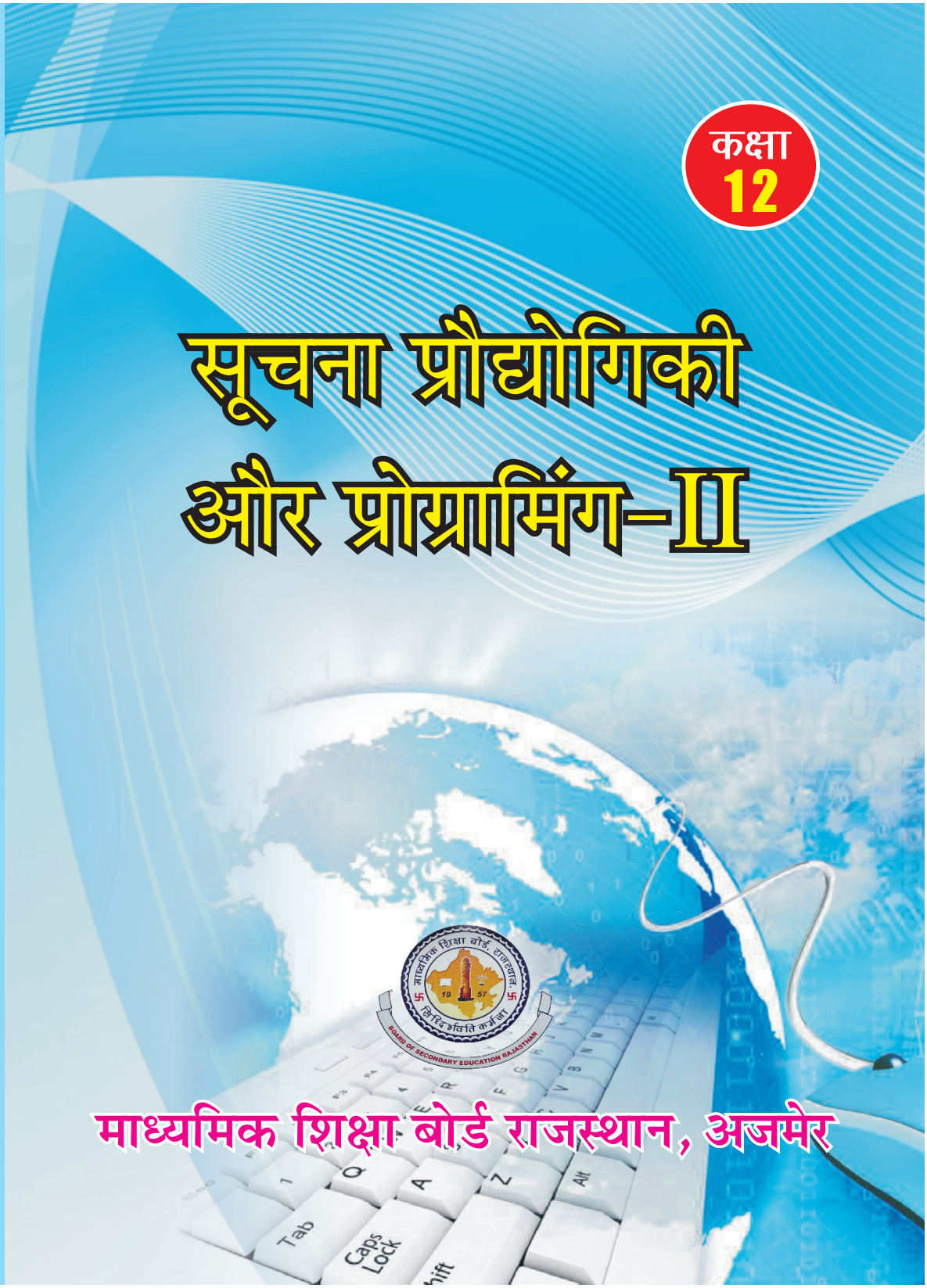
सूचना प्रौद्योगिकी और प्रोग्रामिंग-II

कक्षा 12

कक्षा
12

सूचना प्रौद्योगिकी और प्रोग्रामिंग-II



माध्यमिक शिक्षा बोर्ड राजस्थान, अजमेर

सूचना प्रौद्योगिकी और प्रोग्रामिंग –II

कक्षा XII

माध्यमिक शिक्षा बोर्ड राजस्थान, अजमेर

पाठ्यपुस्तक निर्माण समिति

सूचना प्रौद्योगिकी और प्रोग्रामिंग – II

कक्षा XII

संयोजक

हरजी राम चौधरी

सहायक आचार्य

राजकीय अभियांत्रिकी महाविद्यालय

अजमेर, राज़

लेखकगण :

विष्णु प्रकाश शर्मा

सहायक आचार्य

राजकीय अभियांत्रिकी महाविद्यालय

अजमेर, राज़

अनिल टेलर

सहायक आचार्य

राजकीय अभियांत्रिकी महाविद्यालय

अजमेर, राज़

पाठ्यक्रम समिति

सूचना प्रौद्योगिकी और प्रोग्रामिंग –II

कक्षा XII

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)

आमुख

इस पुस्तक में कम्प्यूटर प्रौद्योगिकी और प्रोग्रामिंग की आवश्यक अवधारणाओं को शामिल किया गया हैं। इस पुस्तक का समग्र उद्देश्य आपको विभिन्न डेटा स्ट्रक्चर, C++, DBMS और उसके अनुप्रयोगो के बारे मे परिचय कराना है।

इस पुस्तक की सामग्री हाल ही में संशोधित माध्यमिक शिक्षा बोर्ड, राजस्थान के पाठ्यक्रम के अनुसार है। इस पुस्तक मे 15 अध्याय शामिल हैं और प्रत्येक अध्याय का अपना एक महत्व है।

अध्याय 1 से 5, ये अध्याय पहली इकाई का हिस्सा हैं और इनमे डेटा स्ट्रक्चर, का परिचय जिसमें ऐरे, सॉर्टिगं स्टैक, क्यु और लिन्क लिस्ट को शामिल किया गया हैं।

अध्याय 6 से 12, ये अध्याय दूसरी इकाई का हिस्सा है इनमें C++ भाषा का परिचय जिसमें प्रोग्राम की कम्पायलिगं और लिंकिगं के तरीके, ऑपरेटस्, एक्सप्रेशनस और कन्ट्रोल स्ट्रक्चर, फंक्शन, क्लासेस और आब्जेक्स, कन्सट्रक्टर और डिसट्रक्टर, ऑपरेंटर ओवरलोडिंग और इनहेरिटेन्स को शामिल किया गया है

अध्याय 13 से 15, ये अध्याय तीसरी इकाई का हिस्सा है और इनमें DBMS का परिचय जिसमें DBMS की अवधारणायें ,रिलेशनल डेटाबेस की अवधारणायें और PL/SQL के आधार बिन्दुओं को शामिल किया गया है।

हम उन सबका हद्य से आभार प्रकट करते है, जिन्होनें इस कार्य को पूरा करने में हमारा समर्थन किया। हम उनका भी आभार प्रकट करते हैं जिन्होनें पुस्तक के लिये विविध सामग्री एकत्रित करने, सम्पादन और समीक्षा करने में सहयोग किया।

अन्त में हम विशेष रुप से अपने परिवार वालो के आभारी हैं जिनके सहयोग के बिना इस पुस्तक को समय पर पूरा करना सम्भव नहीं था।

पुस्तक में सुधार के लिए हम सुझावो का स्वागत करते हैं।

संयोजक एवं लेखकगण

पाठ्यक्रम

सूचना प्रौद्योगिकी और प्रोग्रामिंग –II

कक्षा XII

यूनिट–1: सी भाषा के उपयोग से डेटा स्ट्रक्चर:

डेटा संरचनाएँ के लिए परिचय: परिभाषा, डेटा संरचनाएँ का वर्गीकरण। डेटा संरचनाओं पर क्रियाऐं। **सारणियों:** सारणी परिभाषा, प्रतिनिधित्व और विश्लेषण, एकल और बहुआयामी, सारणी, पता गणना, सारणियों के उपयोग, सी में स्ट्रिंग ऑपरेशन। **डायनेमिक स्मृति आबंटन और संकेत:** परिभाषा, पॉइंटर का घोषित और प्रारम्भिकरण करना। स्थिर और डायनेमिक स्मृति आबंटन का अर्थ। स्मृति आबंटन फंक्शन: malloc, calloc, free, and realloc. **रिकर्शन:** परिभाषा, सी में रिकर्शन, रिकर्सिव प्रोग्राम–द्विपद गुणांक, फिबोनैकी, और GCD के प्रोग्राम। **संर्चिंग:** मूलभूत खोज तकनीक, तकनीक खोज विधि: अनुक्रमिक खोज, बाइनरी सर्च। अनुक्रमिक और बाइनरी सर्च के बीच तुलना। **सार्टिंग**– सामान्य पृष्ठभूमि: परिभाषा, विभिन्न प्रकार: बबल सॉर्ट, सिलेक्शन सॉर्ट, मर्ज सॉर्ट, इंसर्शन सॉर्ट, क्विक सॉर्ट। **स्टैक:** परिभाषा, सारणी के द्वारा स्टैक को बनना, स्टैक पर क्रियाऐं: इन्फ़िक्स, प्रीफिक्स और पोस्टफिक्स नोटेशन। स्टैक के उपयोग। **क्यू**– परिभाषा, सारणी के द्वारा क्यू को बनना। क्यू के प्रकार: सरल क्यू, सकुर्लर क्यू, क्यू पर क्रियाऐं। **लिंक्ड लिस्ट**– परिभाषा, लिंक्ड लिस्ट के भाग, लिंक्ड लिस्ट बनना, लिंक्ड लिस्ट के फायदे और नुकसान, लिंक्ड लिस्ट के प्रकार: सिंगली लिंक्ड लिस्ट, डबली लिंक्ड लिस्ट।

यूनिट–2: ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग भाषा – C++:

C++ प्रोग्राम की संरचना, कम्पाइलिंग एवं लिंकिंग, टोकंस, की वर्ड आइडेंटिफायर, कांस्टेंट, बेसिक डेटा टाइप, डिराइव्ड डेटा टाइप, टाइप कम्पोबिलिटी, वेरिएबल की घोषणा, C++ में ऑपरेटर, स्कोप रिजोलूशन ऑपरेटर। एक्सप्रेशन और उनके प्रकार, स्पेशन असाइनमेंट एक्सप्रेशन, इम्प्लिसिट कन्वर्शन,

ऑपरेटर ओवरलोडिंग, ऑपरेटर प्रिसीडेंस, कण्ट्रोल स्ट्रक्चर, C++ में फंक्शन: फंक्शन प्रोटोटाइप, कॉल बी रेफ़रेस, फंक्शन ओवरलोडिंग, क्लासेज एण्ड ऑब्जेक्ट: क्लास परिभाषा, मेम्बर फंक्शन परिभाषा, इनलाइन फंक्शन, एक्सेस मोडिफायर, ऐरे, स्टेटिक डेटा मेम्बर, स्टेटिक मेम्बर फंक्शन, फ्रेंड फंक्शन, फ्रेंड क्लास, रिटर्निंग ऑब्जेक्ट्स पॉईटंर टू मेम्बर। कन्स्ट्रक्टर: पैरामीटर कन्स्ट्रक्टर, एक क्लास में मल्टिपल कन्स्ट्रक्टर, कन्स्ट्रक्टर विद डिफाल्ट आर्गुमेंट, डायनामिक इनिशियालिज़ेशन ऑफ़ ऑब्जेक्ट कॉपी कन्स्ट्रक्टर, डेस्ट्रक्टर ऑपरेटर ओवरलोडिंग, युनेरीवर्चुअल वर्चुअल ऑपरेटर ओवरलोडिंग, बाइनरी ऑपरेटर ओवरलोडिंग, इन्हेरिटेंस: डीराइव्ड क्लासेज, सिंगल इन्हेरिटेंस, प्राइवेट मेम्बर इन्हेरिटेंस, मल्टीलेवल इन्हेरिटेंस, मल्टिपल इन्हेरिटेंस, हायरार्कीकल इन्हेरिटेंस, हाइब्रिड इन्हेरिटेंस, वर्चुअल वेब क्लासेज, एबस्ट्रेक्ट क्लासेज।

यूनिट–3: रिलेशन डेटाबेस मैनेजमेंट सिस्टम:

मूल डेटाबेस अवधारणाओं: डेटाबेस अवधारणा का परिचय डेटा मॉडल:– E-R मॉडल E-R आरेखों पदानुक्रमित, नेटवर्किंग, संबंधपरक डेटा मॉडल, आनुपातिक संरचना– संबंधपरक डेटा बेस मॉडल के

CODD की विशेषताओं के नियम– (संबंध) डेटा की कमीः referential विखंडन बाधाओं, निकाय संबंधी बाधाओं, बाधाओं की तरह प्राथमिक कुंजी प्रतिबंध, अद्वितीय, चेक बाधा मजबूत संस्था है कमजोर एंटिटी। सामान्यीकरणः परिचय– उद्देश्य का सामान्यीकरण परिभाषा के कार्यात्मक निर्भरता (एफडी) संबंधपरक डेटाबेस डिजाइन। **SQL परिचयः** SQL, डेटा डेफिनेशन भाषा (DDL), डेटा मैनीपुलेशन भाषा (DML), डेटा नियंत्रण भाषा (DCL), डेटा क्वेरी भाषा (DQL), और सभी आदेशों के लाभ। प्रारूप मॉडलः चरित्र, सांख्यिक दिनांक स्वरूप मॉडल। ऑपरेटरः तार्किक, मान, वाक्यविन्यास और क्वेरी व्यंजक ऑपरेटरों– सेट ऑपरेटरों। कार्यः चरित्र, अंकगणित, दिनांक और समय, समूह और विविध कार्यों, कमिट, रोलबैक, सवेपॉइंट क्वेरीज द्वारा समूह और खंड सम्मिलित हों द्वारा आदेश का उपयोगः एक एकल आदेश के परिणाम, परिणाम, समूहीकरण तालिका, क्वैरी में मिलती है, प्रकार की मिलती है, उप प्रश्नों में प्रवेश करें। PL\SQLः PL\SQL की मूल बातें, डेटा प्रकार, कंट्रोल स्ट्रक्चर, PL\SQL से डेटाबेस एक्सेस, डेटाबेस कनेक्शन, कर्सर प्रबंधन का परिचय, इम्प्लिसिट और एक्सप्लिसिट कर्सर, त्रुटि संभालना, पूर्व निर्धारित एवं प्रयोक्ता निर्धारित एक्सपेशन, प्रोसीजन एवं फंक्शन का परिचय उनका ओवरलोडिंग, डेटाबेस ट्रिगर का परिचय।

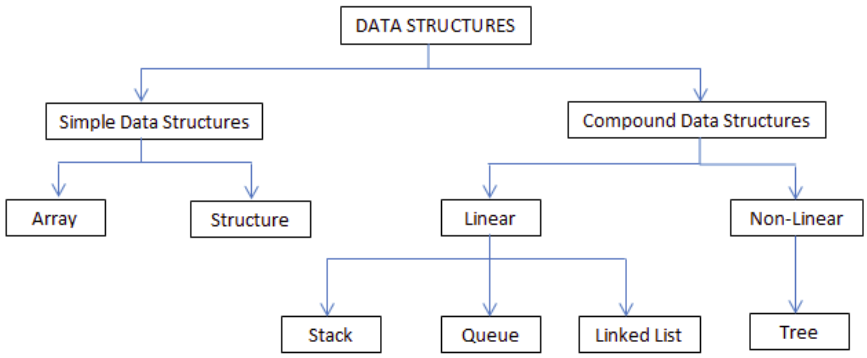
विषय सूची

अध्याय 1

डाटा स्ट्रक्चर का परिचय

डाटा स्ट्रक्चर डेटा को एकत्रित (कलेक्ट) और आयोजित(ऑर्गनाइज) करने का एक तरीका है जिससे हम प्रभावी तरीके से इन डेटा पर कार्रवाई (आपरेशन) कर सके। डाटा स्ट्रक्चरडेटा के बेहतर व्यवस्था और भंडारण के लिए होते है। डेटा स्ट्रक्चर किसी भी संगठन के डेटा का एक तार्किक और गणितीय दृश्य है। उदाहरण के लिए, डेटा के रूप में खिलाड़ी का नाम विराट और उम्र 26 है। यहा विराट स्ट्रिंग प्रकार का डेटा है और 26 पूर्णांक प्रकार का डेटा है। हम इस डेटा को एक रिकार्ड के रूप में जैसे एक खिलाड़ी रिकॉर्ड की तरह व्यवस्थित कर सकते हैं। अब हम खिलाड़ी रिकॉर्ड को एक फाइल या डेटाबेस में एक डाटा स्ट्रक्चर के रूप में संचित (स्टोर) कर सकते हैं उदाहरण के लिए धोनी 30, गंभीर 31, सहवाग 33।

डाटा स्ट्रक्चर का वर्गीकरण:



सरल डाटा स्ट्रक्चर

ये आम तौर पर प्रिमिटिव डाटा टाइप्स जैसे इंटिजर्स, रियल, करैक्टर, बूलियन से बनाया जाता है।सरल डाटा स्ट्रक्चर निम्नलिखित दो प्रकार के होते हैं:

1. ऐरे
2. स्ट्रक्चर

यौगिक डाटा स्ट्रक्चर

सरल डेटा स्ट्रक्चर को विभिन्न तरीकों में संयोजित करके जटिल डाटा स्ट्रक्चर बनाये जा सकते है। ये निम्नलिखित दो प्रकार के होते हैं:

1. रेखीय डाटा स्ट्रक्चर(Linear Data Structure)

ये एकल स्तर के डाटा स्ट्रक्चर होते हैं। इनके तत्व एक अनुक्रम (सीक्वेंस) बनाते है इसलिए इन्हे रेखीय डाटा स्ट्रक्चर कहते है।

ये निम्नलिखित प्रकार के होते हैं:

1. स्टैक
2. क्यु
. लिंक लिस्ट

2. गैर रेखीय डाटा स्ट्रक्चर(Non-linear Data Structure)

ये बहुस्तरीय डाटा स्ट्रक्चर होते हैं। गैर रेखीय डाटा स्ट्रक्चर के उदाहरण ट्री और ग्राफ हैं।

डाटा स्ट्रक्चर पर ऑपरेशन: डाटा स्ट्रक्चर पर किये जाने वाले बुनियादी ऑपरेशन इस प्रकार है:

इनर्सशन (Insertion): इनर्सशन का अर्थ एक डाटा स्ट्रक्चर में एक नये डेटा तत्व को जोडना।

डिलिशन (deletion): डिलिशन का अर्थ एक डाटा स्ट्रक्चर में एक डेटा तत्व को हटाना यदि वह मौजूद है।

सर्च (Search): एक डाटा स्ट्रक्चर में निर्दिष्ट डेटा तत्व को खोजने को सर्च कहते है।

ट्रर्वसिंग (Traversing):एक डाटा स्ट्रक्चर में मौजूद सभी डेटा तत्वों के प्रसंस्करण (प्रोसेसिंग) को ट्रर्वसिंग कहते है।

सॉर्टिंग (Sorting):डाटा स्ट्रक्चर के तत्वों को एक निर्दिष्ट क्रम में व्यवस्थित करने को सॉर्टिंग कहते है।

मर्जिंग (Merging):दो एक ही प्रकार के डाटा स्ट्रक्चर के तत्वों का संयोजन कर उसी प्रकार के एक नये डाटा स्ट्रक्चर बनाने को मर्जिंग कहते हैं।

एल्गोरिथ्म: एल्गोरिथ्म तर्क या निर्देशों का एक परिमित सेट है जो एक निश्चित पूर्वनिर्धारित कार्य को पूरा करने के लिए लिखी जाती है। एल्गोरिथ्म पूरा कोड़ या प्रोग्रम नहीं होता,यह सिर्फ मूल समस्या का समाधान है, और इसे एक अनौपचारिक उच्च स्तरीय विवरण के रूप में जैसें शुडोकोड़ (pseudocode) या फ्लोचार्ट से व्यक्त किया जा सकता है। यदि एल्गोरिथ्म को निष्पादित करनें में कम समय एवंम् मैमोरी की आवश्यकता होती है तो कुशल और तेज एल्गोरिथ्म कहते है। एक एल्गोरिथ्म का प्रदर्शन निम्नलिखित गुण के आधार पर मापा जाता है:

- स्पेस जटिलता (Space Complexity)
- समय जटिलता (Time Complexity)

स्पेस जटिलता(Space Complexity)

एल्गोरिथ्म के निष्पादन के दौरान आवश्यक मैमोरी को स्पेस जटिलता कहते है। बहु उपयोगकर्ता सिस्टम के लिए और जब सीमित रूप से मैमोरी उपल्बध हो तब इसे गंभीरता से लिया जाना चाहिए।आम तौर पर एक एल्गोरिथ्म को निम्नलिखित घटकों के लिए मैमोरी की आवश्यकता होती है

इन्सट्रक्शन स्पेसः यह प्रोग्राम के निष्पादन योग्य संस्करण को सचंय करने के लिए आवश्यक मैमोरी स्पेस है। यह स्पेस निश्चित या स्थायी होती जोकि प्रोग्राम में कोड की लाइनों की संख्या पर निर्भर करती है।

डेटा स्पेसः यह सभी कॉन्सटेन्ट और वैरिएबल मानों को सचिंत करने के लिए आवश्यक स्पेस है।

समय जटिलता (Time Complexity)

एक प्रोग्राम के पूर्ण निष्पादन के लिए आवश्यक समय का प्रतिनिधित्व करने के लिए एक तरीका है। एल्गोरिथ्म की समय जटिलता को सबसे अधिक Big O संकेतन का उपयोग करके व्यक्त किया है। एल्गोरिथ्म द्वारा किये जाने वाले प्राथमिक फंक्शनों की संख्या को गिन कर समय जटिलता की गणना की जाती है। और क्यों कि एल्गोरिथ्म की कार्यक्षमता अलग–अलग इनपुट डेटा के साथ अलग–अलग हो सकती है इसलिए एल्गोरिथ्म की वर्स्ट केस (बुरी से बुरी) समय जटिलता का उपयोग करते है। यह किसी भी इनपुट आकार के लिए एल्गोरिथ्म द्वारा लिया जाने वाला अधिकतम समय होता है।

महत्वपूर्ण बिंदु

- डेटा स्ट्रक्चर किसी भी संगठन के डेटा का एक तार्किक और गणितीय दृश्य है।
- सरल डेटा स्ट्रक्चर को विभिन्न तरीकों में संयोजित करके जटिल डाटा स्ट्रक्चर बनाये जा सकते है।
- एल्गोरिथ्म तर्क या निर्देशों का एक परिमित सेट है जो एक निश्चित पूर्वनिर्धारित कार्य को पूरा करने के लिए लिखी जाती है।
- एल्गोरिथ्म के निष्पादन के दौरान आवश्यक मैमोरी को स्पेस जटिलता कहते है। बहु उपयोगकर्ता सिस्टम के लिए और जब सीमित रूप से मैमोरी उपल्बध हो तब इसे गंभीरता से लिया जाना चाहिए।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्र 1 जब एल्गोरिथ्म की दक्षता का निर्धारण करते है तो स्पेस पहलू मापा जाता है

(अ) एल्गोरिथ्म द्वारा आवयश्क अधिकतम मैमोरी की।

(ब) एल्गोरिथ्म द्वारा आवयश्क न्यूनतम मैमोरी की।

(स) एल्गोरिथ्म द्वारा आवयश्क औसत मैमोरी की।

(द) एल्गोरिथ्म द्वारा आवयश्क अधिकतम डिस्क मैमोरी की।

प्र 2 एक एल्गोरिथ्म के लिए औसत मामले की जटिलता है

(अ) एल्गोरिथ्म के औसत मामले का विश्लेषण बहुत अधिक जटिल है।

(ब) एल्गोरिथ्म के औसत मामले का विश्लेषण बहुत सरल है।

(स) कभी कभी अधिक जटिल और कभी कभी अधिक सरल होता है।

(द) उपरोक्त में से कोई नहीं।

प्र 3 एल्गोरिथ्म की दक्षता का निर्धारण करने के लिए समय का पहलू मापा जाता है

(अ) माइक्रोसेकंड की गिनती

(ब) प्रमुख आपरेशनों की संख्या की गणना

(स) बयानों की संख्या की गणना

(द) एल्गोरिथ्म के किलोबाइट की गिनती

प्र 4 निम्नलिखित में से रैखिक डेटा स्ट्रक्चर है?

(अ) ट्री (ब) ग्राफ

(स) ऐरे (द) इनमें से कोई भी नहीं

प्र 5 निम्नलिखित में से रैखिक डेटा स्ट्रक्चर नहीं है?

(अ) ऐरे (ब) लिंक लिस्ट

(स) ऊपर के दोनों (द) इनमें से कोई भी नहीं

लघुत्तरात्मक प्रश्न

प्र 1 डेटा स्ट्रक्चर क्या है?

प्र 2 एक एल्गोरिथ्म की कुशलता के लिए दो मुख्य उपाय क्या हैं?

प्र 3 समय जटिलता क्यों महत्वपूर्ण है?

प्र 4 रेखीय डेटा स्ट्रक्चर के उदाहरण दीजिऐ।

निबंधात्मक प्रश्न

प्र 1स्पेस जटिलता की गणना कैसे की जा सकती है?

प्र 2 डेटा स्ट्रक्चर का क्या उपयोग है?

प्र 3 यौगिक डेटा स्ट्रक्चर को समझाओ।

उत्तरमाला

उत्तर 1: अ उत्तर 2: स उत्तर 3: ब
उत्तर 4: स उत्तर 5: द

अध्याय 2

ऐरे (Array)

ऐरे,समरूप डेटा तत्वों के परिमित क्रमो का एक संग्रह है जोकि क्रमिक मैमोरी स्थानों में संग्रहित होते है।

यहाँ शब्द

परिमित का अर्थ डेटा रेंज निर्धारित होनी चाहिए।

क्रम का अर्थ डेटा निरंतर मैमोरी स्थानों में संग्रहित किया जाना चाहिए।

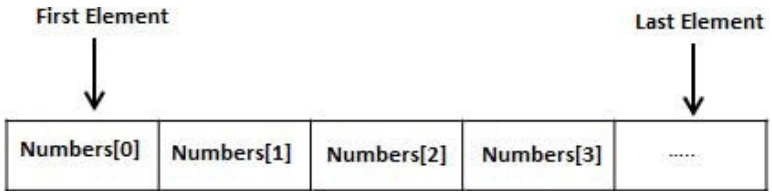
समरूप का अर्थ डेटा एक ही प्रकार का होना चाहिए।

ऐरे दो प्रकार का होता हैं:

1. एकल या एक आयामी ऐरे
2. बहु आयामी ऐरे

एकल या एक आयामी ऐरे:

आइटमों की एक सूची के लिए केवल एक सबस्क्रिप्ट का उपयोग करके एक वैरिएबल नाम दिया जा सकता है और इस तरह के वैरिएबल को एकल सबस्क्रिप्टेड वैरिएबल या एकल आयामी ऐरे कहा जाता है।



एक आयामी ऐरे की घोषणा (डिक्लेरेशन):किसी भी अन्य वैरिएबल की तरह, ऐरे को भी उपयोग से पहले डिक्लेयर किया जाना चाहिए ताकि कम्पाईलर उनके लिए मैमोरी में स्पेस आंवटित कर सके। ऐरे को निम्न प्रकार से डिक्लेयर किया जाता है:

type variable-name[size];

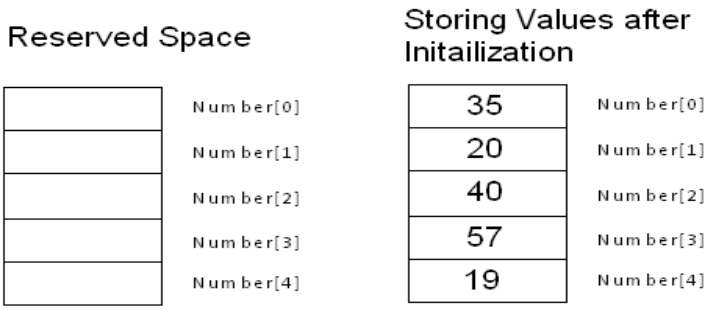
उदाहरण

```
int groupt[10];
float height[50];
char name[10];
```

टाईप ऐरे में संग्रहित होने वाले तत्वों के प्रकार को बताता है जैसे कि int, float, और char एवंम् वैरिएबल नेम ऐरे के नाम को बताता हैजैसे कि height, group और name है साईज ऐरे में सग्रंहित किये जा सकने वाले तत्वों कि अधिकतम संख्या को इंगित करता है। सी प्रोग्रामिंग भाषा, करेक्टर स्ट्रींग को करेक्टर के ऐरे के रूप में ही प्रबंध करता है।

पांच तत्वों के लिए एक ऐरे की घोषणाःint number[5];

नीचे दिखाये अनुसार कंप्यूटर मैमोरी में पांच भंडारण स्थानों को आरक्षित कर लेता है क्यों की ऐरे का साइज 5 है–



एकल या एक आयामी ऐरे का प्रारंभ: एक ऐरे के डिक्लेरेशन के बाद उसके तत्व प्रारंभ किये जाते है सी प्रोग्रामिंग में एक ऐरे निम्न चरणों में प्रारंभ किया जा सकता है:

– कंपाइल टाइम

– रन टाइम

कंपाइलटाइम प्रारंभ:जब एक ऐरे के डिक्लेरेशन के साथ उसे प्रारंभ किया जाता है तो ऐरे निम्न प्रकार से प्रारंभहोगा : type array-name[size] = { list of values };

लिस्ट में मानो को कोमा से अलग किया जाता है उदहारण के लिए

int number[3] = { 0,5,4 };

ऊपर दिए गए स्टेटमेंट में 3 आकार का एक नंबर नाम का ऐरे है और हर तत्व को वैल्यू आवंटित होगी। लिस्ट में वैल्यू की संख्या ऐरे साईज की तुलना में कम है, तो यह केवल कुछ ऐरे तत्वो को वैल्यू आवंटित करेगा। शेष तत्वों को स्वचालित रूप से शून्य आवंटित हो जायेगा।

याद रखें, यदि घोषित आकार की तुलना में अधिक वैल्यू है, तो एक त्रुटि का उत्पादन होगा।

रन टाइम प्रारंभ: एक ऐरे को स्पष्ट रूप से चलाने के लिए रन टाइम आरंभ किया जा सकता है उदाहरण के लिए निम्नलिखित सी प्रोग्राम के खंड पर विचार करें।

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

उदाहरण के लिए ऊपर कीबोर्ड से वैल्यू देते है रन टाइम में लूपिंग स्टेटमेंट जरुरी है असाइनमेंट ऑपरेटर की सहायता से एक एक वैल्यू ऐरे में स्टोर करते है

एक आयामी ऐरे का प्रोग्राम:

```
/ * ऐरे में तत्वो को स्टोर करने और प्रिंट करने के लिए सरल C प्रोग्राम * /
        #include<stdio.h>
        void main()
        {
int array[5],i;
              printf("Enter 5 numbers to store them in array \n");
            for(i=0;i<5;i++)
            {
                scanf("%d",&array[i]);
            }
                 printf("Element in the array are - \n \n");
            for(i=0;i<5;i++)
            {
                 printf("Element stored at a[%d] = %d \n",i,array[i]);
            }
                getch();
        }
```

इनपुट(Input)– Enter 5 elements in the array – 23 45 32 25 45

आउटपुट (Output) – Elements in the array are –

Element stored at a[0]-23

Element stored at a[1]-45

Element stored at a[2]-32

Element stored at a[3]-25

Element stored at a[4]-45

बहु आयामी ऐरे: ऐरे का ऐरे एक बहुआयामी ऐरे कहलाता है। सामान्य रूप से बहुआयामी ऐरे की घोषणा निम्न प्रकार से होती है।

type variable-name [size1] [size2] --- [sizeN];

बहुआयामी ऐरे का सरलतम रूप दो आयामी ऐरे है उदाहरण:

int x [3] [4];

उपरोक्त x एक दो आयामी (2डी) ऐरे है। ऐरे में 12 तत्व है। यहाँ x 3 पंक्ति के साथ तालिका के रूप में ऐरे है। और प्रत्येक पंक्ति में 4 स्तंभ है।

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

दो आयामी (2डी) ऐरे का प्रारंभ:एक आयामी ऐरे की तरह, 2डी ऐरे को भी दोनों प्रकार (कंपाइल टाइम व रन टाइम) से प्रारंभ किया जा सकता है

कंपाइल टाइम आरंभीकरण –जब एक ऐरे के डिक्लेरेशन के साथ उसे प्रारंभ किया जाता है तो दो आयामी ऐरे निम्न प्रकार से प्रारंभहोगा :

int table-[2][3] = {
{ 0, 2, 5}

```
{ 1, 3, 0}
};
```

रन टाइम आरंभीकरण–एक ऐरे को स्पष्ट रूप से चलाने के लिए रन टाइम आरंभ किया जा सकता है दो आयामी ऐरे को लूप स्ट्रक्चर की मदद से आरम्भ करते है। दो लूप स्ट्रक्चर उपयोग में ली जाती है। जिसमें आउटर लुप पक्ति के लिए एवंम इनर लूप कॉलम के उपयोग में आती है। उदाहरण के लिए निम्नलिखित सी प्रोग्राम के खंड पर विचार करें।

```
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&ar1[i][j]);
}
}
```

2डी ऐरे का प्रोग्राम:

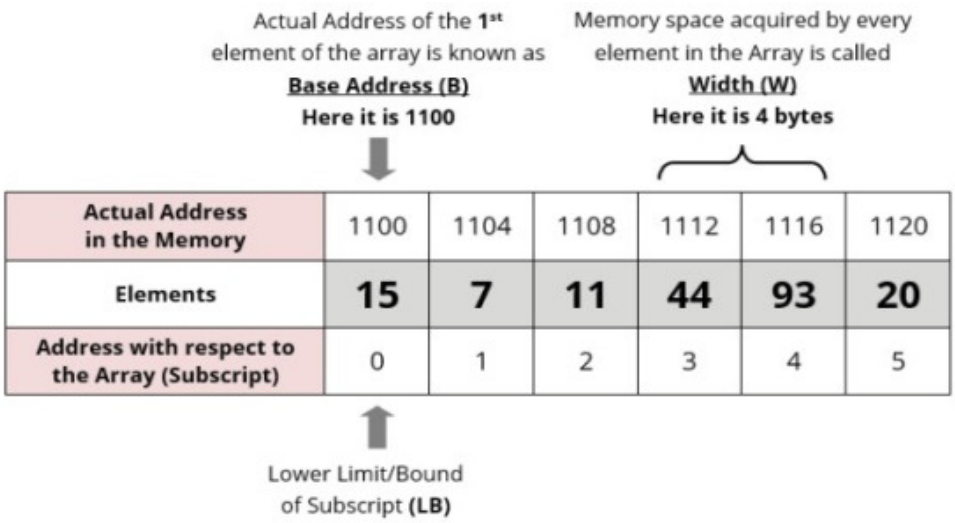
```
/ * 2–डी ऐरे का सी प्रोग्राम * /

        #include<stdio.h>
        #include<conio.h>
         void main()
{
            int array[3][3],i,j,count=0;

              /* Run time Initialization */
for(i=1;i<=3;i++)

            {
for(j=1;j<=3;j++)
{
count++;
array[i][j]=count;
printf("%d \t",array[i][j]);
}
printf("\n");
}
```

```
        getch();
}
```

Output –

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

एकल (एक) आयामी ऐरे में पता गणना:



एक ऐरे "A [I]" के एक तत्व की गणना निम्न सूत्र के उपयोग से करते है–

Address of A [I] = B + W * (I – LB)

Where,

B = आधार पता

W = ऐरे में उपस्थित एक तत्व की स्टोरेज साईज (बाइट में)

I = जिस तत्व का पता ज्ञात करना है उसका सबस्क्रिप्ट

LB = नीचली सीमा/उपलब्ध नहीं होने पर शून्य माने 0 (शून्य)

उदाहरण:

एक ऐरे [1300 --- --1900] का आधार पता 1020 और प्रत्येक तत्व का आकार मैमोरी में 2 बाइट्स के रूप में है। B [1700], का पता गणना किजिए

उतर :

दिए गए मान निम्न है B = 1020, LB = 1300, W = 2, I= 1700

A [I] का पता = B + W * (I – LB)

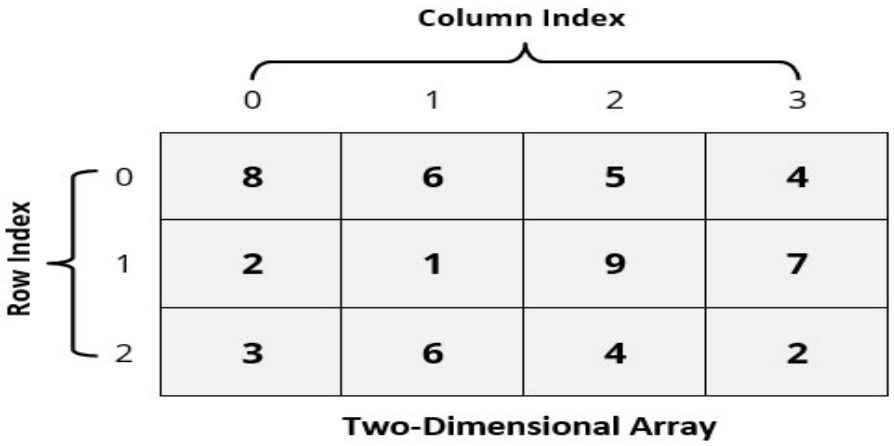
= 1020 + 2 * (1700 – 1300)

= 1020 + 2 * 400

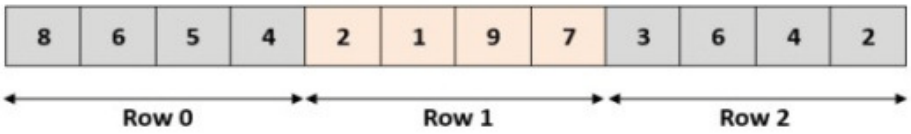
= 1020 + 800

= 1820 [Ans]

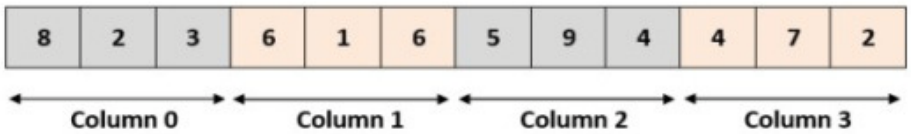
मल्टी (दो) आयामी ऐरे में पता गणना:मैमोरी में एक 2–डी ऐरे के तत्वों को संग्रह करते समय इन्हें क्रमिक मैमोरी लोकेशन आवंटित किये जाते है। इसलिए उसके भंडारण को सक्षम करने के लिए2–डी ऐरे को लीनियराइज करते है। लीनियराइज करने के दो तरीके होते है। रो (पंक्ति) मेजर और कॉलम (स्तंभ) मेजर।



Row-Major (Row Wise Arrangement)



Column-Major (Column Wise Arrangement)



ऐरे के किसी तत्व "A [I] [J]" के पता की गणना नीचे दिए गए दो प्रकार से की जा सकती है

(क) पंक्ति प्रमुख प्रणाली (Row Major System)

(ख) कॉलम प्रमुख प्रणाली (Column Major System)

पंक्ति प्रमुख प्रणाली:

पंक्ति प्रमुख प्रणाली में एक लोकेशन का पता निम्न सूत्र का उपयोग करके किया जाता है:

A [I][J] तत्व का पता= B + W * [N * (I – Lr)+ (J – Lc)]

स्तंभ (कॉलम) प्रमुख प्रणाली:

कॉलम प्रमुख प्रणाली में एक लोकेशन का पता निम्न सूत्र का उपयोग करके किया जाता है:

A [I][J] तत्व का पता = B + W * [(I – Lr)+ M * (J – Lc)]

यहाँ पे,

B = आधार पता

I = जिस तत्व का पता ज्ञात करना है उसका पक्ति सबस्क्रिप्ट

J = जिस तत्व का पता ज्ञात करना है उसका स्तंभ सबस्क्रिप्ट

W = ऐरे में उपस्थित एक तत्व की स्टोरजे साईज (बाइट में)

Lr = पंक्ति की नीचली सीमा/उपलब्ध नहीं होने पर शून्य माने 0 (शून्य)

Lc = स्तंभ की नीचली सीमा / उपलब्ध नहीं होने पर शून्य माने 0 (शून्य)

M = मैट्रिक्स में पंक्तियों की संख्यां

N = मैट्रिक्स में स्तंभों की संख्यां

नोट: एक मैट्रिक्स में आमतौर पर पंक्तियों और स्तंभों की संख्या (इस तरह A [20] [30] या A[40] [60]), दी जाती है

लेकिन A[Lr- – – – – Ur, Lc- – – – – Uc] के रूप में दिया जाता है, तो पंक्तियों और स्तंभों की संख्या निम्नलिखित तरीकों का उपयोग कर गणना कर करते हैं

पंक्तियाँ (M) की संख्या की गणना = (Ur – Lr) + 1

स्तंभों (N) की संख्या की गणना = (Uc – Lc) + 1

एवं अन्य प्रक्रिया आवश्यकतानुसार पंक्ति मेजर और स्तंभ मेजर समान रहेगी

उदाहरण:

एक ऐरे ,X[-15 ..... .10, 15 ..... 40],जिसमें एक बाईट स्टोरेज की आवश्यकता है। अगर शुरुआत स्थान 1500 है तो लोकेशन x[15] [20], ज्ञात किजिए।

M = (Ur – Lr) + 1 = [10 – (- 15)] +1 = 26

N = (Uc – Lc) + 1 = [40 – 15)] +1 = 26

(I) उपरोक्त की कॉलम मेजर गणनाः

दिए गए मान हैं B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, M = 26

A [I][J] का पता = B + W * [(I – Lr) + M * (J – Lc)]

= 1500 + 1 * [(15 – (-15)) + 26 * (20 – 15)] = 1500 + 1 * [30 + 26 * 5] = 1500 + 1 * [160] = 1660 [Ans]

(II) उपरोक्त की पक्ति मेजर गणनाः

दिए गए मान हैं B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, N = 26

A [I][J] का पता = B + W * [N * (I – Lr) + (J – Lc)]

= 1500 + 1* [26 * (15 – (-15))) + (20 – 15)] = 1500 + 1 * [26 * 30 + 5] = 1500 + 1 * [780 + 5] = 1500 + 785

= 2285 [Ans]

ऐरे पर बुनियादी ऑपरेशन: निम्नलिखित ऑपरेशन ऐरे पर किये जा सकते है

(क) ट्रर्वसिंग **(Traversing):** एक डाटा स्ट्रक्चर में मौजूद सभी डेटा तत्वों के प्रसंस्करण (प्रोसेसिंग) को ट्रर्वसिंग कहते है।

(ख) इनर्सशन **(Insertion):** इनर्सशन का अर्थ एक डाटा स्ट्रक्चर में एक नये डेटा तत्व को जोडना।

(ग) डिलिशन **(deletion):** डिलिशन का अर्थ एक डाटा स्ट्रक्चर में एक डेटा तत्व को हटाना यदि वह मौजूद है।

(घ) सर्च **(Search):** एक डाटा स्ट्रक्चर में निर्दिष्ट डेटा तत्व को खोजने को सर्च कहते है।

(ड़) अपडेट **(Update)–** दिए गए सूचकांक में एक तत्व अपडेट करता है।

ट्रर्वसिंग **(Traversing)**: एक डाटा स्ट्रक्चर में मौजूद सभी डेटा तत्वों के प्रसंस्करण (प्रोसेसिंग) को ट्रर्वसिंग कहते है। निम्नलिखित एल्गोरिथ्म से लीनियर ऐरे को ट्रर्वस कर सकते है।

1. Repeat For I = LB to UB
2. Apply PROCESS to A[I]

[End of For Loop]

3. Exit

इनर्सशन **(Insertion)**: इनर्सशन का अर्थ एक डाटा स्ट्रक्चर में एक नये डेटा तत्व को जोडना। निम्नलिखित एल्गोरिथ्म से लीनियर ऐरे में इनर्सट कर सकते है।

Algorithm: Let LA be a Linear Array (unordered) with N elements and K is a positive integer such that K<=N. Following is the algorithm where ITEM is inserted into the Kth position of LA –

1. Start
2. Set J = N
3. Set N = N+1
4. Repeat steps 5 and 6 while J >= K
5. Set LA[J+1] = LA[J]
6. Set J = J-1
7. Set LA[K] = ITEM
8. Stop

C Program for Insertion:

```
#include <stdio.h>

main() {
int LA[] = {1,3,5,7,8};
int item = 10, k = 3, n = 5;
int i = 0, j = n;

printf("The original array elements are :\n");

for(i = 0; i<n; i++) {
printf("LA[%d] = %d \n", i, LA[i]);
}

n = n + 1;

while( j >= k) {
LA[j+1] = LA[j];
j = j - 1;
}

LA[k] = item;

      printf("The array elements after insertion :\n");
```

```
for(i = 0; i<n; i++) {
printf("LA[%d] = %d \n", i, LA[i]);
}
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

The original array elements are :

LA[0] = 1

LA[1] = 3

LA[2] = 5

LA[3] = 7

LA[4] = 8

The array elements after insertion :

LA[0] = 1

LA[1] = 3

LA[2] = 5

LA[3] = 10

LA[4] = 7

LA[5] = 8

डिलिशन **(deletion)**: डिलिशन का अर्थ एक डाटा स्ट्रक्चर में एक डेटा तत्व को हटाना यदि वह मौजूद है।निम्नलिखित एल्गोरिथ्म से लीनियर ऐरे के तत्वों को डिलीट कर सकते है।

Algorithm: consider LA is a linear array with N elements and K is a positive integer such that K<=N. Following is the algorithm to delete an element available at the Kth position of LA.

1. Start

2. Set J = K

3. Repeat steps 4 and 5 while J < N

4. Set LA[J-1] = LA[J]

5. Set J = J+1

6. Set N = N-1

7. Stop

C Program for Deletion:

```
#include <stdio.h>
main() {
int LA[] = {1,3,5,7,8};
int k = 3, n = 5;
int i, j;
printf("The original array elements are :\n");

for(i = 0; i<n; i++) {
printf("LA[%d] = %d \n", i, LA[i]);
}

j = k;

while( j < n) {
LA[j-1] = LA[j];
j = j + 1;
}

n = n -1;

printf("The array elements after deletion :\n");

for(i = 0; i<n; i++) {
printf("LA[%d] = %d \n", i, LA[i]);
}
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
The original array elements are :
LA[0] = 1
LA[1] = 3
LA[2] = 5
```

LA[3] = 7

LA[4] = 8

The array elements after deletion :

LA[0] = 1

LA[1] = 3

LA[2] = 7

LA[3] = 8

सर्च **(Search)**: एक डाटा स्ट्रक्चर में निर्दिष्ट डेटा तत्व को खोजने को सर्च कहते है लीनियर ऐरे में सर्च दो प्रकार की होती है

(क) लीनियर सर्च

(ख) बाइनरी सर्च

लीनियर सर्च **:**लीनियर सर्च बुनियादी और सरल सर्च एल्गोरिथ्म है। लीनियर सर्च में तत्व और मान को तब तक सर्च करते है जब तक मिल नहीं जाता हैं इसमें दिये गये तत्वों को ऐरे में उपलब्ध सभी तत्वों से तुलना करते हैं एवंम मिलान होने पर ऐरे इनडेक्स का मान प्राप्त होता है अन्यथा –1। लीनियर सर्च सॉटेड और अनर्सोटेड मानों पर लागु करते है। जब तत्वों की संख्या कम हो। लिनीयर सर्च की निम्नलिखित एल्गोरिथ्म है।

Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Following is the algorithm to find an element with a value of ITEM using sequential search

1. Start
2. Set J = 0
3. Repeat steps 4 and 5 while J < N
4. IF LA[J] is equal ITEM THEN GOTO STEP 6
5. Set J = J +1
6. PRINT J, ITEM
7. Stop

C Program for Searching:

```
#include <stdio.h>
main() {
int LA[] = {1,3,5,7,8};
int item = 5, n = 5;
```

```
int i = 0, j = 0;

printf("The original array elements are :\n");

for(i = 0; i<n; i++) {
printf("LA[%d] = %d \n", i, LA[i]);
}

while( j < n){
if( LA[j] == item ) {
break;
}

j = j + 1;
}

printf("Found element %d at position %d\n", item, j+1);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

Output

The original array elements are:

LA[0] = 1

LA[1] = 3

LA[2] = 5

LA[3] = 7

LA[4] = 8

Found element 5 at position 3

बाइनरी सर्चः बाइनरी सर्च सॉटेड ऐरे या लिस्ट पर लागू किया जाता है। सर्वप्रथम हम दिये गये तत्व की ऐरे के बीच के तत्व से तुलना करते है यदि तत्व के मान का मिलान हो जाता है तो ऐरे का इनडेक्स मान रिर्टन करता है। यदि तत्व का मान कम है तो निचले आधे हिस्से में होगा अन्यथा ऊपरी आधे हिस्से में होगा। बाइनरी सर्च का उपयोग तत्वों की संख्या अधिक होने पर किया जाता हैं निम्नलिखित बाइनरी सर्च की एल्गोरिथ्म है।

Compare x with the middle element.

If x matches with middle element, we return the mid index.

Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.

Else (x is smaller) recur for the left half.

C Prgram for Binary Search:

```
#include <stdio.h>

#define MAX 20

// array of items on which linear search will be conducted.
int intArray[MAX] =
{1,2,3,4,6,7,9,11,12,14,15,16,17,19,33,34,43,45,55,66};

void printline(int count) {
int i;

for(i = 0;i <count-1;i++) {
printf("=");
}

printf("=\n");
}

int find(int data) {
int lowerBound = 0;
int upperBound = MAX -1;
int midPoint = -1;
int comparisons = 0;
int index = -1;

while(lowerBound <= upperBound) {
printf("Comparison %d\n" , (comparisons +1) );
printf("lowerBound : %d, intArray[%d] =
%d\n",lowerBound,lowerBound,
intArray[lowerBound]);
printf("upperBound : %d, intArray[%d] =
%d\n",upperBound,upperBound,
intArray[upperBound]);
```

```
comparisons++;

// compute the mid point
// midPoint = (lowerBound + upperBound) / 2;
midPoint = lowerBound + (upperBound - lowerBound) / 2;

// data found
if(intArray[midPoint] == data) {
index = midPoint;
break;
} else {
// if data is larger
if(intArray[midPoint] < data) {
// data is in upper half
lowerBound = midPoint + 1;
}
// data is smaller
else {
// data is in lower half
upperBound = midPoint -1;
}
}
}
printf("Total comparisons made: %d" , comparisons);
return index;
}

void display() {
int i;
printf("[");

// navigate through all items
for(i = 0;i<MAX;i++) {
printf("%d ",intArray[i]);
}

printf("]\n");
}
```

```
main() {
printf("Input Array: ");
display();
printline(50);

//find location of 1
int location = find(55);

// if element was found
if(location != -1)
printf("\nElement found at location: %d" ,(location+1));
else
printf("\nElement not found.");
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
Input Array: [1 2 3 4 6 7 9 11 12 14 15 16 17 19 33 34 43 45 55 66 ]

Comparison 1
lowerBound : 0, intArray[0] = 1
upperBound : 19, intArray[19] = 66
Comparison 2
lowerBound : 10, intArray[10] = 15
upperBound : 19, intArray[19] = 66
Comparison 3
lowerBound : 15, intArray[15] = 34
upperBound : 19, intArray[19] = 66
Comparison 4
lowerBound : 18, intArray[18] = 55
upperBound : 19, intArray[19] = 66
Total comparisons made: 4
Element found at location: 19
```

अपडेट **(Update)–** दिए गए सूचकांक में एक तत्व अपडेट करता है। निम्नलिखित एल्गोरिथ्म एक तत्व को ऐरे में अपडेट करता है

Consider LA is a linear array with N elements and K is a positive integer such that K<=N.

Following is the algorithm to update an element available at the Kth position of LA.

1. Start

2. Set LA[K-1] = ITEM

3. Stop

C Program for Updation:

```
#include <stdio.h>
main() {
int LA[] = {1,3,5,7,8};
int k = 3, n = 5, item = 10;
int i, j;

printf("The original array elements are :\n");

for(i = 0; i<n; i++) {
printf("LA[%d] = %d \n", i, LA[i]);
}

LA[k-1] = item;

printf("The array elements after updation :\n");

for(i = 0; i<n; i++) {
printf("LA[%d] = %d \n", i, LA[i]);
}
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

Output

The original array elements are :

LA[0] = 1

LA[1] = 3

LA[2] = 5

LA[3] = 7

LA[4] = 8

The array elements after updation :

LA[0] = 1

LA[1] = 3

LA[2] = 10

LA[3] = 7

LA[4] = 8

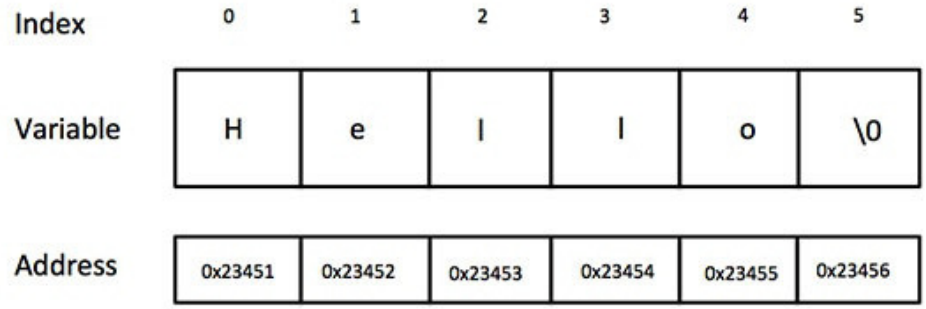
'C' में करैक्टर स्ट्रिंग:स्ट्रिंग्स वास्तव में एक आयामी करैक्टर ऐरे है। जिसके अन्त में Null करैक्टर '\0' होता है।

निम्नलिखित डिक्लेरेशन और इनिशलाईजेशन "Hello" शब्द से बनी स्ट्रिंग क्रियेट करता है। ऐरे के अन्त में Null करैक्टर जोडनें के लिए करैक्टर स्ट्रिंग की लम्बाई "Hello" शब्द में कुल अक्षरो की संख्या से एक अधिक है।

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

```
char greeting[] = "Hello";
```

इस स्ट्रिंग का मैमोरी में प्रदर्शन निम्नानुसार है।



वास्तव में आप स्ट्रिंग के अंत में Null करैक्टर को इन्सर्ट नहीं करते, सी कम्पायलर स्वचालित रूप से स्ट्रिंग के अंत में Null करैक्टर को इन्सर्ट कर देता है जब यह ऐरे को इनिशिलाईज करता है। उपरोक्त स्ट्रिंग को प्रिन्ट करना –

```
#include <stdio.h>
int main () {
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
printf("Greeting message: %s\n", greeting );
```

return 0;

}

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

Greeting message: Hello

सी भाषा में निम्नलिखित फंक्शन होते है।

क्र.स.	फंक्शन	उदेश्य
1	strcpy(s1, s2);	स्ट्रिंग s2को s1 में कोपी करता है।
2	strcat(s1, s2);	स्ट्रिंग s1के अन्त में s2 को जोड़ता है।
3	strlen(s1);	s1स्ट्रिंग की लम्बाई को बताता है।
4	strcmp(s1, s2);	यदि स्ट्रिंग s1एवंम s2समान है तो 0 रिर्टन करता है अन्यथा यदि s1<s2 तो 0 से कम, और यदि s1>s2 तो 0 से ज्यादा रिर्टन करता है।
5	strchr(s1, ch);	स्ट्रिंग s1में करेक्टर ch की पहली आवृत्ति का पोंइन्टर देता हैं।
6	strstr(s1, s2);	स्ट्रिंग s1में स्ट्रिंग s2 की पहली आवृत्ति का पोंइन्टर देता हैं।

निम्नलिखित उदाहरण उपरोक्त फंक्शनों में से कुछ का उपयोग दर्शाते है।

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

int main () {

char str1[12] = "Hello";
char str2[12] = "World";
char str3[12];
int  len ;

/* copy str1 into str3 */
strcpy(str3, str1);
printf("strcpy( str3, str1) :  %s\n", str3 );
```

```
/* concatenates str1 and str2 */
strcat( str1, str2);
printf("strcat( str1, str2):   %s\n", str1 );

/* total lenghth of str1 after concatenation */
len = strlen(str1);
printf("strlen(str1) :  %d\n", len );

return 0;
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
strcpy( str3, str1) :  Hello
strcat( str1, str2):   HelloWorld
strlen(str1) :  10
```

स्टैटिक और डायनेमिक मैमोरी अलोकेशन (आबंटन):डायनेमिक मैमोरी अलोकेशन रन टाइम पर किया जाता हैं। जबकि स्टैटिक मैमोरी अलोकेशन रन टाइम से पहले किया जाता है, लेकिन वेरिएबल के मानों को रन टाइम पर बदला जा सकता है

स्टैटिक मैमोरी अलोकेशन रनिंग टाइम में बचत करता है, लेकिन सभी मामलों में यह संभव नहीं हो सकता है। डायनेमिक मैमोरी अलोकेशन इसकी मैमोरी को हीप में सहेजता है और स्टैटिक मैमोरी अलोकेशन इसके डेटा को मैमोरी के डेटा सगमेंन्ट में सहेजता है।

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
//static allocation example using integer array.
int arr[5]; // static memory allocation memory allocated before execution, the size of array should be initialized
for ( int j = 0; j<5; j++) //Waste of memory can be occured.
{
printf("Enter number for Static Array %d: " ,j);
scanf("%d", &arr[j]);
}
```

```
printf("\nThe Static Array is: \n");
for ( int j = 0; j<5; j++)
{
printf("The value of %d is %d\n", j, arr[j]);
}

//dynamic allocation example using integer array
int* array;
int n, i;
printf("\n----------\n\nDynamic Allocation\n");
printf("Enter the number of elements: ");
scanf("%d", &n);
array = (int*) malloc(n*sizeof(int)); //memory is allocated during the execution of the program
//Less Memory space required.
for (i=0; i<n; i++) {
printf("Enter number %d: ", i);
scanf("%d", &array[i]);
}

printf("\nThe Dynamic Array is: \n");

for (i=0; i<n; i++) {
printf("The value of %d is %d\n", i, array[i]);
}
printf("Size= %d\n", i);

system("PAUSE");
return 0;
}
```

मैमोरी अलोकेशन के फंक्शनः सी प्रोग्रामिंग भाषा में मैमोरी अलोकेशन और प्रबन्धन के लिए कई फंक्शन होते है। यह फंक्शन <stdlib.h>हेडर फाईल में होते है।

क्र.स.	फंक्शन और विवरण
1	void *calloc(int num, int size); num तत्वों का एक ऐरे आवंटित करता है। जिसकी size बाइट में होगी।
2	void free(void *address); दिये गये पत्ते के मैमोरी ब्लोक को फ्री करता है।
3	void *malloc(int num); num तत्वों का एक ऐरे आवंटित करता है। जिसकी size बाइट में होगी एवंम् अनइनिशियलाईज होगी।
4	void *realloc(void *address, int newsize); दुबारा मैमोरी अलोकेशन के काम आता है।

निम्नलिखित प्रोग्राम डायनेमिक मैमोरी अलोकेशन का फंक्शन के साथ उदाहरण है।

1.

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

int main() {

char name[100];
char *description;

strcpy(name, "Zara Ali");

/* allocate memory dynamically */
description = malloc( 200 * sizeof(char) );

if( description == NULL ) {
fprintf(stderr, "Error - unable to allocate required memory\n");
}
else {
```

```
strcpy( description, "Zara ali a DPS student in class 10th");
}
printf("Name = %s\n", name );
printf("Description: %s\n", description );
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
Name = Zara Ali
Description: Zara ali a DPS student in class 10th
```

उपरोक्त प्रोग्राम को निम्नानुसार calloc(); फंक्शन का उपयोग करके भी लिखा जा सकता है।

```
calloc(200, sizeof(char));
```

आप किसी भी साईज की मैमोरी अलोकेट कर सकते है। और आपका इस पर पूर्ण नियंत्रण होता है। जबकि इसके विपरीत ऐरे में एक बार इसकी साईज को परिभाषित करने के बाद बदल नहीं सकते है।

2.

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

int main() {

char name[100];
char *description;

strcpy(name, "Angad");

/* allocate memory dynamically */
description = malloc( 30 * sizeof(char) );

if( description == NULL ) {
fprintf(stderr, "Error - unable to allocate required memory\n");
}
```

```
else {
strcpy( description, "Angad is a cute Boy.");
}

/* suppose you want to store bigger description */
description = realloc( description, 100 * sizeof(char) );

if( description == NULL ) {
fprintf(stderr, "Error - unable to allocate required memory\n");
}
else {
strcat( description, "He is in 1st Class");
}

printf("Name = %s\n", name );
printf("Description: %s\n", description );

/* release memory using free() function */
free(description);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

Name = Angad

Description: Angad is a cute Boy.He is in 1st Class.

