम्बन्धी कथन (Decision Making Statement)

   (a) if कथन (b) if else कथन (c) switch कथन

2. लूप वाले कथन

   (a) for लूप (b) while लूप (c) do-while लूप

3. अन्य कन्ट्रोल कथन.

   (a) break (b) continue (c) goto

1.14 if कथन (if statement)

यह एक शक्तिशाली कन्ट्रोल कथन है जिसमें पहले कन्डीशन को जाँचा जाता है उसके बाद परिणाम के आधार पर अन्य कथनों का निष्पादन किया जाता है। if कथन निम्न चार प्रकार के होते हैं।

(i) साधारण if कथन (simple if statement)

(ii) if-else कथन (if else statement)

(iii) नीड़ित if else कथन (Nested if else statement)

(iv) else if सोपान (else if stairs)

1.14.1 साधारण if कथन (Simple if statement)

इस कथन में सबसे पहले कन्डीशन को जाँचा जाता है। यदि वह True होती है तो यह कथन समूह को निष्पादित करेगा अन्यथा नहीं करेगा। कथन समूह में एक या एक से ज्यादा कथन हो सकते हैं। इसका सामान्य रूप निम्न है :

```
if (कन्डीशन)
{
कथन–समूह
}
```

यदि कथन समूह में एक ही कथन होने पर मंझला कोष्ठक ({}) लगाने की आवश्यकता नहीं होती है। जैसा नीचे उदाहरण में दर्शाया गया है।

```
(a) if (a > 10) printf("a is greater than 10");
(b) if (b == 5)
        printf("b is equal to 5");
(c) if (n%2 ==0)
        printf("N is a even number");
```

(d) if (a>10||b==5)

```
{
    printf("a is greater than10");
    printf("b is equal to 5");
}
```

उदाहरण (a) (b) और (c) में एक कथन होने के कारण मंझला कोष्ठक ({}) लगाने की आवश्यकता नहीं है। लेकिन उदाहरण (d) में दो कथन होने के कारण मंझला कोष्ठक लगाना जरूरी होता है।

1.14.2 if-else कथन (if-else statement)

इस कथन का नियंत्रण प्रवाह दो दिशाओं में नियंत्रण करता है। सबसे पहले कन्डीशन को जांचा जाता है। यदि कन्डीशन True है तो प्रथम कथन समूह निष्पादित होता है। False होने पर दूसरा कथन–समूह निष्पादित होता है। if-else कथन को निम्न प्रकार लिखा जाएगा –

```
if (कण्डीशन)
{
        कथन समूह 1
}
else
{
        कथन समूह 2
}
```

if-else कथनों के उदाहरण निम्न है :

(a)
```
if (a > b)
        printf("a is greater than b");
else
        printf("b is greater than a");
```

(b)
```
if (n%2==0)
        printf("N is Even Number") ;
else
        printf("N is odd Number") ;
```

(c)
```
if (Basic > 50000)
{
        HRA = BASIC * 0.15 ;
        DA = BAISC * 0.61 ;
}
else {
        HRA = BASIC * 0.10
        DA  = BASIC * 0.61 ;
}
```

उदाहरण (a) और (b) में एक कथन होने के कारण मंझला कोष्ठक ({}) लगाने की आवश्यकता नहीं है। लेकिन उदाहरण (c) में दो कथन होने के कारण मंझला कोष्ठक लगाना जरूरी होता है।

प्रोग्राम 8 : 'C' भाषा में एक प्रोग्राम लिखिये जो यह ज्ञात करे कि दिया गया वर्ष लीप वर्ष है या नहीं।

```
#include<stdio.h>
#include<conio.h>
/*It is C program to check for leap Year*/
main()
{
    int year,leap;
    printf("\nEnter the year :");
    scanf("%d",&year);
    if(year % 100 == 0) /* Checking for century*/
        if(year % 400 == 0)/* Century is leap year*/
            printf("It is century and leap year");
        else
            printf("It is century but not leap year");
    else if(year % 4 == 0) /* Ckecking for leap year*/
            printf("It is leap year");
         else
            printf("It is not leap year");
    getch();
}
```

Input/Output:

```
Enter the year :2000
It is century and leap year
Enter the year :2001
It is not leap year
Enter the year :2004
It is leap year
Enter the year :1900
It is century but not leap year
```

1.14.3 नीड़ित if-else कथन (Nested if else statement) :

जब if-else कथन में दूसरा if-else कथन लिखा जाता है तो उसे नीड़ित if-else (Nested if-else) कहते हैं।

```
if( कण्डीशन 1)
{
    if (कण्डीशन 2)
    {
```

```
            कथन समूह 1
        }
        else
        {
            कथन समूह 2
        }
        कथन समूह 3
    }
    else
    {
        कथन समूह 4
    }
```

उपरोक्त कन्ट्रॉल कथन में पहले कण्डीशन 1 जांची जाती है। यदि यह False है तो नियन्त्रण प्रवाह कथन समूह 4 को निष्पादित करना प्रारम्भ कर देता है। इससे पूर्व के सभी कथन समूहों को छोड़ देता है। यदि कण्डीशन 1 True है तो कण्डीशन 2 जांची जाती है। यदि कण्डिशन 2 true है तो कथन समूह 1 का निष्पादन करेंगा अन्यथा False होने पर कथन समूह 2 का निष्पादन करेंगा। तद्पश्चात् कथन समूह 3 को निष्पादित कर कंट्रोल प्रवाह if else कथन से बाहर आ जाएगा तथा कथन समूह 4 को निष्पादित नहीं करेगा।

उदाहरण :

```
if(x>=40)
{
    if(x>=65)
    {
        printf("FIRST DIVISION");
    }
    else
    {
        printf("SECOND DIVISION");
    }
}
else
{
    printf("FAIL");
}
```

1.14.4 else if सोपान (else if stairs)

इस प्रकार के if कथन बहुमार्गी कथनों के निष्कर्ष के काम आते हैं। इस कथन का प्रयोग तब किया जाता है जब एक–एक करके कण्डीशन लगानी होती है। इसको निम्न प्रकार लिखा जाता है।

```
if  (कण्डीशन 1)
{
        कथन समूह 1
}
else if (कण्डीशन 2)
{
        कथन समूह 2
}
else if (कण्डीशन 3)
{
        कथन समूह 3
}
else
{
        कथन समूह 4
}
```

1.14.5 switch कथन (switch statement)

यह एक बहुमार्गी कण्डीशनल नियन्त्रण कथन है। इसमें भी if-else की भाँति एक कण्डीशन होती है। switch कथन में व्यंजक या चर के मानों के अनुसार विभिन्न कथन समूह निष्पादित किये जाते हैं। लेकिन व्यंजक या चर का मान पूर्णांक या करैक्टर ही होना चाहिए। इसका स्वरूप निम्न प्रकार होता है।

```
switch (व्यंजक या चर)
{
        case मान 1 :    कथन समूह 1
                        break ;
        case मान 2 :    कथन समूह 2
                        break ;
        case मान n :    कथन समूह n
                        break ;
        default  :      कथन समूह
}
```

यहाँ पर switch, case, break तथा default की–वर्ड है। सर्वप्रथम switch कथन में व्यंजक या चर का मान जांचा जाता है। इस मान को case की–वर्ड के साथ लिखे अचर के साथ तुलना की जाती है। यदि यह मान किसी case के बराबर पाया जाता है तो उस case से सम्बन्धित कथन समूह को निष्पादित कर दिया जाता है और break; कथन मिलते ही वह switch कथन से बाहर आ जाता है अन्यथा यह इसके बाद सभी case कथनों को निष्पादित करता जाता है। यदि व्यंजक या चर का मान किसी case मान से नहीं मिलता तो

default वाले कथन समूह को निष्पादित करता है। यदि default कथन switch कथन में नहीं लिखा है तो व्यंजक या चर का मान किसी case मान से नहीं मिलने पर कोई भी कथन–समूह निष्पादित नहीं होता और नियन्त्रण प्रवाह switch कथन से बाहर आ जाता है।

उदाहरण :

```
switch (N) /* N is an integer value */
{
case 1 : printf("ONE") ;
         break ;
case 2 : printf("TWO") ;
         break ;
case 3 : printf("THREE") ;
         break ;
case 4 : printf("FOUR") ;
         break ;
case 5 : printf("FIVE") ;
         break ;
default : printf("Enter Between 1-5")
}
```

उपरोक्त उदाहरण में नम्बर शब्दों में प्रिन्ट हो जाएगे। N का मान अचर (1, 2, 3, 4 और 5) से तुलना करेगा। यदि यह किसी अचर के बराबर होता है तो वह उस अंक को शब्दों में प्रिन्ट कर देगा अन्यथा डिफाल्ट कथन को निष्पादित करेगा और कन्ट्रॉल प्रवाह switch कथन से बाहर आ जाएगा।

प्रोग्राम 9 : 'C' भाषा में एक प्रोग्राम लिखिये जो एक छात्र की ग्रेड बता सके। ग्रेड निम्न प्रकार होंगे।

100 - 90	A+
89 - 80	A
79 - 70	B+
69 - 60	B
56 - 50	C
50 - 0	Fail (switch कथन द्वारा)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
/* It is C program to Use switch Statement.*/
main()
{
    int marks,Rollno;
    char Name[20],grade[5];
    clrscr();
    printf("Enter The Name of a Student    : ");
```

```
    gets(Name); /*Library function to read a string*/
    printf("Enter The Roll No. of a Student : ");
    scanf("%d",&Rollno);
    printf("Enter The Marks Obtained       : ");
    scanf("%d",&marks);
    switch(marks/10) /* Interger Division*/
    {
        case 10:
        case  9: strcpy(grade,"A+");break;
        case  8: strcpy(grade,"A");break;
        case  7: strcpy(grade,"B+");break;
        case  6: strcpy(grade,"B");break;
        case  5: strcpy(grade,'C');break;
        default: strcpy(grade,"Fail");
    }
    printf("\nName of a Student        : %s",Name);
    printf("\nThe Roll No. of a Student : %d",Rollno);
    printf("\nThe Marks Obtained       : %d",marks);
    printf("\nThe Grade Obtained       : %s",grade);
getch();
}
```

Result of student as per the above program:

Input:
Enter The Name of a Student : Sunil Methi
Enter The Roll No. of a Student : 8542
Enter The Marks Obtained : 85
Output:
Name of a Student : Sunil Methi
The Roll No. of a Student : 8542
The Marks Obtained : 85
The Grade Obtained : A
Input:(for second student)
Enter The Name of a Student : Yashika Gupta
Enter The Roll No. of a Student : 8645
Enter The Marks Obtained : 97
Output:
Name of a Student : Yashika Gupta
The Roll No. of a Student : 8645

The Marks Obtained : 97

The Grade Obtained : A+

1.15 लूपिंग (Looping)

निर्णय सम्बन्धी कथनों के अतिरिक्त प्रोग्राम में कुछ कथनों को एक से अधिक बार निष्पादित करने की आवश्यकता होती है। कुछ कथनों का निष्पादन बार–बार उसी क्रम में दोहराया जाता है तब यह क्रिया लूपिंग (looping) कहलाती है। for, while तथा do-while तीन लूप कथन 'C' भाषा में होते हैं। इस सभी कथनों के दो भाग होते हैं –

(i) लूपिंग काया (looping body)

(ii) नियन्त्रण कथन (condition)

लूप कथन में नियन्त्रण कथन की जांच होती है और यदि यह True रहता है तो लूप निष्पादित होता है अन्यथा नियंत्रण प्रवाह लूप से बाहर आ जाएगा।

1.15.1 for लूप (for loop)

for कथन का सामान्य स्वरूप निम्न होता है :

```
for (exp1 ; exp2 ; exp3)
{
        लूप काया
}
```

जहाँ exp1 : यह लूप के काउण्टर चर की आरम्भिक मान बताते है और यह व्यंजक एक ही बार निष्पादित होता है।

exp2 : यह एक कण्डीशन व्यंजक होता है। अगर यह True होता है तो लूप निष्पादित होता रहता है।

exp3 : वृद्धि / ह्रास कथन से काउण्टर चर का मान बढ़ाने / घटाने के लिए होता है।

उदाहरण :

(a)
```
for (i = 0 ; i < 10 ; i++)
        printf("%d", i) ;
```

(b)
```
for (j = 1 ; j <= 20 ; j += 2)
{
        printf("%d\n",j) ;
        SUM = SUM + j ;
{
printf("SUM = %d", SUM);
```

(c)
```
for (j = 1, i = 10; j < i ; j ++, i - -)
        printf("%d %d", j, i);
```

प्रोग्राम 10 : 10 पूर्णांक संख्याओं को जोड़ने हेतु प्रोग्राम लिखिये।

```
#include<stdio.h>
#include<conio.h>
main()
{
```

```
    int N,sum=0,i;/*Sum initialize by zero at declaration*/
    for(i=1; i<=10;i++)
    {
        printf("Enter %d  Number :",i);
        scanf("%d",&N);
        sum += N; /*short hand assignment operator.*/
    }
    printf("\nSum of ten integer Number = %d",sum);
    getch();
}
```

Input/Output:

Enter 1 Number :23
Enter 2 Number :47
Enter 3 Number :98
Enter 4 Number :45
Enter 5 Number :85
Enter 6 Number :93
Enter 7 Number :123
Enter 8 Number :243
Enter 9 Number :24
Enter 10 Number :45
Sum of ten integer Number = 826

1.15.2 while लूप (while loop)

जब लूप का निष्पादन करने से पूर्व यह ज्ञात नहीं होता कि लूप को कितनी बार चलाना है तो यह एक कन्डीशन पर निर्भर होता है। जब तक यह कण्डीशन True रहती है तब तक लूप के कथन बार–बार निष्पादित होते रहते हैं। जैसे ही कन्डीशन False होती है लूप का निष्पादन रुक जाता है। इसको निम्न प्रकार लिखा जाता है :

```
while (कन्डीशन)
{
        लूप काया
}
```

उदाहरण :

(a)
```
while (i < 10) / * i का प्रारम्भिक मान 1 है * /
{
        printf("%d", i) ;
        i++ ;
}
```

```
(b)     while (j <= 20) / * initial value of j is one * /
        {
                printf("%d", j) ;
                SUM = SUM + j ;
                j += 2 ;
        }
        printf("SUM = %d", SUM) ;
```

प्रोग्राम 11 : एक प्रोग्राम लिखिए जो दिये गये नम्बर के अंकों का योग ज्ञात करें।

```
#include<stdio.h>
#include<conio.h>
/*It is C program to Add the digit of a given number*/
/*e.g. 12345 ,sum = 1 + 2 + 3 + 4 + 5 */
main()
{
    int m,N,sum=0;/* Sum initialize by zero at declaration*/
    printf("Enter a Number :");
    scanf("%d",&N);
    while(N != 0)
    {
        m = N % 10;
        sum = sum + m;
        N = N / 10;
    }
    printf("\nSum of digits = %d",sum);
    getch();
}
```

Input/Output:

Enter a Number :24563

Sum of digits = 20

1.15.3 do-while लूप (do-while loop)

do-while लूप भी while लूप कि तरह एक लूपिंग कथन है। while लूप में पहले कन्डीशन को जाँचा जाता है उसके बाद लूप काया को निष्पादित किया जाता है। जबकि do-while लूप में पहले लूप काया को निष्पादित किया जाता है उसके बाद कन्डीशन को जाँचा जाता है। अतः do-while लूप कम से कम एक बार तो निष्पादित होता है। do-while लूप भी while लूप की तरह ही True कन्डीशन पर निष्पादित होता है। इसका सामान्य रुप निम्न प्रकार होता है :

```
do
{
  लूप काया
}while (कन्डीशन) ;
```

उदाहरण :

```
(a)     do
        {
                printf("%d", i) ; / * /initial value of i is one */
                i++;
        }while (i<10) ;
(b)     do  / * initial value of j is one * /
        {
                printf("%d", j) ;
                SUM+=j ;
                j+=2 ;
        }       while (j<=20) ;
```

प्रोग्राम 12 : एक 'C' भाषा में प्रोग्राम लिखिये जो दिये गये नम्बर को उल्टा लिखे।

```
#include<stdio.h>
#include<conio.h>
/*It is C program to reverse a given number*/
/*e.g. 32512 print as 21523*/
main()
{
    int m,N,rev=0;
    printf("Enter a Number :");
    scanf("%d",&N);
    do
    {
        m = N % 10;
        rev = rev * 10 + m;
        N = N / 10;
    }while(N != 0);
    printf("\nReverse Number is = %d",rev);
    getch();
    return 0;
}
```

Input/Output of the above program:

Enter a Number :32512

Reverse Number is = 21523

1.15.4 नीड़ित लूप **(Nested Loop)**

जब एक लूप के अन्दर दूसरा लूप चले (loop in side loop) तो इसे नीड़ित लूप कहते हैं। जैसे एक for लूप के अन्दर दूसरा for लूप होना। इसमें ध्यान रखने योग्य बात यह है कि बाहर के लूप का समापन बाद में, अन्दर के लूप का समापन पहले होता है जैसा चित्र में दिखाया गया है–

```
for (    condition    )
{
        for (    condition    )
        {
        }
}
```

उदाहरण :–

(a)
```
for (i = 0 ; i < 10 ; i ++)
        for (j = i; j < 10 ; j ++)
                printf("%d", i) ;
```

printf कथन 55 पूर्णांक नम्बरों को प्रिन्ट करेगा।

(b)
```
for (i = 1 ; i < =10; i ++)
{       j = 1 ;
        while (j < = 10)
        {
                printf("%d", i*j) ;
                j++ ;
        }
        printf("\n") ;
}
```

उपरोक्त लूप 1 से 10 तक पहाड़े (Tables) प्रिन्ट करेंगे।

1.15.5 break कथन (break statement)

break कथन का उपयोग साधारणतया लूप में या switch कथन में किया जाता है। यह लूप को बीच में समाप्त करने के लिए काम आता है। अगर नीड़ित लूप में break कथन निष्पादित होता है तो वह उसी लूप से बाहर आएगा जिस लूप में break कथन निष्पादित हुआ है। इसको निम्न प्रकार से लिखते हैं :

उदाहरण :

(a)
```
for (i = 1 ; i < = 10 ; i + +)
{
        scanf("%d", &N) ;
        SUM = SUM + N ;
        if (SUM> = 200)
                break ;
}
```

उपरोक्त लूप दो तरह से समाप्त हो सकता है, यदि i का मान 10 हो जाए या SUM का मान 200 से अधिक हो जाए तो break कथन के निष्पादित होने पर भी लूप समाप्त हो जाएगा।

1.15.6 continue कथन (continue statement)

'C' भाषा में break तथा continue, दोनो कथन लूप के निष्पादन को बीच में रोकने के लिए उपयोग किये जाते हैं। परन्तु break कथन से लूप का निष्पादन रूक जाता है तथा नियन्त्रण प्रवाह लूप के

बाहर वाले कथन को निष्पादित करना आरम्भ कर देता है। जबकि continue कथन का निष्पादन होने पर लूप के मात्र उसी पुनरावर्तन (iteration) के निष्पादन को छोड़कर अगले पुनरावर्तन (iteration) का निष्पादन शुरू कर देता है।

इसको निम्न प्रकार लिखते हैं :

```
continue ;
```

उदाहरण :

```
do
{
        scanf("%d", &N) ;
        it (N < 0)
        {
                printf("Error") ;
                continue ;
        }
        / * other statements * /
} while (N <= 50) ;
```

उपरोक्त उदाहरण में, N के ऋणात्मक मान के लिए यह Error प्रिन्ट करेगा तथा continue कथन का निष्पादन होने के कारण यह लूप को शुरू से दुबारा (Next iteration) निष्पादित करेगा।

महत्वपूर्ण बिन्दु

1. कथनों को अलग करने के लिये अर्द्धविराम (;) लगाना आवश्यक है। यह कथन की समाप्ति का सूचक है। प्री–प्रोसेसर डायरेक्टिव के बाद अर्द्धविराम नहीं लगता।
2. C प्रोग्राम भाषा एक केस संवेदनशील भाषा है अर्थात उच्च केस (upper case) तथा निम्न केस (lower case) भिन्न–भिन्न होते है।
3. printf कथन संदेशों को निर्गत युक्ति (output devices) तक पहुँचाता है।
4. scanf भी एक फलन का आवाहन करता है जो इनपुट युक्ति (input devices) से इनपुट लेता है।
5. 'C' भाषा में सभी 32 की–शब्द छोटे अक्षरों (Lower Case Letter) में होते हैं।
6. 'C' भाषा संकारक धनी भाषा है। इसमें अंकगणितय, तुलनात्मक, तार्किक, बिटवाइज आदि संकारक होते हैं।
7. किसी व्यंजक को हल करने के लिए प्राथमिकता तथा साहचर्य नियम का उपयोग किया जाता है।
8. main() फलन भी यूजर डिफाइन फलन कहलाता है तथा यह प्रोग्राम में होना आवश्यक है।
9. scanf(), getchar(), getch(), getche() तथा gets() इनपुट फलन है।
10. printf(), putchar() तथा puts() आउटपुट फलन है।
11. 'C' भाषा में निर्णयात्मक कथन if, if-else तथा switch कथनों का उपयोग किया जाता है।
12. लूपिंग के द्वारा प्रोग्राम के किसी भाग को एक से ज्यादा बार निष्पादित कर सकते हैं। for, while तथा

do while लूप 'C' भाषा में उपयोग करते हैं।

13. break कथन, switch या लूप कथन का निष्पादन रोकने हेतु किया जाता है तथा कन्ट्रोल प्रवाह कथन से बाहर आ जाता है।

14. continue कथन के द्वारा लूप के किसी को निष्पादन होने से रोका जा सकता है।

अभ्यासार्थ प्रश्न

बहुचयनात्मक प्रश्न

1. if(1)

   printf("True");

   else printf("false");

   उपरोक्त कथन प्रिन्ट करेंगा–

   (अ) true (ब) false

   (स) कथन गलत है (द) उपरोक्त में से कोई नहीं.

2. 'C' भाषा की खोज किसने की ?

   (अ) केन थॉम्पसन (ब) डेनिस रिची

   (स) मार्टिन रिचर्डस (द) डॉनोवन

3. 'C' भाषा में प्रत्येक कथन को पृथक करने के काम में आता है?

   (अ) & (ब) ! (स) ; (द) "

4. 'C' भाषा के सभी प्रोग्राम में किसका लिखना अतिआवश्यक है–

   (अ) ग्लोबल चर (ब) main() फलन

   (स) पॉइन्टर (द) फलन

5. ऐसे शब्द जिनका उपयोग व अर्थ पहले से ही निश्चित होता है–

   (अ) चर (ब) अचर

   (स) अभिज्ञानक (द) आरक्षित शब्द

6. निम्न में से कौनसा आरक्षित शब्द नहीं है –

   (अ) auto (ब) while

   (c) switch (द) total

7. निम्न में से वैद्य अचर है –

   (अ) 4, 78 (ब) 0X23A

   (स) 177A (द) 5.7, 632

8. इसमें से कौनसा अभिज्ञानक वैद्य नहीं है –

   (अ) 5XYZ (ब) total_subject

   (स) Stud_mark (द) XSXY

9. long double डाटा प्रकार के लिए printf में कन्ट्रोल स्ट्रिंग होगी –

(अ) %d (ब) %lf (स) %ld (द) %Lf

10. % (प्रतिशत) संकारक का उद्देश्य है –

(अ) जोड़ना (ब) भाग देना

(स) शेषफल निकालना (द) गुणा करना

11. यदि i = 8 तथा b = - -i +3 तो b का मान होगा–

(अ) 11 (ब) 10 (स) 9 (द) 12

12. लूप का निष्पादन रोकने के लिए कौनसा कथन उपयोग में लाया जाता है –

(अ) break (ब) stop

(स) end (द) उपरोक्त में से कोई नहीं

13. for (i=0;i<=5;i++)

```
for(j=i;j<=5;j++)
    printf("India");
```

उपरोक्त प्रोग्राम में India कितनी बार प्रिन्ट होगा

(अ) 6 (ब) 21 (स) 36 (द) 30

14. case की–वर्ड किस कथन में उपयोग होता है ?

(अ) switch (ब) for

(स) do while (द) while

15. नीड़ित लूप संरचना में सबसे पहले समाप्त होने वाला लूप होगा –

(अ) बाहर वाला लूप (ब) अन्दर वाला लूप

(स) दोनों एक साथ (द) उपरोक्त में से कोई नहीं

अतिलघूत्तरात्मक प्रश्न

1. आरक्षित शब्द क्या होते हैं ?
2. प्राचल कितने प्रकार के होते हैं ?
3. 'C' भाषा में main( ) फलन का उपयोग कितनी बार किया जाता है।
4. ++ संकारक का क्या उपयोग है ?
5. getchar() फलन के द्वारा क्या करते हैं ?
6. आउटपुट किस फलन के द्वारा दिया जा सकता है ?
7. default कथन किस संरचना में उपयोग किया जाता है ?
8. नीड़ित if else कथन कैसे लिखते हैं ?
9. continue कथन का उपयोग कहाँ किया जाता है ?
10. 'C' भाषा में कितनी लूप संरचनाए है ?
11. do while संरचना को उदाहरण सहित समझाइये ?

लघूत्तरात्मक प्रश्न

1. एल्गोरिथम से क्या तात्पर्य है ?
2. अचर किसे कहते है ?
3. चर की घोषणा किस प्रकार होती है ?
4. स्ट्रिंग चर और करैक्टर चर में अन्तर बतलाइये।
5. अभिज्ञानक किसे कहते हैं ?
6. त्रिआधारी ऑपरेटर को समझाइयें।
7. संकारकों की प्रायिकता (Precedence) क्या होती है ?समझाइये।
8. संकारकों के लिए साहचर्य नियमों की आवश्यकता कब होती है।
9. 'C' भाषा को मिडिल लेवल भाषा क्यों कहा जाता है ?
10. नीडित लूप किसे कहते है ?
11. do-while कथन को समझाइये।
12. if कथन के सभी प्रकारों को लिखिये।
13. switch कथन को किस प्रकार लिखते हैं, फारमेट लिखिये ?
14. continue तथा break में अन्तर स्पष्ट कीजिए ?
15. नीचे लिखे प्रोग्राम का आउटपुट बताइये–

(a)
```
#include<stdio.h>
        main ()
        {
                int i = 1, x = 1 ;
                for (i = 1 ; i < 10, i++)
                {
                        printf("%d\n", x+i);
                        x = x + 1;
                }
        }
```

(b)
```
# include <stdio.h>
# define p 10
        {
                int k =1, w = p;
                while (k <=w)
                {
                        printf("%d\n",k);
                }
        }
```

निबन्धात्मक प्रश्न

1. 'C' भाषा के प्रोग्राम की संरचना बताइये।
2. प्रोग्राम संरचना से क्या तात्पर्य है ?
3. डेटा टाईप कितने प्रकार के होते हैं ?समझाइये।
4. 'C' भाषा में काम आने वाले इनपुट/आउटपुट कथनों को समझाइये।
5. संकारकों की प्रायिकता तथा साहचार्य नियम से आप क्या समझते हैं। उदाहरण सहित समझाइये।
6. सभी इनपुट फलनों को उदाहरण सहित समझाइये ?
7. आउट फलनों के द्वारा किस प्रकार आउटपुट प्राप्त किया जाता है ?उदाहरण सहित समझाइये।
8. निम्न व्यजकों को 'C' के समतुल्य व्यंजकों में बदलिये।

   (i) $a + \frac{b}{c} + d$

   (ii) $A = \pi r^2$

   (iii) $x = \frac{-b+\sqrt{b^2-4ac}}{2a}$

   (iv) $c = a^2 + b^2$

9. तीन संख्याओं को पढ़कर उसमें से सबसे बड़ी संख्या ज्ञात करने के लिए कन्डीशनल संकारक का उपयोग करके प्रोग्राम लिखो।
10. लूपिंग के लिए कौन–कौन से कथन उपयोग किये जाते हैं?वर्णन कीजिए।
11. 1 से 50 तक की सभी विषम संख्याओं को प्रिन्ट करने हेतु 'C'भाषा में प्रोग्राम लिखिये।
12. while लूप और do-while लूप में अन्तर उदाहरण सहित समझाइये।
13. निम्नलिखित श्रेणी का योग ज्ञात करने के लिए प्रोग्राम लिखिये।

$$1 + x + x^2 + x^3 + ...x^n$$

उत्तरमाला

1.	अ	2.	ब	3.	स	4.	ब	5.	द
6.	द	7.	ब	8.	अ	9.	द	10.	स
11.	ब	12.	अ	13.	ब	14.	अ	15.	ब

पाठ–2

'सी' भाषा की प्रोग्रामिंग
(Programming in 'C' Language)

2.1 एरे (Array)

अगर हमें 10 छात्रों के अंको की सूची बनानी हो तो इसके लिए हमें 10 चर लेने होंगे। अगर छात्रों की संख्या बढ़ती जाएगी तो चर की संख्या बढ़ती जाएगी और यह प्रोग्राम जटिल होता जाएगा। 'C' भाषा में इस प्रकार की समस्या के हल हेतु एरे (Array) का उपयोग किया जाता है।

एरे एक चर है जो एक साथ एक ही प्रकार के आंकड़ों को एकत्रित कर सकता है। अभी तक हमने पढ़ा कि एक समय में चर का एक ही मान होता है। किन्तु कई बार ऐसी आवश्यकता होती है जब हम एक प्रकार के आपस में सम्बन्धित मानों का उपयोग करते हैं। ऐसे में यह कठिन है कि सभी के मान अलग–अलग चर में स्टोर (store) करें। अतः एक ही चर के द्वारा इन सभी को दर्शाया जा सकता है। इसमें समूह के प्रत्येक घटक को उसके सूचकांक (Index) द्वारा उल्लेख करते हैं। आंकड़ों का प्रकार char, int, float, double etc. हो सकता है। यदि किसी कक्षा में 50 छात्र हो तो उनके अंकों को स्टोर (store) करने के लिए marks नामक एरे का उपयोग कर सकते हैं जिसे marks[50] के नाम से जानेंगे। इसमें कोष्ठक [ ] में लिखा नम्बर 50 बताता है कि चर marks कुल 50 आंकड़ों का समूह है। प्रत्येक घटक उसकी स्थिति या सूचकांक के द्वारा पहचाना जाता है। जैसे marks [8] का अर्थ है marks नामक एरे में नवें स्थान पर स्टोर अंक।

उदाहरण :

दस अवयवों की एक एरे P[10] के दस अवयव निम्न होंगे।

P[0], P[1], P[2], . . . P[8], P[9]

2.1.1 एरे की घोषणा (Array Declaration)

एरे को भी अन्य चरों के समान एरे के उपयोग से पूर्व घोषित करना आवश्यक है। एरे की घोषणा में तीन चीजों की घोषणा की जाती है :

(i) एरे का प्रकार

(ii) एरे का नाम

(iii) एरे की क्षमता (कुल अवयवों की संख्या)

जब एरे घोषित किया जाता है तो एरे के साइज के बराबर स्थान मैमोरी में आवंटित कर दिया जाता है। एरे की घोषणा निम्न प्रकार करते हैं :

<एरे का प्रकार> <एरे का नाम><[साइज]>

उदाहरण :

int marks[50];

यहाँ int marks[50] बताता है कि marks नामक एरे का आकार 50 है तथा यह पूर्णांक को भण्डारित कर सकता है।

```
float salary[100];
double total[20];
char emp_name[20];
```

2.1.2 एरे पर प्रक्रिया **(Processing on Array)**

एरे पर संक्रियाऐं एक–एक अवयव पर संक्रिया करके की जाती है। ऐसा करने के लिए लूप की आवश्यकता होती है। एक बार लूप चलाने पर एरे के प्रत्येक अवयव पर संक्रिया होती है।

उदाहरण :

```
for ( i = 0; i<10; i++)
        scanf("%d",&A[i]) ;
```

उपरोक्त कथनों से एरे के 10 मानों का इनपुट लिया जाएगा। यह नीचे लिखे प्रोग्राम से स्पष्ट हो जाएगा।

प्रोग्राम 1: 'C' भाषा में 10 नम्बरों को व्यवस्थित करने हेतु प्रोग्राम लिखिए।.

```
#include<stdio.h>
#include<conio.h>
/*It is C program to sort 10 numbers using selection sort. */
int main()
{
    int A[10];
    int i,j,temp;
    printf("\nEnter Ten Integer Values :\n");
    for(i=0;i<10;i++)
        scanf("%d",&A[i]));
    /* Selection Sorting Algorithm */
    for(i=0;i<9;i++)
        for(j=i+1;j<10;j++)
            if(A[i]) > A[j])
            {
                temp = A[i];
                A[i] = A[j];
                A[j] = temp;
            }
    printf("\nThe Sorted Data are :\n");
    for(i = 0; i<10;i++)
        printf("%d  ", A[i]);
    getch();
    return (0);
}
```

Result:

Enter Ten Values :

5 19 45 63 7 4 78 12 89 56

The Sorted Data are :

4 5 7 12 19 45 56 63 78 89

2.1.3 एरे के प्रकार **(Type of Array)**

मुख्यतः एरे दो प्रकार के होते हैं :

1. एक विमीय एरे (One Dimensional Array)
2. बहु विमीय एरे (Multi Dimensional Array)

1. एक विमीय एरे **(One Dimensional Array)**

एक विमीय एरे में एक रो (Row) या एक कॉलम (column) होती है। उदाहरणार्थ– एक कक्षा में 50 छात्र हैं और उनके अंकों को भण्डारित करने के लिए एक विमीय एरे का उपयोग करते हैं। यदि एरे का नाम Marks हो तो :

```
Marks [ 0 ] =   77
Marks [ 1 ] =   55
Marks [ 2 ] =   67
Marks [ 3 ] =   65
Marks [ 4 ] =   89
        :
        :
        :
Marks [ 47 ] =  78
Marks [ 48 ] =  48
Marks [ 49 ] =  52
```

2. बहुविमीय एरे **(Multi Dimensional Array)**

कई बार हमें ऐसी आवश्यकता होती है जब एक विमीय एरे में आंकड़ों का भंडारण नहीं हो सकता है। जैसे यदि किसी कक्षा में 20 छात्र हो तथा प्रत्येक विद्यार्थी के पाँचों विषयों के अंकों का भण्डारण करना हो तो हमें सारणी (जिसमें 20 Row तथा 5 column) की आवश्यकता होती है। बहुविमीय एरे ऐसी सारणी के समान होती है जिसमें दो या अधिक विमायें होती है।

इसको लिखने का प्रकार निम्न है :

<एरे का प्रकार><एरे का नाम>[अधिकतम सीमा1][अधिकतम सीमा 2]..... .....[अधिकतम सीमा n]

यदि दो विमीय एरे (Two Dimensional Array) है तो नाम के साथ दो कोष्ठक ([ ][ ]) आयेंगे।

उदाहरण :

A[5][5] इसमें A[0][0] पहला अवयव है A[0][1] दूसरा अवयव, इसी प्रकार A[4][4] अन्तिम अवयव होगा। इसमें 5 Row तथा 5 column है। इसको चित्र 2.1 में दिखाया गया है।

	0	1	2	3	4
0	9	6	16	24	61
1	16	72	95	97	55
2	49	57	40	45	25
3	29	64	5	18	2
4	10	12	98	59	26

चित्र : **2.1 A[5][5]** का चित्रण

2.1.4 बहुविमीय एरे का प्रारम्भिकरण **(Initialization of Multi Dimensional Array)**

बहुविमीय एरे का प्रारम्भिकरण भी एक विमीय एरे की तरह होता है।

उदाहरण :

int MAT[3][3]={0,1,2,3,4,5,6,7,8,};

उपरोक्त प्रारम्भिकरण में पंक्ति और कॉलम को अलग–अलग प्रदर्शित नहीं किया गया है। उपरोक्त एरे को निम्न प्रकार पंक्ति और कॉलम में भी प्रदर्शित कर सकते है।

int MAT[3][3]={ {0,1,2,} {3,4,5,} {6,7,8,} };

पहली पंक्ति के मान पहले कोष्ठक में होंगे। दूसरी व तीसरी पंक्ति के मान क्रमशः दूसरे व तीसरे कोष्टक में होंगे। अगर कोष्ठक में दिए गए कुल मान भंडारित क्षमता से कम हो तो बाकी मान अपने आप शून्य हो जाएँगे।

उदाहरण :

int MAT[3][3]={ {1,2,3,} {4,5,6,} };

इस एरे की कुल क्षमता 9 है तथा कोष्ठक में कुल 6 मान है अतः बाकि तीन मान शून्य होंगे।

एरे के मान निम्न होंगे।

MAT[0][0]=1	MAT[0][1]=2	MAT[0][2]=3
MAT[1][0]=4	MAT[1][1]=5	MAT[1][2]=6
MAT[2][0]=0	MAT[2][0]=0	MAT[2][2]=0

प्रोग्राम 2 : दो 3 x 3 मैटरिक्स को गुणा करने हेतु 'C' भाषा का प्रोग्राम लिखिए।

```
#include<stdio.h>
#include<conio.h>
main()
{
    /*It is C program to Multiply two 3 x 3 Matrix*/
```

```
        /*All elements of Array C are initialized by Zero*/
        int i,j,k,A[3][3],B[3][3],C[3][3]={0};
        /* Read Matrix A & B*/
        printf("\nEnter Matrix A :\n");
        for(i=0; i<3; i++)
                for(j=0; j<3; j++)
                        scanf("%d",&A[i][j]);
                printf("Enter Matrix B :\n");
        for(i=0; i<3; i++)
                for(j=0; j<3; j++)
                        scanf("%d",&B[i][j]);
        /* Multiply Two Matrix A And B*/
        for(i=0; i<3; i++)
                for(j=0; j<3; j++)
                        for(k=0; k<3; k++)
                                C[i][j] += A[i][k] * B[k][j];
                printf("\nMultiply Matrix C :\n");
        for(i=0; i<3; i++)
        {
                for(j=0; j<3; j++)
                        printf("%d " ,C[i][j]);
                printf("\n");
        }
        getch();
}
```

Result :

```
Enter Matrix A :
1 2 3
3 2 1
6 8 4
Enter Matrix A :
1 4 5
6 2 1
5 3 8
Multiply Matrix C :
28  17  31
20  19  25
```

74 52 70

प्रोग्राम 3 : 10 नम्बरों को व्यवस्थित करने हेतु 'C' भाषा में प्रोग्राम लिखिए।

```
#include<stdio.h>
#include<conio.h>
/*It is C++ program to sort 10 numbers using Bubble sort.*/
int main()
{
        /*Array A declare with initialization.*/
        int a[10]={5,4,89,12,78,6,19,45,63,7};
        int i,j,temp;
        /* Bubble Sorting Algorithm*/
        for(i=0;i<9;i++)
                for(j=0;j<9-i;j++)
                        if(a[j] > a[j+1])
                        {
                                        temp = a[j];
                                        a[j] = a[j+1];
                                        a[j+1] = temp;
                        }
        printf("\nThe Sorted Data are :\n");
        for(i = 0; i<10;i++)
                printf("%4d" ,a[i]);
        getch();
}
```

Result:

The Sorted Data are :

4 5 6 7 12 19 45 63 78 89

2.2 स्ट्रिंग (String)

स्ट्रिंग संप्रतीकों (characters) का समूह है। 'C' भाषा में स्ट्रिंग को एरे के संप्रतीकों (Array of Char) द्वारा घोषित करते हैं। संप्रतीकों के समूह को डबल कोटेशन (" ") में बन्द किया हो तो वह अचर स्ट्रिंग होती है।

उदाहरण :

"C language is better than other Computer languages." /* It is string constant*/

2.2.1 स्ट्रिंग चर की घोषणा तथा प्रारम्भिकरण (Declaration and initialization of string variable)

एक स्ट्रिंग को संप्रतीक के एरे के (Array of char) रूप में घोषित करते हैं। स्ट्रिंग को निम्न प्रकार घोषित करते हैं।

char<string name>[size]

इसमें size यह बताता है कि इस स्ट्रिंग में कितने संप्रतिक होंगे। उदाहरण :

```
char name[20];
char address[30];
```

उपरोक्त उदाहरणों में 20 और 30 संप्रतीक आ सकते हैं। लेकिन वास्तव में इनमें 19 और 29 संप्रतीक ही स्टोर कर सकते हैं क्योंकि स्ट्रिंग का अन्तिम संप्रतीक हमेशा NULL char ('\0') होना चाहिए। NULL char स्ट्रिंग की समाप्ति का द्योतक होता है।

यहाँ पर यह ध्यान रखना जरूरी है कि अगर हम पूरी स्ट्रिंग की संक्रिया एक साथ कर रहे हैं तो NULL char अपने आप स्ट्रिंग के अंत में जुड़ जाएगा। अगर हम स्ट्रिंग की संक्रिया char by char कर रहे है तो प्रोग्रामर को यह ध्यान रखना अति आवश्यक है कि वह NULL char को संप्रतीक के रूप में स्टोर करें। क्योंकि इस केस में कम्पाइलर NULL char को स्टोर नहीं करता है। स्ट्रिंग का प्रारम्भिकरण निम्न प्रकार किया जा सकता है।

```
name[20]="RAMSHANKAER";
name[20]={'R','A','M','S','H','A','N','K','E','R','\0'};
```

उपरोक्त प्रारम्भिकरण में भी हमको यही ध्यान रखना है कि अगर सम्पूर्ण स्ट्रिंग एक अचर की भांति प्रारम्भिकरण में उपयोग आती है तो NULL char की आवश्यकता नहीं होगी। अन्यथा उसमें NULL char अंत में स्टोर करना होगा। जब हम प्रारम्भिकरण करते है तो स्ट्रिंग की साईज को लिखना जरूरी नहीं होता है।

उदाहरण :

```
name[]="RAM SHANKER";
```

उपरोक्त उदाहरण में कोष्ठक में साईज नहीं लिखी है। यह स्ट्रिंग अचर की साईज के अनुसार बना देता है। उपरोक्त उदाहरण में स्ट्रिंग की साईज 12 होगी।

2.2.2 स्ट्रिंग का इनपुट **(Input a String)**

स्ट्रिंग को पढ़नें (read) या इनपुट करवाने के कई तरीके हैं। scanf() फलन के द्वारा स्ट्रिंग को पढा जा सकता है।

उदाहरण :

```
char name[20];
scanf("%s", name);
```

scanf() फलन का उपयोग करते समय स्ट्रिंग के नाम के साथ एमपरसेन्ड (& ampersand) ऑपरेटर का उपयोग नहीं करते हैं क्योंकि स्ट्रिंग के नाम ही स्ट्रिंग के पते (Address) को प्रदर्शित करती है। उपरोक्त तरीके से हम स्ट्रिंग को खाली स्थान (Blank Space) आने तक ही पढ़ (read) सकते हैं। अगर उपरोक्त scanf() फलन के इनपुट Ram Shanker देते है। यह Ram को ही स्ट्रिंग में स्टोर करेगा। अगर हमें इस समस्या का समाधान करना हो तो हम स्ट्रिंग के लिए निंयत्रक स्ट्रिंग को बदलेंगे।

उदाहरण :

```
scanf("%[^\n]", Name);
```

यह स्ट्रिंग को नयी लाइन संप्रीतक आने से पहले तक पढ़ेगा (read) अगर हम इनपुट देते हैं और उसके बाद एन्टर की (Enter Key) दबाते है तो स्ट्रिंग में Ram Shanker स्ट्रिंग स्टोर हो जाएगी। हम स्ट्रिंग को char by char भी read कर सकते हैं।

उदाहरण :

```
int i = - 1;
```

```
char name[20], ch;
while(ch!='\n')
{
        ch=getchr();
        name[i]=ch;
}
name[i]='\0' /*store NULL Char at end.*/
```

उपरोक्त कोड को for लूप की सहायता से निम्न प्रकार लिख सकते हैं।

```
for(i=0; (name[i] = getchar())!='\n'; i++);
        name[i]='\0';
```

उपरोक्त तरीको से भी स्ट्रिंग नयी लाइन तक इनपुट करा सकते हैं। स्ट्रिंग इनपुट करने के लिए एक लाइब्रेरी फलन (gets()) भी होता हैं वह भी स्ट्रिंग को नयी लाइन तक ही पढ़ता है।

उदाहरण :

```
gets(name);
```

2.2.3 स्ट्रिंग प्रिन्ट हेतु **(Print a string)**

printf() तथा puts() फलनों की सहायता से स्ट्रिंग को प्रिंट कर सकते हैं।

उदाहरण :

```
printf("%s", name);
        या
gets(name);
```

उपरोक्त दोनों तरीकों से स्ट्रिंग NULL Char तक प्रिन्ट होगी।

2.2.4 स्ट्रिंग के लिए लाइब्रेरी फलन **(Library Function for string) :**

C भाषा में स्ट्रिंग के लिए बहुत सारे फलन होते है उनमें कुछ महत्वपूर्ण फलन सारणी 2.1 में दर्शाये गये है।

सारणी 2.1 : स्ट्रिंग फलन (string.h)

	फलन	कार्य
1.	strcat( )	दो स्ट्रिंगों को जोड़ने के लिए
2.	strcmp( )	दो स्ट्रिंगों की तुलना करने के लिए
3.	strcpy( )	एक स्ट्रिंग से दूसरी स्ट्रिंग में कॉपी करने के लिए
4.	strlen( )	एक स्ट्रिंग की लम्बाई ज्ञात करने के लिए

1. strcat( ) फलन

इसको निम्न प्रकार लिखते हैं :

```
strcat(string1, string2);
```

string1 और string2 दोनों character array है। जब फलन strcat() निष्पादित होगा तो string1 के अन्त में string2 में जुड़ जाएगी।

उदाहरण :

```
char st1[10]="Computer";
char st2[10]=" System";
strcat (st1, st2);
```

निष्पादन के बाद स्ट्रिंग होगी–

```
st1="Computer System";
st2="System";
```

इसमें st1 के अन्त में st2 जुड़ गई है। इसमें यह ध्यान रखना होगा कि पहला argument चर होना चाहिए।

2. strcmp( ) फलन

इसको निम्न प्रकार लिखते है :

```
n = strcmp(string1, string2);
```

यह string1 और string2 की तुलना करेगा तथा निम्न परिणाम देता है।

string1 == string2	n का मान शून्य होगा।
string1 < string2	n का मान ऋणात्मक होगा।
string1 > string2	n का मान धनात्मक होगा।

उदाहरण :

```
strcmp ("this","that")
```

उपरोक्त उदाहरण में this बड़ी स्ट्रिंग है क्योंकि स्ट्रिंग के प्रथम दो संप्रतीक (char) समान है तथा तीसरा संप्रतीक i, a संप्रतीक से बड़ा (ASCII value of i is 105 and a is 97) है । जिसका ASCII मान बड़ा होता है, वह बड़ी स्ट्रिंग कहलायेगी। तथा यह ASCII Value में अन्तर को वापस करता है। उपरोक्त उदाहरण में 8(105-97=8) को वापस करेगा।

उदाहरण :

```
strcmp("Kapil","Anish");        return –7
strcmp("Mohan","Mohan");        return zero
strcmp("Anish","Anil");         return +7
```

तुलना करने के लिए strcmpi() फलन भी होता है जो स्ट्रिंग के केस (upper and lower) को अनदेखा (ignore) करता है अर्थात दोनों को समान मानता है। जबकि strcmp( ) फलन दोनों केसों में अन्तर रखता है।

3. strcpy ( ) फलन

इसको निम्न प्रकार लिखते हैं :

```
strcpy (string1, string2);
```

इसमें string1चर तथा string2 चर या अचर हो सकते हैं। इसमें string2 के संप्रतीक string1 में कॉपी हो जाएँगें।

उदाहरण :

```
strcpy (State, "Rajasthan");
```

यह अचर स्ट्रिंग "Rajasthan", State चर में कॉपी कर देगा।

4. strlen ( ) फलन

इसको निम्न प्रकार लिखते है :

n = strlen (string);

यह string की लम्बाई (total character) को n में स्टोर कर देगा। String चर या अचर हो सकता है।

उदाहरण :

n = strlen ("Rajasthan");

n का मान 10 होगा। इसमें NULL Char ('\0') को भी जोड़ा जाता है।

2.2.5 दो विमीय स्ट्रिंग एरे (Two Demensional String Array)

अगर हमें कई स्ट्रिंग एक साथ स्टोर करनी होती है तो हम दो विमीय स्ट्रिंग एरे का उपयोग करते है। इसकी घोषणा निम्न प्रकार करते हैं।

```
char name [10][20];
```

उपरोक्त एरे 10 स्ट्रिंग तथा प्रत्येक स्ट्रिंग की लम्बाई 20 तक हो सकती है।

इस स्ट्रिंग को निम्न प्रकार पढ़ा (read) जा सकता है।

```
for (i= 0; i < 10; i++)
    scanf ("%s", name[i]);
/* or gets (name[i]);*/
```

और निम्न प्रकार प्रिन्ट कर सकते हैं।

```
for (i= 0; i < 10; i++)
printf ("%s", name[i]);
/* puts (name[i]);*/
```

प्रोग्राम 4 : कोई स्ट्रिंग पेलिनड्रोम (Palindrome) है या नहीं ज्ञात करने हेतु प्रोग्राम लिखिए।

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
    int m,n,flag=1;
    char str[80];
    clrscr();
    printf("\nEnter a String :");
    gets(str);
    m=0;
    n= strlen(str) - 1;
    while(m <= n)
    {
```

```
                    if(str[m] != str[n])
                    {
                            flag = 0;
                            break;
                    }
                    m++;
                    n - -;
        }
        if(flag==1)
                    printf("\nString is Palindrome");
        else
                    printf("\nString is not Palindrome");
        getch();
        return (0);
}
```

2.3 फलन **(Function)**

एक बड़ा प्रोग्राम बनाना कठिन होता है। अतः 'C' भाषा के प्रोग्राम को छोटे–छोटे भागों में विभक्त कर दिया जाता है। यह छोटे–छोटे भाग किसी एक संक्रिया को पूर्ण करते हैं। इन सभी भागों को क्रमबद्ध तरीके से इस प्रकार जोड़ा जाता है कि जटिल समस्या हल हो जाए।

सभी उच्च स्तरीय भाषाओं में यह प्रावधान है कि उस कथन खण्ड को अलग नाम देकर स्वतन्त्र रूप से लिखा जा सके तथा जब भी और जितनी भी बार उस कथन खण्ड की आवश्यकता हो उसका आवाहन् कर लिया जाता है। इस स्वतन्त्र खण्ड (प्रोग्राम) को उप प्रोग्राम या फलन कहते हैं।

प्रत्येक समस्या के लिए अलग–अलग प्रोग्राम खण्ड बनाए जाते हैं। इन खण्डों को ही फलन कहते हैं। इसकी परिभाषा निम्न प्रकार लिख सकते हैं। "A function is a self contained program segment that is used for some specific well defined task."

फलन का C भाषा में महत्वपूर्ण स्थान है। main(), scanf(), printf() फलनों को हम प्रोग्राम में उपयोग कर चुके हैं। 'C' भाषा के फलनों को दो वर्गो में विभक्त किया जा सकता है।

2.3.1 **main()** फलन

सभी 'C' भाषा के प्रोग्रामों का निष्पादन main() फलन से शुरू होता है। main() फलन की परिभाषा इस तरह की जाती है।

```
main()
{
    कथन
}
```

यह फलन 'C' भाषा में सही है। इसमें main() फलन कोई मान वापस नहीं करता है लेकिन नये कम्पाइलरों में main() फलन int मान को वापस करता है। अगर हमको मान वापस नहीं करना है तो main() से पहले void Key-word को उपयोग करना होगा।

उदाहरण :

```
void  main()
{
    कथन
}
```

main() फलन को प्रोटोटाइप निम्न प्रोटोटाइप में से किसी से मिलना जरूरी होता है।

```
int main()
void main()
int main (int argc, char * argv[])
void main (int argc, char * argv[] )
```

अगर main() फलन के साथ int या void कीवर्ड उपयोग नहीं करेगें तो यह int मान ही वापस भेजेगा। इसी तरह अगर किसी फलन के साथ वापस करने का प्रकार नहीं लिखा होता तो वह भी int प्रकार का मान वापस करेगा।

3.3.2 फलन के लाभ **(Advantages of Function)**

1. फलन के द्वारा प्रोग्राम को छोटे–छोटे भागों में विभक्त कर दिया जाता है जिससे बड़े प्रोग्राम को आसानी से लिख सकते हैं।
2. फलन लिखने पर प्रोग्राम को आसानी से पढ़ और समझ सकते हैं।
3. फलन के द्वारा हम प्रोग्राम में आसानी से कोई नया फलन भी जोड़ सकते हैं। जिससे प्रोग्राम में आसानी से बदलाव कर सकते हैं।
4. फलन के द्वारा जटिल समस्याओ को आसानी से हल कर सकते हैं।
5. फलन के द्वारा प्रोग्राम में त्रुटि को आसानी से ढूंढ़ सकते हैं।
6. फलन में दोहराने (Repeat) वाले कथन एक साथ लिख दिये जाते हैं जिससे उन्हें मुख्य प्रोग्राम में बार–बार लिखने की आवश्यकता नहीं होती है। इससे प्रोग्राम का आकार छोटा हो जाता है।
7. इनके प्रयोग से जटिल प्रोग्रामों को समझना आसान हो जाता है।
8. 'C' भाषा में फलन एक उप प्रोग्राम है जो किसी विशेष कार्य को करने के लिए बनाया जाता है।

'C' भाषा में एक प्रोग्राम में एक या एक से अधिक फलन होते हैं। प्रत्येक प्रोग्राम में main( ) फलन का एक और केवल एक बार आना आवश्यक है। इस प्रकार का फलन उपयोगकर्ता को कोई भी मान वापस नहीं देता है। इसके अतिरिक्त अन्य कई फलन भी 'C' भाषा में पहले से उपलब्ध है, जैसे scanf(), pow() आदि। कोई भी फलन एक बार में केवल एक मान ही वापस करता है। किसी प्रयोक्ता परिभाषित फलन का उपयोग करने के लिए उसे मुख्य फलन में call करते हैं। जब कोई फलन मुख्य प्रोग्राम में call होता है, तब नियन्त्रण वर्तमान निर्देश छोड़कर, जहाँ फलन लिखा गया है वहां चला जाता है। तत्पश्चात् पूरे फलन को निष्पादित करने के बाद पुनः उस निर्देश, जहाँ उसने नियंत्रण छोड़ा था, के बाद वाले निर्देश पर आता है। हम मुख्य फलन में एक ही फलन को कई बार call कर सकते हैं।

2.3.3 फलनों के प्रकार **(Types of Functions)**

फलनों को मुख्य रूप से दो भागों में बाँटा जा सकता है :

1. लाइब्रेरी फलन (Library Functions)
2. प्रयोक्ता परिभाषित फलन (User Defined Functions)

1. लाइब्रेरी फलन **(Library Functions)**

जो फलन भाषा में पहले से ही परिभाषित होते हैं, लाइब्रेरी फलन कहलाते हैं। इन्हें सीधा ही प्रोग्राम में उपयोग ले सकते हैं। लगभग प्रत्येक उच्च स्तरीय भाषा में इस प्रकार की सुविधा उपलब्ध है। सामान्य रूप

से प्रयोग में आने वाले कई फलनों का संग्रह लाइब्रेरी के रूप में बहुत उपयोगी रहता है। इससे हमें कई बड़े प्रोग्राम बनाने में आसानी रहती है। सारणी 2.2 में लाइब्रेरी फलन दर्शाये गए हैं।

सारणी 2.2 : लाइब्रेरी फलन

S. No.	Name of Function	Purpose of Function	Header file name
1.	getchar()	returns the next character typed on the keyboard.	stdio.h
2.	putchar()	outputs a single character to the screen.	stdio.h
3.	printf()	outputs a single character to the screen.	stdio.h
4.	scanf()	returns the next character typed on the keyboard.	stdio.h
5.	isdigit()	returns non-0 if arg is digit 0 to 9	ctype.h
6.	isalpha()	returns non-0 if arg is a letter of the alphabet	ctype.h
7.	isalnum()	returns non-0 if arg is a letter or digit	ctype.h
8.	islower()	returns non-0 if arg is lowercase letter	ctype.h
9.	isupper()	returns non-0 if arg is uppercase letter	ctype.h
10.	acos()	returns arc cosine of arg	math.h
11.	sqrt()	returns square root of num	math.h
12.	fabs()	returns absolute value of num	math.h
13.	asin()	returns arc sine of arg	math.h
14.	atan()	returns arc tangent of arg	math.h
15.	cos()	returns cosine of arg	math.h
16.	exp()	returns natural logarithim e	math.h

2. प्रयोक्ता परिभाषित फलन **(User Defined Functions)**

सभी आवश्यकताओं को पूरा कर सके ऐसी लाइब्रेरी का होना बहुत मुश्किल है। बहुत बार ऐसी आवश्यकता होती है कि हमें निर्देशों के समूह को कई बार निष्पादित करना होता है और ऐसा समूह किसी फलन के रूप में लाइब्रेरी में उपलब्ध नहीं होता। तब प्रयोक्ता परिभाषित फलनों की आवश्यकता पड़ती है। ऐसे फलन जिन्हें प्रयोक्ता अपनी आवश्यकतानुसार बनाता है, प्रयोक्ता परिभाषित फलन (Use Defined Function) कहलाते हैं। एक प्रोग्राम में एक से ज्यादा प्रयोक्ता परिभाषित फलन बनाए जा सकते हैं। एक प्रयोक्ता परिभाषित फलन को मुख्य फलन के ऊपर अथवा नीचे दोनों तरफ लिख सकते हैं।

2.3.4 फलन की घोषणा एवं परिभाषा **(Declaration of Function and Definition of a function)**

'C' भाषा में किसी प्रयोक्ता फलन को निम्न प्रकार से परिभाषित करते हैं।

```
वापसी का प्रकार फलन–नाम (प्राचलों की सूची)
{
        स्थानीय चरों की सूची (Local Variables)
        फलन प्रक्रिया में लिखे जाने वाले कथन की फलन काया
        (function body) (व्यंजक व चर)
}
```

फलन का नाम एक अभिज्ञानक (Identifier) की तरह होता है। किसी फलन को कॉल करने पर फलन

में लिखे निर्देशों के अनुरूप निश्पादन के पश्चात् हमें एक मान प्राप्त होता है। निश्चित रूप से इस मान का एक प्रकार होगा अतः फलन को परिभाषित करते समय, साथ ही हम वापसी का प्रकार भी देते हैं जिसमें बताते हैं कि return कथन के द्वारा जो परिणाम वापस मुख्य फलन में जायेगा, वह किस प्रकार का होगा। यदि हम वापसी का प्रकार नहीं देते हैं तो फलन integer मान ही वापस करता है। एक फलन में return कथन एकाधिक बार भी उपयोग में ले सकते हैं तथा यह भी आवश्यक नहीं है कि यह कथन अंत में ही लिखा जाए, आवश्यकतानुसार कहीं भी लिखा जा सकता है।

प्रोग्राम 5 : फलन द्वारा दो नम्बरों में से छोटा नम्बर ज्ञात करने हेतु C भाषा में प्रोग्राम लिखिए।

```
#include<stdio.h>
#include<conio.h>
/* It is C program to illustrate Function*/
/* It is use to find biggest number between two numbers*/
main() /* Main function */
{
    int A,B,big; /*Local variable for main function*/
    printf("\n Enter two number :");
    scanf("%d %d",&A,&B);
    big = fun_big(A,B); /*Function Call By Value*/
    printf("The biggest number : %d",big);
    getch();
}

/* Function return an integer value */
int fun_big(int C,int D)
{
    if( C > D )
        return(C);
    else
        return(D);
}
```

Result :

Enter two number : 45 85

The biggest number : 85

Enter two number : 105 85

The biggest number : 105

इसमें सबसे पहले मुख्य फलन (main) को निष्पादित करते हैं। जब मुख्य फलन में नियंत्रण प्रवाह (Flow Control) में fun_big फलन पर आ जाता है तथा उसमें लिखे निर्देशों को निष्पादित करता है। जिस समय नियंत्रण मुख्य फलन से प्रयोक्ता परिभाषित फलन पर आता है उस समय मुख्य फलन में फलन के नाम के साथ कोष्ठक में दिये गये मान भी प्रयोक्ता परिभाषित फलन में दिए गए चरों में आ जाते हैं।

जैसे उपर्युक्त फलन में फलन का निष्पादन पूर्ण होने के बाद नियंत्रण पुनः मुख्य फलन में चला जाता है

और आगे लिखे हुए निर्देशों की अनुपालना करता है।

इस प्रकार का फलन जिसमें हम प्रयोक्ता परिभाषित फलन को Call करते है "Calling Function" कहलाता है तथा वह प्रयोक्ता परिभाषित फलन जो Call होता है "Called Function" कहलाता है।

2.3.5 वास्तविक प्राचल **(Actual Parameters)** एवं औपचारिक प्राचल **(Formal/Dummy Parameters)**

जब हम मुख्य फलन में किसी प्रयोक्ता परिभाषित फलन को call करते हैं तो उस समय फलन के साथ कोष्ठक में जो पैरामीटर दिये जाते हैं, वे वास्तविक प्राचल (Actual Parameter) कहलाते हैं। वास्तविक प्राचल के रूप में कोई चर, स्थिरांक व एड्रेस भी हो सकते हैं।

प्रयोक्ता परिभाषित फलन की घोषणा या निर्धारण के समय जो प्राचल दिए गए होते हैं, वे औपचारिक प्राचल (Formal/Dummy Parameter) कहलाते हैं।

प्रयोक्ता परिभाषित फलन को Call करने पर वास्तविक प्राचल का मान औपचारिक प्राचल में कॉपी हो जाता है। इस बात का ध्यान रखना पड़ता है कि Call करते समय वास्तविक प्राचल की संख्या व औपचारिक प्राचल की संख्या बराबर हो।

उदाहरण :

```
int a ;
main ( )
{
   int x, y;
   int z;
   printf("Enter the first number\n");
   scanf("%d",&x);
   printf("Enter the second number\n");
   scanf("%d",&y);
   z = mul (x,y); /*वास्तविक प्राचल (x, y) */
   printf("%d\n",z);
}
int mul(int a, int b)/* औपचारिक प्राचल */
{
   int c;
   c = a * b;
   return(c) ;
}
```

इस समय a = x तथा b = y हो जायेगा।

2.3.6 फलन को **Call** करना **(Calling a Function)**

किसी भी मुख्य फलन (main function) में हम फलन को उसके नाम से call करते हैं। हमें यह हमेशा ध्यान रखना चाहिए कि मुख्य फलन में फलन का नाम हमेशा किसी व्यंजक में दायें हाथ (Right hand) की तरफ आना चाहिए। फलन दो प्रकार से Call किए जाते हैं –

1. मान द्वारा आह्वान (Call by Value)
2. एड्रेस द्वारा आह्वान (Call by Address)

1. मान द्वारा आवाहन् **(Call by Value)**

जब हम किसी फलन को Call करते समय उसमें प्राचलों के रूप में कोई चर व कोई स्थिरांक

फलन में भेजते हैं तो इस प्रकार का आवाहन् Call by Value कहलाता है।

उदाहरण :

```
main ()
{
    int mark1,mark2,mark3;
    float d;
    printf("Enter the first marks\n");
    scanf("%d",&mark1);
    printf(Enter the second marks\n");
    scanf("%d",&mark2);
    printf("Enter the third marks\n");
    scanf("%d",&marks3);
    d=ave(mark1, mark2, mark3); /*function call*/
    printf("Average of three marks is %f\n",d);
}
float ave(int x, int y, int z)
{
    float f;
    f = (x + y + z)/3;
    return (f);
}
```

