Programming in 'C'

Introduction to the C Language

कंप्यूटर शब्द की उत्पति अर्ग्रेज़ी भाषा के 'कंप्यूट' शब्द से हुई है जिसका अर्थ है गणना करना| कम्पूटर का विकास गणितीय गणनाओं को हल करने के लिए किया गया है| कंप्यूटर का निर्माण **चार्ल्स बेबेज** ने किया| अतः इन्हें **कंप्यूटर का जनक या पिता** कहा जाता है| चार्ल्स बेबेज एक गणित के प्रोफेसर थे जिन्होंने पहली बार कंप्यूटर का निर्माण सन 1822 मे किया|

कैम्ब्रिज विश्वविद्यालय (University of Cambridge) ने सन 1960 मे एक कम्प्यूटर प्रोग्रामिंग लैंग्वेज को डेवलप किया। इस लैंग्वेज को उन्होने बेसिक कंबाइंड प्रोग्रामिंग लैंग्वेज (Basic Combined Programming Language – BCPL) नाम दिया। इस लैंग्वेज को आम बोल-चाल की भाषा मे बी (B) कहा जाने लगा। सन 1972 मे बेल्ल प्रयोगशाला (Bell Labs) में कम्प्यूटर वैज्ञानिक "डेनिश रिची" द्वारा '**सी (C)**' भाषा लैंग्वेज को डेवलप किया गया।

"C" का विकास अमेरिका में सन् 1972 में हुआ। AT &T Bell Laboratory के कम्प्यूटर वैज्ञानिक **डेनिस रिची** ने इस का विकास किया था। ''सी'' एक शक्तिशाली भाषा है जिसमें हम एप्लीकेशन सॉफ्टवेयर व सिस्टम सॉफ्टवेयर दोनों तरह के सॉफ्टवेयर बना सकते हैं। इसमें सामान्य अंग्रेजी शब्दों के माध्यम से प्रोग्राम बनाए जाते हैं, जो कि समझने व बनाने में आसान होते हैं। ''सी'' एक हाई लेवल Structured Programming Language भाषा हैं, यानी सूचनाओं के एक निश्चित क्रम में Program Run होता है।

सी (C) एक सामान्य उपयोग में आने वाली कम्प्यूटर की प्रोग्रामन भाषा है। इसका विकास **डेनिस रिची ने बेल्ल टेलीफोन प्रयोगशाला** में सन् **1972** में किया था जिसका उद्देश्य यूनिक्स संचालन तंत्र का निर्माण करना था।

सी (C) कम्प्यूटर की एक प्रोग्रामिंग भाषा है, इसका मुख्य उद्देश्य यूनिक्स संचालन तंत्र (UNIX Operating System) का निर्माण करना था। सी प्रोग्रामिंग भाषा (C Programming Language) को डॉस ऑपरेटिंग सिस्टम (DOS operating system) और यूनिक्स ऑपरेटिंग सिस्टम (Unix Operating System) दोनो मे ही प्रयोग किया जा सकता है। जितनी भी मॉडर्न प्रोग्रामिंग लैंग्वेज है, वे सभी 'सी' प्रोग्रामिंग भाषा में ही आधारित है।

कम्प्यूटर मुख्यतः एक ही भाषा यानी मशीनी भाषा (Zero 0 or One1) बाइनरी को ही समझता है। फिर भी मोटे तौर पर कम्प्यूटर भाषा को निम्नानुसार तीन भागों में बांटा गया है। ये High Level Languagè हैं, जिनमें एक ऐसा Software या Program होता है जो इन High Level Languagè के Program Codes को मशीनी भाषा के Low Level Codes में Convert करने का काम करता है, जिन्हें Computer समझता है।

सी प्रोग्रामिंग लैंग्वेज की विशेषताएं (Features of C programming language) –

''सी'' अन्य कई भाषाओं से काफी सरल है। अन्य हाई लेवल भाषाओं की तुलना में ''सी'' काफी लचीली भाषा है। ''सी'' ही एक ऐसी भाषा है, जिसमें कम्प्यूटर के हार्ड वेयर के साथ भी काम किया जा सकता है। इसके द्वारा मेमोरी मेनेजमेन्ट किया जा सकता है। सबसे बडी खासियत ''सी'' की पोर्टेबिलिटी है। यानी ''सी'' भाषा में लिखे गए प्रोग्राम किसी भी अन्य कम्प्यूटर वातावरण में चल सकते हैं। ''सी'' एक फंक्शनल भाषा है यानी इसमें सभी काम विभिन्न प्रकार के फंक्शनस् को यूज करके किया जाता है। ''सी'' में कोई इनपुट आउटपुट ऑपरेशन नहीं है। ''सी'' कम्पाइलर सभी इनपुट आउटपुट का काम लाइब्रेरी फंक्शन के द्वारा करता है।