'C' में पोंइन्टर **(Pointers)**:पोंइन्टरएक वेरिएबल है जिसका मान अन्य वेरिएबल का ऐड्रस (मैमोरी लोकेशन का ऐड्रस) होता है। किसी भी अन्य वेरिएबल और कॉन्सटेन्ट कि तरह पोंइन्टर को भी उपयोग में लेने से पहले इसे डिक्लेयर करना आवश्यक है। पोंइन्टर डिक्लेरेशन का साधारण रूप निम्नानुसार है।

type *var-name;

यहाँ type पोंइन्टर प्रकार है। यह सी के मान्य डेटा प्रकारों में से होना चाहिये और var-name पोंइन्टर वेरिएबल का नाम है। पोंइन्टर को डिक्लेयर करने के लिए उपयोग में लिया जाने वाला तारांकन ''*'' गुणा में उपयोग लिये जाने वाले तारांकन चिन्ह के समान ही है कुछ मान्य पोंइन्टर डिक्लेरेशन निम्नानुसार है

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

सभी पोंइन्टरों के मानों का डेटा प्रकार एक ही होता है, यह एक लोंग हेक्साडेसिमल नम्बर होता है और यह एक मैमोरी पत्ते को प्रदर्शित करता है। विभिन्न डेटा प्रकारों के पोंइन्टरों के बीच फर्क सिर्फ उनके द्वारा पोंइट किये गये वेरिएबल और कॉन्सटेन्ट के डेटा प्रकार में ही होता है।

निम्नलिखित उदाहरण पोंइन्टर वेरिएबल के उपयोग को प्रदर्शित करता है।

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

int  var = 20;   /* actual variable declaration */
int  *ip;        /* pointer variable declaration */

ip = &var;  /* store address of var in pointer variable*/

printf("Address of var variable: %x\n", &var  );

/* address stored in pointer variable */
printf("Address stored in ip variable: %x\n", ip );

/* access the value using the pointer */
printf("Value of *ip variable: %d\n", *ip );

return 0;
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
Address of var variable: bffd8b3c
Address stored in ip variable: bffd8b3c
Value of *ip variable: 20
```

NULL पोंइन्टरः

यदि आपके पास पोंइन्टर वेरिएबल को असाइन करने के लिए कोई सही पत्ता नहीं हो तब इसे NULL वैल्यू असाइन करनी चाहिऐ। यह वेरिएबल डिक्लेरेशन के समय पर किया जाता है। पोंइन्टर वेरिएबल जिसे NULL वैल्य असाइन की गई है उसे NULL पोंइन्टर कहते हैं।

NULL पोंइन्टर एक शून्य मान वाला कॉन्सटेन्ट होता है और यह कई मानक लाईब्रेरी में परिभाषित हैं

निम्नलिखित प्रोग्राम पर विचार करे।

```
#include <stdio.h>
int main () {
int  *ptr = NULL;
printf("The value of ptr is : %x\n", ptr  );
return 0;
}
```

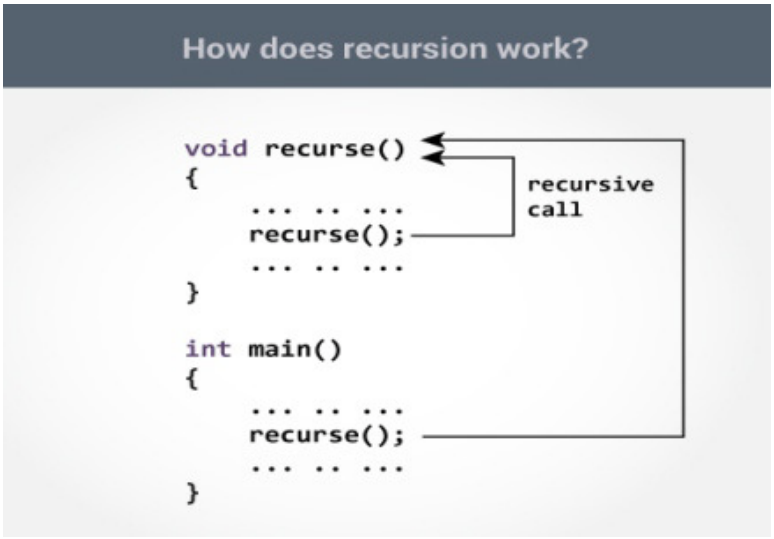
जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

The value of ptr is 0

पोंइन्टर को जाँचने के लिए if स्टेटमेंट का उपयोग निम्नानुसार किया जा सकता हैः

```
if(ptr)     /* succeeds if p is not null */
if(!ptr)    /* succeeds if p is null */
```

'C' मे Recursion(रिकर्शन):रिकर्शन स्व–समान (self-similar) तरीके से आइटमों को दोहराने की एक प्रक्रिया है। प्रोग्रामिंग भाषाओं में यदि एक प्रोग्राम आपको एक ही फंक्शन के अंदर उसी फंक्शन को कॉल करने की अनुमति देता है तब इसे रिकर्शन फंक्शन के रूप में जाना जाता है। अन्य शब्दो में जब एक फंक्शन अपने आप को ही कॉल करता है तो इसे रिकर्सिव फंक्शन कहते है।



रिकर्शन निम्नानुसार काम करता है:

```
void recurse()
{
... .. ...
recurse();
... .. ...
}

int main()
{
... .. ...
recurse();
... .. ...
}
```

रिकर्शन फंक्शन के लिए शर्तें:

1. सभीरिकर्शन फंक्शन में एक बेस मानदंड (Termination condition) होनी आवश्यक है और इसके लिए वह खुद को कॉल नहीं करना चाहिए।
2. जब भी एक फंक्शन खुद को कॉल करे तब यह बेस मानदंड के करीब आये।

रिकर्शन के उदाहरण निम्नानुसार है:

(क) फिबोनैकी सीरीज (Fibonacci Series)

(ख) बिनोमिअल कोफिसिएंट (Binomial coefficient)

(ग) GCD

(क) फिबोनैकी सीरीज

फिबोनैकी सीरीज कि संख्याऐ निम्नानुसार पूर्णांक अनुक्रम में हैं:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 ....

फिबोनैकी सीरीज में पहले दो नंबर 0 और 1 हैं और प्रत्येक बाद वाला नंबर पिछले दो नंबरों का योग है। गणितीय संदर्भ में, फिबोनैकी संख्याओं कि Nth टर्म निम्नानुसार रेकर्रेंस संबंध द्वारा परिभाषित है:

fibonacci(N) = फिबोनैकी संख्याओं कि Nth टर्म

fibonacci(N) = fibonacci(N - 1) + fibonacci(N - 2);

जहाँ पर, fibonacci(0) = 0 और fibonacci(1) = 1

निम्नलिखित प्रोग्राम फिबोनैकी संख्याओं कि Nth टर्म निकालने के लिए रिकर्शन का उपयोग करता है। Nth फिबोनैकी संख्या निकालने के लिए यह सर्वप्रथम (N-1)th और (N-2)th ज्ञात करता है और फिर दोनों का योग करता है।

रिकर्शन का उपयोग करके फिबोनैकी सीरीज को Nth टर्म तक प्रिंट करने के लिए सी प्रोग्रामः

निम्नलिखित प्रोग्राम, उपयोगकर्ता से स्केनफ फंक्शन का उपयोग करके इनपुट के रूप में फिबोनैकी सीरीज के पदों की संख्या को लेता है। इसमें ऊपर बातये अनुसार रिकर्शन का उपयोग करते हुए 'fibonacci' नामक यूजर परिभाषित फंक्शनहै जो इनपुट के रूप में एक पूर्णांक N लेता है और Nth फिबोनैकी संख्या लौटता है। जब पदों की संख्या <2 होगी तब रिकर्शन समाप्त हो जाएगा क्योंकि फिबोनैकी सीरीज के पहले दो क्रम 0 और 1 होते है।

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

int fibonacci(int term);
int main(){
int terms, counter;
printf("Enter number of terms in Fibonacci series: ");
scanf("%d", &terms);
/*
*  Nth term = (N-1)th therm + (N-2)th term;
*/
printf("Fibonacci series till %d terms\n", terms);
for(counter = 0; counter < terms; counter++){
printf("%d ", fibonacci(counter));
}
getch();
return 0;
}
/*
* Function to calculate Nth Fibonacci number
* fibonacci(N) = fibonacci(N - 1) + fibonacci(N - 2);
*/
int fibonacci(int term){
/* Exit condition of recursion*/
if(term < 2)
return term;
return fibonacci(term - 1) + fibonacci(term - 2);
}
```

Program Output

Enter number of terms in Fibonacci series: 9
Fibonacci series till 9 terms
0 1 1 2 3 5 8 13 21

(ख) बिनोमिअल कोफिसिएंट का प्रोग्रामः

```
#include<stdio.h>
int fact(int);
void main()
{
int n,r,f;
printf("enter value for n & r\n");
scanf("%d%d",&n,&r);
if(n<r)
printf("invalid input");
else f=fact(n)/(fact(n-r)*fact(r));
printf("binomial coefficient=%d",f);
}
int fact(int x)
{
if(x>1)
returnx*fact(x-1);
else return 1;
}
```

The sum of the 10 binomial coefficients of the form C 9 k?
45

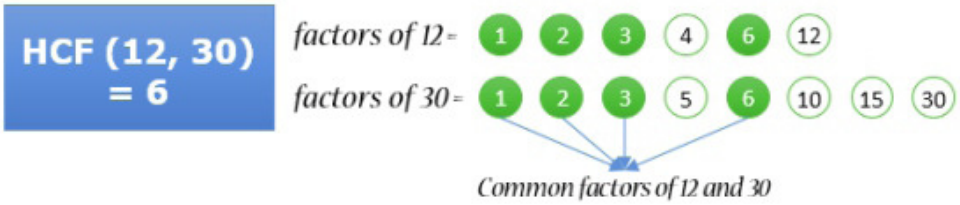
(ग) दो नंबरो का **GCD** निकालना**:**

Input first number: 10

Input second number: 15

Output GCD: 5

Logic to find GCD using recursion



GCD निकालना के लिए इयूक्लिडियन एल्गोरिथ्मः

Begin:
function gcd(a, b)
If (b = 0) then
return a
End if
Else
return gcd(b, a mod b);
End if
End function
End

Program to find GCD using recursion:

C program to find GCD (HCF) of two numbers using recursion:

```
#include <stdio.h>

/* Function declaration */
int gcd(int a, int b);

int main()
{
int num1, num2, hcf;

/* Reads two numbers from user */
printf("Enter any two numbers to find GCD: ");
scanf("%d%d", &num1, &num2);

hcf = gcd(num1, num2);
```

```
printf("GCD of %d and %d = %d\n", num1, num2, hcf);

return 0;
}
```

Recursive approach of euclidean algorithm to find GCD of two numbers:

```
int gcd(int a, int b)
{
if(b == 0)
return a;
else
return gcd(b, a%b);
}
```

Output:

```
Enter any two numbers to find GCD: 12
30
GCD of 12 and 30 = 6
```

महत्वपूर्ण बिंदु

- ऐरेसमरूप डेटा तत्वों के परिमित क्रमो का एक संग्रह है जोकि क्रमिक मैमोरी स्थानों में संग्रहित होते है।
- मैमोरी में एक 2–डी ऐरे के तत्वों को संग्रह करते समय इन्हें क्रमिक मैमोरी लोकेशन आवंटित किये जाते है।
- एक डाटा स्ट्रक्चर में मौजूद सभी डेटा तत्वों के प्रसंस्करण (प्रोसेसिंग) को ट्रर्वसिंग कहते है।
- पोंइन्टरएक वेरिएबल है जिसका मान अन्य वेरिएबल का ऐड्रस (मैमोरी लोकेशन का ऐड्रस) होता है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्र1 लीनियर सर्च में वर्स्ट केस कब होता है?

(अ) आइटम ऐरे के बीच में हो

(ब) आइटम ऐरे में बिल्कुल भी नहीं हो

(स) आइटम ऐरे में पीछे हो

(द) आइटम ऐरे में पीछे हो या बिल्कुल नहीं हो

प्र2 लीनियर सर्च एल्गोरिथ्म की जटिलता है?

(अ) $O(n^2)$ (ब) $O(logn)$

(स) $O(nlogn)$ (द) $O(n+1)$

प्र3 लीनियर सर्च एल्गोरिथ्म में औसत मामले कब होते हैं

(अ) जब आइटम ऐरे के बीच में कहीं हो

(ब) जब आइटम ऐरे में बिल्कुल भी नहीं हो

(स) जब आइटम ऐरे के पिछले हिस्से में हो

(द) जब आइटम ऐरे के पिछले हिस्से में हो या नहीं हो

प्र4 दिये गये मान से किसी तत्व के स्थान को ढूँढना है:

(अ) ट्रर्वस (ब) सर्च

(स) सॉट (द) इनमें से कोई भी नहीं

प्र5 निम्न में से कौन सा मामला जटिलता सिद्धांत में मौजूद नहीं है

(अ) सबसे अच्छा मामला (ब) सबसे खराब मामला

(स) औसत के मामले (द) अशक्त मामले

लघुत्तरात्मक प्रश्न

प्र1 बाइनरी खोज की समय जटिलता क्या है?

प्र2 ऐरे से आपका क्या मतलब है?

प्र3 स्ट्रिंग क्या है?

प्र4 सूचक से आपका क्या मतलब है?

प्र5 गतिशील स्मृति आबंटन क्या है?

निबंधात्मक प्रश्न

प्र1 उदाहरण के साथ दो आयामी ऐरे समझाओ।

प्र2 विस्तार से malloc फंक्शन को समझाओं।

प्र3 रिकर्शन के लिए कोनसा डेटा स्ट्रक्चर का उपयोग किया जाता हैं?

प्र4 बाइनरी सर्च,लीनियर सर्च से क्यो बेहतर है?

प्र5 Character स्ट्रिंग को समझाओ।

उत्तरमाला

उत्तर 1: द उत्तर 2: द उत्तर 3: अ

उत्तर 4: ब उत्तर 5: द

अध्याय 3

सॉर्टिंग(Sorting)

सॉर्टिंग (Sorting) एक विशेष स्वरूप में डेटा को व्यवस्थित करने को संदर्भित करता है। सॉर्टिंर्ग एल्गोरिथ्म डेटा को एक विशेष क्रम में व्यवस्थित करने का तरीका निर्दिष्ट करती है। सबसे आम क्रम संख्यात्मक या वर्णानुक्रम हैं। यदि डेटा एक क्रमबद्ध तरीके से संग्रहित किया गया है तो सॉर्टिंग का सर्वाधिक महत्व डाटा सर्च को आसान बनाने में है। सॉर्टिंग डेटा को ओर अधिक पठनीय प्रारूप में प्रदर्शित करने के लिए भी प्रयोग कि जाती है। वास्तविक जीवन में सॉर्टिंग के कुछ उदाहरण हैं:

टेलीफोन निर्देशिका–

टेलीफोन निर्देशिका, लोगों के टेलीफोन नंबरो को उनके नाम के अनुसार क्रमबद्ध करके संग्रहीत करती है ताकी नामों को आसानी से सर्च किया जा सकता है।

शब्दकोश– शब्दकोश मे शब्द वर्णमाला के क्रम मे संग्रहीत किये जाते है इसलिए किसी भी शब्द को सर्च करना आसान हो जाता है।

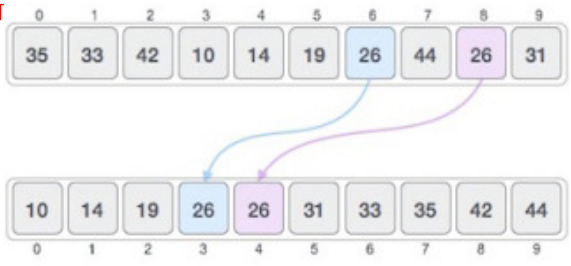
इन–प्लेस सॉर्टिंग और नाट–इन–प्लेस सॉर्टिंग(In-place Sorting and Not-in-place Sorting):

सॉर्टिंग एल्गोरिथ्म को तुलना और कुछ डेटा तत्वों के अस्थायी भंडारण के लिए कुछ अतिरिक्त स्थान की आवश्यकता हो सकती है। इन–प्लेस सॉर्टिंग एल्गोरिथ्म को किसी भी अतिरिक्त जगह की आवश्यकता नहीं होती है और इसलिए इन्हे सॉर्टिंग इन–प्लेस कहा जाता है, उदाहरण के लिए, ऐरे के भीतर ही सॉर्टिंग। बबल सॉर्ट इन–प्लेस सॉर्टिंग का एक उदाहरण है।

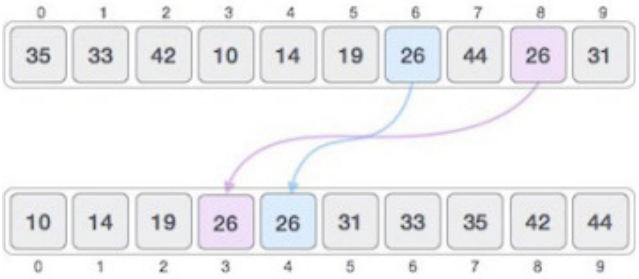
हालांकि, कुछ सॉर्टिंग एल्गोरिथ्म में, प्रोग्राम को स्पेस की आवश्यकता है जोकि तत्वो, जिन्हे सॉर्ट करना है के बराबर या उनसे अधिक हो सकती है और इसलिए इन्हे नाट–इन–प्लेस सॉर्टिंग कहा जाता है। मर्ज–सॉर्ट नाट–इन–प्लेस सॉर्टिंग का एक उदाहरण है।

स्टेबल और अनस्टेबल सॉर्टिंग (Stable and Unstable Sorting):

सॉर्टिंर्ग एल्गोरिथ्म, तत्वो को सॉर्ट करने के बाद, एक जैसे तत्वो के क्रम जिसमे वो प्रकट होते है को परिवर्तित नहीं करती है उनको स्टेबल सॉर्टिंग कहा जाता है।



सॉर्टिंग एल्गोरिथ्म, तत्वो को सॉर्ट करने के बाद, एक जैसे तत्वो के क्रम जिसमे वो प्रकट होते है को परिवर्तित करती है उनको अनस्टेबल सॉर्टिंग कहा जाता है।



एक एल्गोरिथ्म की स्टेब्लिटी (Stability) मायने रखती है जब हम मूल तत्वों का क्रम बनाए रखना चाहते हैं उदाहरण के लिए एक टपल में।

अडप्टिव और नॉन–अडप्टिव सॉर्टिंग एल्गोरिथ्म(Adaptive and Non adaptive Sorting) :

यदि सॉर्टिंग एल्गोरिथ्म, सॉर्ट करने वाली लिस्ट में पहले से ही सॉर्टेड तत्वों का लाभ लेती है तब उसे अडप्टिव कहा जाता है। अर्थात सॉर्टिंग के दौरान यदि स्रोत (source) सूची में पहले से ही कुछ तत्व सॉर्टेड है तब अडप्टिव एल्गोरिथ्म इसे ध्यान में रखते हुए उनका क्रम पुनः नहीं बदलती।

एक नॉन–अडप्टिव सॉर्टिंग एल्गोरिथ्म सूची में पहले से ही सॉर्टेड तत्वो को ध्यान में नही रखती। वे तत्व सॉर्टेड है या नही की पुष्टि करने के लिए हर एक तत्व के क्रम को बदलती हैं।

महत्वपूर्ण शर्तें

सॉर्टिंग तकनीको पर चर्चा के दौरान आम तौर कुछ शब्दावली का प्रयोग किया जाता है, यहाँ उनका एक संक्षिप्त परिचय हैं:

बढता क्रम (Increasing Order):

मानों का एक अनुक्रम बढते हुए क्रम में कहा जाता है, यदि बाद का तत्व अपने पिछले वाले तत्व से अधिक है। उदाहरण के लिए, 1, 3, 4, 6, 8, 9, बढते क्रम में हैं, क्योंकि यहा हर अगला तत्व अपने पिछले वाले तत्व से अधिक है।

घटता क्रम (Decreasing Order):

मानों का एक अनुक्रम घटते हुए क्रम में कहा जाता है, यदि बाद का तत्व अपने पिछले वाले तत्व से कम है। उदाहरण के लिए, 9, 8, 6, 4, 3, 1, घटते क्रम में हैं क्योंकि यहा हर अगला तत्व अपने पिछले तत्व से कम है।

गैर–बढ़ता क्रम (Non- Increasing Order):

मानों का एक अनुक्रम गैर–बढ़ते हुए क्रम में कहा जाता है, यदि बाद का तत्व अपने पिछले वाले तत्व से कम या उसके बराबर है। यह क्रम तब होता है जब अनुक्रम में डुप्लिकेट मान हो। उदाहरण के लिए, 9, 8, 6, 3, 3, 1, गैर बढ़ते क्रम में हैं क्यों यहा हर अगला तत्व अपने पिछले तत्व से कम या उसके बराबर (3 के मामले में) है।

गैर–घटता क्रम (Non-Decreasing Order):

मानों का एक अनुक्रम गैर–घटते हुए क्रम में कहा जाता है, यदि बाद का तत्व अपने पिछले वाले तत्व से अधिक या उसके बराबर है। यह क्रम तब होता है जब अनुक्रम में डुप्लिकेट मान हो। उदाहरण के लिए, 1, 3, 3, 6, 8, 9, गैर–घटते क्रम में हैं क्यों यहा हर अगला तत्व अपने पिछले तत्व से अधिक या उसके बराबर (3 के मामले में) है।

बबल (Bubble) सॉर्ट:

बबल सॉर्ट एक साधारण सॉर्टिंग एल्गोरिथ्म है। यह सॉर्टिंग एल्गोरिथ्म, तुलना–आधारित एल्गोरिथ्म है जिसमें सन्निकट तत्वों के प्रत्येक जोड़े की तुलना की जाती है और अगर वे क्रम में नहीं है तब तत्वों को बदला जाता हैं। इस एल्गोरिथ्म कि औसत (average) और सबसे खराब (worst case) स्थिति मे कॉम्प्लेक्सिटी $O(n^2)$ है जहाँ n सॉर्ट किये जाने वाले तत्वों की संख्या हैं और इसलिए यह एल्गोरिथ्म बड़े डेटा सेट के लिए उपयुक्त नहीं है।

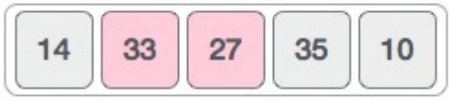
बबल सॉर्टिंग कैसे काम करती है?

हम उदाहरण के लिए एक अनसोर्टेड ऐरे ले रहे है। बबल सॉर्ट $O(n^2)$ समय लेती है, इसलिए हम इसे छोटा और सटीक रख रहे हैं।

बबल सॉर्ट, सबसे पहले दो तत्वों के साथ शुरू होती है, कोनसा बडा है यह जाँच करने के लिए उनकी तुलना करती है।

इस मामले में, 33 मान 14 से अधिक है, इसलिए यह पहले से सोर्टेड है। आगे हम 27 से 33 की तुलना करते है।

हम पाते हैं कि 27, 33 से छोटा है और इन दोनो मानो को बदली किया जाना चाहिए।



नई ऐरे इस तरह दिखनी चाहिए –

आगे हम 33 और 35 की तुलना मे पाते हैं कि दोनों पहले से ही सोर्टिड स्थितियों में हैं।

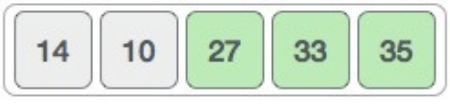
फिर हम अगले दो मानों, 35 और 10 को देखते है।

हम जानते हैं कि 10, 35 से छोटा है इसलिए वे सोर्टेड नहीं हैं।

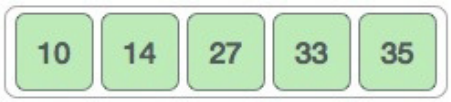
हम इन मानों को स्वैप करते है। हम पाते हैं कि हम ऐरे के अंत तक पहुँच चुके हैं। एक पुनरावृत्ति (iteration) के बाद, ऐरे इस तरह दिखना चाहिए –

अब हम दिखा रहे हैं की एक ऐरे प्रत्येक पुनरावृत्ति के बाद किस तरह दिखना चाहिए। दूसरी पुनरावृत्ति के बाद, यह इस तरह दिखना चाहिए –

ध्यान दें कि प्रत्येक पुनरावृत्ति के बाद, ऐरे के अंत में कम से कम एक मान चलता जाता है।



और जब किसी स्वैप की आवश्यकता नहीं रहती तब बबल सॉर्ट यह जान जाता है कि ऐरे पूरी तरह से सॉर्ट हो गया है।



Algorithm:

हम यहा यह मान रहे कि तत्वों की लिस्ट एक ऐरे मे है और स्वैप फंक्शन ऐरे तत्वों को स्वैप करता है।

```
BubbleSort
for all elements of list
if list[i] > list[i+1]
swap(list[i], list[i+1])
end if
end for

return list

end BubbleSort
```

बबल सॉट के लिए **C** प्रोग्राम**:**

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

#define MAX 10

int list[MAX] = {1,8,4,6,0,3,5,2,7,9};

void display() {
int i;
printf("[");
// navigate through all items
for(i = 0; i < MAX; i++) {
printf("%d ",list[i]);
}
```

```
printf("]\n");
}

void bubbleSort() {
int temp;
int i,j;

bool swapped = false;

// loop through all numbers
for(i = 0; i < MAX-1; i++) {
swapped = false;

// loop through numbers falling ahead
for(j = 0; j < MAX-1-i; j++) {
printf("     Items compared: [ %d, %d ] ", list[j],list[j+1]);

// check if next number is lesser than current no
//   swap the numbers.
//  (Bubble up the highest number)

if(list[j] > list[j+1]) {
temp = list[j];
list[j] = list[j+1];
list[j+1] = temp;

swapped = true;
printf(" => swapped [%d, %d]\n",list[j],list[j+1]);
}else {
printf(" => not swapped\n");
}
```

```
}

// if no number was swapped that means
//   array is sorted now, break the loop.
if(!swapped) {
break;
}

printf("Iteration %d#: ",(i+1));
display();
}

}

main() {
printf("Input Array: ");
display();
printf("\n");

bubbleSort();
printf("\nOutput Array: ");
display();
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
Output
Input Array: [1 8 4 6 0 3 5 2 7 9 ]
Items compared: [ 1, 8 ]  => not swapped
Items compared: [ 8, 4 ]  => swapped [4, 8]
Items compared: [ 8, 6 ]  => swapped [6, 8]
Items compared: [ 8, 0 ]  => swapped [0, 8]
```

```
Items compared: [ 8, 3 ]  => swapped [3, 8]
Items compared: [ 8, 5 ]  => swapped [5, 8]
Items compared: [ 8, 2 ]  => swapped [2, 8]
Items compared: [ 8, 7 ]  => swapped [7, 8]
Items compared: [ 8, 9 ]  => not swapped
Iteration 1#: [1 4 6 0 3 5 2 7 8 9 ]
Items compared: [ 1, 4 ]  => not swapped
Items compared: [ 4, 6 ]  => not swapped
Items compared: [ 6, 0 ]  => swapped [0, 6]
Items compared: [ 6, 3 ]  => swapped [3, 6]
Items compared: [ 6, 5 ]  => swapped [5, 6]
Items compared: [ 6, 2 ]  => swapped [2, 6]
Items compared: [ 6, 7 ]  => not swapped
Items compared: [ 7, 8 ]  => not swapped
Iteration 2#: [1 4 0 3 5 2 6 7 8 9 ]
Items compared: [ 1, 4 ]  => not swapped
Items compared: [ 4, 0 ]  => swapped [0, 4]
Items compared: [ 4, 3 ]  => swapped [3, 4]
Items compared: [ 4, 5 ]  => not swapped
Items compared: [ 5, 2 ]  => swapped [2, 5]
Items compared: [ 5, 6 ]  => not swapped
Items compared: [ 6, 7 ]  => not swapped
Iteration 3#: [1 0 3 4 2 5 6 7 8 9 ]
Items compared: [ 1, 0 ]  => swapped [0, 1]
Items compared: [ 1, 3 ]  => not swapped
Items compared: [ 3, 4 ]  => not swapped
Items compared: [ 4, 2 ]  => swapped [2, 4]
Items compared: [ 4, 5 ]  => not swapped
Items compared: [ 5, 6 ]  => not swapped
```

Iteration 4#: [0 1 3 2 4 5 6 7 8 9]

Items compared: [0, 1] => not swapped

Items compared: [1, 3] => not swapped

Items compared: [3, 2] => swapped [2, 3]

Items compared: [3, 4] => not swapped

Items compared: [4, 5] => not swapped

Iteration 5#: [0 1 2 3 4 5 6 7 8 9]

Items compared: [0, 1] => not swapped

Items compared: [1, 2] => not swapped

Items compared: [2, 3] => not swapped

Items compared: [3, 4] => not swapped

Output Array: [0 1 2 3 4 5 6 7 8 9]

चयन (Selection) सॉर्टिंग:चयन सॉर्ट एक सरल सॉर्टिंग एल्गोरिथ्म है।

यह एक इन–प्लेस तुलना–आधारित सॉर्टिंग एल्गोरिथ्म है इसमें सूची दो भागों में विभाजित होती है, सॉर्ट किया हुआ भाग बाईं ओर तथा सॉर्ट न किया हुआ भाग दायीं ओर रहता है। शुरू में, सॉर्ट किया गया हुआ भाग खाली रहता है और संपूर्ण सूची सॉर्ट न किये हुआ भाग में होती है। अवर्गीकृत (अनसोरटेड) ऐरे में से सबसे छोटा तत्व का चयन किया जाता है और इसे ऐरे में सबसे बाएँ तत्व के साथ बदली किया जाता है, और वह तत्व सॉर्ट किये हुआ ऐरे का एक हिस्सा बन जाता है। यह प्रक्रिया अवर्गीकृत ऐरे की सीमा को एक तत्व दायीं ओर बढाती चली जाती है। इस एल्गोरिथ्म कि औसत (average) और सबसे खराब (worst case) स्थिति मे कॉम्प्लेक्सिटी $O(n^2)$ है, जहाँ n सॉर्ट किये जाने वाले तत्वों की संख्या हैं और इसलिए यह एल्गोरिथ्म बड़े डेटा सेट के लिए उपयुक्त नहीं है।

चयन सॉर्टिंग कैसे काम करती है?

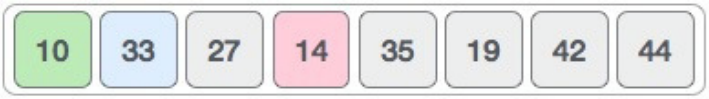
एक उदाहरण के रूप में निम्नलिखित दर्शाया हुए ऐरे पर विचार करें:

क्रमबद्ध लिस्ट में प्रथम स्थान के लिए, पूरी लिस्ट की क्रमिक रूप से जांच होती है। पहली स्थिति जहां 14 को वर्तमान में संग्रहित करना है, हम पूरी लिस्ट को सर्च करते है और पाते हैं कि 10 निम्नतम मान है।

इसलिए हम 14 को 10 से बदलते है। एक पुनरावृत्ति के बाद 10 जो कि लिस्ट में न्यूनतम मान है, सॉर्टेड लिस्ट में पहली स्थिति में दिखाई देता है।

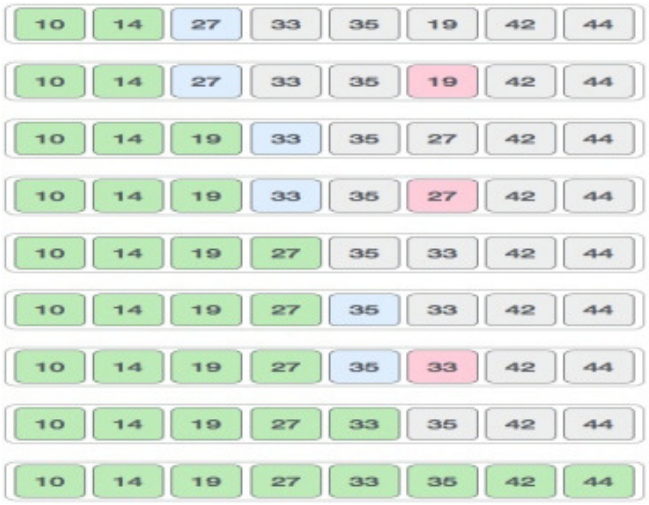
दूसरे स्थान के लिए जहां 33 है, हम एक रेखीय ढंग से बाकी लिस्ट की स्कैनिंग शुरू करते हैं।

हम पाते हैं कि 14 लिस्ट में दूसरा सबसे कम मान है और यह दूसरे स्थान पर होना चाहिए। हम इन मानों को स्वैप करते हैं।



दो पुनरावृत्तियों के बाद, दो कम से कम मान एक क्रमबद्ध ढंग से शुरुआत में आ जाते हैं।

यही प्रक्रिया ऐरे में बाकी के आइटम के लिए लागू कि जाती है। पूरी सॉर्टिंग प्रक्रिया का एक सचित्र चित्रण निम्नानुसार है:



एल्गोरिथ्म:

Step 1 – Set MIN to location 0

Step 2 – Search the minimum element in the list

Step 3 – Swap with value at location MIN

Step 4 – Increment MIN to point to next element

Step 5 – Repeat until list is sorted

चयन सॉट के लिए **C** प्रोग्राम**:**

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 7
int intArray[MAX] = {4,6,3,2,1,9,7};
void printline(int count) {
int i;
for(i = 0;i <count-1;i++) {
printf("=");
}
printf("=\n");
}
void display() {
int i;
printf("[");
// navigate through all items
for(i = 0;i<MAX;i++) {
printf("%d ", intArray[i]);
}
printf("]\n");
}
void selectionSort() {
int indexMin,i,j;
// loop through all numbers
```

```
for(i = 0; i < MAX-1; i++) {
// set current element as minimum
indexMin = i;
// check the element to be minimum
for(j = i+1;j<MAX;j++) {
if(intArray[j] < intArray[indexMin]) {
indexMin = j;
}
}
if(indexMin != i) {
printf("Items swapped: [ %d, %d ]\n" , intArray[i], intArray[indexMin]);
// swap the numbers
int temp = intArray[indexMin];
intArray[indexMin] = intArray[i];
intArray[i] = temp;
}
printf("Iteration %d#:",(i+1));
display();
}
}
main() {
printf("Input Array: ");
display();
printline(50);
selectionSort();
printf("Output Array: ");
display();
printline(50);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

Output

Input Array: [4 6 3 2 1 9 7]

==========

Items swapped: [4, 1]

Iteration 1#:[1 6 3 2 4 9 7]

Items swapped: [6, 2]

Iteration 2#:[1 2 3 6 4 9 7]

Iteration 3#:[1 2 3 6 4 9 7]

Items swapped: [6, 4]

Iteration 4#:[1 2 3 4 6 9 7]

Iteration 5#:[1 2 3 4 6 9 7]

Items swapped: [9, 7]

Iteration 6#:[1 2 3 4 6 7 9]

Output Array: [1 2 3 4 6 7 9]

मर्ज **(Merge)** सॉर्टिंग**:** मर्ज सॉर्ट डिवाइड (विभजित) एंड कॉन्कर (जीत) पर आधारित एक सॉर्टिंग तकनीक है। इसकी सबसे खराब मामले में (worst-case) कॉम्प्लेक्सिटी O(nlogn) होने के कारण यह सबसे अच्छी एल्गोरिथ्म में से एक है। मर्ज सॉर्ट पहले ऐरे को दो बराबर हिस्सों में तोडती है और फिर उन्हें एक क्रमबद्ध ढंग से जोड़ती है।

मर्ज सॉर्ट कैसे काम करती है?

मर्ज सॉर्ट को समझने के लिए हम एक निम्नलिखित अवर्गीकृत ऐरे लेते है

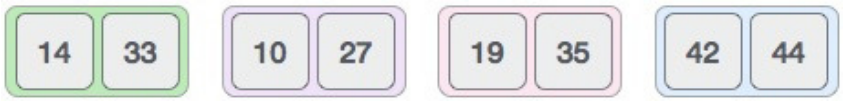
हम जानते हैं कि मर्ज सॉर्ट पहले पूरी ऐरे को पुनरावृत्तीय तरीके से बराबर हिस्सों में बांटती है जब तक कि परमाणु (atomic)या अविभाज्य मान प्राप्त नहीं हो जाते हैं। हम यहाँ देखते है कि 8 मानो की एक ऐरे 4 आकार की दो ऐरे में बंट गयी है।

यह मूल मानों की उपस्थिति के अनुक्रम को नहीं बदलता है। अब हम इन दो ऐरे को हिस्सों में विभाजित करते है।

हम आगे इन ऐरे को ओर विभाजित करते है और हमे परमाणु मान प्राप्त होते है जिनको ओर अधिक विभाजित नही किया जा सकता।

अब, हम उन्हें ठीक उसी तरीके से सम्मलित करते है जैसे उन्हें तोड़ा था। कृपया इन सूचियों को दिए गए रंग कोड पर ध्यान दें।

हम पहले प्रत्येक लिस्ट के तत्व की तुलना करते है और फिर एक क्रमबद्ध ढंग से उन्हें एक दूसरी लिस्ट में सम्मलित करते है। हम जानते हैं कि 14 और 33 सॉर्टेड स्थिति में ही हैं। हम 27 और 10 की तुलना करते है और 2 मानों की लक्ष्य लिस्ट में हम पहले 10 को डालते है और उसके पीछे 27 को। हम 19 और 35 का क्रम बदलते है जबकि 42 और 44 को क्रमिक रूप से रखा जाता है।



संयोजन चरण के अगले पुनरावृत्ति में, हम दो डेटा मानों की लिस्ट की तुलना करते है, और उन्हें एक सॉर्टेड क्रम में डेटा मानों की लिस्ट में मर्ज केर देते है।

अंतिम विलय के बाद, लिस्ट इस तरह दिखेगी –

Algorithm:

मर्ज सॉर्ट लिस्ट को पुनरावृत्तीय तरीके से बराबर हिस्सों में बांटती है जब तक कि उसे ओर अधिक विभाजित नही किया जा सकता। परिभाषा के अनुसार, अगर लिस्ट में केवल एक ही तत्व है, तो यह लिस्ट सॉर्टेड है। फिर, मर्ज सॉर्ट छोटी सॉर्टेड सूचियों को इस तरह सम्मलित करती है ताकि नयी बनने वाली सूचि भी सॉर्टेड ही रहे।

Step 1 – अगर लिस्ट में केवल एक ही तत्व है, तो यह लिस्ट सॉर्टेड है

Step 2 – लस्ट को पुनरावृत्तीय तरीके से दो बराबर हिस्सों में बांटना जब तक कि उसे ओर अधिक विभाजित नही किया जा सकता।

Step 3 – छोटी सॉर्टेड सूचियों को इस तरह सम्मलित करना ताकि नयी बनने वाली सूचि भी सॉर्टेड ही रहे।

मर्ज सॉर्ट के लिए C प्रोग्राम:

```
#include <stdio.h>
#define max 10
int a[10] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44 };
int b[10];
void merging(int low, int mid, int high) {
int l1, l2, i;
for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
if(a[l1] <= a[l2])
b[i] = a[l1++];
else
b[i] = a[l2++];
}
while(l1 <= mid)
b[i++] = a[l1++];

while(l2 <= high)
b[i++] = a[l2++];

for(i = low; i <= high; i++)
a[i] = b[i];
}
void sort(int low, int high) {
int mid;
if(low < high) {
mid = (low + high) / 2;
sort(low, mid);
sort(mid+1, high);
merging(low, mid, high);
} else {
return;
```

```
}
}
int main() {
int i;
printf("List before sorting\n");
for(i = 0; i <= max; i++)
printf("%d ", a[i]);
sort(0, max);
printf("\nList after sorting\n");
for(i = 0; i <= max; i++)
printf("%d ", a[i]);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

List before sorting

10 14 19 26 27 31 33 35 42 44 0

List after sorting

0 10 14 19 26 27 31 33 35 42 44

निवेशन (Insertion) सॉर्टिंग:यह एक इन–प्लेस तुलना–आधारित सॉर्टिंग एल्गोरिथ्म है। इसमें एक उप–लिस्ट बनाये रखी जाती है जो हमेशा सॉर्टेड रहती है। उदाहरण के लिए, एक ऐरे के निचले हिस्से को सॉर्टेड बनाए रखा जाता है। एक तत्व जिसे इस सॉर्टेड उप–लिस्ट में सम्मलित किया जाना है, इसकी उचित जगह सर्च करके फिर इसे वहाँ डाला जाता है। इसलिए इसका नाम, निवेशन सॉर्ट है।

ऐरे को क्रमिक रूप से सर्च किया जाता है और अवर्गीकृत आइटम को स्थानांतरित किया जाता हैं और उन्हें सॉर्टेड उप–लिस्ट में डाला (एक ही ऐरे में) जाता है। इस एल्गोरिथ्म कि औसत (average) और सबसे खराब (worst case) स्थिति मे कॉम्प्लेक्सिटी $O(n^2)$ है, जहाँ n सॉर्ट किये जाने वाले तत्वों की संख्या हैं और इसलिए यह एल्गोरिथ्म बड़े डेटा सेट के लिए उपयुक्त नहीं है।

निवेशन सॉर्टिंग कैसे काम करती है?

हम उदाहरण के लिए एक अवर्गीकृत ऐरे ले रहे है।

निवेशन सॉर्टिंग पहले दो तत्वों की तुलना करती है।

यहा 14 और 33 दोनों ही आरोही क्रम में पहले से ही हैं। अभी के लिए, 14 सॉर्टेड उप–लिस्ट में है।

निवेशन सॉर्टिंग आगे 33 की 27 से तुलना करती है।

और पाती है कि 33 सही स्थिति में नहीं है।

यह 33 को 27 के साथ स्वैप करती है।

यह सॉर्टेड उप–लिस्ट के सभी तत्वों के साथ की जाँच करती है। यहाँ हम देखते सॉर्टेड उप–लिस्ट में केवल एक तत्व 14 है और 27, 14 से अधिक है, इसलिए सॉर्टेड उप–लिस्ट की अदला–बदली के बाद भी यह सॉर्टेड रहेगी।

अब तक सॉर्टेड उप–लिस्ट में 14 और 27 है। इसके बाद, यह 33 की 10 से तुलना करती है।

यह मान एक सॉर्ट क्रम में नहीं हैं।

इसलिए हम उन्हें स्वैपकरते है।

हालांकि, स्वैपिंग 27 और 10 को अवर्गीकृत बनाता है।

इसलिए, हम उन्हें भी स्वैप करते है।

फिर हम 14 और 10 को अवर्गीकृत क्रम में पाते हैं।

हम उन्हें फिर से स्वैप करते है। तीसरी पुनरावृत्ति के अंत तक सॉर्टेड उप लिस्ट मे 4 मान हो जाते है।

इस प्रक्रिया तब तक जारी रहती है जब तक सभी अवर्गीकृत मान सॉर्टेड उप–लिस्ट में शामिल नहीं हो जाते है।

Algorithm:

अब हम यह सॉर्टिंग तकनीक कैसे काम करती है कि एक बड़ी तस्वीर जानते है, इसलिए हम निवेशन (Insertion) सॉर्टिंग के सरल स्टेप्स प्राप्त कर सकते हैं।

चरण 1 –अगर लिस्ट में केवल एक ही तत्व है, तो यह लिस्ट सॉर्टेड है।

चरण 2– अगला तत्व लेवे।

चरण 3–सॉर्टेड उप सूची के सभी तत्वों के साथ की तुलना करें।

चरण 4 – सॉर्टेड उप सूची के उन सभी तत्वों को शिफ्ट करे जो सॉर्ट किये जाने वाले मान से अधिक है।

चरण 5– मान सम्मिलित करें।

चरण 6– दोहराएँ जब तक सूची सॉर्ट नही हो जाता है।

निवेशन (Insertion)सॉर्टिंग के लिए C प्रोग्राम:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 7
int intArray[MAX] = {4,6,3,2,1,9,7};
```

```
void printline(int count) {
int i;
for(i = 0;i <count-1;i++) {
printf("=");
}
printf("=\n");
}
void display() {
int i;
printf("[");
// navigate through all items
for(i = 0;i<MAX;i++) {
printf("%d ",intArray[i]);
}
printf("]\n");
}
void insertionSort() {
int valueToInsert;
int holePosition;
int i;
// loop through all numbers
for(i = 1; i < MAX; i++) {
// select a value to be inserted.
valueToInsert = intArray[i];
// select the hole position where number is to be inserted
holePosition = i;
// check if previous no. is larger than value to be inserted
while (holePosition > 0 && intArray[holePosition-1] > valueToInsert) {
intArray[holePosition] = intArray[holePosition-1];
holePosition--;
printf(" item moved : %d\n" , intArray[holePosition]);
}
if(holePosition != i) {
printf(" item inserted : %d, at position : %d\n" ,
valueToInsert,holePosition);
// insert the number at hole position
intArray[holePosition] = valueToInsert;
}
printf("Iteration %d#:",i);
```

```
display();
}
}
main() {
printf("Input Array: ");
display();
printline(50);
insertionSort();
printf("Output Array: ");
display();
printline(50);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
Input Array: [4 6 3 2 1 9 7 ]
==================================================
Iteration 1#:[4 6 3 2 1 9 7 ]
item moved : 6
item moved : 4
item inserted : 3, at position : 0
Iteration 2#:[3 4 6 2 1 9 7 ]
item moved : 6
item moved : 4
item moved : 3
item inserted : 2, at position : 0
Iteration 3#:[2 3 4 6 1 9 7 ]
item moved : 6
item moved : 4
item moved : 3
item moved : 2
item inserted : 1, at position : 0
Iteration 4#:[1 2 3 4 6 9 7 ]
Iteration 5#:[1 2 3 4 6 9 7 ]
```

item moved : 9
item inserted : 7, at position : 5
Iteration 6#:[1 2 3 4 6 7 9]
Output Array: [1 2 3 4 6 7 9]

त्वरित (Quick) सॉर्टिंग: त्वरित एक प्रकार की अत्यंत कुशल सॉर्टिंग एल्गोरिथ्म है और डेटा के ऐरे को छोटे ऐरे में विभाजन करने पर आधारित है। एक बड़ा ऐरे दो ऐरे में विभाजित किया जाता है, जिनमें से एक ऐरे में निर्धारित मान (जिसके आधार पर विभाजन किया जाता है और इसे पाइवोट कहते हैं) की तुलना में छोटे मान रखता है, और दूसरे ऐरे में पाइवोट मान से अधिक मानों को रख जाता है।

त्वरित सॉर्टिंग एक ऐरे को विभजित करती है और उसके बाद दो परिणामस्वरूप सब–ऐरे को सॉर्ट करने के लिए खुद को बारी बारी से दो बार कॉल करती है। यह एल्गोरिथम बड़े आकार के डेटा सेट के लिए काफी कुशल है।

इस एल्गोरिथ्म कि औसत (average) और सबसे खराब (worst case) स्थिति मे कॉम्प्लेक्सिटी O(nlogn) है, जहाँ n सॉर्ट किये जाने वाले तत्वों की संख्या हैं।

त्वरित सॉर्टिंग में विभाजन:

निम्नलिखित उदाहरण यह बताता हैं कि कैसे एक ऐरे में पाइवोट मान को सर्च किया जाता है।

पाइवोट मान लिस्ट को दो भागों में बटता हैं। और बारी बारी से, हम प्रत्येक उप–सूचियों के लिए पाइवोट मान का पता लगाते है जब तक की सभी सूचियों में केवल एक ही तत्व नहीं रह जाता ।

A[6]	A[7]	A[8]	A[9]	A[10]	A[11]	A[12]
98	84	65	108	60	96	72
72	84	65	108	60	96	98
72	84	65	98	60	96	108
72	84	65	96	60	98	108

पाइवोट एल्गोरिथमः

चरण 1 –सबसे अधिक सूचकांक मान को पाइवोट चुने।

चरण 2 – धुरी को छोड़कर सूची के दाईं तथा बायीं ओर इंगित करने के लिए दो वेरिएबल ले

चरण 3 – बांया वेरिएबल कम सूचकांक को इंगित करता है

चरण 4 – जबकि दांया वेरिएबल उच्च सूचकांक को इंगित करता है

चरण 5 – जब तक बांये वेरिएबल की वैल्यू पाइवोट से कम है दांये चले

चरण 6 – जब तक दांये वेरिएबल की वैल्यू पाइवोट से अधिक है बांये चले

चरण 7 – यदि दोनों चरण 5 और चरण 6 मैच नहीं करते तो बांये और दांये को स्वैप करे

चरण 8 – यदि बांया ≥दांया पॉइंट जहा वे मिलते है नया पाइवोट होगा

त्वरित सॉर्टिंग एल्गोरिथमः

त्वरित एल्गोरिथम का उपयोग बारी बारी से करते है जब तक की हम छोटे संभव विभाजन तक नहीं पहुच जाते। फिर प्रत्येक विभाजन पर त्वरित सॉर्ट की कार्रवाई की जाती है। हम निम्न रूप में त्वरित सॉर्ट की एल्गोरिथम को परिभाषित करते है –

चरण 1 – सबसे दाएँ सूचकांक मान को पाइवोट बनाए।

चरण 2 – पाइवोट मान का उपयोग कर ऐरे का विभाजन करे।

चरण 3 –रिक्रसीवेली बाएँ विभाजन पर त्वरित सॉर्ट लगाये।

चरण 4 – रिक्रसीवेली दॉये विभाजन पर तत्वरित सॉर्ट लगाये।

त्वरित सॉर्टिंग के लिए C प्रोग्राम:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 7
int intArray[MAX] = {4,6,3,2,1,9,7};
void printline(int count) {
int i;
for(i = 0;i <count-1;i++) {
printf("=");
}
printf("=\n");
}
void display() {
int i;
printf("[");
// navigate through all items
for(i = 0;i<MAX;i++) {
printf("%d ",intArray[i]);
}
printf("]\n");
```

```
}
void swap(int num1, int num2) {
int temp = intArray[num1];
intArray[num1] = intArray[num2];
intArray[num2] = temp;
}
int partition(int left, int right, int pivot) {
int leftPointer = left -1;
int rightPointer = right;
while(true) {
while(intArray[++leftPointer] < pivot) {
//do nothing
}
while(rightPointer > 0 && intArray[--rightPointer] > pivot) {
//do nothing
}
if(leftPointer >= rightPointer) {
break;
} else {
printf(" item swapped :%d,%d\n",
intArray[leftPointer],intArray[rightPointer]);
swap(leftPointer,rightPointer);
}
}
printf(" pivot swapped :%d,%d\n", intArray[leftPointer],intArray[right]);
swap(leftPointer,right);
printf("Updated Array: ");
display();
return leftPointer;
}
void quickSort(int left, int right) {
if(right-left <= 0) {
return;
} else {
```

```
int pivot = intArray[right];
int partitionPoint = partition(left, right, pivot);
quickSort(left,partitionPoint-1);
quickSort(partitionPoint+1,right);
}
}
main() {
printf("Input Array: ");
display();
printline(50);
quickSort(0,MAX-1);
printf("Output Array: ");
display();
printline(50);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
Input Array: [4 6 3 2 1 9 7 ]
==================================================
pivot swapped :9,7
Updated Array: [4 6 3 2 1 7 9 ]
pivot swapped :4,1
Updated Array: [1 6 3 2 4 7 9 ]
item swapped :6,2
pivot swapped :6,4
Updated Array: [1 2 3 4 6 7 9 ]
pivot swapped :3,3
Updated Array: [1 2 3 4 6 7 9 ]
Output Array: [1 2 3 4 6 7 9 ]
```

महत्वपूर्ण बिंदु

- सॉर्टिंग एल्गोरिथ्म डेटा को एक विशेष क्रम में व्यवस्थित करने का तरीका निर्दिष्ट करती है।
- बबल सॉर्ट एक साधारण सॉर्टिंग एल्गोरिथ्म है। यह सॉर्टिंग एल्गोरिथ्म, तुलना–आधारित एल्गोरिथ्म है जिसमें सन्निकट तत्वों के प्रत्येक जोड़े की तुलना की जाती है।
- मर्ज सॉर्ट डिवाइड (विभजित) एंड कॉन्कर (जीत) पर आधारित एक सॉर्टिंग तकनीक है। इसकी सबसे खराब मामले में (worst-case) कॉम्प्लेक्सिटी O(n log n) होने के कारण यह सबसे अच्छीएल्गोरिथ्म में से एक है।
- त्वरित एक प्रकार की अत्यंत कुशल सॉर्टिंग एल्गोरिथ्म है और डेटा के ऐरे को छोटे ऐरे में विभाजन करने पर आधारित है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्र1 बबल एल्गोरिथ्म की जटिलता है

(अ) O(N) (ब) $O(N^2)$

(स) O(logN) (द) O(NlogN)

प्र 2 मर्ज एल्गोरिथ्म की जटिलता है

(अ)O(N) (ब) $O(N^2)$

(स) O(logN) (द) O(NlogN)

प्र3 चयन एल्गोरिथ्म की जटिलता है

(अ) O(N) (ब) $O(N^2)$

(स) O(NlogN) (द) O(logN)

प्र4 कौन सा अच्छा सॉर्टिंग एल्गोरिथ्म है।

(अ) चयन सॉर्टिंग (ब) निवेशन सॉर्टिंग

(स) त्वरित सॉर्टिंग (द) कोई नहीं

प्र5 त्वरित क्रमबद्ध एल्गोरिथ्म की जटिलता है।

(अ) O(N) (ब) O(logN)

(स) $O(N^2)$ (द) O(NlogN)

लघुत्तरात्मक प्रश्न

प्र1 सॉर्टिंग क्या है?

प्र2 स्थिर सॉर्टिंग क्या है?

प्र3 इन–प्लेस सॉर्टिंग क्या है?

प्र4 त्वरित एल्गोरिथ्म के लिए सबसे खराब स्थिति (worst case) का रन टाइमहै।

प्र5 त्वरित एल्गोरिथ्म के लिए सबसे खराब स्थिति (worst case) है।

निबंधात्मक प्रश्न

प्र1 मर्ज सॉर्टिंग विस्तार में समझाईए।

प्र2 कोनसी सबसे अच्छी सॉर्टिंग एल्गोरिथ्म हैं और क्यों हैं?

प्र3 त्वरित सॉर्टिंग समझाईए।

प्र4 selection और Insertion सॉर्टिंग के बीच अंतर?

प्र5 स्थिर और अस्थिर सॉर्टिंग के बीच क्या अंतर है?

उत्तरमाला

उत्तर 1: ब उत्तर 2: दउत्तर 3: ब

उत्तर 4: स उत्तर 5: द

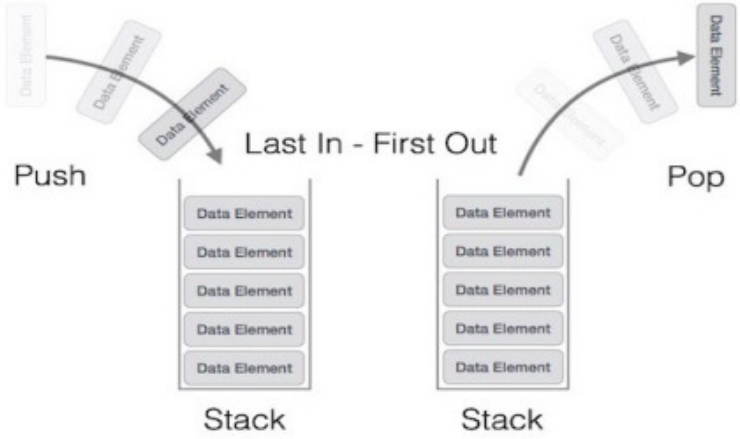
अध्याय 4

स्टैक और क्यु

स्टैक(Stack): स्टैक एक एब्सट्रैक्ट डाटा टाइप (एडीटी) है जिसका प्रयोग आमतौर पर सभी प्रोग्रामिंग भाषाओं में किया जाता है। उदाहरण के लिए–ताश के पत्तों की स्टैक या एक डेक या प्लेटों की स्टैक वास्तविक दुनिया में एक स्टैक की तरह बर्ताव करती है

वास्तविक दुनिया में स्टैक केवल एक छोर पर ऑपरेशन की अनुमति देती है। उदाहरण के लिए– हम केवल कार्ड या प्लेट स्टैक के ऊपर से तत्व ररव या निकाल सकते हैं। इसी तरह, स्टैक एडीटी केवल एक छोर पर डेटा के ऑपरेशन की अनुमति देता है। किसी भी समय, हम केवल स्टैक के शीर्ष तत्व का उपयोग कर सकते हैं। यह सुविधा स्टैक को LIFO(अंतिम–इन–फर्स्ट–आउट) डेटा स्ट्रक्चरबनाता है। यहाँ पर जो तत्व अंत में जोड़ा जाता है पहले हटाया जाता है स्टैक शब्दावली में जोड़ने को पुश (Push) तथा हटाने को पॉप (Pop) आपरेशन कहा जाता है।

स्टैक प्रेजेंटेशन (Presentation):निम्नलिखित चित्र मेंएक स्टैक और इसके ऑपरेशन को दर्शाया गया है –



एक स्टैक को ऐरे, स्ट्रक्चर, पॉइंटर और लिंक्ड लिस्ट के माध्यम से इम्पलीमेंट किया जा सकता है। एक स्टैक फिक्स साइज या डायनामिक साइज दोनों में से एक प्रकार का हो सकता है। हम यहाँ एक ऐरे का उपयोग कर स्टैक को इम्पलीमेंट कर रहे है जो एक फिक्स साइज स्टैक है।

बुनियादी ऑपरेशन:स्टैक ऑपरेशन का उपयोग स्टैक को इनिशलायजिंग और डीइनिशलायजिंग करने के लिए किया जाता है। इसके अलावा एक स्टैक निम्नलिखित दो प्राथमिक कार्यों के लिए प्रयोग किया जाता है –

Push() –एक तत्व स्टैक में जोड़ना

Pop()– एक तत्व स्टैक से हटाना

जब हम स्टैक में कोई तत्व जोड़ते है तब स्टैक के कुशलता से उपयोग के लिए उसका स्टेटस चेक करते है इसके लिए निम्नलिखित फंक्शन का उपयोग करते है–

peak() – स्टैक के शीर्ष डेटा तत्व को हटाये बिना प्राप्त करना

isFull() –स्टैक के भरे होने की जाच करना

isEmpty() – स्टैक के खाली होने की जाच करना

हर समय स्टैक में अंतिम जोड़े गए तत्व के लिये एक पॉइंटर का उपयोग करते है जो की स्टैक के टॉप को बताता है इसे TOP के नाम से जाना जाता है टॉप वेरिएबल बिना हटाये स्टैक के टॉप की वैल्यू बताता ह।

स्टैक फंक्शन के प्रोसीजर –

peek():

peek() फंक्शन के लिए एल्गोरिथ्म–

```
begin procedure peek
return stack[top]
end procedure
```

peek() फंक्शन का **C** भाषा में इम्प्लीमेंटेशन –

```
int peek() {
return stack[top];
}
```

isfull():

isfull() फंक्शन के लिए एल्गोरिथ्म–

```
begin procedure isfull
if top equals to MAXSIZE
return true
```

else
return false
endif
end procedure
isfull() फंक्शन का **C** भाषा में इम्प्लीमेंटेशन –
bool isfull() {
if(top == MAXSIZE)
return true;
else
return false;
}
isempty():
isempty() फंक्शन के लिए एल्गोरिथ्म –
begin procedure isempty
if top less than 1
return true
else
return false
endif
end procedure
isempty()फंक्शन का C भाषा में इम्प्लीमेंटेशन–
Example
bool isempty() {
if(top == -1)
return true;
else
return false;
}

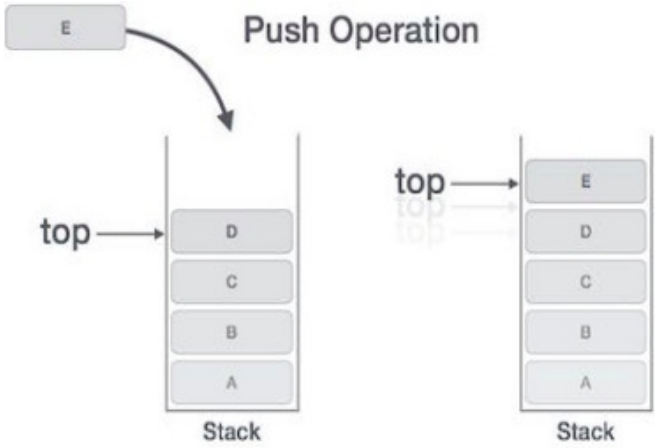
पुश ऑपरेशन(**Push Operation**):स्टैक में एक नया डेटा तत्व जोड़ने या डालने की प्रक्रिया को पुश आपरेशन कहते है पुश आपरेशन निम्नलिखित स्टेप्स की एक श्रृंखला है।

Step 1 – Checks if the stack is full.

Step 2 – If the stack is full, produces an error and exit.

Step 3 – If the stack is not full, increments top to point next empty space.

Step 4 – Adds data element to the stack location, where top is pointing.



Step 5 – Returns success.

यदि स्टैक को लिंक लिस्ट से इम्पलीमेंट करते है तो स्टेप 3 में डायनामिक मेमोरी आवंटित करनी होगी।

पुश आपरेशन के लिए एल्गोरिथ्म–

```
begin procedure push: stack, data
if stack is full
return null
endif
top ← top + 1
stack[top] ← data
end procedure
```

एल्गोरिथ्म का C में इम्प्लीमेंटेशन –

```
void push(int data) {
if(!isFull()) {
top = top + 1;
stack[top] = data;
} else {
printf("Could not insert data, Stack is full.\n");
```

}

}

पॉप आपरेशन(Pop Operation): स्टैक से एक डेटा तत्व को हटाने की प्रक्रिया को पॉप आपरेशन कहते है जब पॉप आपरेशन को एक ऐरे की मदद से इम्पलीमेंट करते है तो वास्तव में डाटा तत्व को हटाने की बजाए टॉप वेरिएबल को एक से घटाते है जबकि लिंक लिस्ट से इम्पलीमेंट करने पर वास्तव में डाटा तत्व को हटा कर मेमोरी को deallocates किया जाता है पॉप आपरेशन निम्नलिखित स्टेप्स की एक श्रृंखला है –

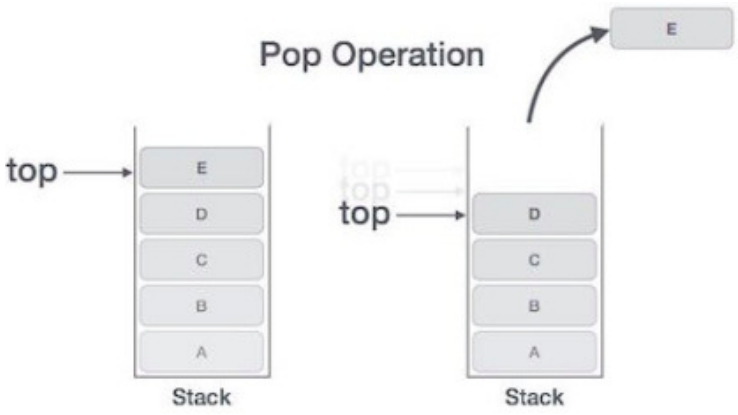
Step 1 – Checks if the stack is empty.

Step 2 – If the stack is empty, produces an error and exit.

Step 3 – If the stack is not empty, accesses the data element at which top is pointing.

Step 4 – Decreases the value of top by 1.

Step 5 – Returns success.