इस उदाहरण में फलन को Call करते समय चरों को भेज रहे हैं।

2. संदर्भ द्वारा आह्वान **(Call by Reference)**

किसी फलन को Call करते समय कोई चर या स्थिरांक फलन में न भेजकर उसका एड्रेस (Address) भेजते हैं यह Call by Reference कहलाता है। यहाँ हमें एक बात का ध्यान रखना पड़ता है कि जिस फलन में हम एड्रेस भेज रहे हैं उस फलन में वह चर, जो एड्रेस को संग्रहित करता है वह साधारण चर नहीं होकर एक संकेतक (Pointer) चर होना चाहिए। साधारण चर एड्रेस संग्रहित नहीं कर सकते हैं।

उदाहरण :

```
main ()
{
    int a,b;
    float d;
    printf("Enter the first number\n");
    scanf("%d",&a);
    printf("Enter the second number\n");
    scanf("%d",&b);
    d=ave(&a,&b);
    printf("Average of two number is %f\n",d);
```

```
}
float ave(int*x, int *y)
{
    float f;
    f = (*x + *y)/2
    return (f)
}
```

यहाँ फलन में ave(&a, &b) से a,b के addresses फलन में भेजा गया है। माना कि यह addresses 1000 व 1050 हैं, जिससे x का मान 1000 तथा y का मान 1050 होगा। यदि int *x, int *y की जगह हम int x, int y दे देते हैं तो गलत होता क्योंकि यहाँ x व y दो साधारण चर हो जाएँगे जो a व b के addresses को संग्रहीत नहीं कर सकेंगे।

2.3.7 फलन की विभिन्न श्रेणियाँ **(Categories of Functions)**

फलन को उसके प्राचलों के फलन में भेजने के अनुसार विभिन्न श्रेणियों में विभक्त करते हैं।

1. प्राचल रहित व वापसी रहित मान के साथ फलन (Function with No Arguments and No Return Values).
2. प्राचल सहित व वापसी रहित मान के साथ फलन (Function with Arguments But No Return Values).
3. प्राचलों व वापसी मान दोनों के साथ फलन (Function with Arguments and Return Value)

1. प्राचल रहित व वापसी रहित मान के साथ फलन **(Function with No Arguments and No Return Values)**

इस प्रकार के फलनों में हम फलन में ना तो कोई प्राचल भेजते हैं तथा ना ही फलन हमें वापस कोई वापसी मान भेजता है। इस प्रकार की श्रेणी में केवल नियंत्रण ही मुख्य फलन से प्रयोक्ता परिभाषित फलन में तथा प्रयोक्ता परिभाषित फलन से मुख्य फलन में आता व जाता है। किसी प्रकार का कोई डाटा उसके साथ नहीं होता है। इस प्रकार की श्रेणी का उपयोग केवल नियंत्रक को एक स्थान से दूसरे स्थान भेजने के लिए किया जाता है।

उदाहरण :

```
main ()
{
    printf("You are in Main program\n");
    ave();
}
void ave(int x, int y, int z)
{
    float f;
    printf("Enter the first marks\n");
    scanf("%d",&x);
    printf(Enter the second marks\n");
    scanf("%d",&y);
    printf("Enter the third marks\n");
```

```
    scanf("%d",&z);
    f = (x + y + z)/3;
    printf("Average of three marks is %f\n",d);
}
```

2. प्राचल सहित व वापसी रहित मान के साथ फलन **(Function with Argument But No Return Values)**

इस श्रेणी में हम फलन का आह्वान करते समय मुख्य फलन से प्राचलों को भेजते हैं, लेकिन फलन मुख्य फलन को किसी प्रकार का कोई भी मान वापस नहीं भेजता है। इसमें डाटा को हम मुख्य फलन में ही पढ़ लेते हैं तथा बाद में उसको प्रयोक्ता परिभाषित फलन में भेज देते हैं।

इस प्रकार की श्रेणी में एक बात का विशेष ध्यान रखना पड़ता है कि जब प्राचलों को मुख्य फलन से फलन में भेजा जाता है तब वास्तविक प्राचलों का डेटा टाइप तथा औपचारिक प्राचलों का डेटा टाइप समान होना चाहिए। इसमें वास्तविक प्राचलों के मान एक एक करके उसके साथ लिखे औपचारिक प्राचलों में कॉपी कर दिए जाते हैं।

उदाहरण :

```
main ()
    {
        int mark1,mark2,mark3;
        float d;
        printf("Enter the first marks\n");
        scanf("%d",&mark1);
        printf(Enter the second marks\n");
        scanf("%d",&mark2);
        printf("Enter the third marks\n");
        scanf("%d",&mark3);
        ave(a,b,c);
    }
    void ave(int x, int y, int z)
    {
        float f;
        f = (x + y + z)/3;
        printf("Average of three marks is %f\n",f);
    }
```

जब भी sum () फलन का नाम आता है तब नियंत्रक a व b चरों का मान प्रयोक्ता परिभाषित फलन में आता है जहाँ x = a तथा y निष्पादन पूरा होने के बाद नियंत्रण बिना कोई वापसी मान लिए मुख्य फलन में चला जाता है।

3. प्राचलों व वापसी मान के साथ फलन : **(Function with Arguments and Return Value)**

इस प्रकार की श्रेणी में प्रयोक्ता परिभाषित फलन को आवाहन् करते समय वास्तविक प्राचलों को भेजा जाता है औपचारिक प्राचलों द्वारा ग्रहण कर लिये जाते हैं। जब प्रयोक्ता परिभाषित फलन का निश्पादन पूरा हो जाता है तब नियंत्रण वापसी मान के साथ मुख्य फलन में जाता है। जब प्रयोक्ता

परिभाषित फलन में सभी मानों को मुख्य फलन में आगे कहीं काम में लेना होता है तब इस श्रेणी का उपयोग करते हैं। यहां रिटर्न कथन का उपयोग कर प्रयोक्ता परिभाषित फलन से वापसी मान प्राप्त करते हैं। रिटर्न कथन एक बार में केवल एक मान ही वापस करता है। अतः हम कह सकते हैं कि जब भी कभी हम किसी फलन का आह्वान करते हैं तो वह वापसी मान के रूप में एक आह्वान पर केवल एक मान ही वापस करता है। यहाँ उदाहरण में रिटर्न कथन लिखना जरूरी होता है तथा यह रिटर्न कथन अपने आप float मान की ही वापसी करता है।

उदाहरण :

```
main ()
{
    int mark1,mark2,mark3;
    float d;
    printf("Enter the first marks\n");
    scanf("%d",&mark1);
    printf("Enter the second marks\n");
    scanf("%d",&mark2);
    printf("Enter the third marks\n");
    scanf("%d",&mark3);
    d=ave(mark1, mark2, mark3);
    printf("Average of three marks is %f\n",d);
}
float ave(int x, int y, int z)
{
    float f;
    f = (x + y + z)/3;
    return (f);
}
```

2.4 सीमा के नियम (Scope rules)

हम जिन चरों के फलन में उपयोग करते है वह प्रोग्राम उसका उपयोग कहाँ कर सकते हैं यह निर्धारण सीमा के नियम द्वारा किया जाता है। प्रोग्राम में जिन चरों का उपयोग किया जाता है वह दो प्रकार के होते है।

1. लोकल चर (Local variable)
2. ग्लोबल चर (Global variable)

1. लोकल चर

यह फलन के अन्दर घोषित होते हैं। फलन के हैडर में लिखें चर भी लोकल चर कहलाते हैं। लोकल चरों की सीमा फलन की सीमा ही होती है। अतः लोकल चर केवल फलन के अन्दर ही कार्य करते हैं।

उदाहरण :

```
#include<stdio.h>
```

```
main()
{
    int a=5, b=20; /*Local variables*/
    printf("%d", b);
    fun();
}
fun()
{
int b=10;\*Local variable*/
printf("%d", b);
/*printf("%d", a); uncommented cause and error*/
}
```

output:

20 10

यहाँ पर तीन चर घोषित किए गए है। a और b चर main( ) फलन में b चर fun( ) फलन में घोषित किया गया है। दोनों b चर एक दूसरे से भिन्न हं। जिस फलन में जो चर घोषित है उसका उपयोग उसी फलन में कर सकते है। जैसे ं चर का उपयोग fun( ) फलन में नहीं कर सकते हैं।

2. ग्लोबल चर **(Global Variable)**

यह प्रोग्राम के एक स्थान पर ही घोषित होते हैं तथा इनको प्रोग्राम के किसी भी भाग में उपयोग कर सकते हैं। ग्लोबल चर हैडर फाईल के बाद एवं main() फलन से पहले घोषित होते हैं।

उदाहरण :

```
#include<stdio.h>
int a, b, c:/* Global declaration of vaiable*/
main()
{
    a=10; b=20; c=30;
    fun1();
    printf("a=%d  b=%d  c=%d", a , b, c);
}
funl()
{
    printf("\na=%d  b=%d  c=%d", a, b, c);
    a + = 5;
    c + = 10;
    return;
}
```

output:

a=10 b=20 c=30

a=15 b=20 c=40

a, b और c का मान किसी भी फलन में बदल सकते है। यहाँ यह ध्यान देने योग्य बात यह है कि अगर कोई चर (एक ही नाम से) ग्लोबल और लोकल दोनों जगह घोषित है तो उस फलन में लोकल चर का उपयोग ही होगा ग्लोबल चर का उपयोग नहीं होगा। ग्लोबल चर उस फलन में उपयोग होगा जहाँ पर उस नाम से लोकल चर घोषित नहीं है।

उदाहरण :

```
#include<stdio.h>
int a=10, b=20;
main()
{
        a=20; c=30;
        printf("a=%d b=%d c=%d",a, b,c);
}
```

output:

a=20, b=20, c=30

यहाँ पर a लोकल चर का मान प्रिन्ट करेगा। क्योकि लोकल चर की प्रायिकता ग्लोबल चर से अधिक होती है।

2.5 संचित वर्ग के प्रकार **(Storage classes type)**

किसी भी चर को दो प्रकार से समझ सकते हैं।

1. डाटा के प्रकार के आधार पर (Data Type)
2. संचित वर्ग के आधार पर (Storage class)

डाटा के प्रकार के आधार पर हम चर में स्टोर होने वाले मान को निर्धारित करते हैं तथा संचित वर्ग के आधार पर उस चर की सीमाएं और जीवन समय (life time) निर्धारित होता है।

संचित वर्ग चार प्रकार के होते है–

(i) automatic
(ii) external
(iii) static और
(iv) register

इनको क्रमशः auto, extern, static और register कीवर्ड का उपयोग करके लिखते हैं।

उदाहरण :

```
auto int x, y, z;
extern float r1, r2;
register char ch1;
static int c, d;
```

संचित वर्ग का कीवर्ड डाटा के प्रकार के कीवर्ड से पहले लिखते हैं। ग्लोबल और लोकल चरों के बारे में हम अध्याय के पैराग्राफ 2.4 में पढ़ चुके हैं।

2.5.1 **Automatic** चर

Automatic चर फलन के अन्दर घोषित होते है। तथा यह फलन के लिए लोकल चर होते हैं।

इनकी सीमा फलन के अन्दर होती है तथा इनका जीवन काल (life time) फलन कॉल होने से फलन समाप्त होने तक होता है। एक ही नाम से चर कई फलनों में घोषित कर सकते हैं। यह चर आपस में स्वतन्त्र होते है। Automatic चरों को घोषित करने के लिए auto कीवर्ड को उपयोग करते हैं।

उदाहरण :

auto int A;

अगर किसी चर का संचित वर्ग नहीं लिखा है तो automatic प्रकार का चर बन जाएगा (Default storage class is Automatic)।

उदाहरण :

```
#include<stdio.h>
#include<conio.h>
main()
{
        /*Function main() scope start here*/
        auto int total1=10;
        auto int total2=30;
        printf("\ntotal1 = %d",total1);
        fun();
        printf("\ntotal1 = %d",total1);
}/*Function main() scope End here*/
fun()
{
        /*Function fun() scope start here*/
        auto int total1=30;
        printf("\ntotal1 = %d",total1);
        /*printf("total2 = %d",total2);
        uncommented to get an error*/
}/*Function fun() scope End here*/
```

Result :

total1 = 10

total1 = 30

total1 = 10

उपरोक्त प्रोग्राम में total1 चर main() और fun() दोनों फलनों में घोषित है। यह दोनों total1 एक दूसरे से स्वतन्त्र होते हैं तथा फलन fun() कॉल के पहले तथा बाद में main() फलन के चर total1 का मान नहीं बदलेगा। तथा total2 को fun() फलन में उपयोग नहीं कर सकते हैं। यह केवल main() फलन में ही उपयोग हो सकता है।

2.5.2 External चर

External चर automatic चर की तरह एक फलन के लिए नहीं होते हैं। यह जिस फलन में

घोषित होते है उसके बाद आने वाले सभी फलनों में इस चर का उपयोग कर सकते हैं। यह ग्लोबल चर की तरह कार्य करते हैं। इसकी सीमा (scope) उस फलन से शुरू (जहाँ घोषित होते है) होती है तथा प्रोग्राम के अन्त तक होती है। तथा जब उस फलन का call करते है जिसमें यह घोषित है तब से इसका जीवन समय (life time) शुरू होता है तथा प्रोग्राम समाप्त होने तक रहता है। घोषित होने के बाद आने वाले सभी फलनों में इसको उपयोग कर सकते है। इसको घोषित करने के लिए extern कीवर्ड का उपयोग करना होता है।

उदाहरण :

```
extern int A, B, C;
extern char D, E;
```

2.5.3 Static चर

Static चर भी फलन के अन्दर ही घोषित होते हैं। और उनकी सीमा automatic चर की तरह होती है। Static चर फलन में लोकल चर की तरह होते हैं तथा यह चर अपने मान को सुरक्षित रखते है। जिस फलन में static चर घोषित है उसका प्रथम बार कॉल करने पर static चर घोषित हो जाता है तथा static चर का मान शून्य हो जाएगा। यदि कोई प्रारम्भिकरण नहीं किया गया है। तथा इसका जीवन समय फलन को प्रथम बार कॉल करते ही शुरू होता है तथा प्रोग्राम समाप्ति तक रहता है। फलन में static चर का अन्तिम मान होगा वह सुरक्षित रहेगा तथा जब हम फलन को दुबारा कॉल करेगें तो उसका वह मान उपयोग में आएगा। इसको घोषित करने के लिए static कीवर्ड का उपयोग करते हैं।

उदाहरण :

```
#include<stdio.h>
main()
{
    int I;
    for (I = 1; I <= 10; I ++);
    fun();
}
void fun();
{
    static int K=0;
    printf("%d", K);
    K++;
}
```

main() में फलन में fun() फलन को लूप के द्वारा 10 बार कॉल किया गया है। जब फलन को पहली बार कॉल करेगें तो इसका प्रारम्भिक मान शून्य (i.e. K=0) हो जाएगा। और यह K का मान शून्य प्रिन्ट कर देगा। और K मान 1 बढकर शून्य से 1 हो जाएगा। जब फलन दुबारा call होगा तो K का अन्तिम मान (प्रथम call का) 1 प्रिन्ट होगा और K मान 1 से बढकर 2 हो जाएगा तीसरी बार कॉल होने पर अन्तिम मान (दूसरे call का) 2 प्रिन्ट होगा और K मान 3 हो जाएगा। और यह K के दस मानों को प्रिन्ट करेगा। इसका आउटपुट निम्न होगा

```
0 1 2 3 4 5 6 7 8 9
```

2.5.4 Register चर

उपरोक्त तीन प्रकार के चर मैमोरी address पर स्टोर होते हैं। लेकिन Register प्रकार के चर

CPU के रजिस्ट्रर में स्टोर होते हैं। ज्यादातर उन चरों को register प्रकार घोषित करते हैं जो प्रोग्राम में बहुत ज्यादा उपयोग में आ रहे हैं। यह प्रोग्राम के निष्पादन का समय कम करने में उपयोगी होते हैं।

उदाहरण :

```
register int A, B, C;
register cahr x, y, z;
```

यह automatic चर की तरह कार्य करते है तथा फलन के लिए लोकल होते हैं। अगर हमनें इनको register प्रकार का चर घोषित कर दिया है तो जरूरी नही है कि वह registger प्रकार का ही घोषित हो। अगर CPU register खाली नहीं होगा तो यह automatic प्रकार का चर बन जाएगा। सभी चरों को संक्षिप्त में सारणी 2.3 मे दर्शाया गया है।

सारणी 2.3

वर्ग का प्रकार	कहाँ स्टोर होगा	प्रारम्भिक मान **(default)**	सीमा **(Scope)**	जीवन काल **(life time)**
Automatic	Memory	Garbage मान	जिस फलनमें घोषित है उसकी सीमामें कार्य करता है	जब तब कन्ट्रोलफलन में रहेगा जहाँ चर घोषित है।
External	Memory	शून्य	घोषित स्थान से सम्पूर्ण प्रोग्राम में।	घोषित होने के बाद प्रोग्राम समाप्ति तक।
Static	Memory	शून्य	जिस फलनमें घोषित है उसकी सीमामें कार्य करता है।	जिस फलन मेंघोषित है उसके प्रथम कॉलसे लेकर प्रोग्राम समाप्ति तक कार्यकरता है।
Register	CPU register	Garbage मान	जिस फलनमें घोषित है उसकी सीमा में कार्य करता है।	जब तक कन्ट्रोलफलन में रहेगा जहाँ चरघोषित है।

2.6 आव्यूहों के साथ फलनों का आवाहन् **(Arrays and Functions)**

मुख्य फलन से प्रयोक्ता परिभाषित फलन में यदि आव्यूह को भेजना हो तो मुख्य फलन में प्राचालों के रूप में हम उस आव्यूह (Array) का नाम तथा उसी आव्यूह की भण्डारण क्षमता (Size) देते हैं।

उदाहरण :

```
average (a,n);
```

यहां average प्रयोक्ता परिभाषित फलन का नाम हैं जहां a व n प्राचलों के रूप में दिए गए हैं जो क्रमशः आव्यूह का नाम व आव्यूह का आकार है। इस फलन की परिभाषा निम्नानुसार होगी।

```
float average (float arr[ ], int n1);
```

मुख्य फलन में average (a,n) वाला कथन निष्पादित करते समय नियंत्रण आव्यूह का नाम व आकार लेकर प्रयोक्ता परिभाषित फलन में आ जायेगा। float arr [] = a तथा n1 = n हो जायेगा।

arr के साथ '[ ]' कोष्ठक यह दर्शाते हैं कि arr एक आव्यूह है तथा इसका आकार n1 के बराबर है। यहाँ आव्यूह का आकार '[ ]' कोष्ठक में देने की आवश्यकता नहीं होती है क्योंकि यह आकार वास्तविक

प्राचल से निर्धारित होता है।

उदाहरण :

```
float average (float arr [],int n1);
main()
{
        float a[40],r ;
        int n;
        printf("Enter the size of the array\n");
        scanf("%d",&n);
        printf("Enter the element of the array\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        r=average(a,n);
        printf("The average of the element is%f",r);
}
float average(float arr[], int n1);
{
        int i, sum = 1;
        float ave;
        for(i=0;i<n1;i++)
        {
                sum = sum + arr[i];
        }
        ave = sum/n1;
        return(ave);
}
```

उदाहरण में हम मान लेते हैं कि आव्यूह का अधिकतम आकार 40 है लेकिन इस आव्यूह में हमने n = 10 मान ही भण्डारित किये हैं। जब मुख्य फलन में average (a, n) आता है। इसका मतलब average (a, 40) है। तब नियंत्रण इन दोनों को लेकर फलन को निष्पादित करने चला जाता है। जैसे ही नियंत्रण फलन में जाता है उस समय arr[ ] आव्यूह में वही मान भण्डारित हो जाते हैं जो a में थे तथा nl का मान n के बराबर हो जाता है। तब इस फलन को इन मानों के आधार पर निष्पादित किया जाता है तथा result को return कथन से पुनः मुख्य फलन में भेज दिया जाता है।

2.7 पुनरावर्तन **(Recursion)**

पुनरावर्तन वह प्रक्रिया है जिसमें कोई फलन अपने आप (itself) का ही आह्वान (call) करता है जब तक कि कोई विशेष वर्णित कन्डीशन संतुष्ट (satisfy) नहीं होती है। पुनरावर्तन फलन लिखने हेतु दो बातों का ध्यान रखना होता है।

1. फलन अपने आप का स्वयं आह्वान (call) करें।
2. फलन में कोई ऐसी कन्डीशन होनी चाहिये जिसके सन्तुष्ट होने पर पुनरावर्तन रूक जाए (Termination condition).

साधारणतया किसी फलन को अन्य फलन में ही call किया जाता है। बहुत सी प्रोग्रामिंग भाषाओं में यह सुविधा उपलब्ध है कि हम एक फलन को उसी फलन में भी कॉल कर सकते हैं तो इसे रिकर्शन कहते हैं। एक फलन को जब हम किसी अन्य फलन में call करते हैं तो प्रोग्राम नियंत्रण साधारणतया इसकी समाप्ति पर पुनः उसी फलन में आ जाता है। परन्तु जब फलन को उसी में call करते हैं तो वह पुनः उसी फलन में आ जाता है। यहाँ इस बात का खतरा रहता हैं कि कन्ट्रोल प्रवाह कभी भी फलन से बाहर नहीं आए। ऐसी स्थिति में प्रोग्राम चलेगा और जब कम्प्यूटर में संसाधन उपलब्ध नहीं होंगे तब त्रुटि संदेश देकर प्रोग्राम असामान्य रूप से बन्द हो जायेगा। अतएव इस बात का ध्यान रखना अत्यंत आवश्यक है कि फलन में कोई ऐसा प्रतिबन्ध हो जो फलन का निष्पादन बन्द कर सके और फलन को call करते समय ऐसा डाटा उसमें जाए जिसमें निष्पादन समाप्ति की परिस्थिति भी बने।

प्रोग्राम 6 : किसी दिये गये नम्बर का factorial रिकर्शन फलन द्वारा ज्ञात करना।

```
#include<stdio.h>
#include<conio.h>
/* It is C program to calculate the factorial of N*/
main() /* Main function */
{
      int N,f1; /*Local variable for main function*/
      int fact(int N);/* Function Prototype*/
      printf("\n Enter a number :");
      scanf("%d",&N);
      f1 = fact(N); /* Function Call By Value*/
      printf("The Factorial of %d is = %d",N,f1);
      getch();
}
/* Function return an integer value */
int fact(int k)  /* Function Defination */
{
      if(k==1)
            return(1);
      else
      return(k*fact(k-1));
}
```

Result :

Enter a number : 5

The Factorial of 5 is = 120

Enter a number : 7

The Factorial of 7 is = 5040

माना number = 5
प्रथम पद में return(5 * fact(4))
द्वितीय पद में return (5 * 4 * fact(3))
तृतीय पद में return (5 * 4 * 3 * fact(2))
चतुर्थ पद में return (5 * 4 * 3 * 2*fact(1))
पांचवां व अन्तिम पद में return (5 * 4 * 3 * 2 * 1)

प्रोग्राम 7 : इसी प्रकार हम एक अन्य उदाहरण के द्वारा रिकर्शन फलन को समझते हैं।
माना हमे xy का मान बारम्बारतः गुणा (successive multiplication) से ज्ञात करना है।

```
int power(int a, int b);
main()
{
        int x,y,z;
        printf("Enter the Number\n");
        scanf("%d\n",&x);
        printf("Enter the power\n");
        scanf("%d\n",&y);
        z = power(x,y);
        printf("The result is\n");
        printf("%d\n",z);
}
int power(int a, int b)
{
        if (b==1)
                return a;
        else
                return(a*power(a,b-1));
}
```

प्रोग्राम 8 : फिबोनेसी संख्या का N वीं संख्या ज्ञात करने हेतु प्रोग्राम लिखिए।

```
#include<stdio.h>
#include<conio.h>
/* It is C program to calculate Nth Fibonacci Number*/
/* Fibonacci Number Start with 1,1,2,3,5,8,13,21,34,55*/
main()
{
        int N,f1;
        int fib(int N);/* Function Prototype*/
        printf("\nEnter a number :");
        scanf("%d",&N);
```

```
        f1 = fib(N); /* Function Call By Value*/
        printf("The Nth Fibonacci Number is = %d",f1);
        getch();
}
/* Function return an integer value */
int fib(int K)  /* Function Defination */
{
        int i,f1=1,f2=1,f3; /*Local variable for fact function*/
        for(i=1;i<=K-2;i++)
        {
                f3 = f1 + f2;
                f1 = f2;
                f2 = f3;
        }
        return(f3);
}
```

Result :
Enter a number : 5
The Nth Fibonacci Number is = 5
Enter a number : 10
The Nth Fibonacci Number is = 55

प्रोग्राम 9 : द्विघातीय समीकरण $ax^2 + bx + c = 0$ का हल करने हेतु फलन लिखिए।

```
        #include<stdio.h>
        #include<conio.h>
        #include<math.h>
        main()
        {
                void Q_EQ(int p,int q,int r);
                int a,b,c;
                printf("Enter the value of a,b and c :");
                scanf("%d%d%d",&a,&b,&c);
                Q_EQ(a,b,c);
                getch();
        }

        void Q_EQ(int p,int q,int r)
        {
                float r1,r2,D;
```

```
        D = q * q - 4.0 * p * r;
        if(D < 0)
            printf("Roots are imaginary");
        else
        {
            if(D == 0)
            {
                printf("Roots are Equals\n");
                r1 = r2 = - q / (2.0 * p);
            }
            else
            {
                r1 = (- q - sqrt(D)) / (2.0 * p);
                r2 = (- q + sqrt(D)) / (2.0 * p);
            }
            printf("\nRoots are : %f %f", r1,r2);
        }
    }
```

प्रोग्राम 10 : एक प्राइम फलन लिखिए जो कि 1 रिटर्न करें यदि दिया गया आरग्यूमेन्ट प्राइम संख्या हो अन्यथा 0 रिटर्न करें।

```
    #include<stdio.h>
    #include<conio.h>
    #include<math.h>
    main()
    {
        int prime(int p);
        int flag,N;
        printf("\nEnter the value of N :");
        scanf("%d",&N);
        flag = prime(N);
        if(flag == 1)
            printf("\nNumber is prime");
        else
            printf("\nNumber is not prime");
        getch();
    }

    int prime(int p)
```

```
{
    int i,r1;
    r1 = (int)sqrt(p);
    for(i=2; i<=r1; i++)
        if(p % i == 0)
            return(0);
    return(1);
```

2.8 संकेत (Pointer)

पॉइन्टर (Pointer) एक उपयोगी तथा शक्तिशाली टूल है। प्रत्येक चर जिस स्थान पर प्राथमिक मैमोरी में स्टोर होता है। उसे उस चर का पता (Address) कहते है। प्रत्येक चर के लिए एक मैमोरी एड्रेस (Address) होता है।

पॉइन्टर एक चर है जिसमें आँकड़े का पता (मैमोरी में आंकड़े का पता) भंडारित (Store) होता है। C भाषा में उच्च तथा जटिल प्रोग्राम बनाने के लिए पॉइन्टर का उपयोग किया जाता है।

कोई भी चर अपने आकड़े के प्रकार (data type) के अनुसार मैमोरी भंडारित करता है । char प्रकार का डाटा 1 बाइट्स में int प्रकार का डाटा 2 बाइट्स में तथा float प्रकार का डाटा 4 बाइट्स में भंडारित होता है। इसमें प्रत्येक बाइट्स के ग्रुप का एक मैमोरी पता होता है जो उस चर के साथ होता है। जैसे int प्रकार का डाटा 2 बाइट्स लेता है। और इसका पता क्रमशः FAB व FAC है तो उस चर का पता FAB ही होगा। यह पहले मैमोरी सेल का Address होता है।

माना d एक चर है जो किसी आँकड़े को प्रदर्शित करता है। कम्पाइलर इस चर को उसके आँकड़े के प्रकार के अनुसार बाइट्स आवंटित कर देता है। जहाँ स्वयं का मान भंडारित (store) करता है। इस आँकड़े को आसानी से एक्सेस (Access) कर सकते है। माना d चर में हमनें 10 स्टोर किया है।

K पॉइन्टर एक प्रकार का चर है यह केवल मैमोरी पता ही स्टोर कर सकता है अतः K के अन्दर मैमोरी पता 4522(d का मैमोरी में पता) स्टोर होगा।

उपरोक्त कार्य हेतु C भाषा में दो एकल संकारक (operator) उपयोग में लिये जाते है। * संकारक द्वारा पॉइन्टर चर को घोषित करते है तथा पॉइन्टर चर में स्टोर एड्रेस (Address) पर मान ज्ञात करने के उपयोग आता है। इस संकारक को मान संकारक (value operater) भी कहते है तथा & operator के द्वारा किसी चर का मैमोरी पता ज्ञात किया जा सकता है। जैसे d चर का पता ज्ञात करना हो तो हमें &d लिखना होता है। अतः पॉइन्टर चर में d का एड्रेस लिखने के लिए निम्न कथन का उपयोग करते है।

K = &d;

उपरोक्त विवरणानुसार *K और d समान मान को प्रदर्शित करेंगे।

संकेत चर को घोषित निम्न प्रकार करते है।

```
int *A;
float *B;
double *C;
```

प्रोग्राम 11 : पॉइन्टर का उपयोग करते हुए पाँच पूर्णाक संख्याओं को जोड़ने हेतु प्रोग्राम लिखिए।

```
#indclude<stdio.h>
#include<conio.h>
```

```
#include<alloc.h>
main()
{
    int I, *A,sum=0, B;/* A is a poniter type variable */
    A = &B;
    for(I=1; I < = 5, I++)
    {
                    printf("Enter a number:");
                    scanf("%d", A);
                    /*don't use ampersand sign with pointer variable.*/
                    sum=sum + *A;
    }
    printf("\nSum of Five Numbers = %d", sum);
    getch();
}
```

2.8.1 संकेतक व्यंजक

संकेतक व्यंजक ऐसे व्यंजक है जिनमें संकेतक चर का उपयोग होता है। यदि P1 और P2 और दो संकेतक चर है। जिनकी घोषणा तथा प्रारम्भन किया गया है।

1. निर्धारण व्यंजक

जिसमें निर्धारण किया है उसे निर्धारण व्यंजक कहते हैं।

उदाहरण :

```
P2++;
--P1;
P1 = P1 + 1;
*P2 = *p2 + 1
```

प्रथम तीन व्यंजकों के द्वारा पॉइन्टर चर में स्टोर मैमोरी Address में एक लोकेशन की वृद्धि अथवा ह्रास होगा। तथा चौथे व्यंजक में P2 में स्टोर मैमोरी Address पर स्टोर मान में एक की वृद्धि होगी।

2. अंकगणितीय व्यंजक

```
P2 = P2 + 1;
P2 = P1 + 1;
```

पॉइन्टर चर में मैमोरी location बढ़ा तथा घटा सकते है। निम्न अंकगणितीय संक्रियाएँ जो संकेतकों पर वैद्य नहीं है निम्न है :

P1 / P2	दो संकेतकों को घटाया जा सकता है यदि वे एक ही प्रकार के
P1 / 50	अवयवों को इंगित कर रहे हो। सामान्यतः दो संकेतकों को
P1 + P2	तभी घटाया जा सकता है जब वे किसी एरे के अवयवों
P1 * P2	को इंगित कर रहे हों ।

P1 * 17

3. तुलनात्मक व्यंजक

संकेतकों पर तुलनात्मक संकारकों का प्रयोग भी किया जाता है।

उदाहरण :

P2 > P1
P1 == P2
P2 >= P1
P1 != P2

2.8.2 संकेतक के लाभ (Advantages of pointers)

1. आंकड़ो की सारणी को संकेतक कुशलता से नियंत्रित करते है।
2. संकेतक के द्वारा फलन में सूचना को भेज सकते है जिससे यह फलन में किये गये बदलाव को कॉल किये गये चरों में भी अपने आप बदलाव हो जाता है।
3. ये प्रोग्राम निष्पादन की गति बढ़ा देते हैं।
4. इसके द्वारा एरे के तत्व को आसानी से एक्सेस (Access) कर सकते हैं।
5. संकेतक की सहायता से फलन से बाहर घोषित चर को भी एक्सेस कर सकते हैं।

2.9 गतिज मैमोरी निर्धारण (Dynamic Memory Allocation)

यहां पर मैमोरी निर्धारण दो प्रकार से की जाती है।

(1) स्थिर मैमोरी निर्धारण

(2) गतिज मैमोरी निर्धारण

1. Static मैमोरी का निर्धारण प्रोग्राम निष्पादन से पहले ही निर्धारित हो जाता है। यह एरे में उपयोग होता है। तथा इसको निष्पादन के समय बदल नहीं सकते हैं।

गतिज मैमोरी का निर्धारण निष्पादन शुरू होने के बाद किया जा सकता है। इसका निर्धारण आवश्यकता अनुसार किया जाता है। C भाषा में गतिज मैमोरी निर्धारण के फलन सारणी 2.4 में दर्शाए गए है।

सारणी 2.4

फलन	वापसी मान	कार्य
malloc (size)	Pointer void	आवश्यकता के अनुसार दिये गये size की मैमोरी एलोकेट होगी।
calloc (size)	Pointer void	आवश्यकतानुसार दिये गये size की मैमोरी एलोकेट होगी तथा यह दिये गये size के ब्लॉक में विभक्त हो जाएगी।
free (संकेतक)	कोई नहीं	संकेतक में निर्धारण मैमोरी Free हो जाएगी।
realloc (size)	Pointer void	पूर्व निर्धारित मैमोरी को घटाने व बढ़ाने हेतु ।

1. molloc( ) फलन

malloc() फलन द्वारा मैमोरी का ब्लॉक एलोकेट कर सकते हैं। malloc() फलन दिये गये साइज

की मैमोरी एलोकेट होगी तथा इस मैमोरी का टाइप void होगा हम type casting के द्वारा void मैमोरी को दूसरे डाटा प्रकार की मैमोरी में बदल सकते हैं। malloc () फलन को निम्न प्रकार लिख सकते हैं।

Str = malloc (size in byte)

malloc द्वारा एलोकेट मैमोरी का type बदलना हो तो (अगर Str int तरह के डाटा मैमोरी का address स्टोर करेगा) तब malloc फलन को निम्न प्रकार लिखेंगे

Str=(int*)malloc(size in byte)

उदाहरण :

```
int *integer;
integer=(int *) malloc (sizeof(int) * 10);
```

उपरोक्त कथन 20 byte मैमोरी में ब्लॉक एलोकेट करेगा। एक integer नम्बर 2 byte में स्टोर होता है और यह integer Type का pointer वापस करता है।

उदाहरण :

```
char *cptr;
cptr=(char *) malloc(20);
```

यह भी 20 byte memory allocate करेगा तथा char type का pointer वापस (return) करेगा। इसमें चित्र 4.5 में दर्शाया गया है।

2. calloc( ) फलन

calloc() फलन द्वारा भी मैमोरी ऐलोकेशन किया जाता है। यह समान साइज के ब्लॉक में मैमोरी को विभक्त करता है। इस को निम्न प्रकार लिखते हैं।

Str = (type cast*) calloc (n, sizeof(element))

दिये गये अवयव की साइज के n ब्लॉक एलोकेट करेगा तथा प्रत्येक ब्लॉक में शून्य स्टोर होगा।

उदाहरण :

```
int *cptr,
cptr = (int *) calloc(10, size of (int));
```

2 बाइटस के 10 ब्लॉक को एलोकेट करेंगें तथा प्रत्येक ब्लॉक में शून्य स्टोर होगा।

3. मैमोरी छोड़ना **(Releasing Memory)**

जब कोई मैमोरी ब्लॉक हमें उपयोग नहीं करना होता है या प्रोग्राम में उस मैमोरी की आवश्यकता नहीं होती है तो हम उस मैमोरी को छोड़ देते हैं जिससे वह भविष्य में किसी डाटा के लिए एलोकेट कर सके । इसके लिए free फलन का उपयोग करते है। इसको निम्न प्रकार लिखते हैं।

free(Str);

Str एक संकेतक है तथा जिसमें calloc( ) या malloc( ) फलन द्वारा मैमोरी एलोकेशन की गयी है।

4. मैमोरी ब्लॉक के साइज को बदलना **(Altering the size of memory block)**

realloc( ) फलन द्वारा मैमोरी ब्लॉक के साइज को घटा या बढ़ा सकते हैं।

उदाहरण :

malloc( ) फलन द्वारा एलोकेट मैमोरी

Str=malloc(size)

तब इसको realloc( ) फलन द्वारा साइज निम्न प्रकार घटा या बढ़ा सकते हैं।

Str=realloc(ptr, new_size)

नई साइज का मैमोरी एलोकेशन होगा।

नोट : अगर मैमोरी पर्याप्त नहीं होने पर malloc( ) या calloc( ) फलन पॉइन्टर चर में NULL को वापस (retrun) करेगा।

2.10 संकेतक और एरे (Pointer and Array)

हम पहले पढ़ चुके है कि एरे को घोषित करते समय ही एरे के लिए स्थान निश्चित हो जाता है। जिसमें एरे के अवयवों को भंडारित कर दिये जाते हैं। इस एरे का बेस पता (Base Address) उस एरे के प्रथम अवयव के स्थान का पता होता है।

माना एक एरे A को निम्न प्रकार से घोषित की गयी है :

int A[5] = {7, 9, 15, 12, 14};

यह भी माना कि इसका बेस पता 2000 है। चूंकि पूर्णांक (int) दो बाइट का स्थान घेरता है अतः एरे A के पाँच अवयव निम्न प्रकार से भंडारित होंगे।

हमें यह भी ज्ञात है कि एरे का नाम भी एरे का बेस पता को बतलाता है अतः A और &A[0] दोनों ही एरे के पहले अवयव का पता (Base Address) बताते है। तथा एरे के अवयव मैमोरी में लगातार स्टोर होते हैं।

इसी प्रकार पॉइन्टर चर भी उस मैमोरी ब्लॉक के बेस एड्रेस को ही स्टोर करता है। अतः एरे और पॉइन्टर को एक दूसरे के साथ आपस मे बदल सकते हैं।

उदाहरण :

```
int*ptr;
int A[5] = {7, 9, 15, 12, 14};
                ptr = A/* ptr = &A[0]*/
```

उपरोक्त उदाहरण में ptr और A दोनों ही एरे की तरह कार्य करेगें। दोनों को ही हम पॉइन्टर रूप मे तथा परम्परागत रूप में उपयोग कर सकते हैं।

जैसे ptr[1] का मान 9 होगा तथा *(A + 1) का मान 9 होगा। इसको निम्न उदाहरण से समझा जा सकता है।

प्रोग्राम 12 :

```
#include<stdio.h>
#include<conio.h>
main()
{
    int *ptr,I;
    int A[5] = {7, 9, 1, 2, 4};
    ptr = A;
    clrscr();
```

```
    for(I=0;I<+5;I++)
    {
                    printf("\nI = %d ptr[I] = %d",I,ptr[I]);
                    printf(" *(A+I) = %d &ptr[I] = %x",*(A+I),&ptr[I]);
                    printf(" (A+I) = %x",ptr[I]);
    }
    getch();
}
```

आउटपुट निम्न होगा :

I = 0	ptr[I] = 7	*(A+I) = 7	&ptr[I] = ffca	(A+I) = ffca
I = 1	ptr[I] = 9	*(A+I) = 9	&ptr[I] = ffcc	(A+I) = ffcc
I = 2	ptr[I] = 1	*(A+I) = 1	&ptr[I] = ffce	(A+I) = ffce
I = 3	ptr[I] = 2	*(A+I) = 2	&ptr[I] = ffd0	(A+I) = ffd0
I = 4	ptr[I] = 4	*(A+I) = 4	&ptr[I] = ffd2	(A+I) = ffd2

उपरोक्त प्रोग्राम में पॉइन्टर चर को परम्परागत रूप में तथा एरे को पॉइन्टर रूप में लिखा गया है। जोकि यह दर्शाता है कि एक विमीय एरे तथा पॉइन्टर को आपस में बदला जा सकता है। यहाँ पर ध्यान रखना आवश्यक है कि अगर पॉइन्टर चर में 1 को जोड़ते है तो उसमें 2 बाइट्स बढ़ते हैं। यह इसमें स्टोर डाटा प्रकार पर निर्भर करता है। यदि यह long int होगा तो पॉइन्टर चर में 1 को जोड़ते है तो चार बाइट बढ़ते हैं।

प्रोग्राम 13 : 10 पूर्णांकों को व्यवस्थित करने हेतु सलेक्शन शॉर्ट के लिए 'सी' भाषा में प्रोग्राम लिखिए।

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<string.h>
/*It is C program to sort 10 Numbers. */
int main()
{
    int *A;
    /* Array A declared as pointer type. */
    int i,j,temp;
    A = (int *)malloc(20);
    printf("\nEnter Ten Numbers :\n");
    for(i=0;i<+10;i++)
                    scanf("%d",(A+i));
    /* Sorting Algoritham */
    for(i=0;i<+9;i++)
                    for(j=i+1;j<+10;j++)
                            if(*(A+i) > *(A+j))
```

```
                    {
                        temp = *(A+i);
                        *(A+i) = *(A+j);
                        *(A+j) = temp;
                    }
    printf("\nThe Sorted Numers are :\n");
    for(i = 0; i<+10;i++)
                        printf("%d\n", *(A+i));
    getch();
    return 0;
}
```

Result:

Enter Ten Numbers :

20 10 40 60 50 30 80 70 90 75

The Sorted Numbers are :

10 20 30 40 50 60 70 75 80 90

प्रोग्राम 14 : संकेतक का उपयोग करते हुए दो स्ट्रिंग जोड़ने के लिए प्रोग्राम लिखिए।

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<string.h>
main()
{
    char *s1,*s2,*s3;
    int n1,i,j;
    /*Memory Allocation */
    printf("\nEnter the length of first string  : ");
    scanf("%d",&n1);
    s1 = (char *)malloc(n1+1);
    printf("\nEnter the length of second string : ");
    scanf("%d",&n1);
    s2 = (char *)malloc(n1+1);
    s3 = (char *)malloc(strlen(s1) + strlen(s2) -1);
    printf("\nEnter Two strings :\n");
    scanf("%s %s",s1,s2);
    /* Algorithm */
    strcpy(s3,s1);
    i=strlen(s3);
    j=0;
```

```
    while(s2[j] != '\0')
    s3[i++] = s2[j++];
    s3[i] = '\0';
    printf("Strings are : \n %s %s %s",s1,s2,s3);
    getch();
}
```

2.11 संरचनाएँ (Structures)

अभी तक हमनें एरे के बारे में पढ़ा जिसमें सभी अवयव एक ही प्रकार के होते हैं। लेकिन संरचनाओं (structures) में विभिन्न प्रकार के अवयवों को एक साथ रख सकते हैं। अतः एक संरचना में उपयोगकर्ता (user) के अनुसार int, float, double, char और array प्रकार अलग–अलग आँकड़ों का उपयोग कर सकते हैं। संरचना के प्रत्येक अवयव को सदस्य (member) कहते हैं। माना कि हमें कक्षा के 40 विद्यार्थियों के लिए उनके नाम (name), Basic, पता (Address) तथा HRA भंडारित करने हो तो अभी तक हमने पढ़ा की एरे सबसे अच्छा तरीका है। लेंकिन इनमें नाम, रोल नं., पता और प्राप्तांक के लिए अलग अलग एरे लेकर उसमें आंकड़ों को भंडारित करते हैं। परन्तु संरचनाओं का उपयोग करके हम इसे आसान बना सकते हैं। इसमें एक संरचना Employee बनायेंगे जिसके सदस्य नाम, बेसिक, पता तथा HRA होगें। इस संरचना का एरे बनाकर हम 40 कर्मचारियों की सूचना को भंडारित कर सकते हैं।

2.11.1 संरचना की घोषणा (Declarartion of Structure)

संरचना की घोषणा करना कठिन होता है क्योंकि संरचना में प्रत्येक सदस्य की घोषणा की जाती है। संरचना को घोषित करने के लिए रचना (syntax) निम्नलिखित है :

```
struct <tag>
{
        <data type> member1;
        <data type> member2, member3;
        ……
        ……..
        <data type> membern;
};
```

उपरोक्त संरचना घोषणा में struct की वर्ड है। tag संरचना का अभिज्ञानक (identifier) है। तथा member1, member2, member3.....member_n अलग अलग सदस्य घोषित किए गए हैं। सदस्यों का प्रकार पॉइन्टर, एरे, साधारण चर तथा अन्य संरचना हो सकती है। जिन सदस्यों का डाटा का प्रकार समान हो उनको एक साथ घोषित कर सकते हैं। जैसे उपरोक्त घोषणा में datatype member2, member3; को एक साथ घोषित किया गया है।

उदाहरण :

एक employee संरचना को घोषित कीजिए जिसके सदस्य निम्न है :

name, basic, address, HRA

```
struct employee
{
```

```
        char    name[20];
        int     basic;
        char    address[25];
        int     HRA;
    };
```

उपरोक्त employee संरचना उपयोगकर्ता घोषित संरचना कहलाती है। अब employee जैसी कई संरचनाएँ घोषित कर सकते है।

उदाहरण :

```
struct employee emp1, emp2;
```

emp1 और emp2 employee तरह की संरचनाएँ होगी। उपरोक्त दोनों घोषणाओं को एक ही कथन में सम्मिलित कर सकते हैं ।

उदाहरण :

```
struct tag
{
    <data type> member 1
    <data type> member2
    ..........
    ..........
    data type member n;
}   vaiable1, variable2;
```

इस दशा (case) में tag को लिखना जरूरी नहीं है। उपरोक्त उदाहरणों को निम्न प्रकार लिखेंगे।

```
struct employee
{
    char    name[20];
    int     basic;
    char    address[25];
    int     HRA;
}emp1, emp2;
```

उपरोक्त उदाहरण में हमनें साधारण चर तथा एरे चर का उपयोग किया है।

उदाहरण :

```
struct employee
{
    char    name[20];
    int     basic;
    char    address[25];
```

```
    int     HRA;
}emp1, emp2;
```

हम संरचना के अन्दर दूसरी संरचना को भी सदस्य बना सकते हैं।

उदाहरण :

```
struct  date
{
    int  day;
    int  month;
    int  year;
};
struct employee
{
    char    name[20];
    int     basic;
    char    address[25];
    int     HRA;
    struct  date DOB;
}emp1;
```

उपरोक्त उदाहरण में DOB सदस्य, एक date की तरह की संरचना हैं। यह employee संरचना के अन्दर लिखा गया है।

हम संरचना को घोषित करते समय भी प्रारम्भिकरण कर सकते है। उदाहरण :

struct employee emp1 = {"Sangeeta Gupta", 8000, "153, Arya Nagar", 1200, 25, 01, 1976};

पहला मान struct के पहले सदस्य में स्टोर होगा दूसरा मान संरचना के दूसरे सदस्य में स्टोर होगा।

2.11.2 मैमोरी मैप **(Memory Map)**

कई संरचना मैमोरी में अलग–अलग सदस्य के लिए अलग–अलग मैमोरी एक साथ ऐलोकेट होती है। उस संरचना की साइज उसमें होने वाले सदस्यों के अलग–अलग साइज के योग के बराबर होगी। उपरोक्त employee संरचना का मैमोरी मैप निम्न होगा–

इस संरचना में उपयोग होने वाली मैमोरी होगी :

20 + 2 +25 + 2 + 2 + 2 + 2 = 55 bytes

इसकी साइज की गणना करने हेतु sizeof संकारक होता है जिससे इसकी साइज ज्ञात की जा सकती है।

उदाहरण :

```
sizeof(employee)
output – 55
```

2.11.3 संरचना की प्रक्रिया

संरचना के सदस्य ज्यादातर अलग–अलग ही प्रक्रिया में भाग लेते हैं। संरचना के सदस्यों को निम्न प्रकार एक्सेस (Access) कर सकते हैं।

<संरचना का नाम>.<संरचना का सदस्य>

इसमें .(dot) संकारक का उपयोग किया जाता है। निम्न उदाहरण को देखिए।

```
struct employee
{
    char    name[20];
    int     basic;
    char    address[25];
    int     HRA;
    struct  date DOB;
}emp1;
```

st1 संरचना के सदस्यों को एक्सेस निम्न प्रकार करेंगे।

```
emp1.basic = 8000
emp1.HRA = 1200
strcpy(emp1.name, "Yashika');
```

इसी तरह से हम संरचना के सभी सदस्यों को पढ़ और लिख सकते हैं।

संरचना के सदस्यों को पढ़ने के लिए :

```
scanf("%d', &emp1.basic);
scanf("%d', &emp1. HRA);
scanf("%s', emp1.name);
```

संरचनाओं के सदस्यों को लिखने के लिए

```
printf("%d', emp1.basic);
printf("%d', emp1. HRA);
printf("%s', emp1.name);
```

अगर किसी संरचना के अन्दर संरचना को घोषित किया गया है तो उप सदस्यों के एक्सेस करने हेतु .(dot) संकारक का उपयोग करते हैं। इसको निम्न प्रकार लिखते हैं

<संरचना का नाम>.<संरचना का सदस्य>.<संरचना का उप सदस्य>

उपरोक्त उदाहरणों में DOB सदस्य date की संरचना है जो निम्न प्रकार घोषित हैं।

```
struct date
{
    int  day;
    int  month;
    int  year;
};
```

DOB सदस्य को निम्न प्रकार एक्सेस करेंगे।

```
emp1.DOB.day=25
emp1.DOB.Month=01
emp1.DOB.Year=1976,
```

अगर एक संरचना के दो चर घोषित किए गए हो तो उनमें Assignment संकारक का उपयोग कर एक चर का मान दूसरे चर में स्टोर कर सकते हैं।

उदाहरण :

struct student emp1={"Jahanvi Gupta", 12000, "Alwar", 1200, 01, 11, 2006}, emp2;

अगर हम assigment संकारक का उपयोग करें तो वाक्य नियमानुसार लिख सकते हैं।

```
emp2 = emp1;
```

इस कथन के द्वारा emp1 के सदस्यों का मान emp2 में कॉपी हो जाएगा। समान संरचना चरों के सदस्य से सदस्य में कॉपी करने की जरूरत नहीं होती है।

2.12 संरचना और एरे (Structure and Array)

हम संरचना सदस्य में एरे को सदस्य के रूप में धोषित कर सकते हैं जैसा कि उपरोक्त उदाहरणों में किया गया है।

हम संरचना का एरे (structure of array) भी बना सकते है। जिससे अगर हमें 100 कर्मचारियों के डाटा को स्टोर करना हो तो हम संरचना का एरे बना कर सकते हैं। इसको निम्न प्रकार घोषित करते हैं:

```
struct employee
{
    char    name[20];
    int     basic;
    char    address[25];
    int     HRA;
    struct  date DOB;
};
struct employee emp1[100];
```

उपरोक्त उदाहरण में emp1 dks employee संरचना का एरे बनाया गया है। इसको निम्न प्रकार ऐक्सेस किया जाता है।

संरचना का नाम[subscript Value].संरचना का सदस्य

उदाहरण :

```
emp1[0].basic=12000;
emp1[1].basic=8000;
```

2.13 संरचना और फलन

हम फलन में संरचना के सदस्यों को साधारण चरों की तरह भेज सकते हैं।

उदाहरण :

Calculate(emp1.basic,emp1.HRA);

Basic और HRA संरचना emp1 के सदस्य है। हम सम्पूर्ण संरचना को भी एक साथ फलन में भेज सकते हैं। जैसे emp1 एक employee तरह की संरचना है तो

Calculate(emp1); /*calling*/

उदाहरण :

void Calculate (struct employee Temp)

{

कथन

}

emp1 के सभी सदस्य temp के सदस्यों में कॉपी हो जाएगा।

प्रोग्राम 15 : Employee संरचना को घोषित करो जिसके सदस्य निम्न है।

name, address, basic pay, HRA और *DA* हैं। इसके मान स्टोर करो तथा कुल वेतन की गणना करो।

```
#include<stdio.h>
#include<conio.h>
main()
{
    struct employee
    {
                    char name[20];
                    char address[25];
                    int basic_pay;
                    float da,hra;
                    float gross_pay;
    };
    struct employee e1;
    /* Read structure */
    printf("\nEnter Employee Data :");
    printf("\nName     : ");
    gets(e1.name);
    printf("Address  : ");
    gets(e1.address);
    printf("Basic Pay : ");
    scanf("%d",&e1.basic_pay);
    printf("DA(%)    : ");
    scanf("%f",&e1.da);
    printf("HRA(%)   : ");
```

```
    scanf("%f",&e1.hra);
    /* Gross Pay Calculation */
    e1.gross_pay = e1.basic_pay + e1.basic_pay * e1.da/100 +
                                        e1.basic_pay * e1.hra/100;
    /* Printing */
    printf("\nEmployee Data :");
    printf("\nName     : %s",e1.name);
    printf("\nAddress  : %s",e1.address);
    printf("\nBasic Pay : %d",e1.basic_pay);
    printf("\nDA       : %5.2f%",e1.da);
    printf("\nHRA      : %5.2f%",e1.hra);
    printf("\nGross Pay : %8.2f",e1.gross_pay);
    getch();
}
```

Results:***(Input data)***

Enter Employee Data :
Name : Sunil Methi
Address : 153, Arya Nagar, Alwar.
Basic Pay : 9375
DA(%) : 43
HRA(%) : 15
Employee Data :***(Output Data)***
Name : Sunil Methi
Address : 153, Arya Nagar, Alwar.
Basic Pay : 9375
DA : 43.00%
HRA : 15.00%
Gross Pay : 14812.50

प्रोग्राम 16 : एक कक्षा के छात्रों के नाम, रोल नं. तथा चार विषयों के प्राप्तांकों को संरचना में उपयोग करते हुए भंडारित करो तथा प्रत्येक छात्र के लिए नाम, रोल नं. तथा कुल प्राप्तांकों को प्रिन्ट करने हेतु प्रोग्राम लिखिए।

```
#include<stdio.h>
#include<conio.h>
main()
{
        int i,j,tot;
        struct
        {
            char    name[20];
```

```
        int     roll_no;
        int     sub1,sub2,sub3,sub4;
    }student[5]; /* In this case tag not required*/
    /* Read structure*/
    for(i=0; i<+=4; i++)
    {
        printf("\nEnter Student %d Data :",i+1);
        printf("\nName      : ");
        scanf(" %[^\n]",student[i].name);
        printf("Roll Number   : ");
        scanf("%d",&student[i].roll_no);
        printf("Marks Subject 1 : ");
        scanf("%d",&student[i].sub1);
        printf("Marks Subject 2 : ");
        scanf("%d",&student[i].sub2);
        printf("Marks Subject 3 : ");
        scanf("%d",&student[i].sub3);
        printf("Marks Subject 4 : ");
        scanf("%d",&student[i].sub4);
    }
    /* Printing */
    for(i=0; i<+=4; i++)
    {
        printf("\nName      : %s",student[i].name);
        printf("\nRoll Number   : %d",student[i].roll_no);
        tot=student[i].sub1+student[i].sub2
        +student[i].sub3+student[i].sub4;
        printf("\nTotal Marks   : %d",tot);
    }
    getch();
}
```

महत्वपूर्ण बिन्दु

1. एरे दो प्रकार के होते है। एक विमीय एरे एवं बहुविमीय एरे।
2. प्रोग्राम का वह भाग जो अलग नाम से निर्देशों के समूह के रूप में पहचाना जाता है। फलन कहलाता है।
3. main () फलन भी यूजर डिफाइन फलन कहलाता है तथा यह प्रोग्राम में होना आवश्यक है।
4. प्रयोक्ता परिभाषित फलन की घोषणा या निर्धारण के समय जो प्राचल दिये गये होते हैं, वे औपचारिक प्राचल (formal /dummy parameter) कहलाते हैं।

5. जिन प्राचलों के साथ फलन कॉल किया जाता है वह actual parameters कहलाते हैं।
6. फलन दो प्रकार से call किये जाते हैं।
   (i) call by value (मान द्वारा आह्वान)
   (ii) call by reference (रेफरेन्स द्वारा आह्वान)
7. पुनरावर्तन वह प्रक्रिया है जिसमें कोई फलन स्वयं (itself) का ही आवाहन् करता है।
8. पुनरावर्तन फलन में कोई ऐसी कण्डीशन होनी चाहिए जिसके सन्तुष्ट होने पर पुनरावर्तन फलन रुक जाये।.
9. पॉइन्टर चर किसी साधारण चर का एड्रेस स्टोर करता हैं।
10. पॉइन्टर चर के द्वारा गतिज मैमोरी (Dynamic Memory) का निर्धारण किया जा सकता है।
11. calloc, malloc के द्वारा मैमोरी एलोकेट करते हैं।
12. free फलन के द्वारा मैमोरी को छोड़ देते हैं।
13. संरचना के द्वारा अलग–अलग प्रकार के आंकड़ों को एक साथ एकत्रित कर सकते हैं।
14. संरचना के सदस्यों को .(dot) ऑपरेटर के द्वारा एक्सेस किया जाता है।
15. संरचना के द्वारा लिंक लिस्ट बना सकते हैं।

अभ्यासार्थ प्रश्न

बहुचयनात्मक प्रश्न :

1. एक ही प्रकार के विभिन्न मानों का समूह कहलाता है :
   (अ) एरे (ब) फलन (स) स्ट्रिंग (द) इसमें से कोई नहीं
2. float arr[3][2] में कितने अवयव होते हैं –
   (अ) 2 (ब) 3 (स) 6 (द) 9
3. निम्न में से कौनसा कथन असत्य है :
   (अ) ग्लोबल चर सारे प्रोग्राम में उपयोग किया जा सकता है।
   (ब) auto चर main() फलन में घोषित किए जाते हैं।
   (स) लोकल चर केवल उसी फलन या ब्लॉक में कार्य नहीं करते हैं।
   (द) लोकल चर एक ही नाम से अलग–अलग फलनों में घोषित किये जा सकते हैं।
4. फलन int add(int x) का आवाहन् करने के लिए निम्न में से कौन सा कथन सत्य है :
   (अ) add(); (ब) add(x); (स) add(int x); (द) int add (int x);
5. फलन को कितने प्रकार से कॉल कर सकते हैं :
   (अ) 2 (ब) 1 (स) 3 (द) 4
6. फलन कितने प्रकार के होते हैं :
   (अ) 1 (ब) 2 (स) 4 (द) 3
7. संकेतक चर में स्टोर होता है।
   (अ) पूर्णांक मान (ब) कोई भी मान
   (स) किसी अन्य चर का एड्रस (द) इनमें से कोई नहीं
8. int B = 10;
   int *A=&B; printf("%d", *A) प्रिन्ट करेगा।
   (अ) 10 (ब) A चर का एड्रेस
   (स) B चर का एड्रेस (द) प्रोग्राम के अनुसार प्रिन्ट करेगा।

9. int *A
A= (int*)malloc(sizeof(int)*10);
printf("%d",A); प्रिन्ट करेगा।
(अ) A चर का एड्रेस (ब) एरे के प्रथम मान एड्रेस
(स) एरे का प्रथम मान (द) कोई मान प्रिन्ट नहीं करेगा।

10. संरचना का सदस्य हो सकता है।
(अ) पॉइन्टर चर (ब) पूर्णांक चर (स) फ्लोटिंग चर (द) उपरोक्त सभी

11. किसी संरचना के सदस्यों को एक्सेस करने हेतु कौन से चिह्न को उपयोग में लेते हैं।
(अ) . (dot) (ब) * (स) ® (द) &

12. a = 1011
b = 1111 तथा
x = a & b तो x का मान होगा –
(अ) 10 (ब) 11 (स) 12 (द) 13

13. यदि किसी संघ (Union) में int, float तथा double प्रकार के आँकड़े हैं तो उपरोक्त संघ के लिए कितनी मैमोरी एलोकेट होगी –
(अ) 2 bytes (ब) 4 bytes (स) 10 bytes (द) 8 bytes

अतिलघूत्तरात्मक प्रश्न

1. return कथन कहां लिखा जाता है ?
2. प्राचल कितने प्रकार के होते हैं ?
3. लोकल चर किन्हें कहते हैं ?
4. '\0' संप्रतीक किस हेडर फाइल में सम्मिलित होता है ?
5. एक विमा वाली एरे को किस प्रकार घोषित करते हैं ?.
6. किसी पॉइन्टर चर को किस प्रकार घोषित करते हैं ?
7. int A;
A के पते को किस प्रकार प्रिन्ट करते हैं ?
8. free() फलन के द्वारा मैमोरी को कैसे छोड़ते हैं ?
9. संरचना में सदस्यों को किस प्रकार घोषित करते है ?
10. संरचना के सदस्यों का प्रारम्भिकरण के द्वारा मान कैसे लिखते हैं ?
11. malloc() फलन किस हैडर फाइल में उपलब्ध है ?
12. संरचना में सदस्यां को किस प्रकार घोषित करते हैं ?
13. संरचना के सदस्यों को प्रारम्भिकरण के द्वारा मान कैसे लिखते हैं ?
14. हम किसी चर के लिए बिट्स किसके द्वारा एलोकेट करवाते हैं ?
15. नया डाटा टाइप बनाने के लिए कौनसा की–वर्ड काम में लेते हैं ?

लघूत्तरात्मक प्रश्न

1. लोकल चर किन्हें कहते हैं ?
2. दो विमाओं वाली एरे की घोषणा का एक उदाहरण लिखो।
3. फलन को किस प्रकार घोषित करते है?

4. फलन कितने प्रकार के होते हैं ?
5. एरे की घोषणा किस प्रकार करते हैं।
6. पुनरावर्तन की दो मुख्य कण्डीशन बताइए।
7. पॉइन्टर चर क्या हैं ?
8. int *A={10, 20, 30};
   अगर A का बेस एड्रस 2000 है तो सभी मानों के एड्रस लिखिए ?
9. एक पॉइन्टर चर द्वारा एरे को किस प्रकार घोषित करते हैं ?
10. संरचना क्या है ?समझाइये।
11. छात्र संरचना जिसके सदस्य name, address, roll_No. है लिखिए।
12. बिट् फिल्ड्स के द्वारा मेल/फिमेल का मान 0 या 1 स्टोर करने के लिए किस प्रकार घोषित करेंगे।
13. निरूपित आँकड़ों का एक उदाहरण लिखिए।

निबन्धात्मक प्रश्न

1. पॉइन्टर के लाभ लिखिये।
2. पॉइन्टर के द्वारा एरे को किस प्रकार घोषित करते हैं ?उदाहरण सहित समझाइए।
3. संरचना का उपयोग करते हुए 10 नाम, पते तथा फोन नंबर भंडारित करो। इन्हें नाम के वर्णमाला क्रम में व्यवस्थित करो। इन्हें नाम के वर्णमाला क्रम में व्यवस्थित करो। व्यवस्थित क्रम को प्रिन्ट करने हेतु प्रोग्राम लिखिए।
4. एक स्ट्रिंग की रीड़ करो तथा उसमें उपयोग वर्णमाला के अक्षरों की आवशत्ति (frequency) ज्ञात करने हेतु 'सी' भाषा का प्रोग्राम लिखिए।
5. फलन के लिए संकेतक किस प्रकार घोषित होते हैं ? उदाहरण सहित समझाइए।
6. निम्न सदस्यों के लिए बिट् फिल्ड्स दर्शाते हुए संरचना लिखिए।
   i. be male or female
   ii. have one of the eight different hobbies
   iii. be single, married, divorced or widowed
7. निरूपित आँकड़ा क्या होता है ?12 माह को हम निरूपित आँकड़े कि तरह किस प्रकार लिखेंगे?