पॉपआपरेशन के लिए एल्गोरिथ्म–

begin procedure pop: stack
if stack is empty
return null
endif
data ← stack[top]
top ← top - 1
return data
end procedure

एल्गोरिथ्म का C में इम्प्लीमेंटेशन –

```
int pop(int data) {
if(!isempty()) {
data = stack[top];
top = top - 1;
return data;
} else {
printf("Could not retrieve data, Stack is empty.\n");
}
}
```

स्टैक के उपयोग–निम्नलिखित कार्यो के लिये स्टैक का उपयोग किया जा सकता है।

(क) अंकगणित अभिव्यक्ति मूल्यांकन(अरिथमैटिक एक्सप्रेशन इवैल्यूएशन)

(ख) बकट्रैकिंग

(ग) स्मृति प्रबंधन(मेमोरी मैनेजमेंट)

(क) अंकगणित अभिव्यक्ति मूल्यांकन(अरिथमैटिक एक्सप्रेशन इवैल्यूएशन): अरिथमैटिक एक्सप्रेशन लिखने के तरिके को नोटेशन कहते है एक अरिथमैटिक एक्सप्रेशन को तीन अलग अलग तरीको लेकिन समान नोटेशन में लिख सकते है बिना सार या आउटपुट के बदले। निम्नलिखित तीन नोटेशन है–

इन्फिक्स नोटेशन

उपसर्ग (पोलिश) नोटेशन

पोस्टफिक्स (रिवर्स पोलिश) नोटेशन

इन एक्सप्रेशन का नाम ऑपरेटर के उपयोग के अनुसार दिया गया है।

इन्फिक्स नोटेशन

हम एक एक्सप्रेशन a - b + c लिखते है जिसमे में ऑपरेटर ऑपरेंड के मध्य इस्तेमाल किया गया है ये एक इन्फिक्स नोटेशन है इसका मनुष्य के लिये पढना,लिखना और बोलना आसान है लेकिन कंप्यूटिंग उपकरणों के लिए मुश्किल है एक एल्गोरिथ्म में इन्फिक्स एक्सप्रेशन को प्रोसेस करने लिए अधिक टाइम और स्पेस की आवश्यकता होती है।

उपसर्ग (पोलिश) नोटेशन

इस नोटेशन में ऑपरेटर ऑपरेंड के आगे लिखा होता है उदाहरण के लिए +ab जो की इन्फिक्स नोटेशन a + b के समान है उपसर्ग नोटेशन को पोलिश नोटेशन भी कहते है।

पोस्टफिक्स नोटेशन

पोस्टफिक्स नोटेशन को रिवर्स पोलिश नोटेशन कहते है इसमें ऑपरेटर ऑपरेंड के बाद में होता है उदाहरण के लिए ab+ जो की इन्फिक्स नोटेशन a + b के समान है।

स्टैक का उपयोग एक नोटेशन को दूसरे नोटेशन में रूपांतरण के लिए किया जाता है।

(ख) बकट्रैकिंग:बकट्रैकिंग का प्रयोग एल्गोरिथ्म में किया जाता है जहां किसी पथ के साथ स्टेप्स होते है जो किसी स्टार्ट पॉइंट से किसी उद्देश्य तक हो। उदाहरण के लिए

एक भूलभुलैया के माध्यम से अपना रास्ता सर्च करना।

एक ग्राफ में एक पॉइंट से दूसरे पॉइंट तक रास्ता पता करना।

उपरोक्त सभी मामलो में एक पॉइंट से दुसरे पॉइंट तक जाने के लिए बहुत सारे विकल्प होते है यदि एक पॉइंट से दुसरे पॉइंट पर जाने के बाद वापस पहले पॉइंट पर आना हो और अन्य विकप्ल चुनना हो।
फिर, समाधान के लिए स्टैक का इस्तेमाल किया जा सकता है। रिकर्शन एक अन्य ईस्ट समाधान है जिसको स्टैक की मदद से इम्पलीमेंट कर सकते है।

(ग) स्मृति प्रबंधन: कोई भी आधुनिक कंप्यूटर अपने प्रोग्राम को रन करने के लिए प्राथमिक स्मृति प्रबंधन मॉडल के रूप में एक स्टैक उपयोग करता है।

क्यु (Queue):क्यु एक एब्सट्रेक्ट डेटा स्ट्रक्चर है जो कुछ हद तक स्टैक के समान है। स्टैक के विपरीत, एक क्यु अपने दोनों शीरों पर खुला होता है। एक शीरा तत्व को जोड़ने (enqueue) और दूसरा शीरा तत्व को हटाने(dequeue) के उपयोग में आता है क्यु डेटा स्ट्रक्चर पहले आओ पहले जाओ (FIFO) के सिद्धांत पर काम करता है अर्थात, पहले संग्रहीत डेटा आइटम पहले हटा दिए जायंगे।

क्यु के लिए एक वास्तविक दुनिया में एक सिंगल लेन का रोड जिसमे जो साधन पहले प्रवेश करेगा पहले बाहर आएगा एक उदाहरण के रूप में देखा जा सकता है।

Queue

क्यु प्रेजेंटेशन (Presentation):अब हम समझ गए की क्यु में दोनों सिरो को अलग अलग कारण के लिए उपयोग में लेते है निम्नलिखित चित्र की सहायता से क्यु का डाटा स्ट्रक्चर के रूम में प्रेजेंटेशन है।

एक स्टैक की तरह क्यु को भी ऐरे, स्ट्रक्चर, पॉइंटर और लिंक्ड लिस्ट के माध्यम से इम्पलीमेंट किया जा सकता है। एक क्यु फिक्स साइज या डायनामिक साइज दोनों में से एक प्रकार का हो सकता है। हम यहाँ एक ऐरे का उपयोग कर क्यु को इम्पलीमेंट कर रहे है जो एक फिक्स साइज क्यु है।

क्यु के बुनियादी ऑपरेशन: क्युऑपरेशन का उपयोग क्यु को इनिशलायजिंग और डीइनिशलायजिंग करने के लिए किया जाता है।निम्नलिखित क्यु के बुनियादी आपरेशन है।

enqueue() – एक तत्व क्यु में जोड़ना

dequeue() – एक तत्व क्यु से हटाना

क्यु के कुशलता से उपयोग के लिए निम्नलिखित फंक्शनस का उपयोग करते है–

peek() – क्यु के शीर्ष डेटा तत्व को हटाये बिना प्राप्त करना

isfull() – क्यु के भरे होने की जाच करना

isempty() – क्यु के खाली होने की जाच करना

क्यु के सपोर्टिव फंक्शनस निम्न है।

peek()

peek() फंक्शन के लिए एल्गोरिथ्म–

begin procedure peek

return queue[front]

end procedure

peek() फंक्शन का **C** भाषा में इम्प्लीमेंटेशन –

Example

```
int peek() {
return queue[front];
```

}

isfull():

यहाँ पर हम क्यु को इम्पलीमेंट करने के लिए एक आयामी ऐरे का उपयोग कर रहे है क्यु के भरे होने की जाच के लिए हम केवल रियर पॉइंटर को चेक करते है की वह मैक्स साइज के बराबर है या नहीं। यदि हम क्यु को सर्कुलर लिंक लिस्ट से इम्पलीमेंट करते है तो अलग एल्गोरिथ्म होगी।

isfull( ) फंक्शन के लिए एल्गोरिथ्म–

begin procedure isfull

if rear equals to MAXSIZE

return true

else

return false

endif

end procedure

isfull( )फंक्शन का **C** भाषा में इम्प्लीमेंटेशन –

```
bool isfull() {
if(rear == MAXSIZE - 1)
return true;
else
return false;
}
```

isempty():

isempty() फंक्शन के लिए एल्गोरिथ्म–

begin procedure isempty

if front is less than MIN OR front is greater than rear

return true

else

return false

endif

end procedure

यदि फ्रंट का मान मिन या 0 से कम है तो इसका मतलब क्यु को इनिसलाएज नहीं किया है और क्यु खाली है।

isempty() फंक्शन का C भाषा में इम्प्लीमेंटेशन –

```
bool isempty() {
if(front < 0 || front > rear)
return true;
else
return false;
```

एनक्यु **(Enqueue)** आपरेशन: क्यु में दो डाटा पॉइंटर फ्रंट और रियर होते है इसलिए इसके आपरेशन स्टैक से कठिन होते है क्यु में डाटा तत्व को जोड़ने (Insert) के लिए निम्नलिखित स्टेप्स का उपयोग करते हैं।

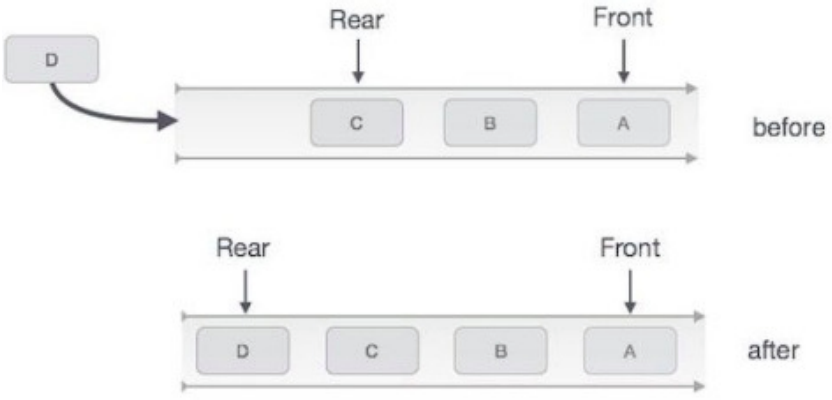
Step 1 – Check if the queue is full.

Step 2 – If the queue is full, produce overflow error and exit.

Step 3 – If the queue is not full, increment rear pointer to point the next empty space.

Step 4 – Add data element to the queue location, where the rear is pointing.

Step 5 – return success.



Queue Enqueue

एनक्यु (Enqueue) आपरेशन के लिए एल्गोरिथ्म–

```
procedure enqueue(data)
```

```
if queue is full
return overflow
endif
rear ← rear + 1
queue[rear] ← data
return true
end procedure
```

एनक्यु (Enqueue) आपरेशन का C भाषा में इम्प्लीमेंटेशन –

```
int enqueue(int data)
if(isfull())
return 0;
rear = rear + 1;
queue[rear] = data;
return 1;
end procedure
```

डीक्यु (Dequeue) आपरेशन:डाटा तत्व को क्यु से हटाने का काम दो भागो में किया जाता है एक उस डाटा तत्व को एक्सेस करना जहा पॉइंटर पॉइंट कर रहा हो और दूसरा उसको वहाँ से हटाना। निम्नलिखित स्टेप्स से डीक्युआपरेशन परफॉर्म किया जाता है–

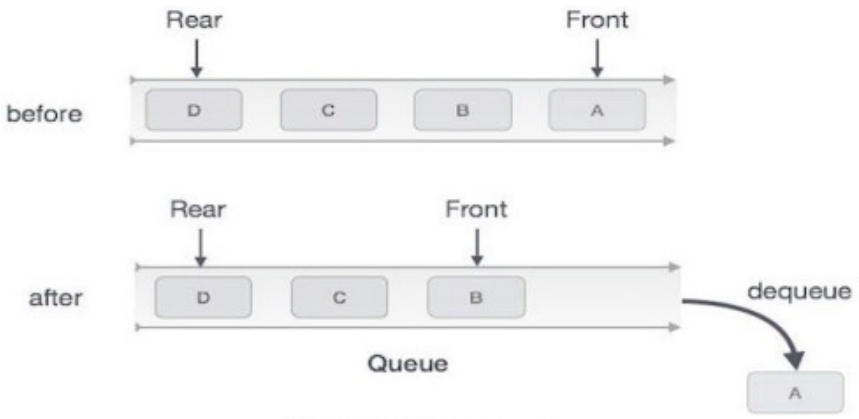
Step 1 – Check if the queue is empty.

Step 2 – If the queue is empty, produce underflow error and exit.

Step 3 – If the queue is not empty, access the data where front is pointing.

Step 3 – Increment front pointer to point to the next available data element.

Step 5 – Return success.



Queue Dequeue

डीक्यु (Dequeue) आपरेशन के लिए एल्गोरिथ्म–

```
procedure dequeue
if queue is empty
return underflow
end if
data = queue[front]
front ← front + 1
return true
end procedure
```

डीक्यु (Dequeue) आपरेशन का C भाषा में इम्प्लीमेंटेशन –

```
int dequeue() {
if(isempty())
return 0;
int data = queue[front];
front = front + 1;
return data;
}
```

महत्वपूर्ण बिंदु

- स्टैक एक एब्सट्रैक्ट डाटा टाइप (एडीटी) है जिसका प्रयोग आमतौर पर सभी प्रोग्रामिंग भाषाओं में किया जाता है।
- एक स्टैक को ऐरे, स्ट्रक्चर, पॉइंटर और लिंक्ड लिस्ट के माध्यम से इम्पलीमेंट किया जा सकता है। एक स्टैक फिक्स साइज या डायनामिक साइज दोनों में से एक प्रकार का हो सकता है।
- क्यु एक एब्सट्रेक्ट डेटा स्ट्रक्चर है जो कुछ हद तक स्टैक के समान है। स्टैक के विपरीत, एक क्यु अपने दोनों शीरों पर खुला होता है।
- कोई भी आधुनिक कंप्यूटर अपने प्रोग्राम को रन करने के लिए प्राथमिक स्मृति प्रबंधन मॉडल के रूप में एक स्टैक उपयोग करता है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्र1 निम्न में से कौन सा नाम स्टैक से संबंधित नहीं है।

(अ)FIFO सूची (ब)LIFO सूची

(स)POP (द)Push

प्र2 शब्द Push और POP किस से संबंधित है।

(अ) ऐरे (ब) लिस्ट

(स) स्टैक (द) ऊपर के सभी

प्र3 एक डेटा स्ट्रक्चर जहां तत्वों का जोड़ना या हटाना किसी भी सिरे पर किया जा सकता है लेकिन बीच में नहीं।

(अ) लिंक लिस्ट (ब) स्टैक

(स) क्यु (द) डीक्यु

प्र4 ग्राफ में Breadth First Traversal के लिए आवश्यक डेटा स्ट्रक्चर है।

(अ) स्टैक (ब) ऐरे

(स) क्यु (द) टी(Tree)

प्र5 एक क्यु है।

(अ)FIFO लिस्ट (ब)LIFO लिस्ट

(स) ओर्डड ऐरे (द) रैखिक Tree

लघुत्तरात्मक प्रश्न

प्र1 स्टैक को परिभाषित करें।

प्र2 क्यु को परिभाषित करें।

प्र3 पुश ऑपरेशन क्या है?

प्र4 पॉप ऑपरेशन क्या है?

निबंधात्मक प्रश्न

प्र1 स्टैक डेटा स्ट्रक्चर के एपलीकेशन को समझाओं।

प्र2 स्टैक ऑपरेशन को विस्तार से समझाओ।

प्र3 विस्तार में सरक्युलर क्यु को समझाओ।

प्र4 डीक्यु को समझाओ।

उत्तरमाला

उत्तर 1: अउत्तर 2: स उत्तर 3: द

उत्तर 4: स उत्तर 5: अ

अध्याय 5

लिंक्ड लिस्ट(LinkedList)

लिंक लिस्ट एक लीनियर डाटा स्ट्रक्चर होता है जिसमे तत्वो की सीरीज इस तरह होती है कि प्रत्येक तत्व अपने अगले तत्व को पॉइंट करता है लिंक लिस्ट में प्रत्येक तत्व को नोड कहते है।आसान भाषा में लिस्ट एक तत्वो कि सीरीज है जिसमे तत्व एक दूसरे से जुड़े हुए है। लिंक लिस्ट ऐरे के बाद सबसे अधिक काम आने वाला डाटा स्ट्रक्चर है लिंक्ड लिस्ट की अवधारणा को समझने के लिए निम्नलिखित महत्वपूर्ण शब्द हैं ।

नोड(Node)– प्रत्येक नोड में डाटा आइटम और अगले नोड का एड्रेस होता है।

नेक्स्ट(Next)– एक पॉइंटर फील्ड होता है जिसमे नेक्स्ट नोड का एड्रेस होता है।

लिंक्ड लिस्ट का प्रेजेंटेशन(**Representation):**लिंक लिस्ट को नोड्स कि चैन के रूप में प्रदर्शित कर सकते है जहां पर हरेक नोड अगले नोड को पॉइन्ट करता है

लिंक्ड लिस्ट के प्रकार**:**

लिंक लिस्ट के निम्नलिखित विभिन्न प्रकार हैं।

सिंगल लिंक लिस्ट (Single Linked List) –केवल फॉरवर्ड पॉइंटर होता है

डबल लिंक लिस्ट(Doubly Linked)–फॉरवर्ड और बैकवर्ड पॉइंटर होता है

सर्कुलर लिंक लिस्ट(List Circular Linked List)– अंतिम तत्व पहले तत्व को पॉइन्ट करता है

लिंक लिस्ट के लाभ:निम्नलिखित लिंक लिस्ट के लाभ हैं

(क) लिंक्ड लिस्ट डायनामिक डाटा स्ट्रक्चर है।

(ख) लिंक लिस्ट रन टाइम के दौरान विकसित और सिकुड़ सकती है।

(ग) लिंक लिस्ट में इंसर्शन और डिलेशन आसान होता हैं।

(घ) कुशल स्मृति उपयोग, यानी स्मृति के पूर्व आवंटित कि कोई जरूरत नहीं।

(ड) एक्सेस टाइम फास्ट होता है मेमोरी ओवरहेड के बिना कॉन्स्टेन्ट टाइम में बढ़ सकती है।

(च) लीनियर डाटा स्ट्रक्चर जैसे स्टैक,क्यू को लिंक लिस्ट कि मदद से आसानी से इम्पलीमेंट कर सकते है।

लिंक लिस्ट के नुकसान:निम्नलिखित लिंक लिस्ट के नुकसान हैं

(क) यदि आवश्यक मेमोरी का पता हो तो मेमोरी वेस्टेज होता है।

(ख) सर्च कर पाना मुश्किल है।

बुनियादी आपरेशन:लिंक लिस्ट के निम्नलिखित बुनियादी आपरेशन है

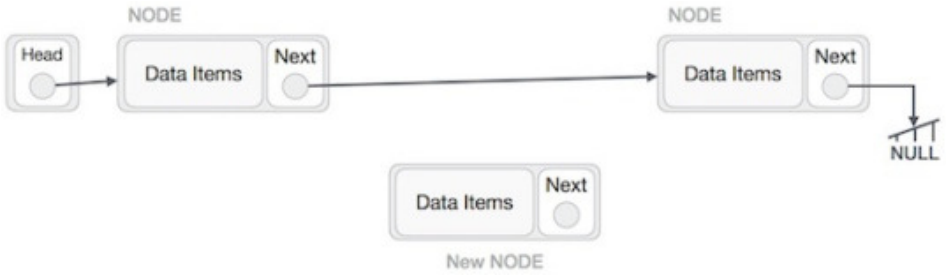
इनर्सशन (Insertion): इनर्सशन का अर्थ एक डाटा स्ट्रक्चर में एक नये डेटा तत्व को जोडना।

डिलिशन (deletion):डिलिशन का अर्थ एक डाटा स्ट्रक्चर में एक डेटा तत्व को हटाना यदि वह मौजूद है।

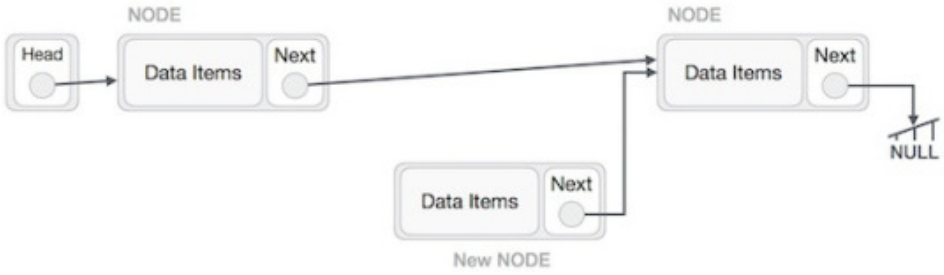
सर्च (Search):एक डाटा स्ट्रक्चर में निर्दिष्ट डेटा तत्व को खोजने को सर्च कहते है।

डिस्प्ले (Display): पूर्ण लिस्ट को डिस्प्ले करता है

इनर्सशन(Insertion) ऑपरेशन:इनर्सशन का अर्थ एक डाटा स्ट्रक्चर में एक नये नोड को जोड़ना है। हम यहाँ निम्नलिखित चित्र कि मदद से समझेंगे। सबसे पहले एक नया नोड बनाइये और इन्सर्ट करने के लिए लोकेशन पता करिये

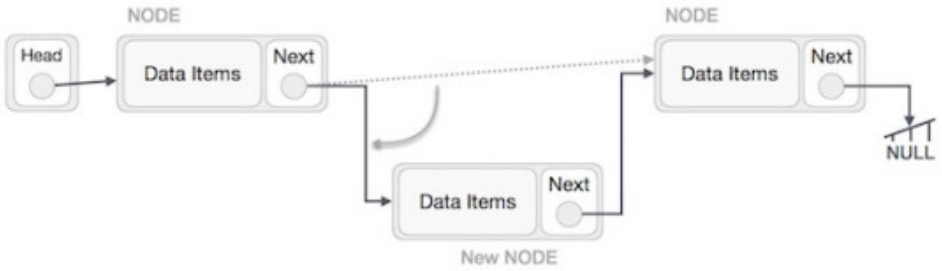


कल्पना कीजिए कि हम एक नोड B(NewNode),A(LeftNode) और C(RightNode) के बीच इन्सर्ट करना चाहते हैं। तब B.next C को पॉइन्ट करेगा और NewNode.next -> RightNode;अब यह इस तरह दिखेगा

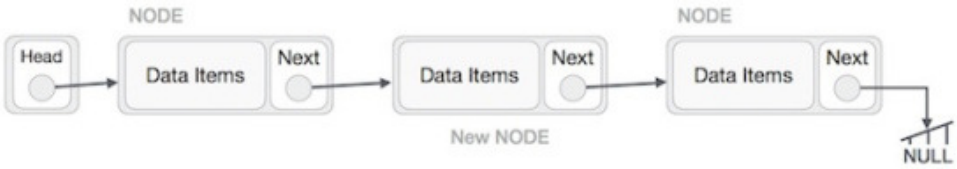


अब लेफ्ट नोड नए नोड को पॉइन्ट करेगा।

LeftNode.next -> NewNode;

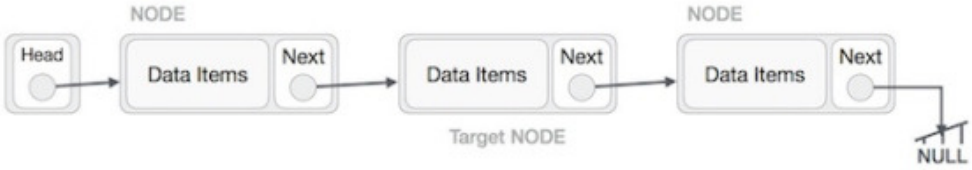


अब दोनों नोड्स के बिच में नए नोड को इन्सर्ट कर देंगे फिर नयी लिस्ट इस प्रकार होगी–



यदि नोड लिस्ट के शुरू में इन्सर्ट करना हो तो समान विधि अपनानी होगी और अंत में इन्सर्ट करना हो तो अंतिम नोड नए नोड को पॉइंट करेगा और नया नोड Null को पॉइंट करेगा

डिलिशन (deletion) ऑपरेशन:डिलिशन का अर्थ एक डाटा स्ट्रक्चर में एक डेटा तत्व को हटाना यदि वह मौजूद है। डिलेशन भी एक से अधिक स्टेप्स का प्रोसेस है चित्र की मदद से देखते है कि सबसे पहले सर्चिंग का उपयोग कर डिलीट करने वाले तत्व को खोजते है।



अब टारगेट नोड के पहले वाला नोड उसके बाद वाले नोड को पॉइंट करेगा–LeftNode.next -> TargetNode.next;

अब टारगेट नोड जिस नोड को पॉइंट कर रहा था वो लिंक निम्नलिखित कोड से हट जायेगा।

TargetNode.next –> NULL;

अगर हमें जरुरत है तो डिलीट किये गए नोड को रख सकते है अन्यथा हम मेमोरी को deallocate कर सकते है।

लिंक लिस्ट के ऑपरेशन्स के लिए C प्रोग्राम:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
struct node {
int data;
int key;
struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;

//display the list
void printList() {
struct node *ptr = head;
printf"\n[ ");

//start from the beginning
while(ptr != NULL) {
printf("(%d,%d) ",ptr->key,ptr->data);
ptr = ptr->next;
}

printf(" ]");
}
```

```
//insert link at the first location
void insertFirst(int key, int data) {
//create a link
struct node *link = (struct node*) malloc(sizeof(struct node));

link->key = key;
link->data = data;

//point it to old first node
link->next = head;

//point first to new first node
head = link;
}

//delete first item
struct node* deleteFirst() {

//save reference to first link
struct node *tempLink = head;

//mark next to first link as first
head = head->next;

//return the deleted link
return tempLink;
}

//is list empty
bool isEmpty() {
return head == NULL;
}

int length() {
int length = 0;
struct node *current;

for(current = head; current != NULL; current = current->next) {
```

```
length++;
}

return length;
}

//find a link with given key
struct node* find(int key) {

//start from the first link
struct node* current = head;

//if list is empty
if(head == NULL) {
return NULL;
}

//navigate through list
while(current->key != key) {

//if it is last node
if(current->next == NULL) {
return NULL;
} else {
//go to next link
current = current->next;
}
}

//if data found, return the current Link
return current;
}

//delete a link with given key
struct node* delete(int key) {

//start from the first link
struct node* current = head;
struct node* previous = NULL;
```

```
//if list is empty
if(head == NULL) {
return NULL;
}

//navigate through list
while(current->key != key) {

//if it is last node
if(current->next == NULL) {
return NULL;
} else {
//store reference to current link
previous = current;
//move to next link
current = current->next;
}
}

//found a match, update the link
if(current == head) {
//change first to point to next link
head = head->next;
} else {
//bypass the current link
previous->next = current->next;
}

return current;
}

void sort() {

int i, j, k, tempKey, tempData;
struct node *current;
struct node *next;

int size = length();
```

```
k = size ;

for ( i = 0 ; i < size - 1 ; i++, k-- ) {
current = head;
next = head->next;

for ( j = 1 ; j < k ; j++ ) {

if ( current->data > next->data ) {
tempData = current->data;
current->data = next->data;
next->data = tempData;

tempKey = current->key;
current->key = next->key;
next->key = tempKey;
}

current = current->next;
next = next->next;
}
}
}

void reverse(struct node** head_ref) {
struct node* prev   = NULL;
struct node* current = *head_ref;
struct node* next;

while (current != NULL) {
next  = current->next;
current->next = prev;
prev = current;
current = next;
}

*head_ref = prev;
}
```

```
main() {
insertFirst(1,10);
insertFirst(2,20);
insertFirst(3,30);
insertFirst(4,1);
insertFirst(5,40);
insertFirst(6,56);

printf("Original List: ");

//print list
printList();

while(!isEmpty()) {
struct node *temp = deleteFirst();
printf("\nDeleted value:");
printf("(%d,%d) ",temp->key,temp->data);
}

printf("\nList after deleting all items: ");
printList();
insertFirst(1,10);
insertFirst(2,20);
insertFirst(3,30);
insertFirst(4,1);
insertFirst(5,40);
insertFirst(6,56);

printf("\nRestored List: ");
printList();
printf("\n");

struct node *foundLink = find(4);

if(foundLink != NULL) {
printf("Element found: ");
printf("(%d,%d) ",foundLink->key,foundLink->data);
printf("\n");
} else {
```

```
printf("Element not found.");
}
delete(4);
printf("List after deleting an item: ");
printList();
printf("\n");
foundLink = find(4);
if(foundLink != NULL) {
printf("Element found: ");
printf("(%d,%d) ",foundLink->key,foundLink->data);
printf("\n");
} else {
printf("Element not found.");
}
printf("\n");
sort();
printf("List after sorting the data: ");
printList();
reverse(&head);
printf("\nList after reversing the data: ");
printList();
}
```

If we compile and run the above program, it will produce the following result –

Output

```
Original List:
[ (6,56) (5,40) (4,1) (3,30) (2,20) (1,10) ]
Deleted value:(6,56)
Deleted value:(5,40)
Deleted value:(4,1)
Deleted value:(3,30)
Deleted value:(2,20)
```

Deleted value:(1,10)
List after deleting all items:
[ ]
Restored List:
[(6,56) (5,40) (4,1) (3,30) (2,20) (1,10)]
Element found: (4,1)
List after deleting an item:
[(6,56) (5,40) (3,30) (2,20) (1,10)]
Element not found.
List after sorting the data:
[(1,10) (2,20) (3,30) (5,40) (6,56)]
List after reversing the data:
[(6,56) (5,40) (3,30) (2,20) (1,10)]

महत्वपूर्ण बिंदु

- लिंक लिस्ट एक लीनियर डाटा स्ट्रक्चर होता है जिसमे तत्वो की सीरीज इस तरह होती है कि प्रत्येक तत्व अपने अगले तत्व को पॉइंट करता है लिंक लिस्ट में प्रत्येक तत्व को नोड कहते है।
- लीनियर डाटा स्ट्रक्चर जैसे स्टैक,क्यू को लिंक लिस्ट कि मदद से आसानी से इम्पलीमेंट कर सकते है।
- लिंक्ड लिस्ट डायनामिक डाटा स्ट्रक्चर है

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्र1 लिंक लिस्ट सबसे उपयुक्त हैं

(अ) डेटा के स्थायी संग्रह के लिए

(ब) लगातार बदल रहे स्ट्रक्चर के आकार और डेटा के लिए

(स) ऊपर की दोनों स्थिति के लिए

(द) उपरोक्त में से कोई नहीं

प्र2 आम तौर पर नोड्स के संग्रह को _____ कहा जाता है।

(अ) स्टैक (ब) लिंक्ड लिस्ट

(स) स्टैक (द) पोंइन्टर

प्र3 निम्न में से कौन सा लिंक्ड लिस्ट का एक प्रकार नहीं है

(अ) डबल लिंक लिस्टम (ब) सिंगल लिंक्ड लिस्ट

(स) सरक्युलर लिंक्ड लिस्ट (द) हाइब्रिड लिंक्ड लिस्ट

प्र4 लिंक लिस्ट आम तौर पर _____ स्मृति आवंटन के उदाहरण के रूप में जाना जाता है।

(अ) स्थिर (ब) डायनेमिक

(स) कम्पाईल टाईम (द) इनमे से कोई नहीं

प्र5 एक सरक्युलर लिंक लिस्ट में

(अ) सभी तत्व सिक्वैंशियल तरीके से जुड़े होते हैं।

(ब) इसमे कोई शुरुआत और कोई अंत नहीं होता है।

(स) अवयव पदानुक्रम में व्यवस्थित होते हैं।

(द) सूची के भीतर आगे और पीछे चंक्रमण की अनुमति होती है।

लघुत्तरात्मक प्रश्न

प्र1 लिंक लिस्ट को परिभाषित करें।

प्र2 हैडर लिंक लिस्ट क्या है?

प्र3 ऐरे और लिंक लिस्ट में कौन बेहतर है?

प्र4 सरक्युलर लिंक लिस्ट को परिभाषित करें।

निबंधात्मक प्रश्न

प्र1 डबल लिंक लिस्टको समझाओ।

प्र2 सिंगल और डबल लिंक लिस्ट के बीच अंतर को बताओ।

प्र3 लिंक लिस्ट किस प्रकार की स्मृति आवंटन से जुड़ा हुआ हैं?

प्र4 लिंक लिस्ट का उपयोग समझाओ।

उत्तरमाला

उत्तर 1: ब उत्तर 2: ब उत्तर 3: द

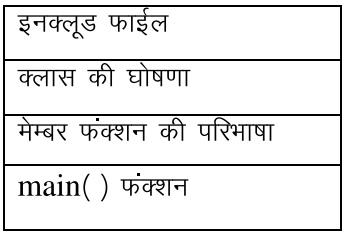
उत्तर 4: ब उत्तर 5: ब

अध्याय – 6

C++ के साथ शुरूआत

6.1 C++ प्रोग्राम की संरचना

एक C++ प्रोग्राम में चार अनुभाग होते हैं जैसे – चित्र 6.1 में दर्शाया गया है। ये अनुभाग अलग–अलग सॉर्स फाइलों में भी रखे जा सकते है और उसके बाद अलग–अलग या एक साथ भी कम्पाइल किये जा सकते है।



चित्र 6.1 C++ प्रोग्राम की संरचना

6.2 C++ का एक सरल प्रोग्राम

आउटपुट स्क्रीन पर "Hello World" प्रिन्ट करने का प्रोग्राम।

प्रोग्राम 6.1 C++ का एक सरल प्रोग्राम

```
#include<iostream>          //include header file
using namespace std;
int  main()
{
        cout<<"Hello World";       //print "Hello World"
        return 0;
}
```

प्रोग्राम 6.1 का आउटपुट होगा–

```
Hello World
```

C++ प्रोग्राम की विशेषताए

- C की तरह C++ प्रोग्राम भी फंक्शनस का एक संग्रह है।
- एक C++ प्रोग्राम में main ( ) फक्शन अनिवार्य है।

- C प्रोग्राम की तरह C++ प्रोग्राम में स्टेटमेन्ट अर्द्धविराम (;) से समाप्त होते हैं।

कमेन्टस्

- // (डबल श्लेस) कमेन्ट एक लाईन को कमेन्ट करने के लिए प्रयोग किया जाता है।

उदाहरण :–

```
// This is my first C++ program.
```

- /* */ एक से अधिक लाईनों को कमेन्ट करने के लिए प्रयोग किया जाता है।

उदाहरण :–

```
/* This is my
first C++ program */
```

iostream फाईल

- जिन स्टेटमेंट से पहले # चिन्ह् लगा हो उनको प्रिप्रोसेसर डाइरेक्टीव स्टेटमेन्ट कहते है।
- इनको C++ प्रोग्राम के शुरू में लिखा जाता है।
- प्रिप्रोसेसर इस तरह के स्टेटमेंट को प्रोग्राम कम्पायल होने से पूर्व प्रोसेस करता है।
- # include < iostream >यह स्टेटमेंट iostream फाईल के कन्टेन्ट को प्रोग्राम के साथ जोड़ता है।
- इसमें cout आइडेंटीफायर और इनसर्सन ऑपरेटर की घोषणा होती है।

Namespace

- यह प्रोग्राम में प्रयुक्त आयडेंटीफायर्स के स्कोप को परिभाषित करता है।
- Namespaceस्कोप का प्रयोग करने के लिए 'using namespace std' लिखते है।
- 'std' वो नेमस्पेश है जहाँ C++ स्टेडर्ड क्लास लाइब्रेरी परिभाषित है।

6.3 कम्पाइलिगं एवं लिकिंग

कम्पाइलिंग एवं लिकिंग की प्रकिया ओपरेटिंग सिस्टम पर निर्भर करती है।

Linux OS

g++ कमांड का प्रयोग C++ प्रोग्राम का कम्पाइलिंग व लिकिंग के लिए किया जाता है।

उदाहरण :– g++ abc.cpp.

यह कमांड abc.cpp फाईल में लिखे गये प्रोग्राम को कम्पाइल करती है। कम्पाइलर एक ओब्जेक्ट फाईल abc.o का निमार्ण करता है और स्वतः ही लाइब्रेरी फंक्शनस के साथ लिंक होकर एग्जीक्यूबल फाईल का निमार्ण करता है। डिफॉल्ट एग्जीक्यूबल फाईल का नाम a. out होता है ।

MS DOS

Turbo C++ or Borland C++कम्पाइलर इन्टीग्रेटेड डवलपमेन्ट एनवार्यमेन्ट कोMS DOSमें उपलब्ध कराते है। ये एक एडीटर भी उपलब्ध कराता है जिसके साथ फाईल, एडीट, कम्पाइल और रन ओपसंस होते है।

- फाईल ऑपशन – सॉर्स फाईल को बनाने व सेव करने के लिए।
- एडीट ऑपशन – सॉर्स फाईल को एडीट करने के लिए।
- कम्पाइल ऑपशन – प्रोग्राम को कम्पाइल करने के लिए।
- रन ऑपशन – प्रोग्राम को कम्पाइल, लिंक, और रन एक साथ करने के लिए।

6.4 टोकन्स

प्रोग्राम की सबसे छोटी इकाई को टोकन्स कहा जाता है।

प्रोग्राम में निम्न लिखित टोकन्स होते है।

- कीवर्डस ,
- आयडेन्टीफायर्स
- कांन्सटेंट
- स्ट्रीगंस

प्रोग्राम इन्ही टोकन्स के प्रयोग से लिखा जाता है।

6.5 कीवर्डस

यह रिजर्व वर्डस होते है जिनका अर्थ प्रोग्रामर के द्वारा बदला नहीं जा सकता है। इनका प्रयोग वेरिएबल, कांस्टेट और अन्य यूजर–डिफाइंड प्रोग्राम इकाईयों के नाम के लिए नहीं किया जा सकता है।C++ कीवर्डस् की सम्पूर्ण लिस्ट टेबल 6.1 में दर्शाया गया है। इनमें से कई कीवडर्स C और C++ में एक समान है।

टेबल 6.1 **C++** कीवर्डस

asm	double	new	switch
auto	else	operator	template
break	enum	private	this
case	extern	protected	throw
catch	float	public	try

char	for	register	typedef
class	friend	return	union
const	goto	short	unsigned
continue	if	signed	virtual
default	inline	sizeof	void
delete	int	static	volatile
do	long	struct	while

6.6 आइडेंटीफायर एवं कांस्टेंट

वेरिएबल, फंक्शन, ऐरे, क्लास इत्यादि के नाम जो प्रोग्रामर के द्वारा दिये जाते है उन्हें आइडेंटीफयर कहा जाता है। इन आइडेंटीफयर को नाम देने के लिए हर एक भाषा के अपने नियम होते है। निम्नलिखित नियम जो C और C++ में एक समान है।

- केवल अग्रेजी वर्णमाला के अक्षर, अंको और अंडरस्कोर का प्रयोग कर सकते है।
- किसी अंक के साथ नाम की शुरूआत नहीं की जा सकती है।
- अंग्रेजी के छोटे और बड़े अक्षर अलग–अलग माने जाते है।
- कीवर्ड का प्रयोग वेरिएबल के नाम के लिए नहीं किया जा सकता है।

फिक्सड वेल्यू जो प्रोग्राम के एक्जीक्यूशन के दौरान बदलती नहीं है उन्हें कांस्टेट कहा जाता है। C++ में विभिन्न प्रकार के कांस्टेट होते है जैसे इंटीजर, करेक्टर, फ्लोटिंग–पोइंट नम्बर और स्ट्रींग।

उदाहरण :–

256 // डेसिमल इंटीजर

34.12 // फ्लोटिंग–पोईन्ट नम्बर

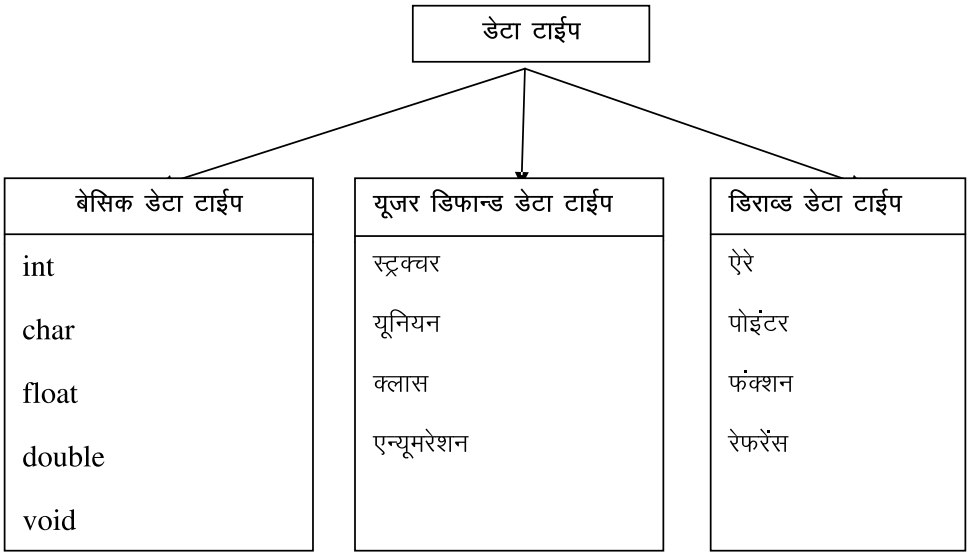
025 // ऑक्टल इंटीजर

“ Hello”// स्ट्रींग कांस्टेट

‘ Z ’// करेक्टर कांस्टेट

6.7 बेसिक डेटा टाईप

डेटा टाईप को चित्र 1.2 में दर्शाये अनुसार विभाजित किया जा सकता है–



चित्र 1.2 डेटा टाईप

बेसिक डेटा टाईप के साथ विभिन्न प्रकार के मोडिफायर प्रयोग किये जाते है। जिनको उनके नाम से पहले लिखे जाते है। void डेटा टाईप के साथ इनका प्रयोग नही किया जाता है। मोडिफायर का प्रयोग करेक्टर और इन्टीजर बेसिक डेटा टाईप के साथ किया जाता हैं।long मोडिफायर का प्रयोग double के साथ किया जाता है। मोडिफायर निम्न प्रकार के होते है

- signed
- unsigned
- short
- long

बेसिक डेटा टाईप और मोडिफायर के सभी कोम्बीनेशन की लिस्ट उनके साइज और रेंज टेबल 6.2 में दर्शाया गया है।

टेबल 6.2 बेसिक डेटा टाईप के साइज और रेंज

डेटा टाईप	साइज	रेंज
char	1	-128 to 127
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2	-32768 to 32767
unsigned int	2	0 to 65535

signed int	2	-32768 to 32767
short int	1	-128 to 127
long int	4	-2147483648 to 2147483647
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to1.1E+4932

6.8 यूजर डिफाइन्ड डेटा टाईप

स्ट्रक्चर और यूनियन

वास्तविक समस्याओं के निराकरण के लिए बेसिक डेटा टाईप पर्याप्त नहीं होते हैं। बेसिक डेटा टाईप और अन्य डेटा टाईप के समूह को स्ट्रक्चर कहा जाता है। स्ट्रक्चर का सिन्टेक्स इस प्रकार होता है

```
struct स्ट्रक्चर का नाम
{
.....
डेटा टाईप मेम्बर1;
डेटा टाईप मेम्बर2;
.....
};
```

एक स्टूडेन्ट का उदाहरण लेते हैं, जिसके कई ऐट्रिब्यूटस् होते है जैसे नाम, उम्र, प्रतिशत इत्यादि।

```
struct student
{
   char name[20];
    int age;
    float percentage;
};
struct student student1, student2;
```

यहाँ student1 और student2 यूजर डिफाइन्ड डेटा टाईप 'student'के वेरिएबल है।

यूनियन भी स्ट्रक्चर की तरह होता है। लेकिन उनमें एक अन्तर होता है। स्ट्रक्चर टाईप की साइज उनके सभी मेम्बर के टाईप की साईजों के योग के बराबर होती है। जबकि यूनियन का साइज उसके सबसे बड़े मेम्बर के टाईप के साइज के बराबर होती है।

उदाहरण :–

```
union item
{
   int m;
  float x;
  char c;
}it1;
```

यहाँ एक वेरिएबल it1 की घोषणा करता है। जिसका टाईप item है। इस यूनियन मे तीन मेम्बर है। हर एक का अलग डेटा टाईप है। यद्यपि केवल एक समय में एक ही का प्रयोग कर सकते है। वेरिएबल it1 मैमोरी में चार बाइट का स्थान लेगा क्योंकि इसका सबसे बड़ा सदस्य float टाईप का है जो वेरिएबल x है। अगर हम item को स्ट्रक्चर डिफाईन्ड करते है तब it1 वेरिएबल मेमौरी में सात बाइट का स्थान लेगा। हम कह सकते है कि यूनियन मेमौरी की बचत करने वाला स्ट्रक्चर का विकल्पहै।

क्लास

क्लास C++ भाषा का महत्वपूर्ण फिचर है। किसी अन्य बेसिक डेटा टाईप की तरह क्लास टाईप के वेरिएबल की घोषणा कर सकते है। क्लास के वेरिएबल को ऑब्जेक्ट कहा जाता है। हम क्लास की विस्तार से चर्चा अध्याय 9 में करेगें।

एन्यूमरेटेड डेटा टाईप

यह एक तरीका है जिसके द्वारा नामों को संख्याओं के साथ जोड़ा जाता है। enum कीवर्ड से 0, 1, 2 इत्यादि संख्याओं को नामों की लिस्ट के साथ जोड़ा जाता है।

उदाहरण :–

```
enum color{red, green, blue};
```

स्वतः ही red को 0, green को 1 और blue को 2 की संख्या प्रदान हो जाती है।

हम डिफाल्ट वेल्यूज को बाह्य रूप से इंटीजर वेल्यूज को एन्यूमरेटरस को प्रदान करके ओवरराइड भी कर सकते है।

उदाहरण :–

```
enum color{red, green=3, blue=8};
```

यहाँ red को 0 संख्या स्वतः ही प्रदान हुई है।

6.9 डिराइव्ड डेटा टाईप

ऐरे

यह एक ही प्रकार के एलिमेन्टस् का समूह है।

उदाहरण :–

```
int number[5]={2, 7, 8, 9, 11};
```

यहाँ number एक ऐरे है जिसका साइज 5 है और उसमें पाँच इंटीजर टाईप के ऐलिमेन्टस् है।

फंक्शन

फंक्शन प्रोग्राम का एक भाग होता है। जो एक कार्य करने के लिए प्रयोग किया जाता है। एक प्रोग्राम को फंक्शनस में विभाजित करना प्रोग्रामिंग भाषा के मुख्य सिद्वान्तों में से एक है। प्रोग्राम में विभिन्न स्थानों पर कॉलिंग और उपयोग करके प्रोग्राम के आकार को कम करता है। हम फंक्शन की विस्तार से चर्चा अध्याय 8 में करेंगें।

पोइंटर

पोइंटर एक वेरिएबल होता है जो एक दूसरे वेरिएबल के एड्रेस को रखता है।

उदाहरण :–

```
int x=5;        //integer variable
int *ptr;       // integer pointer variable
ptr= &x;        //address of x assigned to ptr
*ptr=10;        //the value of x is changed from 5 to 10
```

रेफरेंस टाईप

रेफरेंस टाईप के वेरिएबल को रेफरेंस वेरिएबल कहा जाता है।

उदाहरण :–

```
int x=10;
int & y=x;
```

यहाँ x एक इंटीजर टाईप का वेरिएबल हैऔर y उसका एक उपनाम है।

```
cout << x;
cout << y;
```

दोनों 10 प्रिन्ट करेगें। स्टेटमेंट

```
x=x+5;
```

x और y दोनों की वेल्यू को 15 में बदल देगा।

6.10 टाईप कम्पेटिबिलिटि

C++ टाईपकम्पेटिबिलिटि के संदर्भ में C से बहुत ही स्ट्रीक्ट है। int, short int और long intतीनों अलग–अलग टाईप है। इनको टाईपकास्ट करना आवश्यक है जब इनकी वेल्यूज एक दूसरे को प्रदान की जाती है। एक ऑपरेशन के ऑपरेडन्स् का टाईप ऑपरेशन के साथ टाईपकम्पेटिबिटिबल होना आवश्यक है। टाईप कम्पेटिबिलिटि को हासिल करने के लिए दो प्रणालीहै।

(i) बाह्य टाईप कनवर्जन

टाईप कास्ट ऑपरेटर का प्रयोग करते हुए वेरिएबल या एक्सप्रेशनस् का बाह्य टाईप कनवर्जन किया जाता है।

प्रोग्राम 6.2बाह्य टाईप कनवर्जन

```
#include<iostream>
using namespace std;
int  main()
{
    int i=5;
    float f=30.57;
    cout<<"i="<<i;
    cout<<"\nf="<<f;
    cout<<"\nfloat(i)="<<float(i);
    cout<<"\nint(f)="<<int(f);
    return 0;
}
```

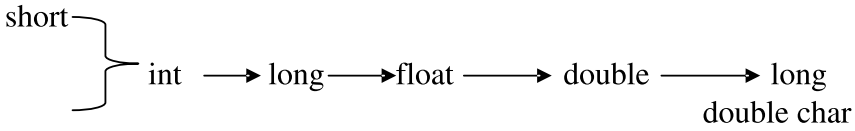
प्रोग्राम 6.2 का आउटपुट होगा–

```
i=5
f=30.57
float(i)=5
int(f)=30
```

(ii) स्वतः टाईप कनवर्जन

जब एक एकसप्रेशन में मिश्रित डेटा टाईप होते हैतब कम्पाइलर नियमानुसार स्वतः ही टाईप कनवर्जन कर देता है। छोटे टाईप का बडे. टाईप में स्वतः ही कनवर्जन होने का नियम होता

है। जब भी charया short int किसी एक्सप्रेशन में होते है तब इनको int में कनवर्जन कर दिया जाता है। इसे इंटिग्रल वाइडनिग कनवर्जन कहा जाता है। निम्न आकृति स्वतः टाईप कनवर्जन के नियम को दर्शाता है।



6.11 वेरिएबल की घोषणा

C में सभी वेरिएबल की घोषणा प्रोग्राम के शुरूआत में की जाती है। C++ में भी यह सत्य है लेकिन वेरिएबल की घोषणा प्रोग्राम में किसी भी जगह पर करने की अनुमति होती है।

उदाहरण :–

```
int main()
{
        int x,y;                    //variable declaration
        cin>>x>>y;
        int sum=x+y; //variable declaration
        cout<<sum;
}
```

महत्वपूर्ण बिंदु

- C++ फंक्शनस का संग्रह है।
- हर एक C++ प्रोग्राम में main ( ) फक्शन अनिवार्य है।
- C++ प्रोग्राम में स्टेटमेन्ट अर्द्धविराम (;) से समाप्त होते हैं।
- स्टेटमेंट से पहले # चिन्ह् लगा हो उनको प्रीप्रोसेसर डाइरेक्टीव स्टेटमेन्ट कहते है।
- प्रोग्राम की सबसे छोटी इकाई को टोकन्स कहा जाता है।
- रिजर्व वर्डस जिनका अर्थ प्रोग्रामर के द्वारा बदला नहीं जा सकता है उन्हें कीवर्डस कहा जाता है।
- वेरिएबल, फंक्शन, ऐरे, क्लास इत्यादि के नाम जो प्रोग्रामर के द्वारा दिये जाते है उन्हें आइडेंटीफयर कहा जाता है।

- फिक्सड वेल्यू जो प्रोग्राम के एक्जीक्यूशन के दौरान बदलती नहीं है उन्हें कांस्टेट कहा जाता है।
- बेसिक डेटा टाईप और अन्य डेटा टाईप के समूह को स्ट्रक्चर कहा जाता है।
- एक ही प्रकार के एलिमेन्टस् का समूह को ऐरे कहा जाता है।
- पोइंटर एक वेरिएबल होता है जो एक दूसरे वेरिएबल का एड्रेस रखता है।
- C++ टाईपकम्पेटिबिलिटि के संदर्भ में C से बहुत ही स्ट्रीक्ट है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्रश्न 1. एक C++ प्रोग्राम की संरचना में कौनसा अनुभाग होता हैं?

(अ) क्लास की घोषणा (ब) मेम्बर फंक्शन की परिभाषा

(स) main( ) फंक्शन (द) उपरोक्त सभी

प्रश्न 2. एक लाईन को कमेन्ट करने के लिए किस चिन्ह् प्रयोग किया जाता है

(अ) \\ (ब) // (स) || (द) !!

प्रश्न 3. प्रिप्रोसेसर डाइरेक्टीव स्टेटमेंट से पहले किस चिन्ह् प्रयोग किया जाता है?

(अ) $ (ब) # (स) & (द) *

प्रश्न 4. C++ प्रोग्राम की कम्पाइलिंग व लिकिंग के लिए किस कमांड का प्रयोगकिया जाता है?

(अ) g++ (ब) a++ (स) y++ (द) Z++

प्रश्न 5.इनमे से कोनसा एक टोकन है?

(अ) कीवर्ड (ब) आइडेंटिफायर

(स) ऑपरेटर (द) उपरोक्त सभी

प्रश्न 6 इनमे से कोनसा एक बेसिक डेटा टाईप नहीं है?

(अ) int (ब) char

(स) float (द) class

अति लघूउत्तरात्मक प्रश्न

प्रश्न 1. टोकन्स क्या होते है ?

प्रश्न 2. कीवर्ड क्या होते है ?

प्रश्न 3. आइडेंटिफायर क्या होते है ?

प्रश्न 4. कांस्टेट क्या होते है ?

प्रश्न 5. स्ट्रक्चर और यूनियन में क्या अन्तर है ?

लघूउत्तरात्मक प्रश्न

प्रश्न 1. डेटा टाईप के वर्गीकरण का वर्णन करें ?

प्रश्न 2. ऐन्यूमरेटेड डेटा टाईप किसे कहते है ?

प्रश्न 3. रेफरेंस टाईप किसे कहते है ?

निबंधात्मक प्रश्न

प्रश्न 1. C++ प्रोग्राम की कम्पाइलिंग और लिंकिग लिनक्स ऑपरेटिंग सिस्टम पर करने का वर्णन कीजिए ?

प्रश्न 2. स्वतः और बाह्य टाईप कनवर्जन का उदाहरण सहित वर्णन करें ?

उत्तरमाला

1: द 2: ब 3: ब 4: अ 5: द 6: द

अध्याय –7

ऑपरेटर, एक्सप्रेशन और कन्ट्रोल स्ट्रक्चर

7.1 C++ में ऑपरेटर

C के सभी ऑपरेटर C++ में वैध है। इनके अलावा C++ में कुछ नये ऑपरेटर जोडे. गये है जो निम्नलिखित है

- Insertion ऑपरेटर (<<) :– यह इसके दायी तरफ के वेरिएबल के कन्टेन्ट को आउटपुट स्क्रीन पर प्रिन्ट करता है।
- Extraction ऑपरेटर (>>) :– यह कीबोर्ड से वेल्यू लेता है और इसके दायी तरफ के वेरिएबल को प्रदान करता है।
- स्कोप रिजोलूशन ऑपरेटर (::) :– C++ एक ब्लोक स्ट्रक्चरड भाषा है। एक वेरिएबल के नाम को अलग –अलग ब्लोक में प्रयोग कर सकते है। वेरिएबल का स्कोप इसकी घोषणा की जगह और ब्लोक के अंत के बीच में होता है। एक वेरिएबल जिसकी घोषणा ब्लोक के अन्दर की गयी है वह उस ब्लोक के लिए लोकल होता है। स्कोप रिजोलूशन ऑपरेटर का प्रयोग वेरिएबल के ग्लोबल वर्जन को एक्सेस करने के लिए किया जाता है।

प्रोग्राम 7.1 :– स्कोप रिजोलूशन ऑपरेटर

```
#include<iostream>
using namespace std;
int x=10;          //global variable
int  main()
{
   int x=20;       //x re-declared , local to main
   {
          cout<<"Inner block\n";
          int x=30;       //x declared again, local to inner block
          cout<<"x="<<x<<"\n";
          cou<<"::x="<<::x<<"\n";
    }
   cout<<"Outer block\n";
```

```
    cout<<"x="<<x<<"\n";
    cout<<"::x="<<::x<<"\n";
    return 0;
}
```

प्रोग्राम 7.1 का आउटपुट होगा–

```
Inner block
x=30
::x=10
Outer block
x=20
::x=10
```

- new ऑपरेटर :– यह ऑपरेटर पर्याप्त मात्रा में डाटा ऑब्जेक्ट को मेमौरी प्रदान करता है।

```
int *p = new int;
```

उपरोक्त स्टेटमेंट इंटीजर डाटा ऑब्जेक्ट को पर्याप्त मात्रा में मेमौरी प्रदान करता है।

- delete ऑपरेटर :– यह ऑपरेटर मेमौरी को पुनः आवंटितकरता है। जब डेटा ऑब्जेक्ट की आगे आवश्यकता नहीं होती है। इससे मुक्त की गयी मेमौरी दूसरे प्रोग्रामों के लिए पुनः उपयोग में लायी जा सकती है।

उदाहरण :–

```
delete p;
```

उपरोक्त स्टेटमेंट मेमौरी जिसे पोइंटर p द्वारा अंकित किया गया है उसे पुनः आवंटितकरता है।

7.2 एक्सप्रेशन और उसके प्रकार

एक एक्सप्रेशन ऑपरेटर, कांस्टेंट और वेरिएबल का कॉम्बीनेशन है जो भाषा के नियम के अनुसार व्यवस्थित होता है। एक्सप्रेशन के निम्न प्रकार होते है।

- कांस्टेंट एक्सप्रेशन
- इंटीग्ररल एक्सप्रेशन
- फ्लोट एक्सप्रेशन
- पोइंटर एक्सप्रेशन
- रिलेशनल एक्सप्रेशन
- लोजिकल एक्सप्रेशन
- बीटवाइज एक्सप्रेशन

- कांस्टेंट एक्सप्रेशन :– इसमें केवल कांस्टेंट वेल्यूज होती है।

उदाहरण :–20+10*5.2

- इंटीग्ररल एक्सप्रेशन :– जो एक्सप्रेशन स्वतः और बाह्य टाईप कनवर्जन के बाद इंटीजर परिणाम देते है ।

उदाहरण :– x+y*10

x+'a'

5+int(7.5)

जहाँ x और y इंटीजर वेरिएबल है।

- फ्लोट एक्सप्रेशन :– जो एक्सप्रेशन सभी तरह के टाईप कनवर्जन के बाद फ्लोट टाईप परिणाम देते है।

उदाहरण :– a+b/5

7+float(10)

जहाँ a और bफ्लोट टाईप के वेरिएबल है।

- पोइंटर एक्सप्रेशन :– पोइंटर एक्सप्रेशन का परिणाम एड्रेस वेल्यू होता है।

उदाहरण :–ptr=&x;

ptr+1

जहाँ x एक वेरिएबल है और ptr एक पोइंटर है।

- रिलेशनल एक्सप्रेशन :– जो एक्सप्रेशन बूलियन टाईप का परिणाम देतें है। जो सत्य और असत्य हो सकता है।

उदाहरण :–x<=y

a==b

रिलेशनल एक्सप्रेशन को बूलियनएक्सप्रेशन भी कहा जाता हैं ।

- लोजिकल एक्सप्रेशन :– जो एक्सप्रेशन दो या दो से अधिक रिलेशनल एक्सप्रेशन को जोड़ता है और बूलियन टाईप का परिणाम देतें है।

उदाहरण :– x>y && x==5

a==20 || y==10

- बीटवाइज एक्सप्रेशन :– इस तरह के एक्सप्रेशन को बिट स्तर के डेटा मेनुपूलेशन के लिए प्रयोग किया जाता है। इनका उपयोग बिट्स की टेस्टींग और शिफ्टींग के लिए किया जाता है।

उदाहरण :–

a<<3 // तीन बिट्स को बायी तरफ शिफ्ट करता है।

x>>1 // एक बिट को दायी तरफ शिफ्ट करता है।

7.3 विशेष असाइनमेन्ट एक्सप्रेशनस

- चैन्ड असाइनमेन्ट :–

a=b=10;

पहले 10 वेल्यू b को प्रदान की जाती है उसके बाद a को।

- एम्बेडेड असाइनमेन्ट :–

a=(b=20)+5;

(b=20) एक असाइनमेन्ट एक्सप्रेशन है जिसे एम्बेडेड असाइनमेन्ट कहा जाता है। यहाँ पर वेल्यू 20, b को दी जाती है और उसके बाद परिणाम 25, a को दिया जाता है।

- कम्पाउंड असाइनमेन्ट :– यह असाइनमेन्ट ऑपरेटर और एक बाइनरी आरिथमैटिक ऑपरेटर का सयुंक्त रूप है।

उदाहरण :–

a=a+5; को a+=5; के रूप में भी लिख सकते है।

+= ऑपरेटर को कम्पाउंड असाइनमेन्ट ऑपरेटर या शॉर्ट हेंड असाइनमेन्ट ऑपरेटर कहा जाता है।

7.4 ऑपरेटर प्रिसिडेंस और एसोसिएटिविटी

अगर एक से ज्यादा ऑपरेटर किसी एक्सप्रेशन में हो तब,C++ भाषा में ऑपरेटर की प्राथमिकता के लिए परिभाषित नियम होते है। उच्च प्राथमिकता वाले ऑपरेटर निम्न प्राथमिकता वाले ऑपरेटर से पहले संपादित होते है। इस नियम को ऑपरेटर प्रिसिडेंस कहा जाता है।

ऑपरेटर की एसोसिएटिविटी

अगर दो या दो से अधिक ऑपरेटर एक समान प्रिसिडेंस के एक ही एक्सप्रेशन में होते है तो जिस आर्डर में वे संपादित होते है उसे ऑपरेटर की एसोसिएटिविटी कहते है। C++ ऑपरेटर की सम्पूर्ण लिस्ट उनकी प्रिसिडेंस और एसोसिएटिविटी के साथ टेबल 7.1 में दी गयी है।

टेबल 7.1 ऑपरेटर प्रिसिडेस और एसोसिएटिविटी

ऑपरेटर प्रिसिडेस	एसोसिएटिविटी
::	बायें से दायें
->, ., (), [], ++, --, ~, !, unary+, unary-, unary*	बायें से दायें
Unary &, (type), sizeof, new, delete	दायें से बायें
*, /, %	बायें से दायें

+, -	बायें से दायें
<<, >>	बायें से दायें
<, <=, >, >=	बायें से दायें
==, !=	बायें से दायें
&	बायें से दायें
^	बायें से दायें
|	बायें से दायें
&&	बायें से दायें
||	बायें से दायें
?:	बायें से दायें
=, *=, /=, %=, +=	दायें से बायें
<<=, >>=, &=, ^=, |=, ,(comma)	बायें से दायें

7.5 कन्ट्रोल स्ट्रक्चर

तीन तरह के कन्ट्रोल स्ट्रक्चर होते है।

- सिकवेंश स्ट्रक्चर
- सलेक्शन स्ट्रक्चर
- लूप स्ट्रक्चर

C++ में सभी बेसिक कन्ट्रोल स्ट्रक्चर होते है। और उनको विभिन्न प्रकार के कन्ट्रोल स्टेटमेन्ट के द्वारा लागू किया गया है।

- सिकवेंश स्ट्रक्चर :– स्टेटमेन्ट को जैसे प्रोग्राम में लिखा जाता है वैसे ही संपादित किया जाता हैं।

उदाहरण :–

```
--------
स्टेटमेन्ट 1
स्टेटमेन्ट 2
--------
```

- सलेक्शन स्ट्रक्चर :– दो या उससे अधिक संपादन के पथ जिनमें से एक को चुना जाता है अगर शर्त पूरी होती है।

उदाहरण :–

if स्टेटमेन्ट

if(expression is true)

```
{
        statements;
}
```

if-else स्टेटमेन्ट

```
if(expression is true)
{
        statements;
}
else
{
        statements;
}
```

switch स्टेटमेन्ट

```
switch(expression)
{
        case 1: statements;
                break;
        case 2: statements;
                break;
        case 3: statements;
                break;
        default : statements;
}
```

- लूप स्ट्रक्चर :– स्टेटमेन्ट शून्य या उससे अधिक बार संपादित होते है ।

उदाहरण :–

for स्टेटमेन्ट

for लूप का प्रयोग किया जाता है। जब किसी कार्य को पूर्व निर्धारित संख्या के बराबर दोहराया जाता है।

```
for(initial value; test condition; increment/decrement)
{
    statements;
}
```

while स्टेटमेन्ट

while लूप के अन्दर के स्टेटमेन्ट जब तक कंडिशन सत्य है तब तक संपादित होता है। इसे प्रि–टेस्ट कंडिशन लपू भी कहा जाता है।

```
while(condition is true)
{
        statements;
}
```

do-while स्टेटमेन्ट

do-while लूप को पोस्ट टेस्ट कंडिशन लूप कहा जाता है। यह लूप कम से कम एक बार तो संपादित होता है।

```
do
{
        statements;
}while(condition is true);
```

महत्वपूर्ण बिंदु

- C के सभी ऑपरेटर C++ में वैध है।
- C++ एक ब्लोक स्ट्रक्चरड भाषा है।
- एक एक्सप्रेशन ऑपरेटर ,कांस्टेंट और वेरिएबल का कॉम्बीनेशन है जो भाषा के नियम के अनुसार व्यवस्थित होता है।
- उच्च प्राथमिकता वाले ऑपरेटर निम्न प्राथमिकता वाले ऑपरेटर से पहले संपादित होते है।
- C++ में सभी बेसिक कन्ट्रोल स्ट्रक्चर होते है और उनको विभिन्न प्रकार के कन्ट्रोल स्टेटमेन्ट के द्वारा लागू किया गया है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्नः

प्रश्न 1. इनमें से कौनसा ऑपरेटर इसके दायी तरफ के वेरिएबल के कन्टेन्ट को आउटपुट स्क्रीन पर प्रिन्ट करता है?

(अ) << (ब) >> (स) :: (द) &

प्रश्न 2. इनमें से कौनसा ऑपरेटर प्रयाप्त मात्रा में डाटा ऑब्जेक्ट को मेमौरी प्रदान करता है?

(अ) Insertion ऑपरेटर (ब) Extraction ऑपरेटर

(स) new ऑपरेटर (द) delete ऑपरेटर

प्रश्न 3. एक्सप्रेशनस **a=(b=20)+5**में वेरिएबल 'a' का क्या मान होगा?

(अ) 20 (ब) 25 (स) 5 (द) 30

प्रश्न 4. इनमे से कोनसा शॉर्ट हेंड असाइनमेन्ट ऑपरेटर है?

(अ) += (ब) -= (स) *= (द) उपरोक्त सभी

प्रश्न 5. सलेक्शन स्ट्रक्चर को किस कन्ट्रोल स्टेटमेन्ट के द्वारा लागू किया गया है?

(अ) if स्टेटमेन्ट (ब) if-else स्टेटमेन्ट

(स) switch स्टेटमेन्ट (द) उपरोक्त सभी

प्रश्न 6 लूप स्ट्रक्चर को किस कन्ट्रोल स्टेटमेन्ट के द्वारा लागू किया गया है?

(अ) for स्टेटमेन्ट (ब) while स्टेटमेन्ट

(स) do-while स्टेटमेन्ट (द) उपरोक्त सभी

अति लघूउत्तरात्मक प्रश्न

प्रश्न 1. ऑपरेटर प्रिसीडेंस को परिभाषित करें ?

प्रश्न 2. ऑपरेटर की संबद्वता को परिभाषित करें ?

प्रश्न 3. विभिन्न प्रकार के कन्ट्रोल स्ट्रक्चर क्या होते है ?

प्रश्न 4. एक्सप्रेशनस क्या होते हैं ?

लघूउत्तरात्मक प्रश्न

प्रश्न 1. स्कोप रिजोलुशन ऑपरेटर का क्या उपयोग है ?

प्रश्न 2. new और delete ऑपरेटर का क्या उपयोग है ?

प्रश्न 3. C++ में सलेकशन कन्ट्रोल स्ट्रक्चर कैसे लागू किया गया है वर्णन कीजिए ?

निबंधात्मक प्रश्न

प्रश्न 1. विभिन्न प्रकार के एक्प्रेशन का उदाहरण सहित वर्णन किजिए ?

प्रश्न 2. विभिन्न प्रकार के लूपिंग स्टेटमेंटों का वर्णन किजिए ?

उत्तरमाला

1: द 2: स 3: ब 4: द 5: द 6: द

अध्याय 8

C++ में फंक्शन

8.1 परिचय

फंक्शन प्रोग्राम का एक भाग होता है जिसे किसी कार्य के लिए प्रयोग किया जाता है। एक प्रोग्राम का फंक्शनस में विभाजित करना प्रोग्रामिंग भाषा के मुख्य सिद्वान्तों में से एक है। प्रोग्राम में विभिन्न स्थानों पर कॉलिंग और उपयोग करके प्रोग्राम के आकार का कम करता है। C++में फंक्शन को और अधिक विश्वसनीय और लचीला बनाने के लिए कई नये फिचर जोड़े गये है।

8.2 फंक्शन प्रोटोटाईप

फंक्शन प्रोटोटाईप कम्पाइलर को फंक्शन के बारे में सूचना देता है। जैसे आरग्यूमेन्ट की संख्या और उनके टाईप और रिटर्न टाईप। फंक्शन प्रोटोटाईप एक कॉलिंग प्रोग्राम में घोषणा करने वाला स्टेटमेन्ट होता हैं और इसका सिन्टेक्स निम्न प्रकार का होता है।

```
type function_name(arguments-list);
```

उदाहरण :–

```
int sum(int a, int b);
```

फंक्शन की घोषणा में आरग्यूमेन्ट के नाम छदम वेरिएबल होते है और वे ऐच्छिक होते है।

```
int sum(int, int);
```

एक वैध फंक्शन की घोषणा है।

8.3 कॉल–बाई–रेफरेंस

कॉल–बाई–वेल्यू पेरामीटर भेजने वाली विधि में कॉल्ड फंक्शन कॉलिंग प्रोग्राम में वास्तविक पेरामीटर को कॉल्ड फंक्शन द्वारा फोरमल पेरामीटर में कॉपी किया जाता है। फोरमल पेरामीटर पर किये बदलाव कॉलिंग प्रोग्राम में नहीं होता है। कॉलिंग प्रोग्राम के वास्तविक पेरामीटर मे बदलाव के लिए हम कॉल–बाई–रेफरेंस पेरामीटर भेजने वाली विधि का प्रयोग करते हैं

उदाहरण :–

प्रोग्राम : 8.1 कॉल–बाई–रेफरेंस

```
#include<iostream>
using namespace std;
int  main()
{
```

```
int count=0;
void update(int &);
cout<<"count="<<count<<"\n";
update(count);
cout<<"count="<<count;
return 0;
}
void update(int &x)
{
        x=x+1;
}
```

प्रोग्राम 8.1 का आउटपुट होगा–

```
count=0
count=1
```

यहाँ पर कॉल्ड फंक्शन update में वेरिएबल x, main फंक्शन में वेरिएबल count का उपनाम बन जाता है। x की वेल्यू में एक की बढोतरी से main फंक्शन में count की वेल्यू बढ़ जाती है।

8.4 रिटर्न–बाई–रेफरेंस

एक फंक्शन रेफरेंस भी रिटर्न कर सकता है।

प्रोग्राम : 8.2 रिटर्न–बाई–रेफरेंस

```
#include<iostream>
using namespace std;
int main()
{
        int x=6,y=9;
        int & min(int &, int &);
        min(x,y)=-1;
        cout<<"x="<<x<<"\n";
```

```
        cout<<"y="<<y;
        return 0;
}

int & min(int &a, int &b)
{
         if(a<b)
         return a;
         else
         return b;
}
```

प्रोग्राम 8.2 का आउटपुट होगा–

x=-1

y=9

8.5 फंक्शन ओवरलोडिंग

फंक्शन के एक समान नाम लेकिन उनकी आरग्यूमेन्ट लिस्ट में भिन्नता और अलग–अलग कार्य करना इसे फंक्शन ओवरलोडिंग कहा जाता है। सही फंक्शन को कॉल करना उसके आरग्यूमेन्ट की संख्या और टाईप पर निर्भर करता है न की उसके रिटर्न टाईप पर।

प्रोग्राम : 8.3 फंक्शन ओवरलोडिंग

```
#include<iostream>
using namespace std;
int sum(int, int);
int sum(int, int, int);
int main()
{
        cout<<"Sum of two numbers is "<<sum(5,10);
        cout<<"\n";
        cout<<"Sum of three numbers is "<<sum(10,20,30);
        return 0;
```

```
}
int sum(int x, int y)
{
        return(x+y);
}
int sum(int a, int b, int c)
{
        return(a+b+c);
}
```

प्रोग्राम 8.3 का आउटपुट होगा–

```
Sum of two numbers is 15
Sum of three numbers is 60
```

उपरोक्त प्रोग्राम में sum() फंक्शन को ओवरलोड किया गया है। जब हम sum() फंक्शन को दो आरग्यूमेन्ट भेजते है तब दो आरग्यूमेन्ट वाला फंक्शन कॉल होता है। जब हम sum() फंक्शन को तीन आरग्यूमेन्ट भेजते है तब तीन आरग्यूमेन्ट वाला फंक्शन कॉल होता है।

8.6 इनलाईन

जब किसी फंक्शन को कॉल किया जाता हैं संपादन का कन्ट्रोल कॉलिंग फंक्शन से कॉल्ड फंक्शन को चला जाता हैं। कॉल्ड फंक्शन संपादन होने के बाद संपादन का कन्ट्रोल पुनः कॉलिंग फंक्शन को चला जाता हैं। जब फंक्शन का आकार छोटा होता है तब इस प्रकिया में काफी समय बर्बाद हो जाता है। इस समस्या का समाधान इनलाईन फंक्शन है। एक इनलाईन फंक्शन का विस्तार इसे कॉल करने पर होता है। कम्पाइलर फंक्शन कॉल स्टेटमेंट को उसकी बॉडी से विस्थापित कर देता है।

महत्वपूर्ण बिंदु

- एक फंक्शन प्रोग्राम का भाग होता है जिसे किसी कार्य के लिए प्रयोग किया जाता है।
- फंक्शन प्रोटोटाईप कम्पाइलर को फंक्शन के बारे में सूचना देता है।
- एक फंक्शन रेफरेंस भी रिटर्न कर सकता है।
- फंक्शन के एक समान नाम लेकिन उनकी आरग्यूमेन्ट लिस्ट में भिन्नता और अलग–अलग कार्य करना इसे फंक्शन ओवरलोडिंग कहा जाता है।
- एक इनलाईन फंक्शन का विस्तार इसे कॉल करने पर होता है जिसमें कम्पाइलर फंक्शन कॉल स्टेटमेंट को उसकी बॉडी से विस्थापित कर देता है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न :

प्रश्न 1. एक वैध फंक्शन की घोषणा कौनसी है?

(अ) int fun(int a, int b); (ब) int fun(int, int);

(स) अ. और ब. दोनों (द) इनमें से कोई नहीं

प्रश्न 2. पेरामीटर भेजने वाली किस विधि में कॉलिंग प्रोग्राम में वास्तविक पेरामीटर को कॉल्ड फंक्शन के फोरमल पेरामीटर में कॉपी किया जाता है

(अ) कॉल–बाई–रेफरेंस (ब) कॉल–बाई–वेल्यू

(स) कॉल–बाई–एड्रेस (द)) इनमें से कोई नहीं

प्रश्न 3. फंक्शन ओवरलोडिंग में सही फंक्शन को कॉल करना किस पर निर्भर नहीं करता है?

(अ) आरग्यूमेन्ट की संख्या (ब) आरग्यूमेन्ट के टाईप

(स) फंक्शन केरिटर्न टाईप (द) इनमें से कोई नहीं

प्रश्न 4. एक समान फंक्शन के नाम लेकिन उनकी आरग्यूमेन्ट लिस्ट में भिन्नता और अलग–अलग कार्य कर सकते है इसे कहा जाता है

(अ) फंक्शन ओवरलोडिंग (ब) ऑपरेटर ओवरलोडिंग

(स) क्लास ओवरलोडिंग (द) इनमें से कोई नहीं

अति लघूउत्तरात्मक प्रश्न

प्रश्न 1. फंक्शन किसे कहते है ?

प्रश्न 2. ऑपरेटर ओवरलोडिंग किसे कहते है ?

प्रश्न 3. इनलाईन फंक्शन किसे कहते है ?

लघूउत्तरात्मक प्रश्न

प्रश्न 1. फंक्शन प्रोटोटाईप किसे कहते है ?

प्रश्न 2. कॉल –बाई– वेल्यू और कॉल –बाई– रेफरेंस में क्या अन्तर है ?

प्रश्न 3. स्ट्रक्चरड प्रोग्रामिंग में फंक्शन के क्या फायदे है ?

निबंधात्मक प्रश्न

प्रश्न 1. दो वेल्यूज को आपस में अदला–बदली कॉल–बाई–रेफरेंस प्रणाली के द्वारा करने का प्रोग्राम लिखे ?

प्रश्न 2. वृत्त के क्षेत्रफल और आयत के क्षेत्रफल की गणना के लिए फंक्शन को ओवरलोड करने का प्रोग्राम लिखे ?

उत्तरमाला

1: स 2: ब 3: स 4: अ

अध्याय 9

क्लास और ऑब्जेक्ट

9.1 परिचय

क्लास ऑब्जेक्ट ऑरिएनटेड प्रोग्रामिंग भाषा का एक महत्वपूर्ण फीचर है। क्लास की अवधारणा को C में स्ट्रक्चर से लिया गया है। यह एक यूजर डिफान्ड टाईप को बनाने का और लागू करने का नया तरीका है। इस अध्याय में हम क्लास और ऑब्जेक्ट के विभिन्न अवधारणाओं पर चर्चा करेगें।

9.2 क्लास को परिभाषित करना

क्लास एक यूजर डिफान्ड टाईप है जो डेटा और फंक्शन को एक साथ बाँधे रखता है। क्लास की घोषणा में इसके डेटा और मेम्बर फंक्शन की घोषणा होती है। क्लास की घोषणा का प्रारूप इस प्रकार होता है –

```
class class_name
{
  private:
        variable declaration;
        function declaration;
public:
         variable declaration;
        function declaration;
};
```

क्लास के मेम्बर जिसकी घोषणा प्राईवेट अनुभाग में है उनको उसी क्लास के मेम्बर ही एक्सेस कर सकते है। क्लास के मेम्बर जिनकी घोषणा पब्लिकअनुभाग में हुई है उनको क्लास के बाहर से एक्सेस कर सकते है। स्वतः ही क्लास के मेम्बर प्राईवेट होते है। डेटा को क्लास के प्राईवेट अनुभाग में घोषित करना डेटा हाइडिंग कहा जाताहै औरयह ऑब्जेक्ट ऑरिएनटेड प्रोग्रामिंग भाषा का एक महत्वपूर्ण फिचर है।

क्लास का एक उदाहरण

```
class point
{
        int x,y;                    // private by default
```

```
public:
        void input(int a, int b);
        void output(void);
};
```

ऑब्जेक्ट की घोषणा

बेसिक डेटा टाईप की तरह हम क्लास टाईप के वेरिएबल की घोषणाकर सकते है। इन वेरिएबल को ऑब्जेक्ट कहा जाता है।

उदाहरण के लिए :–

```
point p, q;
```

यहाँ दो ऑब्जेट p और qघोषित कियेगये है।

क्लास के मेम्बर को एक्सेस करना

हम केवल क्लास के पब्लिक मेम्बरस को उस क्लास के ऑब्जेक्ट के द्वारा एक्सेस कर सकते है। पब्लिक मेंम्बर फंक्शन को एक्सेस करने का प्रारूप इस प्रकार है

```
object_name.function_name(arguments list);
```

उदाहरण के लिए

```
p.input(10,20);
```

ऑब्जेक्ट pके द्वारा input()फंक्शन को कॉलकिया गया है। स्टेटमेंट p.x=10; मान्य नहीं है क्योंकि x को प्राईवेट घोषित किया गया है और इसे केवल क्लास के मेंम्बर फंक्शन ही सीधे एक्सेस कर सकते है न की ऑब्जेक्ट के द्वारा ।

9.3 मेम्बर फंक्शन को परिभाषित करना

क्लास के मेंम्बर फंक्शन को क्लास के अन्दर और क्लास से बाहर परिभाषित किया जा सकता है।

क्लास के अन्दर

मेंम्बर फंक्शन की घोषणा को क्लास के अन्दर उसकी वास्तविक परिभाषा से विस्थापित किया जाता है। क्लास के अन्दर परिभाषित मेंम्बर फंक्शन इनलाईन फंक्शन माने जाते है।
उदाहरण के लिए :–

```
class point
{
        int x,y;
public:
        void input(int a, int b)
```

```
        {
                x=a;
                y=b;
        }
        void output(void)
        {
                cout<<"x="<<x<<"\n";
                cout<<"y="<<y;
        }
};
```

क्लास के बाहर

मेंम्बर फंक्शन जिनकी घोषणा क्लास के अन्दर की गयी हो उनको क्लास से बाहर अलग से परिभाषित करना होता है।

मेम्बर फंक्शन को परिभाषित करने के लिए प्रारूप :–

```
return_type class_name:: function_name(arguments)
{
        function body
}
```

यहाँ class_name दर्शाता है कि फंक्शन इस क्लास से सम्बंधित है।

उदाहरण के लिए :–

```
class point
{
        int x,y;
public:
        void input(int a, int b);
        void output(void);
};
void point :: input(int a, int b)
        {
                x=a;
                y=b;
        }
```

```
        void point :: output(void)
{
        cout<<“x=”<<x<<“\n”;
        cout<<“y=”<<y;
}
```

प्रोग्राम 9.1 क्लास के साथ एक प्रोग्राम :–

```
#include<iostream>
using namespace std;
class point
{
        int x,y;
public:
        void input(int a, int b);
        void output(void);
};
void point :: input(int a, int b)
        {
                x=a;
                y=b;
        }
        void point :: output(void)
        {
                cout<<“x=”<<x<<“\n”;
                cout<<“y=”<<y;
        }
int main()
{
        point p;
        p.input(5,10);
        p.output();
return 0;
}
```

प्रोग्राम 9.1 का आउटपुट होगा–

x=5

y=10

9.4 एक्सेस मोडिफायरस

public और private कीवर्डस को एक्सेस मोडिफायर्स कहा जाता है। चूकि ये क्लास के मेंम्बर को एक्सेस करने की प्रणाली को नियंत्रित करते है। क्लास के पब्लिक मेंम्बर को क्लास के बाहर से एक्सेस किया जा सकता हैं सामान्यतः क्लास के मेम्बर फंक्शन को पब्लिक अनुभाग में रखा जाता है। क्लास के प्राईवेट मेंम्बर क्लास के बाहर से एक्सेस नहीं किये जा सकतें है। यहाँ तक कि उस क्लास के ऑब्जेक्ट के द्वारा भी नहीं किये जा सकते हैं। सामान्यतः क्लास के डेटा मेंम्बर को प्राईवेट अनुभाग में रखा जाता हैं।

9.5 क्लास के भीतर ऐरे

ऐरे भी क्लास के डेटा मेंम्बर के रूप में हो सकते है।

उदाहरण के लिए :–

प्रोग्राम 9.2 क्लास के भीतर ऐरे

```
#include<iostream>
using namespace std;
class data
{
        int a[5];
public:
        void getdata(void);
        void showdata(void);
};
void data :: getdata(void)
{
        cout<<“Enter the elements of array\n”;
        for(int i=0; i<5; i++)
        {
          cin>>a[i];
        }
}
void data :: showdata(void)
```

```
{
        cout<<"Array elements are\n";
        for(int i=0; i<5; i++)
        cout<<a[i]<<"\t";
}
int main()
{
    data d;
    d.getdata();
    d.showdata();
return 0;
}
```

प्रोग्राम 9.2 का आउटपुट होगा–

```
Enter the elements of array
6       5       9       8       1
Array elements are
6       5       9       8       1
```

9.6 स्टेटिक डेटा मेंम्बर

क्लास के डेटा मेंम्बर को स्टेटिक के रूप में भी घोषित किया जा सकता है। स्टेटिक डेटा मेंम्बर की विशेषताएँ इस प्रकार है –

- इसकी प्रांरभिक वेल्यू शून्य होती है जब इसके क्लास का पहला ऑब्जेक्ट बनाया जाता है।
- केवल एक ही प्रतिलिपी इस डेटा मेंम्बर की बनती है और इसे क्लास के सभी ऑब्जेक्ट साझा करते है।
- चूंकि यह सम्पूर्ण क्लास के साथ जुड़ा हुआ है इसे क्लास वेरिएबल भी कहा जाता है।

प्रोग्राम 9.3 स्टेटिक डेटा मेंम्बर

```
#include<iostream>
using namespace std;
class data
{
        static int x;
```

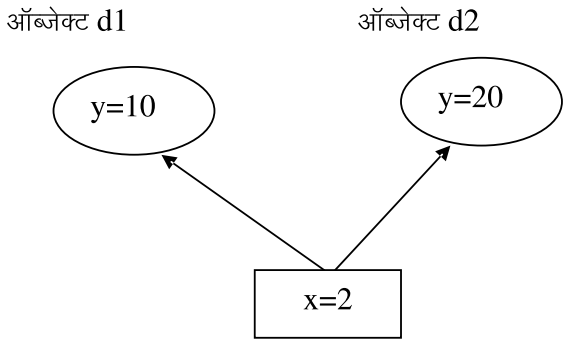
```
        int y;
        public:
        void getdata(int a)
        {
                y=a;
                x++;
        }
        void show_x(void)
        {
                cout<<"x="<<x<<"\n";
        }
};
int data :: x;   // static member definition
int main()
{
        data d1, d2;    // x is initialized to zero
        d1.show_x();
        d2.show_x();
        d1.getdata(10);
        d2.getdata(20);
        cout<<"After reading data"<<"\n";
        d1.show_x();
        d2.show_x();
        return 0;
}
```

प्रोग्राम 9.3 का आउटपुट होगा–

```
x=0
x=0
After reading data
x=2
x=2
```

उपरोक्त प्रोग्राम में जब ऑब्जेक्ट घोषित किये जाते है तब स्टेटिक डेटा मेंम्बर x की प्रांरभिक वेल्यू शून्य हो जाती है। हर बार जब फंक्शन कॉल होता है तब x की वेल्यू में एक की वृद्वि हो जाती है। चूंकि वेरिएबल x को दोनों ऑब्जेक्ट के बीच साझा किया गया है। x का मान हर बार 2 प्रिन्ट हुआ है। स्टेटिक डेटा मेंम्बर को प्रारंभिक वेल्यू भी दी जा सकती है जब इसे क्लास के बाहर परिभाषित किया जाता है। जैसे

int data :: x=5;



चित्र **9.1** स्टेटिक डेटा मेंम्बर

9.7 स्टेटिक मेंम्बर फंक्शन

स्टेटिक कीवर्ड के साथ घोषित मेंम्बर फंक्शन को स्टेटिक मेंम्बर फंक्शन कहा जाता है। स्टेटिक मेंम्बर फंक्शन की निम्नलिखित विशेषताएँ होती है

- यह केवल क्लास के दूसरे स्टेटिक डेटा मेंम्बर और मेंम्बर फंक्शन को एक्सेस कर सकते है।
- इनको क्लास के नाम की सहायता से कॉल किया जाता है।

प्रोग्राम 9.4 स्टेटिक मेंम्बर फंक्शन

```
#include<iostream>
using namespace std;
class test
{
        int x;
        static int y;
public:
        void set_xy(int a)
        {
                x=a;
```

```
        y++;
    }
    void show_x(void)
    {
        cout<<"x="<<x<<"\n";
    }
    staic void show_y(void)
    {
        cout<<"y="<<y;
    }
};
int test :: y;
int main()
{
test t1, t2;
t1.set_xy(10);
t2.set_xy(20);
t1.show_x();
t2.show_x();
test::show_y();        // calling static function
return 0;
}
```

प्रोग्राम 9.4 का आउटपुट होगा–

```
x=10
x=20
y=2
```

9.8 फ्रेंड फंक्शन

जैसा की हम जानते है कि क्लास के प्राईवेट मेंम्बर को क्लास के बाहर से एक्सेस नहीं किये जा सकते है। एक फ्रेंड फंक्शन क्लास के प्राईवेट डाटा को उस क्लास के ऑब्जेक्ट के जरिए एक्सेस कर सकते हैं। जब एक फंक्शन दो क्लासों में एक समान हो, सामान्यतः हम उस फंक्शन को दोनों के लिए फ्रेंड बना लेते है। इस फंक्शन को friend कीवर्ड के साथ घोषित किया जाता है। एक फ्रेंड फंक्शन की निम्नलिखित विशेषताएँ होती है

- इसे सामान्य फंक्शन की तरह कॉल किया जाता है।

- क्लास के ऑब्जेक्ट की सहायता से कॉल नहीं किया जा सकता हैं।
- यह क्लास के मेंम्बर को केवल उस क्लास के ऑब्जेक्ट की सहायता से एक्सेस कर सकता है।
- इसे क्लास के अन्दर कही भी घोषित किया जा सकता है।
- सामान्यतः इसके आरग्यूमेन्ट ऑब्जेक्ट होते है।

प्रोग्राम 9.5 फ्रेंड फंक्शन

```
#include<iostream>
using namespace std;
class test
{
        int  x,y;
        public:
        void getdata(int a, int b)
        {
                x=a;
                y=b;
        }
        friend int sum(test t);
};
        int sum(test t)
        {
                return(t.x+t.y);
        }
        int main()
        {
                test q;
                q.getdata(10,20);
                cout<<"Sum="<<sum(q);
                return 0;
        }
```

प्रोग्राम 9.5 का आउटपुट होगा–

```
Sum=30
```

फ्रेंड क्लास

एक क्लास का मेंम्बर फंक्शन दूसरी क्लास का फ्रेंड फंक्शन हो सकता है।

उदाहरण के लिए :–

```
class A
{
void fun();     // member function of A
};
class B
{
- - - - - - -
friend void A::fun();
- - - - - - - -
};
```

ध्यान दे कि क्लास B में फ्रेंड फंक्शन को क्लास का नाम और स्कोप रिजोलुयशन ऑपरेटर के साथ घोषित किया गया है। फंक्शन fun() क्लास A का मेंम्बर फंक्शन है और क्लास B का फ्रेंड फंक्शन है। अगर एक क्लास के सभी मेंम्बर फंक्शन दूसरी क्लास में फ्रेंड घोषित कर दिये जाते हैं तब उस क्लास को फ्रेंड क्लास कहा जाता है।

उदाहरण के लिए :–

```
class C
{
- - - - - - -
friend class A;                 //All member functions of class A are friend
to C
- - - - - - -
};
```

प्रोग्राम 9.6 फ्रेंड फंक्शन

```
#include<iostream>
using namespace std;
class second; //forward declaration
class first
{
        int x;
public:
```

```
        void set_value(int a)
        {
                x=a;
        }
        friend void max(first, second);
};
class second
{
        int y;
public:
        void set_value(int b)
        {
                y=b;
        }
        friend void max(first, second);
};
void max(first f, second s)
{
        if(f.x>s.y)
        cout<<“Maximum is”<<f.x;
        else
        cout<<“Maximum is”<<s.y;
}
int main()
{
        first A;
        second B;
        A.set_value(10);
        B.set_value(20);
        max(A,B);     // calling friend function
        return 0;
}
```

प्रोग्राम 9.6 का आउटपुट होगा–

Maximum is 20

9.9 रिटर्निगं ऑब्जेक्ट

पिछले अनुभाग में हमने देखा कि फ्रेंड फंक्शन ऑरग्यूमेन्ट के रूप में ऑब्जेक्ट लेते हैं। फ्रेंड फंक्शनऑब्जेक्ट भी रिटर्न कर सकते हैं।

प्रोग्राम 9.7 रिटर्निगं ऑब्जेक्ट

```
#include<iostream>
using namespace std;
class vector
{
        int V[3];
public:
        void set_vector(void)
        {
                cout<<"Enter three numbers\n";
                for(int i=0; i<3; i++)
                cin>>V[i];
        }
        void display(void)
        {
                for(int i=0; i<3; i++)
                cout<<V[i]<< ",";
        }
        friend vector sum(vector, vector);
};
vector sum(vector p, vector q)
{
        vector r;
        for(int j=0;j<3; j++)
        r.V[j]=p.V[j]+q.V[j];
        return r;
}
int main()
```

```
{
        vector v1, v2,v3;
        v1.set_vector();
        v2.set_vector();
        v3=sum(v1,v2);
        cout<<"First vector is:";
        v1.display();
        cout<<"\n";
        cout<<"Second vector is:";
        v2.display();
        cout<<"\n";
        cout<<"Resultant vector is:";
        v3.display();
        return 0;
}
```

प्रोग्राम 9.7 का आउटपुट होगा–

```
Enter three numbers
3       -2      5
Enter three numbers
-8      6       7
First vector is: 3,-2,5,
Second vector is: -8,6,7,
Resultant vector is: -5,4,12
```

9.10 मेंम्बर के लिए पोइन्टर

हम एक क्लास के मेंम्बर का एड्रेस पोइंटर को असाइन कर सकते है।

उदाहरण के लिए :–

```
class X
{
        int a;
public:
        void show();
};
```

हम क्लास X के मेंम्बर a के लिए पोइंटर परिभाषित इस प्रकार कर सकते हैं–

```
int X:: *p=&X::a;
```

X:: * का मतलब क्लास X के मेंम्बर के लिए पोंइटर।

&X:: * का मतलब क्लास X का मेंम्बर a का एड्रेस।

स्टेटमेंट int *p=&a; कार्य नहीं करेगा। पोइंटर p का उपयोग डेटा मेंम्बर a को मेंम्बर फंक्शन और फ्रेंड फंक्शन के अन्दर एक्सेस करने के लिए कर सकते है।

उदाहरण के लिए :–

```
void show()
{
        X x;    // object created
        cout<<x.*p;  //display value of a
        cout<<x.a;    //same as above
}
```

हम क्लास के मेंम्बर फंक्शन के लिए पोंइटर सेट कर सकते है। मेंम्बर फंक्शन को डिरेफरेंसिंग ऑपरेटर (.*) की सहायता से कॉल कर सकते हैं।

उदाहरण के लिए :–

```
X x;              // object created
void (X::*pf)()=&X::show;
(x.*pf)();        //invoke show()
```

यहाँ pf मेंम्बर फंक्शन show() के लिए पोइंटर हैं।

महत्वपूर्ण बिंदु

- क्लास एक यूजर डिफान्ड टाईप है जो डेटा और फंक्शन को एक साथ बाँधे रखता है।
- स्वतः ही क्लास के मेम्बर प्राईवेट होते है।
- क्लास के मेंम्बर फंक्शन को क्लास के अन्दर और क्लास से बाहर परिभाषित किया जा सकता है।
- public और private कीवर्ड को एक्सेस मोडिफायर्स कहा जाता है।
- क्लास के डेटा मेंम्बर को स्टेटिक के रूप में भी घोषित किया जा सकता है।
- स्टेटिक कीवर्ड के साथ घोषित मेंम्बर फंक्शन को स्टेटिक मेंम्बर फंक्शन कहा जाता है।

- एक फ्रेंड फंक्शन क्लास के प्राईवेट डाटा को उस क्लास के ऑब्जेक्ट के जरिए एक्सेस कर सकते हैं।
- हम एक क्लास के मेंम्बर का एड्रेस पोइंटर को असाइन कर सकते है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न :

प्रश्न 1. एक यूजर डिफान्ड टाईप जो डेटा और फंक्शन को एक साथ बाँधे रखता है उसेकहा जाता है

(अ) ऑब्जेट (ब) क्लास (स) ऐरे (द) पोइंटर

प्रश्न 2. स्वतः ही क्लास के मेम्बर होते है

(अ) पब्लिक (ब) प्राईवेट (स) प्रोटेक्टेड (द) इनमें से कोई नहीं

प्रश्न 3. इनमें से कोनसा एकएक्सेस मोडिफायर है?

(अ) public (ब) private (स) अ. और ब. दोनों (द) इनमें से कोई नहीं

प्रश्न 4. इनमें से कोनसा स्टेटिक डेटा मेंम्बर के संदर्भ में सत्य है ?

(अ) इसकी प्रांरभिक वेल्यू शून्य होती है जब इसके क्लास का पहला ऑब्जेक्ट बनाया जाता है।

(ब) केवल एक ही प्रतिलिपी इस डेटा मेम्बर की बनती है।

(स) इसे क्लास वेरिएबल भी कहा जाता है।

(द) उपरोक्त सभी

प्रश्न 5 इनमें से कोनसा स्टेटिक मेंम्बर फंक्शन के संदर्भ में सत्य है?

(अ) स्टेटिक कीवर्ड के साथ घोषित किया जाता है।

(ब) केवल क्लास के दूसरे स्टेटिक डेटा मेंम्बर और मेंम्बर फंक्शन को एक्सेस कर सकते है।

(स) इनको क्लास के नाम की सहायता से कॉल किया जाता है।

(द) उपरोक्त सभी

प्रश्न 6. इनमें से कोनसा फ्रेंड फंक्शन के संदर्भ में सत्य है?

(अ) इसे सामान्य फंक्शन की तरह कॉल किया जाता है।

(ब) इसे क्लास के अन्दर कही भी घोषित किया जा सकता है।

(स) सामान्यतः इसके आरग्यूमेन्ट ऑब्जेक्ट होते है।

(द) उपरोक्त सभी

अति लघूउत्तरात्मक प्रश्न

प्रश्न 1. क्लास किसे कहते है ?

प्रश्न 2. ऑब्जेक्ट किसे कहते है ?

प्रश्न 3. फ्रेंड क्लास किसे कहते है ?

लघूउत्तरात्मक प्रश्न

प्रश्न 1. प्राईवेट और पब्लिक एक्सेस मोडिफायर में क्या अन्तर है ?

प्रश्न 2. स्टेटिक डेटा मेम्बर के गुण क्या है ?

प्रश्न 3. स्टेटिक मेंम्बर फंक्शन के क्या गुण होते है ?

निबंधात्मक प्रश्न

प्रश्न 1. फ्रेंड फंक्शन किसे कहते है ? इसके गुण लिखों ?

प्रश्न 2. एक 'Complex' क्लास बनाइए जो एक Complex नम्बर को बताता है और दो कॉम्पलेक्स नम्बरों को जोड़ने और घटाने के लिए मेंम्बर फंक्शन परिभाषित करने का प्रोग्राम लिखो ?

प्रश्न 3. दो क्लासों के डेटा मेंम्बर की अदला–बदली करने का फ्रेंड फंक्शन की सहायता से प्रोग्राम लिखों ?

उत्तरमाला

1:ब 2:ब 3:स 4: द 5: द 6:द

अध्याय –10

कंस्ट्रक्टर और डिस्ट्रक्टर

10.1 परिचय

पिछले अध्याय में क्लासों के उदाहरणों में हम मेंम्बर फंक्शन जैसे input() का उपयोग किया है। क्लास के प्राईवेट डेटा मेंम्बर को इनिशियलाइज(प्रांरभिक वेल्यू देना) करने के लिए फंक्शन कॉल स्टेटमेंट का पहले से घोषित ऑब्जेक्ट के साथ प्रयोग किया जाता है। ये फंक्शन डेटा मेंम्बर को उनको ऑब्जेक्ट बनने के समय इनिशियलाइज करने में असमर्थ है। C++ का उद्धेश्य यह है कि क्लास भी बेसिक डेटा टाईप की तरह हो। जब एक क्लास टाईप वेरिएबल (ऑब्जेक्ट) को घोषित किये जाते है तब ठीक उसी तरह इनिशियलाइज हो जिस तरह से बेसिक डेटा टाईप के वेरिएबल को इनिशियलाइज किया जाता है।

इस अध्याय में, हम एक विशेष मेंम्बर फंक्शन जिसे कंस्ट्रक्टर कहा जाता है उसकी चर्चा करेगें। जो कि ऑब्जेक्ट को इनिशियलाइज करने के लिए काम आते है। जब इन्हें घोषित किया जाता है। एक दूसरा विशेष मेंम्बर फंक्शन डिस्ट्रक्टर जो उन ऑब्जेक्ट को खत्म करता है जो आगे के लिए काम नहीं आते है।

10.2 कंस्ट्रक्टर

कंस्ट्रक्टर एक विशेष प्रकार का मेंम्बर फंक्शन है उसके क्लास के ऑब्जेक्ट को इनिशियलाइज करने के लिए काम आते हैं। कंस्ट्रक्टर स्वतः ही कॉल हो जाते है जब उसके क्लास के ऑब्जेक्ट घोषित किया जाते है। कंस्ट्रक्टर फंक्शन की निम्नलिखित विशेषताएँ होती है

- इसका नाम क्लास के नाम जैसा ही होता है।
- ये क्लास के पब्लिक अनुभाग में घोषित होने चाहिए ।
- जब ऑब्जेक्ट घोषित किया जाते है तब ये स्वतः ही कॉल हो जाते है।
- इनका कोई रिटर्न टाईप नहीं होता है, यहाँ तक की void भी और इस प्रकार ये कोई वेल्यू रिटर्न नहीं करता है।
- इनको इनहेरिट नहीं किया जा सकता है।
- इनके एड्रेस को एक्सेस नहीं कर सकते है।

उदाहरण के लिए :–

```
class point
{
```

```
        int x,y;
public:
point(void);  //constructor declared
---------
---------
};
point :: point(void)  //constructor defined
{
        x=0;
        y=0;
}
```

जब हम क्लास point के ऑब्जेक्ट को घोषित करते हैं

उदाहरण के लिए :–

```
point p;
```

क्लास के अन्दर कंस्ट्रक्टर स्वतः ही कॉल हो जाता है और प्राईवेट डेटा मेंम्बर x और y को शून्य से ऑब्जेक्ट p के लिए इनिशियलाइज कर देते है। जो कंस्ट्रक्टर आरग्यूमेन्ट नहीं लेते है उसे डिफॉल्ट कंस्ट्रक्टर कहा जाता है। अगर ऐसा कोई कंस्ट्रक्टर क्लास में परिभाषित नहीं है तब कम्पाइलर डिफॉल्ट कंस्ट्रक्टर उपलब्ध करवाता है उस क्लास के ऑब्जेक्ट घोषित करने के लिए।

10.3 पैरामीटरराइज्ड कंस्ट्रक्टर

जो कंस्ट्रक्टर आरग्यूमेन्ट लेते है उन्है पैरामीटराइज्ड कंस्ट्रक्टर कहा जाता है।

उदाहरण के लिए :–

```
class point
{
        int x, y;
public:
        point(int a, int b);     // parameterized constructor
        {
        --------
```

```
        --------
        }
};
point::point(int a, int b)
{
        x=a;
        y=b;
}
```

पैरामीटराइज्ड कंस्ट्रक्टर को दो तरीको से कॉल किया जा सकता है।

- point p= point(10,20); //explicit call

यह स्टेटमेंट एक ऑब्जेक्ट pको 10 और 20 वेल्यू से इनिशियलाइज करता हैं ।

- point p(10,20); //implicit call

यह स्टेटमेंट उपरोक्त स्टेटमेंट की तरह काम करता है।

कंस्ट्रक्टर फंक्शन को इनलाईन फंक्शन के रूप में भी परिभाषित किया जा सकता है।

उदाहरण के लिए :–

```
class point
{
        int x, y;
public:
       point(int a, int b)
        {
                x=a;
                y=b;
        }
--------
--------
};
```

कंस्ट्रक्टर के ऑरग्यूमेन्ट किसी भी टाईप के हो सकते है सिवाय उस क्लास के जिससे ये सम्बंधित हैं।

उदाहरण के लिए :–

```
class X
{
--------
--------
public:
X(X);
};
```

अमान्य है।

लेकिन एक कंस्ट्रक्टर अपनी क्लास का रेफरेंस एक ऑरग्यूमेन्ट के रूप में ले सकता है।

उदाहरण के लिए :–

```
class X
{
--------
--------
public:
X(X&);
};
```

यह वैद्य है और कंस्ट्रक्टर को कॉपी कंस्ट्रक्टर कहा जाता है।

प्रोग्राम 10.1: पैरामीटराइज्ड कंस्ट्रक्टर

```
#include<iostream>
using namespace std;
class rectangle
{
        int length;
        int breadth;
public:
        rectangle(int a, int b)
```

```
        {
                length=a;
                breadth=b;
        }
        void area()
        {
                cout<<"Area="<<length*breadth;
        }
};
int main()
{
        rectangle r(5,10);
        return 0;
}
```

प्रोग्राम 10.1 का आउटपुट होगा–

```
Area=50
```

10.4 कंस्ट्रक्टर ओवरलोडिंग :–

एक क्लास में एक से ज्यादा कंस्ट्रक्टर हो सकते है और इसे कंस्ट्रक्टर ओवरलोडिंग कहा जाता हैं ।

प्रोग्राम 10.2: कंस्ट्रक्टर ओवरलोडिंग

```
#include<iostream>
using namespace std;
class point
{
         int x,y;
public:
        point()// no argument constructor
        {
```

```
        x=0;
        y=0;
    }
    point(int a)    //one argument constructor
    { x=y=a;}
    point(int m, int n)     //two arguments constructor
    {
        x=m;
        y=n;
    }
    void show()
    {
        cout<<"x="<<x<<"\n";
        cout<<"y="<<y<<"\n";
    }
};
int main()
{
    point p1;
    point p2(5);
    point p3(7,11);
    cout<<"Coordinates of p1 are\n";
    p1.show();
    cout<<"Coordinates of p2 are\n";
    p2.show();
    cout<<"Coordinates of p3 are\n";
    p3.show();
    return 0;
}
```

प्रोग्राम 10.2 का आउटपुट होगा–

Coordinates of p1 are

x=0

y=0

Coordinates of p2 are

x=5

y=5

Coordinates of p3 are

x=7

y=11

उपरोक्त प्रोग्राम में, क्लास point में तीन कंस्ट्रक्टर है। पहला बिना आरग्यूमेन्ट का कंस्ट्रक्टर है। इसका उपयोग बिना किसी प्रारंभिक वेल्यू का ऑब्जेक्ट बनाने के लिए किया जाता है। जब हम अन्य कंस्ट्रक्टर परिभाषित करते हैं तब हमें कम्पाइलर को संतुष्ट करने के लिए बिना आऱ्ग्यूमेन्ट का कंस्ट्रक्टर परिभाषित करना होगा। दूसरा कंस्ट्रक्टर एक वेल्यू पैरामीटर के रूप में लेता है और ऑब्जेक्ट को इनिशियलाइज करता है। तीसरा कंस्ट्रक्टर दो आऱ्ग्यूमेन्ट लेता है और इन दो वेल्यू से ऑब्जेक्ट को इनिशियलाइज करता है।

10.5 डिफॉल्ट आऱ्ग्यूमेन्ट के साथ कंस्ट्रक्टर

कंस्ट्रक्टर डिफॉल्ट आऱ्ग्यूमेन्ट ले सकते है।

उदाहरण के लिए :–

```
point(int a,  int b=0);
```

ध्यान दे कि डिफॉल्ट आऱ्ग्यूमेन्ट दायें से बायें दिये जाते है। आऱ्ग्यूमेन्ट bकी डिफॉल्ट वेल्यू शून्य है। तब यह स्टेटमेंट

```
point p(5);
```

aको 5 वेल्यू असाइन करता है और bको शून्य (डिफॉल्ट द्वारा) लेकिन स्टेटमेंट

```
point(7,11);
```

a को 7 वेल्यू असाइन करता है और b को 11 क्योंकि जब वास्तविक पैरामीटर दिये गये हो तब वे डिफॉल्ट आरग्यूमेन्ट को ओवरराइड कर देते है।

अगर एक आरग्यूमेन्ट वाला कंस्ट्रक्टर भी इस कंस्ट्रक्टर के साथ मौजूद है तब कॉल करने वाला स्टेटमेंट

```
point p(5);
```

यह फैसला लेने में असमर्थ है कि कौनसे कंस्ट्रक्टर को कॉल किया जाये और एक अस्पष्टता की स्थिति उत्पन्न हो जाती है। कम्पाइलर एक त्रुटि संदेश उत्पन्न कर देता है।

10.6 ऑब्जेक्ट का डाइनामिक इनिशियलाइजेशन

एक ऑब्जेक्ट की इनिशियल वेल्यू रन टाइम पर दी जा सकती है। डाइनामिक इनिशियलाइजेशन का फायदा यह है कि हम विभिन्न प्रकार के इनपुट कंस्ट्रक्टर ओवरलोडिंग के द्वारा दे सकते है।

प्रोग्राम 10.3 ऑब्जेक्ट का डाइनामिक इनिशियलाइजेशन

```
#include<iostream>
using namespace std;
class shape
{
        float length, breadth;
        float radius;
        float area;
public:
        shape() { }
        shape(float r)
        {
                radius=r;
                area=3.14*r*r;
        }
        shape(float l, float b)
        {
                length=l;
                breadth=b;
                area=length*breadth;
        }
        void display()
        {
                cout<<“Area=”<<area<<“\n”;
        }
};
int main()
{
```

```
        shape circle, rectangle;
        float r, l, b;
        cout<<"Enter the radius of circle\n";
        cin>>r;
        circle=shape(r);
        cout<<"Enter the length and breadth of rectangle\n";
        cin>>l>>b;
        rectangle=shape(l,b);
        cout<<"Area of circle\n";
        circle.display();
        cout<<"Area of rectangle\n";
        rectangle.display();
        return 0;
}
```

प्रोग्राम 10.3 का आउटपुट होगा–

```
Enter the radius of circle
5
Enter the length and breadth of rectangle
17      8
Area of circle
78.5
Area of rectangle
136
```

10.7 कॉपी कंस्ट्रक्टर

कंस्ट्रक्टर जो एक ऑब्जेक्ट को उसी क्लास के दूसरे ऑब्जेक्ट से घोषित और इनिशियलाइज करता है, उसे कॉपी कंस्ट्रक्टर कहा जाता है। एक कॉपी कंस्ट्रक्टर उसी क्लास के ऑब्जेक्ट का रेफरेंस आरग्यूमेन्ट के रूप में लेते है।

प्रोग्राम 10.4 कॉपी कंस्ट्रक्टर

```
#include<iostream>
using namespace std;
class product
{
        int code;
```

```
public:
        product(){  } // default constructor
        product(int x)//parameterized constructor
        {
                code=x;
        }
        product(product &y)          //copy constructor
        {
                code=y.code;         //copy the value
        }
        void display(void)
        {
                cout<<code;
        }
};
int main()
{
        product p1(10);
        product p2(p1);       //copy constructor called
        product p3=p1;        //again copy constructor called
        cout<<"Code of p1:";
        p1.display();
        cout<<"\nCode of p2:";
         p2.display();
        cout<<"\nCode of p3:";
        p3.display();
        return 0;
}
```

प्रोग्राम 10.4 का आउटपुट होगा–

Code of p1:10

Code of p2:10

Code of p3:10

नोटः– जब प्रोग्राम में कॉपी कंस्ट्रक्टर परिभाषित नहीं होता है तब कम्पाइलर अपना कॉपी कंस्ट्रक्टर उपलब्ध कर देता है।

10.8 डिस्ट्रक्टर :–

क्लास का एक विशेष प्रकार का मेंम्बर फंक्शन जिसका प्रयोग कंस्ट्रक्टर के द्वारा बनाये गये ऑब्जेक्ट को खत्म करने के लिए किया जाता है। डिस्ट्रक्टर फंक्शन की निम्नलिखित विशेषताएँ होती है

(1) इसका नाम क्लास के नाम के जैसा होता है। लेकिन इसके नाम से पहले tilde(~) चिन्ह होता है।

(2) यह आरग्यूमेन्ट नहीं लेता है और वेल्यू भी रिटर्न नहीं करता है।

(3) जब किसी प्रोग्राम या ब्लोक या फंक्शन से बाहर आते है तब यह कम्पाइलर के द्वारा स्वतः ही कॉल होता है

निम्नलिखित प्रोग्राम दर्शाता है कि डिस्ट्रक्टर फंक्शन कम्पाइलर के द्वारा स्वतः ही कॉल होता है।

प्रोग्राम 10.5 डिस्ट्रक्टर फंक्शन

```
#include<iostram>
using namespace std;
class sample
{
sample()                    // Constructor
{
cout<<“Object created\n”;
}
~sample()                   // Destructor
{
cout<<“Object destroyed”;
}
};

int main()
{
```

```
sample s;
 return 0;
}
```

प्रोग्राम 10.5 का आउटपुट होगा–

```
Object created
Object destroyed
```

डिस्ट्रक्टर का उपयोग ऑब्जेक्ट को आंवटित मेमौरी को रन टाइम पर मुक्त करने के लिए किया जाता है। ताकि मुक्त मेमौरी का पुनः उपयोग दूसरे प्रोग्राम या ऑब्जेक्ट के लिए किया जा सके। किसी ऑब्जेक्ट को मेमौरी का आवंटन कंस्ट्रक्टर फंक्शन में new ऑपरेटर से किया जाता है और डिस्ट्रक्टर फंक्शन में delete ऑपरेटर से आवंटित मेमौरी को पुनः लिया जाता है।

प्रोग्राम 10.6 डिस्ट्रक्टर का उपयोग मेमौरी को मुक्त करने के लिए

```
#include<iostream>
using namespace std;
class sample
{
        char *t;
public:
        sample(int length)
        {
        t=new char[length];
        cout<<"Character array of length"<<length<<"created";
        }
        ~sample()
        {
        delete t;
        cout<<"\nMemory de-allocated for the character array";
        }
};
int main()
{
```

```
sample s(10);
return 0;
}
```

प्रोग्राम 10.5 का आउटपुट होगा–

```
Character array of length 10 created
Memory de-allocated for the character array
```

महत्वपूर्ण बिंदु

- कंस्ट्रक्टर एक विशेष प्रकार का मेंम्बर फंक्शन है यह उसके क्लास के ऑब्जेक्ट को इनिशियलाइज करने के लिए काम आते हैं।
- कंस्ट्रक्टर के ऑरग्यूमेन्ट किसी भी टाईप के हो सकते है सिवाय उस क्लास के जिससे ये सम्बंधित हैं।
- एक कंस्ट्रक्टर अपनी क्लास का रेफरेंस एक ऑरग्यूमेन्ट के रूप में ले सकता है।
- कंस्ट्रक्टर जो एक ऑब्जेक्ट को उसी क्लास के दूसरे ऑब्जेक्ट से घोषित और इनिशियलाइज करता है, उसे कॉपी कंस्ट्रक्टर कहा जाता है।
- जब प्रोग्राम में कॉपी कंस्ट्रक्टर परिभाषित नहीं होता है तब कम्पाइलर अपना कॉपी कंस्ट्रक्टर उपलब्ध कर देता है।
- क्लास का एक विशेष प्रकार का मेंम्बर फंक्शन जिसका प्रयोग कंस्ट्रक्टर के द्वारा बनाये गये ऑब्जेक्ट को खत्म करने के लिए किया जाता है उसे डिस्ट्रक्टर कहा जाता है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न :

प्रश्न 1.इनमें से कोनसाकंस्ट्रक्टरस के संदर्भ में सत्य है?

(अ) इसका नाम क्लास के नाम जैसा ही होता है।

(ब) ये क्लास के पब्लिक अनुभाग में घोषित होने चाहिए ।

(स) जब ऑब्जेक्ट घोषित किये जाते है तब ये स्वतः ही कॉल हो जाते है।

(द) उपरोक्त सभी

प्रश्न 2. कंस्ट्रक्टर जो आरग्यूमेन्ट लेते है उन्हेंक्या कहा जाता है?

(अ) डिफॉल्ट कंस्ट्रक्टर (ब) बिना आऱ्ग्यूमेन्ट का

(स) पैरामीटराइज्ड कंस्ट्रक्टर (द) इनमें से कोई नहीं

प्रश्न 3. कंस्ट्रक्टर जो एक ऑब्जेक्ट को उसी क्लास के दूसरे ऑब्जेक्ट से घोषित और इनिशियलाइज करता है, उसे कहा जाता है।

(अ) डिफॉल्ट कंस्ट्रक्टर (ब) कॉपी कंस्ट्रक्टर

(स) पैरामीटराइज्ड कंस्ट्रक्टर (द) इनमें से कोई नहीं

प्रश्न 4. इनमें से कोनसाडिस्ट्रक्टरस के संदर्भ में सत्य है?

(अ) इसका नाम क्लास के नाम के जैसा होता है। लेकिन इसके नाम से पहले tilde(~) चिन्ह होता है।

(ब) यह आरग्यूमेन्ट नहीं लेता है और वेल्यू भी रिटर्न नहीं करता है।

(स) यह कम्पाइलर के द्वारा स्वतः ही कॉल होता है जब किसी प्रोग्राम या ब्लोक या फंक्शन से बाहर आते है।

(द) उपरोक्त सभी

अति लघूउत्तरात्मक प्रश्न

प्रश्न 1. कंस्ट्रक्टरस क्या होते हैं ?

प्रश्न 2. पैरामीटराइज्ड कंस्ट्रक्टरस क्या होते हैं ?

प्रश्न 3. कंस्ट्रक्टर ओवरलोड़िग किसे कहते है ?

लघूउत्तरात्मक प्रश्न

प्रश्न 1. कंस्ट्रक्टरस की क्या विशेषताएँ होती हैं ?

प्रश्न 2. डिस्ट्रक्टरस क्या होते है इसकी विशेषताएँ लिखों ?

प्रश्न 3. डिस्ट्रक्टरस के उपयोग लिखों ?

निबंधात्मक प्रश्न

प्रश्न 1. डिफॉल्ट आरग्यूमेन्ट के साथ कंस्ट्रक्टर का वर्णन कीजिए ?

उत्तरमाला

1:द 2:स 3:ब 4: द

अध्याय –11

ऑपरेटर ओवरलोडिंग

11.1 परिचय

ऑपरेटर ओवरलोडिंग C++ भाषा का महत्पवपूर्ण फिचर है। इस फिचर से हम यूजर–डिफान्ड टाईप के दो वेरिएबल को जोड़ सकते है उसी तरह से जिस तरह से हम बेसिक डेटा टाईप से करते है। किसी ऑपरेटर को एक डाटा टाईप के लिए विशेष मिनिंग देना ऑपरेटर ओवरलोडिंग कहा जाता है। C++ में सभी ऑपरेटर को ओवरलोड कर सकते है। सिवाय निम्नलिखित ऑपरेटर के –

- क्लास मेंम्बर एक्सेस ऑपरेटर (., .*)
- स्कोप रिजोल्यूशन ऑपरेटर (::)
- साइज ऑपरेटर (sizeof)
- कंडिशनल ऑपरेटर (?:)

जब हम एक ऑपरेटर को ओवरलोड करते है इसका मूलरूप बरकरार रहता है। उदाहरण के लिए अगर हम + ऑपरेटर को दो मैट्रीक्स को जोडने के लिए करते है तब भी इस ऑपरेटर से दो संख्याओं को भी जोड़ सकते है।

11.2 ऑपरेटर फंक्शन

एक ऑपरेटर को अतिरिक्त मिनिंग देने के लिए हम एक विशेष फंक्शन काम में लेते है जिसे ऑपरेटर फंक्शन कहा जाता है। ऑपरेटर फंक्शन का प्रोटोटाईप इस प्रकार होता हैं–

```
return_type class_name :: operator op(arguments list)
{
        function body
}
```

जहाँ operator एक कीवर्ड है और op एक ऑपरेटर है जिसे ओवरलोड किया जाना है। ऑपरेटर फंक्शन एक क्लास का मेंम्बर फंक्शन या फ्रेंड फंक्शन होना चाहिए। उनमें फर्क यह है कि मेंम्बर फंक्शन यूनरी ऑपरेटर के लिए कोई आरग्यूमेंन्ट नहीं लेता है और बाइनरी ऑपरेटर के लिए एक आरग्यूमेन्ट लेता है जबकि फ्रेंड फंक्शन यूनरी ऑपरेटर के लिए एक आरग्यूमेन्ट लेता है और बाइनरी ऑपरेटर के लिए दो ऑपरेटर लेता है।

11.3 यूनरी ऑपरेटर को मेंम्बर फंक्शन से ओवरलोड करना

हम पोस्ट फिक्स इंक्रिमेंट ऑपरेटर (++) को उदाहरण के लिए लेते है। यह केवल एक ऑपरेड लेता हैं और इसकी वेल्यू को एक से बढ़ा देता है जब बेसिक डेटा टाईप के साथ प्रयोग किया जाता है। हम इस ऑपरेटर को ओवरलोड करेंगें ताकि इसका प्रयोग एक ऑब्जेक्ट के साथ किया जा सके और इस ऑब्जेक्ट के हर एक डेटा आइटम की वेल्यू को एक बढ़ा देगा।

प्रोग्राम11.1 पोस्ट फिक्स इंक्रिमेंट ऑपरेटर को ओवरलोड करना

```
#include<iostream>
using namespace std;
class point
{
        int x,y;
public:
         void getdata(int a, int b)
         {
                  x=a;
                  y=b;
         }
         void show(void)
         {
                cout<<"x="<<x;
                cout<<"y="<<y<<"\n";
         }
         void operator++(int)
        {
                x++;
                y++;
        }
};

int main()
{
 point p;
 p.getdata(5,8);
```

```
 cout<<"p:";
 p.show();
 p++;          // invoke operator function
 cout<<"p++:";
 p.show();
return 0;
}
```

प्रोग्राम 11.1 का आउटपुट होगा–

p: x=5 y=8

p++: x=6 y=9

ऑपरेटर फंक्शन में int का प्रयोग यह दर्शाता है कि हम पोस्टफिक्स इंक्रिमेंट ऑपरेटर को ओवरलोड कर रहे है न कि प्रिफिक्स इंक्रिमेंट ऑपरेटर को।

11.4 यूनरी ऑपरेटर को फ्रेंड फंक्शन से ओवरलोड करना

हम प्रिफिक्स डिक्रिमेंट ऑपरेटर (--) को उदाहरण के लिए लेते हैं। जो एक ऑपरेड लेता है और उसकी वेल्यू को एक कम कर देता है जब बेसिक डेटा टाईप के साथ प्रयोग किया जाता हैं। हम इस ऑपरेटर को ओवरलोड करेंगें ताकि इसका प्रयोग ऑब्जेक्ट के साथ किया जा सके और इस ऑब्जेक्ट के हर एक डेटा आइटम की वेल्यू को एक कम कर देगा।

प्रोग्राम 11.2 :– प्रिफिक्स डिक्रिमेंट ऑपरेटर को ओवरलोड करना

```
#include<iostream>
using namespace std;
class point
{
        int x,y;
public:
         void getdata(int a, int b)
         {
                  x=a;
                  y=b;
         }
         void show(void)
         {
                  cout<<"x="<<x;
```

```
            cout<<"y="<<y<<"\n";
        }
        friend void operator--(point &s)
        {
            s.x=s.x-1;
            s.y=s.y-1;
        }
};
int main()
{
 point p;
 p.getdata(7,10);
cout<<"p:";
 p.show();
 --p;
cout<<"--p:";
 p.show();
return 0;
}
```

प्रोग्राम 11.2 का आउटपुट होगा–

p: x=7 y=10

p++: x=6 y=9

ध्यान दे कि ऑपरेटर फंक्शन में आरग्यूमेन्ट रेफरेंस के द्वारा भेजा गया है। अगर हम वेल्यू के द्वारा भेजते यह काम नहीं करता क्योंकि जो बदलाव ऑपरेटर फंक्शन में किये गये main() फंक्शन में प्रतिबिम्बित नहीं होगें। निम्नलिखित ऑपरेटर्स को फ्रेंड फंक्शन से ओवरलोड नहीं किया जा सकता है–

- असाइनमेंट ऑपरेटर =
- फंक्शन कॉल ऑपरेटर ()
- सब्सक्रिप्टींग ऑपरेटर []
- क्लास मेंम्बर एक्सेस ऑपरेटर ->

11.5 बाइनरी ऑपरेटर को मेंम्बर फंक्शन से ओवरलोड करना

हम बाइनरी प्लस ऑपरेटर (+) को उदाहरण के लिए लेते है। जो दो ऑपरेड लेता है और दोनों का योग कर देता है। जब इसका प्रयोग बेसिक डेटा टाईप के साथ किया जाता है। हम इस ऑपरेटर को दो मैट्रिक्स को जोड़ने के लिए उपयोग करेंगे।

प्रोग्राम 11.3 :– बाइनरी प्लस ऑपरेटर (+) को ओवरलोड करना

```
#include<iostream>
using namespace std;
class matrix
{
        int mat[2][2];
public:
          void getmatrix(void);
          matrix operator+(matrix);
          void showmatrix(void);
};
void matrix::getmatrix(void)
 {
  for(int i=0; i<2; i++)
    for(int j=0; j<2; j++)
      {
        cout<<"Enter the number:";
        cin>>mat[i][j];
        }
 }
matrix matrix::operator+(matrix m)
{
 matrix temp;
 for(int i=0; i<2; i++)
    for(int j=0; j<2; j++)
      temp.mat[i][j]=mat[i][j]+m.mat[i][j];
 return temp;
}
```