उत्तरमाला

1.	अ	2.	स	3.	ब	4.	स	5.	अ
6.	ब	7.	ब	8.	अ	9.	ब	10.	द
11.	अ	12.	ब	13.	द				

पाठ–3

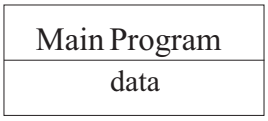
ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग
(Object Oriented Programming)

प्रोग्राम एवं प्रोग्रामिंग

हम जानते हैं कि किसी निश्चित कार्य को पूरा करने हेतु लिखे गए अनुदेशों (instruction) के समूह को प्रोग्राम कहते हैं और कम्प्यूटर में एक प्रचालन (operation) को एक अनुदेश कहते हैं। सामान्यतया, एक प्रोग्राम को एक तार्किक प्रक्रिया के रूप में देखा जा सकता है जिसमें डेटा दिया जाता है (input), उस डेटा को संसाधित (process) किया जाता है और फिर, परिणाम के रूप में हमें डेटा (output) प्राप्त होता है। किसी समस्या का समाधान कम्प्यूटर द्वारा करने के लिए उसे, कम्प्यूटर समझ सके ऐसे रूप में बदलना होता है, इस प्रक्रिया को प्रोग्रामिंग कहते हैं। ऐसा करने के लिए हम कुछ भाषाओं का प्रयोग करते हैं। जैसे फोर्ट्रान, पास्कल, कोबोल इत्यादि। ये भाषाएँ प्रक्रियात्मक भाषाएँ हैं । 1960–70 के दशक तक ये भाषाएँ बहुत प्रभावी रूप से काम ली गयीं, परन्तु इस समय तक प्रोग्राम बड़े एवं जटिल होने लगे थे।

3.1 असंरचित प्रोग्रामिंग

प्रोग्रामिंग सीखने के समय प्रारम्भिक स्तर पर छोटे और साधारण प्रोग्राम हम बनाते हैं, उनमें एक ही मुख्य प्रोग्राम होता है। मुख्य प्रोग्राम का अर्थ है – आदेशों का एक समूह है जिसमें डेटा पूरे प्रोग्राम के दौरान उपलब्ध रहता है और उसे बदला जा सकता है।



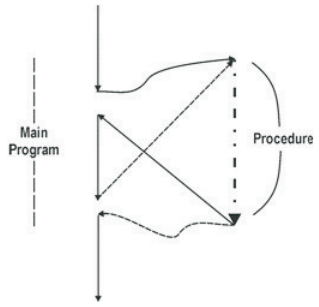
चित्र 3.1: असंरचित प्रोग्रामिंग – प्रोग्राम डेटा को सीधे ही संसाधित करता है

यदि प्रोग्राम का आकार बढ़े तो इस प्रकार की प्रोग्रामिंग में बहुत सारी समस्याएं हो जाती हैं। उदाहरण के लिए, यदि एक ही प्रकार के आदेशों के समूह को हमें पुनः काम लेना हो तो उस समूह को हमें दुबारा लिखना पड़ेगा, इससे अनावश्यक रूप से प्रोग्राम का आकार बढ़ता है, उसे सुधारने में कठिनाइयाँ आना स्वाभाविक है। इस समस्या का समाधान यह आया कि इस समूह को प्रोग्राम से निकाल कर एक नया नाम दे दिया जाए, और उस नाम का सन्दर्भ प्रोग्राम में उपयोग किया जाए। इस प्रकार प्रक्रियात्मक प्रोग्रामिंग का आधार बना। इसमें एक मुख्य प्रोग्राम में अन्य प्रोग्रामों का उपयोग किया जा सकता है। इन प्रोग्रामों को सहायक प्रक्रियाओं (Produdure) के रूप में समझा जा सकता है, जिन्हें हम प्रोसीजर कहते हैं।

3.2 प्रक्रियात्मक प्रोग्रामिंग (**Procedural Programming**)

प्रक्रियात्मक प्रोग्रामिंग की सहायता से प्रोग्राम का आकार छोटा किया जा सकता है, उसमें त्रुटियों की सम्भावना कम होती है, और उसका रख–रखाव भी आसान हो जाता है। मुख्य प्रोग्राम में जब किसी प्रोसीजर को कॉल किया जाता है तो प्रोग्राम का नियंत्रण उस प्रोसीजर पर जाता है, कंप्यूटर उस

निर्देश–समूह को संचालित करता है, और समाप्ति के पश्चात नियंत्रण को मुख्य प्रोग्राम में वापिस जिस अनुदेश से भेजा था, उसके ठीक अगले अनुदेश पर भेज देता है।

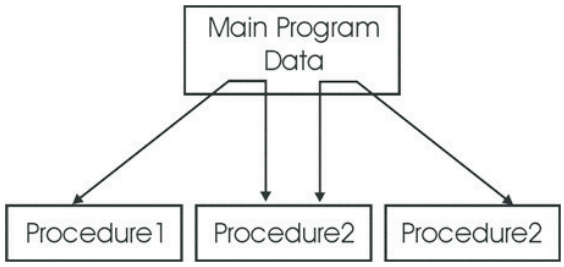


चित्र 3.2: प्रोसीजर का निष्पादन

प्रोग्राम का नियंत्रण,जिस अनुदेश के द्वारा प्रोसीजर के निष्पादन के लिए भेजा गया था, निष्पादन के पश्चात उसके अगले अनुदेश पर आता है ।

इस प्रकार की प्रोग्रामिंग में, प्रोसीजर की सुविधा हाने से हम प्रोग्राम का आकार कम कर पाते हैं, साथ ही उसे त्रुटि–रहित भी कर पाते हैं।

अब हम ऐसा भी मान सकते हैं कि एक प्रोग्रम बहुत से प्रोसीजर की श्रृंखला है। मुख्य प्रोग्राम से प्रत्येक प्रोसीजर को कॉल किया जाता है, डेटा को संसाधित किया जाता है और परिणाम मुख्य प्रोग्राम को प्रेषित कर दिए जाते हैं। इस प्रकार जब पूरी श्रृंखला सम्पूर्ण हो जाती है तब मुख्य प्रोग्राम से अंतिम परिणाम प्राप्त हो जाता है। मुख्य प्रोग्राम, प्रोसिजरों को डेटा प्राचलों के रूप में देता और लेता है।



चित्र 3.3: प्रक्रियात्मक प्रोग्रामिंग

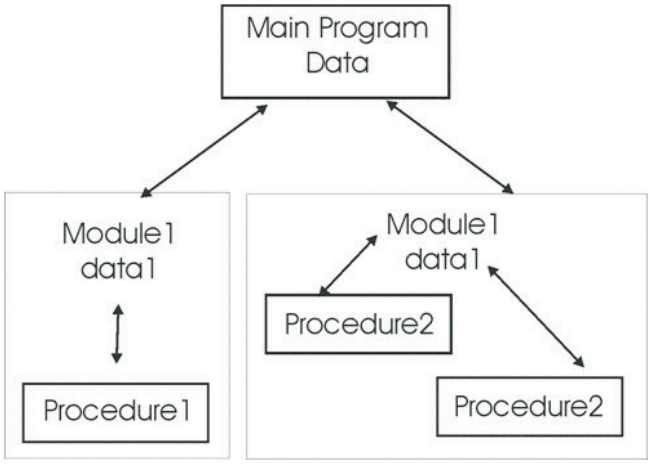
मुख्य प्रोग्राम के नियंत्रण में –प्रोसिजरों का निष्पादन (execution)एवं डेटा का आदान–प्रदान होता है।

इस प्रकार हम ऐसा कह सकते हैं कि मुख्य प्रोग्राम को छोटे–छोटे टुकड़ों में बाँट देते हैं जिन्हें प्रोसीजर कहते हैं। सामान्य प्रकार के प्रोसीजर जो अन्य प्रोग्रामों में काम आ सकें और इसी प्रकार प्रयोक्ता स्वयं द्वारा बनाए गए प्रोसीजर भी दूसरे प्रोग्राम में काम ले सकें, इसके लिए यह आवश्यक है कि प्रोसीजर अलग से संग्रहीत किये जा सकें।

3.3 मोड्यूलर प्रोग्रामिंग

मोड्यूलर प्रोगामिंग में प्रोसिजरों को सामूहिक रूप से एक मोड्यूल के रूप संग्रहीत किया जाता है। अपनी सुविधा के अनुसार, एक मोड्यूल में समान प्रकार के संसाधन करने वाले प्रोसिजरों को रखा जाता

है। संग्रहीत प्रोसिजरों को मुख्य प्रोग्राम में सीधे काम लिया जा सकता है। मुख्य प्रोग्राम बहुत से छोटे टुकड़ों में बंट जाता है और प्रोसिजरों के माध्यम से डेटा संसाधन होता है।



चित्र 3.4 मोड्यूलर प्रोग्रामिंग

प्रत्येक मोड्यूल का स्वयं का डेटा होता है। जब एक मोड्यूल के प्रोसिजरों का उपयोग किया जाता है तब प्रत्येक मोड्यूल अपनी आतंरिक डेटा व्यवस्था को बनाए रखता है। परन्तु एक मोड्यूल एक प्रोग्राम में एक समय में एक बार ही हो सकता है।

प्रक्रियात्मक भाषाओं के लाभ

- सामान्य प्रोग्रामिंग के लिए बहुत अच्छी होती है।
- बहुत सी समस्याओं के लिए कोड उपलब्ध हो सकता है, तो हो सकता है कि पुनः प्रोग्रामिंग (reprogramming) की आवश्यकता न पड़े।
- सीपीयू पर प्रोग्राम चलाना हो उसकी विशेष जानकारी होना आवश्यक नहीं है– जैसा कि मशीन भाषा में आवश्यक होता है।
- दूसरे कम्पाइलर की सहायता से किसी अन्य सीपीयू पर इसी प्रोग्राम को चलाया जा सकता है।

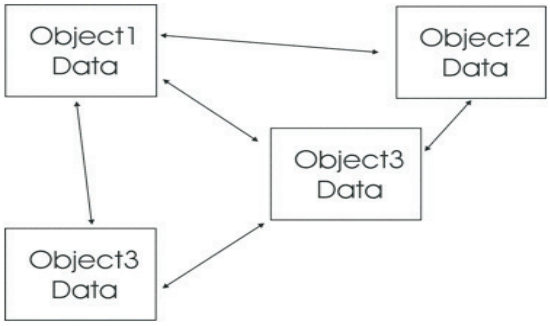
प्रक्रियात्मक भाषाओं की सीमायें

- बहुत सारी भाषाओं की उपलब्धता होने से प्रोग्रामर को किसी एक भाषा में बंध कर रहना होता है, और कोबोल के लिए अलग–अलग प्रकार की विशेषज्ञता की आवश्यकता होती है।
- भाषा–विशेष के प्रोग्रामिंग अनुदेशों की सूक्ष्म जानकारी और ज्ञान आवश्यक होता है।
- कृत्रिम बुद्धिमत्ता (Artificial Intelligence) की आवश्यकता वाली समस्याओं के लिए जहाँ तर्क अस्पष्ट हों (fuzzy logic), प्रक्रियात्मक भाषाएँ अनुकूल नहीं हैं।

3.5 ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग (**Object Oriented Programming-OOP**) की अवधारणा

ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग (Object Oriented Programming), प्रक्रियात्मक प्रोग्रामिंग (procedural programming) से भिन्न है। ऑब्जेक्ट पर आधारित भाषाओं में प्रोग्राम क्लास और ऑब्जेक्ट

पर आधारित होते हैं। इनका उपयोग–प्रयोग हम करते हैं, इनमें लिखे मेथड्स के माध्यम से। OOP के आने से पहले, प्रक्रियात्मक भाषाओं का प्रयोग होता था। इन भाषाओं में प्रोग्रामिंग अनुदेश और डेटा अलग–अलग होते हैं। इसी कारण, प्रोग्रामों के गलत होने की सम्भावना अधिक रहती है। यह समस्या और गंभीर हो जाती है जब प्रोग्राम की लम्बाई अधिक हो, अर्थात उसमें अनुदेशों की संख्या ज्यादा हो, अथवा जिस समस्या का समाधान करने हेतु प्रोग्राम लिखा गया गया है, वह समस्या स्वयं में ही जटिल हो। OOP के मूल में सोच यह है कि एक प्रोग्राम के प्रोसीजर, फंक्शन या उप–प्रोग्राम और उनमें काम आने वाले डेटा एक साथ होंगे तो उनको काम में लेना सरल हो जाएगा।



चित्र 3.5: ऑब्जेक्ट ओरियन्टेड प्रोग्रामिंग

3.5.1 ऑब्जेक्ट (**Object**)

OOP की अवधारणा में ऑब्जेक्ट एक वस्तु के रूप में माना जाता है। वह सब कुछ जो अस्तित्व में है और जिसे छू कर, देख कर अथवा तार्किक रूप से अनुभव किया जा सकता है वह सब वस्तु (object) है, और संभव है कि वह वस्तु अन्य छोटी–छोटी वस्तुओं का समूह हो।

एक ऑब्जेक्ट बहुत सी गतिविधियाँ कर सकती है। ये गतिविधियाँ ऑब्जेक्ट के व्यवहार को परिभाषित करता हैं। उदाहरण के लिए, दुपहिया वाहन एक ऑब्जेक्ट है। बाइसिकल एक दुपहिया है, और यह दुपहिया कई अन्य वस्तुओं से मिल कर बना है जैसे पहिये, सीट, हैंडल, पैडल इत्यादि। हम देख सकते हैं कि बहुत से ऐसे दुपहिया हैं जिनकी विशेषताएँ मिलती हैं, कार्यप्रणाली मिलती है, जैसे, सभी दुपहियों में दो पहिये होंगे ही। परन्तु सभी में पैडल हों, आवश्यक नहीं है। एक अन्य उदाहरण, यदि हम मानें कि "Student" एक ऑब्जेक्ट है, इसके माध्यम से एक विद्यार्थी का नाम, पता, कक्षा जैसी सूचनाओं को सम्बंधित किया जा सकता है। इसी प्रकार (Student Status) एक अन्य ऑब्जेक्ट है जिससे उपस्थिति, विषय, परीक्षा में प्राप्तांक, परीक्षा परिणाम इत्यादि के बारे में जाना जा सकता है।

एक ऑब्जेक्ट में आदेश और डेटा संग्रहीत किये जा सकते हैं, और वे सब विशेष संसाधन के लिए निर्मित किये जाते हैं। ऑब्जेक्ट का उपयोग उसमें उपलब्ध आदेशों के माध्यम से किया जा सकता है। एक ऑब्जेक्ट किसी अन्य ऑब्जेक्ट को सन्देश के माध्यम से कोई क्रिया करने के लिए भी कह सकता है। डेटा अथवा आदेश के माध्यम से ऑब्जेक्ट क्रिया करने के लिए अनुरोध करने वाला ऑब्जेक्ट, प्रेषक ऑब्जेक्ट (Sender Object) कहलाता है तथा जो ऑब्जेक्ट प्राप्त करता है वह प्रापक ऑब्जेक्ट (Receiving Object) कहलाता है।

चित्र 3.6 ऑब्जेक्ट संप्रेषण

क्रियान्वयन का नियंत्रण अब प्रापक ऑब्जेक्ट के पास आ जाता है और वह तब तक रहता है जब तक वह आदेश पूरा नहीं हो जाता है। आदेश के निष्पादन के पश्चात, नियंत्रण पुनः प्रेषक ऑब्जेक्ट के पास चला जाता है। उदाहरण के लिए, यदि हम दो ऑब्जेक्ट इस प्रकार मानें – पहला exam, जिसमे कि विद्यार्थी का नाम जानने के लिए exam ऑब्जेक्ट, सन्देश के माध्यम से student नामक ऑब्जेक्ट का उपयोग कर सकता है।

प्रेषक ऑब्जेक्ट सन्देश के माध्यम से प्रापक ऑब्जेक्ट को सूचना भी भेज सकता है, इसे प्राचल (argument) कहा जाता है। प्रापक ऑब्जेक्ट, प्रेषक ऑब्जेक्ट को सदैव एक मान लौटाता है। इस मान का उपयोग प्रेषक ऑब्जेक्ट आगे के संसाधन में करता है। उदाहरण के लिए, studentstatus ऑब्जेक्ट विद्यार्थी की उपस्थिति के मान को बदलना चाहता है, इसके लिए वह नया मान एक सन्देश में प्राचल के रूप में भेजता है। इस उदाहरण में जो मान प्रापक ऑब्जेक्ट लौटाएगा, उसकी उपेक्षा हो जाती है।

3.5.2 मेथड (Method)

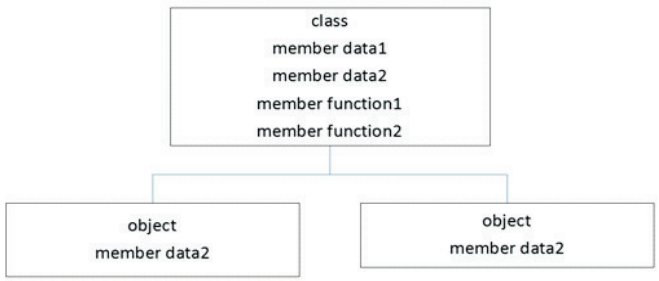
प्रेषक ऑब्जेक्ट से प्राप्त होने वाले संदेशों को प्रापक ऑब्जेक्ट किस प्रकार समझता है?किस प्रकार वे सन्देश, और उनके साथ प्राप्त प्राचल संसाधित किये जाते हैं?जब एक ऑब्जेक्ट कोई सन्देश प्राप्त करता है तब उस सन्देश के साथ वाला प्रोग्राम कोड निष्पादित होता है। दूसरे शब्दों में कहें तो ये सन्देश निर्धारित करते हैं कि ऑब्जेक्ट का व्यवहार किस प्रकार का रहेगा और ऑब्जेक्ट में लिखा गया प्रोग्राम कोड निर्धारित करता है कि ऑब्जेक्ट क्या करेगा। सन्देश के साथ जुड़े हुए कोड को मेथड (method) कहते हैं।

जब ऑब्जेक्ट को कोई सन्देश प्राप्त होता है, उस सन्देश में मेथड का नाम भी होता है, इस से निर्धारित होता है अब कौनसा प्रोग्राम कोड निष्पादित होगा। और तत्पश्चात निष्पादन हेतु नियंत्रण उस कोड को दे दिया जाता है। वह कोड निष्पादित होता है और उसका जो परिणाम है वह आगे भेज दिया जाता है।

प्रक्रियात्मक भाषाओं के सन्दर्भ में जिन्हें प्रोसीजर कहा जाता है, मेथड उसी प्रकार का निर्देशों का समूह है। मेथड का नाम, प्रोसीजर का नाम और मेथड में लिखा गया प्रोग्राम कोड, प्रोसीजर में लिखा गया कोड है। एक ऑब्जेक्ट को सन्देश भेजना, एक प्रोसीजर को काम में लेने जैसा ही है।

3.5.3 क्लास (Class)

ऑब्जेक्ट, प्रोग्राम के निष्पादन के समय काम आता है। इसकी परिभाषा क्लास के माध्यम से दी जाती है। क्लास एक डेटा टाइप है और ऑब्जेक्ट एक वेरियेबल। क्लास को परिभाषित करने के पश्चात हम आवश्यतानुसार उसके ऑब्जेक्ट वेरियेबल बना सकते हैं। जो डेटा और प्रोग्राम कोड हम परिभाषा में लिखते हैं, वह प्रोग्रामिंग की आवश्यकता के अनुरूप होते हैं। उन्हें सदस्य डेटा और सदस्य फंक्शन कहा जाता है।



चित्र 3.7 क्लास

3.6 आब्जेक्ट आरियेन्टेड प्रोग्रामिंग की विशेषताएं

पुनः प्रयोज्यता (**Reusability**)

किसी समस्या के हल के लिए लिखा गया प्रोग्राम, मात्र उसी समस्या का हल हो सकता है। इसका अर्थ यह हुआ कि मिलती–जुलती समस्याओं के लिए भी हमें नए प्रोग्राम लिखने पड़ेंगे। ऐसा इसलिए हुआ या होता है क्योंकि परम्परागत प्रोग्रामिंग में डेटा और प्रोग्राम अलग अलग होते हैं। नयी समस्या के लिए नए प्रकार के डेटा भी हो सकते हैं, और कुछ थोड़ा–बहुत बदलाव समस्या के आधार पर तर्कों में भी हो सकता है। परन्तु पुराना प्रोग्राम इसके लिए काम नहीं आ सकता। ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग में यह व्यवस्था है कि उपलब्ध समाधानों में बदलाव करते हुए नयी समस्याओं के लिए समाधान लिखे जा सकते हैं, अर्थात नयी समस्या के लिए प्रोग्राम करने के लिए हमें प्रारम्भ से पुनः प्रयास करने की आवश्यकता नहीं है, हम पहले से बने हुए प्रोग्राम में आवश्यक्तानुसार परिवर्तन करके वही समाधान प्राप्त कर सकते हैं। पुनः प्रयोज्यता ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग की संभवतः सबसे महत्त्वपूर्ण विशेषताओं में से एक है।

एनकैप्सुलेशन (**Encapsulation**) एवं एबस्ट्रेक्शन (**Abstration**)

प्रक्रियात्मक भाषाओं में डेटा अलग से परिभाषित किया जाता है और उसे काम लेने वाले फंक्शन अलग से। परन्तु, ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग में, जैसा कि हमने जाना क्लास को परिभाषित करते समय हम डेटा और फंक्शन को एक साथ लिखते हैं। इसे सम्पुटीकरण (encapsulation) कहा जाता हैं। एबस्ट्रेक्शन (Abstraction) का तात्पर्य है कि एक क्लास में जो भी डेटा परिभाषित किया गया है, उसका उपयोग बिना अन्य विवरण और स्पष्टीकरण के उपयोग में लिया जा सकता है। एनकैप्सुलेशन और एबस्ट्रेक्शन का लाभ ये होता है कि डेटा संरचना और संकारकों का उपयोग वैसा ही हो जाए जैसा प्रोग्रामर ने इन्हें परिभाषित करते समय सोचा है। इस प्रकार हम क्लास में दी गयी सूचनाओं को छिपा (information hiding) सकते हैं जिस से प्रोग्रामिंग अधिक सुरक्षित हो जाती है।

बहुरूपता (**Polymorphism**)

एक ऑब्जेक्ट का व्यवहार और कार्यान्वयन अलग–अलग होते हैं। एक ही सन्देश के लिए बहुत से ऑब्जेक्ट काम कर सकते हैं अथवा एक ही ऑब्जेक्ट विभिन्न रूपों में काम लिया जा सकता है। प्रोग्राम के निष्पादन के समय जैसी आवश्यकता होती है ऑब्जेक्ट का रूप वैसा हो सकता है। उदाहरण के लिए, addition एक क्लास है जिसे हम गणित की योग क्रिया (addition operation) के लिए परिभाषित कर रहे हैं। इसमें add नामक एक मेथड है जिस में हम दो इन्टीजर वेरियेबल को जोड़ने का कोड लिखते हैं, इसी क्लास में add नाम से एक और मेथड है जिस में हम दो फ्लोट वेरियेबल को जोड़ने का कोड लिखते हैं। अब यदि प्रोग्राम के निष्पादन के समय add मेथड में दो इन्टीजर वेरियेबल का उपयोग करते हैं तो पहला मेथड काम आयेगा, और यदि दो फ्लोटवेरियेबल काम में लेते हैं तो दूसरा मेथड काम आएगा। यहाँ ध्यान देने वाली बात यही है कि add नाम से दो मेथड होते हुए भी निष्पादन के समय भेजे गए डेटा के आधार पर वांछित मेथड काम आएगा। अर्थात एक ही नाम अलग–अलग रूप में काम आ रहा है। बहुरूपता की व्यवस्था के कारण, प्रेषक ऑब्जेक्ट और प्रापक ऑब्जेक्ट के मध्य संदेशों के आदान–प्रदान संभव हो जाता है।

वंशानुक्रम (**Inheritance**)

वंशानुक्रम (Inheritance), ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग की एक और महत्त्वपूर्ण संकल्पना है। एक क्लास को परिभाषित करने के पश्चात यदि हमें वैसी ही एक और क्लास को और परिभाषित करना हो – एक ऐसी क्लास जिसके गुणधर्म वही हों जो पहली क्लास के हैं और साथ में कुछ और गुणधर्म भी हम जोड़ना चाहें – वंशानुक्रम की व्यवस्था से हम ऐसा कर पाएंगे। ऐसी व्यवस्था में हम पुनः प्रयोज्यता का भी उपयोग कर रहे हैं, इस कारण से दुबारा से वही सारा काम करने की आवश्यकता नहीं है जो पहली क्लास को परिभाषित करते समय हमने किया था। इससे समय भी बचता है और प्रोग्राम के रख रखाव में भी सुविधा

होती है।

किसी क्लास के गुणधर्म लेने वाली क्लास को सब क्लास (sub class) या डिराईव्ड क्लास (derived class) कहा जाता है और जिस क्लास से गुणधर्म आगे लिए जाते हैं उसे सुपर क्लास (Super Class) अथवा बेस क्लास (Base Class) कहा जाता है।

3.7 समस्या समाधान का ऑब्जेक्ट ओरिएंटेड दृष्टिकोण (Object Oriented Problem Solving Approach)

अपने दैनंदिन जीवन में हम जिस प्रकार अपने कार्य करते हैं अथवा समस्याओं के समाधान ढूंढते हैं, उसी प्रकार का दृष्टिकोण ही ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग उपयोग आता है। समस्या के समाधान में सहायक होने वाले ऑब्जेक्ट्स की पहचान करना और उन्हें निश्चित क्रम में उपयोग में लेना, अपनी समस्याओं के समाधान के लिए साधारणतया हम ऐसा ही करते हैं। समस्या में आने वाले ऑब्जेक्ट्स के बारे में सोचते हैं और उनका उपयोग किस प्रकार करेंगे जिससे कि समस्या का समाधान हो सके। दूसरे शब्दों में, ऑब्जेक्ट–ओरिएंटेड समस्या के समाधान में हम ऐसे ऑब्जेक्ट्स बनाते हैं जिनसे उस समस्या का समाधान हो सके। ऑब्जेक्ट को भेजे गए सन्देश के आधार पर विभिन्न संक्रियाएं होती है जिनसे समस्या का समाधान हो जाता है।

ऑब्जेक्ट–ओरिएंटेड समस्या के समाधान की प्रक्रिया को चार चरणों में विभाजित किया जा सकता है।

1. समस्या को पहचानना (Identify the problem)
2. समाधान हेतु आवश्यक ऑब्जेक्ट्स को पहचानना (Identify the objects needed for the solution)
3. ऑब्जेक्ट्स को भेजे जाने वाले संदेशों को पहचानना (Identify messages to be sent to the objects)
4. उन संदेशों के क्रम का निर्माण करना जिस से समस्या का समाधान प्राप्त हो (Create a sequence of messages to the objects that solve the problem)

3.8 ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग के लाभ

- प्रोग्रामिंग करना आसान है। एक क्लास को पूरी तरह से जांचने के बाद काम में लेने की सुविधा के कारण त्रुटि होने की सम्भावना कम हो जाती है।
- क्लास को एक "ब्लैक बॉक्स" के रूप में माना जा सकता है, उसकी आतंरिक संसाधन व्यवस्था की जानकारी के बिना, उसमें उपलब्ध मेथड्स को उपयोग किया जा सकता है ।
- एक ही प्रकार के प्रोग्राम को लिखने एवं जांचने का अनावश्यक प्रयास बच जाता है। अपनी आवश्यकता के अनुरूप क्लास को सीधे काम में लिया जा सकता है। दूसरे अर्थों में कहें तो प्रोग्राम कोड का पुनः उपयोग किया जा सकता है।
- कोड को किसी अन्य कम्पाइलर के माध्यम से दूसरे सीपीयू के अनुरूप बनाया जा सकता है अर्थात कोड पोर्टेबिलिटी की सुविधा मिलती है।

3.9 C++ के मूल तत्त्व

C++ ऑब्जेक्ट की अवधारणा का उपयोग करने हेतु क्लास (Class) उपयोग करता है। क्लास को बनाने के लिए class कीवर्ड का उपयोग किया जाता है। क्लास एक प्रयोक्ता परिभाषित डेटा टाइप है जिसका उपयोग करते हुए प्रोग्रामर अपनी प्रोग्रामिंग के लिए उपयोगी रियल ऑब्जेक्ट कंप्यूटर में परिभाषित

कर सकता है और फिर उस पर आधारित संसाधन से प्रोग्रामिंग कर सकता है। परिभाषित करते समय हम ऑब्जेक्ट नहीं बनाते हैं, अपितु उसका एक मॉडल बना देते हैं, जिसे ही क्लास कहते हैं। तत्पश्चात आवश्यकतानुसार इस क्लास का उपयोग करते हुए ऑब्जेक्ट बनाते हैं। उदाहरण के लिए, हम student नामक क्लास परिभाषित करते हैं, और प्रोग्राम में आवश्यकतानुसार, विभिन्न कक्षाओं के लिए विभिन्न ऑब्जेक्ट्स बना सकते हैं।

प्रत्येक क्लास में कुछ विशेषताओं को परिभाषित किया जाता है, ये वे विशेषताएं हैं जो एक ऑब्जेक्ट में हम चाहते हैं, इन्हें ऑब्जेक्ट का गुणधर्म (attributes) कहा जाता है। आवश्यक नहीं है कि सारे गुणधर्म प्रत्येक ऑब्जेक्ट में हों, इस व्यवस्था के लिए हम एक और नयी क्लास बना सकते हैं जिसमें सारे गुणधर्म पहले से परिभाषित क्लास के होंगे, परन्तु साथ ही में कुछ नए गुणधर्म भी होंगे। यह सुविधा पदानुक्रम (inheritance) के माध्यम से हमें उपलब्ध होती है।

उदाहरण

```
class class_name
{
private:
 Data_Members;
 Member_Functions;

public:
 Data_Members;
 Member_Functions;
};
```

क्लास को उपयोग में लेने के लिए ऑब्जेक्ट्स बनाते हैं, जिन्हें हम क्लास का उदाहरण (instance) कहते हैं। क्लास एक परिभाषा है और ऑब्जेक्ट एक चर। क्लास के दूसरे हिस्से में इन गुणधर्मों को प्रयोग, उपयोग में लेने हेतु कुछ फंक्शन लिखे जाते हैं जिन्हें मेम्बर फंक्शन कहते हैं।

प्रत्येक प्रोग्रामिंग भाषा कुछ चिह्नों, विशिष्ट शब्दों और नियमों का एक समूह होती है। इन सबके उपयोग से एक प्रोग्राम लिखा जाता है। कुछ ऐसे तत्व होते हैं जो प्रत्येक भाषा में पाए जाते हैं। C++ में उपलब्ध कुछ तत्वों की चर्चा अब हम करेंगे।

C++ का अक्षर समुच्चय (Character Set)

Letters	A-Z, a-z
digits	0-9
Special Characters	Space + - * / ^ \ () [] {} = ! = >< ' " $, ; : % ! & ? # < = (विशेष अक्षर) >= @
Formatting characters	backspace, horizontal tab, vertical tab, form feed and carriage return

3.10 टोकन (Tokens)

एक टोकन अक्षरों का समूह होता है। प्रोग्रामर इनका उपयोग करते हुए प्रोग्राम लिखता है। C++

भाषा में उपलब्ध टोकन हैं : Keywords] Identifies] Literal] Integer Constants.

कुँजी शब्द (**Keywords**) : C++ भाषा में कुछ शब्द हैं जिन्हें पहले से ही भाषा में परिभाषित किया गया है। ये सुरक्षित शब्द हैं जिनका अर्थ पहले से ही कम्पाइलर को ज्ञात है।

परिज्ञापक (**Identifiers**) : C++ में विभिन्न डेटा तत्वों (data elements) के लिए प्रतीकात्मक नामों (Symbolic Name) का उपयोग करने की सुविधा प्रोग्रामर को उपलब्ध है। प्रतीकात्मक नाम को सामान्यतया परिज्ञापक कहा जाता है। C++ के अक्षर समुच्चय से अक्षर लेकर ये नाम बनाए जाते हैं। नाम बनाने के लिए नियम इस प्रकार हैं :

- एक परिज्ञापक के नाम में अल्फाबेट्स (A-Z, a-z), अंक (0–9), और/अथवा अंडर स्कोर (_) हो सकता है।
- प्रथमाक्षर संख्या नहीं हो सकता।
- यह सुरक्षित शब्द नहीं होना चाहिए।

अचर (**Constant**)

वे डेटा तत्व जिनका मान पूरे प्रोग्राम में कभी नहीं बदला जा सकता, स्थिरांक अथवा अचर कहलाते हैं। इंटीजर कांस्टेंट : इंटीजर कांस्टेंट अर्थात पूर्ण संख्याएं जिनमें कोई भिन्नात्मक भाग ना हो। C++ तीन प्रकार के इंटीजर कांस्टेंट की व्यवस्था प्रदान करता है।

- डेसीमल इंटीजर कांस्टेंट : संख्याएं, प्रथम अंक (0) शून्य नहीं हो सकता। जैसे 78, –168, + 4
- ऑक्टल इंटीजर कांस्टेंट : संख्याएं, प्रथम अंक (0)शून्य होता है। जैसे 014
- हेक्साडेसीमल इंटीजर कांस्टेंट : संख्याएं, प्रथम दो अक्षर (0x) अथवा (0X) होते हैं। जैसे 0X24

कैरेक्टर कांस्टेंट : एक अथवा अधिक करैक्टर जो कि एकल उद्धरण चिह्न में लिखे गए हों, जैसे 'I', '4' इत्यादि। जिन कैरेक्टर्स को कीबोर्ड द्वारा सीधा प्रिंट नहीं किया जा सकता, जैसे टैब, बेक स्पेस इत्यादि, C++ में इस प्रकार के कैरेक्टर्स, स्केप सीक्वेंस की सहायता से प्रदर्शित किये जा सकते हैं।

फ्लोटिंग कांस्टेंट : भिन्नात्मक संख्याएं। इन्हें भिन्न रूप में भी लिखा जा सकता है अथवा घातांकीय रूप में। जैसे 3.0, –0.342।

स्ट्रिंग कांस्टेंट : दोहरे उद्धरण चिह्न (Double Quation Mark) में लिखी गयी अक्षरों की श्रृंखला को स्ट्रिंग कांस्टेंट कहते हैं। स्ट्रिंग के अंत में '\0' जोड़ा जाता है, जो स्ट्रिंग के अंत का द्योतक है। जैसे, "TECHNOLOGY" शब्द को स्मृति में "TECHNOLOGY\0" से संग्रहीत किया जाता है और इसका आकार 12 कैरेक्टर्स का होगा।

डेटा प्रकार (**Data Types**)

मूल डेटा प्रकार (**Basic Data Type**)

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float

Double floating point	double
Valueless	void
Wide character	wchar_t

निम्नांकित डेटा–प्रकार परिवर्तकों (data type modifiers) के उपयोग से कई मूल डेटा प्रकारों में परिवर्तन किया जा सकता है।

- signed
- unsigned
- short
- long

डेटा का प्रकार और कंप्यूटर की मैमोरी में डेटा संग्रहीत करने के लिए कितनी मेमोरी की आवश्यकता होगी, तथा न्यूनतम व अधिकतम मान क्या हो सकते हैं, इसे निम्नांकित तालिका में दर्शाया गया है

Type	Typical Bit Width	Typical Range
char	1byte	-128 to +127
unsigned char	1byte	0 to 255
int	4bytes	-2147483648 to 2147483647
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
long int	4bytes	-2.147,483,648 to 2,147,483,647
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3-4e +/- 38 (~7 digits)
double	8bytes	+/- 1-7e +/- 308 (~15 digits)
long double	8bytes	+/- 1-7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

आप जिस कम्पाइलर और कंप्यूटर का उपयोग कर रहे हैं उसके कारण डेटा प्रकारों का कंप्यूटर की मेमोरी में आकार, यहाँ दी गयी सूची से भिन्न हो सकते हैं।

इनपुट–आउटपुट

C++ की स्टैण्डर्ड लाइब्रेरी में एक हैडर फाइल है – iostream.h, जिसका उपयोग डेटा को कीबोर्ड से पढने और स्क्रीन पर प्रदर्शित करने क लिए किया जा सकता है।

निम्नांकित C++ स्ट्रीम ऑब्जेक्ट का उपयोग इनपुट–आउटपुट के लिए किया जा सकता है।

cout console output

cin console input

cout ऑब्जेक्ट

cout का उपयोग सन्देश को स्क्रीन पर प्रदर्शित करने के लिए << संकारक (operator) के संयोजन के साथ किया जाता है।

cout<< "Hello World"; // स्क्रीन पर Hello world प्रदर्शित करता है

cout<< 250; // 250 की संख्या स्क्रीन पर प्रदर्शित करता है

cout<< sum; // वेरिएबल sum के मान को स्क्रीन पर प्रदर्शित करता है

यदि वेरियेबल और कांस्टेंट के संयोजन को साथ में प्रदर्शित करना हो तो << का उपयोग एक से अधिक बार किया जा सकता है।

cout<< "Area of Crop Field is "<< area<< " square meter" ;

cin ऑब्जेक्ट

कीबोर्ड के माध्यम से प्रयोक्ता द्वारा एक मान को इनपुट करने के लिए cin का उपयोग किया जा सकता है। कीबोर्ड द्वारा प्राप्त मान को मेमोरी में संग्रहीत करने के लिए >> निष्कर्षण (extraction) संकारक का उपयोग किया जाना आवश्यक है।

cin >> marks; // कीबोर्ड से डेटा प्राप्त होगा वह marks चर में संग्रहीत हो जाएगा।

महत्त्वपूर्ण बिन्दु

1. प्रक्रियात्मक प्रोग्रामिंग की सहायता से प्रोग्राम का आकार छोटा किया जा सकता है, उसमें त्रुटियों की सम्भावना कम हो जातीहै।
2. मोड्यूलर प्रोगामिंग में प्रोसिजरों को सामूहिक रूप से एक मोड्यूल के रूप संग्रहीत किया जाता है।
3. प्रक्रियात्मक भाषायें सामान्य प्रोग्रामिंग के लिए बहुत अच्छी है।
4. क्लास एक डेटा टाइप है और ऑब्जेक्ट एक वेरियेबल।
5. नयी समस्या के लिए प्रोग्राम करने के लिए हमें प्रारम्भ से पुनः प्रयास करने की आवश्यकता नहीं है, हम पहले से बने हुए प्रोग्राम में आवश्यकतानुसार परिवर्तन करके वही समाधान प्राप्त कर सकते हैं।
6. क्लास को परिभाषित करते समय हम डेटा और फंक्शन को एक साथ लिखते हैं। इसे सम्पुटीकरण (encapsulation)कहा जाता है।
7. पृथक्करण (abstraction) का तात्पर्य है कि एक क्लास में जो भी डेटा परिभाषित किया गया है, उसका उपयोग बिना अन्य विवरण और स्पष्टीकरण के उपयोग में लिया जा सकता है।
8. प्रोग्राम के निष्पादन के समय जैसी आवश्यकता होती है ऑब्जेक्ट का रूप वैसा हो सकता है।
9. प्रक्रियात्मक भाषाओं के सन्दर्भ में जिन्हें प्रोसीजर कहा जाता है, मेथड उसी प्रकार का निर्देशों का समूह है। मेथड का नाम, प्रोसीजर का नाम और मेथड में लिखा गया प्रोग्राम कोड, प्रोसीजर में लिखा गया कोड है।

अभ्यासार्थ प्रश्न

बहुचयनात्मक प्रश्न

1. बहुत प्रकार के रूप लेने की योग्यता कहलाती है
   (अ) वंशानुक्रम (Inheritance) (ब) बहुरूपता (Polymorphism)
   (स) सदस्य फंक्शन (Member function) (द) संपुटीकरण (Encapsulation)
2. एक ऑब्जेक्ट के गुणधर्मों को बाहर निकालने की प्रक्रिया कहलाती है।
   (अ) बहुरूपता (Polymorphism) (ब) वंशानुक्रम (Inheritance)
   (स) अमूर्तता (Abstraction) (द) डेटा आच्छादन (Data hiding)
3. C++की कौनसी सुविधाएं उसे शक्तिशाली बनाती हैं?
   (अ) सरल कार्यान्वयन (Easy implementation)
   (ब) पुराने कोड का पुनः उपयोग (Reusing old code)
   (स) नया कोड लिखना (Writing new code)
   (द) उपर्युक्त सभी (All of the above)
4. निम्न में से क्या C++ में उपलब्ध OOP की सुविधा नहीं है?
   (अ) संपुटीकरण (Encapsulation) (ब) अमूर्तता (Abstractio)
   (स) बहुरूपता (Polymorphism) (द) अपवाद (Exceptions)
5. ऑब्जेक्ट–ओरिएंटेड कहलाने के लिए प्रोग्रामिंग भाषा में आवश्यक है
   (अ) संपुटीकरण (Encapsulation) (ब) अमूर्तता (Abstraction)
   (स) बहुरूपता (Polymorphism) (द) उपर्युक्त सभी (All of these)
6. कौनसा कथन असत्य है?
   (अ) किसी विशिष्ट कार्य को करने हेतु कोड का एक हिस्सा फंक्शन कहलाता है।
   (ब) फंक्शन की सहायता से बढ़े और जटिल प्रोग्राम को छोटे और सरल टुकडों में बांटा जा सकता है।
   (स) उपलब्ध कोड का प्रयोग करते हुए सामान्य कार्यों को करने हेतु फंक्शन सहायक हैं।
   (द) प्रोग्राम के फंक्शन को मात्र एक ही बार काम में लिया जा सकता है।
7. एक क्लास में परिभाषित फंक्शन के लिए क्या नाम है?
   (अ) सदस्य चर (Member Variable) (ब) सदस्य फंक्शन (Member function)
   (स) क्लास फंक्शन (Class function) (द) क्लासिक फंक्शन (Classic function)
8. OOPS की किस अवधारणा का अर्थ है – मात्र आवश्यक सूचना को ही बाहर दर्शाना
   (अ) संपुटीकरण (Encapsulation) (ब) अमूर्तता (Abstraction)
   (स) डेटा आच्छादन (Data hiding) (द) डेटा बंधन (Data binding)
9. वास्तविक समय के उदाहरणों, जैसे वाहन, कर्मचारी, के बारे में सूचना संग्रह तथा उनको बेहतर ढंग से समझने के लिए कौनसा दृष्टिकोण ज्यादा बेहतर है?
   (अ) प्रक्रियात्मक दृष्टिकोण (ब) ऑब्जेक्ट ओरिएंटेड दृष्टिकोण
   (स) मोड्यूलर दृष्टिकोण (द) उपर्युक्त में से कोई नहीं
10. ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग के लाभ
    (अ) कोड का पुनः उपयोग (ब) डेटा का बेहतर उपयोग
    (स) त्रुटि रहित (द) उपर्युक्त सभी

अतिलघूत्तरात्मक प्रश्न

1. प्रक्रियात्मक प्रोग्रामिंग में असंरचित प्रोग्रामिंग की अपेक्षा त्रुटियों की सम्भावना कम होती है क्योंकि प्रोग्राम का आकार ..... किया जा सकता है। (छोटा)
2. OOP में प्रथम O का तात्पर्य है .....। (ऑब्जेक्ट)
3. वंशानुक्रम में डेटा और ..... को नयी क्लास में लिया जा सकता है। (मेथड)
4. C++ में बसें एक .....होता है। (प्रयोक्ता परिभाषित डेटा प्रकार)
5. पुनः प्रयोज्यता का लाभ है .....। (पहले लिखे गए कोड का उपयोग कर पाना)

लघूत्तरात्मक प्रश्न

1. क्लास और ऑब्जेक्ट में अंतर बताइए।
2. प्रोसीजर और मेथड में अंतर कीजिए।
3. C++ में टोकन क्या होते हैं?
4. C++ में करैक्टर कांस्टेंट क्या होते हैं?
5. डेटा प्रकार परिवर्तक (data type modifier) क्या हैं?

निबन्धात्मक प्रश्न

1. OOP की विशेषताओं का वर्णन कीजिए।
2. C++ के मूल डेटा प्रकार समझाइए।
3. cin एवं cout के प्रयोग को उदाहरण की सहायता से समझाइए।
4. Encapsulation एवं Abstractation में अंतर स्पष्ट कीजिए।

उत्तरमाला

1	ब	2	द	3	द	4	द	5	द
6	द	7	ब	8	स	9	ब	10	द

पाठ–4

कम्प्यूटर नेटवर्किंग (Computer Networking)

4.1 डेटा कम्यूनिकेशन मॉडल (Data Communication Model)

प्रस्तावना : वर्तमान में कम्प्यूटर एक आवश्यक सेवा के रूप में उभर कर आया है इसलिए कई कम्प्यूटर आपस में जुड़कर सूचनाओं का आपस में आदान–प्रदान करते हैं तथा आंकड़ों के आपस में एक कमरे से बाहर आदान प्रदान होने से कम्प्यूटर की शक्ति में अभूतपूर्व वृद्धि हुई है।

नेटवर्किंग : नेटवर्क मूल रूप से उन सभी तत्वों का समूह है जो कम्प्यूटरों को आपस में जोड़ने एवं एक से अधिक कम्प्यूटरों के मध्य आँकड़ों के आदान प्रदान हेतु काम आते हैं। नेटवर्किंग का मूल उद्देश्य उत्पादकता को बढ़ाना होता है।

नेटवर्किंग के गुण : नेटवर्क का निर्माण करते समय निम्न मूलभूत गुणों को ध्यान में रखा जाता है:–

(1) मूल्य : इस श्रेणी में उपयोगी तत्त्वों, उनका समायोजन एवं रख–रखाव आता है।

(2) सुरक्षा : इस श्रेणी में सभी तत्त्वों एवं आँकड़ों का बचाव आता है।

(3) गति : आँकड़ों का आदान प्रदान किस गति से होता है, जिससे नेटवर्क को गुणवत्ता मिलती है, इस श्रेणी में आता है।

(4) संस्थिति : इस श्रेणी में नेटवर्क के सभी तत्वों का आपस में भौतिक स्थिति में जुडाव आता है।

(5) मापनीयता : इस श्रेणी में नेटवर्क किस प्रकार नये रिवाज को स्वीकार करता है उसे ध्यान में रखा जाता है।

आंकड़ों का संचार (डेटा कम्युनिकेशन)

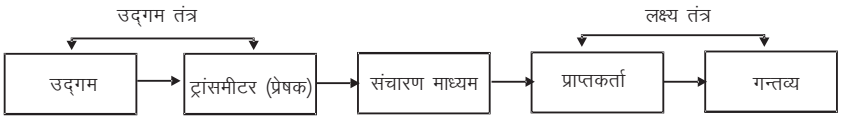
नेटवर्क में काम आने वाले दो उपकरणों के मध्य किसी माध्यम (तार या बेतार) द्वारा आंकड़ों का आदान प्रदान इस श्रेणी में आता है। आंकड़ों के संचार का सीधा सा मतलब उसके आदान–प्रदान से है न कि आंकड़ों की पीढ़ी और उसके परिणाम से आंकड़ों के आदान–प्रदान का प्रभाव तीन बातों पर निर्भर करता है।

(1) सुपुर्दगी – तंत्र सही लक्ष्य को ध्यान में रखकर ही आंकड़ों का आदान–प्रदान करे अर्थात् आंकडे सही, लक्ष्य द्वारा ही प्राप्त किये जायें।

(2) शुद्धता – अगर माध्यम आँकड़ों में कोई परिवर्तन करता है तो यह अनुपयोगी होता है।

(3) सामयिकता – तंत्र आँकड़ों को सही समय पर आदान–प्रदान करता है तो ही इसका महत्व है।

संचार तंत्र



चित्र 4.1 संचार तंत्र

इस चित्र का प्रत्येक खण्ड संचार तंत्र का एक हिस्सा होता है। हर हिस्से का अपना अर्थ एवं अपनी उपयोगिता होती है।

1) उद्‌गम – यह उपकरण आंकड़ों को पैदा करता है।

2) ट्रांसमीटर (प्रेषक) – उद्‌गम उपकरण आँकड़ों को जिस रूप में पैदा करता है उस रूप में माध्यम इनको लक्ष्य की ओर भेजने हेतु स्वीकार नहीं करता अतः ट्रांसमीटर वह उपकरण है जो आंकड़ों को लक्ष्य की ओर भेजने हेतु तैयार करता है।

3) प्रेषक तंत्र – जब दो उपकरण आपस में आँकड़ों का आदान प्रदान करते हैं तो इनके बीच का संप्रेषण तंत्र ही प्रेषक तंत्र कहलाता है।

4) प्राप्तकर्ता – प्राप्तकर्ता वह उपकरण है जो लक्ष्य पर जाने से पहले आँकड़ों को प्रेषक तंत्र से प्राप्त करते है।

5) लक्ष्य – उद्‌गम उपकरण से चलकर आँकड़े प्रेषक तंत्र को पार करते हुए जिस अन्तिम उपकरण पर आकर रूकते हैं वह लक्ष्य कहलाते है।

आंकड़ा संचार तंत्र के अंग – आंकड़ा संचार तंत्र के 5 प्रमुख अंग होते है–

1) संदेश – संदेश वह आंकड़ा या सूचना है जिसे हम दिये गये माध्यम पर प्रेषित करना चाहते है। संदेश विभिन्न रूपों का हो सकता है। जैसे– Text, Image, Video

2) भेजने वाला – यह वह उपकरण है जो कि संदेश भेजता है। यह कोई कम्प्यूटर, टेलीफोन उपकरण या कैमरा इत्यादि हो सकता है।

3) प्राप्त करने वाला – यह वह उपकरण है जो सबसे अंत में संदेश को प्राप्त करता है।

4) माध्यम– उद्‌गम से चलकर संदेश अपने लक्ष्य तक एक रास्ता तय करके पहुंचता है। यही रास्ता जो संदेश को उद्‌गम से लक्ष्य तक पहुँचाता है, माध्यम कहलाता है। माध्यम कोई प्रकाशिक फाइबर, समाक्षीय तार, व्यावर्तित युग्म तार या रेडियो तरंगे हो सकती है।

5) प्रोटोकॉल– प्रोटोकॉल वो नियमों का समूह है जो आँकड़ा संचारण को संचालित करता है। यह दो उपकरणों (उदगम एवं लक्ष्य) के मध्य एक सहमति को निर्सपत करते हैं।

नेटवर्क की परिभाषा

नेटवर्क, कुछ इलैक्ट्रॉनिक उपकरणों का समूह होता है जो कुछ तारों एवं बेतार के माध्यम से आपस में जुड़े होते है। इस माध्यम का काम सूचनाओं को एक छोर से दूसरे छोर तक ले जाना होता है। यह माध्यम इस नेटवर्क के सभी उपभोक्ताओं को इन सभी उपकरणों को आपस में साझा करने की सहमति देते है। नेटवर्क में काम आने वाले इन उपकरणों को नोड़ कहा जाता है। एक नेटवर्क में आवश्यकता अनुसार कितनी भी नोडस हो सकती है। एक नेटवर्क के योग्य एवं प्रभावशाली होने के लिए उसे कई कसौटियों पर खरा उतरना पड़ता है उनमें से मुख्यतया निम्न है–

(1) कार्य–संपादन – कार्य संपदान से तात्पर्य त्रुटि रहित सूचनाओं का आदान–प्रदान होता है। इसमें सूचनाओं का लक्ष्य तक पहुंचने एवं उसके पश्चात लक्ष्य द्वारा लिये गए निर्णय में लगा समय समाहित होता है। किसी नेटवर्क का कार्य संपादन निम्न तथ्यों पर निर्भर करता है–

(अ) उपभोक्ताओं की संख्या – यदि किसी नेटवर्क में ज्यादा उपभोक्ता होते है तो उनका प्रभाव नेटवर्क में सूचनाओं के आदान–प्रदान की गति पर पड़ता है।

(ब) संचार–गति – इससे तात्पर्य आंकड़ों के संचारित होने की गति से है। इसे bps/Kbps/Mbps (bits per second) में नापा जाता है।

(स) माध्यम का प्रकार – उसका तात्पर्य आँकड़ों के आदान–प्रदान में काम में लिए जाने वाले माध्यम से है। आँकड़ों के संचार की गुणवत्ता, काम में लिए जाने वाले माध्यम पर निर्भर

करती है।

(द) उपकरणों का प्रकार– नेटवर्क में काम लिए जाने वाले उपकरण आँकड़ों के आदान–प्रदान की गति एवं क्षमता दोनों को प्रभावित करते हैं। एक उच्च क्षमता वाला कम्प्यूटर काम में लेने पर नेटवर्क कार्य का संपादन प्रभावी रूप से करता है।

(य) सॉफ्टवेयर – नेटवर्क मे काम आने वाले उपकरणों को संचालित करने वाला सॉफ्टवेयर भी नेटवर्क के कार्य संपादन की गुणवत्ता को प्रभावित करता है।

(2) सामंजस्य – सामंजस्य से तात्पर्य लक्ष्य द्वारा पुनः प्रतिक्रिया देने में लिया जाने वाला समय एवं आंकड़ों की शुद्धता का पूर्वानुमान है। उदाहरण के लिए हम यह मान सकते हैं कि एक नेटवर्क का नेटवर्क प्रिन्टर द्वारा किसी पेज को प्रिन्ट देने में लिए गए समय का पूर्वानुमान 3 सेकण्ड है लेकिन वास्तविकता में यह 30 सेकण्ड लेता है मतलब मेरे नेटवर्क में लक्ष्य के साथ सामंजस्य नहीं है।

(3) विश्वसनीयता – विश्वसनीयता नेटवर्क की उपयोगिता का एक मानक है।

(4) पुनः प्राप्ति/बहाली – एक नेटवर्क खराब होने के बाद कितना जल्दी अपनी वास्तविक स्थिति में पुनः आ जाता है, इस पर उसकी उपयोगिता निर्भर करती है।

5) सुरक्षा – हमारे नेटवर्क के उपकरणों, सॉफ्टवेयर एवं आंकड़ों को अनाधिकृत तरीके से उपयोग में लेने से बचाना ही सुरक्षा है। इस श्रेणी में वायर से बचाव भी आता है।

4.2 नेटवर्क टॉपोलॉजी (Network Topologies)

कम्प्यूटरों को एक–दूसरे से जोड़ने की शैली को टॉपोलॉजी कहते है। टॉपोलॉजी की शैली का चुनाव नेटवर्क बनाने की प्रक्रिया में एक अत्यन्त महत्वपूर्ण चरण है। किसी टॉपोलॉजी का चुनाव करने से पूर्व कई पहलुओं पर विचार करना आवश्यक है। इनमें से कुछ पहलुओं का विवरण नीचे प्रस्तुत है–

(1) मूल्य (Cost) – किसी नेटवर्क की कीमत को घटाने के लिए आवश्यक है कि हम उसकी निर्माण की लागत कम रखने का प्रयास करें। ऐसा करने हेतु सूचना संकेतों को बदलने वाले माध्यम का निर्धारण आवश्यक है। जिस माध्यम पर ट्रांसमिशन करना है, उसकी लम्बाई भी आवश्यकतानुसार बदली जा सकती है।

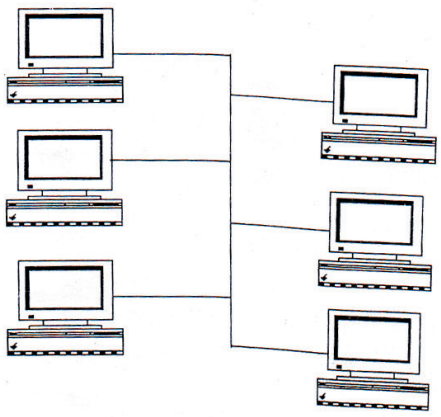
(2) लचीलापन (Flexibility) – चूंकि नेटवर्क का विस्तार पूर्व निर्धारित न होने के कारण नोड्स की संख्या भी निश्चित नहीं की जा सकती। इसी कारण हमारी चुनी हुई टॉपोलॉजी ऐसी होनी चाहिए जिसमें विस्तार की क्षमता हो।

(3) विश्वसनीयता (Reliability) – एक नेटवर्क में दो प्रकार की खराबियाँ हो सकती हे। पहली किसी एक नोड का खराब हो जाना, दूसरी पूरे नेटवर्क का खराब हो जाना। हमें यहाँ इस बात पर गौर करना चाहिए कि किसी एक नोड के खराब हो जाने पर बाकी नेटवर्क पर उसका प्रभाव नहीं पड़े।

टॉपोलॉजी मुख्यतया तीन प्रकार की है, वे इस प्रकार है–

4.2.1 बस टॉपोलॉजी (Bus Topology) –

बस टॉपोलॉजी में सभी कम्प्यूटरों को एक ही केबल से जोड़ा जाता है। हर नोड (कम्प्यूटर) दो दूसरे नोड्स से जुड़ा होता हैं इस शैली में एक बार में एक ही कम्प्यूटर संदेश भेज सकता है।



चित्र 4.2 टॉपोलॉजी सिस्टम

बस टॉपोलॉजी के लाभ

1) यह तकनीक सरल, आसानी से समझ में आने वाली तथा आसानी से काम में आने वाली है।

2) इसमें कम्प्यूटरों को जोड़ने के लिए कम केबल काम में आती है। अतः ये सस्ती पड़ती है।

3) इसमें आसानी से कई कम्प्यूटरों को जोड़ा जा सकता है।

बस टॉपोलॉजी की हानियाँ

1) जब कम्प्यूटर एक–दूसरे के समन्वय स्थापित नहीं कर पा रहे है तो सभी कम्प्यूटर एक साथ ट्रांशमिशन शुरू कर देते है। इसके कारण नेटवर्क धीमा हो जाता है।

2) केबल टूट जाने पर दो कम्प्यूटर आपस में ट्रांसमिशन नहीं कर पाते हैं।

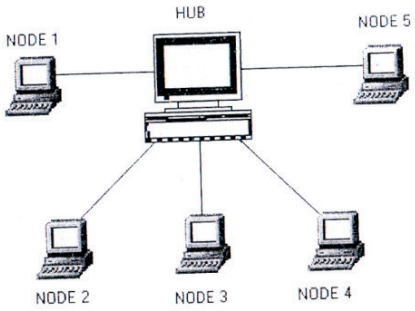
4.2.2 स्टार टॉपोलॉजी (**Star Topology**) : इस टॉपोलॉजी में सभी कम्प्यूटरों को हब या स्वीच से जोड़ा जाता है। हब या स्वीच नेटवर्क के केन्द्र में होता है तथा एक नोड के संकेतों को सभी नोड्स को भेजता है।

स्टार टॉपोलॉजी के लाभ

1. इसमें नये कम्प्यूटर जोड़ना सरल है। कम्प्यूटर को केबल द्वारा हब तक जोड़ों और कम्प्यूटर नेटवर्क में जुड़ जाता है।

2. इसमें हब या स्वीच द्वारा दोषों का आसानी से पता लगाया जा सकता है।

3. एक कम्प्यूटर के काम करना बन्द कर देने से बाकी नेटवर्क पर कोई प्रभाव नहीं पड़ता है।

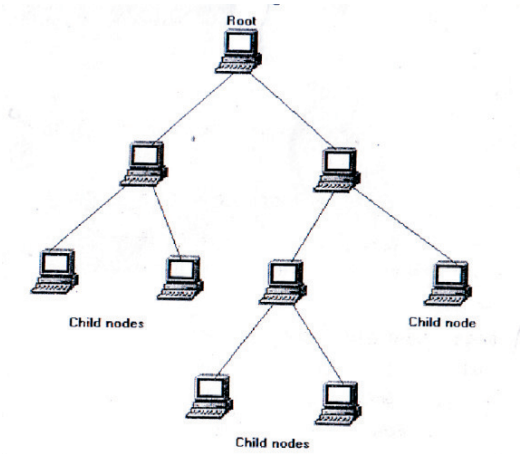


चित्र स. 4.3 स्टार टॉपोलॉजी

स्टार टॉपोलॉजी की हानियाँ

1. हब या स्वीच के खराब होने पर पूरा नेटवर्क काम करना बन्द कर देता है।
2. स्टार टोपोलॉजी महँगी पड़ती है क्योंकि इसमें हर कम्प्यूटर को केन्द्र में स्थित हब या स्वीच से अलग केबल द्वारा जोड़ा जाता है।

4.2.3 ट्री टॉपोलॉजी (**Tree Topology**) : इस टॉपोलॉजी में कम्प्यूटरों को इस प्रकार जोड़ा जाता है जैसे वे पेड की गई शाखाऐं है। सबसे ऊपर स्थित नोड को रूट (Root) कहा जाता है। रूट से शून्य या अधिक चाइल्ड नोड (Child Node) जुड़े होते हैं।



चित्र 4.4 ट्री टॉपोलॉजी

रूट नोड अपने चाइल्ड नोड का जनक (Parent) होता है। हर चाइल्ड नोड के भी कई चाइल्ड नोड हो सकते हैं। इस तरह से हर नोड के जनक नोड होते है केवल रूट नोड के ही जनक नोड नहीं होते है। इस प्रकार की टॉपोलॉजी में एक नोड से दूसरे नोड तक केवल एक पथ होता है।

ट्री टॉपोलॉजी के लाभ

1. इस टॉपोलॉजी में नए कम्प्यूटर चाइल्ड नोड से जोड़ना आसान होता है।
2. इसमें केन्द्रित हब की आवश्यकता नहीं पड़ती है।

ट्री टॉपोलॉजी की हानियाँ

1. कम्प्यूटर को रूट नोड पर जोड़ना आसान नहीं होता है।
2. कम्प्यूटर को रूट नोड पर जोड़ने अथवा हटाने से नेटवर्क बाधित हो सकता है।

रिंग, मैश एवं हाईब्रिड टॉपोलोजीज इन्हीं तीन प्रमुख टॉपोलोजीज का संवर्धित एवं परिवर्तित रूप है।

A. रिंग टॉपोलोजी– इस टॉपोलोजी में सभी कम्प्यूटर एक रिंग के स्वरूप में जुड़े होते है।

B. मैश टॉपोलोजी– इस टॉपोलोजी में सभी कम्प्यूटर एक जाल के रूप में जुड़े होते है।

C. हाईब्रिड टॉपोलोजी– इस प्रकार की टॉपोलोजी दो या दो से अधिक टॉपोलोजीज का मिश्रण होता है।

4.3 लैन (**LAN**), वैन (**WAN**), मैन (**MAN**) की अवधारणा

नेटवर्कों को वृहद् रूप से तीन प्रकार में विभाजित किया जाता है:–

4.3.1 लोकल एरिया नेटवर्क (**LAN**) – यह नेटवर्क एक भवन के अन्दर अथवा कुछ किलोमीटर क्षेत्र में

फैला होता है। ये किसी कार्यालय अथवा कारखाने के विभिन्न कम्प्यूटरों द्वारा सूचना तंत्र संसाधनों के आदान–प्रदान के लिए काम में लिया जाता है।

लैन सामान्यतः छोटे क्षेत्र में ही फैले होते हैं जैसे एक विभाग या एक भवन में । लैन छोटे होते है इसलिए इन्हें संभालना आसान होता है। इनमें साधारणतः शॉर्ट सर्किट एवं अनचाहे संकेतों से कुछ दोष आ जाते है। लैन एक ही केबल से सभी नोड्स को जोड़ता है। ये टेलीफोन केबल का ट्रांसमिशन के लिए उपयोग करते है, इसलिए ये सस्ते पड़ते हैं।

लैन की विशेषताएँ

1) लैन की सबसे महत्वपूर्ण विशेषता इसकी गति है। सामान्यतया इसकी डेटा ट्रांसमिशन गति 10 से 100 एम.बी.पी.एस. (Mbps) होती है। वर्तमान में यह 1 Gbps एवं इससे भी ज्यादा है।

2) लैन काफी लचीला नेटवर्क है, इसमें बगैर सारे नेटवर्क को विघ्न पहुंचाये, कम्प्यूटर को जोड़ा एवं हटाया जा सकता है।

3) चूंकि लैन का क्षेत्र सीमित होता है इसलिए इसमें हम कई प्रकार की टॉपोलॉजी काम में ले सकते हैं।

4.3.2 मेट्रोपॉलिटन एरिया नेटवर्क (मैन) (MAN) – मेट्रोपॉलिटन का मतलब है शहर। यह नेटवर्क बहुत बड़े क्षेत्र में फैला होता है जैसे कि शहर अथवा कस्बा। मैन, लैन का बड़ा स्वरूप है, यह भी लैन द्वारा काम में ली जा रही तकननीक का ही उपयोग करता है। इसको बनाना लैन की तुलना में अधिक जटिल है। यह 60 किलोमीटर तक के क्षेत्र में फैला हो सकता है। यह एक ही शहर के अलग–अलग क्षेत्रों में स्थित बैंक शाखाओं और व्यवसायिक घरों को आपस में जोड़ता है। उदाहरण–शहरों के केबल टी.वी. नेटवर्क।

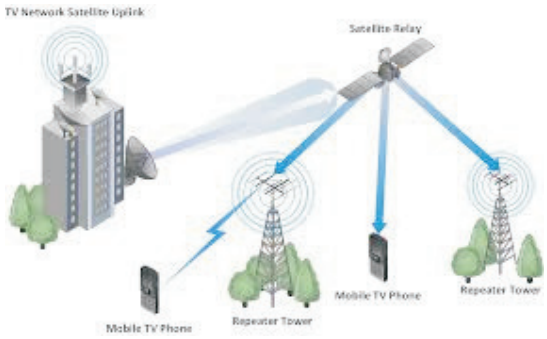
मैन की विशेषताएँ

1) पूरा नेटवर्क एक केन्द्रीकृत मशीन द्वारा संचालित एवं नियंत्रित होता है।

2) मैन का मुख्य ध्येय, सॉफ्टवेयर एवं हार्डवेयर संसाधनों का मिलकर उपयोग करना है।

3) मैन डेटा और ध्वनी दोनों का ट्रांसमिशन कर सकता है।

4.3.3 वाइड एरिया नेटवर्क (वैन) (WAN)– दूसरे नेटवर्को की तुलना में वैन बहुत बड़े क्षेत्र में काम करता है, यह सम्पूर्ण देश अथवा प्रायद्वीपों में भी फैला हो सकता है। वैन में देशों अथवा प्रायद्वीपों के सभी कम्प्यूटर एक–दूसरे से जुड़े होते है। ये कम्प्यूटर डेटा का आदान–प्रदान एवं केन्द्र नियंत्रित ट्रांसमिशन भी कर सकते है। वैन ट्रांसमिशन, तार वाले माध्यम (टेलीफोन लाइन, फाइबर ऑप्टिक्स) अथवा बेतार माध्यम (माइक्रोवैव) से हो सकता है। तीनों तरह के नेटवर्कों में यह सबसे बड़े क्षेत्र में फैले होते है। वैन नेटवर्को को जोड़ना बहुत ही जटिल होता है। इनमें शॉर्ट सर्किट, तार टूटने के कारण या दूसरे सर्किट से दोष आ सकते है।



चित्र 4.5 वैन सिस्टम

वैन की विशेषताएँ

(1) वैन कई नेटवर्क उपकरणों का ट्रांसमिशन के लिए उपयोग करता है जैसे– राउटर, स्विच, और गेटवे।

2) वैन डेटा ट्रांसमिशन के लिए दो प्रकार की स्विचिंग पद्धति का उपयोग करता है।

(अ) पैकेट स्विचिंग

(ब) सर्किट स्विचिंग

3) इनकी डेटा ट्रांसमिशन गति दूसरे नेटवर्कों की तुलना में धीमी होती है।

4.4 मानकीकरण एवं प्रोटोकॉल (Standardization & Protocol)

इलैक्ट्रिक उपकरणों में कम्यूनिकेशन के लिए नियमों के समूह का होना आवश्यक है जो कि प्रेषक और प्राप्तकर्ता के बीच कम्यूनिकेशन के कारक का कार्य करते है। प्रोटोकॉल प्रेषक तथा प्राप्तकर्ता के बीच सही कम्यूनिकेशन को सुनिश्चित करते है। बिना किसी प्रोटोकॉल के प्रेषक व प्राप्तकर्ता कम्पूयनिकेशन को सही तरह से समझ नहीं पाऐगें। ऐसी कई सारे विषय है जिनमें प्रोटोकॉल प्रयोग में लाये जाते है।

प्रोटोकॉल : कम्प्यूटर नैटवर्क में entities संचरण के लिए बहुत सारे सिस्टम से वास्ता रखती है, इन entities का प्रोटोकॉल से सहमत होना जरूरी होता है। नियमों के ऐसे समूह (Set of Rules) जो डेटा कम्प्यूनिकेशन को शासित करते है, प्रोटोकाल कहलाते है। प्रोटोकॉल के निम्न तत्व है–

1) Syntex : किसी भी डेटा का प्रारूप बताता है।

2) Semanties: एक बिट के हर भाग से सम्बन्धित है कि किस प्रकार उसकी व्याख्या की गई है।

3) Timing : डेटा कब भेजा गया है तथा कितनी गति से।

Standards: किसी भी तकनीक या सिस्टम को नियमों के अनुसार विनियमन (Regulation) ही स्टेण्डर्ड (Standard) कहलाता है।

Standards की जरूरत : स्टैण्डर्स उपकरण बनाने वाली कम्पनियों के लिए एक प्रतिस्पर्धात्मक एवं खुला मार्केट बनाते है और उपकरणों के विश्व भर में काम करने के लिए उपयोगी बनाते है। जिससे कोई भी कम्पनी अपनी मनमानी नहीं कर सकें।

Standardization Committees

1) ISO (International Organization for Standardization)

2) ANSI (American National Standards Interface)

3) IEEE (Institute of Electrical and Electronics Engineers)

4) ITU (International Telecommunication Union)

4.5 ट्रांसमिशन (Transmission)

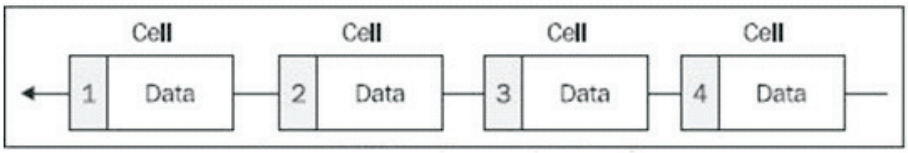
ट्रांसमिशन दो या दो से अधिक उपकरणों के मध्य डेटा का हस्तानान्तरण होता है जिसके अंतर्गत एक या एक से अधिक उपकरणों के मध्य डेटा भेजा एवं प्राप्त किया जाता है।

डेटा ट्रांसमिशन के तरीके

ट्रांसमिशन माध्यम पर डेटा दो तरीके से भेजा जा सकता है– असिंक्रोनस (Asynchronous) एवं सिंक्रोनस (synchronous)

1. असिंक्रोनस डेटा ट्रांसमिशन – असिंक्रोनस ट्रांसमिशन को स्टार्ट – स्टॉप ट्रांसमिशन भी कहा जाता है। असिंक्रोनस ट्रांसमिशन में हर अक्षर के मध्य स्टार्ट स्टॉप बिट जुडी रहती है। भेजने वाला कभी भी

एक अक्षर भेज सकता है जिसे रिसीवर रिसीव करता है। जब तक लाइन का हार्डवेयर, डेटा भेजने के लिए तैयार नहीं हो जाता तब तक असिंक्रोनस कम्युनिकेशन लाइन खाली रहती है। डेटा भेजा जा रहा है इसके बारे में बताने के लिए, बिट की एक श्रंखला प्राप्तकर्ता उपकरण को भेजी जाती है क्योंकि लाइन खाली रहती है। संपूर्ण डेटा भेजने के बाद प्राप्तकर्ता को डेटा समाप्ती की सूचना दी जाती है कि डेटा समाप्त हो चुका है। इसके लिए स्टॉप बिट भेजी जाती है ताकि लाइन को खाली किया जा सके। असिंक्रोनस ट्रांसमिशन में दो अक्षरो के बीच अंतराल अनिर्धारित होता है अर्थात कंप्यूटर डेस्टिनेशन तक, अक्षरो की एक श्रंखला भेज सकता है या डेटा अनियमित अंतराल पर भेजा जा सकता है।



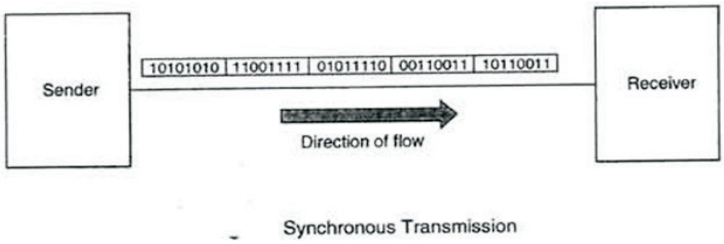
चित्रः 4.6 असिंक्रोनस डेटा ट्रांसमिशन

इसका प्रमुख लाभ यह है कि कंप्यूटर पर स्टोरेज की जरुरत नहीं होती है क्योंकि ट्रांसमिशन अक्षरशः होता है। इसकी ट्रांसमिशन की एक हानि यह है की दो अक्षरो को भेजने के बीच लाइन खाली रहती है।

2. सिंक्रोनस डेटा ट्रांसमिशन – सिंक्रोनस कम्युनिकेशन में दो चैनल होते हैं एक डेटा भेजने के लिए तथा दूसरा सभी लिंकों को एक साथ स्टार्ट रखने के लिए। दो कंप्यूटर को एक साथ स्टार्ट करने के लिए चैनल हार्डवेयर में लगी घड़ी का इस्तेमाल किया जाता है जब एक कंप्यूटर डेटा भेजने के लिए तैयार होता है तो वह रिसीवर के पास बिट का एक मिश्रण भेजता है जिसे सिंक (Sync) करैक्टर कहते हैं चूँकि पहला अक्षर खराब हो सकता है इसलिए दूसरा अक्षर साथ ही भेजा जाता है ताकि यह सुनिश्चित हो जाए की सभी लिंक एक साथ स्टार्ट हो सके।

सिंक्रोनस ट्रांसमिशन अक्षरों का एक ब्लाक बनता है। हर ब्लाक में हैडर और ट्रेलर की सूचना दी जाती है जिसके द्वारा प्राप्त करने वाला कंप्यूटर, भेजने वाले कंप्यूटर से अपनी घड़ी मिलाता है। हैडर में भेजने वाले तथा प्राप्त करने वाली कंप्यूटर की पहचान करने के लिए सूचना होती है। हैडर के बाद, अक्षरों का एक समूह होता है जो वास्तविक इनफार्मेशन (information) होती है तथा ब्लाक के अंत में ट्रेलर होता है। ट्रेलर सन्देश के समाप्त होने की सूचना देता है। इसके बाद एक चेक (Check) अक्षर होता है जो ट्रांसमिशन के दौरान आये दोषों को ढूँढने में मदद करता है।

सिंक्रोनस ट्रांसमिशन का लाभ इसकी दक्षता है। यह हर अक्षर के साथ स्टॉर्ट स्टॉप बिट की जरुरत को दूर करता है। असिंक्रोनस की तुलना में यह उच्च डेटा ट्रांसमिशन गति प्रदान करता है। ब्लाक भेजने के मध्य का अंतराल बहुत कम होता है और ब्लाक अधिकतम गति से भेजा जाता है। सिंक्रोनस ट्रांसमिशन का एक ही दोष है कि इसमें भेजने वाले उपकरण पर स्टोरेज के लिए बफर मेमोरी की आवश्यकता होती है।



चित्रः 4.7 सिंक्रोनस डेटा ट्रांसमिशन

डेटा ट्रांसमिशन की विधियां

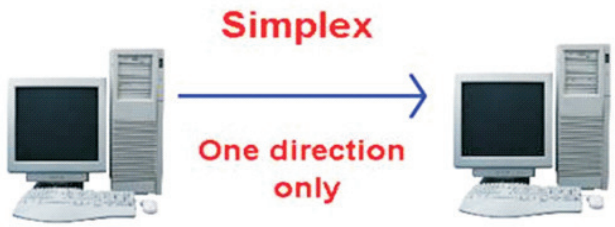
डेटा ट्रांसमिशन की विधियां होती हैं:–

(1) सिंप्लेक्स

(2) हाफ डुप्लेक्स

(3) फुल डुप्लेक्स

सिंप्लेक्स : सिंप्लेक्स ट्रांसमिशन (Simplex Transmission) विधि में एकतरफ डेटा कम्युनिकेशन होता है। सिंप्लेक्स ट्रांसमिशन का उदाहरण टेलीविजन कम्युनिकेशन है। इसमें मुख्य ट्रांसमीटर संकेत भेजता है परंतु इसी जवाब की अपेक्षा नहीं रखता है। क्योंकि प्राप्तकर्ता, ट्रांसमीटर को कोई जवाब नहीं दे सकता। सिंप्लेक्स ट्रांसमिशन का उदाहरण कीबोर्ड भी है क्योंकि वह कंप्यूटर से जुड़ा होता है और कंप्यूटर को केबल (Only) डेटा दे सकता है। एक तरफा ट्रांसमिशन में भी एक संदेश की आवश्यकता होती है जो यह बता सके की प्राप्त कर्ता ने डेटा प्राप्त कर लिया है। सिंप्लेक्स ट्रांसमिशन सामान्यतया काम में नहीं लिया जाता क्योंकि स्वीकृति, नियंत्रण तथा त्रुटि संकेत भेजने के लिए एक वापस मार्ग की आवश्यकता होती है।

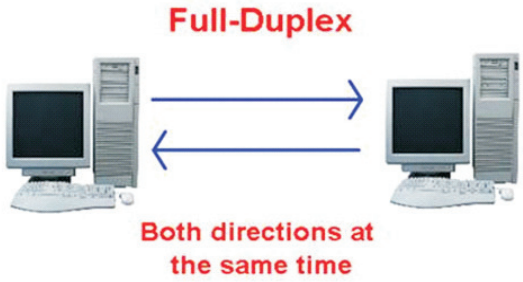


चित्रः 4.8 सिम्पलेक्स डेटा ट्रांसमिशन

हाफ डुप्लेक्स : हाफ डुप्लेक्स विधि में दोनों इकाइयाँ एक ही माध्यम से एक कम्युनिकेशन करती हैं किंतु एक बार में केवल एक ही इकाई डेटा का ट्रांसमिशन कर सकती है। जब एक इकाई डेटा भेज रही होती है तो दूसरी प्राप्त कर रही होती है अतः हाफ डुप्लेक्स लाइन वैकल्पिक रुप से डेटा भेजती और प्राप्त करती है। इसमें दो तारों की आवश्यकता होती है। यह बहुत ही सामान्य तरीका है। इसके द्वारा ध्वनि ट्रांसमिशन भी किया जा सकता है क्योंकि इसमें माना जाता है कि मे एक समय में एक ही व्यक्ति बोल सकता है। यह इकाई को कंप्यूटर से जोड़ने के काम आती है। यह का डेटा का ट्रांसमिशन कर सकती है और कंप्यूटर उसे स्वीकारोक्ति का सन्देश भेजता है। हाफ डुप्लेक्स में अगर दोनों उपकरण एक साथ डेटा भेजने तथा प्राप्त करने की कोशिश करते हैं तो पैकेट्स की टक्कर हो जाती है।

चित्रः 4.9 हाफ डुप्लेक्स डेटा ट्रांसमिशन

फुल डुप्लेक्सः फुल डुप्लेक्स मे सूचना एक ही समय में दोनों दिशाओं में बहती है। फुल डुप्लेक्स विधि के प्रयोग से ट्रांसलेशन की दक्षता से सुधार ला सकते हैं। फुल डुप्लेक्स कम्युनिकेशन में दोनों उपकरण एक साथ दोनों दिशाओं में डेटा भेजता तथा रिसीव कर सकते हैं। इसका प्रमुख उदहारण टेलीफोन सिस्टम है।



चित्रः 4.10 फुल डुप्लेक्स डेटा ट्रांसमिशन

समानान्तरण प्रसारणः इस प्रकार के प्रसारण में एक से अधिक संकेत बिट के रूप में एक साथ प्रेषक से लक्ष्य की ओर बहुतारों द्वारा प्रसारित होते हैं।

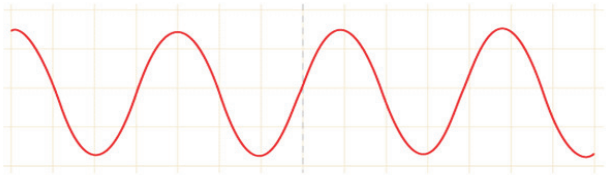
4.6 नेटवर्किंग संकेत (Networking Signals)

सिग्नल, डेटा की इलेक्ट्रिकल, इलेक्ट्रॉनिक्स और ऑप्टिकल रिप्रजेंटेशन (Representation) है जिसको कम्युनिकेशन मीडियम पर भेजा जा सकता है। डेटा ट्रांसमिशन,डिजिटल ट्रांसलेशन या डिजिटल कम्युनिकेशन डेटा का पॉइंट टू पॉइंट तथा पॉइंट टू मल्टी प्वाइंट चैनल पर भौतिक स्थानांतरण है। उदाहरण के लिए एक माइक्रोफोन से कन्वर्ट या परिवर्तित होने वाली ध्वनि वॉइस सिग्नल में कंवर्ट हो जाती है

नेटवर्किंग सिग्नल दो प्रकार के होते हैं–

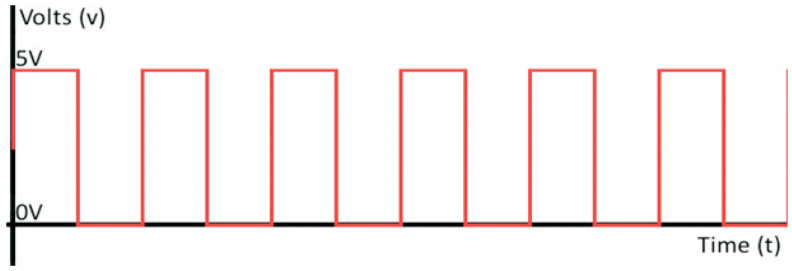
(1) एनालॉग सिग्नल्स (2) डिजिटल सिग्नल्स

एनालॉग सिग्नल्सः एनालॉग सिग्नल एक कंटीन्यूअस सिग्नल है जो समय के अनुसार बदलता है। इलेक्ट्रॉनिक सिग्नल में वोल्टेज, करंट और फ्रीक्वेंसी इनफार्मेशन को प्रस्तुत करने मे बदलते रहते हैं। ज्यादातर सिग्नल्स एनालॉग सिग्नल के रूप में रहते हैं जिनको बाद में डिजिटल सिग्नल्स में कन्वर्ट कर लिया जाता है क्योंकि एनालॉग सिग्नल्स को भेजने वाले उपकरण महंगे आते हैं। एनालॉग को डिजिटल में कनवर्ट करने के लिए आई. सी. परिपथ उपयोग में लाए जाते हैं। बाद में इन सिग्नल्स को एनालॉग में प्राप्त करना होता है इसलिए इन्हें डिजिटल से एनालॉग में कन्वर्ट कर लिया जाता है। सिग्नल्स को कन्वर्ट करते समय तथा भेजने के समय इन में कुछ त्रुटिया भी जुड़ जाती हैं जो कि मुख्य सिग्नल को खराब कर देती है इन्हें डिजिटल त्रुटिया (impairments) कहते हैं। जैसे–जैसे सिग्नल मीडिया पर आगे बढ़ता है उस की क्षमता या एनर्जी धीरे धीरे कम होने लगती है जिन्हें बढ़ाने के लिए एनालॉग सिग्नल में हम एम्पलीफायर का प्रयोग करते हैं।



चित्रः 4.11 एनालॉग सिग्नल्स

डिजिटल सिग्नल्सः डिजिटल सिग्नल इलेक्ट्रॉनिक सिग्नल है जो कि बिट के पैटर्न्स (Paterns) में कन्वर्ट (Convert) होता है। डिजिटल सिग्नल के हर बिंदु पर असतत वैल्यू होती हैं। डिजिटल सिग्नल को जीरो या वन से रिप्रेजेंट किया जाता है। एनालॉग सिग्नल्स की तरह डिजिटल सिग्नल में भी ट्रांसमिशन के समय त्रुटियां आ जाती हैं जिनके के फलस्वरुप वास्तविक सिग्नल्स का स्वरुप बदल जाता है तथा जैसे–जैसे सिग्नल मीडिया के ऊपर आगे बढ़ता है उसकी एनर्जी भी कम होती जाती है। किसी भी सिग्नल की एनर्जी कम से कम इतनी होनी चाहिए ताकि रिसीवर उसके बीच पैटर्न्स को समझ सके अन्यथा रिसीवर सही इंफॉर्मेशन का पता नहीं लगा पाएगा। कम हुए एनर्जी को सही करने के लिए डिजिटल सिग्नल्स में रिपीटर का इस्तमाल किया जाता है। ऑडियो और वीडियो को एक साथ ट्रांसलेट करने के लिए एच. डी. एम. आई. (HDMI) तकनीक उपयोग की जाती है जो डिजिटल सिग्नल्स का उदाहरण है।



चित्रः 4.12 डिजिटल सिग्नल्स

4.7 नेटवर्कों में प्रसारण माध्यम (Transmission Media in Networks)

ट्रांसमिशन माध्यम वह रास्ता है जिस पर ट्रांसमीटर तथा रिसीवर संकेतो का आदान प्रदान करते हैं या डेटा को एक दूसरे तक पहुंचाते हैं। ट्रांसमिशन मीडिया दो उपकरणों के मध्य भौतिक पथ को निर्धारित करता है। ट्रांसमिशन माध्यमों को दो भागों में बाँटा जा सकता है–

(1) निर्देशित ट्रांसमिशन मीडिया (Guided Transmission Media)

(2) अनिर्देशित ट्रांसमिशन मीडिया (Unguided Transmission Media)

निर्देशित ट्रांसमिशन मीडिया (Guided Transmission Media)

निर्देशित माध्यम वह माध्यम है जिसमे सिग्नल भौतिक पथ के अनुसार चलते हैं या पथ से भ्रमित नहीं होते हैं। निर्देशित मीडिया में मीडिया की कैपेसिटी उसकी लंबाई और वह किस प्रकार से जुड़ा हुआ है और उस पर निर्भर करती है।

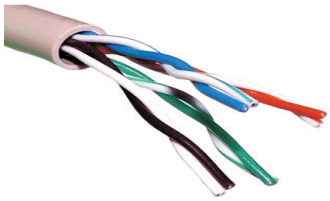
उदाहरणः ट्विस्टेड पेअर केबल, कोक्सिअल केबल तथा ऑप्टिकल फाइबर केबल

ट्विस्टेड पेअर केबल (Twisted pair Cable): ट्विस्टेड केबल टेक्नोलोजी में दो तार आपस में एक दूसरे से लिपटे हुए रहते हैं। दो तारों का आपस में लिपटे रहना एक दूसरे के इलेक्ट्रिक हस्तक्षेप को कम करता है, तथा एक जोड़े का एक दूसरे से लिपटे हुए रहना दूसरे जोड़े के इलेक्ट्रिक हस्तक्षेप को कम करता है। ट्विस्टेड पेअर केवल डिजिटल और एनालॉग सिग्नल्स को ट्रांसमिट कर सकती हैं। ट्विस्टेड पेअर केबल का उपयोग स्थानीय टेलीफोन ट्रांसमिशन के लिए और 1 किलोमीटर तक की छोटी दूरी में मुख्य कंप्यूटर एवं नेटवर्क कंप्यूटर के मध्य डिजिटल डेटा ट्रांसमिशन के लिए होता है डेटा ट्रांसमिशन की गति 100 मीटर की दूरी तक 9600 बिट्स प्रति सेकंड भी हो सकती है।

चित्रः 4.13 ट्विस्टेड पेअर केबल

ट्विस्टेड पेअर केबल दो प्रकार की होती है–

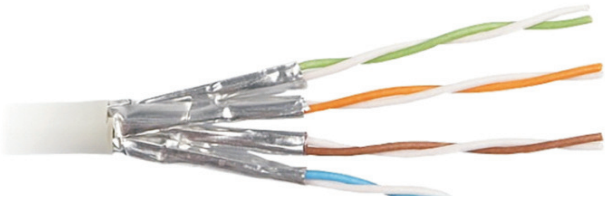
(1) अनशील्डेड ट्विस्टेड पेअर (UTP): यह सबसे प्रचलित तरह की ट्विस्टेड पेअर केबल है और यह लोकल एरिया नेटवर्क की केबलिंग करने में भी प्रचलित हो रही है। UTP साधारणतया टेलीफोन सिस्टम की केबल के रूप में तथा कई कार्यालय भवनों में पहले से ही काम में ले जा रही है। इस तरह की की केबल में क्रॉस टॉक होना संभावित है।



चित्रः 4.14 अनशील्डेड ट्विस्टेड पेअर केबल

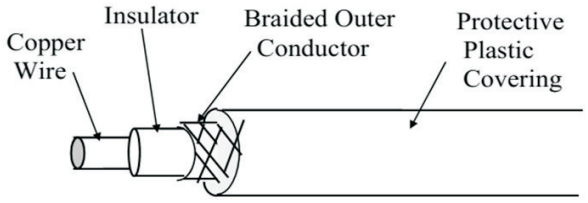
UTP केबल की मुख्य समस्या एक तार के संकेतों का दूसरा तार के संकेतों से मिश्रण है जिसे क्रॉस टॉक कहते हैं UTP में क्रॉस टॉक संभावित है इसलिए शील्डिंग का इस्तेमाल क्रॉस टॉक को कम करने के लिए किया जाता है।

एसटीपी (STP) : उच्च गुणवत्ता वाले गुथे हुए ताबें के तारों की जैकेट का उपयोग करता है। STP तारों के बीच एवं तारों के चारों तरफ से कॉल से लिपटा हुआ होता है यह STP को शानदार अबरोधक बनाता है जिससे जिसके द्वारा ट्रांसमिट हो रहे डेटा का भारी अपराधों से बचाव होता है इसलिए STP लंबी दूरी तक डेटा का ट्रांसमिशन UTP की तुलना में अधिक गति से करते हैं एवं ज्यादा सुरक्षित है।



चित्रः 4.15 शील्डेड ट्विस्टेड पेअर केबल

कोएक्सिअल केबल (**Coaxial Cable**): इसमे एक कड़क तांबे का तार जिसे कोर कहते हैं अवरोधक से लिपटा हुआ होता है। यह अबरोधक गुंथे हुए महीन तारों से ढका होता है इन गुथे तारों के ऊपर एक सुरक्षात्मक प्लास्टिक का खोल चढ़ा होता है। संकेतो का आदान–प्रदान ताम्बे की कोर द्वारा होता है विधुत कवच गुथे हुए तारों से बनाया जाता है।



चित्रः 4.16 कोएक्सिअल केबल

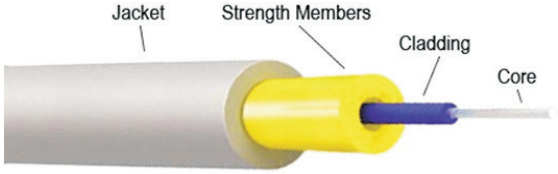
कोएक्सिअल केबल का उपयोग सामान्यता टेलीविजन के नेटवर्क के लिए किया जाता है कोएक्सिअल केबल ट्विस्टेड पेयर केबल की तुलना में लंबी दूरी तक अधिक गति से डेटा का ट्रांसमिशन कर सकती है। यह फाइबर ऑप्टिक केबल से सस्ती होती है एवं आसानी से काम में ली जा सकती है कोएक्सिअल केवल दो प्रकार की होती है

(1) थिकनेट (**Thicknet**) : यह केबल TV नेटवर्क में काम में आती है। यह केबल (Cable) मोटी होती है इन्हें आसानी से मोड़ा नहीं जा सकता इसलिए इसे काम में लेना कठिन होता है

(2) थिननेट (**Thinnet**) : यह अधिकतर नेटवर्किंग में काम ली जाती है। यह केबल लचीली एवं सस्ती होती है तथा आसानी से काम में ली जा सकती है।

फाइबर ऑप्टिक केबल (**Fiber Optic Cable**) : यह बिना तार वाली सबसे नई तकनीक केबल है। यह सुरक्षा एवं डेटा संभालने में सबसे उत्कृष्ट है। यह केबल विद्युत संकेतों के स्थान पर प्रकाश की तरंगों का ट्रांसमिशन करती है। डेटा भेजने का यह सबसे सुरक्षित तरीका है क्योंकि इस केबल में डेटा चुराया नहीं जा सकता। इस केबल का भीतरी हिस्सा काँच अथवा प्लास्टिक का बना होता है जिसमें प्रकाश का ट्रांसमिशन होता है, इस भीतरी भाग के ऊपर एक कांच की परत चढ़ी होती है जो प्रकाश को टकराने के बाद भीतरी भाग में भेजती है।

प्रेषक से डेटा ट्रांसमिशन उपकरण जुड़ा होता है जो कि विधुत संकेतों को प्रकाश तरंगों में परिवर्तित करता है। इन प्रकाश तरंगों का ट्रांसमिशन फाइबर ऑप्टिक केबल के द्वारा किया जाता है। प्राप्तकर्ता कंप्यूटर से पहले, दूसरा डेटा ट्रांसमिशन उपकरण इन विद्युत संकेतों को प्रकाश तरंगों में बदल देता है बाद में यह प्रकाश तरंग प्राप्तकर्ता कंप्यूटर को भेजे जाते हैं। फाइबर ऑप्टिक केबल, कोऐक्सिअल केबल से महंगी होती है। यह 60 मेगाबाइट प्रति सेकेंड से दो गीगाबाइट प्रति सेकंड की गति से डेटा ट्रांसमिशन कर सकती है। तीव्र गति एवं लंबी दूरी तक अधिक डेटा ट्रांसमिशन के लिए फाइबर ऑप्टिक अधिक उपयुक्त है पर यह सबसे महंगा बिना तारवाला ट्रांसमिशन माध्यम है।



चित्रः 4.17 फाइबर ऑप्टिक केबल

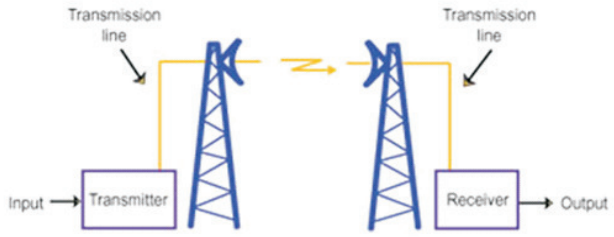
अनिर्देशित ट्रांसमिशन मीडिया (**Unguided Transmission Media**) : अनिर्देशित ट्रांसमिशन माध्यम वह माध्यम है जिसमें रेडिओ तरंगो का उपयोग होता है तथा सिग्नल्स भौतिक पथ के अनुसार नहीं

चलते हैं तथा इसका उपयोग वहाँ किया जाता है जहां पर पहुंचाना मुमकिन ना हो या कठिन हो।

रेडियो ट्रांसमिशन माध्यम : रेडियो तरंगो को आसानी से उत्पन्न किया जा सकता है यह लंबी दूरी तक पहुँच सकती हैं तथा यह इमारतों को भी आसानी से पार कर सकती है इसलिए इनको डेटा ट्रांसमिशन के काम में अधिक लिया जाता है, ट्रांसमिशन के बाद रेडियो तरंगे सभी दिशाओं में विचरण कर सकती हैं इसलिए प्रेषक और प्राप्तकर्ता का एक ही पंक्ति (Direction) में होना आवश्यक नहीं है। पूर्व में रेडियो कम्युनिकेशन टेलीग्राम संदेशों के ट्रांसमिशन के काम में लिए जाते थे। यह रेडियो ट्रांसमिशन कम ट्रांसमिशन गति एवं न्यून आवृति (Frequency) पर होता है पर अब रेडियो ट्रांसमिशन VHF (Very High Frequency) एवं UHF (Ultra High Frequency) पर तीव्र गति से डेटा ट्रांसमिशन करता है। इन दिनों ट्रांसमिशन आवृतियो (Frequencies) का उपयोग अनेक तरीकों से करते हैं।

30 KHZ	न्यून आवृति (Low Frequency)	लम्बी दूरी के टेलीग्राफ संकेतो में काम में लिया जाता है।
3 MHZ	मध्य आवृति (Medium Frequency)	मध्यम दूरी तक बिना तार का संकेत भेजने मे उपयोग।
30 MHZ	तेज आवृति (High Frequency)	लम्बी दूरी तक बिना तार का संकेत भेजने मे उपयोग।
300 MHZ	बहुत तेज आवृति (Very High Frequency)	एफ. एम.(FM) प्रसारण एवं छोटी दूरी के मोबाइल ट्रांसमिशन में उपयोग।
3 GHZ	अत्यंत तेज आवृति (Ultra &High Frequency)	टेलीविजन प्रसारण मे।
30 GHZ	सुपर तेज आवृति (Super High Frequency)	राडार एवं माइक्रोवेव ट्रांसमिशन मे उपयोग।

माइक्रोवेव ट्रांसमिशन माध्यम (**Microwave Transmission Media**) : इन संकेतो को भी TV और रेडियो के संकेतो की तरह बिना केबल के भेजा और प्राप्त किया जाता है। माइक्रोवेव संकेतो का प्रसारण इमारतों के ऊपर लगे एंटीना के द्वारा किया जाता है। प्रेषक एंटीना और प्राप्तकर्ता एंटीना को एक ही रेखा में रखा जाता है क्योंकि माइक्रोवेव संकेत केवल एक ही दिशा में आगे बढ़ते हैं। प्रेषक एंटीना एवं प्राप्तकर्ता एंटीना को एक ही रेखा में करने को लाइन ऑफ फाइट ट्रांसमिशन कहा जाता है। जमीन पर स्थित माइक्रोवेव स्टेशनों को इस प्रकार लगाया जाता है कि वह एक दूसरे से सूचना का आदान प्रदान कर सकेंगे इसलिए इन माइक्रोवेव स्टेशनों को इमारतों की छत पर या पहाड़ पर लगाया जाता है ताकि ट्रांसमिशन पथ अवरोध मुक्त रहे।



चित्रः 4.18 माइक्रोवेव कम्युनिकेशन

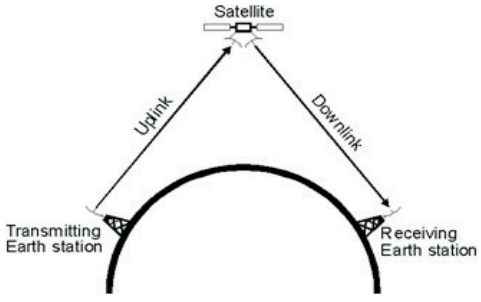
लाइन ऑफ साइट की समस्या के निदान के लिए माइक्रोवेव सिस्टम रिपीटर स्टेशनो का उपयोग करते हैं। माइक्रोवेव ट्रांसमिशन की दूरी सीमित है इसलिए हर 25–30 किलोमीटर के बाद रिपीटर लगाए जाते हैं। प्रेषक स्टेशन माइक्रोवेव संकेतो को भेजते हैं जिन्हें रिपीटर प्राप्त करते हैं, संकेत प्राप्ति के बाद रिपीटर इन्हें शक्ति प्रदान करते हैं ताकि वो और ये दूरी तक पहुंच सके इन्हें शक्ति प्रदान करने के बाद संकेतो को पुनः ट्रांसमिट किया जाता है। माइक्रोवेव ट्रांसमिशन फाइबर ऑप्टिक केबल की तुलना में सस्ता पड़ता है। माइक्रोवेव संकेतो का उपयोग स्थानीय एवं लंबी दूरी तक उच्च गति से ट्रांसमिशन के लिए किया जाता है।

इन्फ्रारेड ट्रांसमिशन माध्यम (**Infrared Transmission Media**) : इसका उपयोग छोटी दूरी के इन्फ्रारेड के लिए किया जाता है। इंफ्रारेड ट्रांसमिशन सस्ता है एवं आसानी से काम में लिया जा सकता है तथा इनका उपयोग करने में कोई कानूनी अड़चन नहीं है क्योंकि यह भवन के अंदर ही काम आती है। इसमें प्रेषक एवं प्राप्त कर्ता के मध्य कोई अवरोध नहीं होना चाहिए। इंफ्रारेड माध्यम संकेतों के ट्रांसमिशन के लिए इंफ्रारेड प्रकाश का प्रयोग करता है। लाइट एमिटिंग डायोड (LED) प्रकाश के संकेतो को ट्रांसमिट करती है और फोटो डायोड प्रकाश संकेतों को प्राप्त करते हैं। क्योंकि इंफ्रारेड संकेत बहुत तेज आवृति पर काम करते हैं इसलिए इनकी डेटा ट्रांसमिशन गति बहुत तेज होती है।

इंफ्रारेड ट्रांसमिशन के उदाहरण हैं TV एवं टेप के रिमोट। उदाहरण एक बड़े कमरे में कई कंप्यूटर जिनमें इन्फ्रारेड ट्रांसमिशन करने को प्राप्त करने वाले उपकरण लगे होते हैं इस तरह की कंप्यूटर इन्फ्रारेड द्वारा लोकल एरिया नेटवर्क से जोड़े जा सकते हैं।

चित्रः 4.19 इन्फ्रारेड कम्युनिकेशन

उपग्रह ट्रांसमिशन (**Satellite Transmission**) : पहली बार उपग्रह में ट्रांसमिशन का उपयोग 1960 मे हुआ जब NASA ने इको उपग्रह का प्रक्षेपण किया।उपग्रह ट्रांसमिशन तभी हो सकता है जब दोनों ऐन्टेना एक रेखा में हो। उपग्रह ट्रांसमिशन एवं माइक्रोवेव ट्रांसमिशन में मुख्य अंतर यह है कि एक ऐन्टेना उपग्रह पर लगा होता है जो भूमध्य रेखा के करीब 3600 किलोमीटर ऊपर जियो सिंक्रोनस कक्षा पर स्थित होता है। यह उपग्रह पृथ्वी की तुलना में स्थिर रहता है और पृथ्वी के सापेक्ष एक बिंदु पर रहता है इस कारण से उपग्रह प्रणाली द्वारा मोबाइल उपकरणों में ट्रांसमिशन किया जा सकता है तथा सभी स्थानों तक पहुँचा जा सकता है।



चित्र 4.20 उपग्रह कम्युनिकेशन

उपग्रह ट्रांसमिशन में 6 GHZ संकेतों का पृथ्वी पर ट्रांसमीटर द्वारा अंतरिक्ष में स्थित उपग्रह ट्रांसमिशन द्वारा प्रयोग होता है। यह संकेत लंबी दूरी तय करने के कारण क्षीण हो जाते हैं। उपग्रह पर एक ट्रांसपोंडर द्वारा इन संकेतो को शक्तिशाली बना कर वापस पृथ्वी पर 4GHZ आवृत्ति पर भेज देते हैं पृथ्वी पर इन संकेतो को रिसीवर द्वारा प्राप्त करते हैं। कक्षा में उपग्रह स्थापित करना बहुत खर्चीला होता है।

उपग्रह ट्रांसमिशन की विशेषताएँ

1. हर उपग्रह का जीवन चक्र 7 से 10 साल तक होता है।
2. पृथ्वी पर स्थित स्टेशन उपयोगकर्ता से कुछ दूरी पर होते हैं। संकेतों को यह दूरी महंगे उच्च गति की ट्रांसमिशन माध्यमो द्वारा तय करनी पड़ती है।
3. उपग्रह ट्रांसमिशन को कोई भी टेप कर सकता है।
4. कई परिस्थितियों में उपग्रह काम नहीं कर पाते क्योंकि उपग्रह सौर ऊर्जा से चलते हैं। सूर्य ग्रहण की स्थिति में पृथ्वी उपग्रह और सूर्य के बीच में आ जाती है जिसके कारण उपग्रह को सौर ऊर्जा मिलना बंद हो जाती है इस वजह से कभी उपग्रह की कार्य क्षमता घट जाती है तो कभी वह कार्य करना बंद कर देते हैं।
5. उपग्रह को भेजने वाली आवृति एवं वापस पृथ्वी पर संकेतों के आने वाली आवृति C बेड (4/6 GHZ) अथवा K4 (11/4 GHZ) की हो सकती है इसलिए उसी आवृति का बड़ा एंटीना चाहिए तथा भूस्थित स्टेशनों को बरसात एवं वातावरण के कारण अवरोधों का सामना करना पड़ता है।

4.8 OSI & TCP/IP Model

OSI और टीसीपी आईपी (TCP/IP) मॉडल की की जानकारी रखने से पहले हमें यह भी जानना आवश्यक है कि इंटरनेट कैसे आयी।

अगर हम इतिहास की बात करें तो इंटरनेट का इतिहास डिपार्टमेंट ऑफ डिफेंस (Department of Defence United States) यूनाइटेड स्टेट्स से शुरू होता है। जहां उनकी विभिन्न जगहों पर पहले से ही कंप्यूटर थे पर उनके बीच कोई भी सूचना का आदान प्रदान नहीं होता था और जो भी सूचना का आदान प्रदान होता था वह सूचना किसी बाहरी स्टोरेज में रखकर भेजी जाती थी इसलिए डिपार्टमेंट ऑफ डिफेंस ने पहल की कि उन के विभिन्न जगहों के कंप्यूटर्स को आपस में जोड़ा जाए इस प्रोजेक्ट को उन्होंने DARPA नाम दिया।

डिपार्टमेंट ऑफ डिफेंस एवं अन्य कई सहायक विभाग संगठनों ने रिसर्च द्वारा कुछ प्रोटोकॉल्स का निर्माण किया जो कंप्यूटर्स को आपस में जोड़कर सूचना का आदान प्रदान कर सकते थे। पूरी प्रक्रिया को उन्होंने DOD मॉडल का नाम दिया जिसमें मुख्यता चार परते (Layers) थी। बाद में DARPA नाम बदलकर आरपानेट (ARPANET) कर दिया गया और अरपानेट ने एक मॉडल प्रस्तुत किया जिसका नाम था TCP/IP Model. इसमें भी DOD की तरह 4 परते ही थी जिनके नाम निम्नलिखित हैं–

1 एप्लीकेशन लेयर

2 ट्रांसपोर्ट लेयर

3. इन्टरनेट लेयर

4. नेटवर्क एक्सेस लेयर

परतो की अवधारणा नेटवर्क कम्युनिकेशन के काम को विभिन्न हिस्सों मे बाटने के लिए की दी गई।

अरपानेट के साथ साथ एक और संगठन ISO (International Organization for

Standardization) ने एक मॉडल प्रस्तुत किया जिसका नाम OSI Model (Open System Interconnection) दिया गया, यह 7परतो का एक मॉडल था। बातचीत के बाद इसे रिफरेन्स मॉडल का दर्जा दिया गया तथा TCP/IP Model को प्रोटोकॉल मॉडल का दर्जा दिया गया।

TCP/IP Model ही कंप्यूटर नेटवर्क्स में लागू है तथा अगर इसकी विभिन्न परतो की जानकारी लेनी या पढनी होती है तो OSI model को ध्यान में रखा जाता है।

OSI & TCP/IP Model Layered Architecture

OSI Model	TCP/IP Model
Application Layer	Application Layer
Presentation Layer	
Session Layer	
Transport Layer	Transport Layer
Network Layer	Internet layer
Data Link Layer	Network Access layer
Physical Layer	

OSI मॉडल की सुविधा

1. नेटवर्क की बड़ी तस्वीर को समझा जा सकता है।
2. हार्डवेयर और सॉफ्टवेयर एक साथ काम करते हुए देखना मुमकिन है।
3. इसमें क्या नई तकनीक विकसित की है इसकी जानकारी लेना आसान है।
4. अलग–अलग नेटवर्क की समस्याओं का निवारण आसान हो जाता है।
5. अलग–अलग नेटवर्क पर बुनियादी कार्यात्मक संबंध तुलना करने के लिए इस्तेमाल किया जा सकता है।

OSI Model की विभिन्न परतें

इस मॉडल को सात परतों में विभाजित किया गया है (ऊपर से नीचे) जो निम्न है–

1. एप्लीकेशन लेयर (**Application layer**) : यह शीर्ष स्तर है। विभिन्न तरीकों से डेटा (सूचना) के हेर फेर इस परत में किया जाता है। मेल सेवाओं, निर्देशिका सेवाओं, नेटवर्क संसाधन आदि का उपयोग एप्लीकेशन लेयर द्वारा ही होता है।

2. प्रेजेंटेशन लेयर (**Presentation Layer**): यह परत ख्याल रखती है कि डेटा इस तरह से हो कि रिसीवर को जानकारी (डेटा) समझ मे आ जाये और डेटा का उपयोग करने में सक्षम हो जाये। यह परत अनुवादक की भूमिका भी निभाती है।

3. सेशन लेयर (**Session Layer**) : यह परत दो उपकरणों के मध्य बातचीत के सिंक्रोनाइजेशन का काम करती है। डेटा में कोई भी हानि ना हो इसके लिए प्रेजेंटेशन लेयर डेटा का सही तरीके से सिंक्रोनाइजेशन दूसरी तरफ की प्रेजेंटेशन लेयर से करती है।

4. ट्रांसपोर्ट लेयर (**Transport Layer**) : यह मुख्यता सबसे महत्वपूर्ण लेयर है जिसका प्रमुख कार्य डेटा को एक कंप्यूटर से दूसरे कंप्यूटर तक जिम्मेदारी से पहुंचाना होता है। यही परत फैसला करती है कि डेटा का संचरण समानांतर पथ पर होगा या एकल पथ पर होगा। इस परत के प्रमुख कार्य मल्टीप्लेक्सिंग, विभाजन (segmentation) तथा addressing हैं।यह परत डेटा को छोटे–छोटे टुकड़ों में तोड़ देती है।

जिससे डेटा का संचरण सही तरीके से हो सके उसे प्रोटोकॉल डेटा यूनिट (PDU) कहते हैं। ट्रांसपोर्ट लेयर पर प्रोटोकॉल डेटा यूनिट को सेगमेंट (Segment) का नाम दिया गया है। ट्रांसपोर्ट लेयर इन सबके साथ साथ साथ एड्रेसिंग का भी काम करती है जो कि एप्लीकेशंस को नेटवर्क में पहचानने में मदद करती है। ट्रांसपोर्ट लेयर पर एड्रेसिंग के रूप में पोर्ट (Port) नंबर प्रयोग में लाए जाते हैं जो जीरो (0) से लेकर 65535 तक होते हैं।

ट्रांसपोर्ट लेयर द्वारा भेजा गया डेटा अगर प्राप्तकर्ता तक सही नहीं पहुंचता है तो यह ट्रांसपोर्ट लेयर की जिम्मेदारी होती है कि वह उस डेटा को दोबारा भेजें तथा यह सुनिश्चित करें डेटा सही तरीके से सही प्रकार एवं रुप में पहुंच गया है।

5. नेटवर्क लेयर (**Network Layer**) : नेटवर्क लेयर ट्रांसपोर्ट लेयर से प्राप्त सेगमेंट मे अपनी जानकारी जोड़कर पैकेट में बदल देती हैं। नेटवर्क लेयर का प्रमुख कार्य डेटा को एक नेटवर्क से दूसरे नेटवर्क में भेजना होता है जिसे राउटिंग कहते हैं। नेटवर्क लेयर के पास सारणी या राउटिंग टेबल होती है जिससे वह यह निर्धारित करता है कि कौन सा रास्ता डेटा को भेजने के लिए अच्छा होगा। अगर किसी कारण से कोई रास्ता सही नहीं है तो राउटर अपनी राइटिंग टेबल में से दूसरा रास्ते का चयन करता है।

नेटवर्क लेयर राउटिंग के साथ–साथ एड्रेसिंग भी देती है जिसे हम IP एड्रेस कहते हैं।

6. डेटा लिंक लेयर (**Data Link Layer**) : डेटालिंक लेयर नेटवर्क लेयर से प्राप्त पैकेट को अपनी इंफॉर्मेशन जोड़कर फ्रेम में कन्वर्ट कर देती है जो कि बिट्स का एक समूह होता है। डेटा लिंक लेयर त्रुटि का पता करने तथा उसके निवारण का भी कार्य करती है। इन सबके साथ साथ टाटा लिंक लेयर दो महत्वपूर्ण कार्य और करती है जो निम्नलिखित हैं–

1.मीडियम को कैसे उपयोग में लाया जाए

2.एड्रेसिंग मैक एड्रेस के रूप मे (MAC Address)

7. फिजिकल लेयर (**Physical Layer**) : फिजिकल लेयर, डेटा लिंक लेयर से प्राप्त फ्रेम्स को भौतिक संकेतों में बदल देती है। यह परत लिंक को चालू करने, उसको सुचारु बनाए रखने तथा उसको बंद करने के लिए भी जिम्मेदार होती है।

कंप्यूटर नेटवर्क्स मे डेटा, एप्लीकेशन लेयर से जनरेट होता है तथा विभिन्न लेयर्स से प्रोसेस होकर फिजिकल लेयर तक पहुंचता है उसे बाद में विद्युत संकेतो में बदल दिया जाता है। यह संकेत मीडिया के साथ चलते चलते दूसरे उपकरण तक पहुंचते हैं, यहां पर वह उपकरण इन संकेतो को वापस आवश्यक जानकारी मे बदल देता है। यह जानकारी अन्य परतो से होती हुई वापस एप्लीकेशन लेयर को मिलती है जिसे काम करने वाला व्यक्ति या यूजर प्राप्त कर लेता है। डेटा का एप्लीकेशन लेयर से फिजिकल लेयर की ओर जाना कैप्सूलीकरण (Encapsulation) कहलाता है तथा डेटा का फिजिकल लेयर से एप्लीकेशन लेयर की तरफ जाना Decapsulation कहलाता है।

नेटवर्क में हर लेयर्स अपना अपना काम जिम्मेदारी के साथ पूरा करती है।

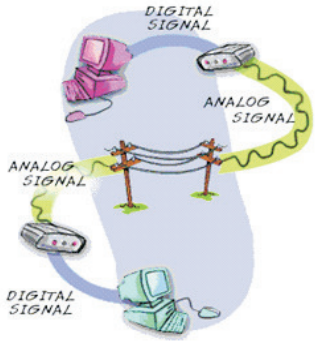
4.9 नेटवर्किंग का अभ्युदय (**Advancements of Networking**)

सूचना को बांटने एवं संसाधनों के उपयोग के लिए जब दो या दो से अधिक कंप्यूटरो को जोड़ा जाता है तो नेटवर्क बनता है। लोकल एरिया नेटवर्क को भी कभी–कभी ट्रांसमिशन माध्यम की क्षमता से ज्यादा दूरी तय करनी पड़ती है तब नेटवर्क उपकरण लोकल एरिया नेटवर्क को लंबी दूरी करने में मदद करते हैं। ये नेटवर्क उपकरण दो अलग–अलग नेटवर्क को ट्रांसमिशन के लिए भी जोड़ सकते हैं सामान्य नेटवर्क उपकरण इस प्रकार है–

मॉडेम (**Modem**) : ट्रांसलेशन एनालॉग एवं डिजिटल में से किसी से भी हो सकता है जब डिजिटल डेटा को टेलीफोन लाइन से संचालित करना होता है तो पहले डिजिटल डेटा को एनालॉग डेटा में बदलना

पड़ता है क्योंकि एनालॉग डेटा ज्यादा तेज गति से ट्रांसमिट होता है। प्रेषक के यहां जिस तकनीक से डिजिटल संदेशों को एनालॉग संदेशों में बदला चाहता है उसे मोड्यूलेशन (Modulation) कहते हैं। संदेश प्राप्तकर्ता के यहां उलटा काम होता है यानी एनालॉग संकेतो को डिजिटल संकेतो में डिमोड्यूलेशन (demodulation) तकनीक द्वारा बदला जाता है। एक विशिष्ट उपकरण जिसे मॉडेम कहते हैं, modulation एवं demodulation दोनों का ही काम करता है अर्थात वह संकेतों को डिजिटल से एनालॉग एवं एनालॉग से डिजिटल में बदल सकता है।

मॉडेम कंप्यूटर के डिजिटल संकेतो को एनालॉग संकेतो में बदलता है ताकि उनका ट्रांसलेशन टेलीफोन लाइन से हो सके। मॉडेम की डेटा ट्रांसमिशन गति बिट्स प्रति सेकंड (bps) में मापी जाती है।



चित्रः 4.21 मॉडेम कम्युनिकेशन

हब (**Hub**) : हब ऐसा उपकरण है जिसके द्वारा कई कंप्यूटरों को एक नेटवर्क में भौतिक रुप से जोड़ा जा सकता हैं। इस उपकरण में पहले सूचना को इकट्ठा किया जाता है तथा बाद में उसे कंप्यूटरों को ट्रांसमिट किया जाता है इसलिए इसे इंटरफेस (Interface) यूनिट कहा जाता है। यह केवल कुछ बिट्स का डेटा ही इकट्ठा कर सकता है इसलिए इसके द्वारा ट्रांसमिशन तीव्र गति से होता है। हब एक ऐसा उपकरण है जो पहले डेटा प्राप्त करता है फिर डेटा को शक्तिशाली बनाता है तथा दूसरे छोर के कंप्यूटर के लिए उन्हें ट्रांसमिट करता है क्योंकि हब डेटा को शक्तिशाली बनाता है इसलिए वह डेटा के साथ संकेतिक अवरोधो को भी शक्तिशाली बना देता है। यह एक सशक्त वायरिंग केंद्र है जो प्रिंटर, स्केनर, कंप्यूटर आदी उपकरणों को लेन (LAN) से जोड़ता है। केवल एक उपकरण ही एक समय में हब द्वारा डेटा ट्रांसमिशन कर सकता है। हब एक ऐसा बिंदु है जिसके द्वारा समस्या का पता लगाकर उसका निदान किया जा सकता है इसकी डेटा ट्रांसमिशन गति 10 मेगा बिट्स प्रति सेकेंड (Mbps) होती है। हब दो प्रकार के होते हैं।

1. पैसिव हब (**PASSIVE HUB**) : यह सबसे सरल हार्डवेयर उपकरण है यह नेटवर्क में कई केबल से डेटा संकेतो को प्राप्त कर सकते हैं।
2. एक्टिव हब (**ACTIVE HUB**) : यह जटिल हार्डवेर उपकरण है जो कि सब अलग–अलग नेटवर्क्स द्वारा दी गई सूचना का परीक्षण एवं नियंत्रण कर सकते हैं।

स्विच (**SWITCH**) : यह फ्रेम के शुरु होते ही प्राप्तकर्ता का पता देखकर स्विच फ्रेम्स को उनके गंतव्य पर भेजना शुरू कर देते हैं। भेजना शुरू करने से पहले यह फ्रेम का पूरा आने का इंतजार नहीं करते हैं। यह डेटा पैकेटों के संकेत अपराधों को दूर करता है जब डेटा पैकेट आते हैं तो उनके हैडर से रिसीवर का पता लगाया जाता है, फिर उन्हें रिसीवर तक भेज दिया जाता है। फ्रेम में बिट्स को एक पूर्व निर्धारित क्रम में

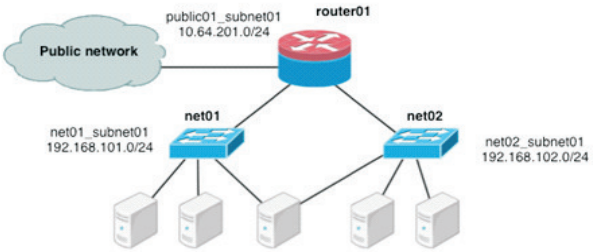
लगाया जाता है जिनमें गंतव्य का पता लगाने, गलती के नियंत्रण, रिसीवर का डेटा तथा फ्रेम के खत्म होने की सूचना के लिए बिट्स होती हैं।

एक स्विच में कितने कंप्यूटर जोड़ सकते हैं यह उस में कितने पोर्ट हैं उस पर निर्भर करता है। स्विच सबसे ज्यादा तेज होते हैं तथा इनकी हर उपकरण के लिए अलग बैंडविड्थ देते हैं। स्विच अतिरिक्त ट्रैफिक को कम करके नेटवर्क की कार्य क्षमता बढ़ाते हैं। वह इस बात को भी सुनिश्चित करते हैं कि सभी उपकरण एक ही समय पर ट्रांसमिशन कर सकें। स्विच जितने उपकरणों से जुड़ा होता है उन सभी के लिए अलग–अलग छोटी बफर मेमोरी रखता है। जब यह डेटा पैकेट प्राप्त करता है तो उसे बफर में सेव कर लेता है फिर उसका पता देख कर उसे उसके गंतव्य तक पहुंचा देता है अगर जिस स्थान से डेटा पैकेज आया है और उसका पता भी वही हो तो स्विच डेटा पैकेट को डिलीट कर देता है तथा उसका उन्हें ट्रांसमिशन नहीं करता है इस तरह अनावश्यक डेटा पैकेटों का ट्रांसमिशन रोककर वह डेटा ट्रैफिक को कम कर देता है और दूसरे शब्दों में कहें तो स्विच एक समझदार हब होता है।

राऊटर (**ROUTER**) : यह उपकरण असमान नेटवर्क के मध्य ट्रांसमिशन करता है। जब कई नेटवर्क को जोड़ कर बड़ा नेटवर्क बनाया जाता है तो नेटवर्क के मध्य एक पथ (Route) होते है, इन पथो को रूट कहते हैं। राउटर आदि उपकरण इन पथो की एक सारणी बनाते हैं जिनमें से वह 2 कंप्यूटरों के मध्य डेटा पथ का पता लगाते हैं। राउटर संदेश भेजने के लिए कई पथो में से सबसे सुगम पथ को चुनते हैं। अगर राउटर को एक पथ पर दोष का पता चलता है तो दूसरे पथ पर संदेश भेजने की कोशिश करता है।

राउटर केवल एक कंप्यूटर से दूसरे कंप्यूटर को डेटा की प्रति ही नहीं भेजते अपितु उनके मध्य रूटिंग सारणी द्वारा सुगम पथ को भी चुनते हैं पथो की जानकारी के लिए राउटर पड़ोसी राउटर को अपनी जानकारी भेजते हैं। पडोसी राउटर इस जानकारी से अपनी जानकारी मिलाकर दूसरे पड़ोसी को भेजते हैं इस तरह सभी राउटर के पास अपनी जानकारी के अलावा पड़ोसी राऊटर की जानकारी भी होती है।

राउटर के पास जब डेटा आता है तो वह उसके प्राप्तकर्ता कंप्यूटर का पता ज्ञात करता है तथा उसके बाद उसे प्राप्तकर्ता नेटवर्क को भेज देता है राउटर की मुख्य विशेषता उसका फायरवैल (FIREWALL) की तरह कार्य करना होता है क्योंकि यह डेटा पैकेटों को परीक्षण कर सकता है इस प्रकार अनचाहे डेटा पैकेटों के नेटवर्क में आने और जाने पर अंकुश लगता है। राऊटर डेटा नियंत्रण प्रणाली के माध्यम से डेटा को कम ट्रैफिक वाले पथों पर भेज पर ट्रैफिक जाम होने से बचाता है।



चित्रः 4.22 राऊटर

गेटवे (**GATEWAY**) : यह उपकरण असमान नेटवर्क को जोड़ता है। कुछ नेटवर्क यह जानकारी चाहते हैं कि डेटा के आने के बाद उन्हें किस तरह सुव्यवस्थित किया जा सकता है। जब हम दो अलग–अलग ऑपरेटिंग सिस्टम वाले दो या दो से अधिक नेटवर्क को जोड़ते हैं तो गेटवे की जरूरत पड़ती है। गेटवे संदेशों का पता और जरूरी प्रोटोकॉल बदलने के साथ उन्हें एक नेटवर्क से दूसरे नेटवर्क को भेजते हैं।

गेटवे प्रेषक नेटवर्क से निर्देशों के समूह को प्राप्तकर्ता नेटवर्क के निर्देशों में बदलता है। गेटवे सामान्यतः राउटर में एक सॉफ्टवेयर होता है। गेटवे अपने से जुड़े सभी नेटवर्क के निर्देशों को समझ सकता है तथा उन्हें एक से दूसरे निर्देशों में बदल सकता है। गेटवे अपने से जुड़े हुए कंप्यूटरों के निवेदन को सर्वर को समझाने वाले निर्देशों में बदलता है। यह सर्वर के संदेशो को प्राप्तकर्ता कंप्यूटर के समझने वाले संदेशो में बदलता है।

4.10 इन्टरनेट प्रोटोकॉल (**Internet Protocol-IP**)

IP एड्रेसिंग नेटवर्क लेयर का एक महत्वपूर्ण कार्य है जो किसी भी उपकरण के उसके नेटवर्क मे होने या दूसरे नेटवर्क में होने पर भी संचरण को मुमकिन बनता है। IP (Version 4) तथा IP (Version 6) दोनों ही पैकेट्स के द्वारा डेटा ले जाने के लिए श्रेणीबद्ध एड्रेसिंग प्रदान करती है। डिजाइन, कार्यान्वयन और एक प्रभावी आईपी (IP) योजना सुनिश्चित करती है कि नेटवर्क प्रभावी ढंग से और कुशलता से काम करेगा ।

यह जानना भी आवश्यक है की कंप्यूटर केवल बाइनरी (Binary) सिस्टम मे काम करते हैं जिसे जीरो (0) या वन (1) से प्रदर्शित करते हैं। कंप्यूटर यूजर/ऑपरेटर द्वारा अपनी भाषा में दिए गए निर्देशों को बाइनरी मे बदलता (Convert) है।

उदाहरण के लिए ASCII स्टैण्डर्ड में लिखे A का बाइनरी कोड 01000001 होता है।

हमे IP एड्रेस के लिए बाइनरी का प्रयोग जानना जरुरी है।

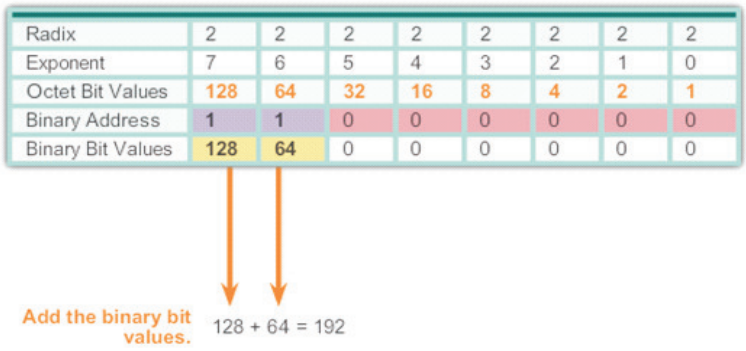
IP (Version 4) : IP (Version 4), 32 बिट का एक बाइनरी एड्रेस होता है। नेटवर्क लेयर पर पैकेट्स, प्रेषक और प्राप्तकर्ता की इस अद्वितीय (Unique) सूचना को शामिल कर लेते है। इसके फलस्वरूप पैकेट में 32 बिट का प्रेषक का तथा 32 बिट का प्राप्तकर्ता का एड्रेस शामिल हो जाता है। बाइनरी का डेसीमल मे बदलना गणित के स्थितीय अंकन (positional notation) पर आधारित है। स्थितीय अंकन (positional notation) अर्थात एक डिजिट, स्थान के अनुसार अलग मूल्य (value) देती है। स्थितीय अंकन (positional notation) मे नंबर का आधार सूत्र (radix) कहलाता है। डेसीमल सिस्टम मे रेडिक्स 10 होता है।

32 बिट के बाइनरी एड्रेस को हम डॉटेड डेसीमल (Dotted Decimals) फॉर्मेट में लिखते है क्योकि मानव बाइनरी को अच्छी तरह से पढ़ और याद नहीं रख सकता तथा कंप्यूटर अपना सारा काम बाइनरी मे करता है।

इस 32 बिट बाइनरी एड्रेस को 8 बिट या ऑक्टल के समूह मे दर्शाया जाता है तथा प्रत्येक 8 बिट का समूह एक बिंदु (Dot) से अलग रहता है।

उदाहरण के लिए 11000000 10101000 00001010 00001010 बाइनरी एड्रेस को डॉटेड डेसीमल में 192. 168.10.10 लिख सकते हैं। बाइनरी नंबर सिस्टम में रेडिक्स 2 होता है अतः संख्या या तो 0 होती है या 1। 8–बिट द्विआधारी बाइनरी संख्या में पद इन मात्राओं का प्रतिनिधित्व करते हैं–

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1



चित्र 4.23 इंटरनेट प्रोटोकॉल

हर ऑक्टल 8 बिट से बना होता है जिसमें बिट या तो 0 होती है या 1। 8 बिट के 4 समूह की मात्रा (Value) 0 से 255 की श्रंखला मे होती है। हर बिट के स्थान की वैल्यू सीधे से उलटी दिशा में 1, 2, 4, 8, 16, 32, 64, 128 होती है। अगर डेसिमल को बाइनरी में बदलना है तो दी गयी वैल्यू को उपरोत संख्याओ के अनुसार विभाजित करे, जिन जिन संख्याओं से मिले उनके स्थान पर 1 तथा अन्य के स्थान पर 0 रखे।

उदाहरण के लिए अगर 155 को बाइनरी में बदलना है तो निम्न विभाजन होंगे

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
128			16	8		2	1
1	0	0	1	1	0	1	1

155 की बाइनरी 10011011 होगी

अगर किसी बाइनरी को डेसीमल मे बदलना है तो उपरोक्त के अनुसार ही बदल सकते हैं। बाइनरी वैल्यू में जिस बिट का मान 1 है उसकी वैल्यू उसकी स्थिति के अनुसार रखे तथा अंत में जोड़ ले।

IP एड्रेस के दो भाग होते है जिसमें एक भाग उसके नेटवर्क तथा दूसरा होस्ट (Host) से सम्बंधित होता है।

IP एड्रेसेज को 5 कक्षाओं (Classes) मे बाँटा गया है

- Class A
- Class B
- Class C
- Class D (मल्टीकास्टिंग)
- Class E (भविष्य के लिए)

किसी भी IP के प्रथम ऑक्टल से उस IP की क्लास का पता लगाया जाता है। IP की श्रृंखला 1 से 255 तक होती है जो क्लासेस के अनुसार निम्न है

- Class A 0-127
- Class B 128-191

- Class C 192-223
- Class D 224-239
- Class E 240-255

उदाहरण के लिए 10.10.12.50 की क्लास A है क्योंकि इसका प्रथम ऑक्टल 10 है जो कि क्लास A की शृंखला में आता है।

IP एड्रेस का कितना भाग नेटवर्क का है कितना होस्ट इसका निर्धारण सबनेट मास्क (Subnet Mask) करता है। हर क्लास के अनुसार अलग अलग सबनेट मास्क निर्धारित किये हैं जो निम्न प्रकार हैं।

- Class A 255.0.0.0
- Class B 255.255.0.0
- Class C 255.255.255.0
- Class D मल्टीकास्टिंग
- Class E भविष्य के लिए

क्लास A में प्रथम ऑक्टल 255 है मतलब 8 बिट की वैल्यूज 1 हैं इसलिए IP एड्रेस का प्रथम ऑक्टल नेटवर्क से सम्बंधित रहेगा, इसके बाद बची हुए 24 बिट होस्ट से सम्बंधित रहेंगी। इसी प्रकार Class B मे प्रथम एवं दूसरा ऑक्टल 255 है मतलब 16 बिट की वैल्यूज 1 हैं इसलिए IP एड्रेस का प्रथम एवं दूसरा ऑक्टल नेटवर्क से सम्बंधित रहेगा, इसके बाद बची हुए 16बिट होस्ट से सम्बंधित रहेंगी। इसी प्रकार Class C मे प्रथम, दूसरा एवं तृतीय ऑक्टल 255 है मतलब 24 बिट की वैल्यूज 1 हैं इसलिए IP एड्रेस का प्रथम एवं दूसरा ऑक्टल नेटवर्क से सम्बंधित रहेगा, इसके बाद बची हुए 8बिट होस्ट से सम्बंधित रहेंगी।

IP (Version 6) : धीरे धीरे जैसे जैसे इन्टरनेट के यूजर बढ़ने लगे वैसे वैसे IP V4 मे आने वाले एड्रेस ही खतम होने लगे और जरूरते बढ़ने लगी इसलिए IP (Version 6) का विकास हुआ। |IP V6 में क्लास की संकल्पना नहीं है। यह 128 बिट का एड्रेस होता है जिसे हेक्साडेसीमल (Hexadecimal) में लिखा जाता है।

यह 2001:0db8:85a3:0000:0000:8a2e:0370:7334 IP V6 एड्रेस का एक उदाहरण है। इस एड्रेस को और छोटा लिखा जा सकता है एक समूह में अग्रणी शून्य हटाया जा सकता है। शून्य मूल्य में से एक या अधिक लगातार समूहों लगातार दो कॉलन का उपयोग करते हुए एक भी खाली समूह के साथ प्रतिस्थापित किया जा सकता है।

2001:db8:85a3:0:0:8a2e:370:7334

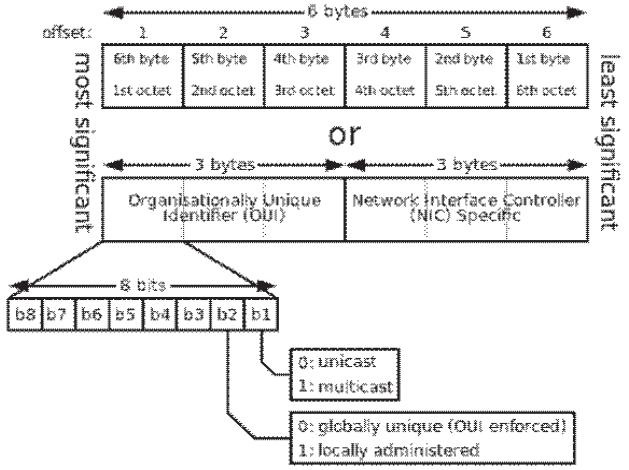
2001:db8:85a3::8a2e:370:7334

4.11 मैक एड्रेस (MAC Address)

मैक एड्रेस (MAC Address) एक अद्वितीय एवं भौतिक पता होता है जो की किसी नेटवर्क कार्ड के भौतिक नेटवर्क में कम्युनिकेशन के लिए दिया जाता है। मैक एड्रेस का उपयोग IEEE Network तकनीको जैसे ईथरनेट (Ethernet) तथा वायरलेस (Wireless) में किया जाता है यह डेटा लिंक लेयर की सबलेयर पर काम करता है। मैक एड्रेस किसी भी इन्टरफेस कार्ड को बनाने वाली संगठन द्वारा दिया जाता है और हार्डवेयर पर स्थित रीड ओनली चिप में स्टोर रहता है। इसे बदला नहीं जा सकता। मैक एड्रेस Institute of Electrical and Electronics Engineers (IEEE) के अनुसार बने मानको के अनुसार होते हैं जो निम्नलिखित है – MAC-48, EUI-48 and EUI-64.