```
 void matrix::showmatrix(void)
 {
   for(int i=0; i<2; i++)
     {
       for(int j=0; j<2; j++)
       cout<<mat[i][j]<<“\t”;
       cout<<“\n”;
     }
}

int main()
{
  matrix m1,m2,m3;
  m1.getmatrix();
  m2.getmatrix();
  m3=m1+m2;
  cout<<"matrix m1:\n";
  m1.showmatrix();
  cout<<"matrix m2:\n";
  m2.showmatrix();
  cout<<"Resultant matrix:\n";
   m3.showmatrix();
return 0;
}
```

प्रोग्राम 11.3 का आउटपुट होगा–

```
Enetr the number: 2
Enetr the number: 3
Enetr the number: 1
Enetr the number: 4
Enetr the number: 6
Enetr the number: 7
Enetr the number: 8
```

Enetr the number: 9
matrix m1:
2 3
1 4
matrix m2:
6 7
8 9
Resultant matrix:
8 10
9 13

उपरोक्त प्रोग्राम में, ऑपरेटर फंक्शन मैट्रिक्स टाईप का एक आरग्यूमेन्ट लेता है। जो कि बाइनरी प्लस ऑपरेटर का दूसरा ऑपरेड है। पहला ऑपरेड m1 का प्रयोग ऑपरेटर फंक्शन को कॉल करने के लिए किया गया है ताकि m1 के डेटा मेंम्बर को ऑपरेटर फंक्शन द्वारा सीधे ही एक्सेस किया गया है।

m3=m1+m2;

निम्न स्टेटमेंट के समान है

m3 =m1.operator+(m2);

बाइनरी ऑपरेटर के लिए, बायी तरफ का ऑपरेड ऑपरेटर फंक्शन को कॉल करने के लिए किया जाता है और दायीं तरफ के ऑपरेटर फंक्शन का आरग्यूमेन्ट के रूप में भेजा जाता है।

11.6 बाइनरी ऑपरेटर को फ्रेंड फंक्शन से ओवरलोड करना

निम्नलिखित प्रोग्राम बाइनरी प्लस ऑपरेटर (+) को फ्रेंड फंक्शन से दो कॉम्पलेक्स संख्याओं को जोडनें का उदाहरण है।

प्रोग्राम 11.4 बाइनरी प्लस ऑपरेटर को फ्रेंड फंक्शन से ओवरलोड करना

```
#include<iostream>
using namespace std;
class complex
{
    float real;
    float imag;
pubic:
      void input(float x, float y)
```

```
{
        real=x;
        imag=y;
}
friend complex operator + (complex a, complex b)
{
        complex c;
        c.real=a.real+b.real;
        c.imag=a.imag+b.imag;
        return c;
}
void show(void)
{
        cout<<real<<"+i"<<imag<<"\n";
}
};
int main()
{
        complex c1,c2,c3;
        c1.input(1.6,6.2);
        c2.input(2.3,3.4);
        c3=c1+c2;               //invoke operator function
        cout<<"C1=";
        c1.show();
        cout<<"C2=";
        c2.show();
        cout<<"C3=";
        c3.show();
        return 0;
}
```

प्रोग्राम 11.3 का आउटपुट होगा–

C1=1.6+i6.2;

C2=2.3+i3.4;

C3=3.9+i9.6;

उपरोक्त प्रोग्राम में ऑपरेटर फंक्शन कॉम्पलेक्स टाईप के दो आरग्यूमेन्ट लेता है और एक कॉम्पलेक्स संख्या को परिणाम के रूप में रिटर्न करता है। निम्नलिखित स्टेटमेंट

c3=c1+c2;

निम्न स्टेटमेंट के समान है

c3=operator+(c1,c2);

महत्वपूर्ण बिंदु

- किसी ऑपरेटर को एक डाटा टाईप के लिए विशेष मिनिंग देना ऑपरेटर ओवरलोडिंग कहा जाता है।
- जब हम एक ऑपरेटर को ओवरलोड करते है इसका मूलरूप बरकरार रहता है।
- ऑपरेटर को अतिरिक्त मिनिंग देने के लिए हम एक विशेष फंक्शन काम में लेते है जिसे ऑपरेटर फंक्शन कहा जाता है।
- ऑपरेटर फंक्शन एक क्लास का मेंम्बर फंक्शन या फ्रेंड फंक्शन होना चाहिए।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न :

प्रश्न 1. किस ऑपरेटर को ओवरलोड नहीं किया जा सकता है?

(अ) स्कोप रिजोल्यूशन ऑपरेटर (::)(ब) क्लास मेंम्बर एक्सेस ऑपरेटर (., .*)
(स) बाइनरी प्लस ऑपरेटर (+) (द) कंडिशनल ऑपरेटर (?:)

प्रश्न 2. बाइनरी ऑपरेटर को ओवरलोड करने के लिए ऑपरेटर फंक्शन मेंम्बर फंक्शन के रूप में कितने आरग्यूमेन्ट लेता है?

(अ) दो आरग्यूमेन्ट (ब) एक आरग्यूमेन्ट
(स) शून्य आरग्यूमेन्ट (द) इनमें से कोई नहीं

प्रश्न 3. यूनरी ऑपरेटर को ओवरलोड करने के लिए ऑपरेटर फंक्शन फ्रेंड फंक्शन के रूप में कितने आरग्यूमेन्ट लेता है?

(अ) दो आरग्यूमेन्ट (ब) एक आरग्यूमेन्ट
(स) शून्य आरग्यूमेन्ट (द) इनमें से कोई नहीं

प्रश्न 4. किस ऑपरेटर को फ्रेंड फंक्शन से ओवरलोड नहीं कर सकते है?

(अ) = असाइनमेंट ऑपरेटर (ब) () फंक्शन कॉल ऑपरेटर
(स)[] सब्सक्रिप्टींग ऑपरेटर (द) उपरोक्त सभी

अति लघूउत्तरात्मक प्रश्न

प्रश्न 1. ऑपरेटर ओवरलोडिंग किसे कहा जाता हैं ?

प्रश्न 2 ऑपरेटर फंक्शन का प्रोटोटाईप लिखों ?

प्रश्न 3. ऑपरेटरस का नाम लिखों जिनको ओवरलोड़ नहीं किया जा सकता है ?

लघूउत्तरात्मक प्रश्न

प्रश्न 1. ऑपरेटर फंक्शन मेंम्बर फंक्शन के रूप में और ऑपरेटर फंक्शन फ्रेंड फंक्शन के रूप में दोनों में अन्तर का वर्णन कीजिए ?

निबंधात्मक प्रश्न

प्रश्न 1. क्लास के ऑब्जेक्ट को त्रृणात्मक बनाने के लिए यूनरी माइनस ऑपरेटर को ओवरलोड करने का प्रोग्राम फ्रेंड फंक्शन का उपयोग करके लिखों ?

प्रश्न 2 बाइनरी प्लस ऑपरेटर को दो स्ट्रींग को जोड़ने के लिए ओवरलोड करने का प्रोग्राम मेंम्बर फंक्शन का उपयोग करके लिखों ?

उत्तरमाला

1:स 2:ब 3:ब 4: द

अध्याय – 12

इनहेरिटेंस

12.1 परिचय

रियूजेबिलीटी C++ भाषा का महत्वपूर्ण फीचर है। इस फिचर के द्वारा पहले से तैयार क्लासों से नई क्लास बनाने के लिए किया जाता है इस प्रकिया को इनहेरिटेंस कहा जाता है। इस फिचर के द्वारा प्रोग्रामर समय और ऊर्जा को बचा सकता है। पहले से तैयार क्लास को बेस क्लास या पेरेंट क्लास या सुपर क्लास कहा जाता है और नई क्लास को डिराइड क्लास या चाइल्ड क्लास या सब क्लास कहा जाता है।

12.2 डिराइव्ड क्लास को परिभाषित करना

डिराइव्ड क्लास को परिभाषित करने का सिन्टेक्स इस प्रकार होता है

```
class derived-class-name : visibility-mode base-class-name
{
        members of derived class.
};
```

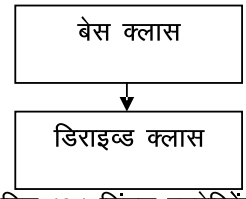
विजिबिलिटी मोड प्राईवेट प्रोटेक्टेड या पब्लिक हो सकता है। डिफॉल्ट विजिबिलिटी मोड प्राईवेट होता है। विजिबिलिटी मोड यह बताता है कि बेस क्लास के फीचर को प्राईवेटली प्रोटेक्टेडली या पब्लिकली इनहेरिट किया गया है। अगर बेस क्लास को डिराइव्ड क्लास के द्वारा प्राईवेटली इनहेरिट किया है तब बेस क्लास के पब्लिक और प्रोटेक्टेड मेंम्बर डिराइव्ड क्लास के प्राइवेट मेंम्बर बन जाते है। बेस क्लास के प्राईवेट मेंम्बर को कभी भी इनहेरिट नहीं किया जा सकता है।

अगर बेस क्लास को डिराइव्ड क्लास के द्वारा प्रोटेक्टेडली इनहेरिट किया गया है तब बेस क्लास के प्रोटेक्टेड और पब्लिक मेंम्बर डिराइव्ड क्लास के प्रोटेक्टेड मेंम्बर बन जाते है।

अगर बेस क्लास को डिराइव्ड क्लास के द्वारा पब्लिकली इनहेरिट किया गया है तब बेस क्लास के प्रोटेक्टेड मेंम्बर डिराइव्ड क्लास के प्रोटेक्टेड मेंम्बर बन जाते है और बेस क्लास के पब्लिक मेंम्बर डिराइव्ड क्लास के पब्लिक मेंम्बर बन जाते है।

12.3 सिंगल इनहेरिटेंस

सिंगल इनहेरिटेंस में एक बेस क्लास और एक डिराइव्ड क्लास होती है।



चित्र 12.1 सिंगल इनहेरिटेंस

निम्नलिखित प्रोग्राम सिंगल इनहेरिटेंस का उदाहरण है।

प्रोग्राम 12.1 सिंगल इनहेरिटेंस

```
#include<iostream>
using namespace std;
class data
{
  protected:
     int x,y;
 public:
          void getdata(int a, int b)
          {
                    x=a;
                    y=b;
          }
          void showdata(void)
          {
                   cout<<"x="<<x<<"\n";
                   cout<<"y="<<y<<"\n";
          }
};
class maximum: public data
{
  public:
           void max(void)
           {
                  if(x>y)
                  cout<<"Maximum is:"<<x;
```

```
            else
            cout<<"Maximum is:"<<y;
        }
};

int main()
{
  maximum m;
  m.getdata(4,9);
  m.showdata();
  m.max();
return 0;
}
```

प्रोग्राम 12.1 का आउटपुट होगा–

x=4

y=9

Maximum is: 9

उपरोक्त प्रोग्राम में, बेस क्लास में दो प्रोटेक्टेड मेंम्बर x और y है। ये दोनों डेटा मेंम्बर केवल बेस क्लास और इसकी तुरन्त डिराइव्ड क्लास के द्वारा एक्सेस होते है। इन दोनों क्लासों के बाहर से नहीं । डिराइव्ड क्लास maximum इन दोनों डेटा मेंम्बर में अधिक वेल्यू की गणना करती है। डिराइव्ड क्लास के द्वारा बेस क्लास के पब्लिक डेरिवेशन के बाद डिराइव्ड क्लास में निम्नलिखित मेंम्बर होगें।

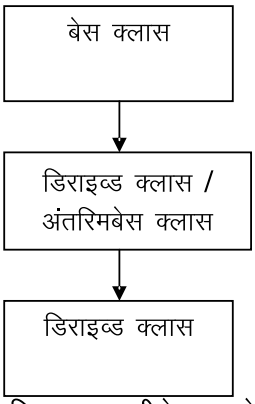
डिराइव्ड क्लास 'maximum'

प्रोटेक्टेड मेंम्बर
x, y
पब्लिक मेंम्बर
getdata()
showdata()
max()

चित्र 12.2 डिराइव्ड क्लास **maximum** के मेंम्बर

12.4 मल्टीलेवल इनहेरिटेंस

एक क्लास को किसी दूसरी डिराइव्ड क्लास से भी डिराइव किया जा सकता है।



चित्र 12.3 मल्टीलेबल इनहेरिटेंस

मल्टीलेवल इनहेरिटेंस में लेवल की कोई सीमा नहीं होती है। निम्नलिखित प्रोग्राम मल्टीलेवल इनहेरिटेंस का उदाहरण है

प्रोग्राम 12.2 मल्टीलेवल इनहेरिटेंस

```
#include<iostream>
using namespace std;
class data1
{
 protected:
   int x;
 public:
          void get_x(int a)
          {
                x=a;

          }
         void show_x(void)
         {
               cout<<"x="<<x<<"\n";
          }
};
```

```
class data2:public data1
{
  protected:
     int y;
 public:
              void get_y(int b)
              {
                     y=b;
               }
             void show_y(void)
             {
                    cout<<"y="<<y<<"\n";
            }
};
class addition: public data2
{
            int z;
 public:
           void sum(void)
           {
                      z=x+y;
           }
           void show_z(void)
           {
                     cout<<"z="<<z<<"\n";
           }
};

int main()
{
             addition a;
             a.get_x(4);
```

```
        a.get_y(7);
        a.sum();
        a.show_x();
        a.show_y();
        a.show_z();
      return 0;
}
```

प्रोग्राम 12.2 का आउटपुट होगा–

```
x=4
y=7
z=11
```

उपरोक्त प्रोग्राम में, डिराइव्ड क्लास 'data2' को बेस क्लास 'data1' से डिराइव की गयी है और यह डेरिवेशन का पहला लेवल है। बेस क्लास 'data1' का प्रोटेक्टेड डेटा मेंम्बर x डिराइव्ड क्लास 'data2' में प्रोटेक्टेड बन जाता है। डेरिवेशन के पहले लेवल के बाद डिराइव्ड क्लास 'data2' में निम्नलिखित मेंम्बर होगें।

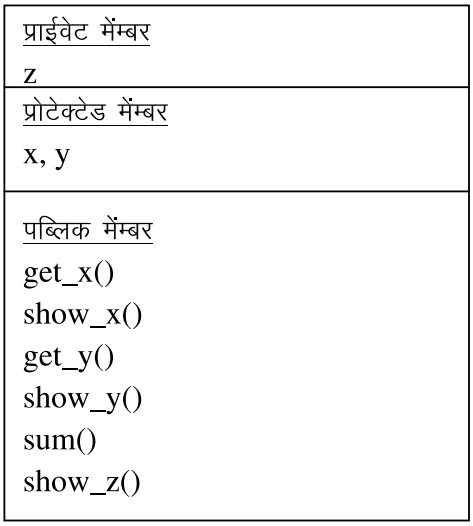
डिराइव्ड क्लास 'data2'

प्रोटेक्टेड मेंम्बर
x, y
पब्लिक मेंम्बर
get_x()
show_x()
get_y()
show_y()

चित्र 12.4 डिराइव्ड क्लास 'data2' के मेंम्बर

क्लास 'addition' को अंतरिम बेस क्लास 'data2' से डिराइव किया गया है और यह इनहेरिटेन्स का दूसरा लेबल है। इनहेरिटेन्स के दूसरे लेवल के बाद डिराइव्ड क्लास 'addition' में निम्नलिखित मेंम्बर होगें।

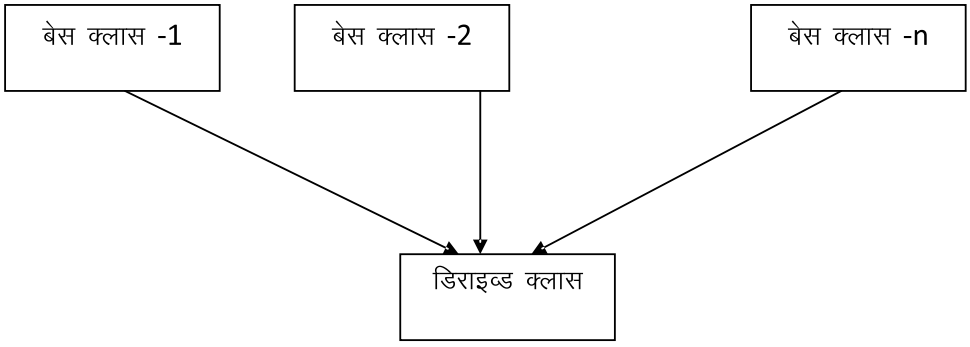
डिराइव्ड क्लास **'addition'**



चित्र 12.5 डिराइव्ड क्लास **'addition'** के मेंम्बर

12.5 मल्टीपल इनहेरिटेंस

जब एक क्लास को दो या दो से अधिक क्लासों के फीचर को इनहेरिट करती है, उसे मल्टीपल इनहेरिटेंस कहा जाता है।



चित्र 12.6 मल्टीपल इनहेरिटेंस

डिराइव्ड क्लास का सिन्टेक्स एक से अधिक बेस क्लासों के साथ इस प्रकार होता है।

```
class derived_class : visibility Base_class-1, visibility Base_class-2, - - - - - - - -
{
        Members of derived class
};
```

निम्नलिखित प्रोग्राम मल्टीपल इनहेरिटेंस का उदाहरण है।

प्रोग्राम 12.3 मल्टीपल इनहेरिटेंस

```
#include<iostream>
using namespace std;
class B1
{
protected:
        int x;
public:
        void get_x(int a)
        {
                x=a;
        }
};
class B2
{
protected:
        int y;
public:
        void get_y(int b)
        {
                y=b;
        }
};
class D : public B1, public B2
{
        int z;
public:
        void multiply(void)
        {
                z=x*y;
        }
        void display(void)
```

```
        {
                cout<<"x="<<x<<"\n";
                cout<<"y="<<y<<"\n";
                cout<<"z="<<z<<"\n";
        }
};
int main()
{
        D d;
        d.get_x(5);
        d.get_y(3);
        d.multiply();
        d.display();
        return 0;
}
```

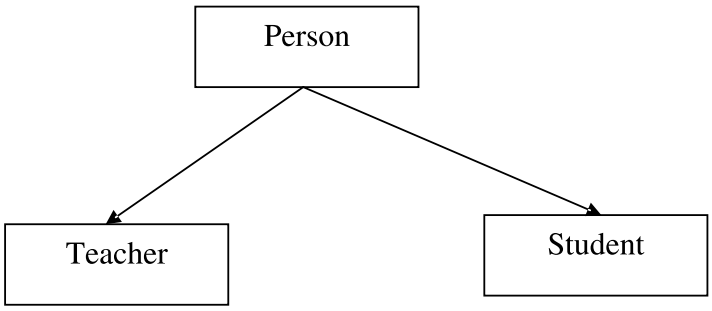
प्रोग्राम 12.3 का आउटपुट होगा–

```
x=5
y=3
z=15
```

12.6 हायरार्कीकल इनहेरिटेंस

जब एक बेस क्लास को दो या दो से अधिक डिराइव्ड क्लासों द्वारा इनहेरिट किया जाता है उसे हायरार्कीकल इनहेरिटेंस कहा जाता है।

उदाहरण के लिए एक विद्यालय में लोगों का वर्गीकरण चित्र 12.8 में दर्शाया गया है।



चित्र 12.8 हायरार्कीकल इनहेरिटेंस का उदाहरण

प्रोग्राम 12.4 हायरार्कीकल इनहेरिटेंस

```
#include<iostream>
#include<string.h>
using namespace std;
class person
{
 protected:
        char name[20];
        int age;
 public:
          void get_person(const char *n, int a)
          {
                strcpy(name,n);
                age=a;
          }
          void show_person(void)
          {
                cout<<"Name:"<<name<<"\n";
                cout<<"Age:"<<age<<"\n";
          }
};
class teacher : public person
{
 char post[10];
 public:
        void get_post(const char *p)
        {
                strcpy(post,p);
        }
        void show_teacher(void)
        {
                show_person();
```

```
                cout<<"post:"<<post<<"\n";
        }
};
class student : public person
{
         int standard;
public:
        void get_standard(int s)
        {
                 standard=s;
         }
         void show_student(void)
         {
                cout<<"Standard:"<<standard<<"\n";
        }
};
int main()
{
          teacher t;
          t.get_person("Ram",30);
          t.get_post("TGT");
          student s;
          s.get_person("Shyam",17);
          s.get_standard(12);
          t.show_teacher();
          s.show_student();
        return 0;
}
```

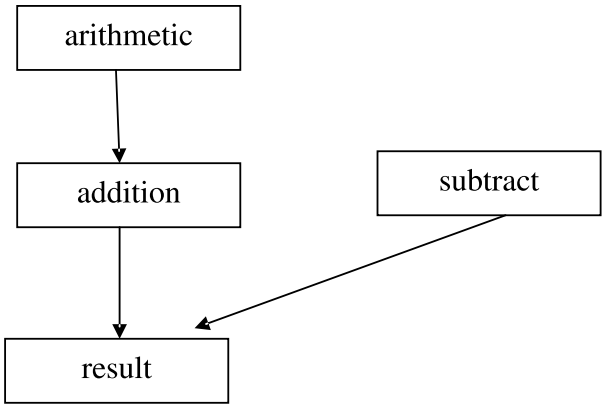
प्रोग्राम 12.4 का आउटपुट होगा–

```
Name: Ram
Age: 30
```

Post: TGT
Name: Shyam
Age: 17
Standard: 12

12.7हाइब्रिड इनहेरिटेंस

दो या दो से अधिक इनहेरिटेंस के प्रारूपों के सम्मलित रूप को हाइब्रिड इनहेरिटेंस कहा जाता है। उदाहरण के लिए चित्र 12.9 में हाइब्रिड इनहेरिटेंस दर्शाया गया है। जो कि मल्टीलेवल इनहेरिटेंस और मल्टीपल इनहेरिटेंस का सम्मलित रूप है।



चित्र 12.9 हाइब्रिड इनहेरिटेंस का उदाहरण

प्रोग्राम 12.5हाइब्रिड इनहेरिटेंस

```
#include<iostream>
using namespace std;
class arithmetic
{
protected:
        int num1, num2;
public:
        void getdata(void)
        {
                cout<<"For Addition:";
                cout<<"\nEnter the first number: ";
```

```
                cin>>num1;
                cout<<"\nEnter the second number: ";
                cin>>num2;
        }
};
class addition:public arithmetic
{
protected:
        int sum;
public:
        void add(void)
        {
                sum=num1+num2;
        }
};

class subtract
{
protected:
        int n1,n2,diff;
public:
        void sub(void)
        {
                cout<<"\nFor Subtraction:";
                cout<<"\nEnter the first number: ";
                cin>>n1;
                cout<<"\nEnter the second number: ";
                cin>>n2;
                diff=n1-n2;
        }
};
class result:public addition, public subtract
```

```
{
public:
        void display(void)
        {
                cout<<"\nSum of "<<num1<<" and "<<num2<<"=
                "<<sum;
                cout<<"\nDifference of "<<n1<<" and "<<n2<<"= "<<diff;
        }
};

int main()
{
        result z;
        z.getdata();
        z.add();
        z.sub();
        z.display();
        return 0;
}
```

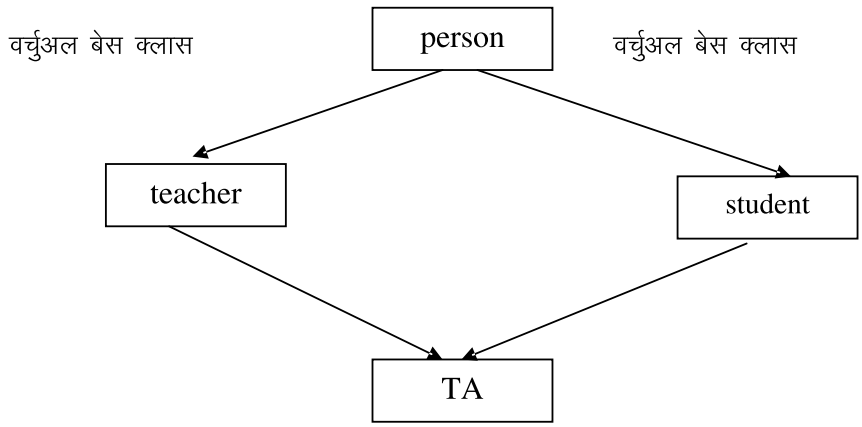
प्रोग्राम 12.5 का आउटपुट होगा–

```
For Addition:
Enter the first number: 5
Enter the second number: 7
For Subtraction:
Enter the first number: 10
Enter the second number: 3
Sum of 5 and 7 is 12
Difference of 10 and 3 is 7
```

12.8 वर्चुअल बेस क्लासेज

एक हाइब्रिड इनहेरिटेंस का उदाहरण लेते है जिसमें इनहेरिटेंस के तीन प्रारूप जो कि मल्टीलेवल, मल्टीपल और हायरार्किकल सम्मलित है जैसे कि चित्र 12.6 में दर्शाया गया है।

क्लास 'TA' (शिक्षक सहायक) की दो प्रत्यक्ष बेस क्लासेज 'teacher' और 'student' जिनकी एक समान बेस क्लास 'person' है। 'TA' क्लास 'person' क्लास के फीचर को दो तरीकों से इनहेरिट करती है। यह स्थिति एक समस्या उत्पन्न करती है। 'person' क्लास के सभी प्रोटेक्टेड और पब्लिक मेंम्बर 'TA' क्लास में दो बार इनहेरिट हो जाते है। पहला क्लास 'teacher' होते हुये और दूसरा क्लास 'student' होते हुये। यह एक अस्पष्टता की स्थिति उत्पन्न करती है और इसे दूर करना चाहिए।



चित्र 12.9 वर्चुअल बेस क्लास

यह अस्पष्टता एक समान बेस क्लास को वर्चुअल बेस क्लास बनाकर प्रत्यक्ष बेस क्लासों को घोषित करने के दौरान दूर की जा सकती है जैसा नीचे दर्शाया गया है।

```
class person
{
   --------
   --------
};
class teacher : virtual public person
{
   --------
   --------
};
class student : virtual public person
{
```

```
    --------
    --------
};
class TA: public teacher, public student
{
    --------
    --------
};
```

जब एक क्लास को वर्चुअल बेस क्लास घोषित करते है तब केवल उस क्लास के पब्लिक और प्रोटेक्टेड मेंम्बर की एक ही प्रतिलिपी इनहेरिट होती है।

12.9 एबस्ट्रेक्ट क्लासेज

अगर एक ही नाम के फंक्शनस बेस और डिराव्ड दोनों क्लासों में प्रयोग किया जाता है, तब बेस क्लास के फंक्शन को वर्चुअल घोषित किया जाता है, उसे वर्चुअल फंक्शन कहा जाता है। एक बिना स्टेटमेंट के वर्चुअल फंक्शन को प्योर वर्चुअल फंक्शन कहा जाता है। एक क्लास में कम से कम एक प्योर वर्चुअल फंक्शन हो तो उस क्लास को एबस्ट्रेक्ट क्लास कहा जाता है। इसका प्रयोग ऑब्जेक्ट बनाने में नहीं किया जाता है। इसका प्रयोग केवल बेस क्लास के रूप में किया जाता है जिसे दूसरी क्लासों द्वारा इनहेरिट किया जाता है। प्योर वर्चुअल फंक्शन को उस क्लास के द्वारा परिभाषित करना जरूरी है जो एबस्ट्रेक्ट क्लास से डिराइव्ड हो। निम्नलिखित प्रोग्राम एबस्ट्रेक्ट क्लास का एक उदाहरण है

प्रोग्राम 12.6एबस्ट्रेक्ट क्लासेज

```
#include <iostream>
using namespace std;
 class Shape
{
protected:
            int width;
            int height;
public:
  virtual int area() = 0;          // pure virtual function
  void getdata(int w, int h)
  {
     width=w;
     height=h;
```

```
  }
};