मैक एड्रेस को बर्न एड्रेस या हार्डवेयर एड्रेस भी कहा जाता है। मैक एड्रेस 48 बिट का एड्रेस होता है

जिसे हेक्साडेसीमल के 2 नंबर्स के 6 समूहों में व्यवस्थित किया गया है। यह समूह हाईफन (–) से अलग रहते हैं। मैक एड्रेस सामान्यतया दो भागों में विभाजित रहता है जिसके प्रथम 3 समूह Institute of Electrical and Electronics Engineers (IEEE) द्वारा किसी संगठन को दिए जाते हैं और अंतिम 3 समूह किसी भी संगठन द्वारे बनाये गए कार्ड पर सीरियल नंबर की तरह होते हैं, इस प्रकार कोई भी मैक एड्रेस किसी अन्य मैक एड्रेस से मिलता नहीं है।



चित्रः 4.24 मैक एड्रेस

4.12 सबनेटिंग (**Subnetting**)

यह हमे पहले से ही ज्ञात है IP की अच्छी प्लानिंग, इम्प्लीमेंटेशन और अच्छे प्रबंधन से नेटवर्क अच्छी तरह एवं सुचारू तरीके से चलता है। IPV4 में दो प्रकार का पदक्रम (Hierarchy) होता है जो कि होस्ट एवं नेटवर्क से सम्बंधित है। कोई भी राऊटर नेटवर्क से सम्बंधित हिस्से की जानकारी से ही किसी भी पैकेट को एक नेटवर्क से दूसरे नेटवर्क मे भेजता है। एक बार नेटवर्क हिस्से का पता लगने के बाद होस्ट एड्रेस केवल उस कंप्यूटर का पता करने के काम आता है जिसको पैकेट भेजना है। जैसे जैसे संगठनो में नेटवर्क की संख्या बढती है यह दो प्रकार कर पदक्रम ना काफी लगने लगता है। अब नेटवक पदक्रम को दुबारा विभाजित करना आवश्यक हो जाता है। किस भी नेटवर्क को अब तीन पदक्रम में विभाजित कर सकते हैं– नेटवर्क, सबनेटवर्क (Subnetwork) और होस्ट। किसी भी नेटवर्क को विभिन्न पदक्रम में विभाजित करने से डेटा पैकेट्स तीव्र गति से संचरित होते हैं और यह नेटवर्क के सह समूह बना देता है।

यह भी हमे पहले से ज्ञात है की IP का कौनसा भाग नेटवर्क से और कौनसा भाग होस्ट से सम्बंधित है इसका पता सबनेट मास्क से करते हैं और क्लासेज के अनुसार पहले से ही सबनेट मास्क बने हुए हैं।

किसी भी नेटवर्क की सबनेटिंग : हर नेटवर्क के पास होस्ट एड्रेसेस की एक वैध श्रंखला होती है। एक ही नेटवर्क में उपस्थित सभी कंप्यूटर या उपकरण या होस्ट एक ही सबनेट मास्क रखते हैं और उस नेटवर्क के मेम्बर होते हैं। IPV4 एड्रेस में 32 बाइनरी बिट्स होती है जो नेटवर्क तथा होस्ट से सम्बंधित होती है। सबनेटिंग, होस्ट बिट्स को नेटवर्क को देने से होती है। किसी भी नेटवर्क के कितने सबनेटवर्क बनेगे यह इस पर निर्भर करता है की कितनी बिट होस्ट से नेटवर्क की दी है या कितनी बिट नेटवर्क ने होस्ट से उधर ली है।

उदाहरण के लिए अगर 1 बिट ली है तो 2 सबनेटवर्क, 2 बिट ली है तो 4, 3 बिट ली है तो 8 सबनेटवर्क बनेंगे। जैसे जैसे होस्ट बिट्स कम होंगी उसके अनुसार किसी भी नेटवर्क मे होस्ट एड्रेस भी कम हो जायेंगे। क्लास C के एक नेटवर्क मे 24 बिट नेटवर्क की तथा 8 बिट होस्ट की होती है अगर होस्ट की 8 बिट्स मे से 1 बिट नेटवर्क को दी जाए तो अब होस्ट में 128 होस्ट एड्रेस ही बचेंगे। मतलब पहले इस नेटवर्क में 256 होस्ट थे और यह एक नेटवर्क था। अब इस नेटवर्क के दो टुकड़े हो गए हैं तथा होस्ट एड्रेसेस के टुकड़े हो गए हैं। हर सब नेटवर्क मे 128– 128 होस्ट एड्रेस होंगे और अब नेटवर्क मे 24 के वजाए 25 बिट होंगी। नेटवर्क बिट्स को किसी भी एड्रेस के बाद स्लैश (/) में भी लिखा जा सकता है जैसे की 192.168.1.0 / 24।

"अगर सरल भाषा में कहा जाए तो सबनेटिंग बिट्स का इधर से उधर होना है"।

किसी भी नेटवर्क में पहला एड्रेस नेटवर्क एड्रेस जो की किसी भी नेटवर्क का प्रतिनिधित्व करता है तथा अंतिम एड्रेस ब्रॉडकास्ट (Broadcast) एड्रेस होता है जो की किसी भी नेटवर्क के सभी उपकरणों को सूचना भेजने में काम आता है। मानकों के अनुसार नेटवर्क एड्रेस तथा ब्रॉडकास्ट एड्रेस को भी उपकरण को सौंप (Assign) नहीं सकते। इसके अलावा के एड्रेसेज को कंप्यूटर या अन्य उपकरणों को सौंपा जा सकता है। इसके अनुसार किसी भी नेटवर्क से 2 एड्रेस काम मे नहीं लिए जा सकते।

उदाहरण के लिए अगर किसी नेटवर्क के होस्ट भाग में 8 बिट है तो उसमें कुल 256 होस्ट एड्रेस होंगे जिनमे से 2 को काम नहीं ले सकते तो 254 एड्रेसेज ही काम आ सकते हैं। इन सब की गणना उसकी क्लास के अनुसार होती है क्योंकि हर क्लास मे अलग अलग संख्या में बिट्स होस्ट भाग में होती है।

सबनेटिंग का प्रश्न दो प्रकार से पूछा जा सकता है– नेटवर्क की जरुरत के अनुसार और होस्ट की जरुरत के अनुसार।

प्रश्न में जिसकी भी जरूरत पूछी है उसके अनुसार गणना की जाती है कि कितनी बिट्स होस्ट भाग मे रखनी है या कितनी बिट्स नेटवर्क भाग को देनी है। जिसकी भी जरुरत हो उसकी संख्या को 2 की घात में गणना करते है उसके अनुसार उतनी ही बिट देते या रखते है।

सबनेटिंग को हम कुछ उदाहरण से समझेंगे–

1. 192.168.1.0 / 24 नेटवर्क की सबनेटिंग इस प्रकार करे की इसके चार नेटवर्क बन जाए

   नेटवर्क की क्लासः

   क्लास का सबनेट मास्कः

   कितनी बिट रखनी या देनी हैः

   कितने नेटवर्क बनेः

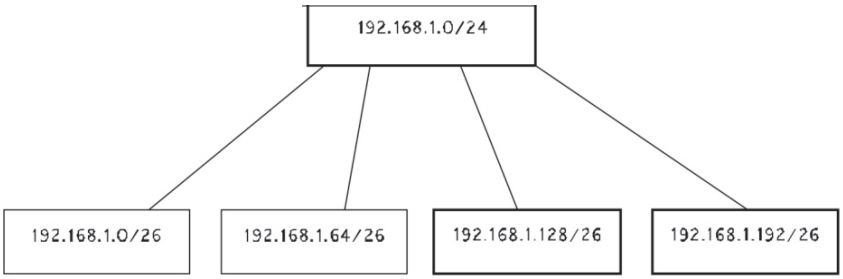
   कितने होस्ट नेटवर्क में बचेः

   इस नेटवर्क की सबनेटिंग करते समय ऊपर लिखे सवालों के जवाब देने है।

उपाय : नेटवर्क की क्लास C है जिसे इस एड्रेस के प्रथम ओकटेट से पहचाना। क्लास C का सबनेट मास्क 255.255.255.0 होता है जो हमने पहले पढ़ा है। कितनी बिट देनी या लेनी है इसका अनुमान हम हमारी जरुरत से लगायेंगे, हमारी जरुरत 4 सब नेटवर्क बनाना है और 4, 2 की घात 2 मे आता है मतलब हमें 2 बिट नेटवर्क को देनी है। बिना सबनेटिंग के इस नेटवर्क मे 24 बिट नेटवर्क की तथा 8 बिट होस्ट की है अब हमे 2 बिट होस्ट से नेटवर्क को देनी है तो अब नेटवर्क मे 26 बिट तथा होस्ट में 6 बिट रह जाएँगी। अब नेटवर्क एड्रेस को निम्न प्रकार से लिखा जा सकता है 192.168.1.0 / 26.

अब इस नेटवर्क के 4 हिस्से या सबनेटवर्क बन गए है और होस्ट एड्रेस बराबर बराबर हिस्सों में बट

गए है। होस्ट हिस्से मे 6 बिट बची है मतलब 2 की घात 6, हर नेटवर्क मे कुल 64 एड्रेस होंगे जिसमे से 2 को उपयोग मे नहीं लाया जा सकता है, 62 एड्रेसेज को ही कंप्यूटर या उपकरणों को सौंपा जा सकता है।

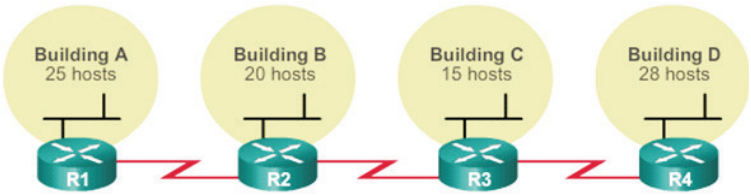


चित्रः 4.25 सबनेटिंग

4.13 IP एड्रेस की योजना (Planning of IP Address)

IP एड्रेस की योजना एक महत्वपूर्ण काम है जिसका सही तरीके से होना जरुरी है। IP एड्रेस किसी भी संगठन मे नेटवर्क के उपलक्ष मे बहुत ही कीमती संसाधन है जिसका उपयोग जरुरत के अनुसार करना जरुरी है। अगर किसी भी नेटवर्क के एड्रेस खत्म हो गए तो नए कंप्यूटर या उपकरण जोड़ें नहीं जा सकते हैं।

पारंपरिक सबनेटिंग मे सभी सबनेटवर्क्स को सामान संख्या में एड्रेस दिए जाते है। यह तभी अच्छा है जब सभी नेटवर्क्स की जरुरत सामान हो जो सामान्यतया नहीं होता है।



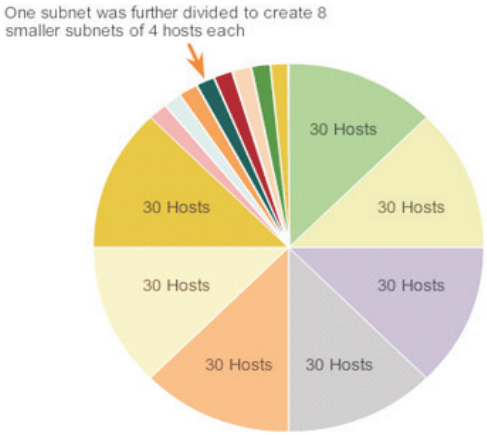
चित्रः 4.26 नेटवर्क्स विभिन्न होस्ट की जरुरत के साथ

चित्र 2.15 के अनुसार कुल 7 नेटवर्क की जरुरत है जिसमें 3 वाइड एरिया नेटवर्क (WAN) तथा 4 लोकल एरिया नेटवर्क है। पारंपरिक सबनेटिंग के अनुसार अगर विभाजन किया जाए तो 3 बिट होस्ट से नेटवर्क को देनी पढ़ेगी जिससे कुल 8 नेटवर्क बनेंगे तथा हर नेटवर्क मे 2 की घात 5, 32 एड्रेसेज होंगे और इससे हमरी जरुरत पूरी होती है।

लेकिन इस प्रकार विभाजन करने से बहुत सारे एड्रेस बेकार जाते है जिनका कोई उपयोग नहीं होगा। चित्र 4.26 के अनुसार 3 वाइड एरिया नेटवर्क है जिसमें 2 राऊटर आपस मे जोड़ने है तो 2 एड्रेस की जरुरत होगी लेकिन राऊटर का हर इंटरफेस एक अलग नेटवर्क बनता है तो 2 अन्य एड्रेस (नेटवर्क और ब्रॉडकास्ट), कुल 4 की जरुरत है लेकिन हम इसको 32 एड्रेस दे रहे हैं जिसके फलस्वरूप 28 एड्रेस बेकार जा रहे है।

इसे वेरिएबल लेंग्थ सबनेट मास्क (VLSM) तकनीक से बचाया जा सकता है। VLSM

तकनीक किसी भी नेटवर्क को असमान तरीके से विभाजित करने मे मदद करती है। VLSM तकनीक पारंपरिक सबनेटिंग जैसी ही है परन्तु इसमे पहले एक नेटवर्क की सबनेटिंग की जाती है फिर सबनेटेड़ (Subneted) नेटवर्क की दुबारा सबनेटिंग करते है जिससे असमान सबनेट मास्क के नेटवर्क प्राप्त होते है तथा एड्रेसेज का बिना किसी उपयोग बेकार जाना एक हद तक बंद हो जाता है।



चित्र: 4.27 टस्टैड द्वारा नेटवर्क की सबनेटिंग

4.14 बेतार सम्प्रेषण (Wireless)

वायरलेस नेटवर्क किसी भी उपकरण को बिना तार किसी माध्यम से जुड़ने की सुविधा देता है। वायरलेस लैन (W LAN), वायरलेस तकनीक का एक वर्गीकृत भाग जो सामान्यतया घरों, दफ्तरों तथा कैंपस मे काम लिया जाता है। यह तकनीक तारों के वजाए रेडियो तरंगो को काम लेती है। यह पहले से स्थित LAN नेटवर्क मे जोड़ी जा सकती है जिससे यूजर उस क्षेत्र मे कभी भी वायरलेस की सुविधा का उपयोग कर सकता है। यह तकनीक ईथरनेट (Ethernet) से मिलती जुलती है।

वायरलेस तकनीक एवं मानक (Standards) : आज की समय मे उत्पादकता अब कोई एक निश्चित स्थान या काम एक निर्धारित समय अवधि के लिए प्रतिबंधित नहीं है, लोगों को अब हवाई अड्डे या घर के लिए किसी भी समय और स्थान पर कार्यालय से जुड़े होने की उम्मीद रहती है। अब कर्मचारी कभी भी अपने ईमेल, वाईस मेसेज तथा नई जानकारी को कभी भी देख सकते हैं। यूजर्स अब यह अपेक्षा रखते है किसी को बिना संपर्क टूटे हमेशा इन्टरनेट से जुड़े रहे।

वायरलेस के फायदे

1. यह कार्य करने मे लचीलापन (Flexibility), उत्पादकता को बढाने, आगे बढने तथा जरुरत के अनुसार अनुकूल वातावरण बनती है।
2. कोई भी यूजर कभी भी कही भी, इन्टरनेट से जुड़ सकता है उसे किसी एक क्षेत्र या जगह पर होने की आवश्यकता नहीं है।
3. वायरलेस से उत्पादकता मे आ रहे खर्च मे कमी आती है।

इसकी निम्न तकनीक है–

1. **Wireless Personal Area Networks (WPAN)** : यह तकनीक कुछ फीट के क्षेत्र मे ही काम करती है। ब्लूटूथ (Bluetooth) इसका उदाहरण है। Bluetooth की मदद से यूजर जिनके

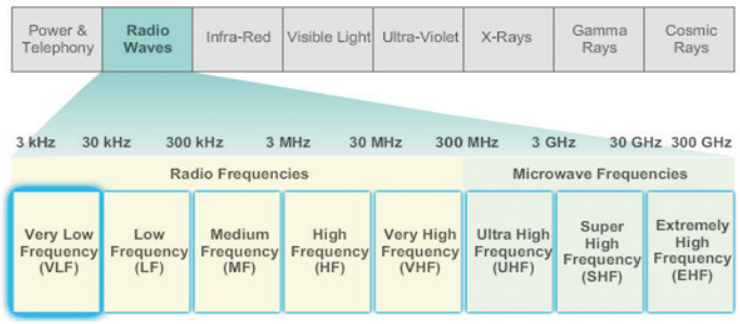
पास मोबाइल है वो कभी भी डेटा एक–दूसरे को भेज सकते है।

2. **Wireless LANs (WLANs)** : यह तकनीक कुछ 100 मीटर के क्षेत्र मे ही काम करती है। जिससे घर, दफ्तर तथा कैंपस मे इन्टरनेट से जोड़ा जा सकता है।

3. Wireless Wide-Area Networks (WWANs) : यह तकनीक बहुत लम्बी दूरीके क्षेत्र मे काम करती है। इस तकनीक के द्वारा घर, शहर, देशो को जोड़ा जाता है। उपग्रह तथा मोबाइल कम्युनिकेशन इसके उदाहरण है।

सभी वायरलेस उपकरण विद्युत चुम्बकीय वर्णक्रम (Electromegnatic Spectrum) के रेडियो तरंगों की सीमा में कार्य करते हैं। रेडियो आवृति का नियमन एवं आवंटन इंटरनेशनल टेलीकम्युनिकेशन यूनियन (ITU) की जिम्मेदारी है। विभिन्न आवृतियाँ तथा बैंड (Band) की सीमाएं विभिन्न प्रयोजनों के लिए आवंटित होती है। यह आवृतिया ज्यादातर भुगतान के बाद मिलती है जबकि कुछ आवृतिया मुक्त है जैसे की Industrial, Scientific, and Medical (ISM) और National Information Infrastructure (UNII) frequency bands.

WLAN (वायरलेस लैन), ISM band के 2.4 GHZ तथा UNII के 5GHZ पर काम करती है। वायरलेस कंयुनिकेशन रेडियो तरंगो के 3 HZ से 300 GHZ की चुम्बकीय वर्णक्रम मे काम करता है।



चित्र: 4.28 चुम्बकीय वर्णक्रम

वायरलेस लैन इन उपकरणों के पास ट्रांसमीटर और रिसीवर कुछ चुनिंदा आवृतियों पर काम करने के लिए होते हैं जो निम्न है–

- 2.4GHZ (802-11b/g/n/ad)
- 5 GHZ (802-11a/n/ac/ad)
- 60 GHZ (802-11ad)

4.15 वायरलेस के मानक (Standards of Wireless)

IEEE 802.11: WLAN मानक को परिभाषित करता है कि कैसे बिना लाइसेंस ISM आवृत्ति (Frequency) बैंड में आरएफ (RF) भौतिक परत और वायरलेस लिंक के मैक उपस्तर के लिए प्रयोग किया जाता है।

IEEE802.11 मानक के विभिन्न कार्यान्वयन (Implementation) को वर्षों में विकसित किया गया है। निम्नलिखित इन मानकों पर प्रकाश डाला गया है:–

802.11: यह 1997 में विकसित हुई तथा अप्रचलित है यह तकनीक 2.4GHZ पर काम करती है और अत्यधिक 2 Mbps की गति प्रदान करती है सामान्यता लेन (LAN) 100 Mbbs पर काम करती थी

जबकि यह 2 Mbps पर काम करती थी इसलिए इसे ज्यादा प्रयोग में नहीं लाया गया तथा इस में काम आने वाले उपकरणों पर एक ऐन्टेना (Antenna) लगा होता था जो प्रेषक और प्राप्तकर्ता का काम करता था।

IEEE 802.11a : यह 1999 में प्रदर्शित (Release) हुई यह 5 GHZ कम भीड़ वाली आवृति पर काम करती है और 54 Mbps की गति प्रदान करती है। यह अधिक आवृति पर काम करती है इसलिए इसका आवृत्त क्षेत्र कम होता है तथा यह दीवारों को कम भेद पाती है।

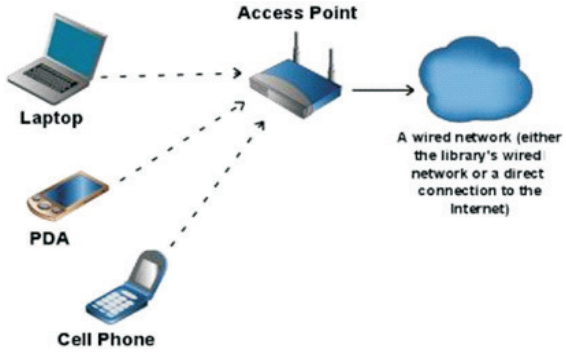
IEEE 802.11b : यह 1999 मे प्रदर्शित (Release) हुई। यह 2.4GHZ की आवृति पर काम करती है तथा 11 Mbps की गति प्रदान करती है।

IEEE 802.11g : यह 1993 मे प्रदर्शित (Release) हुई। यह 2.4GHZ की आवृति पर काम करती है तथा 54Mbps की गति प्रदान करती है। यह पिछली तकनीक IEEE 802.11b के साथ संगत (Compatible) है।

IEEE 802.11n : यह 2009 मे प्रदर्शित (Release) हुई। यह 2.4 व 5 GHZ दोनों आवृतियों पर काम करती है। यह तकनीक 150 से लेकर 600 Mbps की गति प्रदान करती है। इसमे काम मे आने वाली उपकरण पर एक से ज्यादा ऐन्टेना लगे होते है जिसमे MIMO (Multi in Multi Out) तकनीक होती है जिससे वो वेहतर डेटा ट्रांसमिशन करते है। यह 4 ऐन्टेना तक समर्थन (support) करती है। यह पिछली तकनीको 802.11a/b/g के साथ संगत (Compatible) है।

IEEE 802.11ac : यह 2013 मे प्रदर्शित हुई। यह 5 GHZ की आवृति पर काम करती है तथा 450Mbps से 1.3Gbps की गति प्रदान करती है। यह पिछली तकनीको 802.11a/n के साथ संगत (Compatible) है।

IEEE 802.11ad : यह 2014 मे प्रदर्शित हुई तथा इसे WiGig भी कहते है। यह 2.4, 5 व 60 GHZ की आवृति पर काम करती है। और 7 Gbps तक की गति प्रदान कर सकती है अतः यह सबसे तेज वायरलेस तकनीक है।



चित्र 4.29 वायरलेस नेटवर्क

वायरलेस सुरक्षा (**Wireless Security**) : एक वायरलेस नेटवर्क से जुड़े नेटवर्क को सुरक्षित बनाये रखना कठिन है। जो कोई भी नेटवर्क प्रशासन को उपयोग मे ला रहा है उसके लिए सुरक्षा एक महत्वपूर्ण प्राथमिकता है।

वायरलेस नेटवर्क में उससे सम्बंधित खुफिया जानकारी जहाँ तक नेटवर्क आ रहा है वहा तक खुली रहती है या पता होती है। एक हमलावर को शारीरिक रूप से एक WLAN के लिए पहुँच प्राप्त करने

के लिए कार्यस्थल में प्रवेश करने की आवश्यकता नहीं होती है। इसलिए वायरलेस नेटवर्क को सुरक्षा देना बहुत ही आवश्यक है।

वायरलेस पर निम्न प्रकार आक्रमण हो सकते है–

- Wireless intruders
- Rogue apps
- Interception of data
- DoS attacks

वायरलेस को सुरक्षा प्रदान करने के तरीका (**Securing WLANs**)

सुरक्षा हमेशा ही चिंता का विषय है क्योकि नेटवर्क की सीमा की सीमा बदलती रहती है। वायरलेस के संकेत छत, दीवार, घर के बहार या कार्यालय के ठोस माध्यमो से यात्रा कर सकते है। बिना सुरक्षा के वायरलेस किसी भी ईथरनेट नेटवर्क को खुला उपयोग मे छोडने के बराबर है।

वायरलेस को सुरक्षा प्रदान करने के निम्न सुरक्षा व्यवस्थाएँ है–

SSID cloaking : इसमें वायरलेस उपकरण द्वारा छोडे वाले संकेतो (beacon) फ्रेम्स को बंद कर दिया जा है जिससे वायरलेस का नेटवर्क किसी भी उपकरण मे दिखाई नहीं देता है। जिस उपकरण को वायरलेस नेटवर्क से जोड़ना हो तो वायरलेस की पूर्ण जानकारी के बाद ही जोड़ा जा सकता है अतः कोई भी नेटवर्क का पता नहीं लगा सकता है।

MAC addresses filtering: वायरलेस राऊटर मे भौतिक मैक एड्रेस (MAC Address) की एक सूची बना सकते है जिससे किसी भी कंप्यूटर या उपकरण को उस नेटवर्क से जुड़ने के लिए बंद व अनुमति दी जा सकती है।

उपरोक्त दोनों ही तरीको को तोड़ा जा सकता है इसलिए वायरलेस नेटवर्क को एन्क्रिप्ट और प्रमाणित (authenticate) करना आवश्यक है इसलिए 802.11 मानक मे दो प्रकार के तरीके है जो निम्न है –

Open system authentication: इसमे कोई भी ग्राहक या यूजर आसानी से कनेक्ट कर सकता है जहाँ सुरक्षा कोई ज्यादा मायने नहीं रखती। इस प्रकार की सुरक्षा कैफे, होटल की तरह मुफ्त इंटरनेट का उपयोग प्रदान स्थानों में के रूप में , जहां सुरक्षा कोई मायने नहीं रखी जाती है।

Shared key authentication: इसमें कंप्यूटर और राऊटर के मध्य के कम्युनिकेशन को एन्क्रिप्ट (Encrypt) करने के लिए राऊटर मे एक ज़मल या पासवर्ड डालते है जिसकी जानकारी के बिना कोई भी ग्राहक या यूजर कनेक्ट नहीं कर सकता। डेटा को एन्क्रिप्ट (Encrypt) या छुपाने के लिए निम्न सुरक्षा व्यवस्थाएँ है।

7.5.1 WEP (Wired Equivalent Privacy) : यह ऑथेंटिकेशन (authentication) की सबसे प्रथम पीढ़ी (generation) है। यह स्थिर कुंजी (Key) को पासवर्ड के रूप में प्रयोग मे लाती है। यह RC4 अल्गोरिथम को काम मे लेता है।

7.5.2 WPA: स्ट्रोंग सुरक्षा के लिए यह Temporal Key Integrity Protocol (TKIP) encryption algorithm का प्रयोग करती है।

7.5.3 WPA2 : यह इंडस्ट्री मानक के अनुसार वायरलेस नेटवर्क को सुरक्षा प्रदान करने का तरीका है यह TKIP की जगह AES (Advance Encryption Standard) को काम मे लेती है जो शक्तिशाली तरीका है।

4.16 कन्जेसन कण्ट्रोल (Congestion Control)

कन्जेसन कण्ट्रोल वो तकनीक व तरीका है जो नेटवर्क मे पैकेट्स की भीड़ या तो होने से पहले ही ठीक कर देता है अर्थात ऐसा होने नहीं देता या होने के बाद उसे ठीक कर देता है। यह दो प्रकार का होता है

1. ओपन लूप कन्जेसन कण्ट्रोल (Prevention)
2. क्लोज लूप कन्जेसन कण्ट्रोल (Removal)

ओपन लूप कन्जेसन कण्ट्रोल : इस प्रकार के तरीके मे विभिन्न नीतियां(Rules),कन्जेसन को रोकने के लिए प्रयोग मे लायी जाती है। इस तकनीक मे कन्जेसन या तो प्रेषक या फिर प्राप्तकर्ता के स्थान पर रोका जाता है।

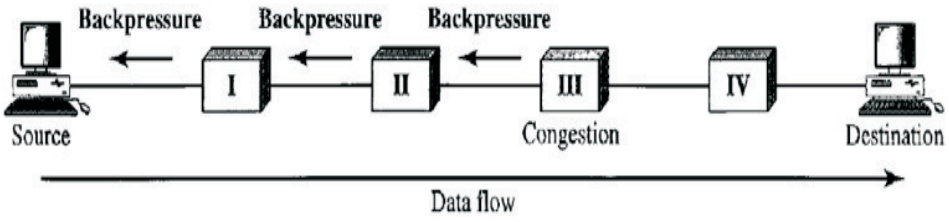
इस तकनीक मे प्रयोग मे लाने वाली नीतियां निम्लिखित हैं–

1. रिट्रांसमिशन नीति (**Retransmission Policy**): इस नीति मे अगर प्रेषक को लगता है की उसके द्वारा भेजा गया पैकेट रास्ते मे नष्ट हो गया है तो उस पैकेट की दुबारा भेजा जाता है, परन्तु यह कन्जेसन उत्पन्न कर सकता है। इसे रोकने के लिए अच्छी नीति की आवश्यकता होती है। इसमें एक घडी का प्रयोग किया जाता है ताकि कन्जेसन ना हो। TCP प्रोटोकॉल पहले से ही इस प्रकार से बना हुआ की कन्जेसन उत्पन्न होने की सम्भावना कम होती है।

2 एकनॉलेजमेंट नीति (**Acknowledgement Policy**) : इस नीति मे अगर प्रेषक भेजे गए हर पैकेट का पुष्टिकरन (Acknowledgement) की उम्मीद रखता है। अगर किसी पैकेट का प्राप्तकर्ता की तरफ प्राप्त होने का पुष्टिकरन नहीं होता है तो प्रेषक पैकेट भेजने की गति कम कर देता है जिससे कन्जेसन की सम्भावना कम हो जाती है।

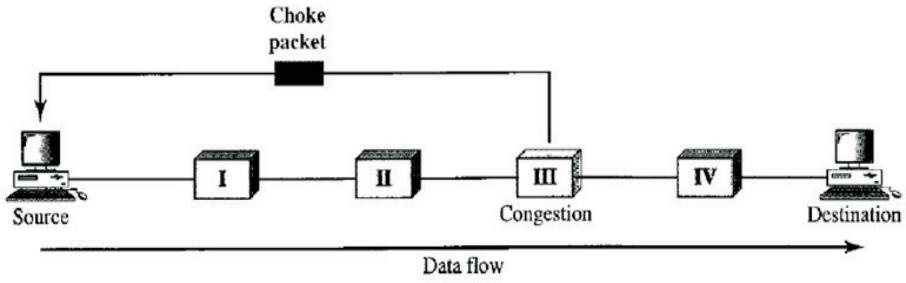
क्लोज लूपकन्जेसन कण्ट्रोल : ओपन लूपकन्जेसन कण्ट्रोल में कन्जेसन होने के बाद उसे ठीक किया जाता है विभिन्न प्रोटोकॉल्स द्वारा विभिन्न तकनीको का प्रयोग किया जाता है।

1 बेकप्रेशर (**Backpressure**) : इस तकनीक मे जिस नोड (Node) पर कन्जेसन होता है वह नोड अपने से पहले वाली नोड से डेटा प्राप्त करना बंद कर देती है जिससे पहले वाली नोड पर कन्जेसन बढ़ सकता है पर ऐसा ही पहले वाली नोड करती है। इसे नोड– नोड कन्जेसन कण्ट्रोल कहते है। यह तकनीक केवल काल्पनिक पथ (Virtual Circuit) नेटवर्क मे प्रयोग मे लायी जाती है।



चित्र: 4.30 बेकप्रेशर कन्जेसन कण्ट्रोल

2 चोक पैकेट (Choke Packet) : इस तकनीक मे एक विशेष प्रकार का पैकेट चोक पैकेट प्रेषक को भेजा जाता है। यह लगभग बेकप्रेशर तकनीक जैसा ही है परन्तु बेक प्रेशर मे सूचना अपने से पहले वाली नोड को भेजी काटी है इसमें सूचना सीधे ही प्रेषक को भेजी जाती है तथा सूचना बीच वाली नोड को नहीं भेजी जाती है।



चित्रः 4.31 चोक पैकेट कन्जेसन कण्ट्रोल

4.17 सेवा की गुणवत्ता (**Quality of Service**)

कंप्यूटर नेटवर्क्स मे सेवा की गुणवत्ता का मतलब किसी भी प्रकार के कम्युनिकेशन के लिए संसाधन आरक्षण (Resource Reservation) है। सेवा की गुणवत्ता का काम किसी यूजर, एप्लीकेशन, डेटा फ्लो को उच्च गुणवत्ता की प्राथमिकिता देना है। सेवा की गुणवत्ता के अनुसार कुछ कारक जैसे की बिट रेट (Bit Rate), डिले (Delay), जिटर (Jitter) और बिट एरर रेट (Bit Error Rate) गुणवत्ता के अनुसार होने चाहिए।

सामान्यतया कहा जाए तो सेवा की गुणवत्ता का मतलब उच्च स्तर की सेवा प्रदान करना है।

प्रवाह विशेषताए : मुख्यतया 4 प्रकार की प्रवाह विशेषताए हैं।

1 विश्वसनीयता (**Reliability**) : विश्वसनीयता एक विशेषता है। अगर विश्वसनीयता कम है तो इसका सीधा मतलव यह नही की पैकेट/अभिस्वीकृति (acknowledgement) का नष्ट होना है जो रिट्रांसमिशन (दुबारा भेजना) को प्रेरित करता है।

2 विलम्ब (**Delay**) : प्रेषक से प्राप्तकर्ता के बीच के विलम्ब को एक हद तक सहन कुछ एप्लीकेशनस मे किया जा सकता है। परन्तु कुछ एप्लीकेशनस जैसे की टेलीफोनी और ऑडियो कांफ्रेसिंग (Audio conferencing) मे विलम्ब सहन नहीं किया जा सकता है।

3 जिटर (**Jitter**) : जिटर का सम्बन्ध एक ही प्रवाह से संवंधित पैकेट्स मे विलम्ब से है। अगर सभी पैकेट प्रेषक से प्राप्तकर्ता तक एक ही समय विलब पर पहुच रहे है मतलब जिटर नहीं है परन्तु अगर पैकेट्स अलग अलग विलम्ब पर पहुच रहे है तो जिटर है जो नेटवर्क मे जाने वाले पैकेट्स के लिए तथा एप्लीकेशनस के लिए अच्छा नहीं है।

4 बैंडविड्थ (**Bandwidth**) : बैंडविड्थ का सीधा सम्बन्ध एक समय मे भेजे गए पैकेट्स की गति से है। अलग अलग एप्लीकेशनस को अलग अलग बैंडविड्थ की जरुरत होती है। अगर बैंडविड्थ जरुरी बैंडविड्थ से ज्यादा या बराबर है तो एप्लीकेशन सही तरीके से काम करेंगी अन्यथा पैकेट्स का नष्ट होने शुरू हो जायेगा जिसके फलस्वरूप उन पैकेट्स को दुबारा भेजना पड़ेगा जो नेटवर्क तथा कंप्यूटर एप्लीकेशनस के लिए अच्छा नहीं है।

4.17.1 सेवा की गुणवत्ता को सुधारने के लिए तकनीक

1. Scheduling
2. FIFO Queuing
3. Priority Queuing
4. Weighted Fair Queuing

4.18 DNS (Domain Name Service)

4.18.1 प्रस्तावना : DNS एक श्रेणीबद्ध (Hierarchal) वितरित प्रणाली है जिससे डोमेन नाम को IP मे बदलते है। कंप्यूटर, मानव जनित नाम को नहीं समझाता और बाइनरी मे काम करता है जबकि मानव बाइनरी नहीं समझता और अपनी भाषा में काम करता है इसलिए यह जरुरी हो जाता है की ऐसा कोई प्रणाली हो जो मानव भाषा के नामो को कंप्यूटर नेटवर्क के एड्रेस IP मे बदल सके।

उदाहरण के लिए www.google.co.in का IP 216.58.196.3, यह DNS से ही मुमकिन है।

4.18.2 इतिहास (**History**) : इन्टरनेट के शुरूआती दिनों मे विश्व मे कुछ कंप्यूटर ही हुआ करते थे जिनके IP एड्रेस आपस मे पता होते थे। बाद मे इन कंप्यूटर्स को मानव जनित या मानव भाषा के नाम दे दिए गए जिसका IP में रूपांतरण कंप्यूटर के ऑपरेटिंग सिस्टम की फाइल होस्ट्स (Hosts.txt) से होने लगा जिसमे कंप्यूटर के नाम के सामने उसका IP एड्रेस लिखा होता था।

परन्तु जैसे जैसे इन्टरनेट प्रचार मे आयी यह काम भी मुश्किल होने लगा अतः अब एक ऐसे प्रणाली की आवश्यकता थी जिससे यह काम आसानी से हो सके। पॉल मोकापेट्रिस ने 1983 में कैलिफोर्निया विश्वविद्यालय, इरविन में डोमेन नेम सिस्टम तैयार किया, और आईएसआई से जॉन पोस्टेल के अनुरोध पर पहला कार्यान्वयन लिखा था।

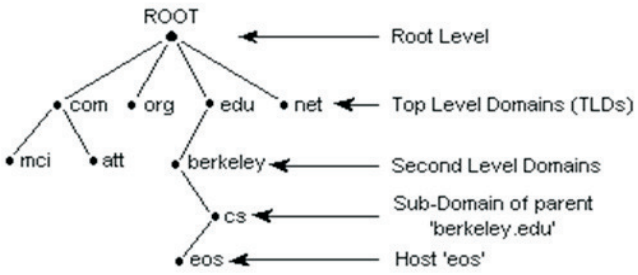
4.18.3 डोमेन नेम स्पेस (**Domain name space**) : डोमेन नाम सिस्टम एक पेड़ जैसी डेटा संरचना है। यह पौधानुमा संरचना कई जोंस मे विभाजित होती है जो एक रूट या डॉट (.) से शुरू होती है। रूट डोमेन या डॉट (.) कई श्रेणियों का बंटा होता है जिन्हें टॉप लेवल डोमेन (Top Level Domain) कहा जाता है।

उदाहरण के लिए com, org, net

टॉप लेवल डोमेन के नीचे पेड़नुमा संरचना मे सेकेंडरी लेवल डोमेन्स होते हैं जिन्हें यूजर अपनी आवश्यकता चुन सकते हैं। एक डोमेन नाम सभी श्रेणियों मे होता है अगर कोई डोमेन नाम एक श्रेणी मे ना तो उसे किसी दूसरी श्रेणी से लिया जा सकता है। सेकेंडरी लेवल डोमेन सबडोमेन्स में विभाजित होते है जिन्हें यूजर अपनी इच्छा के अनुसार अपना सकते है।

mail.google.com

यहाँ मेल सब डोमेन है, गूगलसेकेंडरी लेवल डोमेन नाम है तथा कॉम प्टॉप लेवल डोमेन है यह सभी डॉट (.) से विभाजित रहते हैं।



चित्रः 4.32 डोमेन नाम सिस्टम

प्रत्येक DNS सर्वर किसी भी डोमेन के बारे मे जानकारी जोन फाइल मे रखता है। किसी भी सर्वर के पास निम्नलिखित संसाधन रिकॉर्ड (Resource Record) होते है।

1. NS रिकॉर्ड : यह रिकॉर्ड किसी भी डोमेन की अधिकार (Authority) की जानकारी रखता है।

2. A रिकॉर्ड : ये रिकॉर्ड किसी भी डोमेन की IP का विवरण रखता है।

3. CNAME रिकॉर्ड : यह रिकॉर्ड किसी भी डोमेन के क्कैनोनिकल नाम अर्थात उसके अन्य नाम की जानकारी रखता है।

4. MX रिकॉर्ड : यह उसके मेल रिकॉर्ड की जानकारी रखता है।

हर कंप्यूटर या नेटवर्क उपकरण मे DNS की प्रविष्टी डाली जाती है ताकि वह डोमेन नाम से IP का पता लगा सके। अगर किसी भी कंप्यूटर या नेटवर्क उपकरण में DNS की प्रविष्टी नहीं तो वह डोमेन नाम से किसी भी अन्य कंप्यूटर या उपकरण को ढूँढ या उससे संपर्क नहीं कर पायेगा। किसी भी कंप्यूटर या नेटवर्क उपकरण का किसी भी कंप्यूटर की IP पता करने का निम्न तरीका होता है।

कोई भी कंप्यूटर किसी भी अन्य कंप्यूटर से संपर्क साधने के लिए अपनी DNS की प्रविष्टी में DNS सर्वर से संपर्क करती है तथा अपनी अपना सवाल (डोमेन की IP) पूछता है अगर उस DNS सर्वर के पास उस डोमेन का पता होता है तो वह उसको जवाब दे देता है।

अगर उस सर्वर के पास उस डोमेन का जवाब नहीं होता है तो वह अपने ऊपर वाले या बाद के DNS सर्वर से पूछता है।

अगर कोई भी सर्वर जवाब नहीं देता है तो रूट सर्वर से पुछा जाता है।

4.19 ईमेल सिस्टम (Email System)

ईमेल इन्टरनेट सेवाओ की सबसे प्रसिद्ध सेवा है। इन्टरनेट के शुरूआती दिनों मे ईमेल से केवल छोटे व टेक्स्ट मेसेज ही भेजे जा सकते थे। पर आजकल ईमेल बहुत ही जटिल प्रणाली है जिसकी मदद से टेक्स्ट, चित्र, विडियो को भी भेजा जा सकता है। ईमेल से जानकारी एक साथ एक से ज्यादा प्राप्तकर्ता को भेजी जा सकती है।

ईमेल की सेवाए : ईमेल की निम्नलिखित सेवाए है–

1 इसे आसानी से प्रयोग मे लाया जा सकता है।

2 जानकारी कुछ चन्द सेकंड मे विश्व के किसी भी कोने मे भेजी जा सकती है इसलिए यह बहुत तीव्र गति से कार्य करता है।

3 जब किसी को किसी ईमेल का उत्तर देना होता है तो पहले से आयी जानकारी साथ में रहती है।

4 अगर कोई यूजर अपने ईमेल को प्रयोग मे नहीं ला पा रहा है या किसी काम से बाहर है तो वह स्वतः ईमेल भेजने की तकनीक का प्रयोग कर सकता है जिसमे जिस यूजर ने ईमेल किया है उसको अपने आप मेल चला जायेगा।

5 पुराने पोस्टल सिस्टम पेपर का प्रयोग करते थे जबकि ईमेल इलेक्ट्रॉनिक डेटा का प्रयोग करता है अतः यह पर्यावरण अनुकूल है।

6 ईमेल को किसी भी उपकरण जैसे की कंप्यूटर, लैपटॉप, मोबाइल, टेबलेट पर पघ तथा उसका प्रतिउत्तर भेजा जा सकता है।

ईमेल का आर्किटेक्चर (**Architecture of Email**) : ईमेल के आर्किटेक्चर को समझने से पहले इसके कुछ महत्वपूर्ण तत्वों को समझना आवश्यक है

1 मेल उपयोगकर्ता एजेंट (Mail User Agent) : यह एक एप्लीकेशन या प्रोग्राम होता है जिसकी मदद से यूजर मेल लिखता है और भेज देता है।

2 मेल स्थानांतरण एजेंट (Mail Transfer Agent) : यह सामान्यतया एक सर्वर होता है जो ईमेल

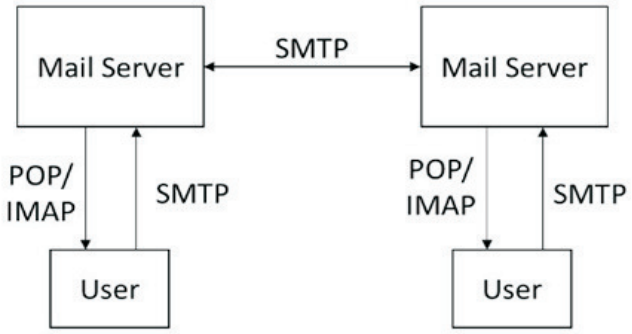
को एक दूसरे को भेजने का कार्य करता है।

3 मेल डिलिवरी एजेंट (Mail Delivery Agent) : यह मेल सर्वर की है तकनीक है जिसमे मेल के प्राप्तकर्ता का पता चलने के बाद उसे उसके इनबाक्स (inbox) मे डाल दिया जाता है।

4 पॉप प्रोटोकॉल (Post Office Protocol) : यह सर्वर से क्लाइंट या यूजर की एप्लीकेशन या प्रोग्राम पर मेल को प्राप्त करने का प्रोटोकॉल है।

5 SMTP (Simple Mail Transfer Protocol) : यह किसी यूजर के प्रोग्राम या एप्लीकेशन से ईमेल को सर्वर तक तथा सर्वर से किसी अन्य सर्वर पर भेजने के काम आता है।

6 Internet Message Access Protocol : यह मेल को प्राप्त करने का नया तरीका है जो कुछ आधुनिक तकनीक का प्रयोग करता है।



चित्रः 4.33 मेल सिस्टम

ईमेल के आर्किटेक्चर को हम चित्र 4.33 से समझने का प्रयास करेंगे। जो निम्नलिखित है–

7 कोई भी यूजर प्रोग्राम या एप्लीकेशन से एक जानकरी लिखता या कोई चित्र उसमे जोड़ता है जिसे अटैचमेंट कहते है। ईमेल लिखते समय कुछ महत्वपूर्ण जानकारी जैसी के प्राप्तकर्ता का एड्रेस, ईमेल का विषय लिखता है जिसे ईमेल का हैडर कहते हैं।

8 अब इस बनाये हुए मेसेज को SMTP प्रोटोकॉल की मदद से उस ईमेल सर्वर को भेज दिया जाता है जिसकी सेवाए यूजर ले रहा है।

9 अब सर्वर प्रेषक द्वारा भेजे गए ईमेल के हैडर मे से प्राप्तकर्ता का एड्रेस के डोमेन को अपने डोमेन से मिलाता है। अगर प्राप्तकर्ता का डोमेन सर्वर के डोमेन से मिल जाता है तो सर्वर अपने डेटाबेस (Database) मे यूजर का नाम ढूंढता है। नाम मिल जाने के बाद सर्वर मेल डिलीवरी एजेंट का काम करते हुए मेल को उसके इनबॉक्स में डाल देता है जो कि POP/IMAP प्रोटोकॉल द्वारा प्राप्त कर पढ़ लिया जाता है।

10 अगर प्राप्तकर्ता एवं सेवर का डोमेन मिलता नहीं है तो सर्वर उस ईमेल को उसके सम्बंधित सर्वर को SMTP प्रोटोकॉल द्वारा भेज देता है। अब दूसरा सर्वर भी उपरोक्त तरीके को काम मे लेते हुए ईमेल को यूजर के इनबॉक्स मे दाल देता है।

POP/IMAP मे अंतर : POP/IMAP दोनों ही प्रोटोकॉल ईमेल को सर्वर से यूजर तक लाने मे प्रयोग आते है। पर इनमे कुछ तकनीकी अंतर है जो निम्न है–

1 जैसे ही यूजर एप्लीकेशन या प्रोग्राम खोलके यूजर नाम और पासवर्ड देता है उसके बाद POP प्रोटोकॉल पूरे ईमेल को सर्वर से यूजर /क्लाइंट (Client) पर डाउनलोड कर देता है, डाउनलोड करते समय POP दो प्रकार से काम करता है या तो ईमेल की एक प्रतिलिपि सर्वर

रखता है या डाउनलोड करने के बाद प्रतिलिपि को वह है नष्ट कर देता है। POP पूरे ईमेल को अटेचमेंट के साथ डाउनलोड कर लेता है।

2 IMAP, POP की तुलना मे ज्यादा शक्तिशाली और जटिल प्रोटोकॉल है और ज्यादा विशेषताए (features), यूजर को देता है।

3 IMAP पूरे ईमेल को डाउनलोड करने के बजाय केवल ईमेल के हैडर को ही डाउनलोड करता है जिससे यूजर को यह पता चल जाता है की मेल कहा से आया है तथा उसकी विषय क्या है।

4 अगर यूजर अपने इनबॉक्स मे किसी ईमेल को बिना पूरे ईमेल को डाउनलोड किये उसे ढूँढ सकता है।

5 पूरे मेल का डाउनलोड ना होना बैंडविड्थ को बचाता है।

6 यूजर अपनी सुविधा के अनुसार नए फोल्डर बना सकता है।

महत्वपूर्ण बिंदु

1. कंप्यूटर को एक दूसरे से जाड़ेने की शैली को टोपोलॉजी कहते है।
2. नेटवर्क को बृहद रूप से तीन प्रकार मे विभाजित कर सकते है – LAN, MAN, WAN
3. प्रसारण माध्यम वो पथ है जिस पर प्रेषक और प्राप्तकर्ता संकेतो का आदान प्रदान करते है।
4. फाइबर ऑप्टिक माध्यम संचरण का सबके तेज पथ है।
5. प्रोटोकॉल नियमो का समूह होता है।
6. स्टार टोपोलॉजी मे सभी उपकरण एक केंद्रीय उपकरण से जुड़े होते है।
7. OSI एक reference मॉडल है तथा TCP/IP, protocol Model है।
8. सिम्पलेक्स मोड मे डेटा एक ही दिशा मे जा सकता है, प्राप्तकर्ता वापस प्रेषक को डेटा नहीं भेज सकता।
9. माइक्रोवेव संकेतो का प्रसारण इमारतों के ऊपर लगे ऐन्टेना द्वारा किया जाता है।
10. OSI मॉडल मे 7 परते तथा TCP/IP में 4 परते होती है।
11. टेलीविजन मे Coaxial Cable काम में आती है।
12. ध्वनि को भेजने के लिए एनालॉग सिग्नल का प्रयोग लिया जाता है।
13. कंप्यूटर अपना काम बाइनरी नम्बर सिस्टम मे करता है।
14. पैकेट्सको एक नेटवर्क से दूसरे नेटवर्क मे भेजने के लिए राऊटर का उपयोग होता है।
15. IPV4, 32 बिट बाइनरी एड्रेस है तथा IPV6 128 बिट का एड्रेस है।
16. IP नेटवर्क लेयर की एड्रेसिंग योजना है।
17. मैक एड्रेस एक भौतिक पता है जो कि बदलता नहीं है। यह हार्डवेयर मे रीड ओनली चिप पर होता है ।
18. Subnetting से एक बड़े नेटवर्क को छोटे छोटे नेटवर्क मे विभाजन के लिए किया जाता है।
19. मॉडेम अनालोग सिग्नल्स को डिजिटल मे तथा डिजिटल को एनालॉग सिग्नल्स मे बदलता है।
20. नेटवर्क मे हब सशक्त वायरिंग केंद्र होता है।
21. गेटवे प्रेषक नेटवर्क से निर्देशों के समूह को प्राप्तकर्ता नेटवर्क के निर्देशों में बदलता है।
22. IPV4 मे 32 बिट बाइनरी एड्रेस को 8 बिट या ओकटेट के समूह मे दर्शाया जाता है तथा प्रत्येक 8 बिट का समूह एक बिंदु (Dot) से अलग रहता है।

23. IPV4 की 5 क्लासेज होती हैं। ये क्रमशः A, B, C, D एवं E है।

24. वायरलेस नेटवर्क किसी भी उपकरण, किसी बिना तार किसी माध्यम से जुड़ने की सुविधा देता है। वायरलेस लैन (W LAN), वायरलेस तकनीक का एक वर्गीकृत भाग जो सामान्यतया घरों, दफ्तरों तथा कैंपस मे काम लिया जाता है।

25. कन्जेसन कण्ट्रोल वो तकनीक व तरीका है जो नेटवर्क मे पैकेट्स की भीड़ या तो होने से पहले ही ठीक कर देता है अर्थात ऐसा होने नहीं देता या होने के बाद उसे ठीक कर देता है।

26. सेवा की गुणवत्ता का काम किसी यूजर, एप्लीकेशन, डेटा फ्लो को उच्च गुणवत्ता की प्राथमिकिता देना है।

अभ्यासार्थ प्रश्न

बहुचयनात्मक प्रश्न

1. इसमे से कौनसा ट्रांसमिशन माध्यम है–
   (अ) मॉडेम (ब) मल्टीप्लेक्सर
   (स) हब (द) कोऐक्सिअल केबल

2. सबसे पुरानी एवं अधिक काम मे आने वाली ट्रांसमिशन लाइन है –
   (अ) कोऐक्सिअल केबल (ब) फाइबर ऑप्टिक
   (स) ट्विस्टेड पेअर (द) उपरोक्त मे से कोई नहीं

3. WAN का मतलब है–
   (अ) वायर एरिया नेटवर्क (ब) लोकल एरिया नेटवर्क
   (स) वाइड एरिया नेटवर्क (द) वायर एक्सेसिबल नेटवर्क

4. OSI मॉडल मे कितनी परते है?
   (अ) 4 (ब) 2 (स) 7 (द) 5

5. कौनसा उपकरण पैकेट्स को एक नेटवर्क से दूसरे नेटवर्क मे भेजता है ?
   (अ) राऊटर (ब) हब
   (स) स्विच (द) गेटवे

6. किस तरह के ट्रांसमिशन मे तरंगे सभी दिशाओ मे जाती है ?
   (अ) रेडियो लिंक (ब) माइक्रोवेव
   (स) इन्फ्रारेड (द) उपग्रह

7. TCP/IP मॉडल की कौनसी परत ट्रांसपोर्टेशन का काम करती है ?
   (अ) एप्लीकेशन (ब) ट्रांसपोर्ट
   (स) नेटवर्क एक्सेस (द) इन्टरनेट

8. एनालॉग सिग्नल की शक्ति बढाने के लिए किस उपकरण कर प्रयोग करते है ?
   (अ) एम्प्लीफायर (ब) ट्रांसमीटर
   (स) रिपीटर (द) ट्रांसपोंडर

9. निम्न मे से किस उपकरण मे टेलीफोन लाइन का उपयोग होता है ?
   (अ) राऊटर (ब) मॉडेम
   (स) स्विच (द) हब

10. निम्न मे से कौन फायरवॉल का भी काम कर सकता है ?
(अ) राऊटर (ब) मॉडेम
(स) स्विच (द) हब

11. कंप्यूटर IP किस नम्बर सिस्टम में लिखता है ?
(अ) बाइनरी (ब) डेसीमल
(स) हेक्साडेसीमल (द) इनमे से कोई नहीं

12. IPV6 कितने बिट का एड्रेस है?
(अ) 128 (ब) 64 (स) 28 (द) 32

13. IP की क्लास D को किस नाम से पुकारते है ?
(अ) ब्राडकास्टिंग (ब) मल्टीकास्टिंग
(स) सबनेटिंग (द) रूटिंग

14. मैक एड्रेस किस लेयर से सम्बंधित है?
(अ) इन्टरनेट (ब) नेटवर्क
(स) डेटा लिंक (द) एप्लीकेशन

15. सबसे कम दूरी पर काम करने वाली तकनीक है–
(अ) 3 जी (ब) वायरलेस
(स) ब्लूटूथ (द) उपग्रह

अति लघूत्तरात्मक प्रश्न –

1. दो तरह के नेटवर्किंग सिग्नल के नाम लिखें।
2. दो तरह की ट्विस्टेड पेअर केबल के नाम लिखे।
3. OSI तथा TCP/IP का पूरा नाम लिखे।
4. नेटवक मे काम आने वाली टोपोलॉजी के नाम लिखे।
5. नेटवर्क मे काम आने वाले दो उपकरणों के नाम लिखो।
6. IP किस परत से सम्बंधित है?
7. मैक एड्रेस किस प्रकार का एड्रेस है?
8. वायरलेस मे काम आने वाली 2 सुरक्षा प्रणाली के नाम लिखो।
9. नेटवर्क मे पैकेट्स की भीड़ को कम करने के लिए किस तकनीक का उपयोग किया जाता है?
10. सेवा की गुणवत्ता की जरुरत क्यों है?
11. DNS का उपयोग क्यों करते है?
12. SMTP का पूरा नाम लिखे।

लघूत्तरात्मक प्रश्न –

1. विभिन प्रकार के तारों वाले ट्रांसमिशन कौन कौनसे है ?
2. दो प्रकार के सिग्नल के बारे मे लिखे ?
3. IP एड्रेस की प्लानिंग क्यों जरुरी है?
4. सबनेटिंग क्यों करते है?
5. मैक एड्रेस को समझाइए।

6. चोक पैकेट प्रणाली को समझाइए।
7. नेटवर्क मे जिटर क्या होता है?
8. मेल ट्रान्सफर एजेंट को समझाइए।
9. DNS की जरुरत क्यों होती है?

निबंधात्मक प्रश्न –

1. OSI मॉडल की परतों का विवरण संक्षेप मे लिखे।
2. डेटा ट्रांसमिशन को समझाइए।
3. नेटवर्किंग में प्रसारण माध्यमों को समझाइए।
4. उपग्रह ट्रांसमिशन क्या है? समझाइए।
5. IP को विस्तार से समझाए।
6. मॉडेम की कार्य प्रणाली को चित्र सहित समझाइए।
7. सबनेटिंग को उदाहरण सहित समझाइए।
8. Congestion कण्ट्रोल क्या है? विस्तार से लिखे।
9. ईमेल भेजने की प्रक्रिया को समझाइए।

उत्तरमाला

1. (द) 2. (स) 3. (स) 4. (स) 5. (अ) 6. (अ) 7. (ब) 8. (अ) 9. (ब) 10. (अ)
11. (अ) 12. (अ) 13. (ब) 14. (स) 15 (स)

पाठ–5

XML

5.1 मार्कअप भाषा का परिचय

मार्कअप भाषा का संदर्भ कोड़ (Code) एवं टोकन्स (Tokens) से है जिसको एक दस्तावेज (document) के अंतर्गत रखा जाता है एवं जिसके द्वारा डेटा की व्याख्या की जाती है। दूसरे शब्दों में मार्कअप का उपयोग डेटा को व्यवस्थित करने के लिये किया जाता है। उदाहरण के लिये HTML एक मार्कअप भाषा है जिसका उपयोग वैब पेज (Web Page) के निर्माण के लिये किया जाता है।

उदाहरण : 5.1

```
<html>
    <head>
        <title> hello from HTML</title>
    <head>
    <body><center><H1>HTML document<!H1></center>
This is an HTML document
    </body>
</html>
```

उपरोक्त उदहारण में जैसे कि दर्शाया गया है कि HTML वैब पेज के महत्वपूर्ण भागों जैसे हैडर, बॉडी, फुटर इत्यादि को ब्राउजर (Internet Explorer, Google Crome, Firefox) के द्वारा इंटरप्रेट (Interpret) करने के लिये उपयोग में आ रही है। HTML मार्कअप, HTML टैग्स (Tags) के द्वारा निर्मित की जाती है। जो कि पूर्व निर्धारित है। HTML टैग्स ब्राउजर को दस्तावेज के डेटा को प्रदर्शित करने का निर्देश देते है। इसी प्रकार से XML भाषा का प्रयोग डेटा को प्रदर्शित करने एवं डेटा को संग्रहित करने में होता है।

5.2 XML परिचय

XML भाषा का निर्माण वर्ड वाईड वैब कंसोटियम (World Wide Web Consortium) द्वारा किया गया है। यह एक प्रकार की मार्कअप (markup) भाषा है। XML भाषा का उपयोग एचटीएमएल (HTML) की भांति डेटा को प्रदर्शित करने के साथ–साथ डेटा को प्रदर्शित एवं संग्रहित भी करना है। XML का उपयोग इन्टरनेट पर सूचनाओं को आदान प्रदान करने हेतु भी होता है। XML में निहित टेक्स्ट को इस तरह से निर्मित किया गया है जिसको मानव एवं मशीन के द्वारा आसानी से समझा जा सके। यह एक स्वयं वर्णात्मक (Self Descriptive) भाषा है एवं XML स्टैण्डर्ड जनरलाईज्ड मार्कअप भाषा (SGML) का उपभाग (Subset) है। इस प्रकार की मार्कअप भाषा को बडे डेटा (Data) संग्रहण (Storage) के लिये निर्मित किया गया था।

5.2.1 HTML एवं XML में अंतर

HTML	XML
1. HTML का अभिप्राय हाईपर टैक्सट मार्कअप भाषा है।	1. XML का अभिप्राय एक्सटेंसिबल (Extensible) मार्कअप भाषा है।
2. HTML डेटा को प्रदर्शित करने एवं HTML वैब पेज को आकर्षित बनाने के लिए निर्मित की गई है।	2. XML डेटा को संग्रहित करने एवं उसके आदान प्रदान के उपयोग में आती है।
3. HTML स्वंय एक मार्कअप भाषा है।	3. XML मार्कअप भाषा को परिभाषित करने हेतु एक फ्रेमवर्क (Framework) प्रदान करती है।
4. HTML Case संवेदनशील नहीं होती है।	4. XML Case संवेदनशील है।
5. HTML का उपयोग क्लाइंट ब्राउजर पर वैब पेज को प्रदर्शित करने के लिए किया जाता है।	5. XML का उपयोग डेटा को डेटाबेस से सूचनाओं के आदान प्रदान के लिए किया जाता है।
6. HTML में पूर्वनिर्मित टैग्स होते हैं।	6. XML टैग्स यूजर्स के द्वारा निर्मित किये जा सकते है।
7. HTML सिर्फ डेटा को प्रदर्शित करती है अतः यह एक स्थिर (static) भाषा है।	7. XML डेटा को संग्रहित एवं आदान प्रदान करने में सक्षम है अतः यह एक डायनैमिक (dynamic) भाषा है।

5.2.2 XML डेटा की संरचना

XML दस्तावेज न केवल डेटा को स्टोर करता है बल्कि डेटा की संरचना (Structure) को भी निर्दिष्ट करने में सक्षम है। XML संरचना जटिल (Complex) डेटा के साथ कार्य करने के हेतु अति महत्वपूर्ण है।

उदाहरणः– के तौर पर अगर हम एक विशाल खाते के विवरण (Account Details) को HTML में स्टोर करने का प्रयास करते है तो उक्त HTML में त्रुटियां होने की संभावनाएं बढ़ जाती है। XML के अंदर हम सिंटैक्स (Syntax) के नियम बना सकते है जो कि दस्तावेज की संरचना को निर्देशित करते हैं जिससे दस्तावेज में त्रुटियां होने की संभावनाएं पूर्णतया समाप्त हो जाती है। XML प्रोसेसर (Processor) के द्वारा XML टेक्स्ट को जांचा जाता है एवं उक्त प्रोसेसर के द्वारा दो महत्वपूर्ण जांचे की जाती है। जो कि निम्न प्रकार है –

1. XML टेक्स्ट सुसज्जित प्रकार (Well Formed) से होना चाहिये।
2. XML वैध (Valid) होना चाहिए।

5.2.3 XML की विशेषताएं

1. एक जटिल एवं असामान्य डेटा को हेंडल करने की क्षमता
2. टेक्स्ट डेटा का विवरण रखना
3. दीर्घकालिक डेटा के भंडारण और डेटा के पुर्नउपयोग (Reuse) के लिए उपयुक्त

4. मानव एवं कम्प्यूटर के अनुकूल प्रारूप
5. मार्कअप भाषा में वर्णित डेटा का विवरण
6. XML, डेटा को ट्री (Tree) संरचना में व्यवस्थित करती है जिसमें एक ही मूल तत्व (Root Element) होता है।

5.3 XML दस्तावेज निर्माण

5.3.1 XML घोषणा (Declaration)

XML दस्तावेज की वैकल्पिक रूप से घोषणा होती है। जिसे निम्न प्रकार से दर्शाया जाता है–

<?xml version="1.0" encoding="UTF-8"?>

उपरोक्त में XML संस्करण (Version) XML दस्तावेज के संस्करण को दर्शाता है तथा encoding=UTF-8 document उपयोग में आने वाली कैरक्टर एनकोडिंग (character encoding) को दर्शाती है।

XML घोषणा, XML प्रोसेसर को XML दस्तावेज को पार्स (Parse) करने में मदद करता है। XML घोषणा अनिवार्य घोषणा है जिसे दस्तावेज में प्रथम पंक्ति में लिखा जाता है।

5.3.2 XML घोषणा के नियम

1- XML घोषणा, XML दस्तावेज की प्रथम पंक्ति में ही लिखा जाता है।
2- XML घोषणा में XML संस्करण निहित होना अनिवार्य है।
3- पैरामीटर (Parameter) का नाम एवं उसके मान (Value) केस संवेदनशील होते है।
4- पैरामीटर का नाम छोटे अक्षरों (Small-Cases) में लिखा जाता है।
5- एकल कोट्स (Single Quotes) अथवा डबल कोट्स (Double Quotes) को उपयोग में लिया जा सकता है।
6- XML घोषणा में समाप्ति टैग (Closing Tag) (/?XML) निहित नहीं होते हैं।

5.3.3 XML Tags और तत्व (element)

XML तत्वों (Elements) के लिए मूलतः तीन प्रकार के टैग्स का उपयोग होता है।

1. प्रारम्भिक टैग्स (Opening Tags)
2. समाप्ति टैग्स (Closing Tag)
3. रिक्त टैग्स (Empty Tag)

1. प्रारम्भिक टैग्स अथवा शुरूआती टैग्स :- हर तरीके के डेटा निहित XML तत्वों की शुरूआत टैग्स अथवा प्रारम्भिक टैग्स से की जाती है।

उदाहरणः–

<Local Address>

2 समाप्ति टैग्स और अन्तिम टैग्सः– जिस XML तत्व की शुरूआत प्रारम्भिक टैग से हुई है। उसका समाप्ति टैग होना भी अनिवार्य है।

उदाहरणः–

</Local Address>

3. रिक्त टैग्स – प्रारम्भिक टैग्स एवं समाप्ति टैग्स के मध्य में स्थित टैक्स्ट को कंटेन्ट (Content) कहा जाता है। परन्तु जिन तत्वों के प्रारम्भिक टैग एवं समाप्ति टैग के मध्य में कोई टैक्स्ट नहीं होता उसे रिक्त टैग तत्व कहा जाता है। रिक्त टैग्स तत्वों को निम्न दो प्रकार से दर्शाया जा सकता है।

a. प्रारम्भिक टैग के तुरन्त बाद अतिन्म टैग

उदाहरणः– <hr></hr>

b. एक पूर्णतया रिक्त तत्व को </hr> एलीमेंट टैग से दर्शाया जाता है।

5.3.4 XML एट्रीब्यूट्स (Attributes)

XML तत्व के अंदर विभिन्न एट्रीब्यूट्स निहित होते है। एट्रीब्यूट का निर्माण XML तत्व की विभिन्न जानकारियों के लिए किया गया है XML एट्रीब्यूट में एट्रीब्यूट का नाम एवं उसके मान को जोड़ा जाता है।

Syntax _

<element-name attribute1 attribute2 content…..>

</element-name>

Example

<student firstname="Mahesh">

<name>Sharma</name>

<grade>A+</grade>

</student>

5.3.5 XML कमेन्ट्स (Comments)

XML कमेन्ट्स, HTML मार्कअप कमेन्ट्स की भांति ही होते हैं। कमेन्ट्स को XML दस्तावेज के कोड को समझने के लिए उपयोग में लिया जाता है। XML कमेन्ट्स, XML कोड के निष्पादन (execution) में किसी भी प्रकार की कोई भूमिका नहीं निभाते है।

Syntax –

एक पंक्ति कमेन्ट्स

<!————Your Comment————>

बहु पंक्ति कमेन्ट्स

Comment <!———— ——-————————————-->

Example –

<?XML Version = "1.0" encoding = "UTF-8"/>

<!—Student grades are updated monthly———>

<classlist>

<student>

<name> Ramesh</name>

<grade>A+</grade>

</student>

```
        <name>Girish</name>
        <grade>A-</grade>
    </student>
</classlist>
```

5.3.6 XML एनटिटीज (Entities)

एक XML दस्तावेज कई भंडारण (Storage) इकाइयों से मिलकर बनता है। जिनको एनटिटी (entity) कहा जाता है। प्रत्येक XML दस्तावेज के पास एक एनटिटी होती है, जिसे दस्तावेज एनटिटी कहा जाता है। दस्तावेज एनटिटी XML प्रोसेसर के लिये शुरूआती बिन्दु (point) एवं प्लेस होल्डर्स (Placeholders) की तरह कार्य करती है।

XML में विभिन्न प्रकार की XML एनटिटीज होती है। परन्तु इस इकाई में हम सिर्फ करेक्टर एनटिटीज का ही अध्ययन करेंगे।

करेक्टर एनटिटीज (Character Entities)-

XML के अंदर कुछ इस प्रकार के सिम्बल्स (symbols) होते है जिनको कोन्टेन्ट (content) अथवा XML टैक्स्ट के रूप में प्रयोग में नहीं लिया जा सकता है। उदाहरण के के तौर पर अगर <(Less than) एवं >(greater than) सिम्बल्स का उपयोग एंगिल टैग (< >) के मध्य में कन्टेट के रूप में यदि स्थापित किया गया तो इस प्रकार का डेटा अस्पष्टता (Ambiguity) निर्मित करता है। इस प्रकार की समस्याओं अथवा अस्पष्टता से बचने के लिए दस्तावेज करेक्टर एनटिटीज का उपयोग किया जाता है। जो कि XML के द्वारा पूर्व निर्धारित है।

उदाहरणः–

```
Ampersand:&amp;
single quote:&apos;
greater than : &gt;
less than:&lt;
Double quote:&quote;
```

इस प्रकार के मुख्य करेक्टरर्स को एनटिटीज के द्वारा प्रदर्शित किया जा सकता है।

करेक्टर एनटिटीज के प्रकार–

करेक्टर एनटिटीज निम्न प्रकार की होती है जो कि

1. पूर्व निर्धारित (Predefined) करेक्टर एनटिटीज
2. Numbered करेक्टर एनटिटीज
3. Named करेक्टर एनटिटीज

5.3.7 XML की विशेषताएं–

1. XML डाटा को HTML से अलग व्यवस्थित रखती है।
2. XML डाटा आदान–प्रदान में उपयोग में आती है।
3. XML डाटा आदान–प्रदान की प्रक्रिया को आसान करती है।
4. XML आपरेटिंग सिस्टम पर निर्भर नहीं है।

5. XML नयी अन्य Internet भाषाओं के निर्माण में सहयोग करती है।

5.5 XML डीटीडी (DTD)

डीटीडी (document type definition) को XML की एक वैध (valid) आधारभूत संरचना के निर्माण हेतु उपयोग में लिया जाता है। यह XML दस्तावेज के लिये उपयोंग में लिये जा सकने वाले तत्वों को प्रदर्शित करती है। XML डीटीडी को निम्न दो प्रकार से उपयोग में लिया जा सकता है।

XML दस्तावेज को दस्तावेज के अंदर स्थापित करके अथवा XML डीटीडी को किसी दूसरे दस्तावेज में परिभाषित कर उसका संदर्भ (reference) मूल XML दस्तावेज में देकर भी उपयोग में लिया जा सकता है।

```
Syntax –
<!doctype element DTP identifier
[
        Declaration 1
        Declaration2
        ____________

]>
```

उपरोक्त सिंटेक्स (Syntax) के विभिन्न पहलुओं का अर्थ निम्न है।

1. डीटीडी को <!DOCTYPE delimiter> से शुरू किया जाता है।
2. तत्व, XML parser को मूल तत्व की जानकारी प्रदान करता है।
3. डीटीडी पहचानकर्ता (Identifier), DTD को पहचानने में उपयोग में आता है। जो कि फाईल के पाथ (Path) अथवा उसके इंटरनेट URL को भी प्रदर्शित करता है।
4. स्क्वायर ब्रेकिटस (Square brackets [ ]) के अंदर ऐच्छिक एन्टिटी घोषणा होती है। जिसे आंतरिक उपभाग (Internal subset) से जाना जाता है।

डीटीटी घोषणा को दो प्रकार से किया जाता है जो निम्न प्रकार है–

1. आन्तरिक डीटीडी घोषणा **(Internal DTD Declaration) -** यदि आन्तरिक डीटीडी की घोषणा को XMLफाईल के अंदर ही किया जाये तो उसे आन्तरिक डीटीडी घोषणा कहा जाता है और यह घोषणा <!DOCTYPE> परिभाषा के अंदर ही की जाती है।

उदाहरणः– :

```
<?XML version ="1.0" ?>
<!DOCTYPE note [
        <!ELEMENT note (to, from heading, body)>
        <!ELEMENT to (#PCDATA)>
        <!ELEMENT from (#PCDATA)>
        <!ELEMENT heading (#PCDATA)>
        <!ELEMENT body (#PCDATA)>
<note>
<to>Mahesh</to>
```

<from>Jaipur</from>

<heading>Reminder</heading>

<body>Please remember it </body>

</note>

उपरोक्त उदाहरणः– में विभिन्न परिभाषाएं निम्न प्रकार है–

!DOCTYPE note "मूल तत्व (note) को प्रदर्शित कर रही है। जिसके अंदर चार तत्व निहित है। जैसे "to, from, heading एवं body"

2. बाह्य डीटीडी घोषणा **(External DTD Declaration) -** यदि डीटीडी को एक अलग फाईल के अंदर परिभाषित कर उसका संदर्भ XML फाईल में जोड़ा जाये तो उसे बाह्य डीटीडी घोषणा कहते है अगर बाह्य डीटीडी घोषणा का संदर्भ <!DOCTYPE> की परिभाषा के अंदर दिया जाता है।

उदाहरणः– :

<?XML version ="1.0" ?>

<!DOCTYPE note SYSTEM "note-dtd">

<note>

<to>Mahesh</to>

<from>Jaipur</from>

<heading>Reminder</heading>

<body>Please remember it </body>

</note>

Note-dtd

<!ELEMENT note (to, from heading, body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT heading (#PCDATA)>

<!ELEMENT body (#PCDATA)>

उपरोक्त उदाहरण में Note-dtd एक बाह्य DTD फाईल है जिसका संदर्भ मूल **XML** दस्तावेज में दिया गया है एवं Note-dtd के अन्तर्गत विभिन्न गुणों जैसे Note, to, from, heading, body को मूल XML में किया जा रहा है।

5.6 XML दस्तावेज स्कीमा (DTD Schema)

XML दस्तावेज स्कीमा के प्रकार डाटा के डाटा टाइपस पर निर्भर करते है। डाटा टाइपस के आधार पर XML दस्तावेज स्कीमा को निम्न दो प्रकारों में बांटा जा सकता है।

1. सामान्य टाइप (Simple Type) 2. जटिल टाइप (Complex Type)

XML स्कीमा (Schema) को सामान्यतः XML स्कीमा परिभाषा (XML Schema definition-XSD) से जाना जाता है। यह XML आधारभूत संरचना, XML कन्टेन्ट (content) के वर्णन एवं दस्तावेज को मान्य (validate) करने में उपयोग आती है।

XML स्कीमा संरचना XML स्कीमा तत्वों, एट्रीब्यूट्स एवं डेटा टाईप्स को परिभाषित करती है। यह डेटाबेस स्कीमा की भांति होती है जो कि डेटाबेस के डेटा का वर्णन करता है।

Syntax –

हम XML दस्तावेज के अन्दर XML स्कीमा को निम्न प्रकार से घोषित (declare) करते है।

<XS : Schema XMLns : xs = http://www.abc.com/2001/XML Schema>

उदाहरणः

```
<?XML version = "1.0" encoding = "UTF-8"?>
<XS: Schema XMLS:XS ="http://www.abc.com/2001/XML Schema>
<XS: element name= "contact">
    <xs : ComplexType>
        <xs: sequence>
            <xs: elements name="name" type="xs: string"/>
            <xs: elements name="Company" type="xs: string"/>
            <xs: elements name="phone" type="xs: int"/>
        </xs: sequence>
    </xs :ComplexType>
</xs: element>
</xs: Schema>
```

5.6.1 XML स्कीमा एवं DTD में अंतर

XML स्कीमा	DTD
1. XML स्कीमा को **XML** के द्वारा निर्मित किया गया है।	**1. DTD** का निर्माण **SGML** सिनटेक्स द्वारा किया गया है।
2. XML स्कीमा डाटा टाइपस के द्वारा तत्व **(elements)** एवं **attributes** को परिभाषित किया जाता है।	**2. DTD** के अंदर डाटा टाइपस निहित होते है।
3. XMLDOM द्वारा परिवर्तित की जा सकती है।	**3. DOM** द्वारा परिवर्तन संभव नहीं है।

5.7 XML मान्यीकरण (Validation)

मान्यीकरण XML दस्तावेज की संरचना को जानने एवं मान्य (validate) करने की एक प्रकिया है। एक XML दस्तावेज को मान्य तब ही किया जा सकता है जब दस्तावेज के कन्टेन्टस डीटीडी के सभी नियमो का पालन करे एवं उसके कन्टेन्टस, तत्वों एवं एट्रीब्यूटस (Contents, elements attribute) से भिन्न न हो।

निम्न दो प्रकार के XML दस्तावेज को मान्य किया जा सकता है–

1. वैल फोर्मड (Well formed) XML दस्तावेज

2. मान्य (Valid) XML दस्तावेज

1) वैल फोर्मड **(Well formed) XML** दस्तावेज

वैल फोर्मड XML सिंटेक्स नियम

a. XML दस्तावेज के अंदर मूल तत्व (root element) होना चाहिये।

b. XML तत्वों के समाप्ति टैग्स होने चाहिए।

c. XML टैग्स केस संवेदनशील होते हैं।

d. XML तत्व उचित रूप से व्यवस्थित (properly nested) होने चाहिये।

e. XML एट्रीब्यूट वैल्यूज "quotes" के मध्य में होनी चाहिये।

उदाहरणः–

Unformed XML Code

```
<email>
    <to> Mr. John
        <body> Hello there!</to>
    </body>
</email>
```

Well formed XML Code

```
<email>
    <to> Mr. Garcia</to>
        <body> Hello there
    </body>
</email>
```

2) मान्य **XML** दस्तावेज : यदि XML दस्तावेजों के वैल फोर्मड होने के सभी नियमों का पालन करे एवं डीटीडी के साथ भी जुडा हो तो उसे मान्य XML दस्तावेज जाना जाता है।

5.8 XML नेमस्पेस (Namespace)

XML नेमस्पेस का परिचय

XML दस्तावेजों के निर्माण के समय जब नये एलीमेन्ट को निर्मित किया जाता है तब वहां एक ही नाम के दो या उससे अधिक एलीमेन्ट के भी निहित होने की संभावाना होती है।

उदाहरणः

```
<?XML version = "1.0" encoding = "ISO-8859-15"?>
<html>
    <body>
        <P>Welcome</P>
        </body>
        <body>
            <height>6' </height>
            <Weight>155 </Weight>
            </body> </html>
```

उक्त उदाहरण में एप्लीकेशन (Application) <body> को जरूरत के हिसाब से लिया गया है। जो मानव संरचना को उदाहरण के तौर पर प्रदर्शित कर रही है। परन्तु उदहारण में दो स्थानों पर किया जा रहा है। जिसे XML नियमों के अनुसार दो स्थानों पर प्रयोग में नहीं लिया जा सकता। उक्त समस्या

से निदान पाने के लिये ही नेमस्पेस का निर्माण किया गया है।

अतः XML नेमस्पेस एक इस प्रकार की प्रक्रिया (Mechanism) है जिसके द्वारा एक ही नाम के तत्वों एवं एट्रीब्यूटस जिनके नाम तो एक ही है परन्तु परिभाषाएं अलग अलग है, उन तत्वों एवं एट्रीब्यूटस को नेमस्पेस के द्वारा एक ही XML दस्तावेज में प्रदर्शित किया जा सकता है।

नेमस्पेस घोषणा – XML दस्तावेजों में XML नेमस्पेस को सुरक्षित गुणों (Reserve Attributes) के द्वारा घोषित किया जा सकता है।

उदाहरणः

<elements XMLs : name = "URL">

1. Namespace XMLs Keywords से शुरू होते है।
2. Name Namespace उपसर्ग (Prefix) है।
3. URL एक Namespace पहचानकर्ता (identifier) है।

उदाहरणः

```
<root>
<h: table XMLs : h "http://www.abc.com/TR/>
        <h :tr>
                <h:td>Aries</h:td>
                <h:td>Bingo</h:td>
        </h:tr>
</h:table>
<h: table XMLs : f"http://www.xyz.com/furniture/>
        <f:widht>80 </width>
        <f:length>120</f:length>
</f:table>
</root>
```

उपरोक्त उदाहरण में h एवं f दो अलग अलग नेमस्पेस गुण (attribute) है जिनको एक ही XML एलीमेंट (h:table) में स्थापित किया गया है। h एवं f दो अलग अलग यूआरएल को प्रदर्शित कर रहे है एवं जिनकी परिभाषाएं भी भिन्न भिन्न है।

डिफॉल्ट नेमस्पेस (Default Namespace) – यह एक इस प्रकार का नेमस्पेस है जो कि नेमस्पेस Prefix का उपयोग नहीं करता।

डिफॉल्ट नेमस्पेस को xmlns के साथ बिना Prefix लगाकर निम्न प्रकार से किया जा सकता है।

```
<xhtml xmlns="http://www.w3c.org/1999/xhtml">
```

महत्वपूर्ण बिन्दु

1- XML का निर्माण वर्ड वाईड वैब कंसोटियम (World Wide Web Consortium-W3C) द्वारा किया गया है। XML SGML(Standard Generally Makrup Language) का उपभाग है।

2- XMLका उपयोग डाक्यूमेंट डेटा को संग्रहित करने एवं सूचनाओं के आदान प्रदान करने में किया जाता है।

3- XML टैग्स तीन प्रकार के होते हैं प्रारम्भिक टैग्स, समाप्ति टैग्स एवं रिक्त टैग्स।

4. एक XML दस्तावेज कई भंडारण इकाईयों से मिलकर बनता है। जिसको एन्टीटी कहा जाता है।

5- XML में कुछ इस प्रकार के सिम्बल्स होते हैं जिनको कंटेन्ट अथवा XML टैक्स्ट के रूप में उपयोग में नहीं लिया जा सकता है। अतः इन सिम्बलों को XML टैक्स्ट अथवा कंटेंट में परिभाषित करने के लिए करेक्टर एन्टीटीज को प्रयोग में लिया जाता है।

6- XML डीटीडी को XML की एक वैध आधारभूत संरचना के निर्माण हेतु उपयोग में लिया जाता है।

7- XML स्कीमा परिभाषा XML की आधारभूत संरचना, XML कंटेन्ट का वर्णन एवं XML दस्तावेजों को मान्य करने में उपयोग में आती है।

8- XML मान्यीकरण XML दस्तावेज की संरचना को जानने एवं मान्य करने की प्रक्रिया है।

अभ्यासार्थ प्रश्न

1. XML का अभिप्राय है–

(अ) X- मार्कअप भाषा (ब) Extensible मार्कअप भाषा

(स) Extra मार्कअप भाषा (द) Example मार्कअप भाषा

2. XML Data का वर्णन किया जाता है–

(अ) XML की वर्णनात्मक नोड़ (Description Node) के द्वारा

(ब) XSL का उपयोग करके

(स) DTD के द्वारा (द) इनमें से कोई नहीं

3. XML संस्करण को निम्न में से किस सिंटेक्स का उपयोग करके घोषित (Declare) किया जाता है?

(अ) <XML Version="1.0"/> (ब) <? XML Version="1.0"?>

(स) <? XML Version="1.0"/> (द) <XML Version="1.0"?/>

4 DTD का अभिप्राय है–

(अ) Dynamic Type Definition (ब) Document Type Definition

(स) Do the Dance (द) Direct type Definition

5. XML का निर्माण डेटा को स्टोर एवं ———————— करने में आता है।

(अ) डेटा आदान प्रदान (ब) XML के निर्माण हेतु

(स) मान्य (Verify) करने हेतु (द) इनमें से कोई नहीं

6. XML निम्न में से किसके समान है?

(अ) Java Script (ब) C Programming

(स) CSS (द) HTML

7. XML का कार्य नहीं है।

(अ) सूचना आदान प्रदान करना (ब) डेटा स्टोर करना

(स) स्टाइल्स की सूचना आदान प्रदान करना (द) सूचना की संरचना

8 XML का उचित सिटेंक्स _____ होता है।

(अ) मैच्युअर (Mature) (ब) वैल पैरामीटराईज्ड (Well Parameterized)

(स) वैल फोर्मड (Well Formed) (द) इनमें से कोई नहीं

9. XML दस्तावेज को निम्न मे से मान्य किया जाता है–

(अ) CGG (ब) DTD (स) Jquery (द) Parses

10. XML का निर्माण किया गया–

(अ) W3C द्वारा (ब) W2C द्वारा (स) HTML द्वारा (द) इनमें से कोई नहीं

लघूरात्मक प्रश्न

1. XML संस्करण सिंटेक्स को किस प्रकार से प्रदर्शित किया जाता है।
2. XML एवं HTML का अभिप्राय पूर्ण रूप में बतायें।
3. मान्य XML दस्तावेज से क्या अभिप्राय है?
4. डीटीडी क्या है?
5. XML एट्रीब्यूट का XML में क्या उपयोग है?

निबन्धात्मक प्रश्न

1. XML के प्रमुख कार्य एवं विशेषताओं को संक्षेप में वर्णन करें।
2. XML डीटीडी को उदाहरण के साथ वर्णन करें।
3. XML स्कीमा का वर्णन करें।

उत्तरमाला

1.	ब	2.	स	3.	ब	4.	ब	5.	अ
6.	द	7.	स	8.	स	9.	ब	10.	अ

पाठ–6

डेटाबेस मैनेजमेंट सिस्टम (DBMS)

डेटाबेस को सामान्यतया डेटा के लिए एक भण्डार के रूप में वर्णित किया जा सकता है। एक डेटाबेस किसी भी संबंधित डेटा का संग्रह है। एक डेटाबेस निर्बाध, तार्किक रूप से सुसंगत, स्वाभाविक सार्थक डेटा, वास्तविक डेटा का संग्रह है।

डेटा तथ्य और आंकडों का एक संग्रह है जो की सूचना (information) प्रदान करने के लिए संसाधित (processed) किया जा सकता है। ज्यादातर डेटा रिकॉर्ड करने योग्य तथ्य का प्रतिनिधित्व करता है। डेटा सूचना प्रदान करने में सहायक होता है जो तथ्यों पर आधारित है।

अगर हमारे पास सभी विद्यार्थियों द्वारा प्राप्त किये प्राप्तांक उपलब्ध है तो हम अव्वल रहने वाले विद्यार्थियों की सूची और अन्य वांछित सूचना निकाल सकते हैं। डेटाबेस प्रबंधन प्रणाली द्वारा इस तरह डेटा संग्रहीत किया जाता है कि आसानी से सूचना का निर्माण, उसको प्राप्त (access) और उसका अधतन (update) किया जा सकता है ।

6.1 डेटाबेस मैनेजमेंट सिस्टम (DBMS)

डेटाबेस मैनेजमेंट सिस्टम (DBMS) एक सॉफ्टवेयर है जिनसे कि डेटाबेस भंडारण, पहुँच, सुरक्षा, बैकअप और अन्य सुविधाएं संचालित करने के कार्य होते हैं। कुछ सामान्य रूप से प्रयुक्त DBMS के उदाहरण SQL, MySQL, Postgre SQL,SQL सर्वर, Oracle आदि हैं।

DBMS का पूरा नाम डेटाबेस मैनेजमेंट सिस्टम है। DBMS, प्रोग्राम्स का एक पूरा संग्रह होता है जो कि users को, डेटाबेस को create और maintain करने के लिए योग्य (enable) बनाता है।

DBMS एक सॉफ्टवेयर सिस्टम है जो कि हमें निम्नलिखित सुविधाएँ उपलब्ध कराता है:

Defining– डेटाबेस में संधारित डाटा के लिए डाटा का प्रकार (data types), सरंचना (structures) और शर्तें (constraint) निर्देशित करता है।

Constructing– डेटा को किसी स्टोरेज डिवाइस में स्टोर करने की प्रक्रिया को DBMS के द्वारा नियंत्रित किया जाता है।

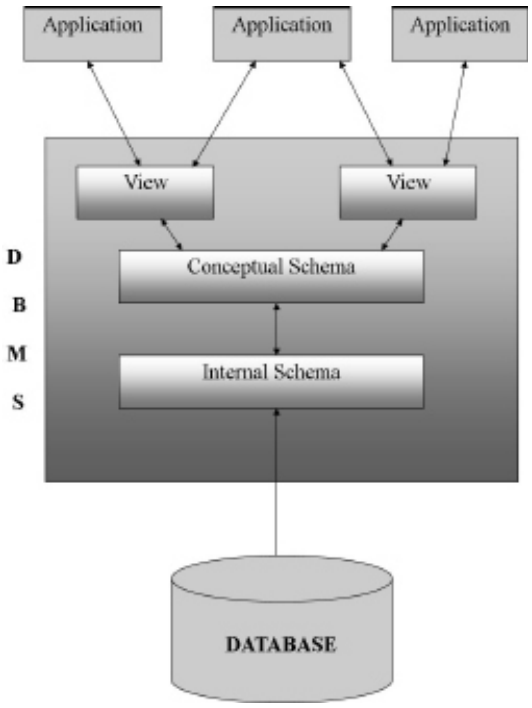
Manipulating– इसमें डेटाबेस में उपस्थित डेटा को पुनः प्राप्त (retrieve) तथा अद्यतन (update) किया जाता है और रिपोर्ट्स को तैयार किया जाता है ।

6.2 डेटाबेस अब्स्ट्रक्शन (Database Abstraction)

डेटाबेस सिस्टम उपयोगकर्ताओं को केवल उनकी आवश्यकतानुसार डेटा उपलब्ध करवाता है और मेमोरी में कैसे डेटा संग्रहीत और प्रबंधित होता है की जानकारी छुपाता है। यह अवधारणा / प्रक्रिया डेटाबेस एबस्ट्रैक्शन कहलाती है।

6.2.1डेटाबेस मैनेजमेंट सिस्टम आर्किटेक्चर (Database Management System Architecture)

डेटाबेस मैनेजमेंट सिस्टम त्रि स्तरीय स्कीमा आर्किटेक्चर (schema architecture) को निरूपित (describe) करता हैं। इसमें तीन सतह/स्तर (Levels) होते है। यह निम्नलिखित हैं:–



चित्र 6.1 डेटाबेस मैनेजमेंट सिस्टम आर्किटेक्चर

1. PHYSICAL LEVEL अथवा INTERNAL LEVEL

यह निरूपित करता है कि डेटाबेस में डेटा कैसे स्टोर होता है?यह abstraction का सबसे निचला सतह/स्तर (lowest level) है।

2. CONCEPTUAL अथवा LOGICAL LEVEL

Conceptual abstraction का अगला ऊँचा लेवल (next higher level) है। यह सतह/स्तर (level) निरूपित करता है कि डेटाबेस में क्या डेटा स्टोर रहता है और उनके मध्य क्या सम्बन्ध (relationship) है ?यह डेटाबेस प्रबन्धक (Database administrator) का कार्य होता है।

3. EXTERNAL अथवा VIEW LEVEL

ज्यादातर डेटाबेस उपयोगकर्ता (user) पूरे डेटाबेस का उपयोग नही करते है लेकिन उसका कुछ भाग ही पढ़ते (read) है। इसलिए वे उस भाग के view level को उपलब्ध कराते है। यह abstraction का सबसे उच्चतम सतह/स्तर (highest level) है। यह किसी विशिष्ट ग्रुप के उपयोगकर्ता (users) के लिए डेटाबेस के एक भाग को निरूपित (describe) करता है।

डेटाबेस के विभिन्न views हो सकते है। डेटाबेस मैनेजमेंट सिस्टम को विभिन्न आधार पर वर्गीकृत करते है, जो कि निम्नलिखित है:–

1. Data models के आधार पर

1. Traditional models: रिलेशनल (Relational), नेटवर्क (Network) तथा हिरार्किकल (Hierarchical) मॉडल्स इसके अंतर्गत आते हैं।
2. Emerging models: ऑब्जेक्ट ओरिएंटेड (object–oriented) तथा ऑब्जेक्ट –रिलेशनल (object–relational) मॉडल्स इसके अंतर्गत आते हैं।

2. उपयोगकर्ताओं के आधार पर पर ∴ एकल उपयोगकर्ता (single user) और एकाधिक उपयोगकर्ता (multiple users)
3. केन्द्रीयकृत (centralized) और वितरित (distributed) उपयोगकर्ता
4. उद्देश्य (purpose) के आधार पर

6.2.2 डेटाबेस सिस्टम सरंचना (Database System Structure)

डेटाबेस सिस्टम को मॉड्यूल (Modules) में विभाजित किया गया हैं तथा हर मॉड्यूल पर System Control से सम्बन्धित जिम्मेदारी होती हैं। Database System को क्रियाओं के अनुरूप दो भागों में विभाजित किया गया हैं:–

(1) स्टोरेज मैनेजर (Storage Manager)

(2) क्वेरी प्रोसेसर (Query Processor)

स्टोरेज मैनेजर (**Storage Manager**) **:** स्टोरेज मैनेजर वह Module हैं जो Low Level Data तथा Application Programs व Queries के मध्य पारस्परिक क्रिया (Interaction) करता हैं। स्टोरेज मैनेजर डेटा के Relational व अधतन (updation) के लिये उत्तरदायी होता हैं। स्टोरेज मैनेजर में निम्न अवयव होते हैं–

(1) Authorization and Integrity Manager:– इसमें डाटा की पूर्णता वाली शर्ते (integrity Constraints) तथा उपयोगकर्ता (users) की अधिकारिता (authority) को भी सुनिश्चित किया जाता हैं।

(2) Transaction Manger:– यह Database की consistancy के लिये उत्तरदायी होता हैं तथा Transaction को लगातार बिना व्यवधान निष्पादित (execute) करने का कार्य भी करता हैं।

(3) File Manager:– इसके द्वारा मेमोरी में स्थान व डाटा स्ट्रक्चर को प्रदर्शित किया जाता हैं।

(4) Buffer Manager:– डाटा को मुख्य मेमोरी में लाने के लिये उत्तरदायी होता हैं।

स्टोरेज मैनेजर के द्वारा विभिन्न Data Structure को Implement किया जाता हैं:–

(1) Data files:– जिसमें डाटा Store रहता हैं।

(2) Data Dictionary: जिसमें डेटा के बारे में डेटा को रखा जाता हैं।

(3) Indices:–इससे डाटा को त्वरित प्राप्त किया जा सकता हैं।

क्वेरी प्रोसेसर **(Query Processor)**:– क्वेरी प्रोसेसर के अवयव हैं:–

(1) DDL Interpreter:– जो DDL निर्देशों को Interpret करता हैं तथा उन निर्देशों की परिभाषा को Data Dictionary में संधारित करता हैं।

(2) DML Compiler:– यह DML निर्देशों को Low–level Instruction में परिवर्तित करने का कार्य करता हैं।

(3) Query Evaluation Engine:– यह DML Compiler द्वारा परिवर्तित Low-Level

निर्देशों को निष्पादित (execute) करने का कार्य करता हैं।

6.3 डेटाबेस मैनेजमेंट सिस्टम (DBMS) के लाभ

1. No data redundancy and consistency: एक ही तरह के डेटा का बहुत सारी जगह प्रतिरूप (duplication) को डेटा redundancy कहते है। उदाहरणार्थ किसी विधार्थी की जानकारी यथा नाम, रोल नंबर, वर्तमान पता और पिता का नाम आदि एक तरह के डेटा का अलग-अलग होना, डेटा असंगति (inconsistency) का कारण बनती है। जिससे स्टोरेज और मूल्य बढता है। लेकिन डेटाबेस मैनजमेंट सिस्टम के कारण डेटा की पुनरावर्ती नहीं होती है ।

2. Restricting Unauthorized Access: डेटाबेस मैनजमेंट सिस्टम में डेटाबेस एडमिनिस्ट्रेटर DBA (Data Base Administrator) सुरक्षा (security) और अनधिकृत (unauthorized) उपयोग को प्रतिबंधित करने का कार्य करता है ।

3. Data Integrity and Security: डेटाबेस मैनजमेंट सिस्टम में सुरक्षा (security) और पूर्णता (integrity) का पूरा ध्यान रखा जाता है। डेटाबेस में किसी भी प्रकार के मान (value) को जोड़ने (insert) से पहले उसे कुछ शर्तों (constraint) को संतुष्ट करना आवश्यक होता है। डेटाबेस में प्रत्येक उपयोगकर्ता को सभी डेटा को access करने की अनुमति नही होती है। जिससे डेटा की पूर्णता (integrity) बनी रहती है।

4- Simple Access: डेटाबेस मैनजमेंट सिस्टम में डेटाबेस को आसानी से प्राप्त (access) किया जा सकता है।

6.4 डेटाबेस मैनजमेंट सिस्टम की विशेषताएँ (Characteristics of DBMS)

डेटाबेस मैनजमेंट सिस्टम की विशेषताएँ निम्नलिखित हैं।

1. डेटाबेस मैनजमेंट सिस्टम की सबसे बड़ी विशेषता से यह है कि इसमें डेटा की पुनरावर्ती (redundancy) को नियंत्रण (control) किया जा सकता है।
2. डेटाबेस मैनजमेंट सिस्टम में डेटा को साझा किया जा सकता है।
3. डेटाबेस मैनजमेंट सिस्टम में सुरक्षा (security) का पूरा ध्यान रखा जाता है।
4. डेटाबेस मैनजमेंट सिस्टम में प्रोसेसिंग की गति अच्छी है।
5. डेटाबेस मैनजमेंट सिस्टम में डेटा स्वतंत्र (independent) होता है।

6.5 Data Base Management System Application के उदाहरण

डेटाबेस का कई उपक्रमों में उपयोग हो रहा है कुछ उदाहरण निम्न है:

(1) बैंकिंग क्षेत्र में डेटाबेस में ही कार्य होता है। कोई भी ग्राहक अपनी बैंक की किसी भी शाखा से लेनदेन कर सकता है। डेटाबेस के उपयोग से पहले ग्राहक को किसी भी अपनी ही शाखा में जाना होता था। डेटाबेस के कारण ही नेट बैंकिंग आदि की सुविधा ग्राहक को प्राप्त हो रही है।

(2) एयरलाइन रिजर्वेशन सिस्टम भी डेटाबेस पर ही निर्भर है। ग्राहक जहाँ अपनी सुविधा से फ्लाइट की बुकिंग कर सकता है वही एयरलाइन के कर्मचारियों को भीं भिन्न भिन्न प्रकार की रिपोर्ट भी प्राप्त हो जाती है जिससे निर्णय तुरंत लिया जा सकता है।

(3) विश्वविद्यालय में विद्यार्थियों एवं शिक्षकों की पूरी जानकारी डेटाबेस में रहती है. विभिन्न पाठ्यक्रमों की जानकारी, किस पाठ्यक्रम में कितने विद्यार्थी अध्ययनरत है उनकी जानकारी तुरंत प्राप्त की जा सकती है। परीक्षा परिणाम सम्बन्धी कार्य भी तीव्रता से सम्पन्न होते है।

(4) रेलवे रिजर्वेशन हो या बस रिजर्वेशन सिस्टम यह भी डेटाबेस पर ही निर्भर है। जहाँ यात्रियों को तुरंत जानकारी उपलब्ध होती है वहीं सम्बंधित विभाग अपनी आवश्यकताओं के अनुरूप भिन्न भिन्न प्रकार की रिपोर्ट्स तुरंत प्राप्त करते है।

(5) अस्पतालों में डॉक्टरो और रोगियों की भिन्न भिन्न प्रकार की जाँच रिपोर्ट स्टोर करना रोग का विश्लेषण करना,दवाइयों की जानकारी रखना डेटाबेस से सुगमता से सम्पन्न हो जाता है।

6.6 संबंधपरक डेटाबेस मैनेजमेंट प्रणाली (Relational Data Base Mangement System)

डाटाबेस सिस्टम में जो विभिन्न तालिकाओं के बीच संबंध बनाए रखे जाते हैं उसे संबंधपरक डेटाबेस प्रबंधन प्रणाली (Relational Data Base Mangement System) कहा जाता है। RDBMS और DBMS भौतिक (physical) डेटाबेस में जानकारी को संग्रहीत करने के लिए उपयोग में लिया जाता है। डेटा की बड़ी मात्रा में होने पर संग्रहण और साथ ही उसे अच्छी तरह से प्रबंधन करने के लिए RDBMS की आवश्यकता है। एक संबंधपरक डेटा मॉडल में इंडेक्सेस, कुंजी, टेबल और अन्य टेबल के साथ उनके संबंधों की जानकारी होती है ।

1970 के दशक में एडगर फ्रैंक कोड्ड ने संबंधपरक डेटाबेस का सिद्धांत पेश किया था। कोड्ड ने संबंधपरक सिद्धांत या मॉडल के लिए तेरह नियमों का प्रतिपादन किया था। विभिन्न प्रकार की डेटा के बीच संबंध संबंधपरक मॉडल की मुख्य आवश्यकता है।

RDBMS, डेटाबेस प्रबंधन प्रणाली की अगली पीढ़ी के रूप में कहा जा सकता है। DBMS एक आधार मॉडल के रूप में एक संबंधपरक डेटाबेस सिस्टम में डेटा संग्रहीत करने के लिए उपयोग किया जाता है। हालाँकि DBMS के बजाय RDBMS का जटिल व्यावसायिक अनुप्रयोगों में उपयोग लिया जाता है ।

6.6.1 ऐंटिटी रिलेशनशिप मॉडल (Entity Relationship Model)

ऐंटिटी रिलेशनशिप मॉडल वास्तविक एंटिटी (real–world entity) और उनके बीच संबंधों की धारणा पर आधारित है। वास्तविक परिदृश्य को डेटाबेस मॉडल तैयार करते समय, ईआर मॉडल एंटिटी सेट, रिलेशन सेट, सामान्य एट्रिब्यूट और कंस्ट्रेंट्स बनाता है।

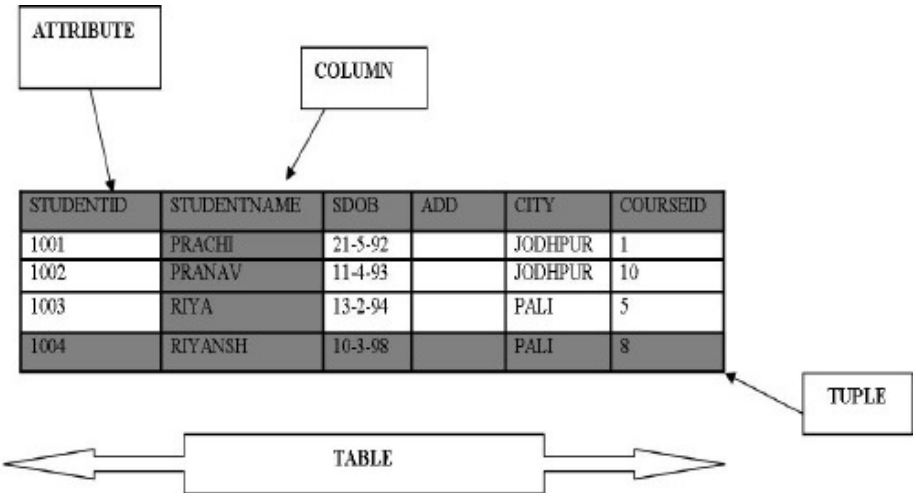
ईआर मॉडल संकल्पनात्मक डिजाइन के लिए उपयोग में लिया जाता है । ईआर माडल निम्न पर आधारित है-

1. **entities** और **attributes**
2. **entities** के मध्य सम्बन्ध (**Relationships**)

इन अवधारणाओं के नीचे समझाया हैं।

Entity : ईआर मॉडल में एंटिटी में वास्तविक दुनिया के गुणों वाली एट्रिब्यूट होती है। हर एट्रिब्यूट में इसकी मानों के समुच्चय (set) को डोमेन द्वारा परिभाषित किया जाता है।

उदाहरण के लिए एक स्कूल डेटाबेस में छात्र एक एंटिटी के रूप में होता है। छात्र के भिन्न भिन्न एट्रिब्यूट जैसे नाम, उम्र, वर्ग,आदि होते है।



चित्र स. 6.2 ऐंटिटी रिलेशनशिप मॉडल

रिलेशनशिप (**Relationship**) : एक से अधिक एंटिटी के मध्य तार्किक (logical) सम्बन्ध रिलेशनशिप कहलाता है। एंटिटी में संबंध विभिन्न तरीकों के साथ मैप की जाती हैं। मैपिंग कार्डीनलिटीज (mapping cardinalities) दो एंटिटी के मध्य सम्बन्धो की संख्या बताता है ।

मैपिंग कार्डीनलिटीज (mapping cardinalities) निम्न होती है:

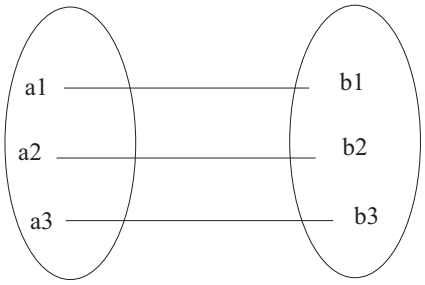
one to one

one to many

many to one

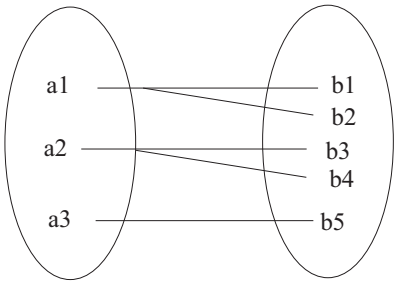
many to many

One to One Entity A का तत्व (element) अधिक से अधिक B Entity के एक तत्व से Connected हो तथा B की एक Entity का तत्व अधिक से अधिक A Entity के एक तत्व से सम्बन्धित हो।



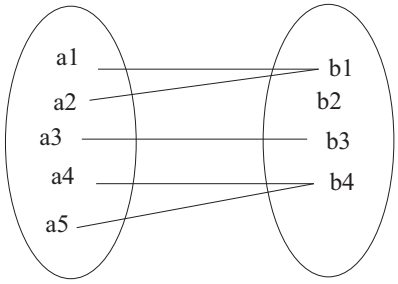
चित्र स. 6.3 वन टू वन रिलेशनशिप

One to Many Entity के तत्व B Entity के एक से अधिक तत्व से सम्बन्धित हो सकते हैं। परन्तु B Entity के तत्व अधिक से अधिक A Entity के एक ही तत्व से सम्बन्धित हो सकते हैं।



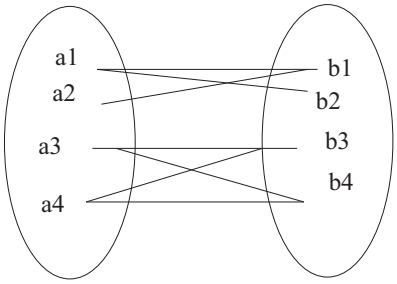
चित्र स. 6.4 वन टू मैनी रिलेशनशिप

Many to One A Entity के तत्व B Entity के एक ही तत्व से सम्बन्धित हो सकते हैं। परन्तु B Entity का एक तत्व एक A Entity के एक से अधिक तत्व से सम्बन्धित हो सकते हैं।



चित्र स. 6.5 मैनी टू वन रिलेशनशिप

Many to Many A Entity के तत्व B Entity के एक से अधिक तत्व से सम्बन्धित हो सकते हैं। और B Entity के तत्व भी A Entity के एक से अधिक तत्व से सम्बन्धित हो सकते हैं।



चित्र स. 6.6 मैनी टू मैनी रिलेशनशिप

डेटा : कच्चे तथ्य और आंकड़े जो एक संगठन (organization) के लिए उपयोगी होते हैं। हम डेटा के

आधार पर निर्णय नहीं ले सकते।

सूचना : अच्छी तरह से संसाधित डेटा को सूचना/जानकारी (information) कहा जाता है। हम जानकारी के आधार पर निर्णय ले सकते हैं। सूचना वर्गों का सेट है जो विशिष्ट डेटा तत्त्व का प्रतिनिधित्व करता है।

रिकार्ड : रिकॉर्ड संबंधित फील्ड्स का संग्रह होता है। रिकॉर्ड में फील्ड अलग डेटा प्रकार के हो सकते हैं।

फाइल : फाइल समान प्रकार के रिकॉर्ड्स के संग्रह को कहा जाता है।

टेबल : टेबल पंक्तियों और स्तंभों का संग्रह है जिसमें उपयोगी डेटा/जानकारी रखा जाता है। टेबल आम तौर पर निष्क्रिय इकाई है जो द्वितीयक मेमोरी (secondary storage) में रखा जाता है को संदर्भित करता है।

रिश्ता (Relation) : रिश्ता (पंक्तियों और स्तंभों का संग्रह) आम तौर पर एक सक्रिय इकाई है जिस पर हम विभिन्न कार्यों का प्रदर्शन कर सकते हैं।

टपल (Tuple) : एक रिश्ते (Relation) में एक पंक्ति (row) को टपल कहा जाता है। संबंध परक मॉडल (relational data base) में डेटा tuples(rows) के संदर्भ में प्रस्तुत किया जाता है।

डोमेन (Domain) : डेटाबेस डोमेन सभी स्वीकार्य मान का समूह डोमेन कहलाता है । उदाहरणः लिंग (gender) के लिए कोई फील्ड उस स्तंभ में प्रविष्टियों की अनुमति डोमेन {पुरुष, महिला} जहाँ केवल उन दो मान रहे हैं में से एक हो सकता है।

6.6.2 कुंजियाँ (Keys)

1 प्राथमिक कुंजी (primary key)

किसी relational table की प्राथमिक कुंजी (primary key) टेबल के प्रत्येक रिकॉर्ड को विशिष्ट रूप (uniquely) से बतलाता (identify) है।

साधारण प्राथमिक कुंजी (Simple primary key) : साधारण प्राथमिक कुंजी केवल एक फील्ड (field) से मिलकर बना होता है।

साधारण प्राथमिक कुंजी (primary key) को निरूपित (define) करना:–

1. साधारण प्राथमिक कुंजी अद्वितीय (unique) होती है।
2. किसी भी टेबल में केवल एक ही primary key हो सकती है।
3. ये single या multi column हो सकती है, multi column साधारण प्राथमिक कुंजी को हम संयोजन प्राथमिक कुंजी composite primary key कहते है।
4. संयोजन प्राथमिक कुंजी में अधिकतम 16 column हो सकते है।

2 Foreign key

रिलेशनल डेटाबेस टेबल में एक foreign key, स्तम्भो (columns) का एक समूह होता है जो कि दो tables में डेटा के मध्य सम्बन्ध (link) उपलब्ध कराता है। कहीं कहीं पर foreign key को referencing key भी कहा जाता है।

जब किसी एक प्राथमिक कुंजी को किसी दूसरे टेबल में प्राथमिक कुंजी के रूप में प्रयोग करते है तो उसे foreign key कहते हैं। foreign key, डेटा में पूर्णता (integrity) को बनाये रखने (maintain) के लिए तरीके (method) उपलब्ध कराता है।

3.संयोजन प्राथमिक कुंजी (Composite Prinary key)

जब कोई प्राथमिक कुंजी बहुत सारें attributes से मिलकर बनी होती है तो उसे हम संयोजित प्राथमिक कुंजी कहते है। रिलेशनल डेटाबेस टेबल में संयोजन प्राथमिक कुंजी दो या दो अधिक columns का समूह होता है जो कि table में प्रत्येक row को uniquely identify करता है।

4 सुपर की (Super key)

RDBMS में एक सुपर कुंजी स्तम्भो का एक संयोजन (combination) होता है जो कि रिकॉर्ड को अद्वितीय रूप से (uniquely) निरूपित करता है।

5 उम्मीदवार कुँजी (candidate key)

कैंडिडेट की (candidate key) वह कुंजी होती है जो कि एक टेबल में प्राथमिक कुंजी की candidate की तरह कार्य करती है। candidate key प्राथमिक कुंजी की जरूरतों को पूर्ण करती है।

6.7 SQL

संरचित क्वेरी भाषा (SQL) डेटाबेस की भाषा है। वर्तमान में सभी रिलेशनल डेटाबेस यथा – MS Acess, Microsoft SQL Server और Oracle जैसे सॉफ्टवेयर SQL का ही उपयोग करते है।

6.7.1 डेटाबेस भाषा (Database languages)

किसी भी सिस्टम में डेटाबेस को बनाने (create) और प्रबंधित (maintain) करने के लिए डेटाबेस भाषाओं (database languages) का उपयोग किया जाता है। डेटाबेस में दो तरह की भाषाओं का उपयोग किया जाता है।

डेटा डेफिनेशन लैंग्वेज (Data Definition Language):

DDL का पूरा नाम Data Definition Language है, यह conceptual schema को निरूपित (define) करने के लिए काम में लिया जाता है तथा यह इस बात की जानकारी भी देता है कि physical devices में इस schema को कैसे क्रियान्वित (implement) किया जाता है। SQL में जो सबसे महत्वपूर्ण DDL statements है, वो निम्न हैं:

1. **CREATE**:– डेटाबेस में objects को बनाने (create) के लिये।
2. **ALTER**:– डेटाबेस के structure में परिवर्तन करने के लिए।
3. **DROP**:– डेटाबेस में से objects को हटाने के लिए।
4. **COMMENT**:– data dictionary में टिप्पणी (comments) के लिए।
5. **RENAME**:– object के नाम को पुनः नामकरण (rename) हेतु ।

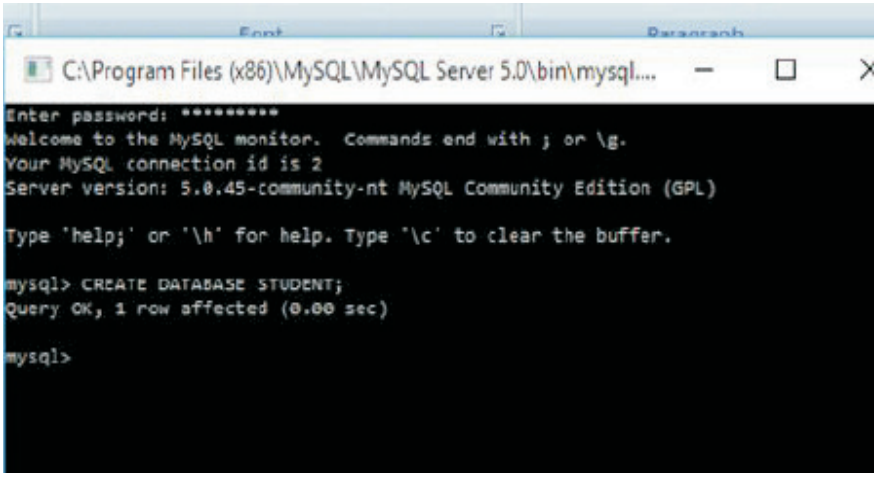
डेटा मैनीपुलेशन लैंग्वेज (Data Manipulation Language)

DML का पूरा नाम Data Manipulation Language है, यह भाषा डेटाबेस में डेटा के परिचालन (manipulate) करने के काम आती है। इसके कुछ उदाहरण निम्नलिखित है:–

1. **SELECT**:– एक डेटाबेस में से डेटा को प्राप्त (retrieve) करने हेतु ।
2. **INSERT**:– टेबल में डेटा को जोड़ने (insert) हेतु।
3. **UPDATE**:– टेबल में मौजूदा डेटा को अधतन (update) हेतु।
4. **DELETE**:– टेबल में से रिकार्ड्स को हटाने (delete) हेतु ।

6.8 MySQL

MySQL एक डेटाबेस प्रबंधन प्रणाली है कि यह relational डेटाबेस को प्रबंधित करने के काम आता है। यह ओपन सोर्स सॉफ्टवेयर है।



चित्र स. 6.7 माई एसक्यूएल इन्टरफेस

6.8.1 MySQL निर्देश (MySQL Commands)

डेटाबेस बनाना (Create Database)

डेटा के साथ कुछ भी कार्य करने से पहले एक डेटाबेस बनाने की जरूरत है। डेटाबेस डेटा का एक कंटेनर है। यह विद्यार्थियों, विक्रेताओं, कर्मचारियों, ग्राहकों अथवा जिसके बारे में भी हम सोच सकते हैं का संधारण (store) करता है। MySQL में डेटाबेस एक संग्रह है जिसमें टेबल, डेटाबेस व्यू ट्रिगर, संग्रहित प्रक्रिया (stored procedure) इत्यादि है जिनके द्वारा डेटा को संग्रह (store) और परिचालन (manipulation) करने के लिए प्रयोग में लिया जाता है। MySQL में डेटाबेस बनाने के लिए निम्न कथन (syntax) का उपयोग करें :

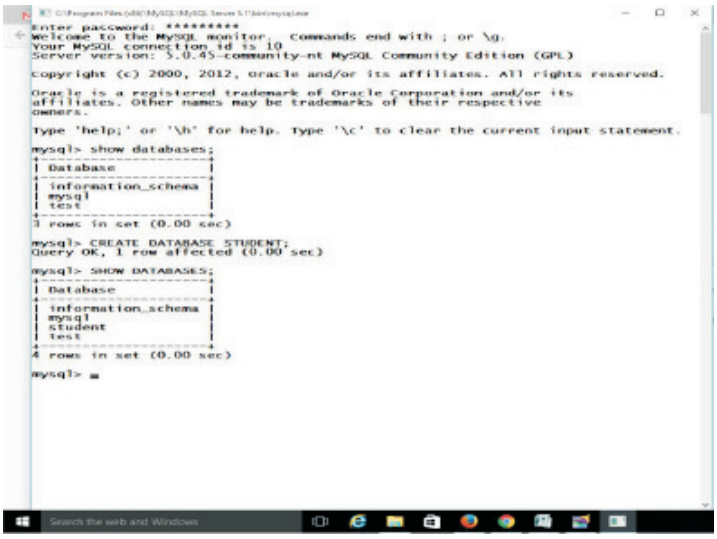
CRETAE DATABASE [IF NOT EXIST] [Name of Database]

डेटाबेस कथन के पश्चात डेटाबेस का नाम है जिसे हम बनाना चाहते है वह लिखना होता है। डेटाबेस नाम के रूप में अर्थपूर्ण (meaningful) और जहा तक संभव हो वर्णनात्मक (descriptive) होना चाहिए।

IF NOT EXIST वैकल्पिक उपनियम (clause) है। इसके उपयोग से नया डेटाबेस बनाते समय पहले से ही डेटाबेस सर्वर में मौजूद डेटाबेस के नाम से बनाने की त्रुटि से रोकता है। डाटाबेस सर्वर में एक ही नाम के दो डेटाबेस नहीं हो सकते है ।

उदाहरण के लिए विद्यार्थियों हेतु student डेटाबेस बनाने के लिए CREATE DATABASE कथन इस प्रकार है

CREATE DATABASE STUDENT;



चित्र स. 6.8

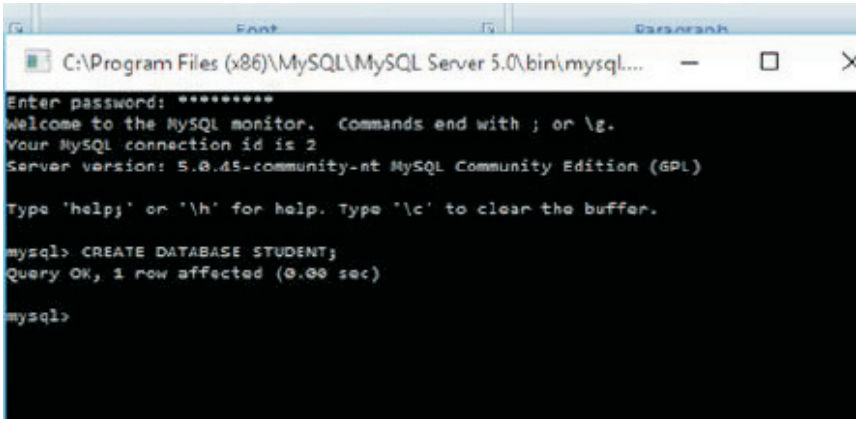
इस कथन को क्रियान्वित करने के बाद MySQL नए डेटाबेस सफलतापूर्वक बनाया या नहीं बनाया गया है के लिए एक संदेश देता है।

डेटाबेस दर्शाना (SHOW DATABASE)

SHOW DATABASE कथन सभी डेटाबेस को MySQL डेटाबेस सर्वर में प्रदर्शित करता है। आप आपके द्वारा बनाए गए डेटाबेस की जाँच करने के लिए या आप एक नया डेटाबेस बनाने से पहले डेटाबेस सर्वर पर सभी डेटाबेस को देखने के लिए SHOW DATABASE कथन का उपयोग कर सकते हैं।

तीन डेटाबेस MySQL डेटाबेस सर्वर में है। Information_schema और mysql हैं जो पहले से उपलब्ध हैं,MySQL स्थापित डिफॉल्ट डेटाबेस है और हमारे द्वारा बनाए गया नया डेटाबेस student है।

SHOW DATABASES;



चित्र स. 6.9 डेटाबेस बनाना

डेटाबेस के साथ काम करने के लिए चयन करना (USE DATABASE)

किसी विशिष्ट डेटाबेस के साथ काम करने से पहले, MySQL को बताना होगा कि कौनसे डेटाबेस का उपयोग हमें करना है। USE कथन का उपयोग कर के इच्छित डेटाबेस के साथ काम किया जा सकता है आप निम्नानुसार USE कथन का उपयोग कर STUDENT डेटाबेस का चयन कर सकते हैं:

USE STUDENT;

अब नई टेबल को बनाना, डेटा क्वेरी,या संग्रहीत प्रक्रिया को बुलाना इत्यादि वर्तमान डेटाबेस अथार्त STUDENT पर क्रियान्वित होगा

डेटाबेस को हटाना (DATABASE DROP)

डेटाबेस को हटाने का मतलब है स्थायी रूप से डेटाबेस को हटाना। सभी डेटा और डेटाबेस के अंदर संबंधित ऑब्जेक्ट्स को स्थायी रूप से हटा दिया जाता हैं और यह पूर्ववत नहीं किया जा सकता। इसलिए यह अतिरिक्त चेतावनी भी देता हैं साथ ही इस निर्देश को सावधानी से निष्पादित करना चाहिए। किसी डेटाबेस को हटाने के लिए आप DROPDATABASE कथन का निम्नानुसार का उपयोग करे

DROP DATABASE [IF EXISTS] database_name;

अगर हमें STUDENT डाटाबेस को हटाना है तो निम्न कथन लिखना होगा।

DROP DATABASE STUDENT;

MySQL डेटा प्रकार (Data Types)

डेटाबेस के टेबल में एक से अधिक स्तंभ विशिष्ट डेटा प्रकार के हो सकते है जैसे अंकीय (NUMERIC) या स्ट्रिंग (string) के रूप में। MySQL अंकीय या स्ट्रिंग के अलावा अधिक तरह के डेटा प्रकार प्रदान करता है। प्रत्येक डेटा प्रकार MySQL में निम्नलिखित विशेषताओं द्वारा निर्धारित किया जा सकता है :

- मान (VALUES) किस तरह के मूल्यों का प्रतिनिधित्व करता है।
- क्या मान (VALUES) एक निश्चित–लम्बाई या चर लंबाई का है।
- मान के डेटा प्रकार को अनुक्रमित किया जा सकता है या नहीं।

अंकीय डेटा (Numeric Data)

MySQL में SQL की भांति सभी तरह के अंकीय डेटा प्रकार जैसे integer, fixed point और floating point उपलब्ध है। इसके अलावा MySQL बिट (BIT) डेटा प्रकार बिट फील्ड मान संग्रहीत करने का अवसर प्रदान करता है।

अंकीय डेटा प्रकार signed और unsigned हो सकते है.