class Rectangle: public Shape
{
public:
 int area()
  {
    return (width * height);
  }
};
class Triangle: public Shape
 {
public:
  int area() {
    return (width * height)/2;
  }
};
 int main(void)
 {
  Rectangle Rect;
  Triangle  Tri;
   Rect.getdata(5,7);
  cout << "Area of Rectangle : " << Rect.area() <<"\n";
  Tri.getdata(6,7);
  cout << "Area of Triangle : " << Tri.area() <<"\n";
  return 0;
}
```

प्रोग्राम 12.6 का आउटपुट होगा–

Area of Rectangle: 35

Area of Triangle: 21

महत्वपूर्ण बिंदु

- पहले से तैयार क्लासों से नई क्लास बनाने की प्रकिया को इनहेरिटेंस कहा जाता है।
- डिफॉल्ट विजिबिलिटी मोड प्राईवेट होता है।
- सिंगल इनहेरिटेंस में एक बेस क्लास और एक डिराइव्ड क्लास होती है।
- मल्टीलेवल इनहेरिटेंस में लेवल की कोई सीमा नहीं होती है।
- जब एक क्लास को दो या दो से अधिक क्लासों के फीचर को इनहेरिट करती है, उसे मल्टीपल इनहेरिटेंस कहा जाता है।
- जब एक बेस क्लास को दो या दो से अधिक डिराइव्ड क्लासों द्वारा इनहेरिट किया जाता है उसे हायरार्कीकल इनहेरिटेंस कहा जाता है।
- दो या दो से अधिक इनहेरिटेंस के प्रारूपों को सम्मलित रूप को हाइब्रिड इनहेरिटेंस कहा जाता है।
- जब एक क्लास को वर्चुअल बेस क्लास घोषित करते है तब केवल उस क्लास के पब्लिक और प्रोटेक्टेड मेंम्बर की एक ही प्रतिलिपी इनहेरिट होती है।
- एक क्लास में कम से कम एक प्योर वर्चुअल फंक्शन हो तो उस क्लास को एबस्ट्रेक्ट क्लास कहा जाता है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न :

प्रश्न 1. इनहेरिटेंस में पहले से तैयार क्लास को क्या कहा जाता है?

(अ) बेस क्लास (ब) पेरेंट क्लास (स) सुपर क्लास (द) उपरोक्त सभी

प्रश्न 2. इनहेरिटेंस में नई क्लास को क्या कहा जाता है?

(अ) डिराव्इड क्लास (ब) चाइल्ड क्लास (स) सब क्लास (द) उपरोक्त सभी

प्रश्न 3. डिफॉल्ट विजिबिलिटी मोड क्या होता है?

(अ) पब्लिक (ब) प्राईवेट (स) प्रोटेक्टेड (द) इनमें से कोई नहीं

प्रश्न 4. इनहेरिटेंस जिसमें एक बेस क्लास और एक डिराइव्ड क्लास होती है उसे क्या कहा जाता है?

(अ) सिंगल इनहेरिटेंस (ब) मल्टीलेवल इनहेरिटेंस

(स) मल्टीपल इनहेरिटेंस (द) हायरार्किकल इनहेरिटेंस

प्रश्न 5. जब एक क्लास को दो या दो से अधिक क्लासों के फीचर को इनहेरिट करती है, उसे क्या कहा जाता है?

(अ) सिंगल इनहेरिटेंस (ब) मल्टीलेवल इनहेरिटेंस

(स) मल्टीपल इनहेरिटेंस (द) हायरार्किकल इनहेरिटेंस

अति लघूउत्तरात्मक प्रश्न

प्रश्न 1. इनहेरिटेंस किसे कहते है ?

प्रश्न 2. सिंगल इनहेरिटेंस किसे कहते है ?

प्रश्न 3. मल्टीलेवल इनहेरिटेंस किसे कहते है ?

प्रश्न 4. मल्टीपल इनहेरिटेंस किसे कहते है ?

प्रश्न 5. हायरार्किकल इनहेरिटेंस किसे कहते है ?

प्रश्न 6. हाइब्रिड इनहेरिटेंस किसे कहते है ?

प्रश्न 7. एबस्ट्रेक्ट क्लास किसे कहते है ?

लघूउत्तरात्मक प्रश्न

प्रश्न 1. विजिबिलिटि मोड का इनहेरिटेंस में प्रभाव का वर्णन कीजिए ?

प्रश्न 2. वर्चुअल बेस क्लास की अवधारणा क्या है ?

निबंधात्मक प्रश्न

प्रश्न 1. एक shape क्लास बनाइए जिसमें एक प्योर वर्चुअल फंक्शन volume हो। shape क्लास को तीन क्लासों cone, cylinder और cube द्वारा इनहेहिट किया हो ये डिराइव्ड क्लासेज प्योर वर्चुअल फंक्शन को आयतन की गणना के लिए परिभाषित करेगें ?

उत्तरमाला

1:द 2:द 3:ब 4:अ 5:स

अध्याय 13

DBMS अवधारणायें

फाइल सिस्टम का परिचय

फाइलों के एक सेट को स्टोर, पुनः प्राप्त और update करने के लिए एक abstraction को फाइल सिस्टम कहा जाता है । एक फाइल सिस्टम में उन चीजें द्वारा निर्दिष्ट(Specified) डेटा संरचनाएँ (structure) भी शामिल हैं। ये डेटा संरचनाएँ multiple फाइलों को बाइट्स की एक धारा के रूप में व्यवस्थित करने के लिए डिजाइनकी गई हैं। एक फाइल सिस्टम में अन्य abstraction नेटवर्क प्रोटोकॉल को भी निर्दिष्ट(Specified) करते हैं। ये किसी दूरस्थ मशीन पर फाइलों तक पहुँच(access) की अनुमति के लिए डिजाइन किएगयेहैं।फाइल सिस्टम डेटा और मेटाडेटा फाइलों के लिए पहुँच प्रबंधित करता है।एक फाइल सिस्टम विश्वसनीयता(reliability) सुनिश्चित करता है और यह सिस्टम की प्रमुख जिम्मेदारी है।

फाइल सिस्टम की हानियां(problems):

- **डेटा redundancy** : एक ही सूचना कई फाइलों में उपलब्ध है। जैसे छात्र पता विभिन्न प्रयोजनों के लिए अलग अलग फाइलों में उपलब्ध है।
- **डेटा Access difficulty**: इसमे जब नया अनुरोध(request) आता है तब नए प्रोग्राम की आवश्यकता होती है क्योंकि, हर समय नया प्रोग्राम नये अनुरोध को पूर्ण करने के लिए उपलब्ध नहीं होता है इसलिये नये आने वाले अनुरोध को पूर्ण करने के लिए नए प्रोग्राम लिखने की आवश्यकता होती है।
- **डेटा isolated है**: डेटा अलग अलग फाइलों में, अलग अलग स्वरूप में है।
- Multiple उपयोगकर्ता, एक ही डेटा एक साथ उपयोग नहीं कर सकते क्योंकि समानांतर अनुरोध के लिए पर्यवेक्षण में कठिनाई होती है।
- डाटाबेस में सुरक्षा को लागू करने में कठिनायाँ आती है।
- **अखंडता संबंधी समस्याये(Integrity issues)**: डेटाबेस पर बाधाओं को सुनिश्चित किया जाना चाहिए।

फाइल सिस्टम के लाभ:

- एकल–अनुप्रयोग(single application) के साथ आसान डिजाइन।
- एक एकल अनुप्रयोग आधारित उन्नत (optimized) संगठन ।
- प्रदर्शन में कुशल।

HIERARCHY OFडेटा:कंप्यूटर सिस्टम में स्टोर किए गए डेटा को निम्न तरह से view किया जा सकता है और हम उसके बाद डेटाबेस प्रबंधन प्रणाली को परिभाषित कर सकते हैं। डेटा तार्किक(logical)रूप मेंनिम्नानुसार organized हैं

1. बिट्स
2. फील्ड
3. रिकॉर्ड
4. फाइलें
5. डेटाबेस

बिट –एक बिट डेटा प्रतिनिधित्व की सबसे छोटी इकाई है (एक बिट का मान एक 0 या 1 हो सकता है)

फील्ड –एक फील्ड characters का एक समूहीकरण होता हैं। एक डेटा फील्ड, एक एंटिटी (object, person, place, or event) के attributes(एक विशेषता या गुणवत्ता) का प्रतिनिधित्व करता है।

रिकॉर्ड– एक रिकॉर्ड attribute के एक संग्रह का प्रतिनिधित्व करता है जो एक वास्तविक दुनिया कि एंटिटी का वर्णन है।

एक रिकॉर्ड में फील्ड होते हैं, प्रत्येक फील्ड एक एंटिटी की attributeका वर्णन करते हैं।

फाइल–संबंधित अभिलेखो का एक समूहहै।एक फाइल में एक प्राथमिक कुंजी वह फील्ड है जिसका मान एक डेटा फाइल में एक रिकॉर्ड की पहचान करता है।

अब हम डेटाबेस प्रबंधन प्रणाली को परिभाषित कर सकते हैं।जैसा नाम से पता चलता है, डेटाबेस प्रबंधन सिस्टम,डेटाबेस और प्रबंधन सिस्टमसे बना है।

डेटाबेस: डेटाबेस क्या है?यह समझने के लिए, हम डेटा से प्रारंभ करते है जो किसी DBMS-का मूलभूत बिल्डिंग ब्लॉक है।

डेटा :तथ्य, figures, आँकड़े आदि का कोई विशेष अर्थ न हो (जैसे 2, XYZ, 19 आदि).

रिकॉर्ड :संबंधित डेटा आइटम का संग्रह।

फाइल– संबंधित अभिलेखों का एक समूह।

डेटाबेस :Interrelated डेटा या डेटा फाइलें या संबंध(संबंधपरक डेटाबेस) का संग्रह।डेटा का यह संग्रह interrelated है तो यह एक संगठन की एक प्रासंगिक जानकारी हो सकती है और जानकारी के इस संग्रह तक application प्रोग्राम के एक सेट का उपयोग करके पहुँचा जा सकता है।

प्रबंधन प्रणाली (**Management system**) : एक प्रबंधन प्रणाली नियमों का एक सेट औरवे प्रक्रियायें है जो डेटाबेस को बनाने में, हेरफेरकरने में और व्यवस्थित करने के लिए हमारी मदद करता है। यह डेटाबेस में डेटा आइटम्स को हटाने, संशोधित करने और जोड़ने के लिए भी हमें मदद करताहै।

DBMS

एक डेटाबेस प्रबंधन सिस्टम (DBMS) interrelated डेटा काएक संग्रह और उन डेटा तक पहुँच प्राप्त करने के लिए प्रोग्राम्स का एक सेट है।

DBMS का लक्ष्य: किसी भी DBMS सिस्टम डिजाइन के निम्न लक्ष्य है

1. मुख्य लक्ष्य किसी भी DBMS सिस्टम की बड़ी निकायों की जानकारी का प्रबंधन करने के लिए है।
2. डेटाबेस में जानकारी संग्रहीत करने के लिए सुविधाजनक तरीका प्रदान करते हैं।
3. डेटाबेस से कुशलता से जानकारी की पुनः प्राप्ति।
4. डेटाबेस में संग्रहीत जानकारी की सुरक्षा।
5. कई उपयोगकर्ताओं द्वारा जानकारी के simultaneous access के दौरान विसंगतियों से बचना।

DBMS के लाभ:

पारंपरिक फाइल सिस्टम पर DBMS के कई फायदे हैं। ये फायदे DBMS को कई अनुप्रयोगों(applications) में और अधिक उपयोगी बनाते हैं। DBMS के निम्नलिखित लाभ हैं।

1. **डेटा redundancy(duplicacy) निकालना:** एक ही सूचना कई स्थानों में संग्रहीत है, तो संग्रहण स्थान और प्रयास बर्बाद होगा। इस redundancy कि समस्या को DBMS ने संभाला है।
2. **डेटा sharing:** कम्प्यूटरीकृत DBMS में, कई उपयोगकर्ता एक ही डेटाबेस को साझा कर सकते हैं।
3. **डेटा Integrity:** हम डेटा intigrity को डेटा intigrity constrains specification द्वारा बनाए रख सकते हैं, जोकि डेटाबेस मे किस प्रकार का डेटा दर्ज

हो सकता है और manipulate हो सकता है के बारे नियम और प्रतिबंध हैं। यह डेटाबेस की विश्वसनीयता को बढता है क्योंकि यह गारंटी देता हैं की समय के किसी भी बिंदु पर डेटाबेस में मौजूद डेटा गलत नहीं हो सकता हैं।

4. **डाटा independence:** application प्रोग्राम जितना संभव हो डेटा प्रतिनिधित्व और भंडारण(storage) के विवरण से independent होना चाहिए। application कोड को ऐसे विवरण से insulate करने के लिए DBMS डेटा का कोई abstract view प्रदान कर सकते हैं।

5. **कुशल डेटा पहुँच (Efficient data access):** एक DBMS कुशलता से डेटा को पुनर्प्राप्त करने के लिए और डेटा संग्रहीत करने के लिए कई किस्म की परिष्कृत(sophisticated) तकनीक का इस्तेमाल कर सकते हैं। अगर डेटा बाह्य भंडारण उपकरणों पर है,तो यह सुविधा विशेष रूप से महत्वपूर्ण हैं।

6. **डेटा अखंडता(integrity) और सुरक्षा:** डेटा का abstract view और integrity constraints उपलब्ध कराने के द्वारा सुरक्षा को सुनिश्चित किया जा सकता हैं। DBMS डेटा का abstract view प्रदान करता है ताकि सभी प्रकार के उपयोगकर्ताओं को सभी प्रकार की जानकारी देखने की जरूरत नहीं है।उपयोगकर्ता द्वारा केवल डेटाबेस के विशेष भाग को देखा जा सकता है।

7. **Reduced application development time:** DBMS स्पष्ट रूप से कई महत्वपूर्ण फंक्शन का समर्थन करता है जो DBMS में संग्रहीत डेटा तक पहुँचने के लिए काम आने वाली ऐसी कई applications के लिए common हैं। यह डेटा के उच्च स्तरीय इंटरफेस के साथ, applications के त्वरित विकास में सुविधा के लिए है।

8. **Recovery in DBMS:**Transaction की विफलता के दौरान डेटाबेस उसकी मूल स्थिति में पुनर्स्थापित हो जाएगा।

Applications of DBMS: लगभग सभी क्षेत्र मे, DBMS के applications है । इनमें से कुछ हैं

- **बैंकिंग**: बैंकिंग क्षेत्र के सभी लेन–देन
- **एयरलाइन:**reservation, schedules, availability
- **विश्वविद्यालयों:**पंजीकरण, ग्रेड।
- **बिक्री:** ग्राहकों, उत्पादों, खरीद।
- **निर्माण:** उत्पादन, माल, आदेश, आपूर्ति श्रृंखला।
- **मानव संसाधन:**कर्मचारी अभिलेखों, वेतन, कर की कटौती।

DBMS का उदाहरण : DBMS जो वर्तमान में उपयोग में हैं।

वाणिज्यिक DBMS:

कंपनी	उत्पाद
Oracle	8i, 9i, 10g
IBM	DB2, यूनिवर्सल सर्वर
Microsoft	Access, SQL सर्वर
Sybase	Adaptiveसर्वर
Informix	डायनेमिकसर्वर

वाणिज्यिकDBMSके साथ, एक व्यापक रूप से इस्तेमाल किया जाने वाला open source DBMS MySQLहै।

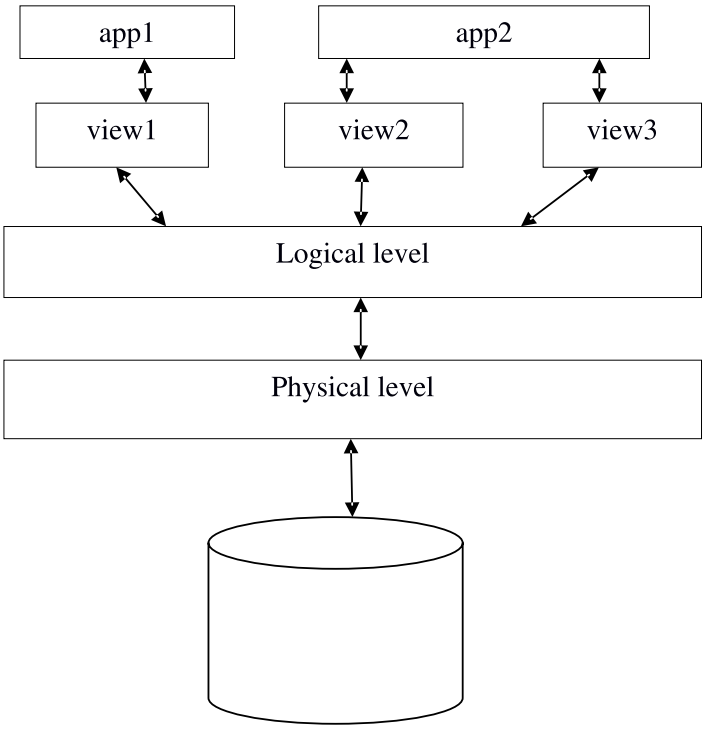
DBMS में abstraction levels : हम ने चर्चा की है की किसी भी DBMS का मुख्य लक्ष्य डेटाबेस के साथ उपयोगकर्ता की interaction को सरल रूप में बनाने के लिए है किसी भी तरह के उपयोगकर्ता(naïve, programmers, sophisticated आदि) किसी भी तरह आसानी से और कुशलता से डेटाबेस से जानकारी प्राप्त कर सकते हैं। डेटा कैसे संग्रहीत और बनाए रखा(maintained) है जैसे कुछ विवरण को छुपाने के लिए डेटा का abstract view मदद करता है।

Physical level:भौतिक स्कीमा निर्दिष्ट(pecifiè) करता है की कैसे रिलेशन वास्तव में द्वितीयक संग्रह डिवाइस में संग्रहीत की जाती हैं। यह रिलेशन की गति बढ़ाने के लिए इस्तेमाल सहायक डेटा संरचनाएँ (इंडेक्स) को भी निर्दिष्ट करता है।

Logical (वैचारिक) level: वैचारिक स्कीमा डेटाबेस में संग्रहीत डेटा और उन डेटा के बीच संबंध का वर्णन करता है, वैचारिक स्कीमा पूरे डेटाबेस के तार्किक संरचना को परिभाषित करता है। उदाहरण के लिए, एक रिलेशनल डेटाबेस में यह डेटाबेस में संग्रहीत सभी संबंधों का वर्णन करता है।

View level:यह वैचारिक स्कीमा का एक शोधन(refinement)है। यह व्यक्तिगत उपयोगकर्ताओं या उपयोगकर्ताओं के समूह के लिए अधिकृत पहुँच और अनुकूलित की अनुमति देता है। हर डेटाबेस एक वैचारिक और एक भौतिक स्कीमा रखते है, लेकिन यह view level पर कई स्कीमारख सकता है। एक view (external स्कीमा) conceptualy एक रिलेशन है, लेकिन इसके रिकॉर्ड डेटाबेस में संग्रहीत नहीं हैं इसके बजाय, वे अन्य संबंधों से अभिकलन(computed) हैं।

चित्र 1 abstraction के इन स्तरों के बीच संबंध दिखाता है।



चित्र 1 डेटा**abstraction levels**

स्कूल डेटाबेस उदाहरण: ऊपर चित्र 1 में दिखाये गये तीन स्कीमा स्कूल डेटाबेस के उदाहरण का उपयोग करके समझे जा सकते है।

भौतिक स्कीम उदाहरण:

रिलेशन unordered फाइलें और students के पहला स्तंभ परindex के रूप में संग्रहीत किया जा सकता है।

वैचारिक(conceptual) स्कीमा उदाहरण:

Students(Roll_no: int, name: varchar, address: varchar, age: integer, class: char)

Subjects(subjectid: char, Sname: char,)

admission(Roll_no: int, ClassName:char, AdmissionDate: date)

External स्कीमा (view)उदाहरण:

Class_info(ClassName:char, Strength:integer)

स्कीमाज और instances:

एक स्कीमा (over all डिजाइन) डेटा का ,डेटा मॉडल के संदर्भ में एक विवरण है। हम एक स्कीमा के बारे में कह सकते हैं कि, डेटा तार्किक(logically) रूप से कैसे संरचित किया जा सकता हैं और शायद ही कभीउसमेपरिवर्तन हो,एक स्कीमा इसकाएक specification है ।

रिलेशनल मॉडल में स्कीमा इस तरह दिखता है:

RelationName (field1: type1,..., fieldn: typen)

Students(Roll_no : int, name : char, age : integer, class : char)

दूसरी ओर एक instance समय के किसी भी विशेष क्षण पर स्कीमा की सामग्री(content) का प्रतिनिधित्व करता है और जो तेजी से बदलता है, लेकिन हमेशा एक स्कीमा के अनुरूप होता हैं। हम स्कीमा और intances की तुलना एक प्रोग्रामिंग भाषा में type और ऑब्जेक्ट्स के type के साथ कर सकते हैं।

students रिलेशन का एक instance, चित्र 2 में निम्नानुसार हैं।

Student table

Roll_no	Name	Age	Address	class
101	Harish	10	Ajmer	5th
105	kailash	20	kota	10th
109	Manish	18	Ahmadabad	9th
120	Ronak	14	Udaipur	8th
135	shanker	13	jaipur	7th

चित्र 2: **student table** का एक **instance** और **student** विवरण

डेटाबेस भाषाओं: एक डेटाबेस सिस्टम में डेटा definition language (DDL) जो डेटा स्कीमा कोनिर्दिष्ट(specifies) करने के लिए और डेटा Manipulation Language (DML) जो डेटाबेस से डेटा के पुनर्प्राप्ति सुविधा के लिए और डेटा update के लिए है। DML मूल रूप से दो प्रकार के होते हैं।

1. **Procedural DML:**इसमें यह आवश्यकता है कि, उपयोगकर्ता द्वारा डेटा और उन डेटा को प्राप्त करने के लिए कि प्रक्रिया(procedure) को specified किया जाना चाहिए ।

2. **Non–procedural DML:** Non–procedural DML में केवल आवश्यक डेटा उपयोगकर्ता द्वारा specified किया जाता है उन डेटा के प्राप्त करने के लिए कि प्रक्रिया को बिना specified करे।एक DBMS, DBMS से सवाल पूछने के लिए एक विशेष भाषा प्रदान करता है जिसेक्वेरी language बुलाया जाता है। पुनः डेटाबेस से प्राप्ति के लिए, हम क्वेरी भाषा जो DML का हिस्सा है, के द्वारा डेटाबेस से क्वेरी की जाती हैं। क्वेरी language और DMLशब्द पर्याय हैं।

DBMS के वर्गीकरण:एक DBMS सिस्टम निम्न मापदंड पर आधारित कई प्रकार के हो सकते हैं।

1. **उपयोगकर्ताओं(users) पर आधारित:**सिस्टम के द्वारा समर्थित (supported)उपयोगकर्ताओं की संख्या एक प्रथम मापदंड है। Single–user सिस्टम्स एक समय में केवल एक उपयोगकर्ता का समर्थन करता है और ज्यादातर पर्सनल कंप्यूटर के साथ उपयोग किया जाता है । Multiuser सिस्टम, multiple उपयोगकर्ताओं का concurrently समर्थन करता है और majority DBMS इसमें शामिल हैं,

2. **Architecture पर आधारित:**कंप्यूटर सिस्टम की संख्या जिस पर डेटाबेस सिस्टम चलता है एक दूसरा मापदंड है । एक centralized या क्लाइंट–सर्वर DBMS एक से अधिक उपयोगकर्ताओं का समर्थन कर सकते हैं, लेकिन DBMS और डेटाबेस स्वयं पूरी तरह से एक एकल कंप्यूटर साइट (सर्वर मशीन) पर रहते हैं और चलता है। एक वितरित (distributed) DBMS (DDBMS) में वास्तविक डेटाबेस और DBMS सॉफ्टवेयर एक कंप्यूटर नेटवर्क से जुड़े कई साइटों पर वितरित किया जाता है। सजातीय DDBMSs एकाधिक साइटों पर एक ही DBMS सॉफ्टवेयर का उपयोग करते हैं।

3. **डेटा मॉडल के प्रकार पर आधारित:** डेटा मॉडल के प्रकार जिस पर DBMS आधारित है तीसरे मापदंड है। DBMS निम्न प्रकार के हो सकते हैं।

- पदानुक्रम (Hierarchical) डेटाबेस।
- नेटवर्क डेटाबेस।
- संबंधपरक(रिलेशनल) डेटाबेस।
- ऑब्जेक्ट–ओरिएंटेड डेटाबेस।

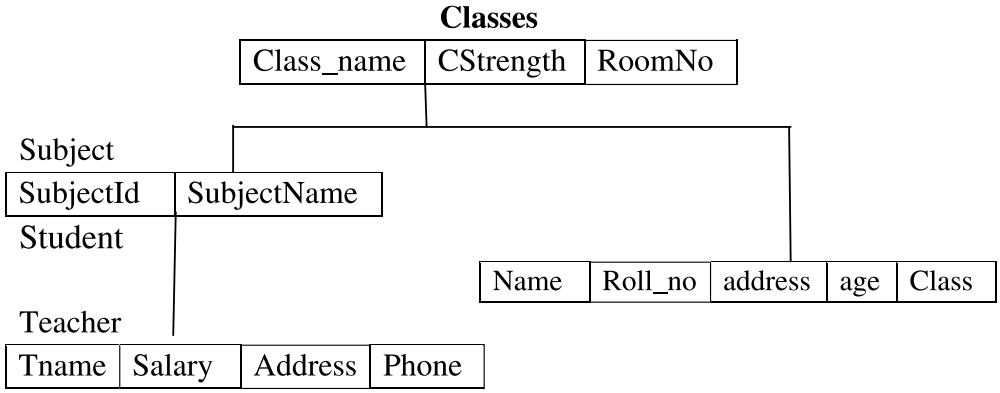
डेटा मॉडल:

एक डेटा मॉडल डेटा, डेटा पर प्रतिबंध और डेटा संबंध के अर्थ का वर्णन करने के लिए अवधारणाओं का एक संग्रह है। Hierarchical, नेटवर्क, संबंधपरक और Object–oriented कुछ डेटा मॉडल हैं।

श्रेणीबद्ध (Hierarchical) डेटा मॉडल :

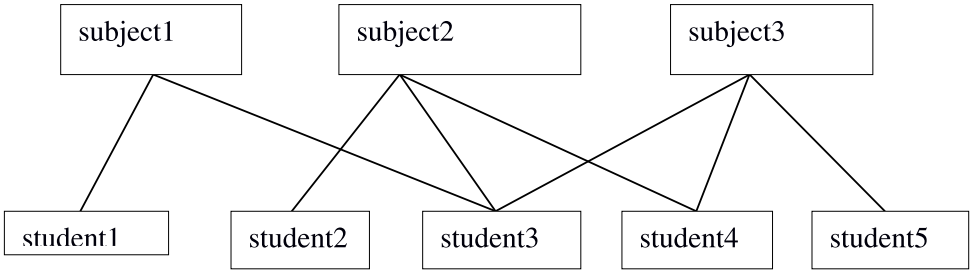
हम पदानुक्रमित मॉडल पर आधारित पुराने सिस्टम देख सकते हैं। पहला पदानुक्रमित DBMS "IMS" था और यह 1968 में जारी किया गया था। पदानुक्रमित DBMS एक–से–अनेक (one–to–many relationships) संबंध के मॉडल लिए उपयोग किया जाता है जो डेटा कोएक treelike संरचना में उपयोगकर्ताओं के लिए पेश किया जाता है। प्रत्येक रिकॉर्ड के भीतर, डेटातत्व, रिकॉर्ड्स के टुकड़ों में व्यवस्थित होते हैं जिन्हें (segments)खंडकहते हैं। उपयोगकर्ता के लिए, प्रत्येक रिकॉर्ड रूट(root) नामक एक toplevel खंड के साथ एक संगठनात्मक चार्ट की तरह लगता है। एक ऊपरी खंड तार्किक रूप से एक निचले खंड से एक पैरेंट–चाइल्ड संबंधों (relationship) से जुड़ा है। एक पैरेंट खंड के एक से अधिक चाइल्ड हो सकते हैं, लेकिन एक बच्चे के केवल एक पैरेंट हो सकते हैं। चित्र 1 से एक पदानुक्रमित संरचना का पता चलता है जो स्कूल प्रबंधन प्रणाली(schoolmanagement system) के लिए इस्तेमाल किया जा सकता है।

रूट खंड classes , जिसमें classes की बुनियादी जानकारी जैसे name, strength, और room number है। तुरंत नीचे दो child segments: subjects(subjectidऔर subject name डेटा युक्त), student (name , address, rollno , age और class डेटा युक्त) हैं। एक subject खंड के तुरंत नीचे teacher(teacher name , salary, address, phone and results evaluations)child segment है:



यह पाया है की बड़ी विरासत सिस्टम (large legacy systems) में जहां उच्च मात्रा लेनदेन प्रसंस्करण (high volume transaction processing) की आवश्यकता है श्रेणीबद्ध DBMS अभी भी इस्तेमाल किया जा सकता है। बैंकों, बीमा कंपनियों, और अन्य high volume उपयोगकर्ताओं को विश्वसनीय पदानुक्रम डेटाबेस का उपयोग कर रह हैं।

नेटवर्क डेटा मॉडल: एक नेटवर्क DBMS डेटा को अनेक–से–अनेक संबंध (many–to–many relationships) के तार्किक रूप में दर्शाती है।एक नेटवर्क DBMS के लिए एक many–to–many relationships student–subject relationships है (नीचे दिए गए चित्र 4 देखें)। एक class में कई subjects और कई students हैं। एक student कई subjects लेता है, और कई students एक विषय रखते है। पदानुक्रम (Hierarchical) और नेटवर्क DBMS पुराना माना जाता है और अब नए डेटाबेस अनुप्रयोगों(applications) के निर्माण के लिए इन्हें इस्तेमाल नहीं कर रहे हैं।



चित्र 4: **student- subject relationship** के लिए नेटवर्क मॉडल

संबंधपरक(रिलेशनल)डेटा मॉडल:

यह एक रिकॉर्ड आधारित डेटा मॉडल है। यह डेटा और डेटा के बीच संबंध का प्रतिनिधित्व करने के लिए रिलेशन (या table) का एक संग्रह का उपयोग करता है। हर रिलेशन attributes (या स्तंभ) की एक सूची रखता है जो अद्वितीय(unique) नाम रखते है। प्रत्येक attributes का एक डोमेन (या प्रकार) होताहै। प्रत्येक संबंध Viylका एक (या पंक्तियाँ) एक सेट रखता है। प्रत्येक टपल संबंध की प्रत्येक attribute के लिए एक मूल्य(value) रखता है। यहाँ डुप्लिकेट टपलस की अनुमति नहींहोती है। यह, ज्यादातर वर्तमान डेटाबेस सिस्टम के द्वारा प्रयुक्त डेटा मॉडल है। नीचे दिए गए table,studenttable का एक उदाहरण है जो student विवरण दिखाता है।

उदाहरण

Student table

Roll_no	Name	Age	Address	class
101	Harish	10	Ajmer	5th
105	Kailas	20	Kota	10th
109	Manish	18	Ahmadabad	9th

120	Ronak	14	Udaipur	8th
135	Shanker	13	Jaipur	7th

चित्रा 5: **Student** डेटाबेस के लिए संबंधपरक डेटा मॉडल

डेटाबेस डिजाइन चरण:

एक DBMS application किसी भी उद्यम(enterprise) के लिए डिजाइन करने के लिए कुछ चरणों का पालन करना चाहिए।

आवश्यकताओं के विश्लेषण (analysis):इस चरण में एक enterprise की डेटा आवश्यकताओं की पहचान की जातीहै।

वैचारिक डेटाबेस डिजाइन:ज्यादातर ईआर मॉडल का उपयोग कर किया जाता है। वैचारिक स्कीमा का निर्माण करने के लिए चुने हुए डेटा मॉडल की अवधारणा को, पहचान किये हुये डेटा पर लागू किया जाता है।

तार्किक(Logical) डेटाबेस डिजाइन:इस चरण में उच्च स्तर वैचारिक स्कीमा कोडेटाबेस सिस्टम के उपयोग होने वालीकार्यान्वयन डेटा मॉडल पर मेप किया जाएगा जैसे RDBMS के लिए यह संबंधपरक मॉडल है।

स्कीमा refinement: स्कीमा को छोटे स्कीमा में परिष्कृत(refine) करने के लिए सामान्यीकरण(normalization) लागू किया जाता है।

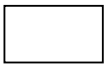
भौतिक(physical) डेटाबेस डिजाइन: यह चरण डेटाबेस के भौतिक सुविधाएँ(physical features) जैसे आंतरिक भंडारण संरचना, फाइल संगठन आदि निर्दिष्ट करता है।

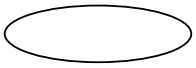
ई–आर(E–R) मॉडल और E–R आरेख:

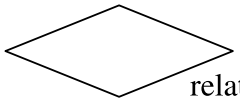
ई–आर(E–R) मॉडल:

एंटिटी – Relationship (E–R) मॉडल एक लोकप्रिय वैचारिक डेटा मॉडल है। मॉडल संग्रहीत डेटा और डेटा पर बाधाओं का वर्णन करता है। ई–आर मॉडल real world को एंटिटीज और एंटिटीज के बीच Relationship के रूप में दिखाता है। एक एंटिटी असली दुनिया(real world) में एक "ऑब्जेक्ट" जोअन्य ऑब्जेक्ट्स से विभक्त हो सकता है।

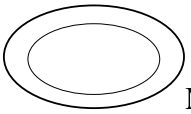
E–R आरेख: यह मूल रूप से एक डेटाबेस की सम्पूर्ण तार्किक संरचना का चित्रमय प्रतिनिधित्व है। इस आरेख में मुख्य घटक निम्नानुसार हैं–

प्रतीक(sumbol) का नाम	**प्रतीक**	**उद्देश्य**
आयत प्रतिनिधित्व करता है		एंटिटी के सेट का

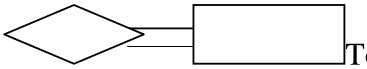
दीर्घवृत्त  attributes का प्रतिनिधित्व करता है

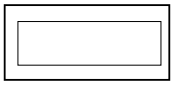
Diamonds  relationships का प्रतिनिधित्व करता है

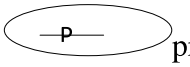
Lines एंटिटी के सेट और एंटिटी के attributes और एंटिटी के सेट और relationship सेट के बीच संबंध का प्रतिनिधित्व करता है

डबल दीर्घवृत्त  Multivalued attributes

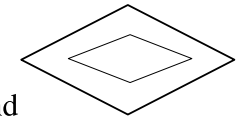
Dashed दीर्घवृत्त  derived attributes का प्रतिनिधित्व करता है

डबल लाइन  Total भागीदारी

डबल आयत  कमजोर एंटिटी सेट का प्रतिनिधित्व करता है

लाइन के साथ एक दीर्घवृत्त 

primary key

Double diamond  Identifying relationship कमजोर एंटिटी के लिए

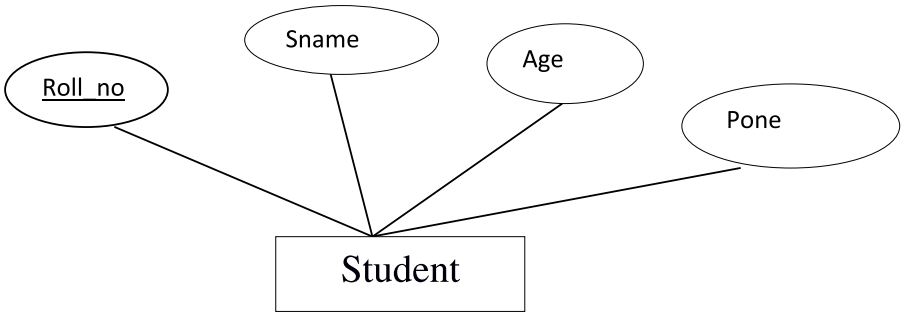
एंटिटी:असली दुनिया में एक "ऑब्जेक्ट" है जिसकी अन्य सभी वस्तुओं से अलग पहचान है। उदाहरण के लिए, एक class, एक teacher, teacher का address, एक student, एक

subject। एक एंटिटी का वर्णन attributes के एक सेट का उपयोग करकिया जा सकता है। प्रत्येक attribute संभव मानों का एक डोमेन रखता है.

एंटिटी सेट: एंटिटी सेट समान ऑब्जेक्ट (उसी प्रकार) के एंटिटीज का एक संग्रह है। एक एंटिटी सेट में, एक एंटिटी का एक ही प्रकार की दूसरे एंटिटी से अंतर करने के लिए attributes के मान का उपयोग किया जाता है। प्रत्येक एंटिटी के सेट के लिए, हमे एक key की पहचान करनी चाहिए। एक key न्यूनतम attributes का सेट है जो एक सेट में एक एंटिटीकि विशिष्ट रूप से पहचान करता है। किसी दी गई एंटिटी के सेट में सभी प्रविष्टिया (मान भिन्न हो सकता है) एक ही attributes का सेट रखती है।

एंटिटी के सेट और attributes का E–R आरेख में प्रतिनिधित्व: आयत E–R आरेख में एक एंटिटी के सेट का प्रतिनिधित्व करता है।

E-R आरेख मेंचित्रमय प्रतिनिधित्व: student एंटिटी का उदाहरण



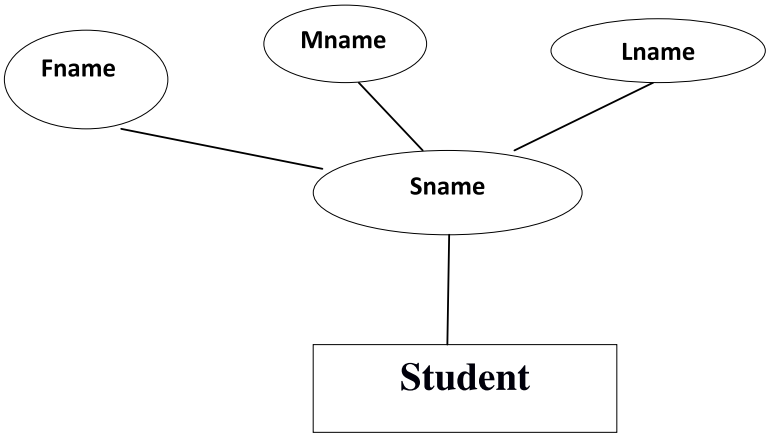
अन्य attribute प्रकार:

Attributesएंटिटीज और relationships के गुण हैंजैसे,Tuples या ऑब्जेक्ट्स के attributes।Attributes निम्न प्रकार के हो सकते हैं और E–R आरेख में अंडाकार या दीर्घवृत्त के रूप में प्रतिनिधित्व करता हैं।

साधारण (Simple) attribute: इस प्रकार के attributes का एक एकल मान होता है जैसे student एंटिटी में Roll_no और age साधारण attributes हैं।

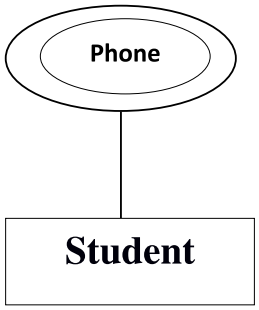
समग्र (Composite) attribute: इस प्रकार के attributes के कई घटक हो सकते हैं जैसे Sname attribute में first name, middle name, last name घटक शामिल हैं।

E–R आरेख में ग्राफिकल प्रस्तुति: student एंटिटी और इसका attribute Sname का उदाहरण।



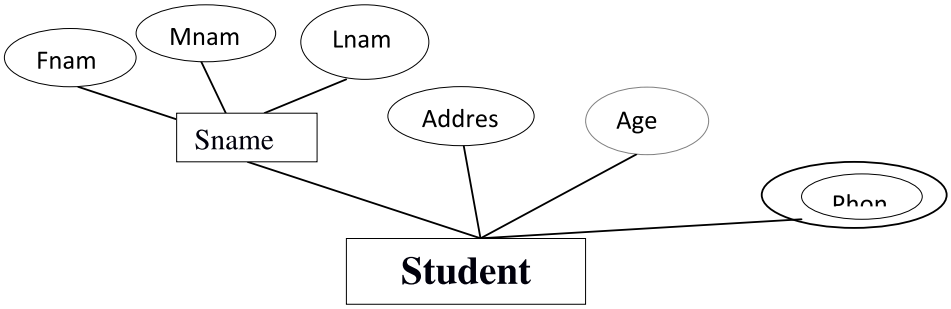
बहु–मान (Multi–valued)attribute: इस तरह की attributes के एक से अधिक मान होते हैं जैसे phone attributes के कई मोबाइल नंबर, लैंड लाइन नंबर, कार्यालय नंबर एक से अधिक मान हैं।

E–R आरेख में ग्राफिकल प्रस्तुति:student एंटिटी और इसका attribute Phone का उदाहरण।



Derived attribute:इस तरह के attributes का मान Multi–valued attributes से अभिकलन किया जा सकता हैं जैसे उम्र, जन्म दिनांक और वर्तमान दिनांक के डेटा से परिकलित कर सकते हैं।

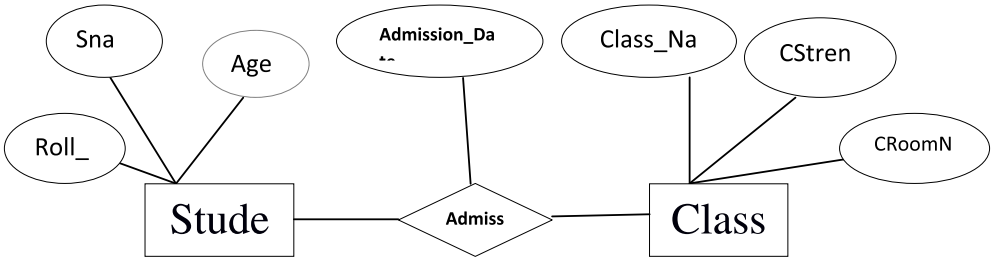
E–R आरेख में ग्राफिकल प्रस्तुति: Composite और Multi–valued attribute के साथ student एंटिटी और इसका attribute age का उदाहरण।



Relationship:एक Relationship दो या दो से अधिक एंटिटीज के बीच एक संबंध है। Relationship जिसमे कि दो एंटिटी सेट शामिल है बायनरी (या दो डिग्री) Relationship कहा जाता हैं ।

Relationship सेट:एक ही प्रकार के Relationship का एक सेट जैसे student का classes में admission। यहाँ admission, student और classes एंटिटी सेट के बीच एक Relationship है। इसकाE–Rआरेख में एक diamondआकार के रूप में प्रतिनिधित्व किया जा सकता है।

E–R आरेख में ग्राफिकल प्रस्तुति:



Attributes of relationships: एक relationship भी (वर्णनात्मक गुण कहा जाता है) अपनी Attributes शामिल कर सकते हैं उदाहरण के लिए मान लें कि student एक विशेष date पर विशेष class में admission लेता है

तो बताये प्रवेश दिनांक(admission date) Attributes कहां शामिल हो? इसकी निम्न दो संभावनाएं है।

प्रथमstudent के साथ?

लेकिन एक student के लिए अलग अलग classes के लिए अलग admission date हो सकते है।

द्वितीयclasses के साथ?

लेकिन एक class एक से अधिक छात्रों के लिए अलग admission date असाइन कर सकते हैं।

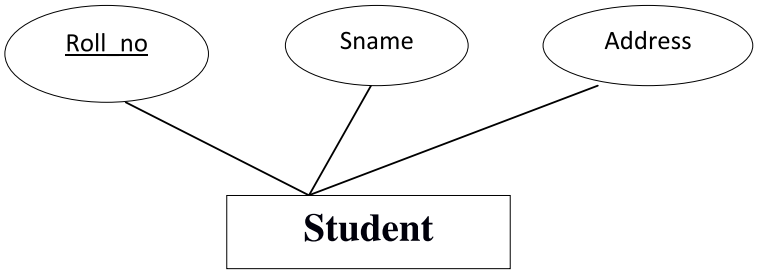
तो ऊपर relationship के E–R आरेख उदाहरण में दिखाया अनुरूप इसे admission के साथ जाना होगा।

बाधायें:

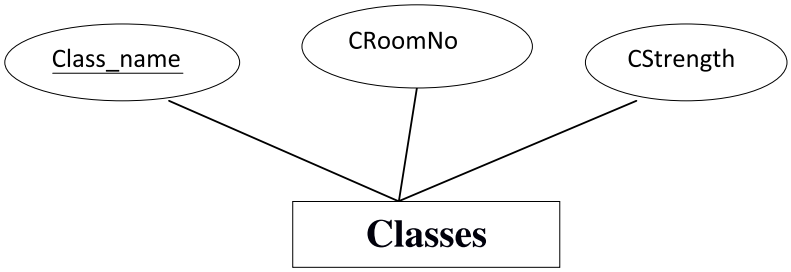
ई–आर मॉडल संग्रहीत डेटा पर और डेटा के ऊपर बाधाओं का भी वर्णन करता है। इन बाधाओं में मैपिंग कार्डिनालिटी, कुंजी(Key) बाधा और भागीदारी बाधा हैं।

Key बाधा या प्रतिबंध :एक key, attributes का एक सेट है जिसका मान ज्यादा से ज्यादा एक एंटिटी सेट के एक एंटिटी से संबंधित हो सकता हैं अर्थात attributes का एक सेट जो किसी एंटिटी कि अद्वितीय रूप से पहचान करे, जैसे उदाहरण के लिए student एंटिटी के सेट कि key, Roll_no हैं। एक एंटिटी सेट कि key को उसकीसभी attributes के रेखांकित द्वारा प्रस्तुत किया जाता है।

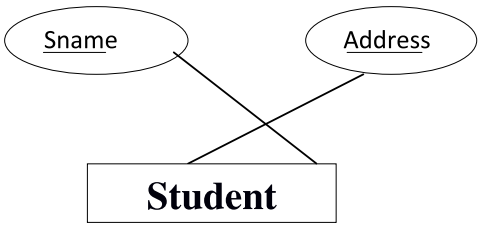
E–R आरेख में ग्राफिकल प्रस्तुति: studentएंटिटीमें,Roll_no attributes का उदाहरण



e.g. 2- classes एंटिटी सेट में primary key, Class_Name है जो अद्वितीय है, क्योंकि हम यह सोचते रहे हैं कि एक स्कूल में बिना किसी भी अनुभाग(section) के 1st से 12 तक 12 अलग अलग classes है।



समग्र कुंजी (Composite Key): दो या अधिक attributes का उपयोग एक key के रूप में कर सकते हैं जैसे Name or Address अकेले छात्र की पहचान नहीं कर सकते हैं लेकिन एक साथ वे एक छात्र की पहचान कर सकते हैं।



Candidate key: Attributes का एक न्यूनतम सेट जोअद्वितीय रूप से एक एंटिटी की पहचान करता है।Candidate key कहा जाता है। e-g.{Roll_no) और {Sname, address} दोनों दो Candidate key है लेकिन {Roll_no, Sname} एक Candidate key नहीं है। अगरकई Candidate keys है, तो हमे एक Candidate key को primary key के रूप में चुनना चाहिए।

मैपिंग कार्डिनालिटी बाधा: बायनरी (या दो डिग्री) Relationship R के लिए मैपिंग कार्डिनालिटी बाधा दो एंटिटी सेट e1 और e2 के बीच निम्न प्रकार कि हो सकती है।

Many to one: e1 में प्रत्येक एंटिटी e2 में 0 या 1 एंटिटी से संबंधित है, लेकिन e2 में प्रत्येक एंटिटी e1 में 0 या और अधिक से संबंधित है।

उदाहरण: एक student एक ही समय में एक class में हो सकते हैं लेकिन एक class में कई छात्र हो सकते हैं।

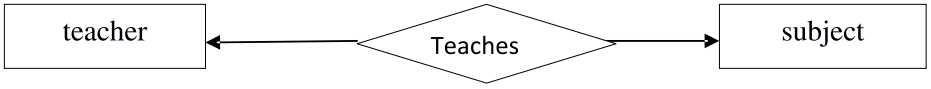
E–R आरेख में ग्राफिकल प्रस्तुति:रेखाओं का उपयोग कर।

Many to many:e1 में प्रत्येक एंटिटी e2 में 0 या अधिक एंटिटी से संबंधित है और इसके विपरीत।

उदाहरण: कई teachers एक class को पढ़ा सकते हैं या कई classes एक ही teacher द्वारा पढ़ा सकते हैं।

One to one: e1 में प्रत्येक एंटिटी e2 में 0 या 1 एंटिटी से संबंधित है और इसके विपरीत।

उदाहरण:एक teacher केवल एक subject जैसे Hindi , English ,Maths सिखा सकते हैं। यह मान है कि स्कूल में प्रत्येक विषय के लिए विशेष शिक्षक निर्धारित है।



One to many: e2 में एक एंटिटी ,e1 में at most one एंटिटी के साथ संबंधित है ।

उदाहरण:विशेष मामले में विषय विशेष शिक्षक स्कूल में उपलब्ध नहीं है, तो एक teacher भी कई विषय सिखा सकते हैं।

भागीदारी (Participation) बाधाओं constraints: एक एंटिटी सेट की सभी एंटिटीज की relationship सेट में भागीदारी नीचे चित्र में दी गई है। एक एंटिटी के लिए एक relationship में भागीदारी बाधाओं के दो प्रकार होते हैं

• **Total:** एंटिटी की प्रत्येक instance relationship में मौजूद है (एक मोटी लाइन द्वारा इसका प्रतिनिधित्व करते हैं)।

उदाहरण: इस उदाहरण में students एंटिटी की भागीदारी totalहै, क्योंकि हर छात्र एक class में प्रवेशजरूर लेता हैं।

E–R आरेख में चित्रमय(graphical) प्रतिनिधित्व(representation): डबल लाइन का उपयोग द्वारा।

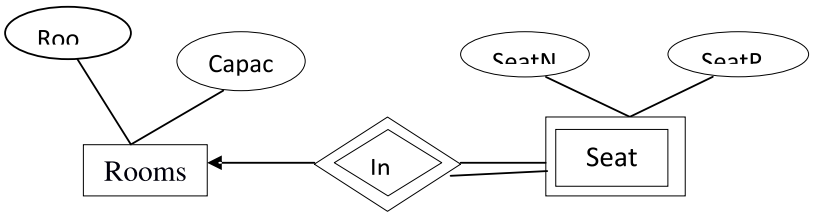
आंशिक(partial): एंटिटी की प्रत्येक instance relationship में मौजूद नहीं है।

उदाहरण: कुछ classes में छात्र प्रवेश नहीं है, तो classes एंटिटी सेट की भागीदारी आंशिक है।

Weak एंटिटीज: कभी–कभी, एक एंटिटी सेट E की key को पूरी तरह से अपनी attributes द्वारा गठित नहीं किया जा सकता, लेकिन अन्य एंटिटी सेट(एक या अधिक) कि keys जिससे E many to many(या one to one) relationship सेट द्वारा लिंक हैके

द्वारा गठित किया जा सकताहै, इसका मतलब है कि एक एंटिटी सेट को अद्वितीय(uniquely) रूप से इस एंटिटी से संबंधित सभी attributes द्वारा नहीं पहचाना जा सकता। ऐसे एंटिटीको weak एंटिटी कहा जाता है।

उदाहरण: मान लीजिए स्कूल के प्रत्येक कमरे में प्रत्येक छात्र के लिए सीट नंबर है। सीटों की विशेषताएँ(attributes) SeatNo और SeatPosition हैं। ये विशेषताएँ एक सीट एंटिटी की विशिष्ट पहचान नहीं कर सकते हैं तो सीट एक कमजोर(weak) एंटिटी है।



weak एंटिटी:को E–R आरेख में डबल डायमंड का उपयोग कर प्रतिनिधित्व किया जा सकताहै। किसी weak एंटिटी के कुछ attributes के किसी अन्य एंटिटी(identifyingया owner एंटिटी) की primary key के साथ संयोजन पर विचार करके weak एंटिटी को पहचाना जा सकता है। संबंधित relationship सेट को identifying relationship कहा जाता है।

एक स्कूल प्रबंधन प्रणाली के लिए डेटाबेस डिजाइन:

सबसे पहले, हम एक स्कूल की निम्नकुछ विशेषतायें (characteristics) मान लेंते हैं।

1. एक स्कूल में कई कक्षाएं 1st से 12th शामिल है।
2. प्रत्येक class में कई विषयों का अध्ययन करते हैं।
3. एक स्कूल में कई शिक्षक काम करते हैं।
4. एक शिक्षक एक ही विषय पढ़ा सकता हैं।
5. प्रत्येक शिक्षक किसी एक ही विषय की कई classes ले सकते हैं।
6. एक छात्र किसी भी कक्षा में प्रवेश ले सकता है।
7. प्रत्येक class में छात्रों की संख्या कुछ भी हो सकती है।
8. प्रत्येक class की अपनी समय सारणी(time table) है।

E–R मॉडलिंग कैसे शुरू कर सकते हैं: एक E–R मॉडल की डिजाइन निम्न के पहचान के द्वारा करसकते हैं।

- एंटिटीज की पहचान
- Relationships की पहचान
- हर एंटिटी के लिए प्रमुख attributes की पहचान
- अन्य प्रासंगिक attributes की पहचान

• Primary key सहित सभी attributes के साथ पूरा E–Rआरेख आरेखित करें

चरण 1: एंटिटीज की पहचान करें:

- •CLASSES
- •STUDENT
- •SUBJECT
- •TEACHER

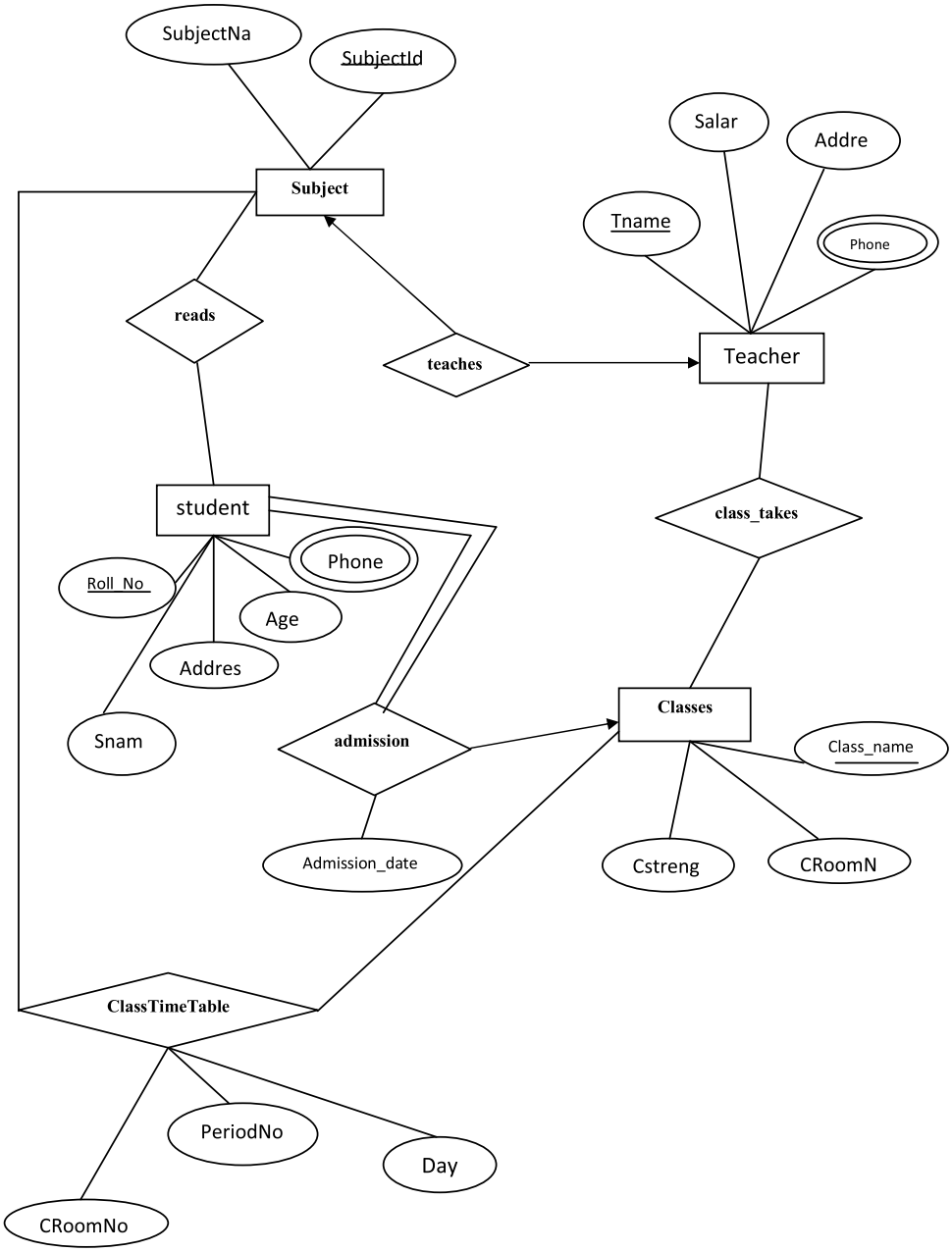
चरण 2:**relationships** की खोज:

- कई classes में कई विषय (subjects) हैं और class time table में कई विषय(subjects) है, इसलिए subjects और classes के बीच कार्डिनालिटी many to many है।
- एक class में कई teacher है, और एक teacher एक विषय के लिए कई classes से संबंधित है, इसलिए classes और teacher के बीच कार्डिनालिटी many to many है।
- एक class में एक से अधिक छात्र हैं और एक छात्र केवल एक ही class में मानते हैं।इसलिए classes और student के बीच कार्डिनालिटी one to Many है।
- एक subject केवल एक शिक्षक द्वारा पढ़ाया जाता है इसलिए subject और teacher के बीच कार्डिनालिटी one to one है।
- एक से अधिक छात्रों कई विषयों को पढ़ता है एक छात्र एक class में कई विषयों को पढ़ता है, इसलिए subjects और student के बीच कार्डिनालिटी many to many है।

चरण 3: प्रमुख **attributes** और अन्य प्रासंगिक **attributes** की पहचान:

- SubjectId key attribute है और SubjectName "subject" एंटिटी के लिए अन्य attributes है।
- Roll_no key attribute और Sname, Age, Address, phone " students एंटिटी के लिए अन्य attributes है
- Tname key attribute औरaddress,phone, salary " Teacher" एंटिटी के लिए अन्य attribute हैं।
- Class_name key attributeऔर CStrength, CRoomNo "classes"एंटिटी के लिए अन्य attributes हैं।
- CRoomNo key attribute और PeriodNo "timeTable"एंटिटी के लिए अन्य attributes है।

स्कूल डेटाबेस का पूरा E–R आरेख:



चित्र 6: 1 से 12 वीं के एक स्कूल का E–R आरेख

E–R का रिलेशनल स्कीमा में अनुवाद: चित्र 6 के E–R आरेख को रिलेशनल डिजाइन में परिवर्तित किया जा सकता है। इस मैपिंग कोकुछ चरणों द्वारा किया जा सकता है।

1. एंटिटी सेट का अनुवाद:एंटिटी सेट के लिए इसकी सभी attributes table के स्तंभ हो जाएगे और सभी key attributes इसकी key स्तंभ हो जाएगे अर्थात,

- Attributes → columns
- Key attributes → key columns

तो स्कूल E–R आरेख की कनवर्ट की गई स्कीमा हैं।
Student(Roll_No , Sname, age, Address, Phone)

Classes(Class_name, Cstrength, CRoomNo)

Teacher(Tname, Salary, Address, Phone)

Subject(SubjectId, SubjecrName)

2. **Relationship** सेट का अनुवाद:एक relationship सेट का भी एक table के रूप में अनुवाद निम्नानुसार किया जा सकता है।

- कनेक्ट किए गए एंटिटी सेट कीसभीkeys,table के स्तंभ हो जाएगे।
- Relationship सेट केattributes (यदि कोई हो),table के स्तंभ हो जाएगे।
- Relationship सेट कि foreign key, भाग लेने वाले निकाय कि primary keyहोगी।
- Relationship सेट की बहुलता table की keyनिम्नानुसार निर्धारित करता है।

(1–to–1 Relationship) के लिए:किसी भी एंटिटी के सेट की primary key किसी Relationship की primary key हो सकती है।

(1–to–many Relationship) या many to 1 Relationship के लिए: relationship set के "many" पक्ष(side) के एंटिटी सेट का primary key, relationship सेट का primary key हो जाएगा।

(Many–to–many Relationship) के लिए:भाग लेने वाले एंटिटी सेट की (primary keys)प्राथमिक कुंजियों का संयोजन, Relationship सेट की primary key हो जाएगा।तो relationship सेट के लिए स्कूल आरेख का कनवर्ट किया गया स्कीमा हैं।

Class_takes(Class_name, Tname)

Admission(Roll_No, Class_name, Admission_date)

Teaches(SubjectId, Tname)

Reads(Roll_No, SubjectId)

ClassTimeTable(Class_name,SubjectId, PeriodNo, CRoomNo, PeriodNo)

विलय के बाद स्कूल डेटाबेस का परिणामी स्कीमा

Student(Roll_No , Sname, age, Address, Phone, Class_name, admission_date)

Classes(Class_name, Cstrength, CRoomNo)

Teacher(Tname, Salary, Address, Phone)

Subject(SubjectId, SubjecrName)

Class_takes(Class_name, Tname)

Teaches(SubjectId, Tname)

Reads(Roll_No, SubjectId)

ClassTimeTable(Class_name, SubjectId, PeriodNo, CRoomNo, PeriodNo)

Normalization का परिचय

एक रिलेशनल डेटाबेस डिजाइन करने के लिए एक और पद्धति है जिसमें हम एक प्रक्रिया का उपयोग करके जिसका नाम normalization है। मुख्य उद्देश्य रिलेशन स्कीमा का एक सेट उत्पन्न करने के लिए है जो हमें बिना redundancy के जानकारी संग्रहीत करने के लिए की अनुमति देता है व साथ साथ में आसानी से जानकारी प्राप्त करने लिए की भी अनुमति देता है। तो सभी डिजाइन करने के लिए, Functional dependencies की अवधारणाओं का उपयोग करके उपयुक्त normal forms में स्कीमा डिजाइन करें।

BAD (खराब) डेटाबेस और normalization का उद्देश्य:

खराब डेटाबेस की अवधारणा और सामान्यीकरण का उद्देश्य को समझने के लिए हम निम्नलिखित student table पर विचार करें।

Student

Roll_no	Name	Age	Address	Phone	class	Subject
101	Harish	10	Ajmer	1234567891	5th	Hindi
101	Harish	10	Ajmer	1234567891	5th	math's

101	Harish	10	Ajmer	1234567891	5th	Sanskrit
120	Ronak	14	Udaipur	22222222222	8th	Hindi
120	Ronak	14	Udaipur	22222222222	8th	Sanskrit

उपरोक्त स्कीमा का विश्लेषण करके हम आसानी से पा सकते हैं कि यह डिजाइन एक अच्छा डेटाबेस डिजाइन नहीं है।

यह डिजाइन क्यों बुरा है?

के लिए studentsubjects(Roll_no, name, age, address, class, subject)पर विचार करें।

यह डिजाइन redundant है क्योंकि एक छात्र का name, age, address, class कई बार, दर्ज की गई है ने प्रत्येक विषयों के लिए जिसमें छात्र पढ़ रहा है।यह redundancy डिजाइन में गंभीर समस्याओं का कारण है।e.g कि यह भंडारण स्थानwaste करता है और डेटाबेस में संभावित inconsistency शुरू करता है। इसीलिए यह डेटाबेस डिजाइन खराब है।एक खराब डेटाबेस डिजाइन डेटाबेस पर कार्रवाई के दौरान भी कई problems(Anomalies) का कारण है ।

Update विसंगतियाँ :यदि एक प्रतिलिपि Updateहोती है तब सब दोहराये डेटा की भी Updateकी जरूरत होती है। उदाहरण के लिए हम एक विशेष छात्र का पता Update करने के लिए हम उस छात्र को सभी tuples Update करते है।

Insertion विसंगतियाँ: जब तक असंबंधित जानकारी संग्रहीत है तब तक कुछ डेटा संग्रहीत नहीं हो सकता है। किसी टपल को (Insert)सम्मिलित करने के लिए फोन पता करने की आवश्यकता है. यह एक रिक्त (null) मान के साथ तय हो सकता है लेकिन null समस्याओं के कारण है या हैंडल करने में मुश्किल है।

deletion विसंगतियाँ : कुछ अन्य, असंबंधित जानकारी खोने के बिना कुछ जानकारी को हटाने संभव नहीं हो सकता है।अगर हम सभी tuples एक दिए गए (class, Roll_no) के लिए हटाएँ तो हम उस एसोसिएशन को खो सकते है।तो यदि हम डिजाइनों में redundancy को निकालना चाहते हैं, तो हमें एक व्यवस्थित दृष्टिकोण की जरूरत होती है। यदि हम एक अच्छा डेटाबेस को डिजाइन करना चाहते हैं, तो हम Dependencies, decompositions और normal forms के उपयोग करते है।

Functional dependencies:एक कार्यात्मक निर्भरता(Functional dependencies या FD) IC का एक प्रकार है जो key की अवधारणा को generalizes करता है। चलो R एक रिलेशन स्कीमा है, X और Y रिलेशन R के nonempty

attributesका सेट है तो R के instancer के लिए हम कहते हैं कि FD (X कार्यात्मक निर्धारित करता है Y) संतुष्ट है: अगर

∀t1, t2 ∈ r, t1.X = t2.X =) t1.Y = t2.Y

X → Y इसका मतलब है कि जब भी R में दो tuples X में सभी attributes पर सहमत हैं, तो वे Y में भी सभी attributes पर सहमत होना होगा।

एक कार्यात्मक निर्भरता का उदाहरण:

कार्यात्मक निर्भरता AB → C के लिए निम्न instance संतुष्ट हैं–

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

एक primary key एक FD की विशेष स्थिति है: अगर $X \rightarrow Y$ रखती है (जहाँ Y सभी विशेषताओं(attributes) का सेट है, तो X एक superkey है।

• FDs रिलेशन की किसी भी instance के लिए रखना चाहिए।यदि FDs का एक सेट दिया है तो हम आम तौर पर अतिरिक्त FDs भी पा सकते हैं।

उदाहरण :यदि एक key दी है, तो हम हमेशा एक superkey पा सकते हैं.

FDs के उपयोग द्वारा keys को पुनर्परिभाषित करने:

K, attributes का एक सेट है जो रिलेशन R के लिए एक key है। यदि

K → (अन्य) सभी attributes R के, अर्थात K एक " superkey key है।

उपरोक्त शर्त को K का कोई उचित सबसेट संतुष्ट नहीं करता है, अर्थात K कम से कम है।

Normalization

Normalization डेटाबेस में डेटा के आयोजन की एक प्रक्रिया है जोडेटारिडंडेंसी insertion विसंगति(anomaly), update विसंगति एवं deletion विसंगति को दूरकरने के लिए काम करता है। इस प्रक्रिया में हम एक दिए गए रिलेशन स्कीमा की जाँच कुछ Normal forms के विरुद्ध करता है यह पता करने करें की यह किसी normal form को संतुष्ट करता है या नहीं। यदि एक रिलेशन स्कीमा किसी normal form को संतुष्ट नहीं करता है, तो फिर हम इसे यह छोटे स्कीमा में विघटित करता है।

Normalization मुख्य रूप से दो उद्देश्य के लिए उपयोग किया जाता है,

- अनावश्यक (अनुपयोगी) डेटा को नष्ट करने के लिए
- यह सुनिश्चित करने के लिए की डेटा dependencies का मतलब हैं अर्थात डेटा संग्रहीत तार्किक है ।

रिलेशनल डेटाबेस डिजाइन: एक रिलेशनल डेटाबेस डिजाइन करने के लिए ,एक दिए गए स्कीमा के लिए यह पता करने की जरूरत है कि यह एक अच्छा डिजाइन है । डिजाइन अच्छा नहीं है, तो फिर हम इसे छोटे स्कीमा में विघटित करतेहै लेकिन यह अपघटन(decompose) अच्छा होना चाहिए। उसके बाद हम कुछ Normal forms के लिए प्रत्येक विघटित स्कीमा की जाँच करें। एक दिए गए रिलेशन स्कीमा एक Normal form में है, तो हम जानते हैं कि यह कुछ समस्याएं उत्पन्न नहीं कर सकता.

Normal forms:विभिन्न Normal forms हैं ।

1. First Normal Form(1NF)
2. Second Normal Form(2NF)
3. Third Normal Form(3NF)
4. BCNF

First Normal Form(1NF):

यदि प्रत्येक फील्ड केवल Atomic मान रखता है(कोई सूचियाँ नहीं और न ही सेट) तो रिलेशन First Normal Form(1NF)में है ।

उदाहरण:

चित्र 7 में नीचे दी गई Student table 1NF में नहीं है, लेकिन चित्र 8 में 1NF है ।

Student

Roll_no	Name	Age	Subject
101	Harish	10	hindi, maths
120	Ronak	14	maths

चित्र 7Student table showing subjects of students not in 1NF

Student

Roll_no	Name	Age	Subject
101	Harish	10	Hindi
101	Harish	10	Maths
101	Harish	10	Sanskrit

120	Ronak	14	Hindi
120	Ronak	14	Maths

चित्र 8: **Student table showing subject of student in 1NF**

Second normal form:

Second normal form के अनुसार किसी भी स्तंभ कि primary key पर partial dependency नहीं होना चाहिए। इसका मतलब है कि किसी table के लिए जिसकी primary key है , table का हर non prime attribute, primary key attribute पर पूरी तरह functionally dependent होना चाहिए । यदि कोई भी स्तंभ केवल primary key के एक भाग पर निर्भर करता है, तो table Second normal form में विफल है।

StudentReadsSubject (Roll_no, subjected, Sname, address, SubjectName)

इस student और subject रिलेशन में primary key attribute Roll_noऔर subjecteId है। नियम के अनुसार, non key attributes Sname और SubjectNameदोनों पर निर्भर होना चाहिए ताकि अलग–अलग दोनों prime key attributes पर लेकिन हम पाते हैं कि Sname को Roll_no द्वारा और SubjectName को SubjectId द्वारा स्वतंत्र रूप से पहचाना जा सकता है। यह partial dependency कहा जाता है, जिसकी Second normal form में अनुमति नहीं है।

हम रिलेशन को दो रिलेशन में तोड़ दिया है । ताकि वहाँ कोई partial dependency मौजूद नहीं है।

Student(Roll_no, Sname, address)

Subject(SubjectId, SubjectName)

Third Normal Form(3NF)

कोई रिलेशन स्कीमा 3NFमें है या नहीं यह चैक करने के लिए हम उस स्कीमा की प्रत्येक FD के लिए निम्न शर्तों को चैक करते है। अगर किसी FD के लिए निम्न शर्तें फेल होती होती है तो वह स्कीमा 3NF में नहीं होगा

शर्तें:

- अगर कोई FD ट्राइवल (trivial) है अर्थात $\beta \rightarrow A$में $(A \in \beta)$, है तो या
- अगर किसी FD के बाँयी तरफ के ऐट्रीब्यूटस् उस स्कीमा की Key है या
- अगर किसी FD के दाँयी तरफ के ऐट्रीब्यूटस् रिलेशन स्कीमा की Key का पार्ट है

BCNF

कोई रिलेशन स्कीमा 3NFमें है या नहीं यह चैक करने के लिए हम उस स्कीमा की प्रत्येक FD के लिए निम्न शर्तों को चैक करते है। अगर किसी FD के लिए निम्न शर्तें फेल होती होती है तो वह स्कीमा 3NF में नहीं होगा

शर्तें:

- अगर कोई FD ट्राइवल (trivial) है अर्थात $\beta \rightarrow A$में $(A \in \beta)$, है तो या
- अगर किसी FD के बाँयी तरफ के ऐट्रीब्यूटस् उस स्कीमा की Key है या

यदि कोई रिलेशन R , BCNF में है तो फिर यह 3NF में भी हो जाएगा। तात्पर्य BCNF implies 3NF but 3NF can not implies BCNF.

डिजाइन लक्ष्य

एक अच्छा रिलेशनल डेटाबेस डिजाइन के लिए एक रिलेशनल स्कीमा Lossless join और dependency preservation के साथ BCNF में होना चाहिए। यदि हम इस लक्ष्य को हासिल नहीं कर सकते हों तो , हम Lossless join और dependency preservation 3NF स्कीमा के साथ स्वीकार करते हैं।

महत्वपूर्ण बिंदु

- DBMS का उपयोग बहुत बडे डाटा सेट को मेन्टेन रखने एवंम् उससे डाटा प्राप्त करने के लिए होता हैं।
- DBMS के कई लाभ है।
- एब्सट्रेक्शन लेवल डाटा इनडिपेन्डेन्स देते है।
- इसका आर्किटेक्चर लेयरड होता है।
- DBMS के कार्यात्मक(functional) घटक(components) , फाइल प्रबंधक बफर प्रबंधक,क्वेरी प्रोसेसर,डेटा फाइल,डेटा dictionary और Indices है।
- एक DBMS सिस्टम उपयोगकर्ताओं पर आधारित, Architecture पर आधारित और डेटा मॉडल के प्रकार पर आधारित कई प्रकार के हो सकते है।
- एंटिटी: असली दुनिया में एक "ऑब्जेक्ट" है जिसकी अन्य सभी वस्तुओं से अलग पहचान है।
- Candidate key: attributes का एक न्यूनतम सेट जोअद्वितीय रूप से एक एंटिटी की पहचान करता है Candidate key कहा जाता है।

- एक खराब डेटाबेस डिजाइन डेटाबेस पर कार्रवाई के दौरान भी कई problemsे(Anomalies) का कारण है जैसे कि Update विसंगतियाँ ,Insertion विसंगतियाँ और deletion विसंगतियाँ।
- एक कार्यात्मक निर्भरता(Functional dependencies या FD) IC का एक प्रकार है जो key की अवधारणा को generalizes करता है।
- यदि कोई रिलेशन R , BCNF में है तो फिर यह 3NF में भी हो जाएगा। तात्पर्य BCNF implies 3NF but 3NF can not implies BCNF.

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्नः

प्रश्न 1. निम्न में से कौन सा एक DBMS का एक लक्ष्य नहीं है।

(अ) बड़ी जानकारी का प्रबंध (ब) कुशल पुनः प्राप्ति

(स) Preventing concurrent access (द) डेटा की सुरक्षा

प्रश्न 2. निम्न में से कौन सा एक वाणिज्यिक DBMS का एक उदाहरण है।

(अ) Oracle (ब) IBM (स) Sybase (द) all

प्रश्न 3. निम्न में से कौन सा एक डेटा abstraction.के सरलतम स्तर है।

(अ) भौतिक (ब) तार्किक

(स) View (द) इनमें से कोई भी देखें

प्रश्न 4. सामान्यीकरण का मतलब है

(अ) Joining relations(ब) अपघटन के सम्बंध में शामिल

(स) दोनों (द) इनमें से कोई नहीं

प्रश्न 5. जो normal form अधिक प्रतिबंधित है।

(अ) 1NF (ब) 2NF (स) BCNF (द) 3NF

अतिलघुत्रात्मक प्रश्नः

प्रश्न 1. DBMS क्याहै?

प्रश्न 2. एक रिकॉर्ड को परिभाषित करो।

प्रश्न 3.भिन्न डेटा रिडंडेंसी के नाम दे।

प्रश्न 4. डेटाबेस स्कीमा को परिभाषित करो।

प्रश्न 5.DBMS में इंडेक्सेस की भूमिका क्या है?

प्रश्न 6.एक क्वेरी भाषा क्या है?

प्रश्न 7. procedural और non -procedural DML बीच क्या अंतर है।?

प्रश्न 8. स्कीमा और instances के बीच क्या अंतर है?

प्रश्न 9. एक डेटाबेस डिजाइन के क्या चरण हैं?

प्रश्न 10. एक एंटिटी क्या है?

लघुत्रात्मक प्रश्नः

प्रश्न 1. Atomicity से आप क्या समझते हो?

प्रश्न 2. तार्किक और भौतिक डाटा independenceके बीच क्या अंतर है?

प्रश्न 3. एक weakएंटिटी क्या है? यह E-R आरेख में आकर्षित करो।

प्रश्न 4. प्राथमिक और समग्र कुंजी के बीच क्या अंतर है?

प्रश्न 5. एक खराब डेटाबेस क्या है।

निबंधात्मक प्रश्नः

प्रश्न 1. DBMS के विभिन्न घटक क्या हैं? उपयुक्त चित्र के साथ समझाओ।

प्रश्न 2. एक डेटा मॉडल क्या है.? श्रेणीबद्ध डेटा मॉडल समझाये? यह नेटवर्क डेटा मॉडल से कैसे अलग है।

प्रश्न 3. E-R आरेख में विभिन्न प्रकार के attributes और relationships की व्याख्या को समझाए और उनके लिए ग्राफिकल प्रतिनिधित्व भी दे।

प्रश्न 4. एक स्कूल अलग अलग classes 1^{st} to 10^{th} से मिलकर बनी है के लिए एक E-R आरेख डिजाइन करो।

प्रश्न 1. normalization क्या है? normalization के विभिन्न रूपों को बताए।

उत्तरमाला

उत्तर 1: स **उत्तर** 2: द **उत्तर** 3: स

उत्तर 4: ब **उत्तर** 5: स

अध्याय – 14

रिलेशनल डाटाबेस की अवधारणायें

रिलेशनल डाटाबेस में बहुत सी टेबलस होती है। हर टेबल का एक यूनिक नाम होता है। व उसमें बहुत से कॉलम (स्तंभ) होते है। हर स्तम्भ का भी एक यूनिक नाम होता है। रिलेशनल डाटाबेस का नाम मेथिमेटिक्स रिलेशन से निकाला गया हो क्योकि दोनों में निकट व्यवहारता है।

टेबल (रिलेशन) :– आर डी बी एम एस में डाटा एक प्रकार के डाटा बेस आबजेक्ट में स्टोर होता है। जिसे हम टेबल कहते है। दूसरे शब्दों में टेबल सम्बधित डाटा एन्ट्रीरीज का संग्रहण हो जिसमें की पक्तियाँ एवम् स्तंभ होते है। निम्नलिखित एक स्टूडेन्ट टेबिल का उदाहरण है।

Student Table

Roll_no	Name	Age	Address	class
101	Harish	10	Ajmer	5th
105	Kailash	20	kota	10th
109	Manish	18	Ahmadabad	9th
120	Ronak	14	Udaipur	8th
135	Shanker	13	Jaipur	7th

चित्र–1 स्टूडेन्ट टेबल रिलेशन

फिल्ड (**Field**) :– किसी टेबिल का Field उसका एक स्तंभ होता है। जो कि उस टेबल में किसी रिकार्ड की specific इर्फोमेशन को रखता है। जैसे की ऊपर दी हुईstudent टेबल मेंRoll_no,name, address और class फिल्ड्स है।

रिकार्ड :– को हम टेबिल की एक पक्ति भी कहते है। तथा यह एक टेबिल की वह individual entry है जो उस टेबिल में है। जैसे कि

Student table

105	**kailash**	**20**	**kota**	**10th**

स्तंभ :–किसी एक टेबिल की वह वर्टिकल एन्ट्री है जो किसी विशिष्ट फिल्ड से सम्बनधित सभी इन्फोमेशन रखता है। जो कि student टेबिल का एक स्तंभ Roll_no है। जो कि निम्न इन्फोमेशन रखता है।

Roll_no
101
105
109
120
135

डोमेन :– किसी फिल्ड की परमिटेड वैल्यू सैट को उसका डोमेन कहते है। उदाहरण स्वरूप field name के लिए डोमेन सभी नामों का सैट है।

डाटा बेस स्कीमा (Database schema) :–डाटा बेस स्कीमा किसी डाटा बेस की लॉजिकल डिजाईन है। जो कि शायद ही बदलती है। जैसे कि student टेबिल का schema है।

Student (Roll_no, name, age, address, class)

डाटा बेस इन्सटेन्स (Database instance) :–किसी डाटा बेस में समय के किसी भी क्षण डाटा के समूह को डाटा बेस इन्सटेन्स कहते है। उदाहरणार्थ निम्नलिखितStudent टेबिलStudent डाटाबेस का एक इन्सटेन्स है। जब हम टेबिल में कोई नई एन्ट्री करे या कुछ टेबिल से delete करेंतो यह किसी भी क्षण बदल सकता है।

Student

RollNo	Name	Age	Address	Class
101	Harish	10	Ajmer	5^{th}
105	Kailash	20	Kota	10^{th}
109	Manish	18	Ahmadabad	9^{th}
120	Ronak	14	Udaipur	8^{th}
135	Shanker	13	Jaipur	7th

प्राइमेरी की :–किसी टेबिल में एक या अधिक फिल्डस (attribute) का ऐसा set जो कि उस टेबिल की किसी भी पर्क्ति अथवा टपल्स को uniquely identify करता हो तो इस attributes के सैट को collectively लेने पर यह उस टेबिल की Primary key कहलाती है। जो एक प्रकार का constraints भी है। student टेबिल की primary key ,Roll_no फिल्ड है क्योकि student टेबिल में इसे फिल्ड के द्वारा सभी छात्रों को uniquely identify किया जा सकता है। एवंम् Roll_no फिल्ड की सहायता से किसी छात्र का रिकार्ड टेबिल से निकाला जा सकता है। जैसे कि अगर Roll_no फिल्ड की वैल्यू 105 लेने पर जो रिकार्ड टेबिल से निकलेगा वह छात्र kailash का होगा।

डाटा कन्सट्रेन्टस(Data constraints) :– किसी टेबिल के स्तंभों पे इस तरह के नियम लागू करना है जो उस टेबिल में डाटा की एन्ट्री की सीमा को निर्धारित करता है की उस टेबिल में केवल उसी प्रकार को डाटा एन्टर हो जो उस डाटाबेस कीconsistency, reliabilityएवंम् accuracy को सुनिश्चित कर सके। एवंम् डाटाबेस में किसी प्रकार का बदलाव जब अधिकृत डाटा बेस यूर्जस करे तब भी डाटाबेस में किसी प्रकार का डाटाconsistency loss ना हो।

डाटा कन्सट्रेन्टस कॉलम (स्तंभ) लेवल ओर टेबिल लेवल हो सकते है। कॉलम लेवल एवंम टेबिल लेवल कन्सट्रेन्टस में मुख्य अन्तर यह है कि कॉलम लेवल कन्सट्रेन्टस एक कॉलम में लगाये जाते है। जबकि टेबिल लेवल कन्सट्रेन्टस पूर्ण टेबिल में लगाये जाते है। डाटा कन्सट्रेन्टस के निम्न उदाहरण है जैसे कि

(1) studentकीClass nullनही हो सकती है।

(2) किन्ही दो छात्रों केRoll_noएक समान नही होगें।

(3) studentरिलेशन की हर एक Classरिलेशन में एकmatching class जरूर रहेगी ।

एक रिलेशन वाले कन्सट्रेन्टस

निम्नलिखित कन्सट्रेन्टस एक रिलेशन वाले कन्सट्रेन्टस है।

1) Not null

2) Unique

3) Check (<predicate>)

1) Not null कन्सट्रेन्टस :– यह कन्सट्रेन्टस किसी भी टेबिल में किसी फिल्ड या एट्रीब्यूट कीnullवेल्यू की एन्ट्री कोप्रतिबंधित करता है। अर्थात यदि किसी फिल्ड के लिए अगर यह कन्सट्रेन्टस लगा हुआ है। और उस टेबिल में कोई आपरेशनजो उस टेबिल में बदलाव कर अगर वेल्यूnullडालने की कोशिश करता है तो वहॉ पेerror जनरेट हो जाती है।

उदाहरण के तौर पर हमstudentटेबिल मेंclass एट्रीब्यूट की वेल्यू nullनहीं चाहते है इसी प्रकार Roll_noएट्रीब्यूटकी वेल्यू भीnullनही होनी चाहिए क्योकि वह उस टेबिल की एक प्राइमेरी की है।

2) Unique कन्सट्रेन्टस :–यह सुनिश्चित करता है कि कोई से दो टपल्स या पक्तियों किसी रिलेशन में सभी प्राइमेरी की एट्रीब्यूट पर बराबर नही हो सकती । अर्थात दोनों टपल्स के लिए हर एट्रीब्यूट की वेल्यू एक समान नही होगी।Unique कन्सट्रेन्टस केन्डीडेड की फोर्म करता है। जो कि किसी रिलेशन में एक से ज्यादा भी हो सकती है

3) Check कन्सट्रेन्टस :–Check कन्सट्रेन्टस को हम डोमेन एवंम् रिलेशनल दोनों डिक्लेरेशन पर लगा सकते है। जब किसी रिलेशन डिक्लेरेशन पर लगा हो तो सभी

टपल्सCheck क्लाज द्वारा specified condition को पूरा करेगा अर्थात्Check क्लाज यह सुनिश्चत करता है कि किसी स्तंभ की सभी वेल्यूज उस पर लगी शर्त को पूरा करेगा ।

डदाहरण स्वरूपstudent टेबिल में classफिल्ड की वेल्यू1^{st},2^{nd},3^{rd},---- ,9^{th},10^{th},11^{th},12^{th}होगी।

check (class in('1^{st}', '2^{nd}','3^{rd}', ----- '9^{th}','10^{th}','11^{th}','12^{th})

एंटिटी इन्टीग्रीटी कन्सट्रेन्टसः–यह सूनिश्चित करता है कि किसी भी टेबिल में कोई दो रिकार्डस या पर्क्तियों या टपल्स डूप्लीकेट नही हो सकते है। इसके अलावा वह फिल्ड जो प्रत्येक रिकार्ड की उस टेबिल में पहचान कर रहा है वह एक यूनिक फिल्ड है तथा इस फिल्ड की वेल्यू कभी भीnullनही होगी ।

एंटिटी इन्टीग्रीटी कन्सट्रेन्टस प्राइमेरी की के द्वारा लगाया जा सकता है। हर एंटिटी के लिए अगर हम प्राइमेरी की को परिभाषित करते है तो वह स्वतः ही एंटिटी इन्टीग्रीटी की पूर्ति करता है। उदाहरणार्थ

Student table

RollNo	Name	Address	Age	Class
110	Komal	jaipur	17	12th
120	Ronak	Udaipur	14	8th
105	kailash	kota	20	10th
107	hari	chittorgarh	10	5h

उक्तStudentटेबिल की प्राइमेरी की अगरRoll_noफिल्डस है तो इस फिल्ड में प्रत्येक छात्र काRoll_noअलग–अलग होगा। साथ ही किसी छात्र के लिए उसकी वेल्यू nullनही होगी अर्थात् सभी का अपना अलगRoll_noहोगाअलग –अलगRoll_noकी वजह से इस टेबिल में कोई दो पक्तियॉ एक समान नहीं होगी

रेफरेनशीयल इन्टीग्रीटी(Referential integrity) :–अगर हम किसी प्रकार से यह सुनिश्चित करना चाहते है कि किसी रिलेशन में कुछ ऐट्रीब्यूट के लिए उनकी वेल्यू वही हो जो किसी अन्य रिलेशन में कुछ ऐट्रीब्यूट के लिए है। अर्थात दोनों टेबिल में कुछ ऐट्रीब्यूट वेल्यू एक समान हो तो यह शर्त रेफरेनशीयल इन्टीग्रीटी कहलाती है। रेफरेनशीयल इन्टीग्रीटी की सुनिश्चितता फोरेन की के द्वारा की जा सकती है।

फोरेन की(Foreign key)इन्टीग्रीटी कन्सट्रेन्टसः– इस कन्सट्रेन्टस को समझने लिए निम्न प्रदर्शित टेबिल को उदाहरण के लिए हम लेते है। यहॉ दो टेबिल studentएवंमClassके नाम से है। एवंम् किसी क्षण उनमे एन्टर वेल्यूस भी प्रदर्शित है।

Student

RollNo	Name	Age	Address	Class
101	Harish	10	Ajmer	5^{th}
105	kailash	20	Kota	10^{th}
109	Manish	18	Ahmadabad	9^{th}
120	ronak	14	Udaipur	8^{th}
135	shanker	13	jaipur	7th

Classes

Class_name	Class_room	Strength
12^{th}	F-1	95
10^{th}	F-2	80
9^{th}	F-3	70
4th	F-4	110

चित्र –2 फोरेन की स्टूडेन्ट रिलेशन टेबिल

उक्त टेबिल Studentमें इस टेबिल कीप्राइमेरी की Roll_noफिल्ड है। जबकि Classesटेबिल की प्राइमेरी कीClass_name फिल्ड है। यहॉ पर हमने यह माना है कि सभी स्टूडेन्ड की एक हीclassविद्यमान है। जैसे की12^{th}की एकclass10^{th}की एकclass 9^{th} की एकclassइसी प्रकार अन्य ,अर्थात एक हीclassके sectionsअलग –अलग नही है। इसलिएclassesटेबिल की प्राइमेरी कीClass_name है।

Studentटेबिल अपने ऐट्रीब्यूट(Roll_no, address, age, name, class) के बीच में एक ऐसा ऐट्रीब्यूट भी रखता हैजो किसी अन्य टेबिल की प्राइमेरीkeyहै।उदाहरण स्वरूपStudentटेबिल मेंClassऐट्रीब्यूट टेबिल की प्राइमेरी की है। अतः Studentटेबिल में इस ऐट्रीब्यूटclassको हम इस टेबिल कीprimary keyकहेगे। जो कि टेबिल Classesको रेफर करेगी।

रिलेशन Studentटेबिल को हम रेफरेनशिंग रिलेशन कहेगें जब कि Classesको रेफरेनशड रिलेशन आफ फोरेन keyकहेगे।किसी ऐट्रीब्यूट को primary key होने के लिए उसका टाइप domain वही होना चाहिऐ जो दूसरे रिलेशन केऐट्रीब्यूट का है तथाforeign key में ऐट्रीब्यूटस की संख्या भी दूसरे रिलेशन के ऐट्रीब्यूटस के बराबर होनी चाहिए अर्थात वेकाम्पीटेबल होने चाहिए।

Foreign key के द्वारा हम निम्नलिखित सुनिश्चित कर सकते है–

(1) Classesटेबिल से हम कोई रिकार्ड तब तक नही हटा(delete)सकते जब तक इसकी रिलेटेड टेबिलStudent मेंमेचिंग रिकार्ड उपलब्ध है।

(2) आप Classes टेबिल में प्राइमेरीkeyकी वेल्यू को तब तक नही बदल सकते जब तक उस रिकार्ड से रिलेटेड रिकार्ड दूसरी टेबिल में उपलब्ध है।

(3) हम Studentटेबिल के Classफिल्ड में कोई नई वेल्यू जब तक नही डाल सकते जब तक वह वेल्यू Classesटेबिल केप्राइमेरी की फिल्ड(class_name) में उपलब्ध नही है।

(4) लेकिन आप foreign key field मेnull वेल्यू डाल सकते है। उक्त आपरेशनस के दौरान अगर हम चाहते है कि यह रिजेक्ट ना हो तो हमें classesका प्रयोगSQLमें करते है।

SQL का परिचयः–

स्ट्रकचर क्यूरी लेग्वेजSQLरिलेशनल ऐलजेबरा एवंम् रिलेशनल केलकूलस के कोमबीनेशन का उपयोग करती है।SQL आर डी बी एम एस की एक स्टेन्डर लेंग्वेज है जो किसी रिलेशनल डाटा बेस को ऑर्गेनाइज करने उसमें उपस्थित डाटा को प्रबंधन करने एवंम् रिलेशनल डाटा बेस में से डाटा रिट्रीवल के लिए उपयोग में लाई जाती है।

डी बी एम एस के विक्रेताओं जैसे कि ऑरेकलIBM, DB2, Sybase and Ingressअपने डाटाबेस के लिएSQLका एक प्रोग्रामिंग लेंग्वेज के तौर पर उपयोग करते है।इसका मूल संस्करण sequelकहलाता था जिसकोIBMने विकसित किया था।

इसके कई संस्करण होते है जैसेSQL-86, SQL89(extended standard), SQL92 and SQL1999 औरवर्तमानसंस्करणSQL-2003, SQLकेवल मात्र एक डाटाबेसqueryलेंग्वेज नहीं है ब्लकि अपने आप में एक स्टेन्र्ड है। जिसकेनिम्नलिखित भाग है।

(1) डेटा डेफीनेशन लेग्वेंज(Data Definition Language)