अंकीय डेटा प्रकार की सारणी निम्न है ::

Numeric Types	**Description**
TINYINT	A very small integer
SMALLINT	A small integer
MEDIUMINT	A medium-sized integer
INT	A standard integer
BIGINT	A large integer
DECIMAL	A fixed-point number
FLOAT	A single-precision floating point number

DOUBLE	A double-precision floating point number
BIT	A bit field

स्ट्रिंग डेटा (String Data)

MySQL में स्ट्रिंग डेटा प्रकार साधारण टेक्स्ट से लेकर बाइनरी डेटा जैसे छविया (images) और फाइलें स्टोर कर सकता है। स्ट्रिंग डेटा प्रकार तुलना और प्रतिमान के आधार पर मिलान करके खोजने का कार्य कर सकता है। निम्न तालिका MySQL में स्ट्रिंग डेटा प्रकार को दर्शाती है :

String Types	**Description**
CHAR	A fixed-length nonbinary (character) string
VARCHAR	A variable-length non-binary string
BINARY	A fixed-length binary string
VARBINARY	A variable-length binary string
TINYTEXT	A very small non-binary string
TEXT	A small non-binary string
MEDIUMTEXT	A medium-sized non-binary string
LONGTEXT	A large non-binary string
ENUM	An enumeration; each column value may be assigned one enumeration member
SET	A set; each column value may be assigned zero or more set members

दिनांक और समय डेटा प्रकार (Date & Time Data Types)

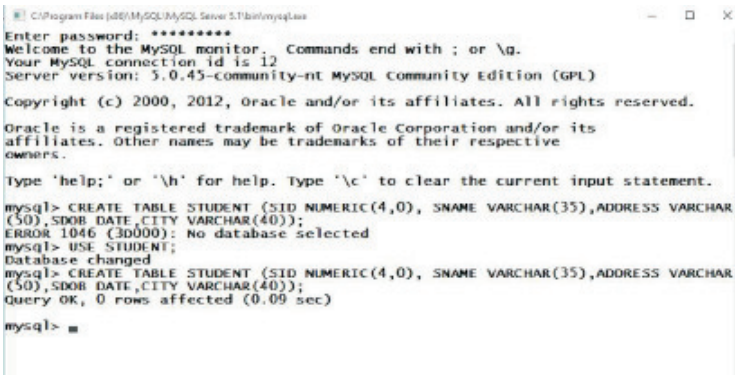
MySQL दिनांक और समय डेटा प्रकार के साथ ही दिनांक और समय का एक संयोजन डेटा प्रकार भी प्रदान करता है। इसके अलावा MySQL टाइमस्टैम्प डेटा प्रकार द्वारा किसी टेबल में एक पंक्ति (row) के परिवर्तन को ट्रैक करने की सुविधा भी प्रदान करता है । अगर आप तारीख और महीना के बिना सिर्फ वर्ष (year) ही संग्रहीत करना चाहते हैं तो वर्ष डेटा प्रकार (Year) का उपयोग कर सकते हैं। निम्न तालिका MySQL में दिनांक और समय डेटा प्रकार को दर्शाती है।

Date and Time Types	**Description**
DATE	A date value in ;YYYY-MM-DD' format
TIME	A time value in ;hh:mm:ss' format
DATETIME	A date and time value in ;YYYY-MM-DD hh:mm:ss' format
TIMESTAMP	A timestamp value in ;YYYY-MM-DD hh:mm:ss' format
YEAR	A year value in YYYY or YY format

डेटाबेस के भीतर किसी नई टेबल का निर्माण करना (CREATE TABLE)

डेटाबेस के भीतर किसी नई टेबल को बनाने के लिए MySQL में टेबल बनाने के कथन का उपयोग करना होगा । टेबल बनाने का कथन MySQL में सबसे जटिल कथन है। टेबल बनाने का कथन निम्नानुसार हैं

CREATE TABLE [IF NOT EXISTS] table_name (List of Columns)

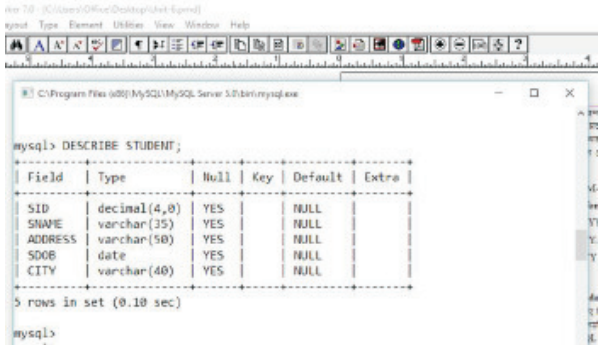


चित्र स. 6.10

टेबल के एट्रीब्यूट्स एवं उसके डेटा प्रकार देखने के लिए DESCRIBE निर्देश का उपयोग किया जाता है :

DESCRIBE TABLE_NAME

उदाहरणार्थ DESCRIBE STUDENT



चित्र स. 6.11 डेटाबेस दर्शाना

टेबल को हटाना (DROP TABLE)

मौजूदा टेबल को हटाने के लिए DROP TABLE कथन का उपयोग करें। DROP TABLE का कथन(सिंटैक्स) निम्नानुसार हैं:

DROP [TEMPORARY] TABLE [IF EXISTS] table_name [, table_name]

उदाहरणार्थ हमें STUDENT टेबल को हटाना है तो निम्न कथन को लिखना होगा

DROP TABLE STUDENT;

DROP TABLE कथन किसी टेबल और इसके डेटा स्थायी रूप से डेटाबेस से हटा देता है। एकाधिक टेबल का उपयोग कर एकल DROP TABLE कथन से भी हटा सकते हैं, उस दशा में प्रत्येक टेबल एक अल्पविराम (,) द्वारा अलग किया जाता हैं।

टेबल में परिवर्तन (ALTER TABLE)

मौजूदा TABLE की संरचना को बदलने के लिए ALTER TABLE कथन का उपयोग करते है। DROP TABLE कथन एक स्तंभ जोड़ने, एक स्तंभ को हटाने, स्तंभ का डेटा प्रकार बदलने प्राथमिक कुंजी जोड़न, टेबल का नाम बदलने और कई अन्य कार्य करने की अनुमति देता है। ALTER TABLE कथन निम्न है :

ALTER TABLE table_name action1[,action2,…]

किसी मौजूदा टेबल की संरचना को बदलने के लिए:

– टेबल का नाम, जो आप परिवर्तन के बाद बदलना चाहते हैं, निर्दिष्ट करें।

– टेबल के लिए लागू करने के लिए इच्छित क्रियाओं के एक सेट की सूची। कोई भी क्रिया यथा एक नए स्तंभ को जोड़ना, प्राथमिक कुंजी को जोड़ना, टेबल का का नाम बदलना आदि कुछ भी हो सकता है। किसी एकल ALTER TABLE कथन में एक से अधिक क्रियाएँ लागू करने के लिए अनुमति प्रदान करता है, प्रत्येक क्रिया को एक अल्पविराम (,) द्वारा अलग किया जाता है।

टेबल में नया स्तंभ जोडना (ADD COLUMN TO TABLE)

कुछ नई आवश्यकताओं के कारण किसी तालिका में कोई नया स्तंभ जोड़ने के लिए ALTER TABLE कथन का उपयोग करते है, उदाहरणार्थ STUDENT टेबल में जन्म तिथि स्तम्भ जोड़ने की जरूरत है। इस स्थिति में ALTER TABLE का उपयोग निम्नानुसार कोई नया स्तंभ जोड़ने के लिए कर सकते हैं:

ALTER TABLE STUDENT ADD COLUMN DOB DATE;

DOB स्तम्भ मौजूदा टेबल के अंतिम स्तम्भ के बाद जुड़ जायेगा।

अगर किसी स्तम्भ के बाद में नया स्तम्भ जोड़ना है तो AFTER और उसके बाद उस स्तम्भ का नाम लिखना होगा। यदि PINCODE स्तम्भ CITY स्तम्भ के बाद जोड़ना है तो ALTER TABLE कथन निम्न प्रकार से लिखा जायेगा।

ALTER TABLE STUDENT ADD COLUMN PINCODE NUMERIC(6,0) AFTER CITY;

स्तम्भ को हटाना (Drop Column)

यदि किसी स्तम्भ को हम टेबल से हटाना चाहें तो हटा सकते है उदाहरणार्थ मान लीजिए आपके टेबल में PINCODE स्तम्भ है और अब आप की आवश्यकता है कि इसे टेबल में संग्रह नहीं करना चाहते हैं और आप इसे हटाना चाहते है तब STUDENT टेबल से PINCODE स्तंभ को निकालने के लिए निम्न कथन लिखा जायेगा :

ALTER TABLE STUDENT DROP COLUMN PINCODE;

टेबल का नाम बदलना (Renaming Table)

किसी टेबल का नाम बदलने के लिए ALTER TABLE कथन का उपयोग कर सकते हैं। यदि STUDENT टेबल का नाम STUINFO करना है तो कथन इस प्रकार होगा

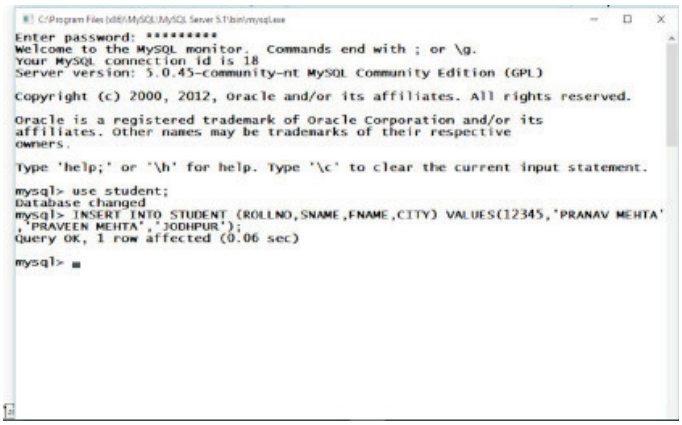
ALTER TABLE STUDENT RENAME TO STUINFO;

इन्सर्ट कथन (INSERT Statement)

INSERT कथन किसी टेबल में एक या अधिक पंक्तियाँ (row) को सम्मिलित करने के लिए उपयोग में लिया जाता है। INSERT कथन का सिंटैक्स है

INSERT INTO TABLE [COLUMN1, COLUMN2,.....] VALUES (VALUE1,VALUE2.....)

स्तम्भ की सूची वैकल्पिक है. सबसे पहले INSERT INTO के बाद निर्दिष्ट टेबल का नाम और लघुकोष्ठक के अंदर अल्पविराम से अलग किए गए स्तंभों की सूची लिखे । VALUES कीवर्ड के पश्चात लघु कोष्ठक के अंदर अल्पविराम से अलग किए गए स्तंभों के मान लिखे।



चित्र स. 6.12 टेबल में डेटा संधारण करना

उदाहरणार्थ, अगर हमें STUDENT टेबल में किसी विद्यार्थी के डेटा संधारित (Store) करना है उस दशा में निम्न INSERT कथन होगा

INSERT INTO STUDENT (ROLLNO,SNAME,FNAME,CITY)
VALUES (12345,'PRANAV MEHTA','PRAVEEN
MEHTA','JODHPUR');

स्तम्भ की सूची वैकल्पिक है अतः इसके बिना भी INSERT कथन लिख सकते है वह निम्न प्रकार होगा

INSERT INTO STUDENT VALUES (12345,'PRANAV
MEHTA','PRAVEEN MEHTA','JODHPUR');

इंसर्ट एकाधिक पंक्तियां (INSERT Multiple Rows)

किसी टेबल में एकाधिक पंक्तियाँ सम्मिलित करने के लिए, INSERT कथन का निम्न सिंटैक्स का उपयोग करें:

INSERT INTO table (column1,column2...) VALUES (value1,value2,...), (value1,value2,...), ……

उदाहरणार्थ, अगर हमें STUDENT टेबल में एक से अधिक विद्यार्थीयों जैसे Pranav, Prachi और Rudra Pratap के डेटा संधारित करना है तो सिर्फ एक INSERT कथन के उपयोग से भी यह संभव है।

INSERT INTO STUDENT VALUES (12345,'PRANAV MEHTA','PRAVEEN MEHTA','JODHPUR'), (12346,'PRACHI MEHTA','PRAVEEN MEHTA','JODHPUR'), (12347,'RUDRA PRATAP','RAKESH MEHTA','JODHPUR');

इसमें प्रत्येक पंक्ति (row) के मान सूची को एक अल्पविराम के द्वारा अलग किया जाता है –

INSERT कथन में मान SELECT कथन के द्वारा भी डाल सकते है। इस विशेषता से

अगर किसी टेबल की पूर्ण अथवा आंशिक रूप से कॉपी करना है तो संभव है

INSERT INTO table_1 SELECT c1, c2, FROM table_2;

उदाहरणार्थ STUDENT टेबल के सभी डाटा कुछ कारणों से एक अन्य टेबल TEMPSTU में भी चाहिए, उस दशा में INSERT कथन निम्न होगा

INSERT INTO TEMPSTU SELECT * FROM STUDENT;

अद्यतन (UPDATE)

डेटाबेस के साथ काम करते समय डेटा अद्यतन (Update) करना सबसे महत्वपूर्ण कार्यों में से एक है। किसी टेबल में मौजूदा डेटा को अद्यतन करने के लिए UPDATE कथन का उपयोग करते है । हम अद्यतन कथन का उपयोग एकल पंक्ति (Row), पंक्तियों के समूह, या किसी टेबल में सभी पंक्तियों के स्तंभ मान बदलने के लिए कर सकते हैं। MySQL अद्यतन कथन निम्न है:

```
UPDATE table_name
SET
  column_name1 = expr1,
  column_name2 = expr2,
  ...
WHERE
  condition;
```

सर्वप्रथम डेटा अद्यतन (update) करने के लिए इच्छित टेबल का नाम UPDATE कीवर्ड के बाद निर्दिष्ट करें। दूसरा, जिस स्तंभ को संशोधित और नया मान (value) देना चाहते हैं उसे SET कथन के साथ निर्दिष्ट करें। एकाधिक स्तंभों को अद्यतन करने के लिए अल्पविराम द्वारा अलग करें।

तीसरा, कौनसी पंक्तियाँ का अद्यतन किया जाएगा उसे WHERE खंड में एक शर्त (Condition) का उपयोग कर निर्दिष्ट करें। WHERE खंड वैकल्पिक है। यदि आप WHERE खंड छोड़ देंगे तो टेबल में सभी पंक्तियों का अद्यतन होगा।

WHERE खंड अत्यंत महत्वपूर्ण है इसका उपयोग सावधानी से करना चाहिए । कभी कभी सिर्फ एक पंक्ति को बदलना चाहते है WHERE खंड को भूलने से सभी पंक्तियाँ अद्यतन हो सकती है।

अद्यतन किसी एकल स्तंभ में

इस उदाहरण में, हम प्रणव के ईमेल को नए ई–मेल pranav@boser-edu-in से अद्यतन करना चाहते हैं। ईमेल सफलतापूर्वक अद्यतन करने के लिए सबसे पहले सुनिश्चित करें कि प्रणव का ईमेल प्राप्त करने के लिए SELECT कथन का उपयोग विद्यार्थीयों की टेबल से चयन करने के लिए करना होगा।

UPDATE STUDENT SET EMAIL= ;pranav@boser-edu-in' WHERE SNAME='PRANAV';

एकाधिक स्तम्भों में अद्यतन (UPDATE Multple Columns) एकाधिक स्तंभ मानों को अद्यतन करने के लिए SET खंड में निर्दिष्ट करने की जरूरत है। उदाहरण के लिए निम्न कथन, अंतिम नाम और ई–मेल दोनों स्तंभ एक रोल नंबर 205502 का अद्यतन करता है।

UPDATE SET LASTNAME='MOHNOT', EMAIL=' pranav@boser-edu-in'WHERE ROLLNO=205502

डिलीट कथन (DELETE Statement)

एक या अधिक डेटाबेस टेबल से डेटा निकालने के लिए DELETE कथन का उपयोग किया जाता

है। किसी टेबल से डेटा को निकालने के लिए DELETE कथन का उपयोग करना होता है । एकल DELETE कथन का उपयोग न केवल एक ही टेबल से बल्कि एकाधिक टेबल से भी रिकॉर्ड को निकालने के लिए भी किया जा सकता है।

DELETE FROM table
[WHERE conditions]
[ORDER BY ...]

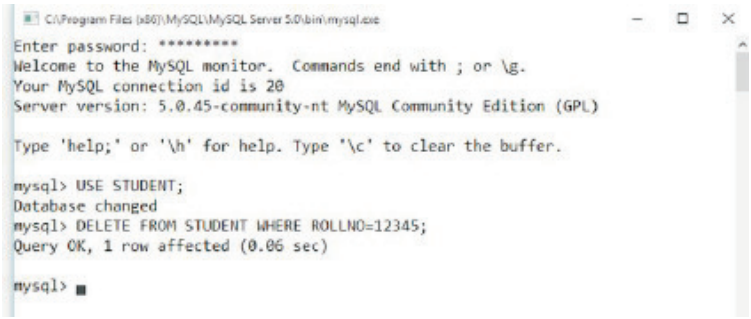
किसी एक टेबल से डेटा को हटाने/निकालने के लिए निम्न DELETE कथन का उपयोग करें:

DELETE FROM table
[WHERE conditions]
[ORDER BY ...] [LIMIT rows]

WHERE CLAUSE कौन–सी पंक्तियाँ आप हटाना चाहते हैं को निर्दिष्ट करता है। अगर रिकॉर्ड WHERE शर्त को पूरा करता है, तो यह टेबल से स्थायी रूप से हटा दी जाती है। यदि WHERE खंड काम में न लिया जाये तब टेबल में से सभी रिकार्ड्स हटा दिए जाते है । ROW_COUNT() फंक्शन DELETE कथन द्वारा हटाए गए रिकार्ड्स की संख्या बताता है।

उदाहरणार्थ रोल नंबर 123456 के विधार्थी के रिकॉर्ड को हटाने के लिए निम्न DELETE कथन उपयोग लेना होगा

DELETE FROM STUDENT
WHERE ROLLNO=123456



चित्र स. 6.14 डेटाबेस से रिकार्ड हटाना

STUDENT टेबल से सभी विद्यार्थियों के रिकार्ड्स को हटाने के लिए, आप WHERE खंड के बिना DELETE कथन का उपयोग निम्नानुसार करें:

DELETE FROM STUDENT;

उपरोक्त उदाहरण से सभी विद्यार्थियों के रिकार्ड्स STUDENT टेबल से हटा दिए गए

एकाधिक टेबल से रिकॉर्ड को हटाना

एकाधिक टेबल से रिकॉर्ड हटाने के लिए निम्न DELETE कथन का उपयोग करना होगा

DELETE table_1, table_2,...

FROM table-refs
[WHERE conditions]
DELETE FROM table_1, table_2,...
USING table-refs
[WHERE conditions]

उदाहरणार्थ एक पाठ्यक्रम बंद होने की दशा में उस पाठ्यक्रम के सभी विधार्थियों के रिकॉर्ड तथा उस पाठ्यक्रम का रिकॉर्ड हटाना हो तो DELETE कथन निम्न होगा

```
DELETE STUDENT,COURSE
FROM STUDENT,COURSE
WHERE STUDENT.COURSEID = COURSE.COURSEID AND
COURSE.COURSEID=1;
```

सलेक्ट कथन (SELECT Statement)

SELECT कथन टेबल से डेटा प्राप्त करने के लिए काम में लिया जाता है। एक टेबल पंक्तियाँ (Rows) और स्तंभ (Column) का संयोजन होता है जोकि एक स्प्रेडशीट की तरह होते हैं। अधिकांशतः हम पंक्तियों का समूह या स्तंभों का समूह अथवा इन दोनों का एक संयोजन देखना चाहते हैं। SELECT कथन का प्रतिफल परिणाम समूह (Result Set) कहलाता है।

परिणाम समूह पंक्तियों (rows) की सूची है जिसमे समान संख्या के स्तम्भ (column) होते है। उदाहरणार्थ विधार्थियों की टेबल देखे , जिसमे स्तंभ (Col) है – रोल नंबर,विद्यार्थी का नाम , जन्म तिथि, ईमेल , पाठ्यक्रम कोड।

ROLLNO	STUNAME	DOB	EMAIL	COURSE CODE
100201	Pranav	2000-2-15	pranav@boser.edu.in	1
100202	Prachi	2000-3-10	prachi@boser.edu.in	2
100203	Riya	2001-5-21	riya@boser.edu.in	1
100204	Rudra Pratap	2000-4-25	rudra@boser.edu.in	1
100205	Riyansh	2002-3-11	riyansh@boser.edi.in	2

हमें कौनसे स्तंभों और पंक्तियों को देखना चाहते है उसके लिए SELECT कथन का उपयोग किया जाता है। उदाहरण के लिए यदि केवल विद्यार्थी का नाम और उसका ईमेल देखना चाहते या सिर्फ विद्यार्थी का नाम और उसकी जन्म तिथि की जानकारी देखना चाहते हैं तब SELECT कथन ऐसा करने में मदद करता है। SELECT कथन का सिंटैक्स निम्न है :

```
SELECT
        column_1, column_2, column_3 ...
FROM
        table_1
        [INNER | LEFT |RIGHT] JOIN table_2 ON conditions
WHERE
```

Conditions

GROUP BY column_1

HAVING group_conditions

ORDER BY column_1

SELECT कथन के कई खंड (Clause) होते हैं, जिनका उल्लेख निम्नानुसार है :

SELECT और उसके बाद अल्पविराम से अलग किए गए स्तंभों (column) की एक सूची या तारांकन (*) यह दर्शाता है कि सभी स्तंभों (columns) को देखना चाहते हैं।

JOIN टेबल के संबंधो की शर्तों पर आधारित अन्य तालिकाओं से डेटा प्राप्त करता है।

WHERE परिणाम समूह (Result Set) में पंक्तियों (Rows) का फिल्टर है।

GROUP BY पंक्तियों का एक समूह बनाता है और aggregate फंक्शन प्रत्येक समूह पर होता है।

HAVING खंड (clause) GROUP BY द्वारा बने समूहों को फिल्टर करता है।

ORDER BY क्रमबद्ध / सॉर्ट करने के लिए स्तंभ की एक सूची निर्दिष्ट करता है।

SELECT और FROM खंड कथन में आवश्यक हैं, अन्य भाग वैकल्पिक हैं।

SELECT कथन डेटा की एक टेबल में आंशिक क्वेरी की अनुमति देता है, उसके लिए SELECT खंड में अल्पविराम से अलग किए गए स्तंभों की एक सूची निर्दिष्ट की जाती है।

उदाहरण के लिए यदि आप केवल विद्यार्थी का नाम, ईमेल आईडी देखना चाहते है तो निम्न क्वेरी का उपयोग करेंगे

SELECT STUNAME,EMAIL

FROM STUDENT;

STUDENT टेबल में सभी स्तंभों के लिए डेटा प्राप्त करने के लिए SELECT खंड में सभी स्तंभ नाम सूची प्रदर्शित कर सकते हैं। सिर्फ तारांकन चिह्न (*) का उपयोग यह संकेत देता है कि आप टेबल के सभी स्तंभों से डेटा प्राप्त करना चाहते हैं। क्वेरी का उपयोग इस तरह करें।

SELECT ROLLNO,STUNAME, FNAME,EMAIL,CITY

FROM STUDENT;

अथवा

SELECT *

FROM STUDENT;

यह STUDENT टेबल के सभी स्तंभ और पंक्तियाँ दर्शायेगा।

GROUP BY

GROUP BY खंड SELECT कथन का एक वैकल्पिक हिस्सा है। GROUP BY खंड पंक्तियों (Rows) के समूह को सारांश पंक्तियों के एक समूह में समूहीकृत करता है । प्रत्येक समूह के लिए एक पंक्ति GROUP BY खंड देता है। दूसरे शब्दों में, यह परिणाम सेट में पंक्तियों की संख्या कम कर देता है।

हम अक्सर aggregate फंक्शन जैसे SUM, AVG, MAX, MIN और COUNT के साथ GROUP BY खंड का उपयोग करते है । Aggregate फंक्शन का SELECT खंड में प्रयोग होता है और यह प्रत्येक समूह के बारे में जानकारी प्रदान करता है

GROUP BY खण्ड का सिंटैक्स है।

```
SELECT
        c1, c2,..., cn, aggregate_function(ci)
FROM
        table
WHERE
         where_conditions
GROUP BY c1 , c2,...,cn;
```

GROUP BY खंड FROM और WHERE खंड के बाद लिखना चाहिए । GROUP BY के बाद अल्पविराम से अलग किए गए स्तंभों एक सूची या expressions जिसे मापदंड के रूप में उपयोग करना चाहते हैं, लिखा जाता है। अगर हम पाठ्यक्रमों के अनुसार का समूह चाहते है तब निम्न कथन लिखा जायेगा।

अगर हम यह जानना चाहते है कि किस पाठ्यक्रम में कितने विधार्थी है हम निम्न क्वेरी के रूप में COURSE स्तंभ के साथ GROUP BY खंड का उपयोग करें ।

Aggregate फंक्शंस पंक्तियों के एक समूह की गणना कर एकल मान प्रदान करता है। GROUP BY खंड अक्सर किसी aggregate फंक्शन के साथ उपयोग किया जाता है जिससे गणना कर प्रत्येक उपसमूह के लिए एक एकल मान प्रदान करता है

उदाहरण के लिए,यदि कितने पाठ्यक्रम है यह मालूम करना चाहते हैं, तब हम COUNT फंक्शन GROUP BY खंड के साथ निम्नानुसार उपयोग कर सकते हैं:

```
SELECT COURSEID,COUNT(*)
FROM COURSE
GROUP BY COURSEID
```

अगर हम पाठ्यक्रमों के अनुसार विद्यार्थियों का समूह चाहते है तब निम्न कथन लिखा जायेगा

```
SELECT *
FROM STUDENT
GROUP BY COURSEID
```

अगर हम यह जानना चाहते है कि किस पाठ्यक्रम में कितने विधार्थी है हम निम्न क्वेरी के रूप में COURSE स्तंभ के साथ GROUP BY खंड का उपयोग करें:

```
SELECT COURSEID,COUNT(*)
FROM STUDENT
GROUP BY COURSEID
```

GROUP BY खंड द्वारा दिए गए समूह को फिल्टर करने के लिए हम HAVING खंड का उपयोग करते है । वर्ष 2003 के बाद में जन्मे विद्यार्थियों की संख्या वर्ष वार मालूम करने के लिए निम्न क्वेरी HAVING खंड को उपयोग में लेते हुए लिखी जाएगी ।

```
SELECT DOB,YEAR(DOB) AS YEAR, COUNT(*)
FROM STUDENT
GROUP BY YEAR
HAVING YEAR(DOB) > 2003;
```

HAVING खंड का उपयोग SELECT कथन में पंक्तियाँ के समूह या समुच्चय हेतु फिल्टर शर्तें निर्दिष्ट करने के लिए किया जाता है। फिल्टर शर्तें GROUP BY खंड में दर्शाये स्तम्भ के साथ ही कार्य करता है अगर GROUP BY खंड छोडा गया है तब HAVING खंड FROM खंड की तरह बर्ताव करता है।

HAVING खंड में पंक्तियों के प्रत्येक समूह के लिए फिल्टर शर्त लागू होता। जबकि WHERE खंड में फिल्टर शर्त हर अलग पंक्ति के लिए लागू होता है।

जब आप किसी टेबल से डेटा क्वेरी करने के लिए SELECT कथन का उपयोग करते हैं, परिणाम समुच्चय (Result set) क्रम (sort) में नहीं होता है।

परिणाम समुच्चय (Result set) को क्रमबद्ध (sort) करने के लिए ORDER BY खंड का उपयोग करना होता है । ORDER BY खंड से :

- परिणाम समुच्चय (Result set) को किसी एकल स्तंभ या एकाधिक स्तंभों द्वारा क्रमबद्ध (sort) कर सकते है ।
- परिणाम समुच्चय (Result set) को आरोही (ascnding) या अवरोही (descending) क्रम में विभिन्न स्तंभों द्वारा क्रमबद्ध (sort) कर सकते है ।

ORDER BY खंड का उपयोग निम्नानुसार करते है –

```
SELECT column1,cloumn2, .....
FROM table_name
ORDER BY column1 [ASC|DSC],column2 [ASC|DSC], .....
```

उदाहरणार्थ हमें विधार्थियो की सूची नाम के आरोही क्रम (alphabatical) में चाहिए , उस दशा में निर्देश निम्नानुसार होगा :

```
SELECT SNAME, ROLLNO
FROM STUDENT
ORDER BY SNAME;
```

उदाहरणार्थ हमें विधार्थियो की सूची नाम के अवरोही (descending) क्रम में चाहिए , उस दशा में निर्देश निम्नानुसार होगा :

```
SELECT SNAME, ROLLNO
FROM STUDENT
ORDER BY SNAME DESC;
```

उदाहरणार्थ हमें विधार्थियो की सूची प्राप्तांक के प्रतिशत के अवरोही (descending) क्रम अर्थात मेरिट लिस्ट चाहिए , उस दशा में निर्देश निम्नानुसार होगा :

```
SELECT ROLLNO, SNAME, TOTALMKS
FROM STUDENT
ORDER BY (TOTALMKS/3)
```

HAVING खंड में पंक्तियों के प्रत्येक समूह के लिए फिल्टर शर्त लागू होता। जबकि WHERE खंड में फिल्टर शर्त हर अलग पंक्ति के लिए लागू होता है.

जब आप किसी टेबल से डेटा क्वेरी करने के लिए का SELECT कथन का उपयोग करते हैं, परिणाम समुच्चय (Result set) क्रम (sort) में नहीं होता है।

परिणाम समुच्चय (Result set) को क्रमबद्ध (sort) करने के लिए ORDER BY खंड का उपयोग करना होता है । ORDER BY खंड से :

परिणाम समुच्चय (Result set) को किसी एकल स्तंभ या एकाधिक स्तंभों द्वारा क्रमबद्ध (sort) कर सकते है । परिणाम समुच्चय (Result set) को आरोही (ascending) या अवरोही(descending) क्रम में विभिन्न स्तंभों द्वारा क्रमबद्ध (sort) कर सकते है ।

ORDER BY खंड का उपयोग निम्नानुसार करते है –

```
SELECT column1,cloumn2, .....
FROM table_name
ORDER BY column1 [ASC|DSC], column2 [ASC|DSC]] .....
```

6.8.2 My SQL Functions

अधिकतर समुच्चय पर काम आने वाले फलन (function) का संक्षिप्त विवरण निम्न है :

AVG : मान (value) के एक समुच्चय का औसत मान परिकलित करता है।

उदाहरणार्थ विधार्थियो के प्रथम विषय के प्राप्तांको का औसत प्राप्तांक का मालूम करना है

```
SELECT AVG(SUB1) FROM STUDENT
```

प्रथम श्रेणी विधार्थियो के प्रथम विषय के प्राप्तांको का औसत प्राप्तांक का मालूम करना है

```
SELECT AVG(SUB1) FROM STUDENT WHERE DIV=”I”
```

निम्नानुसार DISTINCT ऑपरेटर जोड़कर अलग अलग प्राप्तांको का औसत परिकलित करने के लिए औसत फंक्शन का उपयोग कर सकते हैं ।

```
SELECT AVG(DISTINCT SUB1) FROM STUDENT
```

हम अक्सर औसत फंक्शन को GROUP BY खंड के साथ संयोजन के रूप में किसी तालिका में पंक्तियों के प्रत्येक समूह के लिए औसत मान परिकलित करने के लिए का उपयोग में लेते है । पाठ्यक्रमानुसार विधार्थियो के प्रथम विषय के प्राप्तांको का औसत प्राप्तांक का मालूम करना है ।

```
SELECT AVG(SUB1) FROM STUDENT GROUP BY COURSE
```

समूहों के लिए औसत मूल्यों के लिए शर्तें निर्धारित करने के लिए HAVING खंड में AVG फंक्शन का उपयोग कर सकते हैं, पाठ्यक्रमानुसार प्रथम श्रेणी विधार्थियो के प्रथम विषय के प्राप्तांको का औसत प्राप्तांक का मालूम करना है तब निम्न कथन लिखना होगा।

```
SELECT AVG(SUB1) FROM STUDENT GROUP BY COURSE
HAVING AVG(SUB1) >59;
```

COUNT : किसी तालिका में पंक्तियाँ की संख्या बताता है।

COUNT फंक्शन किसी तालिका में पंक्तियों की संख्या देता है। COUNT फंक्शन किसी तालिका में सभी पंक्तियाँ या किसी विशेष स्थिति से मेल खाने वाली पंक्तियों की गणना करने के कार्य आता है। COUNT फंक्शन का सिंटैक्स निम्नानुसार है।

COUNT फंक्शन BIGINT डेटा टाइप में मान प्रदान करता है . कोई मिलान खाने वाले पंक्ति न पायी जाने पर COUNT फंक्शन शून्य मान देता है।

COUNT फंक्शन कई प्रकार से कार्य करता है :

COUNT(*)

COUNT(*) फंक्शन SELECT द्वारा दिए गए एक परिणाम सेट में पंक्तियों की संख्या देता है।

COUNT(*) फंक्शन no–NULL और NULL मान वाली पंक्तियाँ भी गणना में शामिल करता है।

```
SELECT COUNT(*)
FROM STUDENTS
```

गणना करने के लिए कोई शर्त निर्दिष्ट करने के लिए WHERE खंड जोड़ सकते हैं :

```
SELECT COUNT(*)
FROM STUDENTS
WHERE DIV='FIRST';
```

टेबल में अद्वितीय (unique) पंक्तियों की गणना करने के लिए DISTINCT ऑपरेटर COUNT फंक्शन में जोड़ें:

```
COUNT (DISTINCT expression)
SELECT COUNT(DISTINCT COURSEID)
FROM STUDENT;
```

अक्सर COUNT फंक्शन GROUP BY खंड के संयोजन के रूप में विभिन्न समूहों में डेटा चिह्नित करने के लिए का उपयोग आता है ।

```
SELECT COURSEID, COUNT(*)
FROM STUDENT
GROUP BY COURSEID;
```

SUM : मान (value) के एक समुच्चय के योग का परिकलन करता है।

SUM फंक्शन मानों या अभिव्यक्ति के समुच्चय के योग की गणना करने के लिए काम आता है । SUM फंक्शन का सिंटैक्स निम्नानुसार है:

SELECT कथन में SUM फंक्शन का उपयोग में मिलान खाने वाली कोई पंक्तिया नहीं है तब वह शून्य मान प्रदान करेगा . DISTINCT ऑपरेटर समुच्चय में अलग अलग मानों की गणना कर जोड़ प्रदान करेगा ।SUM फंक्शन परिकलन में NULL मानों की उपेक्षा करता है।

```
SUM(DISTINCT expression)
```

उदाहरणार्थ यदि विद्यार्थियों द्वारा जमा किये गए शुल्क का जोड़ करना है :

```
SELECT SUM(FEE) TOTAL FEE
FROM STUDENT;
```

यदि पाठ्यक्रमानुसार विद्यार्थियों द्वारा जमा किये गए शुल्क का जोड़ करना है :

```
SELECT SUM(FEE) TOTAL FEE
FROM STUDENT
GROUP BY COURSEID;
```

SUM का उपयोग HAVING फंक्शन मे एक विशिष्ट शर्त के आधार पर परिणाम फिल्टर करने के लिए कर सकते है।SUM फंक्शन NULL मान को गणना में नजर अंदाज कर देता है।

MIN : मान (value) के एक समुच्चय में सबसे कम मान (value) को ढूढंता है।

MIN फंक्शन मानों के समुच्चय में से न्यूनतम मान देता है। MIN फंक्शन कुछ परिदृश्य में बहुत उपयोगी है जैसे सबसे कम प्राप्तांक निकलना हो , सबसे कम महंगा उत्पाद आदि . MIN फंक्शन का सिंटैक्स

निम्न है:

MIN (expression);

यदि यह ज्ञात करना है कि प्रथम विषय में सबसे कम प्राप्तांक क्या है ?

SELECT SNAME, MIN(SUB1)

FROM STUDENT;

MAX: मान (value) के एक समुच्चय में सबसे अधिक मान (value) को ढूंढ़ता है।

MAX फंक्शन मानों के समुच्चय में से अधिकतम मान देता है। MAX फंक्शन कुछ परिदृश्य में बहुत उपयोगी है जैसे सबसे अधिकतम प्राप्तांक निकलना हो , सबसे महंगा उत्पाद आदि- MAX फंक्शन का सिंटैक्स निम्न है:

MAX(expression);

यदि यह ज्ञात करना है कि प्रथम विषय में सबसे ज्यादा प्राप्तांक क्या है ?

SELECT SNAME, MAX(SUB1)

FROM STUDENT;

STRING function

CONCAT और CONCAT_WS फंक्शन का उपयोग करके एक साथ दो या अधिक स्ट्रिंग्स जोड़ने के लिए किया जाता है ।

CONCAT

CONCAT फंक्शन एक या अधिक स्ट्रिंग लेता है और उन्हें एक एकल स्ट्रिंग में संयोजित करता है। CONCAT फंक्शन एक स्ट्रिंग की एक न्यूनतम आवश्यकता है अन्यथा यह एक त्रुटि देगा । CONCAT फंक्शन का सिंटैक्स दिखाता है।

CONCAT(sting1,string2,string3,.....);

SELECT CONCAT('Suncity','Jodhpur');

SuncityJodhpur परिणाम दर्शायेगा

SELECT CONCAT('Suncity',';,'Jodhpur');

Suncity Jodhpur परिणाम दर्शायेगा

SELECT CONCAT('Suncity',', ;,'Jodhpur');

Suncity,Jodhpur परिणाम दर्शायेगा

CONCAT_WS फंक्शन दो या अधिक स्ट्रिंग मान को एक पूर्वनिर्धारित विभाजक के साथ संयोजित करता है।

CONCAT_WS(seperator,string1,string2,string3,).....

पहला तर्क विभाजक है : string1, string2,.....

CONCAT_WS फंक्शन स्ट्रिंग को विभाजक के साथ जोड़ कर नयी स्ट्रिंग प्रदान करता है।

SELECT CONCAT_WS(',','Pranav']'Mehta');

Pranav,Mehta परिणाम दर्शायेगा

SELECT CONCAT_WS('&','Pranav','Mehta');

Pranav–Mehta परिणाम दर्शायेगा

LENGTH : फंक्शन स्ट्रिंग्स की लंबाई अर्थात स्ट्रिंग में कितने अक्षर स्पेस आदि है उनकी संख्या प्राप्त करने के लिए काम आता है। यह बाइट में मापा जाता है।

CHAR_LENGTH : फंक्शन भी स्ट्रिंग्स की लंबाई अर्थात स्ट्रिंग में कितने अक्षर स्पेस आदि है उनकी संख्या प्राप्त करने के लिए काम आता है। यह करैक्टर में मापा जाता है।

SELECT LENGTH('prachi');

6 परिणाम दर्शायेगा

SELECT LENGTH('riya mehta');

10 परिणाम दर्शायेगा

LEFT फंक्शन निर्दिष्ट लंबाई के साथ एक स्ट्रिंग के बाएं भाग को लौटाने के लिए काम में आता है

LEFT फंक्शन एक स्ट्रिंग फंक्शन है जो एक निर्दिष्ट लंबाई के साथ एक स्ट्रिंग के बाएं भाग को देता है। LEFT फंक्शन का सिंटैक्स है –

LEFT फंक्शन दो तर्क स्वीकार करता है:

1- str स्ट्रिंग वह है जिसमे से हमें substring निकालनी है।

2- length वर्णों की संख्या निर्दिष्ट करता है जो एक सकारात्मक पूर्णांक लौटा दी जाएगी। LEFT फंक्शन से str स्ट्रिंग LENGTH में लिखी संख्या जितने सबसे बाएँ वर्ण देता है। Str या LENGTH का मान NULL है, तो यह एक NULL मान देता है।

अगर LENGTH शून्य या नकारात्मक है तब LEFT फंक्शन एक रिक्त स्ट्रिंग लौटाता है। यदि LENGTH स्ट्रिंग की लंबाई से अधिक है, तो बायाँ फंक्शन संपूर्ण स्ट्रिंग लौटाता है।

RIGHT

RIGHT फंक्शन निर्दिष्ट लंबाई के साथ एक स्ट्रिंग के दाएं भाग को लौटाने के लिए काम में आता है।

RIGHT फंक्शन एक स्ट्रिंग फंक्शन है जो एक निर्दिष्ट लंबाई के साथ एक स्ट्रिंग के दाएं भाग को देता है। RIGHT फंक्शन का सिंटैक्स है –

अगर LENGTH शून्य या नकारात्मक है तब RIGHT फंक्शन एक रिक्त स्ट्रिंग लौटाता है।

TRIM

ट्रिम TRIM फंक्शन उपयोगकर्ता के इनपुट से आम तौर पर कभी कभी अन्य अवांछित वर्ण होते हैं। इन अवांछित वर्णो को डेटा को अद्यतन करने से पहले सही स्वरूप में डेटा रखने की जरूरत होती है। अतः अवांछित अग्रणी और पीछे आने वाली वर्ण को निकालने के लिए TRIM फंक्शन काम आता है ।

SELECT TRIM(; Sample Text ;);

Result :Sample Text

DATE

DATE फंक्शन YYYY–MM–DD प्रारूप में दर्शाता है। DATE का मान string अथवा number में भी दिया जा सकता हैं। यह फंक्शन 1000–01–01 से 9999–12–31 के मध्य का मान स्वीकार करता है।

MONTH

MONTH फंक्शन DATE के मान में से माह का मान प्रदान करता है यह 1 से 12 के मध्य होता है।

SELECT MONTH('2016–02–03')

Result :2

YEAR

YEAR फंक्शन DATE के मान में से वर्ष का मान प्रदान करता है ।

SELECT YEAR('2016–02–03')

Result :2016

DAY

DAY फंक्शन DATE के मान में से माह के दिन का मान प्रदान करता है यह 1 से 31 के मध्य होता है। यह DAYOFMONTH() का पर्याय है।

SELECT DAY('2016–02–15')

Result :15

SYSDATE

SYSDATE() फंक्शन सिस्टम के वर्तमान DATE और टाइम के मान को YYYY–MM–DD HH:MM:SS अथवा YYYYMMDDHHMMSS के रूप स्ट्रिंग अथवा नंबर डेटा के अनुरूप प्रदान करता है ।

SELECT SYSDATE();

Result :2016 –01 –16 13:47 :36

DATEDIFF

DATEDIFF फंक्शन दो दिनांक, दिनांक समय,या टाईमस्टैम्प मान के बीच दिनों की संख्या परिकलित करता है।

महत्वपूर्ण बिंदु

1. डेटाबेस को सामान्य तौर पर डेटा के लिए एक भण्डार के रूप में वर्णित किया जा सकता।

2 डेटा तथ्य और आंकडों का एक संग्रह है जो की सूचना (information) प्रदान करने के लिए संसाधित (processed) किया जा सकता है।

3 डेटाबेस प्रबंधन सिस्टम (DBMS) एक सॉफ्टवेयर है जिनसे कि डेटाबेस भंडारण, पहुँच, सुरक्षा, बैकअप और अन्य सुविधाएं संचालित करने के कार्य होते हैं।

4 डेटाबेस मैनेजमेंट सिस्टम त्रि स्तरीय स्कीमा (schema architecture) को निरूपित (describe) करता हैं।

5 Database System को क्रियाओं अनुरूप दो भागों में विभाजित किया गया हैं:–स्टोरेज मैनेजर (Storage Manager) और क्वेरी प्रोसेसर (Query Processor)

6 डेटाबेस मैनजमेंट सिस्टम की सबसे बड़ी विशेषता से यह है कि इसमें डेटा की पुनरावर्ती (redundancy) को नियंत्रण (control) किया जा सकता है।

7 अच्छी तरह से संसाधित डेटा को सूचना/जानकारी (information) कहा जाता है।

8 किसी भी सिस्टम में डेटाबेस को बनाने (create) और प्रबंधित (maintain) करने के लिए डेटाबेस भाषाओं (database languages) का उपयोग किया जाता है।

9 डेटाबेस में दो तरह की भाषाओं का उपयोग किया जाता है। डेटा डेफिनेशन लैंग्वेज (Data Definition Language) और डेटा मैनीपुलेशन लैंग्वेज (Data Manipulation Language)

10 संरचित क्वेरी भाषा (SQL) डेटाबेस की भाषा है ।

11 MySQL एक डेटाबेस प्रबंधन प्रणाली है कि यह relational डेटाबेस को प्रबंधित करने के काम आता है। यह ओपन सोर्स सॉफ्टवेयर है।

अभ्यासार्थ प्रश्न

बहुचयनात्मक प्रश्न

1. कच्चे तथ्य और आंकड़े हैं:

(अ) डेटा (ब) जानकारी

(स) स्नैपशॉट (द) रिपोर्ट

2. डेटाबेस में वह सुविधा जिससे केवल कुछ रिकॉर्ड को एक्सेस करने के लिए अनुमति देता है:

(अ) फार्म (ब) रिपोर्ट

(स) क्वेरीज (द) तालिकाओं

3. संबंधपरक डेटाबेस क्या है?

(अ) एक जगह संबंधपरक जानकारी संग्रहीत करने के लिए।

(ब) एक डेटाबेस का अन्य डेटाबेसेस सम्बन्ध ।

(स) मानव संबंधों को संग्रहीत करने के लिए एक डेटाबेस।

(द) उपरोक्त में से कोई भी

4. इस कुंजी से प्रत्येक रिकॉर्ड को अद्वितीय रूप से पहचाना जाता है–

(अ) प्राथमिक कुंजी (ब) कुंजी रिकॉर्ड

(स) अद्वितीय कुंजी (द) फील्ड का नाम

5. संपूर्ण डेटाबेस संरचना और स्कीमा की परिभाषा के साथ संबंधित डेटाबेस भाषा है–

(अ) DCL (ब) DML

(स) DDL (द) उपरोक्त सभी

लघूत्तरात्मक प्रश्न

1. डेटाबेस से आप क्या समझते है ?
2. डेटाबेस आर्किटेक्चर को समझाये।
3. MySql के कितने प्रकार के डेटा प्रकार होते है ?
4. MySql में रिकार्ड्स कैसे डाले (insert) किये जाते है ?बताइए।
5- UPDATE फंक्शन की उपयोगिता बताये।

निबन्धातमक प्रश्न

1. डेटाबेस मैनेजमेंट सिस्टम को समझते हुए इसके लाभ बताये।
2. डेटाबेस मैनेजमेंट सिस्टम की विशेषतायें बताये।
3. डेटाबेस भाषाएँ कितने प्रकार की होती है ?विस्तार से बताइए।
4. Mysql में विद्यार्थियों का डेटाबेस कैसे बनायेंगे ?विस्तार से बताइए।
5. Mysql में स्ट्रिंग फंक्शन को समझाए।

उत्तरमाला

1. (अ) 2. (स) 3. (अ) 4. (अ) 5. (ब)

पाठ–7

Php

7.1 वेब डवलपमेन्ट एवं Php की प्रस्तावना

स्क्रिप्टिंग की आधारभूत संरचना

जहां एक और हम कोर प्रोग्रामिंग की बात करे जिसमे संकलन (Compiler) का उपयोग करते है जो कि एक रन–टाईम एनवायरमेंट होता है, जैसे कि किसी कार्य को स्वचालित चलाना हो जिसे मानव ऑपरेटर के द्वारा समय–समय पर आवश्यकता के अनुसार एक्जिक्यूट (Execute) किया जाता है।

स्क्रिप्टिंग एक प्रकार की प्रोग्रामिंग भाषा होती है, जिसमे ज्यादातर इंटरप्रेटर (Interpreter) को काम मे लिया जाता है। स्क्रिप्टिंग शब्द को ज्यादातर स्क्रिप्टिंग भाषा (Scripting Language) के नाम से जाना जाता है। इसे डायनैमिक उच्च स्तरीय सामान्य उद्देश्य (Dynamic High Level General Purpose Language) भाषा भी कहते है। जैसे कि Perl, TCL, Python, Php। इस प्रकार की स्क्रिप्ट (Script) मे छोटे प्रोग्राम (Programme) लिखे जाते है। जिसमे की एक या एक से अधिक पंक्तियों का कोड (code) हो सकता है। स्क्रिप्टिंग भाषा का विकास कुछ विशेष Environment के लिए किया जाता है। इसलिए इसे डोमेन विशेष अथवा सामान्य कार्य भाषा भी कहते है। जिसका कार्य टैक्स्ट प्रोसेसिंग (Text processing) करना होता है जैसे कि Sed, AKW मे होता है।

Scripting के मानक उदाहरणों में शामिल है:

1. BASH/Shell स्क्रिप्टिंग का उपयोग यूनिक्स/लाइनिक्स (Unix/Linux) ऑपरेटिंग सिस्टम के लिए होता है।
2. Php, Jsp, Asp स्क्रिप्टिंग वैब विकास (Scripting Web Development) के लिए।

Php का इतिहास एवं परिचय

Php को रासमॅस लैडार्फ (Rasmus Ledorf) नाम के प्रोग्रामर ने विकसित किया था, जो कि कनाड़ा मूल के रहने वाले थे। इस प्रोग्रामर ने सर्वप्रथम पर्ल स्क्रिप्ट (Perl स्क्रिप्ट) का एक समूह तैयार किया और उस स्क्रिप्ट के समूह को व्यक्तिगत होम पेज टूल (personal home page tool) अर्थात "Php Tool" नाम दिया गया। यह स्क्रिप्ट उन्होंने स्वंय के बायोडॉटा (resume) एवं वैब पेज को दर्शाने व उसे व्यवस्थित करने के लिए विकसित किया गया था। विकासकर्ता ने Php की सार्वजनिक घोषणा 8 जून 1995 को की एवं इस स्क्रिप्ट को CGI (Common Gateway Interface) के रुप मे C प्रोग्रामिंग भाषा मे लिखा ताकि इन स्क्रिप्ट के द्वारा वे HTML फार्मस् व डेटाबेस के साथ सूचना का आदान प्रदान करके

गतिशील पेज (Dynamic page) बना सके। इसीलिए Php को फार्म इंटरप्रेटर (form interpreter) का नाम दिया गया। तत्पश्चात् Php , Php/FI के रुप मे जानी जाने लगी। इस Release मे वे सभी फंक्शनैलटी (Functionality) उपलब्ध थी जो आज Php मे है, जैसे कि HTML फोर्म हैण्डलिंग पर्ल भाषा (form handling, Perl language) की तरह स्क्रिप्ट को उपयोग करना व प्रोग्रामिंग की तरह variable, function, classes declare करना, यह सभी गुणधर्म मौजूद है। फलस्वरुप इस स्क्रिप्टिग का नाम बदलकर Hyper text pre-processor (PHP) रखा गया।

Php के पाँचवें सस्करण (version) से पहले Php बिलकुल भी स्थाई (Stable) नही थी लेकिन पांचवे संस्करण के आने के बाद इन्टरनेट पर आज सबसे ज्यादा प्रचलित Php स्क्रिप्टिंग है। अभी इस समय सातवां संस्करण बाजार में उतारने की तैयारी चल रही है।

प्रोग्रामिंग भाषा के प्रकार (Type of Programming Languages)

कम्पयूटर मे प्रोग्रामिंग भाषा मूलतः दो प्रकार की होती है।

1. कम्पायलर आधारित भाषा (Complier Based languages)
2. इंटरप्रेटर आधारित भाषा (Interpreter Based Languages)

1. कम्प्यायलर आधारित भाषा . कम्पायलर आधारित भाषा के कम्प्यूटर प्रोग्राम जिस कम्प्यूटर आर्केटैक्चर व ऑपरेटिंग सिस्टम के लिए निर्माण किये गये है, उसी कम्प्यूटर आर्केटैक्चर के अनुसार सोर्स कोड को बाइनरी कोड में परिवर्तित कर देता है। अर्थात एक बार किसी प्रोग्राम को हम कम्पाईल कर देते है तो उसके बाद उस प्रोग्राम के सोर्स कोड की आवश्यकता नहीं रहती है। उस प्रोग्राम को बिना सोर्स कोड के हम चला सकते है। उदाहरण के तौर पर C, C++ एवं Java एक प्रकार की कम्पायलर आधारित भाषा है।

2. इंटरप्रेटर आधारित भाषा :- इस प्रकार की प्रोग्रामिंग भाषा को स्क्रिप्टिंग भाषा भी कहा जाता है। ऐसे मे Php भी एक स्क्रिप्टिंग भाषा है, क्योंकि Php के प्रोग्राम को रन करने के लिए इंटरप्रेटर की आवश्यकता होती है। यहां हम Php इंटरप्रटर के रूप में अपाचे वैब सर्वर (Apache web server) को उपयोग में लेंगे।

किसी Web-site को हम दो प्रकार के भागों मे बॉट सकते हैंः–

a) स्टैटिक (Static) या क्लाइंट साईड वैब साईट
b) डायनैमिक (Dynamic) या सर्वर साईड वैब साईट

a) स्टैटिक वैब साईट :– यह एक ऐसी प्रकार की वैब साईट होती है जिसकी सूचना केवल एक बार लिखी जाती है व बहुत कम बार इसे परिवर्तित किया जाता है। यह HTML, CSS, JS से निर्मित होती है। जिसमे कोड़ जब कभी परिवर्तित करना हो तो उसे किसी एडिटर के साथ ओपन करके परिवर्तन कर सकते है। इस अध्याय में हम एप्लीकेशन सोफ्टवेयर जैसे कि ड्रीम विवर या नोट पैड के साथ HTML पेज को उपयोग में लायेगे। उदाहरण के लिए HTML, CSS एवं Java Script से निर्मित वैब साईट केवल ब्राउजर पर डबल क्लिक करने से ओपन हो जाती है। क्योंकि हमारा ब्राउजर केवल एचटीएमएल भाषा समझता है।

b) डायनैमिक वैब साईट (Dynamic Web site):– इस प्रकार की वैब साईट को ज्यादातर वैब एप्लीकेशन (Web Application) के नाम से भी जाना जाता है। इसका कारण यह होता है कि इस प्रकार की वैब साइट डाटाबेस से जुड़ी रहती है एवं यह सर्वर पर एक्जिक्यूट होती है, तत्पश्चात इसका आउट–पुट ब्राउजर में एचटीएमएल के रूप में दिखाई देता है। उदाहरण के लिए किसी विद्यालय के छात्रों की सूचना संग्रहित करने के लिए एचटीएमएल के फार्म (Form) टैग का उपयोग करके हम उसे डेटाबेस में सुरक्षित कर सकते है। इस फार्म द्वारा लिया गया डेटा डेटाबेस की टेबल में सैल के रूप में सुरक्षित रहेगा। टेबल में सैल को पंक्ति (Row) एवं स्तम्भ (Column) के मिलान से बनने वाले सामान्य भाग के नाम से जानते है। यह डेटा भविष्य में सूचनाओं की समीक्षा करने के लिए सुरक्षित रखा जाता है।

हम Php को डेटाबेस माईएसक्यूएल (Mysql) से कनेक्ट (Connect) करके किसी सूचना को जो किसी भी कम्पनी / विद्यालय / संस्थान की हो सकती है उसे सीधे रूप से डेटाबेस में पहुंचाकर सुरक्षित कर सकते है।

किसी भी यूजर (User) को दिखाई देने वाली वैबसाईट को तीन भागों में विभाजित कर सकते है:–

(अ) वैब पेज का प्रकार

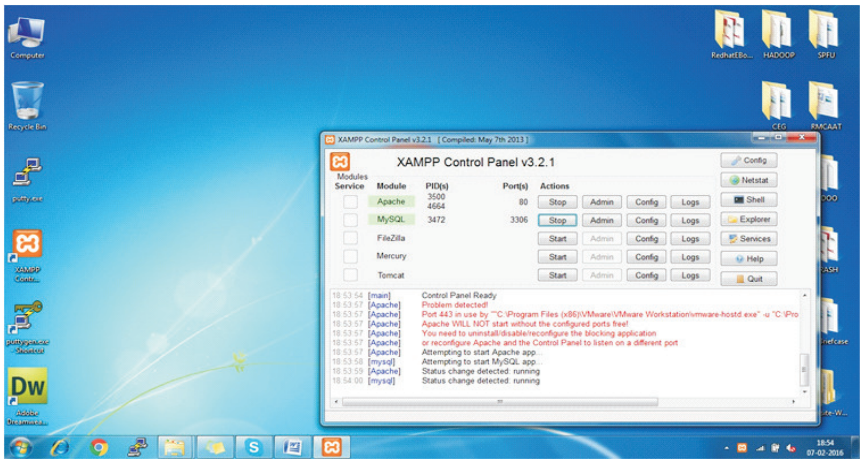
(ब) वैब पेज की स्टाईल (style)

(स) वैब पेज का व्यवहार

किसी भी वैब साईट की संरचना बनाने का कार्य HTML का होता है। वह वैब साईट किस प्रकार की दिखाई देगी यह CSS (Cascading Style Sheet) बताता है। जबकि वैब पेज को यूजर फ्रण्डली व डेटा को मान्य करने के लिए जावा स्क्रिप्ट का उपयोग किया जाता है।

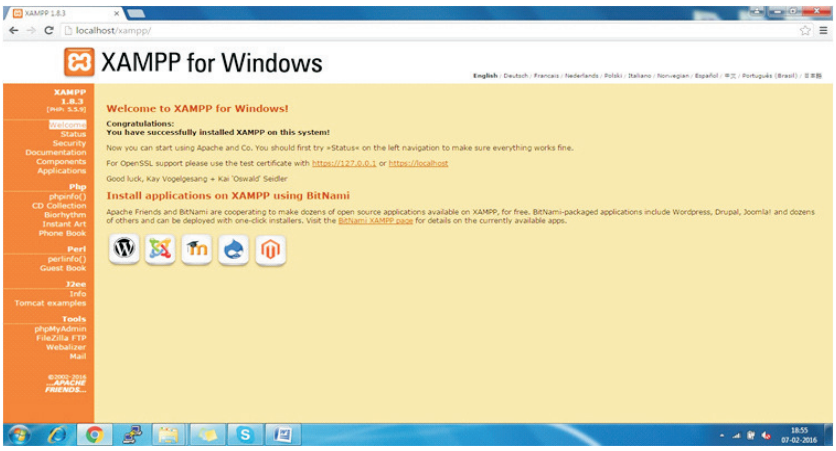
7.1.4 वैब डवलपमेन्ट (Web Development) मे Php की आवश्यकता