(2) डेटा मेनीपूलेशन लेग्वेंज(Data Manipulation Language)

(3) डेटा कन्ट्रोल लेग्वेज (Data Control Language)

ज्यादातर कामर्शियल रिलेशनल डाटा बेस जैसेIBM, Oracle, Microsoft, Sybase आदिSQLका उपयोगकरते है। सभी डाटाबेस SQLके सभी संस्करण में उपस्थित फिचर्स को सपोर्ट नहीं करते है अतः हमेशा अपने डाटाबेससिस्टम के अनुसार उसके संस्करण को देखते हुये ही फिचर्स का उपयोग करना चाहिए।

SQLके लाभः–SQLसभी डाटाबेस सिस्टम चाहे कामर्शियल(Oracle, IBM, DB2, Sybase) हो याऑपन सॉर्स(MySQL ,Postgres) के लिए लाभदायक है। इसके लाभ निम्नलिखित है।

(1) उच्च गति(**High speed**) :–इस प्रकार की लेंग्वेज है जो बडे से बडे ऑर्गेनाइजेशन के डाटाबेस से बहुत हीकुशलता(efficiently)के साथ तथा जल्दी –जल्दी डाटा निकाल सकती है अतः SQLएक उच्च गति की लेंग्वेज हैं।

(2) सीखने की सुविधा(**Easy to learn**):–SQLलेंग्वेज को सीखना बहुत ही आसान है। क्योंकि इसमें प्रोग्राम का किसी प्रकार कालंबा कोड नहीं होता हैं। अर्थात् इसमें ज्यादा कोडिंग की आवश्यकता नहीं होती हैं।

(3) अच्छी तरह से परिभाषित मानक लेंग्वेज(**Well defined standard**) :–SQL एक मानक(standard)भाषा हैजिसकों स्टेन्डराइजANSI & ISOने किया है।

डाटा डेफीनेशन लेंग्वेज(**Data Definition Language**) :–**DDL**,SQLका एक भाग हैं। जिसकी सहायता से हमडाटा बेस स्कीमा को स्पेसीफाई कर सकते है।

DDLकेवल डाटा बेस स्कीमा(schema)को हीspecifiedनहीं करती बल्कि हर रिलेशन के बारे में भी स्पेसीफीकेशनरखती है। कुछ निम्न है।

(1) हर रिलेशन का स्कीमा के लिए

(2) हर ऐट्रीब्यूट की वेल्यूज का डोमेन

(3) कन्सट्रेन्ट्र के लिए

(4) इनडेक्स के लिए

(5) किसी स्कीमा के ऑर्थोराइजेशन एंवम् सिक्योकिटी के लिए

(6) हर रिलेशन के फिजीकल स्टोरेज के लिए

SQL के बेसिक डोमेन टाइप्सः–

मुख्य रूप से उपयोग करने के लिएSQL के सभी विल्ट इन डोमेन टाइप्स निम्न होते है।

(1) **न्यूमेरिक डेटा टाइप्सः**– इसमें निम्न डेटा टाइप्स शामिल है।

(i)IntयाInteger:–बडें साईज के इन्टीजर्स के लिए है। इसके लिए 4 बाईट का स्टोरेज आवश्यक है।

(ii)Small Int:–छोटे साईज के इन्टीजर्स के लिए है। इसके लिए आवश्यक स्टोरेज बाईट 2 है।

(iii)Tiny Int:–अत्यधिक छोटे साईज के साईज या अनसाइन इन्टीजर्स के लिए

(iv) Float(M.D) :–फलोटिगं पाइन्ट नम्बरर्स के लिए । यह केवल साइन नम्बर्स के लिए है यहॉ पर Mउस नम्बर मेंकुल डिजिटस है तथाDडेसिबल नम्बर की संख्या है।

(v) Double (M,D) :–यह डेटा टाइप्स डबल प्रिसीजन वाले फलोटिंग पॉइन्ट नम्बर के लिए है तथा यह REALका समानार्थक शब्दहै।

(2) **स्ट्रींग टाइप्स (String Type)**:–MySQLमें निम्न स्ट्रींग डेटा टाइप्स है

(i)CHAR(C) :– यह डेटा टाइप फिक्सड लम्बाई की स्ट्रींग के लिए है। यहॉ परCइस स्ट्रींग की लम्बाई है जो यूजरदेता है। अगर इस लम्बाई से कम लम्बाई की कोई स्ट्रींग अगर स्टोर करते हैं तो बचे हुए में spaceस्टोर होता है।

(ii)VARCHAR(C):–यह वेरीयेबल लम्बाई की स्ट्रींग के लिए है यहॉ परCस्ट्रींग कीmaximum लम्बाई है जिसकोयूजर स्पेसीफाई करता है।

(3) DateऔरTimeडेटा टाइप्स :– MySQLमें निम्नलिखितTimeऔरDateडेटा टाइप्स है।

(i)Date:–इस डेटा टाइप्स में कोईDate,YYYY-MM-DDके प्रारूप में स्टोर होती है। तथा वह1000-01-01एवंम् 9999-12-31के मध्य हो सकती है।उदाहरण स्वरूपStudentटेबिल में किसी छात्र की ageअगर 1st july, 1980 हो तो यह1980-07-01 के प्रारूप मेंस्टोर होगी ।

(ii)DATETIME:– इस डेटा इाईप के द्वाराDate एंवमTimeके मेल को स्टोर कर सकते है। जिसका प्रारूपYYYY-MM-DD HH:MM:SSहै।यह Date एंवम्Time 1000-01-01 00:00:00 एंवम्9999-12-31 23:59:59 के मध्य हो सकता है। उदाहरणस्वरूप मध्यान्त2:35दिनाँक1st july, 1980 को1980-07-01 14:35:00इस तरह स्टोर कर सकते हैं।

(iii)Time:–यह समय कोHH:MM:SSके प्रारूप में स्टोर करता है।

(iv)Year:–यह वर्ष को2 डिजीट या 4 डिजीट के प्रारूप में स्टोर करता है।

डेटा मेंनीपुलेशन लेंग्वेज(Data Manipulation language):–यह SQLका वह भाग हैं।जिसे हम क्यूरी लेंग्वेज भी कहते है। अतः DMLएंवम् क्यूरी लेग्वेज समानार्थक शब्द है।DMLका उपयोग डाटा जो किरिलेशन में स्टोर है को मेनीपूलेशन (इन्सर्ट ,डिलीट ,अपडेट और रिट्रीवल) करने में करते है।

DML निम्नलिखित मेंनीपूलेशन कमाण्डस् रखता है।

(i) SELECT

(ii) UPDATE

(iii) INSERT

(iv) DELETE

डेटा कन्ट्रोल लेंग्वेज(Data Control Language):–डाटा कन्ट्रोल लेग्वेज SQLकी उपकमाण्डस् का संग्रह है जो डाटा बेस में डाटा की सुरक्षा तथा डाटा को मेंनीपूलेशन के अधिकारों से सम्बंधित है।

DCLमें निम्न कमाण्डस् आती है।

(i) COMMIT

(ii) ROLLBACK

(iii) GRANT

(iv) REVOKE

DDLरिलेडेट कमाण्डस एंवम् उनके सिन्टेक्स :– टेबिल या रिलेशन डिफाइन करने के लिएSQL में CREATE Tableकमाण्डस का प्रयोग किया जाता है। जिसका सिन्टेक्स निम्नानुसार है।

Table createकरने का Syntex :–MySQLमें एक टेबिल बनाने का generic syntaxहै

CREATE TABLE table_Name (F1 D1, F2 D2,……. ,Fn Dn<Integrity Constraints1,……<ICk>);

इस Syntexमें प्रत्येक टेबिल के फिल्ड या ऐट्रीब्यूट का नाम है। तथाDiप्रत्येकFi के अन्तर्गत आने वाली वेल्यूस का डोमेनटाइप है। इसके अलावा हम टेबिल पे कई तरह के कन्सट्रेन्ट लगा सकते है। जैसे कि PRIMARY KEY, FOREIGN KEY आदि है।उदाहरण स्वरूप अगर हम studentके नाम से कोई टेबिल बनानी हो जिसका schema है।

Student(Roll_No, Name, Age, Address, Class)तथा अन्य टेबिल Classesकाschema है।Classes(Class_Name , CRoomNo, Cstrength)अतःSQLमें इन टेबिल के लिए Syntexहोगा

CREATE TABLE Student (Roll_No int , Name CHAR(20), Age int Address VARCHAR(30), Class CHAR(10));

studentटेबिल पे अगर कन्सट्रेन्टस लगाने हो तोCREATE TABLE Student

(Roll_No int NOT NULL AUTO_INCREMENT, Name CHAR(20), Age int NOT NULL,Address VARCHAR(30), Class CHAR(10) NOT

NULL, primary key(Roll_no), foreign key(class) references classes(class_name));

यहाँ पर NOT NULLका प्रयोग उस ऐट्रीब्यूट की वेल्यूNULLना हो इसलिए किया गया हो अगरuserइस फिल्ड मेंNULLवेल्यू डालना चाहेगा तो SQL errarदेगा ।oreign keyकन्सट्रेन्टस का उपयोग ऐसे आपरेशनको रोकने के लिए है जो studentतथा Classesटेबिल में लिंक तोडते हैतथा AUTO_INCREMENTका प्रयोग Roll_noफिल्ड में वेल्यू को एक से आगे से बढाने के लिए किया जाता है। इसकी by defaultवेल्यू एक हाती है। अगर हम चाहते है कि यह sequenceकिसी और वेल्यू से प्रारम्भ हो तो हम इसSyntexका प्रयोग करते है।

ALTER table student AUTO_INCREMENT=100

Classesटेबिल के लिए Syntex

CREATE Table Classes

(Class_Name CHAR(10) NOT-NULL, CRoomNo CHAR(10),

PRIMARY KEY (Class_Name));

उक्त टेबिल को MySQL Promptकी सहायता से बना सकते है जैसे कि

root@host# MYSQL-u root -p

enter passsword : *****

MySQL> use School_Management;

यहॉ परuseकमाडण्स का उपयोग School_Managementके नाम से जो हम सर्वप्रथम डाटा कैसे बनायेगे।

उसमें प्रवेश केलिए करते है।डाटाबेस बनाने के लिए

MySQL> CREATE DATABASE School_Management;

तथा इसके बाद MySQL> use School_Management

MySQL> CREATE TABLE Student

(Roll_No Int NOT NULL AUTO_INCREMENT,Name CHAR(20), Age int NOT NULL,Address VARCHAR(30),Class CHAR(10) NOT NULL,PRIMARY KEY (Roll_No), FOREIGN KEY (Class) REFERENCES Classes(Class_Name));

-> Query ok, o row affected

MySQL>MySQL में किसी भी कमाडण्स को टर्मिनेट करने के लिए अन्त मेंसेमीकालन लगाते है। किसी टेबिल को डाटाबेस से हटाने का Syntexनिम्न है।

Generic syntax:–

DROP TABLE table_name;

Studentटेबिल को हटाने के लिए

DROP TABLE Student;

ALTER tableकमाडण्स :– इन कमाडण्स का उपयोग किसी टेबिल मे नया स्तंभ जोडने(add),delete करने यMODIFY करने में करते है। जिनके Syntexनिम्न है।

(1) नया स्तंभ(**column**)जोडने के लिए:–

ALTER TABLE table_name ADD column_name datatype;

डदाहणार्थClasses टेबिल में एक नयाcolumn *classes* strengthजोड. सकते है।

Alter table classes add class_ strength int

(2) कोई स्तंभ टेबिल को हटाने के लिए:–

ALTER TABLE table_name DROP COLUMN column_name;

(3) किसी स्तंभ का डेटा टाइप्स बदलने के लिए:–

ALTER TABLE table_name MODIFY column_name datatype ;

उदाहरण :–

ALTER TABLE Student MODIFY Age Date;

अन्य उदाहरण:–

CREATE TABLE Teacher (Tname VARCHAR(20), DOB Date, Salary FLOAT(5,2), Address VARCHAR(30), phone int PRIMARY KEY (Tname));

CREATE TABLE Teaches (Tname VARCHAR(20),Class_name CHAR(10),PRIMARY KEY(Tname, Class_name), foreign key(Tname) REFERENCES Teacher(Tname),

FOREIGN KEY(Class_Name)REFECENCES Classes (Classs_Name));

Checkकन्सट्रेन्ट Syntex:–

Checkकन्सट्रेन्टSyntexका उदाहरण निम्नलिखित है।

CREATE TABLE Student

(Roll_No int NOT NULL AUTO_INCREMENT,Name CHAR(20),Age int NOT NULL,Address VARCHAR(30),Class CHAR(10) NOT NULL,PRIMARY KEY(Roll_No),

Check(Class in ('1st', '2nd', '3rd', '4th', '5th', '6th', '7th', '8th', '9th', '10th', '11th', '12th')));

Create Index कमाण्डस :–यह कमाण्ड किसी टेबिल पर इन्डेक्स क्रिऐट करने के काम आती है। हम इन्डेक्स को नही देख सकतेपर यह टेबिल में डाटा तेजी से सर्च करने में मदद करता है इसका Syntexनिम्न है।

CREATE INDEX Index_name ON table_name (column_name)

CREATE INDEX Index on Student(Address)

डेटा मेनीपूलेशन कमाण्डस और उनके रिनट्रेक्स :–

SQL DMLक्माण्डस निम्नलिखित है।

(i) INSERT

(ii) DELETE

(iii) UPDATE

(iv) SELECT

(i) INSERTकमाण्ड :– CREATE TABLEकमाण्ड के उपयोग द्वारा किसी टेबिल को बनाने पर एक खाली टेबिलबनती हैं अर्थात उस टेबिल में किसी प्रकार की कोई वेल्यू या रिकार्ड या टप्लस नहीं होते है।किसी टेबिल में रिकार्ड या डाटा डालने के लिए हम SQL INSERT INTOकमाण्डस का उपयोग करते है।

MySQL Syntax :-

INSERT INTO
Table_Name(Column_Name1,Column_Name2,……,Column_Namen)VALUES(Val e1,value2,……valuen);

स्ट्रींग टाइप्स के डाटा के लिए सभी वेल्यूज को सिंगल या डबल quotes (" ")में लेंगें।

MySQL कमाण्ड Promptसे डाटा का Insertion:–

उदारहण स्वरूप अगर हमेंStudentटेबिल में नई वेल्यूस या डाटा डालना है तो

MySQL> USE School_Management;

MySQL>INSERT INTO Student (Name, Age, Address, Class)

VALUES(“HARI”,15,”Chittorgarh”,”10th”);

इस उदाहरण में हमने Roll_Noनहीं लिया क्योकि टेबिल बनाते वक्त उसको हमने auto incrementदिया है। इसलिए MySQL automaticallyइसकी वेल्यू देगा। अन्य Syntex है।

INSERT INTO Student VALUES (“prakash”,18,”jaipur”,”12th”);

इनसर्ट के अन्यSyntexमें हम एक टपल्स को स्पेसीफइड करने के बजाय हम SELECTके उपयोग के द्वारा एक साथकई टप्लस इनर्सट कर सकते है।

(ii) SQL DELETEकमाण्डसः–DELETE कमाण्डस के द्वारा हम एक पूरा टप्लस DELETE करते है। इसके द्वाराहम किसी एट्रीब्यूट की वेल्यू को DELETEनहीं कर सकते है।

Syntax:-

DELETE FROM T ,WHERE P ;

यहाँ Tएक रिलेशन है जिसमें से टपल्स DELETEकरना है। तथा वह predicate (condition)है। जिसके अनुसारटपल्सDELETEहोगें।

DELETE FROM Student,WHERE Roll_No=105;

उक्तdelete statementके द्वारा हम वह टपल्सDELETEकरेगें जिस छात्रका Roll_No=105 है।

सारे टपल्स DELETEकरने के लिए हम निम्नSyntexलिखते है।

DELETE FROM Student ;

DELETE FROM Classes;

Class =10thके सभी छात्रों के डाटाDELETEकरने के लिए

DELETE FROM Student,WHERE Class =”10th”;

(iii) SQL UPDATEकमाण्डस :– UPDATE कमाण्ड का उपयोग हम किसी टपल्स के विशिष्ट एट्रीब्यूट की वेल्यू को बदलने के लिए करते है। अर्थात अगर हम पूरे टपल्स को नहीं बदलना चाहते है। केवल इसमें किसी ऐट्रीब्यूट की वेल्यू को ही बदलना चाहते है तो हम UPDATEकमाण्ड का उपयोग करते है। जिसका Syntex निम्न है।

Syntax:

UPDATE table_name SET first_field=value1, second_field=value2

[WHERE clause];

हम एक साथ कई फिल्ड की वेल्यू कोUPDATEकर सकते है।

उदाहरण

MySQL> UPDATE Student, SET Class="11th",WHERE Roll_No=12;

इस statementके द्वारा हम उस छात्र की classको बदलना चाह रहे है जिसकाroll_no 12 है। उसकी class को हम 10^{th}कीबजाय 11^{th}कर रहे है।

छात्र का**address**बदलना

UPDATE Student ,SET Address="Ajmer",WHERE Roll_No=102;

Student

RollNo	Name	Age	Address	Class
11	Hari	15	Jaipur	10th
101	Prakash	16	Kota	11th
102	Basu	11	Udaipur	6th
120	Viveka	9	Jaipur	4th

उक्तupdate statement के बादstudent table का instances निम्नलिखित होगा ।

Roll_No	Name	Age	Address	Class
11	Hari	15	Jaipur	10th
101	Prakash	16	Kota	11 th
102	Basu	11	Ajmer	6 th
120	Viveka	9	Jaipur	4 th

(iv) SQL SELECT statement

किसीSQL query expression में मुख्य रूप से निम्नलिखित तीन क्लॉजेसclauses होते है।अर्थात कोई भीSQL query जो हम रिलेशन डाटा बेस के लिए लिखते है। उसकी बुनियादी संरचना(basic structure)मेंउक्त 3 क्लाज होगें।

- SELECTक्लॉज में हम उन ऐट्रीब्यूटस को लिखते जो हमें हमारे आउट पुट रिलेशन में चाहिए।

- FROM क्लॉज में हम उन रिलेशन को लिखते है। जिनको हमेंquery expressionमें उपयोग करना है।FROMक्लाज मे लिखे रिलेशनस का Cartesian productहोता है।
- WHEREक्लॉज में हमpredicate लिखते है। जोFROM clause के रिलशनस के ऐट्रीब्यूटस को लिप्त रखता है।अर्थात जिसकीBoolean वेल्यू(true or false) होती है।

SQL query का निम्न फोर्म (form)होता है।

SELECT AT1, AT2, AT3 , …,ATn,

FROM r1, r2, r3,…rrn,

WHERE P;

यहाँATi एक ऐट्रीब्यूट को प्रदर्शित करता है। और r_i एक रिलेशन को ।P एक predicateहै।

SELECT clause:SELECTक्लाज उन ऐट्रीब्यूट की लिस्ट रखता हैं जो उस रिलेशन से retrieveकरने है।

उदाहरण :–SELECTक्लाज को समझने के लिए हमSchool Management डाटा बेस में से एक टेबिल को लेते है। जोकि निम्नलिखित हैं–

Roll_No	Name	Age	Address	Class
101	Bhagat singh	20	Ajmer	12th
105	Chandra	17	Kota	11 th
12	Shekhar	16	Jaipur	10th
17	DinDayal	15	Udaipur	9 th
90	Rohit	9	Ajmer	5th

SELECTकमाण्ड काSyntaxहैं।

SELECT field_names, FROM realtion_names;

Field namesमें एक साथ कई फिल्डfetchकर सकते है। इसके अलावा हम स्टार(*) भी फिल्ड के नाम के बजायलगा सकते हैं तबSELECTफिल्डFROM क्लाज के रिलेशन के सारे फिल्ड Returnकरेगा ।

उदाहरण

SELECT Roll_No, Age,
FROM Student;

रिजल्ट

RollNo	Age
101	20
105	17
12	16
17	15
90	9

SELECT * FROM Student ;

Output->

RollNo	Name	Age	Address	Class
101	Bhagat	20	Ajmer	12th
105	Chandra	17	Kota	11 th
12	Shekhar	16	Jaipur	10th
17	DeenDayal	15	Udaipur	9 th
90	Rohit	9	Ajmer	5th

Duplicateफिल्ड वेल्यूज को हटाने के लिए SELECT में DISTINCT keywordका उपयोग करते है।

SELECT DISTINCT AddressFROM Student ;

यहाँ Studentटेबिल केAddressफिल्ड को fetchकरने पर उसमें Duplicateवेल्यू Ajmer एक ही बार आयेगी ।

रिजल्ट

Address
Ajmer

Kota
Jaipur
Udaipur

SELECTक्लाज में हमColumn के नाम के साथ साथ निम्नलिखित अर्थमेटिक एक्सप्रेशनभी उल्लेखितकर सकते है।

Description	Operator
Addition	+
Subtraction	-
Division	/
Multiplication	*

डदाहरण के लिए हम Teacherटेबिल को लेते है।

Teacher

Tname	Address	Salary	Phoneno
Radha krishan	Kerla	5000	1234567899
Lalaji	Udaipur	750	9413962123 9999999999
Sarvopalli	Rampur	2000	9312171080 3333333333
leadgevan	Maharastra	9000	5189310510 2121212121

SELECT Tname, Salary *10,

FROM Teacher;

इसqueryको रिजल्ट मेंSalary ऐट्रीब्यूट की वेल्यूज 10 से मल्टीपलाई हो कर मिलेगी।

रिजल्ट

Tname	Salary*10
Radha krishnan	50,000
Lalaji	7500

Sarvopalli	20,000
Leadgevan	90,000

SELECT statement का उपयोग **WHERE**क्लाज के साथ :–WHEREक्लाज मे एक या ज्यादा कन्डीशन्स को लिखते है। जब यह शर्त (condition)संतुष्ट होगी तभी रिलेशन से कोई पंक्ति निकाल सकते है।

उदाहरण स्वरूप Teacherटेबिल के लिए निम्न queryलिखते है।

SELECT Tname, Salary,FROM TeacherWHERE Salary>2000;

तो निम्नResultप्राप्त होगा ।

Tname	Salary
Radha krishnan	5000
Harivansh	9000

WHERE clauseनिम्नलिखित लॉजिकलconnectionsउपयोग करता है।

(i) AND

(ii) OR

(iii) NOT

WHEREक्लाज कम्पेरिजन (Comparision)आपरेटर्स भी उपयोग करता है।

Description	Operators
Less than	<
Less than or equal to	<=
Greater than or equal to	>=
Greater than	>
Equal to	=
Not equal to	!= or <>

उदाहरण :–

SELECT Tname,FROM Teacher,WHERE Salary>2000 AND Address=”Kerala”;

रिजल्ट

Tname
Radha krishnan

(3) डाटा कन्ट्रोल लेंग्वेज कमाण्डसor DCLकमाण्डस :–DCL कमाण्डस डाटाबेस की सुरक्षा से सम्बन्धित है । यहकमाण्डस यूजर्स को विशेषाधिकार प्रदान करती है। ताकि यूजर्स डाटाबेस में कुछ आब्जेक्टस को accessकर सके ।

SQL GRANTकमाण्डस :– रिलेशनल डाटाबेस में एक यूजर के द्वारा बनाये(create)गये आब्जेक्टस कोदूसरे यूजर्स जब तक नही पहुँच(access)सकते जब तक उनको बनाने वाला यूजर्स दूसरे यूजर्स को इसकी सहमती नहीं देता। यह सहमती(permission)GRANTकमाण्ड के उपयोग द्वारा दी जा सकती हैं।

GRANT statement कई तरह के विशेषाधिकार कई आब्जेक्टस टेबिल व्यूपर प्रदान करता हैं ।

Syntax:-

GRANT [type of permission] ON [database name] . [table_name] TO'[user name] '@'localhost'; identified by password;

यहाँपर type of permissionनिम्नलिखित होती है।

- CREATE- नई टेबिल या डाटाबेस बनाने की सहमती देता है।
- DROP . टेबिल या डाटाबेस को हटाने(delete)की सहमती देता है।
- DELETE – टेबिल से पक्ति या टापल्स हटाने(delete)की सहमती देता हैं
- INSERT- टेबिल में पक्ति या टापल्स जोडनें(insert) की सहमती देता है।
- SELECT – इस कमाण्डस के द्वारा टेबिल या डाटाबेस को(read)करने की सहमती देता है।
- UPDATE. टेबिल की पक्तियों कोupdate की सहमती देता हैं

उपर दिये syntaxमें हम डाटा बेस या टेबिल के नाम के स्थान पर अगरasterisk (*) लगाते है। तो यहsyntaxकोई भीडाटाबेस या टेबिल के पहुँच(access)की सहमती(permission)देगा।

Syntax:-

GRANT ALL Privileges ON *.* TO 'new_user @ 'localhost';

यहाँasterisk (*) डाटाबेस व टेबिल को बताता हैं। यह कमाण्ड यूजर कोread, edit ,execute और सभी आपरेशनस कीसहमती सभी डाटाबेस और टेबिल के लिए देता है।

उदाहरण :– MySQL मेंGRANT कमाण्डस को समझते है।

नया यूजर बनाना:–

MySQL में Defaultयूजर्स Rootहोता है। जिसकी सभी डाटाबेस पर फुल पहुँच (full access)होता है। MySQLमें नया यूजर बनाने को syntax है।

Syntax

MySQL> CREATE USER 'new_user' @'localhost'

IDENTIFIED BY 'Password';

उदाहरण –MySQL> CREATE USER '14EEACS350' @'localhost' IDENTIFIED BY '123456';

उदाहरण :–उक्त कमाण्ड द्वारा एक नयायूजर '14EEACS350' के नाम से बनेगा जिसका पासवर्ड'123456' होगा।हॉलाकि इस यूजर को डाटा बेस के साथ कुछ भी करने की सहमती नही है। अभी तक तो इस नये यूजर कोMySQLमें लॉगिनभी नहीं हो सकता है। इसलिए सर्वप्रथम नये यूजर कोसहमती देनी आवश्यक है।

उदाहरण :–GRANT ALL PRIVILEGES ON

School_Management.* TO '14EEACS350'@'localhost'

IDENTIFIED BY '123456';

इस कमाण्ड के द्वारा नये यूजर(14EEACS350)को School Managementडाटाबेस की सभी टेबिल पर सभी पहुँच(access)प्राप्त होगें।एक बार यूजर के लिए सहमतीfinalहोने के बाद निम्न कमाण्ड चलाये

FLUSH PRIVILEGES;

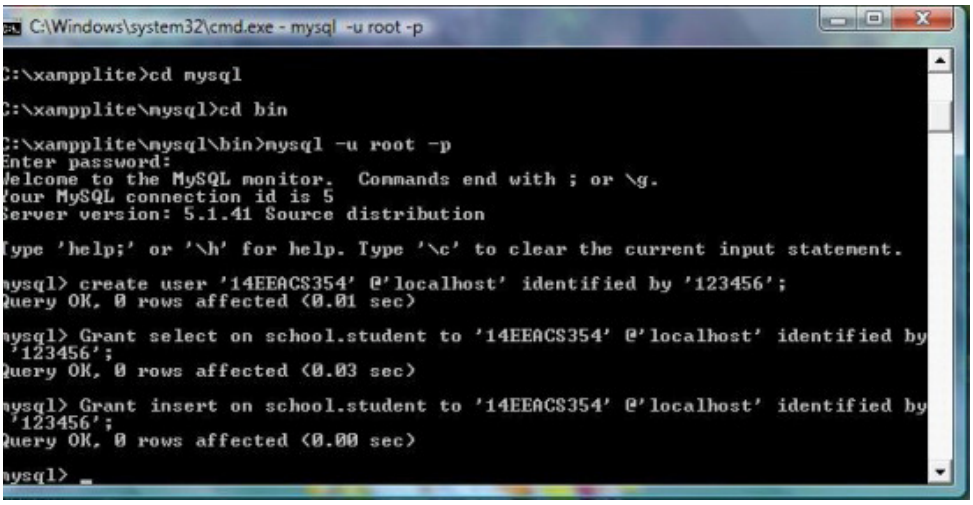
तकि सभी बदलाव प्रभावी हो सके। अगर हम सारे विशेषाधिकार'14EEACS350' यूजर को नही देना चाहते है। तो केवलreadके अधिकार के लिए

GRANT SELECT ON School_Management.Student TO '14EEACS350' @ 'localhost' IDENTIFIED BY '123456';

कमाण्ड चलाये इस कमाण्ड द्वारा 14EEACS350' को केवल देखने का अधिकार वो भी केवल Student टेबिल को प्राप्त होगा।Student टेबिल में INSERTके विशेषधिकार के लिए

GRANT INSERT ON School_Management.Student TO '14EEACS350' @ 'localhost' IDENTIFIED BY '123456';

कमाण्ड का उपयोग करे । अन्य विशेषधिकार भी प्रदान किये जा सकते है।



चित्र–3

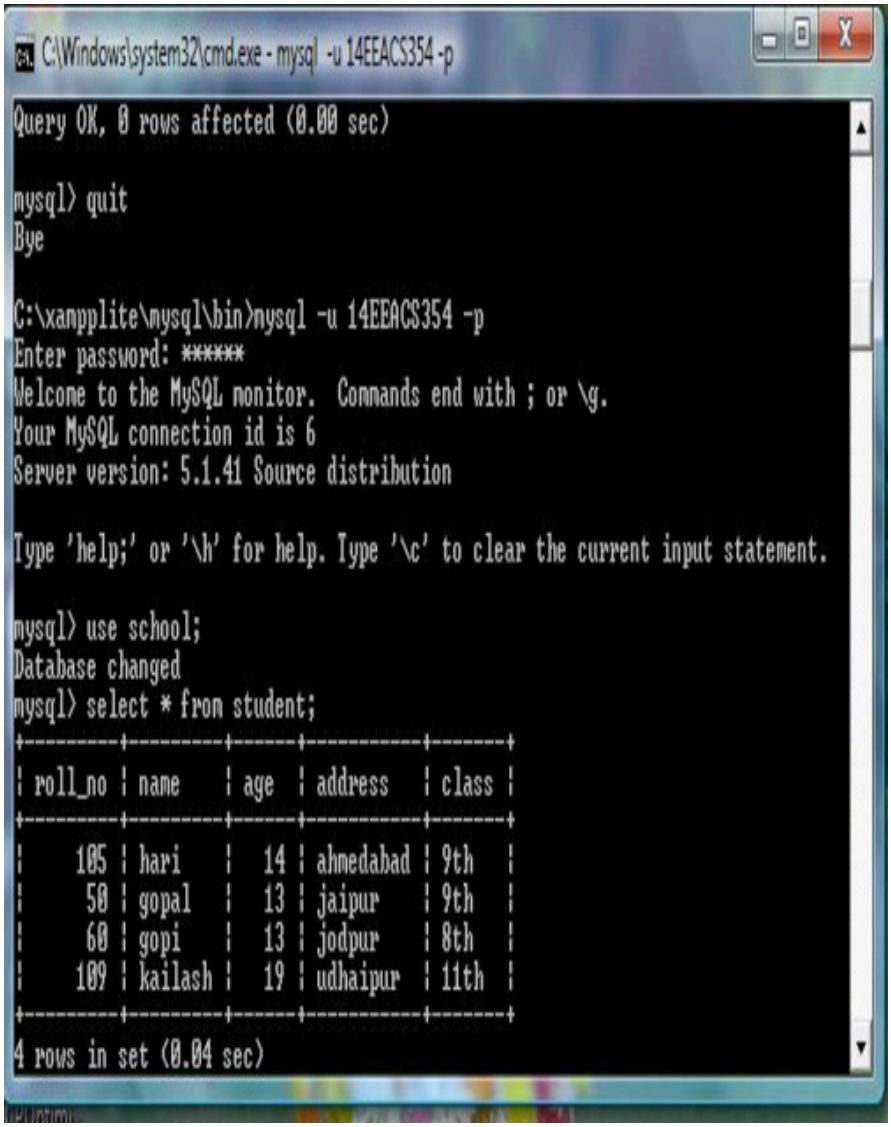
क्योकि हमने केवलStudent टेबिल पर14EEACS350' यूजर को केवल select एवंम्insert विशेषाधिकार प्रदानकिया है। अतः यह यूजरStudentटेबिल में कोई बदलाव(update)अथवा हटाना(delete)कमाण्ड नहीं चलसकता है।इसको देखने के लिए हम एक बार Quitकमाण्ड द्वारा logoutहाते है। एवम् नये यूजर के लिए निम्नलिखित कमाण्ड द्वारा वापिस login करते है।

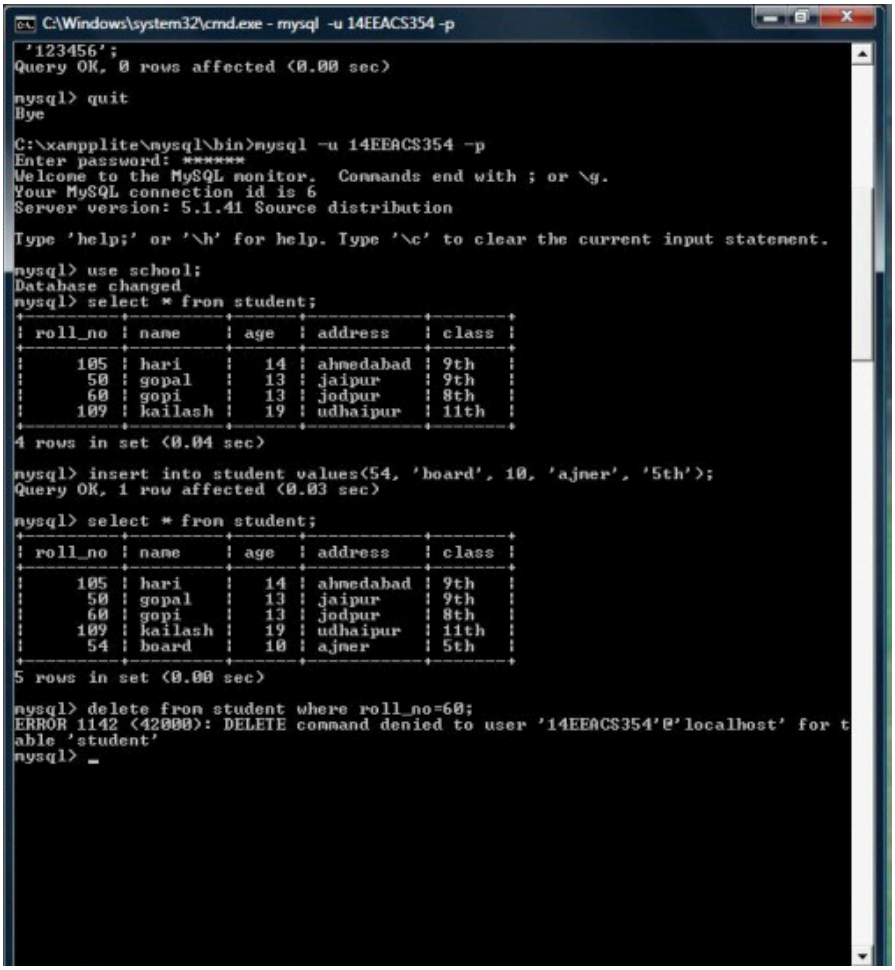
MySQL-u[new username]-P;

MySQL> MySQL-u 14EEACS350 -P;

Enter password :123456;

यहाँStudentटेबिल पर भिन्न भिन्न कमाण्ड चलाते है।जिनके screen shots है।


C:\Windows\system32\cmd.exe - mysql -u 14EEACS354 -p
Query OK, 0 rows affected (0.00 sec)
mysql> quit
Bye
C:\xampplite\mysql\bin>mysql -u 14EEACS354 -p
Enter password: ******
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.1.41 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use school;
Database changed
mysql> select * from student;
roll_no	name	age	address	class
105	hari	14	ahmedabad	9th
50	gopal	13	jaipur	9th
60	gopi	13	jodpur	8th
109	kailash	19	udhaipur	11th
4 rows in set (0.04 sec)



2 REVOKEकमाण्ड :–

REVOKEकमाण्डस के द्वारा किसी आब्जेक्ट (टेबिल) से सहमती वापिस लेने के लिए करते है। जिसकीSyntax GRANTके समान ही है।

Syntax:-

REVOKE [type of permission] ON [database_name].

[table_name] FROM '[user name] '@'localhost';

उदाहरण के तौर पर हमDROP के द्वारा किसी यूजर को हटाने(delete)करने के लिए कर सकते है।

उदाहरण :–DROP USER '14EEACS350' @'localhost';

REVOKE DELETE ON Student FROM 14EEACS350;

(3)COMMITकमाण्डः–COMMITकमाण्ड का उपयोग सभी प्रकार के जो बदलाव डाटाबेस पे किये गये है। उनबदलावों के डाटाबेस में स्थायी करने के लिए किया जाता है।MySQL में हर समय auto commit mode enabledहोता है। जिसका अर्थ कि जैसे ही हम कोईupdate कमाण्डसके द्वारा टेबिल कोmodfiyकरते है। तो यह बदलावMySQL मेंDISK पर होता है। जिससे की यह स्थायी हो सके ।

START TRANSACTIONके द्वारा हम auto commit को disabledकर सकते है।

SQL आपरेटर्स का परिचय :–SQL आपरेटर्स एक प्रकार के रिजर्व (सुरक्षित) शब्द है। जो किसी SQLक्यूरी के WHEREक्लाज में उपयोग किये जाते है। जिनमें मुख्य ऑपरेशन निम्न है।

(i) कम्पेरिजन(Comparison)

(ii) अर्थमेटिक(Arithmetic)

(iii) लोजिकल(Logical)

(iv) शर्त को निगेट करने के कार्य मे आने वाल आपरेटर्स

(i) कम्पेरिजन आपरेटर्स :– आपरेटर्स विवरण निम्नलिखित है।

आपरेटर्स	विवरण
(=)	यह आपरेटर्स दो आपरेटर्स की वेल्यू की समानता या असमानता को चेक करता है। समान होने पर शर्तtrueहोगी।
(<> or !=)	यह आपरेटर्स दो आपरेटर्स की वेल्यू की समानता या असमानता को चेक करता है। अगर वेल्यू समान नही है तो शर्त trueहोगी।
(>)	अगर बॉये तरफ के आपरेन्ड की वेल्यू दॉये तरफ के आपरेन्ड से ज्यादा हो तो शर्त trueहोगी।
(<)	अगर बॉये तरफ के आपरेन्ड की वेल्यू दॉये तरफ के आपरेन्ड से कम हो तो शर्त trueहोगी।
(>=)	अगर बॉये तरफ के आपरेन्ड की वेल्यू दॉये तरफ के आपरेन्ड से ज्यादा या समान हो तोशर्त true होगी।
(<=)	अगर बॉये तरफ के आपरेन्ड की वेल्यू दॉये तरफ के आपरेन्ड से कम या समान हो तो शर्त trueहोगी।

(ii) SQLअर्थमेटिक आपरेटर्स :– आपरेटर्स विवरण निम्नलिखित है।

आपरेटर्स	विवरण
(+)	आपरेटर्स की दोनों आपरेन्ड की वेल्यू को जोडता (add)है।
(-)	बॉयी तरफ के आपरेन्ड की वेल्यू में से दाँयी तरफ की आपरेन्ड की वेल्यू का घटाव (subtract)करता है।
(*)	दोनों तरफ की आपरेन्ड की वेल्यू को गुणा करना ।

(% or मॉडूलस) बाँयी तरफ के आपरेन्ड की वेल्यू को दॉयी तरफ के आपरेन्ड की वेल्यू से भाग(divide)देता है। तथाउनका शेष रिटर्न करता है।

SQLअर्थमेटिक व कम्पेरिजन आपरेटर्स को समझने लिए हम एक बार नीचे दी गईStudent टेबिल, Teacherटेबिलऔर Classesटेबिल का उदाहरण लेते है।

Student

Roll_No	Name	Age	Address	Class
109	Omprakash	9	Jodhpur	4th
111	Prakash	15	Jaipur	10 th
91	Suman	11	Jaipur	6th
75	Shanker	13	Ajmer	8th

Teacher

Tname	Address	Salary	PhoneNo
Radha krishnan	Kerla	3000	1234567899
Rajesh	Jodhpur	5000	1111162123
Lalaji	Ajmer	9000	1111171080
Hariom	Kerla	40000	1111110510

Classes

Class_name	CroomNo	CStrength
12th	F-1	90
6th	F-17	65
9th	S-21	110

उदारहण 1

SELECT * FROM Student,WHERE Age=15;

Output ->

Roll_No	Name	Age	Address	Class
111	Prakash	15	Jaipur	10 th

क्योकिStudentटेबिल में केवल एक ही पक्ति ऐसी है। जिसमें age15 हैं।

उदारहण 2

SELECT * FROM Student, WHERE Age>11;

Output ->

Roll_No	Name	Age	Address	Class
111	Prakash	15	Jaipur	10 th
75	Shanker	13	Ajmer	8th

उदारहण 3

SELECT * FROM Student, WHERE Age <= 13;

Output ->

Roll_No	Name	Age	Address	Class
109	Omprakash	9	Jodhpur	4th
91	Suman	11	Jaipur	6th
75	Shanker	13	Ajmer	8th

(iii) SQLलॉजिकल आपरेटर्स :– आपरेटर्स विवरण निम्नलिखित है।

आपरेटर्स	विवरण
(AND)	WHERE क्लाज में विभिन्न शर्तो को अलाऊ करता है। अर्थातa AND bदोंनों शर्तो aएंवम् b trueहोने परइसका परिणामtrueहोगा।
(OR)	यह WHEREक्लाज में विभिन्न शर्तो को संयुक्त करता है। अर्थात a OR bमें कोई भी शर्त aया bके trueहोनेपर इसका परिणाम trueहोगा।

उदाहरण के लिए हमTeacher टेबिल को लेते है।

(i) SELECT Tname, SalaryFROM Teacher WHERE Salary <= 5000 and Salary >=4000;

उक्त क्यूरी मेंSalary <= 5000 के लिए दो पक्ति में वेल्यू5000 से कम एव समान है। अतः शर्त trueतथा Salary >=4000 के लिए 3 पक्ति में वेल्यूस 4000 से अधिक है। पर दोनों शर्त केवल एक पक्ति के लिएtrue है। अतः इसकापरिणाम होगा।

Tname	Salary
Rajesh	5000

(ii) SELECT Tname, Salary,FROM Teacher,WHERE Salary <= 5000 OR Address = "Kerla";

Output ->

Tname	Salary
Radha krishnan	3000
Rajesh	5000
Hariom	40000

उक्त परिणाम में तीन पक्तिया प्राप्त हुए है। क्योंकि <=5000 के लिए दो पक्ति में वेल्यू इसके कम या समान है। जबकी address=" Kerla" भी दों पक्तियों के लिएtrue है। हॉलाकि एक पक्ति मेंSalaryकी वेल्यू 5000से अधिक पर यह शर्त यहाँ cheek नही होगी अतः परिणाम उक्त प्राप्त होगा क्योकि ORदोनो शर्तो को संयुक्त कर परिणाम देगा ।

आपरेटर विवरण

Not(!) यह आपरेटर आपरेन्ड की वेल्यू को इन्वर्स करता है।

उदाहरण :– SELECT *FROM StudentWHERE !(Roll_No= 111);

Output ->

RollNo	Name	Age	Address	Class
109	Omprakash	9	Jodhpur	4th
91	Suman	11	Jaipur	6th
75	Shanker	13	Ajmer	8th

यहRollNo 111 के अलावा बाकी सभीStudent की सूचना देगा

आपरेटर विवरण

BETWEEN यह आपरेटर्स वेल्यूस के मध्य वेल्यूस को सर्च करता है।

उदाहरण :–

SELECT Tname , SalaryFROM TeacherWHERE Salary BETWEEN 3000 and 5000;

Output ->

Tname	Salary
Radha krishnan	3000
Rajesh	5000

आपरेटर विवरण

ANY इस आपरेटर के द्वारा हम वेल्यू को किसी लिस्ट में अन्य वेल्यू से कम्पेयर (compare)करने काम में लेतें है।

Syntax:- ANY आपरेटर के निम्नलिखितsyntaxहो सकते है।

<ANY, <=ANY, >=ANY, =ANY and <>ANY

उदाहरण :–अगर हमें किसी वेल्यू 5 को लिस्टसे compareकरना हो तो ANYके द्वारा इसcomparision कापरिणाम निम्नानुसार होगा ।

Expression	परिणाम	टिप्पणी
(1)	5 >ANY {0, 5 ,6}	True क्योंकि यहाँ 5, 0 से बड़ा है।
(2)	5 =ANY {0, 5 , 6}	True क्योंकि यहाँ 5, 5 से समान है।
(3)	5 <ANY{0, 5, 6}	True क्योंकि यहाँ 5, 6 से छोटा है।
(4)	5 >ANY{5, 6}	False क्योंकि यहाँ 5 किसी भी नम्बर से बड़ा नही

अर्थात ANYआपरेटर कम से कम एक के लिए शर्त सही होने परTrueदेता है।

आपरेटर विवरण

ALL इस आपरेटर के द्वारा हम एक वेल्यू को किसी अन्य लिस्ट की सभी वेल्यूस से(compare)करने के काम लेतेहै।

अर्थातALL आपरेटर लिस्ट की सभी वेल्यूस के लिए शर्त सही होने पर हीtrue returnकरता है।

उदाहरण :–

Expression	परिणाम	टिप्पणी
(1) 5<ALL{5,6}	False	यहाँ 5 केवल 6 से ही छोटा है।
(2) 5>ALL{0, 3, 4}	True	यहाँ 5 सभी से बड़ा है।
(3) 5<=ALL{5, 6}	True	क्योंकि यहाँ 5ए 6 से छोटा है। तथा 5 के समान है।

उक्त आपरेटर का उपयोग हम आगेsubqueries लिखने में करेंगें

आपरेटर	विवरण
LIKE	इस आपरेटर का उपयोग किसीstringमें pattern matching के लिए करते है।

निम्न दो विशेष करेक्टर के द्वाराPatternको विवरणदे सकतें है।

(1) Underscore (–):– यहाँ(–)करेक्टर कोई भी करेक्टर कोmatchकरता है।

(2)Percent (%) यहाँ(%) करेक्टर कोई भीsub stringकोmatchकरता है।

कुछ डाटाबेस में pattern matchingकेस सेन्सीटीव होती है। जबकि MySQLमें यह(by default) केस इनसेन्सीटीव होती हैअर्थात करेक्टर uppercase (B) करेक्टरlowercase (b) कोmatchकरता है।तथाlowercase,Uppercaseको मैंच करता है।

उदाहरण(1):– अगर हमेंStudent टेबिल से हमें वो सभीName निकालने है। जिनका प्रारम्भSसे होता हैं तो इसकाSQL Syntaxहोगा ।

MySQL> SELECT *FROM Student WHERE Name LIKE 'S%';

Output ->

Roll_No	Name	Age	Address	Class
91	Suman	11	Jaipur	6th
75	Shanker	13	Ajmer	8th

उदाहरण(2) :–वे नाम जिनका अंतshसे होता है।

MySQL> SELECT * FROM Student WHERE Name LIKE '%sh';

Output ->

Roll_No	Name	Age	Address	Class
109	Omprakash	9	Jodhpur	4th
111	Prakash	15	Jaipur	10 th

उदाहरण(3):– वे नाम जो 5 करेक्टर से मिल के बने हो

MySQL> SELECT *FROM Student WHERE Name LIKE '-----';

Output->

Roll_No	Name	Age	Address	Class
91	Suman	11	Jaipur	6th

किसी भीpattern में अगर हम विशेषpattern करेक्टर (%, _) को भी भामिल करना चाहते है। तो हमेंescape character (\) Backslashको उपयोग करते है।इसescape characterको हमें विशेषpattern करेक्टर के तुरंतपहले लगाते है।

उदाहरण :–LIKE 'BJ\%BHARAT%' उस सभी स्ट्रीग कोmatchकरेगा जिनको प्रारम्भ' BJ\%BHARAT%' से होता है।

LIKE 'BJ\\BHARAT%' उन सभी स्ट्रीग को मेच करेगा जिनका प्रारम्भBJ\%BHARAT% से होगा ।

SQL NOT LIKEके उपयोग द्वाराmismatchesको भी ढुंढा जा सकता है।MySQLमेंExtended regular expression के उपयोग द्वारा भी pattern matchingकी जाती है।इसके लिएMySQLमेंREGEXP(RLIKEऔर NOT REGEXP(NOT RLIKE) आपरेटर्स का उपयोग करते है।

"[…]"एक करेक्टर क्लास है जो ब्रेकेटस के अन्दर कोई भी करेक्टर कोmatchकरेगी

उदाहरण :–"[p q r]" "p", "q" or "r"को मैच करती है।

"[a-z]" किसी भी लेटर को मैच करती है।

आपरेटर **विवरण**

IS NULL इस आपरेटर का उपयोग किसी वेल्यू को एक NULLवेल्यू से कम्पेयर करने के काम लेते है।

अगर किसी टेबिल में किसी ऐट्रीब्यूट की वेल्यू उस वक्तAbsenceहै तो उस ऐट्रीब्यूज के लिए इस Absence information को बताने के लिए NULLवेल्यू का उपयोग करते है।

उदाहरण :–के लिए अगर Teacher टेबिल मे किसी टीचर का PhoneNo नही हैं तो वहाँ पर वेल्यूNULLआयेगी औरइसकोtest करने के लिएMySQLमें की–वर्डNULLका उपयोग करेगें ।

SELECT Tname FROM Teacher WHERE PhoneNo IS NULL;

आपरेटर्स एवंमNULLवेल्यू :–

आपरेटर्स	वेल्यू 1	वेल्यू 2	**Output**
+, -, *, or /	वेल्यू 1	null	NULL
+, -, * or /	null	वेल्यू	NULL
>,<, >=,<=,<>	null	वेल्यू	unknown
<,>,,>=,<=,<>,=	वेल्यू	null	unknown
AND	true	unknown	unknown
AND	false	unknown	false
AND	unknown	unknown	unknown
OR	true	unknown	true
OR	false	unknown	unknown
OR	unknown	unknown	unknown
NOT	unknown	unknown	unknown

अर्थात यदि उक्त टेबिल का उपयोग करते हुए यदि WHERE clause में predicateका रिजल्ट false या unknown आता है। तो कोई भी टप्ल रिजल्ट में नहीं आयेगा।

नोटः– सभी aggregateफक्शन(count(*))के अलावा null वेल्यू को जब वेउनके इनपुट लिस्ट में आती हैं, तो उनकोcalculation में नहींलेतें है। अर्थात नजर अंदाज करते हें।

SET Operators:–SQL के SET आपरेशन निम्नलिखित है। जो कि रिलेशनसके ऊपर आपरेटकरते है।

(i) UNION
(ii) UNION ALL
(iii) INTERSET
(iv) EXCEPT

UNION SET आपरेटर्स

इस आपरेटर का उपयोग दो या दो से अधिक SELECT statementके रिजल्ट सेट कोसंयुक्त(combine)करने के काम आता है। यह आपरेटर्स duplicate पक्तियो को हटा(remove)कर रिजल्ट देता है।

नोटः– इस आपरेटर्स के लिए यह आवश्यक है किUNIONहोने वाले दोनोंSELECT statementमें फिल्डस (fields) कीसंख्या एक समान हो तथा उनके डाटा टाइप्स भी एक समान होने चाहिए।

Syntax:- MySQL UNIONआपरेटर syntaxहै।

SELECT ex1, ex2,… ,ex_nFROM tables[WHERE condition]

UNION [DISTINCT]

SELECT ex1, ex2,… ex_nFROM tablesWHERE condition;

यहाँ परDISTINCTकी–वर्ड आवश्यक नहीं है। क्योंकिUNIONआपरेटरduplicate पक्तियों कोstatementमें सेहटा देता है।

उदाहरण :–

SELECT ClassFROM Student

UNION

SELECT Class_nameFROM Classes;

Output ->

Class
4th
10th
6th
8th
12th
9th

यह उदाहरण केवल एक फिल्ड के लिए है अर्थात SELECT statementकेवल एक फिल्डreturnकरता है। दोनोंफिल्डस के यहाँ डाटा टाइप्स समान है। रिजल्ट सेट केcolumn का नाम होगा। यहाँ पहले SELECT statementकेreturnका नाम होगा। क्योंकि हम जानतें है। कि MySQL UNIONआपरेटर्सduplicateको हटाता है। अतः6^{th} यहाँ एक ही

बार आया हैं। और अगर हम duplicate रखना चाहते है तो हम UNION ALL का उपयोग करेंगें।

Syntax :- UNION ALLकाSyntax UNIONजैसा है केवल UNIONके स्थान पर UNION ALL काउपयोग करेगें।

SQL INTERSET आपरेटरः– इस आपरेटर का उपयोग दो या दो से अधिक डाटा सेटस के intersectionके लिए कामआता है। अर्थात यदि दोनों डाटा सेटों में कोई रिकार्ड विद्यमान है तोINTERSETआपरेटर के रिजल्ट में हमें वह रिकार्डप्राप्त होगा अन्यथा अगर कोई रिकार्ड केवल एक ही डाटा सैट में है तो वह रिजल्ट में नहीं आयेगा।

उदाहरणः–

SELECT ClassFROM Student

INTERSET

SELECT Class_name,FROM Classes;

Output ->

Class
6th

नोटः– लेकिन MySQLमेंINTERSET आपरेटर उपलब्ध नहीं है।

MySQLINआपरेटर के द्वारा हम INTERSETआपरेशन कर सकते है।

Syntax:-

MySQL IN आपरेटर syntax

Expression IN(value1 , value2 , …$value_n$);

यहाँ expression वह वेल्यू है जिसको हम test करना है value1,value2,…valuen वह वेल्यूस है जिसमें हमे test वेल्यू को test करता है यदि कोई भी वेल्यू test वेल्यू से matchहोती है। तो IN आपरेटर trueकरता है।

SQL EXCEPT आपरेटरः–SQL EXCEPTआपरेटर दो SELECT statementको संयुक्त करने के उपयोग में आताहैं। और यह इनको संयुक्त इस प्रकार करता हैं कि पहलेSELECT statementकी वह पक्तियाँ जो दूसरेSELECT statementमें नहीं है, को परिणाम स्वरूप देता है।

Syntax:-

SELECT column namesFROM tables[WHERE clause]

EXCEPT

SELECT column namesFROM tablesWHERE clause;

MySQLEXCEPTआपरेटर को भी उपलब्ध नही करवाता है इसके लिए हम NOT IN आपरेटर का उपयोग कर सकते है।

SQL(functions)फंक्शन :–SQLकई प्रकार के उपयोगी built in फंक्शन उपलब्ध करवाता है। जिनका विवरण निम्न प्रकार है।

(1)DateऔरTimeफंक्शन :– में काम आने वाले Dateऔर Timeफंक्शन तथा उनका विवरण नीचे दिया गया है।

फंक्शन का नाम	विवरण
ADDDATE()	Dateको जोड़ता है।
ADDTIME()	Timeको जोड़ता है।
CURDATE()	यह वर्तमान Datereturn करता है।
CURTIME()	यह वर्तमान Time return करता है।
DATE_SUB()	यह दो Dates (subtracts) को घटाता हैं।
NOW()	यह वर्तमान Dateव Timeदेता है।
STR_TO_DATE()	यह स्ट्रींग को Dateमें परिवर्तित करता है।

उदाहरण :–

Syntax:- SELECT ADDDATE(expr, days)

MySQL> SELECT ADDDATE('1980-07-01',32);

Output -> 1980-08-02

उदाहरण :–

Syntax:- SELECT ADDTIME(expr2, expr1)

यहाँexpr1कोexpr2में जोड़कर रिजल्ट देगा ।

MySQL>SELECT ADDTIME('1999-12-31 23:59:59.999999', '11:1:1.000002');

Output -> 2000-01-02 01:01:01.000001

उदाहरण :–

CURDATE syntax :- SELECT CURDATE();

यहाँ वर्तमानYYYY-MM-DD फार्मेट में देगा।

(2) स्ट्रींग(String)फंक्शन :–Stringके उपयोगी स्ट्रींग फंक्शन निम्नानुसार है

नाम	विवरण
ASCII()	यह फंक्शन बांये छोर(left most)के करेक्टर कीnumericवेल्यू देगा।
BIN(N)	यहNकी बाइनरी वेल्यू का स्ट्रींग रिप्रजेन्टेशन (representationदेता है।
BIT_LENGTH(str)	यहstrस्ट्रींग की लम्बाई (length)बीटस में देगा।
CHAR(N)	यह हर आरग्यूमेन्टN को इन्टीजर मानकर उसका स्ट्रींग रिप्रजेन्टेशन देगा। यह स्ट्रींग उन करेक्टर्स सेमिलकर बनेगी जो इन्टीजरCHAR(N) में आरग्यूमेन्ट के तौर पर है।
CHAR_LENGTH(Str)	यहStrस्ट्रींग की लम्बाई करेक्टरस में नापता है।
CONCAT(Str1,Str2,…)	यह आरग्यूमेन्ट स्ट्रींगको concatenate करके प्राप्त स्ट्रींग कोreturn के तौर परदेता है।
FIELD(Str,Str1,Str2,…)	यहStrका इन्डेक्स दी हुई लिस्ट(Str1,Str2,…) में से रिर्टन करता है। अगरStrनहीं मिलती तो 0 रिर्टन करता है।
LOAD_FILE(file_name)	यह फंक्शन फाइल को readकरके उसके कन्टेन्टस को स्ट्रीग रूप में देता है। इसकेउपयोग के लिए serverपर उपस्थित फाईल का पूर्णpath name उल्लेखित करना होता है।
REPLACE(Str, from_Str, to_Str)	यहStrस्ट्रींग को , उसमें सेfrom_Str की सारीoccurrences को to_Strसेreplaceकर के रिर्टन करता है।

उक्त फंक्शन के अलावा की अन्य कई स्ट्रींग फंक्शन MySQLमें हैं ।

उदाहरणः– (1)

MySQL> SELECT ASCII('3')

Output ->

ASCII('3')
51

उदाहरणः– (2)

MySQL> SELECT BIN(2)

Output -> 1 0

उदाहरणः– (3)

MySQL> SELECT BIT_LENGTH('BHARAT')

Output ->

BIT_LENGTH('BHARAT')
48

उदाहरणः– (4)

MySQL> SELECT CONCAT('BH','A','GAT',' ','SI','NGH')

Output -> BHAGAT SINGH

उदाहरणः– (5)

MySQL> UPDATE Student SET Address= LOAD_FILE('pathname') WHERE Roll_No=105 ;

उदाहरणः– (6)

MySQL> SELECT CONCAT(Roll_No, Name, Class) FROM Student

Output ->

CONCAT(Roll_No, Name, Class)
105hari9th
50gopal9th
60gopi8th
109kailash11th

उदाहरणः– (7)

MySQL> SELECT CHAR(66,72,65,82,65,84)

Output->

CHAR(66,72,65,82,65,84)
BHARAT

RANDफंक्शन :– MySQLमें0व1 के बीच में कोई भी नम्बरrandomly निकालने के लिए हमRANDफंक्शनको उपयोग करते है।

उदाहरण :–

MySQL > SELECT RAND(), RAND();

Output-> RAND() RAND()

0.03014567845357
0.93969467893221

SQRTफंक्शनः– यह फंक्शनकिसी नम्बर कासquare root निकालने के काम आता है।

उदाहरण :–

MySQL> SELECT SQRT(64)

Output ->

SQRT
8

अगर हमTeacher टेबिल में Salary field कासquare root निकालना है तो हम उसे निम्न प्रकार उपयोग करेंगें।

MySQL>SELECT Tname, SORT(Salary)FROM Teacher

Output->

Tname	Salary
Radha Krishnan	54.7722557505166
Rajesh	70.7106781188548

Numericफंक्शनः–इस फंक्शनको उपयोग गणितीय आपरेशन में काम में लेते है। कुछ उपयोगी फंक्शननिम्नलिखितहै।

फंक्शन	विवरण
(i)ABS(V)	यह फंक्शनV की पूर्ण वेल्यू देता है।
(ii)GREATEST(n1,n2,…)	यह फंक्शनदी हुई पैरामीटर लिस्ट में से अधिकतम(greatest)वेल्यू देता है।
(iii)INTERVAL(N,n1,n2,n3,----)	यह फंक्शनN की वेल्यू को लिस्ट(,n1,n2,n3,----) से कम्पेयर करता है। औरअगरN<n1है तो0। N<n2है तो 1। N<n3है तो 2 और इसी प्रकार आगे नम्बर रिर्टन करता है।
(iv)LEAST(N1,N2….)	यह GREATESTका विपरीत है।

उदाहरण :–(1) MySQL> SELECT ABS(-6);

Output ->

ABS(-6)
6

उदाहरण :–(2) MySQL> SELECT GREATEST(4,3,7,9,8,0,10,50,70,11)

Output ->

GREATEST(4,3,7,9,8,0,10,50,70,11)
70

उदाहरण :–(3) MySQL> SELECT INTERVAL(4,3,5,8,11,12,17 ,18)

Output ->

INTERVAL(4,3,5,8,11,12,17,18)
1

Aggregate फंक्शनः– MySQLमेAggregateफंक्शनइनपुट के तौर पे वेल्यूस का संग्रहलेते हैऔर आउट पुट में एक वेल्यू देते है।MySQLमेंनिम्न 5 प्रकार केbuilt in Aggregateफंक्शनहोते है।

AVERAGE: Avg()

MAXIMUM: Max()

MINIMUM: Min()

TOTAL: Sum()

COUNT: Count()

यहाँSum एवंम्Averageफंक्शनकी इनपुट वेल्यूस आवश्यक रूप से नम्बर होने चाहिए।जबकि दूसरे आपरेटर्स ,स्ट्रींग के उपर भी काम कर सकते है।

Avg() function फंक्शन :–यह फंक्शनटेबिल के किसी फिल्ड की वेल्यूस का औसत निकालने के उपयोग में आता है।

उदाहरण :–

:- MySQL> SELECT Avg(Salary)FROM Teacher;

Output->

Avg(Salary)
14250.0000

ऊपर दिये उदाहरण मेंAverageफंक्शनSalaryफिल्ड में वेल्यूस का औसत रिर्टन करता है।

Sum() Function फंक्शन :– यह फंक्शनकिसी फिल्ड की सभी वेल्यूस काSumदेता हैं।

उदाहरण :–

MySQL> SELECT Sum(Salary)FROM Teacher;

OUTPUT->

Sum(Salary)
57000

Max() functionफंक्शन :–यह फंक्शनवह रिकार्ड जो किसी रिकार्ड सैट में अधिकतम है को देता है।

उदाहरण :–

MySQL> SELECT Max(Salary) FROM Teacher

Output->

Max(Salary)
40000

Min() functionफंक्शन :–यह फंक्शन निम्नतम वेल्यू वाला रिकार्ड देता है।

उदाहरण :–

MySQL> SELECT Min(Salary) FROM Teacher

Output->

Min(Salary)
3000

Count() functionफंक्शन :–यह फंक्शन टेबिल में रिकार्ड की संख्या गणना के लिए काम में आता है। अर्थात रिकार्डसकी कुल संख्या पता करने के काम आता हैं(counting the number of records)

उदाहरण :–

SELECT Count (*) FROM Student

Output ->

Count(*)
4

उदाहरण :–

SELECT Count (*) FROM Student WHERE Class=”9th”

Output ->

Count(*)
2

SQL ORDER BY क्लॉजः–SQLकेSELECT statementके द्वारा चुनी गई पक्तियों का क्रम कुछ भी हो सकताहै। अगर इन पक्तियों को हमे किसी क्रम में देखना है तो हमेंSQL के ORDER BYक्लॉज का उपयोग उस स्तंभ यास्तंभों के साथ करते है। जिनकों हम क्रय में देखना चाहते है। अर्थातORDER BY के द्वारा हम किसी स्तंभ की वेल्यूस कोबढ़ते(ascending)या घटते(descending)क्रम में देख सकते हैं ।

Syntax:-

SELECT field1,field2….filedn

FROM T1,T2…Tn

ORDER BY field1, field2….filedn [Asc[Desc]];

यहाँ हम क्यूरी के रिजल्ट को एक से ज्यादा फिल्ड पर SQRTकर सकते है। जिसके लिएAscयाDesc है।

उदाहरण :–

MySQL> SELECT Roll_NO, Age FROM Student ORDER BY Age Desc;

Output:-

Roll_No	Age
109	19
105	14
50	13
60	13

उदाहरण :–

MySQL> SELECT Roll_NO, Age FROM Student ORDER BY Age Desc, Roll_No Desc;

Output:-

RollNo	Age
109	19
105	14
60	13
50	13

उदाहरण :–

Query :-उन छात्रों का नाम बताइये जोClassroomनम्बरF-17 में बैठता है।

MySQL> SELECT Name FROM Student, Classes WHERE Class=Class_name AND CRoomNo =”F-17”ORDER BY Name Asc;

इस उदाहरण में हमने विभिन्न टेबिल का उपयोग किया है।क्योंकि हमें विभिन्न टेबिल से सूचनाओं(information)कीआवश्यकता हैं। जैसें कि छात्र का नाम हमStudentटेबिल से प्राप्त होगा जबकि Classroom फिल्डClassesटेबिल मेंहै। अतः हमें दोनों टेबिल को जोड़ते (joine)हुये आवश्यक क्यूरी लिखनी है।इस क्यूरी केWHEREक्लाज में हमने दोनों टेबिल को उनकीprimary keyवforeign keyके द्वारा जोड़ा हैं क्योंकियह दोनों टेबिल एक समान

है। अर्थात दोंनों टेबिल इनके द्वारा लिक्ड़ है। किसी क्यूरी में हम दो या दो से अधिक टेबिल को कॉमन फिल्ड के द्वारा जोड़ते है।

यहाँ हमने SQL ORDERक्लाज का भी उपयोग किया है। क्योंकि हम रिजल्ट को बढ़ते क्रम में चाहते है। अर्थात छात्रों का नाम इनके ऐलफाबेटिकल आर्डर में प्राप्त हो ।

Output->केवल वे छात्र जो F-17 में बैठते है।

Name
--

SQL GROUP BY क्लॉजः–MySQLका यह क्लॉज अत्यन्त उपयोगी इस क्लॉज के उपयोग द्वारा कई महत्वपूर्ण क्यूरीलिखी जा सकती है। GROUP BYक्लास के द्वारा किसी स्तंभ(column)या स्तंभों या ऐट्रीब्सूटस की वैल्यूस का समूहबनाया जा सकता है। अर्थात क्लॉज में दिय गये ऐट्रीब्यूट का उपयोग समूह(GROUP)बनाने के लिए करते है। GROUP BYक्लॉज में दिये गये एट्रीब्यूट या एट्रीब्यूटस की वेल्यूज जिन टप्लस या पक्तियों की लिए एक समान है। वेसभी टप्लस या पक्तियों एक समूह में आयेगी।

GROUP BYक्लॉज को हम निम्न उदाहरण द्वारा समझ सकते है। इसके लिए Studentटेबिल को लेतें है। जिसमेंकिसी क्षण निम्न रिकार्डस है।

Roll_No	Name	Age	Address	Class
1	Ajay	9	Jaipur	4th
2	Vijay	17	Kota	12th
10	Hari	11	Udaipur	7th
17	Shanker	13	Jaipur	8th
21	Om	21	Ajmer	12th
51	Mayank	15	Ajmer	9th
90	Anju	18	Ajmer	11th
53	Suman	12	Ajmer	10th
64	Kamal	10	Kota	4th
500	Komal	16	Udaipur	9th
700	Aryabhatt	11	Jaipur	7th
900	Bodhayan	13	Jodhpur	8th

Figure 5 Student table

उदाहरण :–

क्योरीः–प्रत्येक Classमें पढ़ने वाले छात्रों की संख्या बताइयें।

अगर हम यह क्योरी निम्न प्रकार से लिखगें तो परिणाम गलत प्राप्त होगा।

MySQL> SELECT Count(*) FROM Student

Output->

Count (*)
12

क्योंकि इस Syntaxके द्वारा कुल पढ़नें वालो छात्रों की संख्या प्राप्त होगी अर्थात Studentटेबिल में जितने छात्रों का रिकार्ड उपलब्ध है। वही उस स्कूल में पढ़नें वाले छात्र है। अतः यह Syntax,Studentटेबिल में कुल टपल्स कितने है। उनकी संख्या देगा।सही परिणाम के लिए Count aggregateफंक्शन केGROUP BYसाथ क्लाज का उपयोग करना पड़ता है। जिसकाSyntaxनिम्नानुसार है।

MySQL> SELECT Class, Count(Roll_No) FROM Student GROUP BY Class;

Studentटेबिल में समूह बनने के बाद टेबिल के बादStudentटेबिल कुछ इस प्रकार दिखेगी क्योंकि GROUP BYक्लाज में Classऐट्रीब्यूट के द्वारा समह बनाया जा रहा है। अतः एक समान Classवाली पक्तियाँ एक समूह में दिखेगी।

Output ->

Class	Roll_No	Age	Name	Address
11th	90	18	Anju	Ajmer
12th	2	17	Vijay	Kota
12th	21	21	Om	Ajmer
10th	53	12	Suman	Ajmer
9th	51	15	Mayank	Ajmer
9th	500	16	Komal	Udaipur
8th	17	13	Shanker	Jaipur
8th	900	13	Bhodhayan	Jodhpur
7th	10	11	Hari	Udaipur

7th	700	11	AryaBhatta	Jaipur
4th	1	9	Ajay	Jaipur
4th	64	10	Komal	Kota

Class	Count(Roll_No)
10th	1
11th	1
12th	2
9th	2
8th	2
7th	2
4th	2

जिस columnके द्वाराGROUPबनाया जाता है। उसcolumnपर कोईcalculation Count, Avg, Max, Min आदि aggregate functionके द्वारा कि जा सकती है। अतः इस क्यूरी मेंCount ,aggregate फंक्शन को हर समूहके टपल्स जिनकीClass एक समान है कि लिए लगाया गया है। क्योंकि SELECTक्लॉज मे दो ही फिल्ड़ है। अतःपरिणाम निम्नानुसार प्राप्त होगा ।

Class	Count(Roll_No)
10th	1
11th	1
12th	2
9th	2
8th	2
7th	2
4th	2

नोटः– किसीSELECTक्लॉज में aggregateफंक्शन के बाहर जो भी एट्रीब्यूटस आतें है। वे GROUP BYक्लास केअन्दर भी लिखना है। जैसेSELECTक्लॉज में Classएट्रीब्यूट आने पर वह GROUP BYक्लॉज में भी आया है।

उदाहरण :–

क्योरी:- उनClassके नाम बताइये (प्रत्येकClass)जिनमें पढ़ने वाले छात्रों की संख्या 1 से अधिक है।

Syntax:-

MySQL> SELECT Class, Count(Roll_No)FROM StudentGROUP BY ClassHAVING Count(Roll_No)>1;

Output->

Class	Count(Roll_No)
12th	2
9th	2
8th	2
7th	2
4th	2

यहाँ रिजल्ट में प्रत्येक GROUPसे वही टपल्सSELECTहुये है जिनका Count1 से अधिक है। SQLमें एक किसीटपल्स के लिए कोई शर्त पूर्ण होती है या नहीं यह देखने के लिए WHEREक्लॉज का उपयोग करते है। जबकि GROUP BY क्लॉज के द्वारा बनाये गये समूहों में उपस्थित टपल्स के लिए शर्त को देखने(test)के लिएHAVING क्लॉज काउपयोग करते हैं।WHEREक्लॉज एवंमHAVINGक्लॉज में यह मुख्य अन्तर है।

SQLमेंHAVINGक्लाज में दर्शाये गये predicateको लागूGROUP BYक्लाज के द्वाराGROUPबनाने के बादकरते है। इसलिए इसके साथaggregateफंक्शन भी उपयोग कर सकते है।

नोट :– अगर किसी क्योरी मेंWHERE, HAVING, GROUP BY आते है तो सबसे पहलेWHEREक्लॉज मेंलागू होगा उसके बाद जिन टपल्स के लिए शर्त पूर्ण होगी वे GROUP BY क्लॉज के द्वारा एक समूहों में रखेगें औरअन्त में हर समूह के लिएHAVINGक्लॉज को लागू करेंगें जिन समूहों के लिएHAVINGक्लॉज संतोशजनक नहींरहता वे समूह परिणाम में से हट जाते है।

SQL RENAME आपरेंशन :– SQLमें इस तरह की व्यवस्था है कि हम किसी भी रिलेशन का और ऐट्रीब्यूट का नामबदल सकते है। इसके लिए SQLमे'AS'क्लॉज के उपयोग निम्न प्रकार से करते है।

Old_relation/ attribute_name as new_name

'AS' क्लॉज का उपयोगSELECTव FROMदोनों क्लोजों में किया जा सकता है।

उदाहरण :–

MySQL> SELECT Avg(Salary) AS AvSalary FROM Teacher;

AvSalary
14250.0000

उदाहरण :–

MySQL> SELECT Class, Count(Roll_No) AS totalFROM Student GROUP BY Class;

Output->

Class	total
10th	1
11th	1
12th	2
9th	2
8th	2
7th	2
4th	2

SQL JOIN:–SQLकेJOINkeywordका उपयोग दो या दो से अधिक टेबिलस से डाटा की क्यूरी करने के कामलिए होता है। आपरेशन दो रिलेशन को इनपुट के तौर पर लेते है। और एक रिलेशन आउट पुट के तौर पर देते है। SQLमें दो रिलेशन कोJOINकरने के कई तंत्र(mechanisms)है।

जैसं कि (1) Cartesian product mechanism (2) Inner join (3) Outer join (left, right, full)

ऊपर दिये प्रत्येक join type के लिए एकJoin condition भी जुड़ी हुई होती है। अतः एकJoin expression इन दोनों(join typeऔरJoin expression)से मिलकर बनती है।जिसे हमFROM क्लॉज में उपयोग करते है।JOINको समझने के लिए हम Studentटेबिल नम्बर 5 एवंमClassटेबिल नम्बर 6 जिसमें किसी क्षण निम्न रिकार्ड को लेते है।

Class_name	CRoomNo	CStrength
12th	F-1	90
11th	F-2	75
10th	F-5	99
9th	S-21	110
8th	S-10	70
7th	F-10	85
6th	F-17	65
5th	F-7	60
4th	F-9	55
3th	F-8	50
2th	S-15	35
1th	S-9	60

Studentएवंम् Classटेबिल को JOINकरने के लिए एक क्योरी लिखते है।

MySQL> SELECT Roll_No, Class, CStrengthFROM Student AS St, Classes AS S,WHERE St.Class=S.Class_name;

इस क्योरी मेंFROMक्लॉज मेंStudent टेबिल को RENAMEकर केStएवंम्Classको Sकिया गया है। इनदोनों रिलेशनका Cartesian product होगा जिसमें Stटेबिल के हर टपल्स काSटेबिल के हर टपल्स से JOINहोगा। अतः प्राप्त रिलेशन में कुल टपल्स होगें।

N1 * N2 =12*12=144

यहाँN1 Stटेबिल में टपल्स की संख्या व N2 Sटेबिल में टपल्स की संख्या है।यहाँ पर रिजल्ट टेबिल में टपल्स प्राप्त होगें वो WHEREक्लॉज की शर्तें को पूर्ण करने वाले ही होंगें।

OUTER JOIN आपरेंशन :– निम्न प्रकार के होते है।

(1) LEFT OUTER JOIN (2) RIGHT OUTER JOIN (3) FULL OUTER JOIN

Outer Joinsके साथ निम्नjoin condition का उपयोग करते है।

1) Natural

2) ON (Predicate)

3) Using (A1,A2,…….An)

Left outer join और ON Join condition:–Left join को समझने के लिए निम्न दो टेबिल लेते है।

Classes

Class_name	CRoomNo	CStrength
12th	F-11	90
9th	S-21	110
7th	F-10	85
4th	F-9	55

Admission

Class_name	Roll_No	admission_date
12th	79	2000/7/15
9th	89	2010/8/13
6th	69	2012/7/21
5th	49	2013/7/03

Syntax:- Select Classes, Class_Name, Roll_No, CStrengthFrom Classes Left Outer Join Admission on Classes.class_Name=Admission.Class_Name

यहाँ रिलेशन का नाम उसके ऐट्रीब्यूट के साथ लिखते है। क्योंकि एक समान नाम का ऐट्रींब्यूट एक से ज्यादा रिलेशन में है।अतः अस्पश्टता(ambiguity)को दूर करने के लिए ऐसा किया है।

Output

Class_name	Roll_No	CStrength
12th	79	90
9th	89	110
7th	Null	85
4th	Null	55

Left Outer Join के रिजल्ट में दोनों रिलेशन केMatching tuplesतथाLeftवाले रिलेशन(classes)केunmatched tuples उपस्थित होते है।

Right Outer Joinऔरon Join Condition:–

Syntax:

Classes Right Outer Join admission on classes.Class_Name=Admission.Class_Name

Output

Class_Name	CRoomNo	CStength	Class_Name	Roll_No	Admission_date
12th	F-11	90	12th	79	2000/7/15
9th	S-12	110	9th	89	2010/8/13
Null	Null	Null	6th	69	2012/7/21
Null	Null	Null	5th	49	2013/7/03

Right Outer Join आपरेशन Left Outer Join के समान ही है। किन्तु इसमें Join operation के दाँयी (right)और वाले रिलेशन के unmatched tuples भी आते है। leftरिलशन के एट्रीब्यूटस के लिए वेल्यूस Nullरखेगें।

Full outer join और On condition:–

Syntax:

Classes full outer join admission on classes. Class_Name= Admission.Class_Name

Class_Name	CRoomNo	CStrength	Class_Name	Roll_No	Admission_date
12th	F-11	90	12th	79	2000/7/15
9th	S-12	110	9th	89	2010/8/13
7th	F-10	85	Null	Null	Null
4th	F-9	55	Null	Null	Null
Null	Null	Null	6th	69	2012/7/21
Null	Null	Null	5th	49	2013/7/03

यहाँ दोनों रिलेशन केunmatchedटपल्स भी आयेगें। साथ में दूसरे रिलशन केunmatched टपल्स के लिए Nullआयेगा।

Outer Join और Natural condition:–दो रिलेश्न का NaturalJoinकरने पर उन टपल्स की संख्या उन रिलेशनमें उपस्थित एक समान(common)ऐट्रीब्यूट के द्वारा प्राप्त होते है औरcommonट्रीब्यूटसresultरिलेशन में एक बार आते है। वह क्रम में सबसे पहले आते है।

उदाहरण :–Classes Natural right outer join admission

Class_Name	CRoomNo	CStength	Roll_No	Admission_date
12th	F-11	90	79	2000/7/15
9th	S-12	110	89	2010/8/13
6th	Null	Null	69	2012/7/21
5th	Null	Null	49	2013/7/03

अन्य outer joinभी Natural conditionशर्त के लिए ऊपर दिये अनुसार हम प्राप्त कर सकते है।

Inner join :–

उदाहरण :–

Classes inner Join Admission OnClasses.Class_Name= Admission.Class_Name;

Output->

Class_Name	CRoomNo	CStength	Class_Name	Roll_No	Admission_date
12th	F-11	90	12th	79	2000/7/15
9th	S-12	110	9th	89	2010/8/13

Inner join और natural condition:–

उदाहरण :–

Classes Natural Inner-Join Admission

Output ->

Class_Name	CRoomNo	CStength	Roll_No	Admission_date
12th	F-11	90	79	2000/7/15
9th	S-12	110	89	2010/8/13

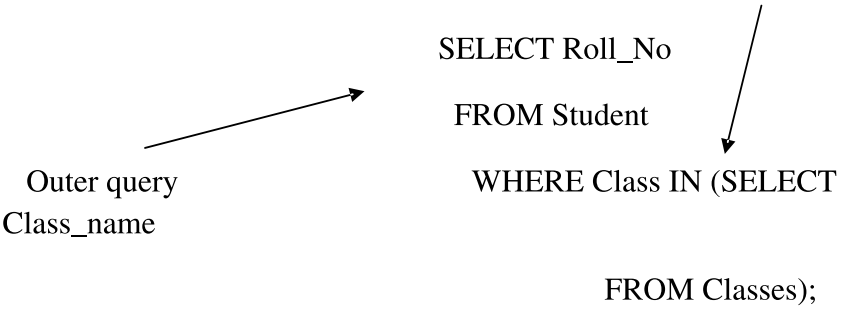
यहाँ केवल एक ही ऐट्रीब्यूट दोनों रिलेशन में समान है। अतः Joinकेवल उस ही ऐट्रीब्यूट के द्वारा होगा।

नोटः– Join condition USINGभी Natural Join के समान ही केवल joiningऐट्रीब्यूट USINGमें (A1,A2,…An) होते है। जबकि Naturalमें सारे ऐट्रीब्यूट जो दोनों रिलेशन में एक समान हैं होते है। यही Naturalएंवम् USING conditionका मुख्य अन्तर है।

SQL Sub queries:– एक Sub queriesइस प्रकार कीSQLक्योरी होती है जो किसी अन्य क्योरी के भीतर नेस्टेड(nested)होती है। इसके अलावाSub queries खुद भी अन्य Sub queriesके भीतर नेस्टेड(nested)हो सकतीहै।Sub queries को inner query क्योरी भी कहते है तथा जिसे क्योरी के भीतर Sub queriesहोती है उसे Outerक्योरी (बाहरी क्योरी) कहते है।

उदाहरण :–

Inner query



Sub queriesके द्वारा दी गई एक वेल्यू को कम्पेयर करने के लिए कम्पेरिशन आपरेटर (=,>=,<= etc)आदि का उपयोगकर सकते है।

उदाहरण के लिएTeacherटेबिल का उपयोग करते है।

SELECT Tname, SalaryFROM TeacherWHERE Salary= (SELECT Max(Salary) FROM Teacher);

Output ->

Tname	Salary
Hariom	40000

उदाहरण :–

उन Teacherके नाम बताइये जिनकी Salary सभीTeacherकी औसतSalaryसे कम है।

SELECT Tname, SalaryFROM TeacherWHERE Salary< (SELECT Avg(Salary) FROM Teacher);

Output ->

Tname	Salary
Radha krishnan	3000
Rajesh	5000
Lalaji	9000

महत्वपूर्ण बिंदु

- डाटा बेस स्कीमा किसी डाटा बेस की लॉजिकल डिजाईन है जो कि शायद ही बदलती है।
- किसी डाटा बेस में समय के किसी भी क्षण डाटा के समूह को डाटा बेस इन्सटेन्स कहते है।
- प्राइमेरी की :– किसी टेबिल में एक या अधिक फिल्डस (attribute) का ऐसा set जो कि उस टेबिल की किसी भी पर्क्ति अथवा टपल्स को uniquely identify करता है।
- रेफरेनशीयल इन्टीग्रीटी की सुनिश्चितता फोरेन की के द्वारा की जा सकती है।
- स्ट्रकचर क्यूरी लेग्वेज(SQL)रिलेशनल ऐलजेबरा एवंम् रिलेशनल केलकूलस के कोमबीनेशन का उपयोग करती है।
- AUTO_INCREMENTका प्रयोगफिल्ड में वेल्यू को एक से आगे बढाने के लिए किया जाता है।
- GRANT ALL Privileges ON *.* TO 'new_user @ 'localhost';

  यहाँasterisk (*) डाटाबेस व टेबिल को बताता हैं। यह कमाण्ड यूजर कोread, edit,execute और सभी आपरेशनस कीसहमती सभी डाटाबेस और टेबिल के लिए देता है।
- GROUP BYक्लॉज में दिये गये एट्रीब्यूट या एट्रीब्यूटस की वेल्यूज जिन टप्लस या पर्क्तियों की लिए एक समान है वेसभी टप्लस या पर्क्तियों एक समूह में आयेगी।
- SQLमें दो रिलेशन कोJOINकरने के कई तंत्र(mechanisms)है जैसे कि 1) Cartesian product mechanism (2) Inner join (3) Outer join (left, right, full)।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्नः

प्रश्न 1. जो एक SQL खंड नहीं है?

(अ) Select (ब) From (स) where (द) condition

प्रश्न 2. SQL का पूर्ण रूप है

(अ) Structure Question language (ब) syntax question language

(स) Structure query language (द) Structure question language

प्रश्न 3. DDL के लिए है।

(अ) डेटा डेफिनेशन लेंग्वेज (ब) डबल डेटा लेंग्वेज

(स) डेटा डेटा लेंग्वेज (द) इनमें से कोई नहीं

प्रश्न 4. Count() है एक

(अ) स्ट्रिंग फंक्शन (ब) सांख्यिक फंक्शन (स) दोनों (द) मौजूद नहीं

प्रश्न 5. Like ऑपरेटर के लिए उपयोग किया जाता है, जैसे

(अ) Concatenating strings (ब) count string character

(स) string matching (द) all

अतिलघुत्रात्मक प्रश्नः

प्रश्न 1. SQL क्या है?

प्रश्न 2. आप SQL from clause द्वारा क्या समझते हो?

प्रश्न 3. SQL select clause के महत्व क्या है?

प्रश्न 4. SQL के प्रकार का नाम दे।

प्रश्न 5. अद्वितीय और प्राथमिक बाधाओं के बीच क्या अंतर है?

प्रश्न 6. डेटाबेस instances.को परिभाषित करो।

प्रश्न 7. हम SQL में order by clause का उपयोग कैसे करते है।

प्रश्न 8. SQL में NULL क्या है?

प्रश्न 9. आप aggregate फंक्शन द्वारा क्या समझते हो ?

प्रश्न10. हम SQL grantकमाण्ड का उपयोग क्यों करते है।

लघुत्रात्मक प्रश्नः

प्रश्न 1. SQL की बुनियादी संरचना क्या है?

प्रश्न 2. विभिन्न DMLकमाण्ड क्या हैं उन के लिए syntaxes दे।

प्रश्न 3. फॉरेन key क्या है? हम किसी टेबिल में एक फॉरेन key कैसे बनाते है।?

प्रश्न 4. SQL के group by clause का उपयोगउदाहरण केद्वारा समझाओं।

प्रश्न 5. Cartesian join और natural join में क्या अंतर है।

निबंधात्मक प्रश्नः

प्रश्न 1. SQL joins, टेबिल्स के उपयुक्त उदाहरण लेने के साथ समझाओ।

प्रश्न 2. aggregate फंक्शन क्या हैं? हम aggregate फंक्शन का उपयोग कैसे करें? प्रत्येक का एक उदाहरण दे।

प्रश्न 3. किसी एकल SQL क्वेरी में where, group by और having clause के उपयोग की व्याख्या, करें ? एक उपयुक्त उदाहरण दे।

प्रश्न 4. दिए गए निम्न स्कीमा पर विचार करें।

students(Roll_no, Sname, age, phone, address, class)

Classes(Class_name, CRoomNo, CStrength)

और निम्न के लिए कोई SQL सिंटैक्स लिखें।

1) कमरा नंबर F-12 में बैठे 5वींclass के छात्रों का नाम का पता लगाए ।

2) अजमेर में रहने वाले 10वीं कक्षा के छात्रों की संख्या का पता लगाएं।

प्रश्न 5. उप क्वेरीज (Sub queries) द्वारा आपका क्या मतलब है? उप क्वेरीज क्यों उपयोगी होती हैं। सेट comparison में उप क्वेरी का उपयोग समझाओ।

उत्तरमाला

उत्तर 1: द उत्तर 2: स उत्तर 3: अ

उत्तर 4: ब उत्तर 5: स

अध्याय 15

PL/SQL के आधार बिन्दु

PL/SQL का अर्थ प्रासिजरल लैंग्वेज के एक्सटेशन से है। इसके महत्वपूर्ण बिन्दु SQL पर अधारित है। इसका निर्माण ऑरेकल कम्पनी ने किया है। जो इसके एक्सटेशन और इनहान्समेन्ट के लिये भी जवाब देय है।

साधारण PL/SQL ब्लॉक :–

यह मुख्यतः SQL और PL/SQL स्टेटमेन्टस से मिलकर बनता है। PL/SQL प्रोग्राम में भी इनका उपयोग होता है।

PL/SQL के तीन भाग :–

घोषणा (Declaration) भाग (गैर जरूरी)

लागुकरण (Execution) भाग (जरूरी)

अपवाद संचालन (Exception Handling) (और त्रुटि) भाग (गैर जरूरी)

PL/SQL Declaration भाग :– खंड का यह भाग जो कि गैर जरूरी है DECLAREनाम के keywordजो कि आरक्षित है से शुरू होता है। इस खंड का उद्देश्य constants, records, variables ओर cursors को घोषित करना है। उपरोक्त सभी विचारार्थ शब्द (प्लेसहोल्डर) Execution भाग में ऑकड़ों का हेरफेर करने के लिये उपयोग में लिये जाते है। प्लेसहोल्डर (constants, variable और records) ऑकड़ों को अस्थायी रूप से संग्रहित करते है।कर्सर भी इस खंड में घोषित किए गए हैं।

Execution भाग

PL/SQL (block) का यह भाग आरक्षित keywordBEGIN और END से शुरू और अन्त होता है। Execution भाग किसी कार्य को सफलतापूर्वक पूर्ण करने के लिये वहाँ जरूरी भाग है जहाँ प्रोग्राम का तथ्य लिखा जाता है। प्रोग्रामेटिकconstructs Loops, सशर्त कथन और SQL कथन Execution भाग के भाग है।

Exception भाग

PL/SQL खंड का यह भाग जो कि गैर जरूरी है आरक्षित keyword EXCEPTION से शुरू होता है। और प्रोग्राम में त्रुटियों के संचालन में उपयोग में लाया जाता हैं ।

खंड की बनावट (Structure of PL/SQL Block):–

```
DECLARE
        Variable declaration
BEGIN
      Program Execution
Exception
      Exception handling
END;
```

साधारण PL/SQL प्रोग्राम का खंड(block) जो '' भारत '' शब्द को प्रदर्शित करता है।

```
SQL> set serveroutput on
1 SQL> begin
2  dbms_output.put_line ('BHARAT');
3 end;
4 /
BHARAT
```

PL/SQL procedure सफलतापूर्वक समाप्त हुई ।

PL/SQL प्रोग्राम के कुछ महत्वपूर्ण बिन्दु इस प्रकार है।

- PL/SQL प्रोग्राम खंड (block) का भाग keyword Begin से शुरू होता है एवं keyword END से समाप्त किया जाता है। जो कि PL/SQL खंड का executable भाग कहलाता है।
- PL/SQL प्रोग्राम खंड कथनों से मिलकर बनता है। प्रत्येक कथन सेमीकॉलन (;) से समाप्त होता है।
- PL/SQL प्रोग्राम खंड के अन्त में फॉरवर्ड स्लैस (/) लगाया जाता है। जो कि प्रोग्राम खंड के कथनों के लागुकरण के लिये उत्तरदायी हैं ।

PL/SQL के लाभ :–PL/SQL खंड के कुछ लाभ इस प्रकार है।

PL/SQL खंड की बनावट :–PL/SQL में, प्रोग्राम के खंड एक दूसरे के साथ लिखे जा सकते हें जिसे नेस्टिंग कहते है। प्रत्येक खंड सम्पूर्ण कार्य के छोटे या बड़े अंश के लिये उत्तरदायी होता है। PL/SQL खंड डाटाबेस में संग्रहित किये जाते है। और पुनः उपयोग में भी लाये जाते है।

बेहतर प्रदर्शन :–PL/SQL इंजन एक साथ कई SQL कथनों का एक खंड की तरह संचालित करता है। जो बेहतर प्रर्दशन और कम नेटवर्क traffic के लिये उत्तरदायी है।

पद्धतिकरण (Procedural) भाषा की क्षमता :–PL/SQL को और शक्तिशाली बनाने हेतु कुछ procedural language कन्सट्रक्ट का उपयोग होता हैं जैसे (if else) और (फॉरलूप्स)

PL/SQL प्लेस होल्डरस :–

प्लेस होल्डरस ऑकड़ों की अस्थायी रूप से संग्रहित करने के लिये जगह देतें है। जो कि PL/SQL खंड के execution के दौरान हेरफेर करने के लिए उपयोग किए जाते हैं। वेरियबल्स कॉन्सटेन्ट और रिकार्डस PL/SQL प्लेसहोल्डरस हैं।

PL/SQL के प्लेसहोल्डरस :–

प्लेस होल्डरस को नाम और (datatype) से वर्णित किया जाता हैं यह परिभाषा उन आँकड़ों पर निर्भर करती है जो हम संग्रहित करते हैं कुछ आँकडों के प्रकार (datatype) जो प्लेस होल्डरस को वर्णित करते है निम्नलिखित है।

Number (n,m) , Char (n) , Varchar2 (n) , Date , Long , Long raw, Raw, Blob, Clob, Nclob and Bfile.

PL/SQL वेरियबल्स :– ये वो प्लेस होल्डरस है जो वेल्यूस को संग्रहित करते है जो PL/SQL खंड के जरिये बदली जा सकती है।

वेरियबल्स को घोषित(declare) करने की साघारण सिन्टेक्स :–

variable_name datatype [NOT NULL := value];

variable_name is the name of the variable.डाटाटाईप यहाँ एक उचित PL/SQL डाटाटाईप है। NOT NULL वेरियबल के लिय एक गैर जरूरी व्याख्या है।Value एवं डिफॉल्ट वेल्यू भी एक गैर जरूरी व्याख्या हैं जहाँ पर आप वेरियबलस को इनिशियलाइज करते है। प्रत्येक वरियबलस की घोषणा एक अलग वाक्य है जो कि सेमीकॉलन से समाप्त होता हैं उदाहरणतः यदि आप एक शिक्षक का वर्तमान वेतन संग्रहित करना चाहते है तो वेरियबल उपयोग में लायेंगे।

DECLARE

salary_variable number (6);

"salary_variable" एक वेरियबल है जिसके datatype नंबर है और जिसकी 6 लम्बाई हैं।

जब कभी वेरियबलस का ब्यौरा NOT NULL से दिया जाता है। उस वेरियबल को initialize करें जब उसे घोषित कर दिया जाता है।

उदाहरणतः निम्नलिखित उदारहण दो वेरियबल की घोषणा करता है जिनमें से एक NOT NULL है।

```
DECLARE
salary_variable number(4);
address varchar2(10) NOT NULL := “ajmer”;
```

कोई भी वेरियबल अपना Value PL/SQL खंड के execution और exception भाग में बदल सकता है जबकि वेरियबलस की Value क्रमशः दो प्रकार से दिया गया हो।

हम मान सीधे तौर पर चर (वेरियबल्स) के लिए असाइन कर सकते हैं। जिसका सिन्टेक्स है।

```
variable_name:= value;
```

डाटाबेस के स्तंभ से सीधे तौर पर मान वेरियबल्स के लिए SELECT.. INTO वाक्य से असाइन कर सकते हैं। जिसका सिन्टेक्स है।

```
SELECT Column_name INTO variable_name FROM table_name [where condition];
```

उदाहरण :– निम्नलिखित प्रोग्राम एक शिक्षक का salary एवं नाम बताता है। जहाँ शिक्षक का नाम“ radhakrishnan” है।

```
DECLARE
 salary_var number(6);
 tname_var char(15) = “radhakrishnan”;
BEGIN
 SELECT salary
 INTO salary_var
 FROM  teacher
 WHERE tname = tname_var;
 dbms_output.put_line(salary_var);
 dbms_output.put_line('The teacher '
        || tname_var || ' has  salary  ' || salary_var);
END;
/
```

टिप्पणी :– उपरोक्त प्रोग्राम के अन्त में फॉरवर्ड स्लैश (/) PL/SQL खंड के लागुकरण को प्रदर्शित करता हैं।

PL/SQL कॉन्सटेन्ट :–

यदि PL/SQL खंड में प्रयुंक्त Value पूरे प्रोग्राम में परिवर्तित न हो तो इस तरह का Value कॉन्सटेन्ट कहलाता हैं।

उदाहरणत :– यदि एक प्रोग्राम जो एक शिक्षक का वेतन 10 प्रतिशत बढ़ा दे आप इस पूरे प्रोग्राम में एक स्थिर मूल्यांकन घोषित कर सकते है। अगली बार यदि फिर से वेतन में वृद्धि करनी पड़े तो उस स्थिर मूल्य की परिवर्तित करने से लक्ष्य प्राप्ति होगी जो कि निसंदेह उस प्रक्रिया से सरल है। जहाँ आप को पूरे प्रोग्राम में Value बदलना पड़ें।

किसी कॉन्सटेन्ट या स्थिर मूल्यांकन की घोषणा

name_constant CONSTANT datatype =VALUE;

name_constant मुख्यतः कॉन्सटेन्ट का नाम है जो कि वेरियबल नाम के ही समान हैं ।

constant (reserved word) नामक शब्द किसी भी स्थिर वेल्यू की घोषणा हेतु उपयोग किया जाता है।

VALUE- यह वह मूल्य या मूल्यांकन है जो घोषित होते समय किसी भी कॉन्सटेन्ट को दिया जाता है। इसके पश्चात् आप मूल्यांकन नहीं कर सकते है।

उदाहरण :– increase_salary की घोषणा हेतु , निम्नलिखित प्रोग्राम आप लिख सकते है।

DECLARE

increase_salary CONSTANT number (4) := 10;

टिप्पणी :– घोषणा के समय कॉन्सटेन्ट का मूल्यांकन आवश्यक है। ऐसा नहीं करने पर प्रोग्राम के execution में error का होना संभव है।

PL/SQL SET Serveroutput ON :–

"SET Serveroutput ON" हमेशा PL/SQL शुरू करने से पूर्व लिखना चाहिये ।PL/SQL प्रोग्राम का execution ऑरेकल इंजन में होता है, इसलिये सर्वर आउटपुट को स्क्रीन पर प्रदर्शित करने हेतु उसकी आवश्यकता पड़ती है अन्यथा परिणाम प्रदर्शित नहीं होगा।

SQL>सर्वर आउटपुट के उपयोग को समझने के लिये हम एक उदाहरण लेते है। प्रोग्राम की पहली लाइन सर्वर आउटपुट को शुरू करती है। तत्पश्चात् वेरियबल्स और कॉन्सटेन्ट का वर्णन होता है। प्लेस होल्डर वर्णित करने के पश्चात् dbms_output.put_line नामक निर्देश का उपयोग वर्णित वेरियबल के नाम को छापने हेतु होता है।

उदाहरण :–

SQL> set serveroutput on

SQL> DECLARE

```
        Sno number(4) NOT NULL := 3
        Sname varchar2(14) := 'Hari';
        Sclass CONSTANT varchar2(10) := '9th';
BEGIN
        dbms_output.put_line('Declared Value:');
        dbms_output.put_line(' Student Number: ' || Sno || ' Student Name:
' || Sname);
        dbms_output.put_line('Constant Declared:');
        dbms_output.put_line(' student Class : ' || Sclass);
END;
/
```

परिणाम प्रदर्शित करने के लिये "set serveroutput on" नामक निर्देश का लागुकरण आवश्यक है।

उपरोक्त कोड़ का आउटपुट

```
Declared Value:
Student  Number: 3 Student Name:  Hari
Constant Declared:
student Class: 9th
```

PL/SQLके नियमबद्ध(Conditional) वाक्य :–

नियमबद्ध वाक्यों के प्रकार एवं रचना निम्न प्रकार है।

IF THEN ELSE वाक्य :– इस प्रकार के वाक्यों में जब अवस्था अनुकूल या TRUE होती है। तब 1 वाक्य का execution होता है, 2 वाक्य को छोड़ दिया जाता है परन्तु जब अवस्था FALSE होती है। तब 1 वाक्य छोड़ कर 2 वाक्य का execution होता है।

वाक्य रचना (Syntaxes)

```
1)
IF condition
THEN
 statement 1;
ELSE
```

```
 statement 2;
END IF;

2)
IF condition 1
THEN
 statement 1;
 statement 2;
ELSIF condtion2 THEN
 statement 3;
ELSE
 statement 4;
END IF
```

PL/SQL में Iterative वाले वाक्य :–

जब हम किसी वाक्य या वाक्यों का execution एक से अधिक बार करना चाहते है तो हम पुनरावृति नियंत्रक वाक्य (iterativecontrol statements) का उपयोग करते हैं।PL/SQL में तीन तरह के लूप होते है

लूप के प्रकार एवं रचना इस प्रकार है

- साधारण लूप
- व्हाइल लूप
- फॉर लूप

साधारण लूप की रचना :–

```
LOOP
  Statements;
  EXIT;
  {or EXIT WHEN condition;}
END LOOP;
```

साधारण लूप का उपयोग करते समय कुछ महत्वपूर्ण बिन्दु पर ध्यान देंवें।

- सदैव लूप के प्रधान भाग से पहले वेरियबल्स का मूल्यांकित(initialize) करें।
- लूप के भीतर वेरियबल के मूल्य में इजाफा जरूर करना चाहिये ।
- यदि लूप से बाहर निकलना चाहे तो EXIT WHEN वाक्य लिखे। यदि EXIT, WHEN के साथ नहीं लिखा जाता हैं तो लूप का execution केवल एक बार होता है।

व्हाइल लूप (While Loop) :–

While Loop के भीतर लिखें सारे वाक्यों का executionतब तक होता है जब तक conditiontrue है। अवस्था का मूल्यांकन प्रत्येक बार तब लूप चलता है के साथ होता है। और जब तक होता है जब तक conditionfalse नहीं होती हैं।

While Loop लूप का साधारण सिन्टेक्स :–

WHILE <condition>

LOOP statements;

END LOOP;

फॉर लूप (FOR LOOP) :–

FOR LOOP के भीतर वाक्यों का तब तक execution होता है जब तक लूप में लिखी संख्या पूरी नहीं हो जाती है। लूप का चलन प्रारम्भिक से अन्तिम तक (integer values given)होता है। काउन्टर के मूल्य में प्रत्येक बार एक संख्या से वृद्धि होना निश्चित है। अन्तिम मूल्य पर पहुँचने पर काउन्टर लूप से बाहर निकल जाता है।

फॉर लूप का साधारण सिन्टेक्स :–

FOR counter IN from….to

LOOP statements;

END LOOP;

- From - Start integer value.
- to - End integer value.

फॉर लूप के execution के लिये कुछ महत्वपूर्ण उपाय :–

- काउन्टर वेरियबल घोशणा खंड में ही घोषित किया जाता है। घोषणा खंड के बाहर घोषित करना अनावश्यक हैं।
- काउन्टर वेरियबल 1 के द्वारा incremented है अनावश्यक वृद्धि करने की जरूरत नहीं है।

- EXIT WHEN वाक्य और EXIT वाक्य फॉर लूप के भीतर उपयोग में लाये जा सकते है परन्तु यह अनुचित है।

PL/SQL कर्सर :–

जब कभी किसी SQL वाक्य का लागुकरण होता है। एक अस्थायी work area मशीन memory में बनता है। जिसे कर्सर कहा जाता है। आँकड़ों की पंक्तियों एवं सलेक्ट वाक्य से सम्बन्धित सूचनायें कर्सर में होती है।

यह अस्थायी कार्य क्षैत्र उन आँकड़ों को संग्रहित करता है जो डाटाबेस से ग्रहण किये जाते है। कर्सर एक से ज्यादा पंक्तियों पर नियंत्रण रख सकता है। परन्तु एक समय में एक पंक्ति का ही क्रियान्वन संभव है। वो सारी पक्तियों का समूह जिस पर कर्सर का नियंत्रण होता है क्रियाशील (active set) समूह कहलाता है।

PL/SQL में कर्सर के प्रकार :– PL/SQL में कर्सर दो प्रकार के होते है।

अंतर्निहित **(Implicit)**कर्सर :– यह पहले से बने होते है जब किसी DML वाक्य जैसे INSERT, UPDATE, और DELETE का लागुकरण किया जाता है। किसी SELECT वाक्य जो केवल एक पंक्ति परिणाम स्वरूप देता है के लिये भी कर्सर उपयोग में जाते हैं।

बाहानिहित **(Explicit)**कर्सर :– इनका निर्माण जरूरी है जब आप एक SELETE वाक्य का लागुकरण होता हैं एव ये वाक्य परिणाम स्वरूप दो पंक्तियों(more than one) प्रदर्शित करता है तो यद्यपि यह कर्सर एक से ज्यादा तथ्य संग्रहित करता है परन्तु एक समय में एक ही तथ्य का क्रियान्वन संभव है। जिसे वर्तमान कालीन (current) पंक्ति कहा जाता है। जब कभी कोई नयी पंक्ति आती है। तब पुरानी पंक्ति वर्तमान स्थिति से अगली स्थिति में आ जाती है। अंतर्निहित एवं बाहानिहित कर्सर की कार्य प्रणाली (functionality) एक समान होती है। परन्तु क्रियान्वन भिन्न है।

अंतर्निहित कर्सर के उदाहरण :–DML वाक्य जैसे INSERT,DELETE,UPDATE एवं SELECT के लागुकरण के लिये अंतर्निहित कर्सर उपयोग में लाये जाते है।

अंतर्निहित कर्सर के कुछ गुण (attributes)है जो DML की संचालन क्रिया (operations)की स्थिति बताते है। ये गुण %FOUND, %NOTFOUND, %ROWCOUNT, और %ISOPEN है।

उदाहरणत:INSERT, UPDATE, और DELETE वाक्यों के लागुकरण के समय कर्सर के गुण हमें ये बताते है कि इन वाक्यों कापंक्तियो पर प्रभाव हुआ है और कितनी पंक्तियाँ प्रभावित हुई है।जब कभी SELECT... INTO वाक्य का लागुकरण होता है तो अंतर्निहित कर्सर ये पत्ता लगाने के लिये उपयोगी हैं कि एक भी पंक्ति उपरोक्त वाक्य के द्वारा प्रदर्शित हुई है या नहीं।यदि कोई पंक्ति नहीं चुनी गई हैतोPL/SQL error देता है ।

%FOUND गुण :– प्राप्त परिणाम TRUE होगा यदि DML वाक्य जैसे INSERT,UPDATE, DELETE और SELECT... INTO कम से कम एक पंक्ति का संचालन करते है। परिणाम FALSE या अनुचित होगा यदि उपरोक्त वाक्यों के द्वारा एक भी पंक्ति का संचालन नहीं होता है।

%NOTFOUND गुण :– परिणाम असत्य या अनुचित होगा यदि DML वाक्य जैसे INSERT, UPDATE, DELETE और SELECT... INTO कम से कम एक पंक्ति का संचालन करें। यदि उपरोक्त वाक्यों से एक भी पंक्ति का संचालन न हो तो प्राप्त परिणाम उचित व सत्य होगा।

%ROWCOUNT गुणः–परिणाम स्वरूप उन सभी पंक्तियों की संख्या देता है जिनका संचालन INSERT,DELETE , UPDATE एवं SELECT के द्वारा हुआ है।

उदाहरणतः विचार करें कि एक PL/SQL खंड जो अंतर्निहित कर्सर के गुणों का उपयोग करता हैं जो कि निम्नाकिंत है।

```
DECLARE  rows_var number(7);
BEGIN
  UPDATE Teacher
  SET salary = salary + 100;
IF SQL%NOTFOUND THEN
   dbms_output.put_line(' salaries not  updated');
ELSIF SQL%FOUND THEN
   rows_var := SQL%ROWCOUNT;
   dbms_output.put_line('Salaries for ' || rows_var || 'teachers  are updated');
END IF;
END;
```

उपरोक्त PL/SQL खंड में टीचर नामक टेबल में उपस्थित सभी शिक्षकों के वेतन को UPDATE किया गया है। यदि किसी शिक्षक का वेतन UPDATE नहीं होता है। तो हमें त्रटि होने का सन्देश मिलता है। जो कि ' salaries not updated' हैं। अन्यथा हमें सन्देश मिलता हैं कि 'Salaries for 100 teachers are updated' यदि 'Teacher' टेबल में 100 पंक्तियों है।

Explicit कर्सर :–Explicit कर्सर का वर्णन PL/SQL खंड के घोषणा भाग में दिया जाता है इसका निमार्ण उस SELECT वाक्य के संदर्भ में किया जाता है जो परिणाम स्वरूप एक से अधिक पंक्तियों प्रदर्शित करता हैं कर्सर को इच्छानुसार उपयुक्त नाम दे सकते हैं।

कर्सर के निर्माण के लिये साधारण रचना :–

CURSOR cursor_name IS select_statement;

- cursor_name – कर्सर का नाम
- select_statement –एक सलेक्ट query जो एक या एक से अधिक पंक्तियाँ प्रदर्शित करे।

Explicit कर्सर के उपयोग हेतु बिन्दु :–

- घोषणा भाग में कर्सर को घोषित करे।
- Execution भाग में कर्सर को सक्रिय (OPEN)करें।
- Execution भाग में PL/SQL वेरियबल्स या तथ्यों में कर्सर से आँकड़ें लावें।
- PL/SQL खंड कि समाप्ति से पहले कर्सर को निश्क्रिय (CLOSE) करें।

संग्रहित कार्य पद्धति (Stored Procedures):– नामांकित PL/SQL खंड जो एक या एक से अधिक कार्यों का क्रियान्वन करें संग्रहित **Procedures** या साधारण **Procedures** कहलाता हैं। **Procedures** में एक हैडर व बॉडी होता है। जहाँ हैडर मे **Procedures** का नाम और पैरामीटर या वेरियबल्स का नाम होता है जो कि **Procedures** को पास किये जातें है। बॉडी में एक declaration भाग execution भाग एवं exception भाग होता हैं। जो कि साधारण PL/SQL खंड के समान ही है। परन्तु एक से अधिक बार उपयोग में लाने हेतु **Procedures** को नामांकित किया जाता है।

Procedures में पैरामीटर को पास करना :–तीन प्रकार से पैरामीटर को **Procedures** में पास कर सकते हैं।

- IN-parameters
- OUT-parameters
- IN OUT-parameters

Procedures के क्रियान्वन हेतु साधारण रचना :–

CREATE [OR REPLACE] PROCEDURE proce_name [parameter list]

IS

Declaration section

BEGIN

Execution section

EXCEPTION

Exception section

END;

IS :Procedures की बॉडी के प्रारम्भ को चिहिन्त करता है एवं PL/SQL के घोषणा भाग के समान ही हैं। IS एवं BEGIN के मध्य लिखा हुआ भाग ही घोषणा भाग कहलाता हैं।

जो भाग बड़े कोश्ठक [ ] में लिखा जाता है। गैर जरूरी होता है। हम CREATE या REPLACE को एक साथ तभी प्रयुक्त करें जब समान नाम से कोई और **Procedures** न हो या अस्तित्व में जो **Procedures** है वर्तमान कोड़ से बदली जाये।

उदाहरणत :– निम्नलिखित उदाहरण एक **Procedures** जिसका नाम 'student_info' है का निर्माण करता है। एवं इस **Procedures** का कार्य विद्यार्थियों की सूचनाएँ प्रदान करना है।

```
1> CREATE OR REPLACE PROCEDURE student_info
2> IS
3>  CURSOR stu_cur IS
4>  SELECT roll_no, Sname, age FROM Student;
5>  stu_rec stu_cur%rowtype;
6> BEGIN
7>  FOR stu_rec in 1..7
8>  LOOP
9>  dbms_output.put_line(stu_cur.Roll_no ll ' ' llstu_cur.Sname
10>    ll ' ' llstu_cur.age);
11> END LOOP;
12>END;
13> /
```

हम दो प्रकार से **Procedure** का लागुकरण कर सकते है।

(1) SQL prompt से

EXECUTE [or EXEC] procedure_name;

(2) दूसरी **Procedure** के अन्तर्गत –**Procedure** के नाम से

procedure_name;

PL/SQLफंक्शन :– यह एक तरह की नामांकित PL/SQL Block है जो कि **Procedure** के समान ही है। फंक्शन एवं **Procedure** में एक बड़ा अन्तर यह है कि फंक्शन में सदैव एक मान दिया जाता है पर **Procedure** में यह हो भी सकता है और नहीं भी।

फंक्शन की साधारण रचना :–

CREATE [OR REPLACE] FUNCTION func_name [parameters list]

RETURN return_datatype;

IS

Declaration_part

BEGIN

Execution_part

Return return_variable;

EXCEPTION

exception_part

Return return_variable;

END;

- Return Type हैडर वाला भाग प्रक्रिया के रिर्टन टाईप को वर्णित करता है। रिर्टन डाटा टाईप कोई भी उपयुक्त डाटा टाईप हो सकता है। जैसे varchar, number etc..
- Execution भाग एवं exception भाग हैडर में वर्णित डाटा टाईप का ही मान वापस देते है।

उदाहरण :– हम एक फंक्शन का निर्माण करतें हैं जिसका नाम "student_info_func' है।

1> CREATE OR REPLACE FUNCTION student_info_func

```
2>   RETURN VARCHAR(10);
3> IS
5>   stu_name VARCHAR(20);
6> BEGIN
7>      SELECT Sname INTO stu_name
8>      FROM student WHERE Roll_no = '58';
9>      RETURN stu_name;
10> END;
11> /
```

उपरोक्त उदाहरण में वो 'Sname' प्राप्त किये जिनका अनुक्रमांक 58 है और उन्हें 'stu_name' वेरियबल में संग्रहित किया। प्रक्रिया के रिर्टन टाईप का प्रकार VARCHAR है जिसकी घोषणा 2 नम्बर पंक्ति में की गई है। पूरी फंक्शन 'stu_name' पंक्ति नम्बर 9 मेंपरिणाम स्वरूप देती है। जिसके आँकड़े का प्रकार VARCHAR है।

Executing फंक्शन :–

- क्योंकि फंक्शन परिणाम स्वरूप एक मान देती है। यह वेरियबल को सौंप सकते है।

  student_name := student_info_func;

  यदि 'student_name' के आँकड़ों का प्रकार VARCHAR हो तो हम विद्यार्थी का नाम फंक्शन के रिर्टन टाईप को इसे सौंपकर संग्रहित कर सकते है।

- SELECT वाक्य के एक अंश की तरह

  SELECT student_info_func FROM student;

- PL/SQL के कुछ वाक्यों में

  dbms_output.put_line(student_info_func);

  यह पंक्ति फंक्शन द्वारा रिर्टन मान को प्रदर्शित करती है।

Exception Handling :–

PL/SQL की एक सुविधा यह हैं कि यह PL/SQL खंड में आये Exceptions कि Handling करता है, जिसे Exception Handling कहा जाता है। Exception Handling का उपयोग कर हम वर्तमान प्रोग्राम या कोड़ की जांच कर सकते है।

जब कभी कोई Exception आता है। Exception के कारण की व्याख्या का एक messages प्राप्त किया जाता हैं। PL/SQLExceptionmessages के तीन भाग होते है।

1) Type of Exception
2) An Error Code
3) A message

Exception Handling से यह निश्चित है कि PL/SQL खंड एकाएक समस्या का कारण नहीं बनता है।

अपवाद संचालन की संरचना

Exception भाग की साधारण रचना :–

```
DECLARE
  Declaration part
 BEGIN
  Exception part
 EXCEPTION
 WHEN excep1name THEN
   Error handling statements
 WHEN excep2name THEN
   Error handling statements
 WHEN Others THEN
  Error handling statements
END;
```

Exception खंड में PL/SQL वाक्य :–

जब कभी कोई Exception आता है। उपयुक्त exception handler की खोज की जाती है। उदारहरण उपरोक्त में यदि 'excep1name ' Exception आता है तो Exception Handling नीचे लिखी पंक्ति के अनुरूप होता है। बावजूद इसके यह संभव नहीं है कि कोड़ की जाँच के दौरान सारी त्रुटियों का आंकलन हो सकें। 'WHEN Others' Exceptions का प्रयोग उन Exceptions Handling के लिए किया जाये जिनका explicitly handling संभव न हो। केवल एक अपवाद एक खंड में संभव है और त्रुटि के संचालन के बाद नियंत्रण Execution भाग को वापिस नहीं दिया जाता है।

DELCARE

Declaration part

BEGIN

DECLARE

Declaration part

BEGIN

Execution part

EXCEPTION

Exception part

END;

EXCEPTION

Exception part

END;

यदि उपरोक्त उदाहरण में नैस्टेड़ PL/SQL खंड हो एवं यदि Exception आँतरिक खंड में हो तो उस Exceptions का निस्तारण PL/SQL के आँतरिक खंड के Exceptions खंड में होना चाहिये अन्यथा नियंत्रण अगले PL/SQL खंड के Exceptions खंड को दे दिया जाता है।

Exception के प्रकार :–तीन प्रकार के Exceptions होते है।

(1) Named System Exceptions

(2) Unnamed System Exceptions

(3) User-defined Exceptions

Named System Exceptions

तंत्र के Exceptionsautomatically ऑरेकल द्वारा चिन्हित किये जाते है यदि कोई प्रोग्राम RDBMS के नियम की पालना नहीं करता है और कुछ ऐसे अपवाद बार–बार विघ्न डालते है। जो कि पूर्व वर्णित एवं ऑरेकल में नामांकित होते है तो ऐसे अपवाद नामांकित तंत्र अपवाद कहलाते है।

उदाहरण :– NO_DATA_FOUND and ZERO_DIVIDENamed System Exceptions है।

Named System Exceptions

(1) बाहयनिहित (explicitly) घोषित नहीं होते है।

(2) अंतर्निहित(implicitly) रूप से जाग्रत होते है जब कभी ऑरेकल की पूर्व वर्णित त्रुटियों होती है।

(3) Exceptions Handling रूटीन में सामान्य नाम के हवालें से चिन्हित कियें जाते है।

उदाहरण :–विचार करिये यदि NO_DATA_FOUND Exceptions किसी कार्यपद्धति में जाग्रत होता है। हम Exceptions Handling हेतु निम्नांकित कोड लिख सकते है।

```
BEGIN
  Execution part
EXCEPTION
WHEN NO_DATA_FOUND THEN
 dbms_output.put_line (' Using SELECT...INTO did not get any row.');
 END;
```

Unnamed System Exceptions

वो सारे तंत्र **Exception** जिनके लिये ऑरेकल की तरफ से नाम नहीं दिया जाता है **Unnamed System Exceptions** कहलाते है। ये **Exceptions** नियमित रूप से जाग्रत नहीं होते हें इन **Exceptions** के साथ एक प्रोग्राम और एक सन्देश होता है।

Unnamed System Exceptions दो प्रकार से होता हैं ।

(1) WHEN OTHERS **Exception handler** का उपयोग कर या

(2) **Exception** कोड़ को नाम देकर **named System Exceptions** की तरह उपयोग करे। EXCEPTION_INIT एक पूर्वनिर्धारित Oracle त्रुटि संख्या उपयोग कर हम प्रोग्राम नामक **Exception** को नाम दे सकते है।

- **Unnamed System Exceptions** को उपयोग करते समय कुछ ध्यान रखने योग्य मुख्य बिन्दु इस प्रकार है।
- ये अंतर्निहित रूप से जाग्रत होते हैं।
- दूसरों के साथ संचालन न होने की स्थिति में ये बाहयनिहित रूप से संचालित होते हैं।
- **Exceptions** को बाहयनिहित रूप से संचालन हेतु उनकी घोषणा PragmaEXCEPTION_INIT

- के उपयोग से होवे एवं संचालन उपयोगकर्ता वर्णित **Exception**के नाम से **Exception** भाग में होना चाहियें।

EXCEPTION_INIT का उपयोग करते हुए Unnamed System Exceptions की घोषणा हेतु साधारण रचना :–

DECLARE

excep_name EXCEPTION;

PRAGMA

EXCEPTION_INIT (excep_name, Err_code);

BEGIN

Execution part

EXCEPTION

WHEN excep_name THEN

handle the exception

END;

User-defined Exceptions

System Exceptions के अलावा हम उन **Exceptions** को जो व्यवसायिक नियम पर आधारित है की बाहयनिहित रूप से वर्णित कर सकते है। और इन्हें user-defined exceptions कहते है। user-defined exceptions के उपयोग हेतु मुख्य बिन्दु :–

- इनकी घोषणा बाहयनिहित रूप से जाग्रत किया जावें ।
- user-defined exceptions के नाम का हवाला देते हुए exception खंड में संचालन किया जावें।

ट्रिगरस

परिभाषा :–PL/SQL खंड की संरचना है जिसका उपयोग DML वाक्य जैसे Insert, Delete, Update के डेटाबेस तालिका पर execution के समय किया जाता है। यह automatically प्रयुक्त होता है जब उपरोक्त DML वाक्यों का execution होता है। डाटाबेस ट्रिंगर के तीन अंश है।

(1) **ट्रिगरिंग इवेन्ट** :– ट्रिंगर के execution के लिये जवाबदेय

(2) **शर्त (Condition)** :– ट्रिंगर execution के लिये शर्त का पूरा होना जरूरी है।

(3) गतिविधि (**Action**) :– ट्रिंगर के execution के समय घटित गतिविधि का ब्यौरा

ट्रिंगर की रचना :–

```
CREATE [OR REPLACE ] TRIGGER name_of_trigger
 {BEFORE | AFTER | INSTEAD OF }
 {INSERT [OR] | UPDATE [OR] | DELETE}
 [OF name_of_col]
 ON table_name
[REFERENCING OLD AS O NEW AS N]
 [FOR EACH ROW]
WHEN (condition)
BEGIN
  --- sql statements --
END;
```

ट्रिंगर **Syntax:**

- CREATE [OR REPLACE] TRIGGER name_of_trigger – In PL/SQL में ट्रिंगर जिसका नाम दिया गया हो के निमार्ण हेतु या फिर ट्रिंगर जो अस्तित्व में है को ओवरराइट करने हेतु इस क्लॉस का उपयोग होता है।
- {BEFORE | AFTER | INSTEAD OF } यह क्लॉस दर्शाता है ट्रिंगर का उपयोग किस समय किया जाना चाहिये उदाहरणतः टेबिल के UPDATE के पहले और पश्चात् INSTEAD OF का उपयोग view के समय ट्रिंगर के निर्माण के लिये किया जाता है।Before और after का उपयोग अपमू के समय ट्रिंगर निर्माण में नहीं किया जा सकता हैं ।
- {INSERT [OR] | UPDATE [OR] | DELETE} यह क्लॉज ट्रिंगरिंग की घटना (event) को बताता है। एक से अधिक ट्रिंगरिंग की घटनाओं के एक साथ होने की स्थिति में घटनाओं को पृथक करनें हेतु keyword का उपयोग होता है। सारी ट्रिंगरिंग की घटनाओं के समय ट्रिगर का उपयोग होता हैं ।
- [OF name_of_col] यह क्लॉज UPDATE ट्रिंगर के साथ उपयोग में लाया जाता है। एक विशेष स्तंभ के जुड़ने पर यदि कोई घटना ट्रिंगर होती है। तो यह क्लॉज उपयोग किया जाता है।

- [ON table_name] यह क्लॉज टेबिल का नाम सत्यापित करता है। या फिर टेबिल का वह view जिसके साथ ट्रिंगर जोड़ा गया है।
- [REFERENCING OLD AS O NEW AS N] इस क्लॉज का उपयोग जो डाटा परिवर्तित हो रहा है पुराने और नए मान को संदर्भित करने के लिए उपयोग किया जाता है।डिफॉल्ट रूप से, आप मानों को old-column_name or :new-column_name के रूप में संदर्भ किया जाता है। संदर्भित OLD or NEW नामों को user-defined नामों से परिवर्तित कर सकतें है। पुराने मान तथ्यों केinserting के समय उल्लेखित नहीं किये जा सकते और नये मान तथ्यों के delete करते समय क्योंकि वो अस्तित्व में नहीं होते हैं।
- [FOR EACH ROW] इसका उपयोग यह बताने में किया जाता है कि ट्रिंगर का उपयोग जब प्रत्येक पंक्ति प्रभावित हो तब किया जावे या नहीं (i.e. a Row Level Trigger)या फिर केवल एक बार उस समय जब पूरे SQL वाक्य का execution हो।
- WHEN (कन्डीशन) यह केवल पंक्ति लेवल टिंगर के लिये उपयुक्त है। ट्रिंगर उन्हीं पंक्तियों के लिये चलते है जो दी हुई शर्तों को पूरी करती हैं।

उदाहरण :– स्टूडेन्ट classes सदैव परिवर्तित होती है। इसलिये students के class के इतिहास को संग्रहित करना आवश्यक है।

PL/SQL Trigger के प्रकार :–

दो प्रकार के ट्रिंगरस है जो मुख्यतः उस स्तर पर आधारित है जहाँ ट्रिंगर का उपयोग होता है।

(1) रो लेवल ट्रिंगर :–जब प्रत्येक रो आधारित प्रविष्टी हटायी या update की जाती है तो तब एक घटना ट्रिंगर होती है।

(2) स्टेटमेन्ट लेवल ट्रिंगर :– प्रत्येक SQL वाक्य के execution हेतु एक घटना टिंगर होती है।

PL/SQL ट्रिंगर execution Hierarchy :– जब ट्रिंगर उपयोग में आता है। तो निम्नलिखित हाइराइकी उपयोग में लायी जाती है।

(1) सबसे पहले जब BEFORE वाक्य ट्रिंगर उपयोग किया जाता है।

(2) उसके बाद जब BEFORE रो लेबल ट्रिंगर उपयोग मे आता है।

(3) उसके बाद जब AFTER रो लेबल ट्रिंगर उपयोग में आता है। यह घटना BEFORE और AFTER रो लेबल ट्रिंगर के मध्य अल्टरनेट देती है।

(4) सबसे अन्त में AFTER स्टेटमेन्ट ट्रिंगर उपयोग में आता है।

हम student table के लिये BEFORE और AFTER स्टेटमेन्ट और रो लेबल ट्रिंगर का निर्माण करते है।

महत्वपूर्ण बिंदु

- PL/SQL के तीन भाग : घोषणा भाग, लागुकरण भाग और अपवाद संचालन भाग होते है।
- "SET Serveroutput ON" हमेशा PL/SQL शुरू करने से पूर्व लिखना चाहिये।
- यदि EXIT, WHEN के साथ नहीं लिखा जाता हैं तो लूप का execution केवल एक बार होता है।
- कर्सर एक से ज्यादा पंक्तियों पर नियंत्रण रख सकता है। परन्तु एक समय में एक पंक्ति का ही क्रियान्वन संभव है। वो सारी पक्तियों का समूह जिस पर कर्सर का नियंत्रण होता है क्रियाशील (active set) समूह कहलाता है।
- फंक्शन एवं **Procedure** में एक बड़ा अन्तर यह है कि फंक्शन में सदैव एक मान दिया जाता है पर **Procedure** में यह हो भी सकता है और नहीं भी।
- ट्रिगर automatically प्रयुक्त होता है जब DML वाक्यों का execution होता है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्रश्न 1. जो एक PL/SQLका हिस्सा नहीं है

(अ) Declare (ब)BEGIN (स) Start (द) End

प्रश्न 2. PL/SQL द्वारा विकसित की है

(अ) IBM (ब) ORACLE (स) Microsoft (द) इनमे से कोइनही

प्रश्न 3. शब्द का चयन करें जो select कथन के साथ उपयोग किया जाना चाहिए।

(अ) Goto (ब) Into (स) Do (द) all

प्रश्न 4. कर्सर कितने प्रकार के है।

(अ) 2 (ब) 4 (स) 5 (द) 1

प्रश्न 5. % FOUND का काम विपरीत है।

(अ) %CURSOR (ब) % NOT COUNT

(स) %NOT FOUND (द) % FOUND COUNT

अतिलघुत्रात्मक प्रश्नः

प्रश्न 1. PL/SQL क्या है?

प्रश्न 2. PL/SQL ब्लॉक में कितने भाग हैं?

प्रश्न 3.PL/SQLमें Declare का उपयोग क्योंकरें ?

प्रश्न 4. PL/SQL में & का उपयोग क्या है।

प्रश्न 5.PL/SQL में Variables कहां घोषित किया जाता है?

प्रश्न 6. PL/SQL select कथन का प्रयोग कैसे किया जाता है ?

प्रश्न 7.exception ब्लॉक का क्या उपयोग है ?

प्रश्न 8. PL/SQL.में variable के प्रकार को बताए।

प्रश्न 9. ट्रिगर क्या है?

प्रश्न 10. हम ट्रिगर का उपयोग कैसे करें ?

लघुत्रात्मक प्रश्नः

प्रश्न 1.%TYPE गुणऔर %ROWTYPE गुण प्रकार के बीच क्या अंतर है?

प्रश्न 2. PL/SQL में EXIT कथन का उपयोग है?

प्रश्न 3.before ट्रिगर क्या है ?

प्रश्न 4.implicit और explicitकर्सर के बीच क्या अंतर है ?

प्रश्न 5.for Loop का सिंटैक्स लिखे।

निबंधात्मक प्रश्नः

प्रश्न 1. कर्सर क्या है? कर्सर का उपयोग क्या है? उदाहरण के साथ explicit कर्सर की व्याख्या करें।

प्रश्न 2. विभिन्न प्रकार के डेटाबेस ट्रिगर्स उदाहरण के साथ समझाइए।

प्रश्न 3.exceptions क्याहै? विभिन्न प्रकार की exceptions की व्याख्या करें।

प्रश्न 4. PL/SQL में लूप के विभिन्न प्रकार समझाइए।

प्रश्न 5 function क्या है? यह Procedure से अलग कैसे है? functions और Procedure के लिए सिंटैक्स समझाइए।

उत्तरमाला

उत्तर 1: स **उत्तर** 2: ब **उत्तर** 3: ब
उत्तर 4: अ **उत्तर** 5: स