Php के बारे में ऐसी अवधारणा है कि Php का प्रयोग केवल वैब पेज को डायनैमिक बनाने के लिए ही किया जाता है ऐसा बिल्कुल नहीं है। हम Php का प्रयोग कई अन्य जरुरतो को पूरा करने मे भी कर सकते है जैसे हम Php को GTK (Genom Tool Kit) के साथ काम लेकर प्लेटफार्म स्वतन्त्र डस्कटॉप एप्लीकेशन भी डवलप कर सकते हैं जो किसी भी प्लेटफार्म पर बिना किसी निर्भरता के एक्जीक्यूट हो सकती है। हम Php को कमाण्ड प्राम्ट जो कि विण्डोज के रन बटन पर क्लिक करने पर सर्च किया जा सकता है, यहां Php को रन करके कई प्रकार के सिस्टम एडमिन के कार्यो को सम्पन्न किया जा सकता है। Php एक प्लेटफार्म स्वतंत्र स्क्रप्टिंग है, जिसे किसी भी ऑपरेटिंग सिस्टम के साथ उपयोग में लाकर रन या एक्जिक्यूट कर सकते है। Php कई प्रकार के वैब सर्वर के अन्तर्गत कार्य करती है। जैसे Apache, IIS व अन्य प्रकार के वैब सर्वर इंटरनेट पर फ्री उपलब्ध होते है। हम यहां XAMPP सोफ्टवेयर जो कि एक ग्रुप सोफ्टवेयर है जिसमें अपाचे, पीएचपी एवं माईएसक्यूअल निहित होते है। नीचे दिये गये चित्र में अपाचे वैब सर्वर को शुरू व बंद करने के प्रोसेस दिखाये गये है।



चित्र 7.1 XAMPP सॉफ्टवेयर का कन्ट्रोल पैनल

उपरोक्त चित्र में Xampp कन्ट्रोल पैनल में अपाचे व माईएसक्यूएल को स्टार्ट व स्टॉप बटन के साथ दिखाया गया है जिसके द्वारा सर्विसेज को हम अपने अनुसार बंद व चालू कर सकते है। तत्पश्चात ब्राउजर ऑपन करके यूआरएल में निम्ननिलिखित टाईप करना होगा :– http://localhost उसके बाद एन्टर Key दबाते ही Xampp की वैलकम स्क्रीन दिखाई देगी। यहां http:// एक प्रोटोकॉल है जिसके द्वारा हम वैब साईट को पोर्ट नम्बर 80 पर कार्य में लाते है।

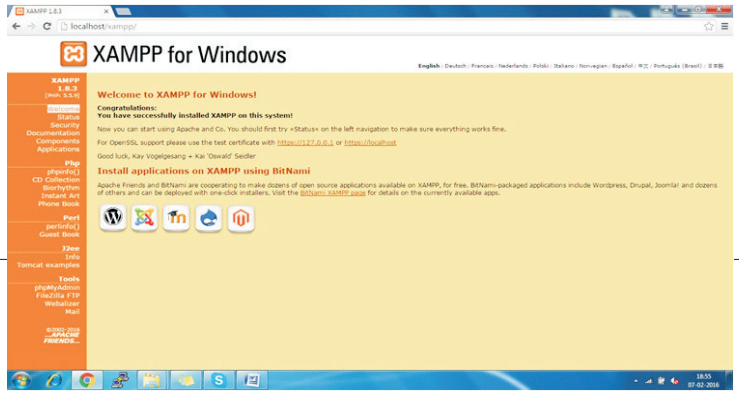


चित्र 7.2 XAMPP सॉफ्टवेयर की स्वागत स्क्रीन

7.1.5 वैब रूट डायरेक्ट्री व Php पेज को लिखना

XAMPP सॉफ्टवेयर में वैब रूट डायरेक्ट्री htdocs नामक एक डायरेक्ट्री होती है जो कि सभी वैब साइट की लोकेशन का समूह होता है। यहां हम उदाहरण के तौर पर सभी पीएचपी पेज को वैब रूट

डायरेक्ट्री में एक सब सह डायरेक्ट्री बनाकर उसमें सेव करते है। जैसे कि यहां हम myceg एक प्रकार की वैब साईट के होस्टिंग की वैब लोकेशन है जिसका पाथ c:\xampp\htdocs\ है। हम Php पेज को किसी एडीटर जैसे– Dream viewer/Notepad/Eclipse मे लिखने के बाद उसे कम्प्यूटर की लोकेशन में निम्न प्रकार से सेव करते है– c:\xampp\htdocs\myceg. यहां myceg एक वैब प्रोजेक्ट है जिसे एक्सेस करने के लिए हमें ब्राउजर में जाकर URL में http://localhost/myceg टाईप करने पर निम्नलिखित आउटपुट दिखाई देगा।



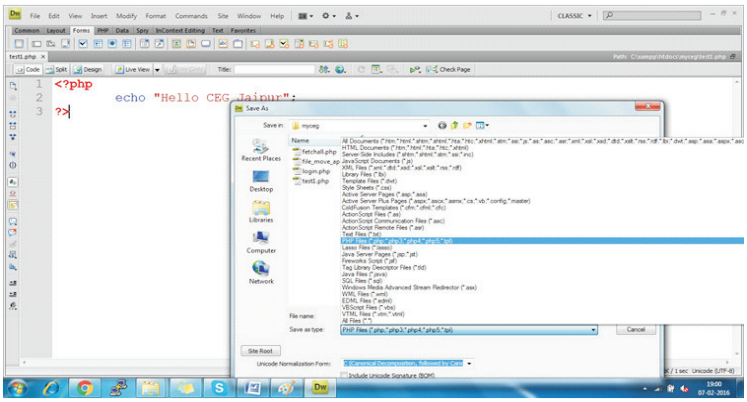
चित्र 7.3 XAMPP सॉफ्टवेयर में वैब प्रोजेक्ट का एक्सेस

Php में टैग्स व स्क्रिप्ट को लिखना

Php स्क्रिप्ट में सभी स्टेटमेंट व कोड को हम <?php//——your code here——?> के मध्य लिखते है, क्योंकि Php का इन्टरप्रेटर (Interpreter) इन्ही टैग्स के बीच के समाहित स्टेटमेंट के कोड को पढ़ता है व उसे सही प्रकार से एक्जीक्यूट करता है। इसलिए हम जितने भी Php प्रोग्राम बनाते है उन सभी प्रोग्रामों के Php कोड को इन्हीं टैग्स के बीच लिखा जाता है। भले ही हमारा कोड एक लाइन का ही क्यों ना हो। जैसा कि दिए गए चित्र 7.4 में Php प्रोग्राम का उदाहरण दिया है। चित्र 7.5 में स्क्रिप्ट को कम्प्यूटर में सुरक्षित करना बताया गया है।

चित्र 7.4 पीएचपी स्क्रिप्ट को ई–लिखावट

अब ऊपर दिए गए कोड को हम सेव करते हैं। पीएचपी प्रोग्राम का एक्सटेंशन (Extension) *.php होना चाहिए। यहां * का अभिप्राय पेज के नाम से है व .php फाईल के एक्टेशन से है।



चित्र 7.5 पीएचपी स्क्रिप्ट को सेव करना

7.1.6 पीएचपी स्क्रिप्ट में चर (Variable) की शुरूआत करना

Php का उपयोग करते समय जब हम कम्प्यूटर में ऐसे मानों को सुरक्षित करना चाहते है जिसका मान स्क्रिप्ट के कोड के अनुसार समय समय पर बदलता रहता है। इसके लिए हम आईडैन्टीफायर का उपयोग करते है उस आईडैन्टीफायर को वैरियेबल आईडैन्टीफायर या केवल चर कहा जाता है।

कम्प्यूटर में विभिन्न प्रकार के मानों के साथ विभिन्न प्रकार की प्रक्रिया करवा के हम अलग–अलग प्रकार के परिणामों को उत्पन्न करवा सकते हैं। इससे पहले हम किसी भी कम्प्यूटर के मान पर प्रोसेसिंग को लागू करके किसी तरह की आवश्यकता के अनुसार परिणामों को प्रदर्शित करें, तब उन मानों को कम्प्यूटर की मैमोरी में सुरक्षित करना अनिवार्य होता है। कम्प्यूटर की मैमोरी में तब तक किसी मान को सुरक्षित नहीं कर सकते जब तक उन्हें कुछ समय होल्ड करने के लिए इन्हें निर्देश ना दी गई हो कि वे मैमोरी में किस प्रकार व कहां जाकर सुरक्षित होंगी। ज्यादातर ऐसा देखा गया है कि कम्प्यूटर यह सुविधा नहीं प्रदान करता कि हम किसी निर्धारित मैमोरी की लोकेशन पर डेटा को लिख (Write) सके व कम्प्यूटर हमें बता देवे कि कसी वैल्यू को कहां सुरक्षित किया गया है। इसके लिए हम आईडेन्टीफायर (Identifier) का उपयोग चर (Variable) को लिखने के लिए करते है।

Php में आईडैन्टीफायर (Identifier):- आईडैन्टीफायर किसी मैमौरी की लोकेशन का एक प्रकार का नाम होता है जिन्हे हम अपनी आवश्यकता के अनुसार स्वंय लिख सकते है और फिर Php उन आईडैन्टीफायर के नामों को जुडी हुई मैमोरी की लोकेशन में सुरक्षित करती है।

जैसेः– $age;

$age=12;

जहॉ 12 एक प्रकार की Value है व $age एक Identifier है।

Example:- <?php

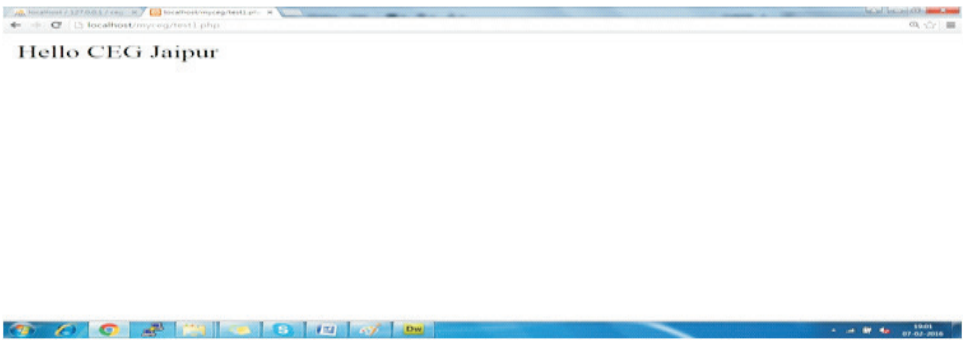
```
$age=123;
echo "Type of variable is".get type($age);
echo "<br>";
echo "End of the code";
?>
```

//output- Type of variable is Integer

उपरोक्त code मे हम देख सकते है कि $age मे एक इंटीजर 123 को Initialize किया गया है। फलस्वरुप $age एक प्रकार का Integer प्रकार का चर (variable) बन गया है जिसे हम Output के रुप मे देखते है।

जब हमे Php में चर को लिखना होता है, तो हम ($) sign का प्रयोग करते है। $ sign के तुरन्त बाद हमे वह नाम लिखना होता है जिसे हम Php द्वारा हमारे मानों को सुरक्षित करने के लिए काम में ली जाने वाली मैमोरी की लोकेशन के साथ एसोसिएट करते है। इसका कारण यह है कि यह वही नाम होता है जिसके द्वारा हम Php के चर के मान को सुरक्षित करने के लिए पूर्व में निर्धारित की गई मैमोरी लोकेशन व उसकी पहुंच को अपने अनुसार कस्टमाईज कर सकते है।

यहां Php के एक उदाहरण में जो कि पिछले पेज मे बनाना बताया गया है। जब हम इसे एक्जीक्यूट करते है तो आउटपुट के तौर पर ऐसा परिणाम दिखाई देता है। अगर हमें अपने वैब प्रोजेक्ट को एक्जीक्यूट करना हो तो ब्राउजर में URL पर कुछ ऐसा टाईप करेगे– http://localhost/myceg फिर एनटर दबाएं तो कुछ ऐसा परिणाम आयेगा।



चित्र 7.6 पीएचपी स्क्रिप्ट को वैब प्रोजेक्ट के अन्तर्गत एक्सेस करना

Php मे Output Statement:- बिना आउटपुट स्टेटमेंट को समझे हम आगे ही नही बढ सकते क्योंकि किसी भी कम्प्यूटर प्रोग्राम के 3 अत्यन्त आवश्यक भाग होते है। जैसे कि–

(अ) INPUT

(ब) OUTPUT

(स) PROCESS.

जैसेः– C, C++ में print( ) एवं cout<< फंक्शन (Function) का उपयोग Output लेने मे किया जाता है। इसी प्रकार से Php मे echo ( ) व print( ) फंक्शन या स्टेटमेंट का उपयोग किया जाता है।

echo statement:- यह एक प्रकार का स्टेटमैन्ट (Statement) है, व आउटपुट लेने के लिए ज्यादातर पीएचपी में इसका उपयोग किया जाता है। ये फंक्शन की तरह भी काम में लाया जा सकता है। इसीलिए जरुरी नही है कि echo के साथ Parenthesis का प्रयोग करे या ना करे।

Syntax:- void echo (String $arg, String $arg)

print( ) Statement:- ये एक प्रकार का स्टेटमेन्ट है ना कि फंक्शन, क्योंकि यह स्टेटमेंट आरग्यूमेंट के रूप में किसी भी डेटाटाईप या मिक्स डेटा टाईप के मानों को स्वीकार करता है एवं इन मानों द्वारा आने वाले आउटपुट को वैब ब्राउजर या टर्मिनल पर आउटपुट के रूप में प्रदर्शित करता हैं

Syntax:- int print (string $argument);

7.2 Php मे कमेन्ट (Comment) करने के तरीके

कमेन्ट किसी भी प्रोग्रामिंग भाषा का बहुत ही महत्वपूर्ण भाग होता हैं जिसमें प्रोग्रामर अपने प्रोग्राम को अलग अलग प्रकार के लोजिक (Logic/Program) को लिखने की दिनॉक एवं इस प्रोग्राम से क्या आउटपुट आयेगा व किसी भी कोड की लाईन को एक्जीक्यूट होने से रोकना इन सभी के लिए हम कमेन्ट स्टाईल का उपयोग करते है। इनके उपयोग से ब्राउजर में सिर्फ बिना कमेन्ट वाली लाईन्स का आउटपुट दिखाई देता है। जबकि कोड के भाग में हमें सभी प्रकार के कमेन्ट हुये कोड प्रदर्शित होते है। इसका तात्पर्य यह हुआ कि कमेन्ट वाला कोड कभी कम्पायलर अथवा इन्टरप्रेटर से रन नहीं किया जा सकता है।

Php मे निम्न प्रकार के कमेन्ट उपयोग में लाये जाते हैः–

1. **"C" भाषा की तरह कमेन्ट करने का तरीका :-** प्रोग्रामिंग करते समय कई बार ऐसा होता है कि हम बिना कोड को डिलिट किये टैस्टिंग के दौरान कुछ लाईन्स का आटपुट देखना चाहते है तो हम "C" भाषा की तरह कमेन्ट का तरीका उपयोग में लाते है।

```
<?Php
/*
        $a = 1234;
        $b = "Hello";        यहां output के तौर पर कुछ भी नही आयेगा क्योंकि
        $c = 55.5;    */    Multiple line को comment किया गया है।
?>
```

2. **"C++" भाषा की तरह कमेन्ट करने का तरीका :-** हम इस प्रकार के कमेन्ट को को // Forward Slash द्वारा प्रदर्शित करते है, जैसेः–

```
<?php
        for ($i=o; $i<=5;$i++)
        {echo $i;
        //echo "Hi";    यहां Single line comment किया गया है जो कि
                        Browser पर output मे नही दिखेगा।
          }
?>
```

3. Unix में कमेन्ट करने का तरीका :- इस प्रकार के कमेन्ट के लिए # का उपयोग करते है। जैसेः–

```
<?php
        for ($i=o; $i<=5;$i++)
        {echo $i;
        # echo "Hi";      यहां Hi print नही होगा क्योंकि # द्वारा comment
                         किया गया है।
        }
?>
```

शाब्दिक (Literal) :- किसी भी तरह के डेटा टाईप के मानों को दर्शाने वाले अक्षरों के संरचनात्मक इकाई को शाब्दिक (Literal) कहा जाता है। Php मे निम्न प्रकार के Litteral काम मे ला सकते है।

1. 12345 //Interger
2. 0XFE// Hexa
3. OX( )// Octal
4. "Hello ceg" // string
5. True, null // Boolean

7.3 Php में डेटा टाईप्स

Php में मुख्यतः 3 प्रकार के डेटा टाईप्स होते हैं।

1. Scaler डेटा टाईप
2. Compound डेटा टाईप
3. Special डेटा टाईप

1. Scaler डेटा टाईप :- इन्हें Scaler/Single डेटा टाईप के नाम से भी जाना जाता है। इन्हें एकल डेटा टाईप कहने का तात्पर्य इनमें सुरक्षित होने वाली एक प्रकार का मान होता है। इनके

द्वारा निम्न डेटा टाईप को प्रदर्शित किया जाता है।

a) Integer
b) Float
c) String
d) Boolean

a) **Integer** डेटा टाईप:- किसी भी दशमलव वाली धनात्मक/ऋणात्मक संख्यात्मक मान को Integer कहा जाता है। Php में हम Integes के मान को Binary, Octal, Decimal अथवा Hexa के रूप में प्रदर्शित करते है जैसे–

Binary	Decimal	Octal	Hexa Decimal
00110	55	07	OXA
11001	-6	-08	-OX7
11001	+150	+045	+OXFF

b) **Float/Double** डेटा टाईप:- किसी दशमलव वाली धनात्मक/ऋणात्मक वाली Value को फ्लोटिंग प्रकार की संख्या कहा जाता है। Floating point numbers को Float, Double और Real numbers भी कहा जाता है। Php मे Float numbers को दशमलव वाली संख्या के रुप में या उसकी घातांक के रुप में भी प्रदर्शित कर सकते है।

जैसेः– (i) 145.6 (ii) 1.3e2 (iii) 8E-10

c) **String**: डेटा टाईप- किसी भी प्रोग्रामिंग का आधार String होता है। इस लिए String में बदलाव को Php में स्वीकार किया गया है। Php मे केवल 256 आधारभूत ASCII संख्या को String रुप में समर्थन करता है। यानि String को हम Unicode के रुप मे उपयोग नही कर सकते है। String शाब्दिक को हम 4 अलग तरीके से विभाजित कर सकते है–

(i) Single Quates
(ii) Double Quates
(iii) Heredoc
(iv) Nowdoc

d) **Boolean** डेटा टाईप:- Php में सत्य व असत्य ऐसे दो मान है, जो Boolean डाटा टाईप को प्रदर्शित करती है जिनके अन्तर्गत मान एक अथवा शून्य हो सकता है।

2. Compound डेटा टाईपः- Php में यह दुसरा प्रकार के डेटा टाईप का ग्रुप होता है। जिसमें Array व Objects को सम्मिलित किया गया है। वास्तव में यौगिक (Compound) डाटा टाईप एक प्रकार का बेसिक डेटा टाईप का ही रुप होता हैः

a) **Array** डेटा टाईप:- Array एक प्रकार के डाटा टाईप के समूह को प्रदर्शित करता है।

b) **Object** डेटा टाईप:- जब हम Php को OOPS (Object oriented Programmig) के तरीके से प्रोग्राम को विकसित करने के लिए उपयोग में लेते है, तब Php हमे आब्जैक्ट (Object) निर्माण करने की सुविधा देता है। आब्जैक्ट एक ऐसी इकाई होती है, जो क्लास में एक संयुक्त रूप से उपयोग में आती है। उदाहरण के लिए Car एक प्रकार की क्लास हो सकती है व उसके मॉडल उसके आब्जेक्ट अर्थात ऑब्जेट क्लास की एक रन टाईम entity होता है या किसी class का स्वरूप (Blue print) होता है।

3. **Special डेटा टाईप:-** इस प्रकार के डेटा टाईप मे रिसोर्स टाईप व Null Data type को सम्मिलित किया गया है।

a) **Resource** डेटा टाईप:- यह सामान्यतया एक Integer Number होता है, जो कि बाह्य रिसोर्स से कनेक्शन स्थापित करता है। जैसेः–

जब हम Mysql Database मे Connection स्थापित करके डेटा को प्राप्त करते है तब Php के लिए Mysql डाटा बेस की टेबल का डेटा एक प्रकार के Resource की तरह काम करता है। इस प्रकार के Integer Number को Resource/Handle कहा जाता है।

b) **Null** डेटा टाईप:- जब हम किसी रिसोर्स से Php का लिंक स्थापित करना चाहते है अथवा किसी Identifier मे सुरक्षित मान को डिलीट करना चाहते है तो हम उस Resource मे Null Value स्थापित कर देते है।

अचर को Php मे लिखना

जब हमें Program मे किसी मैमोरी की लोकेशन पर एक ऐसे मान को सुरक्षित करना हो जो कि पूरे प्रोग्राम के दौरान कभी भी बदल नहीं पाये तो ऐसे मानों को हम स्टोर करने के लिए अचर (Constant Identifier define) उपयोग करते है। अचर को लिखने के लिए हम define( ) फंक्शन का उपयोग निम्न प्रकार से करना होता है–

boolean define (String Name, [Mixtype Value]); जैसा कि उपरोक्त Property मे किसी Identifier को Constant बनाना हो तो हम define ( ) Function use कर रहे है।

Php में फंक्शन का उपयोग :- फंक्शन का कार्य किसी भी प्रोग्रामिंग भाषा में किसी कार्य को जब बार–बार दोहराना हो व बिना किसी विघ्न के जिससे प्रोग्राम के दूसरे भाग पर कोई असर ना पड़े इसके लिए हम फंक्शन का उपयोग करते है।

Function को php मे लिखने के लिए function keyword का उपयोग फंक्शन के नाम से पहले करते है।

जब हम Php मे हमारी जरुरत के अनुसार किसी function को आरम्भ करते है तब हम उस फंक्शन का नाम भी उसी समय लिख देते है ताकि समय आने पर उसे कॉल किया जा सके। जैसे –

Multiplex नाम का कोई Function अगर बनाना है तो उसे Multiplex ( ) के नाम से भी पव्रदर्शित किया जा सकता है यहां () Paranthesis फंक्शन के भाग को दर्शा रहा है।

उदाहरणः–

```
<?php
    function Multiplex ($x,$y)  // function की body
    {     $z= ($x*$y)
          echo $z ;
    }
?>
// Calling     Function in a Script
    <?php          Multiplex (5,4);     // calling of function
?>
```

Syntax:-

```
<?php       function name (Argument)>
            {
                  // body of function
            }
?>
```

Function की calling कुछ ऐसे भी की जाती है।

```
<?php  function name (Argument);  // Function Callng?>
```

7.4 Php में लूपए ऐरे एवं कंट्रोल स्टेटमेंट (Control Statements)

Types of Control Statement:- Php में विभिन्न प्रकार के Control/Conditional Statements को हम 4 भागों में बॉट सकते है।

1. क्रमबद्ध स्टेटमेंट (Sequential Statements)
2. सशर्त स्टेटमेंट (Conditional Statements)
3. पुनरावर्ती स्टेटमेंट (Loop Statements)
4. जम्पिंग स्टेटमेंट (Jumping Statements)

1. Sequential Statements:- जिन स्टेटमेंट को एक्जीक्यूट होने के बाद अगली पंक्ति में क्रमबद्ध

तरीके से डेटा को प्रिन्ट करने के लिए प्रोग्राम का कोड लिखा जाता है वह क्रमबद्ध स्टेटमेंट कहलाते है।

2. Conditional Statements:- कम्प्यूटर प्रोसेसिंग में प्रोग्राम की किसी स्थिति के आधार पर नियंत्रण अपने सामान्य फ्लो को छोडकर किसी अन्य बिन्दु के स्टेटमेंट को अगर एक्जीक्यूट करवाया जावे तो इस प्रकार के चुने हुए एक्जीक्यूसन को सर्शत स्टेटमेंट कहलाते है। जैसे कि– (if——else)

3. Interative Statements:- जब प्रोसेसिंग में कुछ स्टेटमेंट को कंडिसन पर निर्भर रहते हुए बार बार पुनर्रावत्ती करने की आवश्यकता होती है तो उसे लू स्टेटमेंट या पुनरावर्ती स्टेटमेंट कहा जाता है। जैसे– loops.

4. Jumping Statements:- इस प्रकार के स्टेटमेंट हमें प्रोग्राम में किसी एक बिन्दु से दूसरे एक्जीक्यूट बिन्दु पर जम्प करने की सुविधा उपलब्ध कराता है इसलिए इन्हें जम्पिंग स्टेटमेंट भी कहा जाता है। जैसे break, continue

A. if Statement:- सभी स्टेटमेंन्टस में से if-statement सबसे ज्यादा महत्वपूर्ण होता हैं जिसके द्वारा हम स्टेटमेंट के एक्जीक्यूशन के फ्लो को नियन्त्रण में कर सकते है। यह एक द्विमार्गी स्टेटमेंट है जिसमें कण्डीशन के सत्य या असत्य होने के आधार पर निर्भर करते हुए Program का नियंत्रण दो अलग अलग बिन्दुओं पर पहुंचाया जा सकता है।यानी If- statement के अनुसार Program के पास दो प्रकार रास्ते होते है। एक Condition सत्य (True) होने पर व दूसरी असत्य (False) होने पर Execute होती है।

```
if (Expression of control )
Statement 1;
```

इस Syntax के कोष्ठक मे Specified Expression/condition दिखायी जाता है। Condition के सही होने पर सत्य भाग एक्जीक्यूट हो जाता हैं।

```
if (Expression or Condition)
{
        Statement 1;
        Statement2;
}
```

जैसेः– इस प्रकार के statement को समझने के लिए हम एक example ले सकते है कि, दो संख्याओं मे से बडी संख्या ज्ञात करने के लिए Php का Program बनाते है।

```
<?php
        $a=40;
        $b=50;
        if ($x>$y)
```

```
        {
        echo "$a is greater than $b";
        }
        if ($a<$b)
        {
        echo "$b is greater than $a";
        }
    ?>
//Output:-    50 is greater than 40
```

इस प्रकार के If Statement के कोष्ठक में सत्य (True) आता है। इसलिए output के रुप मे प्रोग्राम का कन्ट्रोल if statement के Block में प्रवेश करता है और वहॉ के स्टेटमेंट को Interpret करता है जो हमे Output के रुप में Screen पर दिखाई देता है।

B. If-else Statement:- इस प्रकार की स्टेटमेंट का प्रयोग हम जब करते है जब कोई ऐसी समस्या हो जिसमे केवल दो ही संभावित परिणाम हो सकते है या तो सत्य या असत्य। जैसेः–

```
if (Expression or Condition)
{
    Statements;
    ———
    ———
    ———
} else {
statements;
———
———
}
```

```
Example:-   <?php
                $a=5,  $b=10;
                if($a>$b) {
                    echo "$a is greater";
                    } else {
                    echo "$b is greater";
                    }
```

?>

C. Netsted ——if—else—— Statement:- जब एक If condition के स्टेटमेंट ब्लॉक के अन्दर एक और स्थिति या if-else condition के statement block का प्रयोग किया जाता है, तो इस प्रकार की If condition को Nested——if—else कहते है।

Syntax:-

```
if (Expression or condition 1)
{
if (Expression or condition 2)
{
        Statements;
        ———
        ———
        ———
} else {
        Statements;
        ———
        ———
        ———
}
}else{
        Statemets;
}
```

D. If—else—Ladder— Statement:- जब कम्प्यूटर प्रोग्रामिंग में ऐसी समस्या आती हो जिसे हल करने के लिए बहुत सी कंडीसन में से किसी एक को चुना जाये इस प्रकार की स्थिति को हम क्रमवार तरीके से परखते है और जहाँ भी स्थिति सत्य होती है प्रोग्राम का फ्लो उस कोड के ब्लॉक को एक्जीक्यूट कर देता है।

जैसेः—

```
if (Expression or condition 1)
{
        Statements 1;
        ———
        ———
        ———
}
else if (Expression or condition 2)
```

```
{
        Statements 2;
        ———
        ———
}
else if (Expression or condition 3)
{
        Statements 3;
} else {
& taken;
}
```

उपरोक्त उदाहरणों व उनके लिखने के तरीके (Syntax) से समान output ही रहते है लेकिन else if statement का परिणाम Special else——if statement तुलना में काफी तेज होती है। इसीलिए हमेशा प्रोग्रामिंग करते समय else if का ही प्रयोग करना चाहिए न कि else———if का।

E. Uncondition Statement:

switch-case स्टेटमेंट **:-** इस प्रकार की Statements को चुने हुये स्टेटमेंट भी कहा जाता है और Php का यह केवल इकलौता Unconditional statement है यानि switch-case स्टेटमेंट मे हम किसी तरह की condition तब तक नहीं लिख सकते जबतक कि कोई कम्प्यूटर प्रोग्राम का Expression या Variable नहीं लिखा जाता।

Syntax:-

```
switch (Variable)
{
        case value 1:
        statement block 1;
break;
        case value 2:
        statement block 2;
break;
        case value n:
        statements;
break;
default:
        statements;
```

```
        break;
    }
```

स्विच केस की कार्य प्रणाली – इसकी working मे value, valued, valuen एक ऐसे प्रकार के मान होते है, जिन्हें case Labels भी कहा जाता है। इसके बाद Colon (:) लगाना जरुरी होता है। साथ ही इनके मानों को लिखना भी जरूरी होता है। यदि break ना लिखा जाये तो स्टेटमेंट कोड ब्लॉक के एक्जीक्यूशन के बाद का कंट्रोल जारी रहेगा। इसीलिए प्रत्येक स्टेटमेंट के बाद break लगाना जरुरी होता है।

Example:-

```
<?php
        $per=55.8%;
        $a = (int) $per;
        switch ($a)
{
        case ($a=<85 & $a>=60):
        echo "Ist div.";
        break;
        case ($a>= 45 & $a<= 59):
        echo"second div.";
        break;
        case($a>=36 & $a<=44):
        echo"Third div.";
        break;
default:
        echo "fail";
        break;
}
```

इस उदाहरण में हमने यह देखा कि किसी छात्र की प्रतिशत को हल किया जाये तो उसकी डिविजन उसे कैसे प्रदान की जावेगी यह हमें स्वीच केस से पता लगेगा। इसके लिए हमने 3 प्रकार की सीमा निर्धारित की है। जैसेः–

60—85— Ist Division

45—59—IInd Division

36—44—IIIrd Division

Below 36% Student is fail.

Interations (Loops):- यह एक तीसरे प्रकार का कंट्रोल स्टेटमेंट है, जब प्रोग्राम मे हमे किसी प्रकिया

को बार–बार दोहराना (Repeat) होता है तब हम Looping control statements का उपयोग करते है किसी भी Loop में हमेशा 3 भाग मुख्यतः होने चाहिए–

1. **Initialization**:- इसे Loop का आरम्भिक हिस्सा होता है। जो यह तय करता है कि Loop की शुरुआत कहाँ से होगी इसके लिए (=) Assignment operator का उपयोग किया जाता है।
2. **Condition**:- यह किसी Loop का conditional part होता है जो कि यह तय करता है कि Loop कब तक चलेगा।
3. **Size Increment/Decrement**:- इस part मे Loop किस क्रम मे आगे बढेगा या घटेगा यह हमको Increment/Decrement ऑपरेटर बताता है। जैसेः– $i++, --$i;

   इस उदाहरण में $i Variable में 1 को जोडकर आगे बढाया गया है, जबकि --$i में $i की Value मे से 1 कम होता रहेगा जब तक Loop जारी रहेगा।

   a. for Loop:- यह सर्वाधिक प्रयोग में आने वाला loop है। इस loop मे for Keyword एक पूर्व में समाहित कम्प्यूटर के Reserve word के रूप मे उपयोग में आता है। इस loop के कोष्ठक मे तीनों parts (;) से पृथक करके लिखे जाते है। जब Certainity (निश्चितता) होती है, तब हम for का उपयोग करते है।

   Syntax**:-**
   ```
   for (Initialization; condition; step size part)
   {
       statements;
   }
   ```

   उदाहरणः– हम एक Example लेते है जिससे हम 1 से 20 तक Print करवाने है, break line के साथ।

   Ex.-
   ```
   <?php
   for ($i=1; $i<=20; $i++)
   {
   echo $i. "\n";
   }
   ?>
   ```
   //Output:
   ```
   1
   2
   3
   ```

While Loop:- for Loop की तरह यह भी एक प्रकार का स्टेटमेंट होता जो कि पुनरावृति का कार्य करता है, लेकिन फिर भी अन्य प्रकार के लूप से काफी भिन्न होता है।

Syntax:-

```
while (expr):
    statement
    ...
endwhile;
```

While के कोष्ठक के बाद कभी भी (;) semicolon use नही होता है। जब while के बाद केवल एक ही statement का execution करना होता है तब मझले कोष्ठक का प्रयोग करने की आवश्यकता नही होती है। इसका उपयोग तब किया जाता है जब हमे यह ज्ञात नही है कि database में कितने records है। इसका उपयोग तब किया जाता है, जब हमें यह ज्ञात नहीं हो कि डेटाबेस में कितने रिकार्डस है। इसका अभिप्राय यह हुआ कि While loop अनिश्चितता के लिए एक्जीक्यूट होता है, जिसमे यह पता नही होता कि डाटा कब तक मिलता रहेगा।

Example:- अगर हम Data structure की बात करें जहां Node (Data part) कहां मिलेगा इसकी जानकारी while लूप की कंडीशन कुछ इस प्रकार एक्जीक्यूट होगी।

```
<?php
/* example 1 */

$i = 1;
while ($i <= 10) {
    echo $i++;  /* the printed value would be
                   $i before the increment
                   (post-increment) */
}

/* example 2 */

$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
```

```
endwhile;
?>
```

do While Loop:- यह Php मे उपयोग में आने वाला तीसरे प्रकार का loop है। यह भी अन्य loops की तरह ही तीनो आधारभूत स्टेटमेंट जैसे कि - Initialize, Condition, Increment/Decrement की जरुरत नही होती है। इसकी विशेषता यह है कि check होने वाली condition loop के अंत मे लिखी जाती है।

उदाहरण:-

```
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

Array -

यह एक प्रकार ऐसा डेटा टाइप है जो कि मूल रूप से परफोर्मेंस क्षमता की दृष्टि से अन्य डेटा टाईप से ज्यादा महत्वपूर्ण होता है। ऐसा इसलिए होता है क्योंकि इसमें एक जैसे डेटा टाईप्स का समूह निहित होता है। इसलिए इसे Array नाम से जाना जाता है।

Php में ऐरे लिखने का तरीका

array **array** ([mixed $ . . .])

Array सबसे साधारण डाटा स्ट्रैक्चर होता है। कम्प्यूटर मे हम किसी भी डेटा को जब तक व्यवस्थित तरीके से सुरक्षित नहीं करते तब तक उस डेटा को सही प्रकार से एक्सेस या मैनेज नहीं किया जा सकता।

उदाहरण के तौर पर किसी विद्यालय के विघार्थियों के Roll numbers/Unique Numbers को computer की memory मे सुरक्षित करना चाहते है ताकि उनकी जानकारी तैयार की जा सके। इसके लिए हम सभी प्रकार के मानों को सुरक्षित डेटा टाईप Array मे लिख सकते है। अगर हमें 500 छात्रों का नाम या उनके Roll number को store करने के लिए हमें कम्प्यूटर प्रोग्राम में 500 Variable लेने होंगे जबकि Array मे Program का code लिखना बेहद आसान होगा, जिसमें केवल एक वरियेबल व एक लाइन के कोर्ड से कार्य सम्पन्न हो जावेगा।

Php में भी C, C++, Java व अन्य Programming की तरह बहुत सारे Data को एक साथ Group में Computer की memory में store करना होता हो, तो Array का उपयोग किया जाता है। Php में हम दो प्रकार से Array को बना सकते है–

1- Square Bracket Pair [ ]

2- Array Constructor

1. **Bracket Pair**:- यहां Array $ar[ ] से create किया जाता है व उसकी Value को { }मझले Bracket के द्वारा दिखाया जाता है।

उदाहरणः– $ar [ ] = {"ceg", "jaipur"};

2. **Array Constructor**:- यहां Array को create करने के लिए array ( ) नामक function का उपयोग किया जाता है।

उदाहरणः– $ar = array("maninder", "singh");

दोनो प्रकार की Array variable को print करने के लिए हम print_r( ) function का use करते है। जिसमे सभी मानों का आउटपुट मानव द्वारा पढ़े जा सकने वाले फोरमेट में आता है।

Php मे भी Array C, C++, Java की तरह शून्य से आरम्भ होने वाले सिस्टम (Zero based indexing system) का अनुसरण करता है। जिसमें सुरक्षित होने वाली प्रथम वैल्यू हमेशा Array के Zero (0) इंडक्स पर सुरक्षित होती है। इसीलिए उपरोक्त उदाहरण में हमें [0] दिखाई दे रहा है, जो पहले तत्व की Position को दर्शा रहा है।

जिस तरह हम Bracket system का उपयोग करके विभिन्न प्रकार की Array variable लिख सकते है उसी तरह से हम Array constructs का प्रयोग करके भी हम विभिन्न प्रकार के Array create कर सकते है। जैसेः–

```
<?php
    $array = array(1, 1, 1, 1,  1, 8 => 1,  4 => 1, 19, 3 => 13);
    print_r($array);
?>
```

इस प्रकार की ऊपर उदाहरण में दिखाई देने वाली statement एक Array का निर्माण कर रही है। और उस Array मे हमने तीन string प्रकार के मानों को सुरक्षित किया है।

$_GET Global Array:- ये भी एक प्रकार का Associative Array होता है, जो जो वर्तमान में HTML की स्क्रिप्ट में फोरम के GET मैथड का प्रयोग करते हुए वैब ब्राउजर के एड्रेस में पास किये जाने वाले पैरा मीटर्स होते है।

$_POST Global Array:- ये भी एक प्रकार का Associative Array होता है, जो कि HTML की स्क्रिप्ट में फोरम के पोस्ट मैथड का प्रयोग करते हुए वैब ब्राउजर के एड्रेस में पास किये जाने वाले पैरा मीटर्स को ऐक्सस करने के लिए बना होता है। $_POST से URL में डेटा प्रदर्शित नही होता, मतलब hide रहता है जबकि $_GET में Data URL में show होता है।

7.5 Operators in Php

किसी भी प्रोग्रामिंग भाषा में विभिन्न प्रकार के परिणामों को प्राप्त करने के लिए भिन्न भिन्न प्रकार के गणीतीय/अंकगणीतीय तथा तार्किक गणनाएं करने पड़ते है। इन सभी गणनाओं को हल करने के लिए कुछ विशेष प्रकार के चिन्ह उपयोग में लाये जाते है, वे विशेष चिन्ह प्रोग्रामिंग की अलग अलग प्रकार की समस्याओं को हल करने के लिए निर्देश देते है। इस प्रकार के चिन्हों को ऑपरेटर के नाम से जानते है। साथ ही डेटा को पहचान देने के लिए जिन आईडैन्टीफायर के साथ यह प्रक्रिया अपनाई जाती है उन्हें इन ऑपरेटर्स का ऑपरेण्ड (Operand) कहा जाता है। ज्यादातर पीएचपी में ऑपरेटर दायें से बायें प्रदर्शित हाते है, इस प्रक्रिया को सहचारिता के नाम से जानते है। जबकि किसी एक्सप्रेशन (Expression) दिखाये गये ऑपरेटर के आधार पर कौनसा ऑपरेशन पहले व बाद में घटित होगा इस कार्य कार्य को हम ऑपरेशन की प्रसिडेन्स (Precedence) के नाम से जानते है। इसे समझने के लिए निम्नलिखित टेबल में ऑपरेटर्स की सहचारिता दर्शायी गयी है।

Operator Precedence

Associativity	Operators	Additional Information
non-associative	new	new
left	[	array()
non-associative	++ --	increment/decrement
non-associative	~ - (int) (float) (string) (array) (object) @	types
non-associative	instanceof	types
right	!	logical
left	* / %	arithmetic
left	+ - .	arithmetic and string
left	<< >>	bitwise
non-associative	< <= > >=	comparison
non-associative	== != === !==	comparison
left	&	bitwise and references
left	^	bitwise
left	|	bitwise
left	&&	logical
left	||	logical
left	? :	ternary
right	= += -= *= /= .= %= &= |= ^= <<= >>=	assignment
left	and	logical
left	xor	logical
left	or	logical
left	, (comma)	many uses

1) अंकगणितीय **(Arithmatic) Operators** :- विभिन्न अंकगणीतीय गणनाओं को प्रदर्शित करने के लिये हम निम्न पांच ऑपरेटर का उपयोग करते है।

*, %, +, -, /

Arithmetic Operators

Example	Name	Result
-$a	Negation	Opposite of *$a*.

$a + $b	Addition	Sum of *$a* and *$b*.
$a - $b	Subtraction	Difference of *$a* and *$b*.
$a * $b	Multiplication	Product of *$a* and *$b*.
$a / $b	Division	Quotient of *$a* and *$b*.
$a % $b	Modulus	Remainder of *$a* divided by *$b*.
$a ** $b	Exponentiation	Result of raising *$a* to the *$b*'th power. Introduced in PHP 5.6.

2) **स्ट्रींग (String) Operators**:- Php मे मूलतः स्ट्रींग के लिए दो आपरेटर होते है। जो कि आपस में कड़ी का काम करते है। पहला operator वास्तव में मूल operator है जिसे single dot (.) द्वारा प्रदर्शित करते है। दूसरे प्रकार का बिन्दू ऑपरेटन दशमलव वाली संख्याओं को दर्शाता है जैसे कि 15. 6 इस ऑपरेटर के दायें या बायें में जो भी स्पेश दिया जाता है वह महत्वपूर्ण भूमिका अदा करता है जैसे कि –

```
<?php
        $a = "Hello ";
        $b = $a . "World!"; // now $b contains "Hello World!"

        $a = "Hello ";
        $a .= "World!";      // now $a contains "Hello World!"
?>
```

उपर दिखाये गये उदाहारण में " " में String को दिखाया गया है जबकि Backward slash (\) को dot operator से concate किया गया है।

3 **इंक्रीमेंट एण्ड डिक्रीमेंट (Increment & Decrement) operatar**:- ये ऑपरेटर भी अन्य प्रोग्रामिंग की तरह होते है। इंक्रीमेंट ऑपरेटर को द्वी धनात्मक चिन्ह (++) द्वारा प्रदर्शित किया जाता है साथ ही डिक्रीमेंट ऑपरेटर को ऋणात्मक चिन्ह (--) द्वारा प्रदर्शित किया जाता है। इन ऑपरेटर्स की यह विशेषता है कि इसे एक ऑपरेण्ड के साथ ही काम में लाते है। जो कि निम्न अनुसार है–

Increment/decrement Operators

Example	Name	Effect
++$a	Pre-increment	Increments *$a* by one, then returns *$a*.
$a++	Post-increment	Returns *$a*, then increments *$a* by one.
--$a	Pre-decrement	Decrements *$a* by one, then returns *$a*.
$a--	Post-decrement	Returns *$a*, then decrements *$a* by one.

```
<?php
        echo "<h3>Postincrement</h3>";
        $a = 5;
        echo "Should be 5: " . $a++ . "<br />\n";
        echo "Should be 6: " . $a . "<br />\n";
        echo "<h3>Preincrement</h3>";
        $a = 5;
        echo "Should be 6: " . ++$a . "<br />\n";
        echo "Should be 6: " . $a . "<br />\n";
        echo "<h3>Postdecrement</h3>";
          $a = 5;
          echo "Should be 5: " . $a— . "<br />\n";
          echo "Should be 4: " . $a . "<br />\n";
          echo "<h3>Predecrement</h3>";
          $a = 5;
          echo "Should be 4: " . —$a . "<br />\n";
          echo "Should be 4: " . $a . "<br />\n";
?>
```

4) **Equal operatar** :- इस प्रकार के ऑपरेटर किन्ही दो मानों को आपस में तुलना के लिए काम में लेते है।

Comparison Operators		
Example		**Name Result**
$a == $b	Equal	**TRUE** if *$a* is equal to *$b* after type juggling.
$a === $b	Identical	**TRUE** if *$a* is equal to *$b*, and they are of the same type.
$a != $b	Not equal	**TRUE** if *$a* is not equal to *$b* after type juggling.
$a <> $b	Not equal	**TRUE** if *$a* is not equal to *$b* after type juggling.
$a !== $b	Not identical	**TRUE** if *$a* is not equal to *$b*, or they are not of the same type.
$a < $b	Less than	**TRUE** if *$a* is strictly less than *$b*.
$a > $b	Greater than	**TRUE** if *$a* is strictly greater than *$b*.
$a <= $b	Less than or equal to	**TRUE** if *$a* is less than or equal to *$b*.
$a >= $b	Greater than or equal to	**TRUE** if *$a* is greater than or equal to *$b*

.

5) **Relational operatar :-** यह ऑपरेटर मूलतः असत्य मान वापस (return) करता है।

Php में मुख्यः रूप से निम्न Relational Operators होते है–

Operator	Name	Example	Result
= =	Equal	$x == $y	TRUE if $x is exactly equal to $y
= = =	Identical	$x === $y	TRUE if $x is exactly equal to $y, and they are of the same type.
!=	Not equal	$x != $y	TRUE if $x is exactly not equal to $y.
<>	Not equal	$x <> $y	TRUE if $x is exactly not equal to $y.
!==	Not identical	$x !== $y	TRUE if $x is not equal to $y, or they are not of the same type.
<	Less than	$x < $y	TRUE if $x (left-hand argument) is strictly less than $y (right-hand argument).
>	Greater than	$x > $y	TRUE if $x (left hand argument) is strictly greater than $y (right hand argument).
<=	Less than or equal to	$x <= $y	TRUE if $x (left hand argument) is less than or equal to $y (right hand argument).
>=	Greater than or equal to	$x >= $y	TRUE if $x is greater than or equal to $y.

6) **Logical operatar :-** यह ऑपरेटर तार्किक गणनाओं के मानों को कि वे आपस में बराबर है या नहीं को परखने का कार्य करते है। यें भी दो operand के साथ काम करते है, और True या false मानों को वापस (return) करते है। जो कि निम्न चार प्रकार के होते है–

PHP Logical Operators		
Operator	**Name**	**Example**
and	And	$x and $y
or	Or	$x or $y
xor	Xor	$x xor $y
&&	And	$x && $y

7) **Bitwise operatar:-** पूर्व में हमने अभि तक जितने भी ऑपरेटर्स का अध्ययन किया है वे मुख्य रूपसे किसी चिन्ह की बाईट मान पर कार्य करते है। जबकि बिटवाईज ऑपरेटर ऐसे ऑपरेटर होते है जो किसी चिन्ह की मैमोरी लोकेशन पर सुरक्षित की गई बाईनरी डिजिट यानि बिट पर काम करते है। ये निम्नलिखित है–

		Bitwise Operators
Example	**Name**	**Result**
$a & $b	AND	Bits that are set in both *$a* and *$b* are set.

$a | $b	OR (inclusive or)	Bits that are set in either *$a* or *$b* are set.
$a ^ $b	XOR (exclusive or)	Bits that are set in *$a* or *$b* but not both are set.
~ $a	NOT	Bits that are set in *$a* are not set, and vice versa.
$a << $b	Shift left	Shift the bits of *$a $b* steps to the left (each step means "multiply by two")
$a >> $b	Shift right	Shift the bits of *$a $b* steps to the right (each step means "divide by two")

8) **Assignment operatar** :- असाइनमेंट ऑपरेटर (Assignment Operator) को Php में equal (=) चिन्ह द्वारा प्रदर्शित किया जाता है। किसी कम्प्यूटर प्रोग्राम की अभिव्यक्ति (expression) में बांई ओर होने वाली गणना को यह ऑपरेटर दांयी और ऑपरेण्ड में सुरक्षित कर देता है।

उदाहरण :– $a= $b+$c;

यहां इस उदाहरण में $b व $c की value को जोड कर $a में save किया गया है।

Example:

```
<?php
$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to 4.
?>
```

7.6 Mysql Database

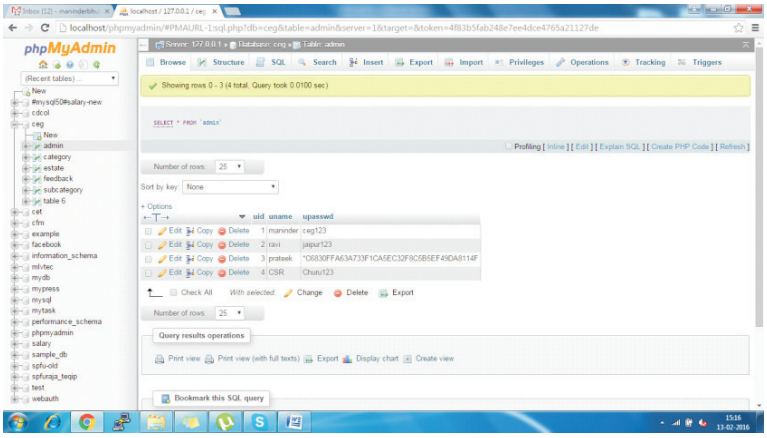
पूर्व अध्याय में हम माई एसक्यूएल डाटा बेस को विस्तृत में अध्ययन कर चुके है व उसे कन्सोल पर कैसे एक्जीक्यूट करते है यह हमने देखा। यहां हम माई एसक्यूएल का ग्राफिकल इंटरफेस देखेंगे व यह जानेगे कि कमाण्ड लाईन के साथ साथ ग्राफिकली कैसे डेटाबेस व उसकी टेबल को बनाया व एक्सेस किया जाता है। एक बार फिर से हम डाटाबेस को परिभाषित करेंगे–

माई एसक्यूएल एक प्रकार का Relational database management system (RDBMS) है। इसमें सभी Information Table रुप में पंक्ति व स्थम्भ के कॉमन हिस्से यानि सैल में सुरक्षित रहती है।

यानि टेबल की सभी फील्ड को स्थम्भ व स्थम्भ को फील्ड के नाम से भी जाना जाता है , लेकिन फिर भी इस बात का ध्यान रखना जरूरी रहता है कि स्थम्भ किसी टेबल के सभी अभिलेख द्वारा समान रूप से शेयर किये जाने वाले गुणों को दर्शाता है, जबकि फिल्ड किसी एक अभिलेख के डेटा को प्रदर्शित करता हैं।

Database and Tables की संरचना ग्राफीकली (Graphical User Interface) निम्नानुसार

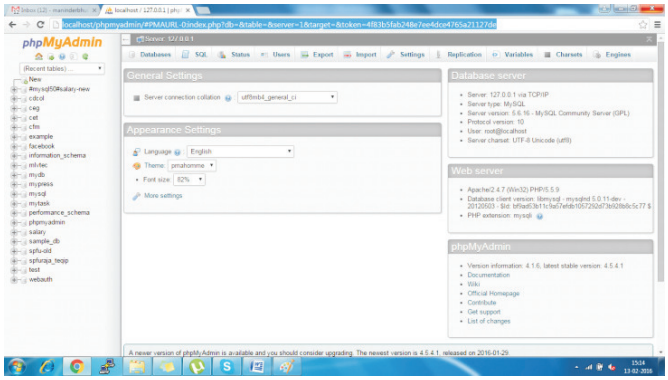
है–



चित्र 7.7 माई एसक्यूएल डाटा बेस व टेबल की संरचना

डाटाबेस में टैबल एक विशेष प्रकार की सूचना का संग्रह होता है। उदाहरण के तौर पर किसी विद्यालय के छात्रों की सूचना अगर हम डाटाबेस की टेबल में सुरक्षित करना चाहें तो बड़ी आसानी से यह सम्भव हो जाता है। इसका अर्थ यह हुआ कि भिन्न भिन्न प्रकार की वैब एप्लीकेशन जो कि डाटाबेस से जुड़ी रहती है जिसमें कि एचटीएमएल के पेज से डेटाबेस में सीधे तौर पर डाटा को भेज कर सुरक्षित कर सकते है।

Mysql with Php my admin:- Phpmyadmin एक प्रकार का फ्री व ऑपन सोर्स सॉफ्टवेयर है। जिसे Xampp server के साथ ही इंस्टाल किया जाता है। (याद रहे कि हम पीएचपी,माई एसक्यूएल व अपाच्य वैब सरर्वर को अलग से इंस्टॉल ना करके बण्डल सॉफ्टवेयर जिसे हम XAMPP सर्ववर के नाम से जानते है को इंस्टाल करते है।) XAMPP द्वारा हम Mysql को GUI(Graphical User Interface) रूप मे access कर सकते है। जिसके द्वारा Database व Table का निर्माण बहुत ही आसानी से किया जा सकता है। जैसे कि नीचे दिये गये चित्र में दर्शाया गया है–



चित्र 7.8 माई एसक्यूएल डाटा बेस को ब्राउजर पर एक्सेस करना

GUI Mysql को एक्सेस करने के लिए हम ब्राउजर के एड्रेसबार में निम्नलिखित स्ट्रिंग को लिखते है–

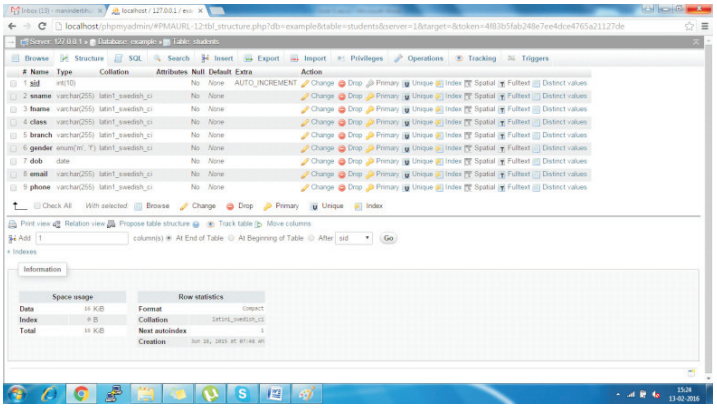
http://1ocalhost/phpmyadmin

पूर्व में दी गई Image में Database का निर्माण करना बताया गया है।

Table बनाने व उसके Constrains (criteria) या प्रतिबन्ध कैसे लगाये जाते है। यह नीचे चित्र में दर्शाया गया है। हमें Constraint के रुप में SQL (Structure Query Language) के अलग–अलग प्रकार लेने होते है। जिसमे Column का डेटा सार्वभौमिक (Unique) बनाया जा सके ताकि कोई एक ही प्रकार का डेटा दुबारा से इंसर्ट ना हो सके। इसके लिए डेटाबेस में हम निम्न प्रतिबन्ध उपयोग में लेते है–

1. Primary key
2. Unique Key
3. Foreign Key
4. Null/Not null
5. Check

अगले चित्र संख्या 7.9 में विद्यालय के छात्रों का डेटाबेस व उनकी टेबल के फील्डस को बनाने का तरीका दर्शाया जा रहा है जिसमें कि विभन्न प्रकार के डाटा टाईप काम में लाये जा रहे है ताकि सभी प्रकार की सूचना जैसे छात्र का नाम उसकी उम्र माता–पिता का नाम व अन्य सूचनाऐं एक साथ सुरिक्षत की जा सकें। जैसेः–



चित्र 7.9 डाटाबेस की टेबल व उसके फील्ड प्रतिबंध के साथ में

ऊपर दिए गए चित्र संख्या 7.9 में सभी प्रकार की सुविधा (Features) को जैसे कि टेबल की फील्डस व उसके प्रकार को सम्मिलित किया गया है। जिसमे Constraints व Data type दोनों को काम मे लाने का तरीका बताया गया है।

पीएचपी का माई एसक्यूएल से कनेक्शन कोड़ (Connectivity Php with Mysql) - जब हम माईएसक्यूएल डेटाबेस को Php के साथ कनेक्ट करते है तो डेटाबेस के डेटा की सूचना व सुरक्षा को भी बहुत ध्यान से रखना होता है। जब भी हम Php के पेज को mysql से connect करते है तो हमेशा user-name व Password को सुरक्षित रखना चाहिए ताकि कोई data को चुरा न सके। Php को mysql से connect करने व database की Table को access करने के लिए हम दो प्रकार के फंक्शन्स का उपयोग करते है–

```
<?php
        mysql_connect("localhost","root","") or die(mysql_error());
        mysql_select_db("ceg") or die(mysql_error());
?>
```

उदाहरण के तौर पर अगर डेटाबेस का नाम है व उसके छात्रों की टेबल जिसका नाम स्टूडेंट है, को php page से connect करना है तो हमे Web root Directory (C:\xampp\htdocs\myceg) मे एक Php की file बनानी पडती है, जिसमे connection का code लिखा जाता है। एवं इस connectivity के code को दूसरे Php page मे दूबारा से कॉल (call) किया जा सकता है। call करने के लिए हमे किसी भी Php मे page के प्रथम लाईन पर Php tags मे (<?php————?>) include ( ) या require ( ) function का उपयोग करके उस connectivity की Php file को Call कर सकते है। जिसके द्वारा Database से सभी php pages का Mysql से connection establish (स्थापित) हो जाता है।

अब इस स्क्रिप्ट को c:\xampp\htdocs\ceg directory में सुरक्षित करना है व इस फाईल का नाम connect.php रखना है व जिस किसी भी page मे इसे लोड़ करना है, तो include( )function मे एक्सेस कर सकते है। जैसेः–

<?php include("connect.php"); ?>

माई एसक्यूएल में डाटा टाईप्स (Mysql Data types):- पूर्व अध्याय में हम कुछ डेटा टाईप्स का अध्ययन पहले ही कर चुके है। यहां हम कुछ बेसिक डेटा टाईप्स को सारांश में प्रस्तुत कर रहे है–

1. Date & Time data type
2. Date data type
3. Time data type
4. Timestamp
5. Bool, Boolean, tinyint (1) Data type
6. varchar

7. char
8. enum
9. int
10. float
11. double
12. text

डाटा बेस के गुणधर्म (Database Attributes):- यह डाटाबेस का एक ऐसा गुणधर्म है जिसमें कई प्रकार के इनबील्ट Attributes होते है जिनमे से Auto increment गुणधर्म प्रमुख है। जैसे– कोई student_id एक प्रकार की टेबल की फील्ड है जिसे हम Primary key प्रतिबन्ध (constraint) बना रहे है ताकि सूचनाऐं बार बार इंसर्ट ना हो पाये साथ ही साथ हमे Table मे यह पता चले कि कौनसा Record कौनसी Row की लोकेशन पर सुरक्षित हुआ है ताकि रिकार्ड को गिनना (count) आसान हो जाये। इसीलिए Table का पहला field हमेशा Integer type व auto increment बनाते है। जिससे इस प्रकार की field मे संख्या अपने आप बढते रहते है व हमे एक बार Table का Data देखने पर यह पता चल जाता है कि कितने रिकार्ड किस लाकेशन पर सुरक्षित हुए है।

इंडैक्स गुणधर्म (Index Attribute):- जिस प्रकार किसी किताब में हजारों शब्द होते है उन शब्दों में से किसी एक शब्द को ढूंढने के लिए हम शब्दावली का उपयोग करके उस विशेष ढूंढे जाने वाले शब्द को आसानी से प्राप्त करते है उसी प्रकार डाटाबेस में मिलियन रिकार्ड में से विशेष प्रकार के डाटा को ढूंढकर लाने का कार्य इंडेक्स गुणधर्म का होता है। कई डेटाबेस एडमिन इसे जानबूझ कर टेबल पर लगाते है जबकि Primary या Unique constraint (प्रतिबंध) table पर या Column पर लगाने से इंडेक्स गुणधर्म अपने आप लागू हो जाते है। स्मरण रहे कि डाटाबेस की टेबल जब बनती है तो वह सदैव unique व Not NULL type प्रतिबंध की होती है।

महत्वपूर्ण बिन्दु

¨ XAMPP software को सर्वप्रथम Install करना चाहिए ताकि हम php व Mysql को Apachee Webserver के साथ काम ले सके।

¨ हमेशा Project web root directory के under create किये जाने चाहिए। (जैसेः– C:\xampp\htdocs\ceg यहां ceg web root directory है)।

¨ Php की Script के code को किसी भी Interface development environment (IDE) में लिखा जा सकता है।

¨ Php tags हमेशा <?php——//your code here——?> के बीच लेने चाहिए।

- echo statement का उपयोग हम output लेने मे कर सकते है।
- Php को Rasmus Lerdorf नामक Programmer ने विकसित किया था।
- Php मे Single Line Comments को // से व्यक्त किया जाता है।
- Php मे 4 प्रकार के Control Statements होते है। क्रमशः Sequential, Conditional, Interactive and Jumping.
- Operator & Orperand Php मे Expression व्यक्त करने के तरीके है।
- MySql एक प्रकार RDBMS (Relational Database Management System) होता है।
- Php की MySql से Connectivity करके हम Front end व Back end बीच मे विभिन्न Operation जैसे– DQL, DML, DDL Operation Perform करते है।
- Apache web server का default port number 80 व 443 होता है। इसी प्रकार mysql का reserve port number 3306 होता है।

अभ्यासार्थ प्रश्न

अतिलघूत्तरात्मक प्रश्न

1- Php को किस Programmer ने विकसित किया था।

(अ) रदरफोर्ड (ब) बिल गेट्स

(स) रेसमस लोडर्फ (द) जेम्स रॉबर्ट

2- Php मे Variable को कैसे Initialize किया जाता है।

(अ) # द्वारा (ब) @ द्वारा

(स) $ द्वारा (द) * द्वारा

3- Php व MySql को use मे लेने के लिए कौनसा Bundled, Open Source Software use किया जाता है।

(अ) JDK (ब) Microsoft.Net

(स) Eclipse (द) XAMPP

4- Apache Web Server के लिए कौनसा Port use किया जाता है।

(अ) 82 (ब) 85

(स) 90 (द). 80

5- MySql के लिए कौनसा Port default काम मे लिया जाता है।

(अ) 3122 (ब) 8080

(स) 3000 (द) 3306

6- Php मे Script को Start करने के लिए कौनसे Tags काम मे लाये जाते है।

(अ) <?php————?> (ब) <!—php —!>

(स) <%php————%> (द) <%php————?>

7. निम्न मे से Loop का प्रकार कौनसा है।

(अ) while (ब). float

(स) int (द) long

8. निम्न मे से कौनसा Statement Conditional है।

(अ) if-else (ब) while

(स) for (द) foreach

9. निम्न मे से कौनसा Statement similar data type के collection के लिए काम मे आता है।

(अ) Array (ब) Union

(स) constructor (द) class

10. निम्न मे से Associative Array का कौनसा प्रकार है।

(अ) $_GET[ ] (ब) Array( )

(स) foreach (द). Class

लघूत्तरात्मक प्रश्न

1- Php मे एक Script को कैसे लिखा जाता है ?

2- Function को Php मे कैसे लिखते है ?

3- Php मे एक Array का निर्माण कैसे करते है ?

4- Php मे एक Expression को लिखकर बताइये।

5- Php मे Operator की Associatively क्या होती है?

6- Operator की List बनाइए।

7- Php मे echo व Print_r( ) Function को समझाइये।

8- Php मे Array व Class मे कोई दो अन्तर बताइये।

9- Php का MySql से Connection का code लिखकर बताइये।

10- mysql_connect( ) function को समझाइये।

निबन्धात्मक प्रश्न

1- Php मे Variable का Declare व Initialize करने के लिए Script को लिखिये।

2- Php को MySql से Connection Establishment व mysql_connect( ) व mysql_select ( ) function को Explain करें।

3- Php मे सभी Arithmetic Operation calculation perform करने का function सहित Script लिखिये।

4- MySql मे कौन–कौनसी Key होती है Explain कीजिए।

5- Php में सभी Operators को Detail मे उदाहरण सहित समझाइए।

उत्तरमाला

1. स 2. स 3. द 4. द 5. द 6. अ 7.अ 8.अ 9. अ 10. अ