1. सी प्रोग्रामिंग लैंग्वेज की मुख्य विशेषता यह है कि इसमे उच्च स्तरीय प्रोग्रामिंग भाषा (High level programming language) के साथ-साथ निम्न स्तरीय भाषा (Low-level language) के भी संपूर्ण गुण पाए जाते है।
2. 'सी' प्रोग्रामिंग भाषा प्रोग्रामर अपनी आवश्यकतानुसार नए फंक्शन्स (functions) परिभाषित कर इनका प्रयोग कर सकता है।
3. सी प्रोग्रामिंग लैंग्वेज में बनाये गए प्रोग्राम के एक्सेक्युट (execute) होने की गति तीव्र होती है।
4. सी प्रोग्रामिंग भाषा मुख्य रूप से गणित, विज्ञान एवं सिस्टम संबंधित कार्यो के लिए उपयोगी होती है।
5. 'सी' प्रोग्रामिंग भाषा में निर्देश (Command) देने के लिए छोटे अक्षर (lower case letters) का ही प्रयोग किया जाता है।

'सी' प्रोग्राम का स्ट्रक्चर (Structure of C Program)

सामान्यतः किसी भी प्रोग्राम के 6 भाग होते हैं।

1. Preprocessor Commands
2. Functions
3. Variables
4. Statements & Expressions (Calculation , decision making)
5. Comments
6. Output

Example of C Program for Windows one.c

```
#include<stdio.h>          /* Preprocessor Commands */
void main ()
{
        /* variable declaration */
        /* Statements */
        printf("\n Welcome In Uttarakhand Sanskrit University");      //function
}
```

1. **Pre-processor Commands** – ये कमांड्स प्रोग्राम की सबसे पहले लाइन में लिखे जाते हैं| इन कमांड्स को लिखने का सीधा सा अर्थ यह है कि ये कमांड्स C के कंपाइलर को निर्देश देते हैं कि कंपाइलर प्रोग्राम को कंपाइल करने से पहले प्रोग्राम के सोर्स कोड में इन फाइलों को भी शामिल कर ले| तो #include<stdio.h> का मतलब है कि प्रोग्राम को कंपाइल करने से पहले stdio.h नाम की हेडर फाइल को भी प्रोग्राम के कोड में शुरआत में जोड़ लें |
2. **Void main () { }:** कोई भी प्रोग्राम इसी { } ब्रैकेट के बीच में लिखा जाता है| प्रोग्राम का कोड एक्सेक्यूट होने की शुरुआत main function से ही होती है।
3. **Comments** – कमेंट्स वास्तव में प्रोग्राम का हिस्सा नहीं होते बल्कि हमें प्रोग्राम के बारे में बताते हैं। यानी जब हम प्रोग्राम लिख रहे होते हैं तो उसके बारे में कोई नोट लिखना चाहें तो कमैंट्स के माध्यम से लिख सकते हैं |
4. **Function** – function वास्तव में एक छोटा सा प्रोग्राम ही होता है जिसे लिखा कहीं और होता है और main प्रोग्राम से बुलाया या कॉल किया जाता है।
5. इसके बाद main function का ब्रैकेट बंद होता जो कि प्रोग्राम का अंतिम स्टेटमेंट होता हैं |

C की फाइल को ***.c*** एक्सटेंशन से सेव करें ***(*** जैसे ***one.c)*** |

Coding Structure of "C" Programs

Computer Program की परिभाषा देना चाहें तो ये कह सकते हैं कि **Computer Instructions** का एक ऐसा सुव्यवस्थित क्रम, जिससे Computer द्वारा किसी समस्या का उचित समाधान प्राप्त हो सके, **Program** कहलाता है।

सबसे पहले किसी प्रोग्राम की कोडिंग की जाती है। फिर प्रोग्राम को कम्पाइल किया जाता है। कम्पाइल करने से प्रोग्राम की हाई लेवल के कोड मशीनी भाषा के बाइनरी डिजिटस् में बदल जाते हैं, जिन्हें हमारा Computer समझ सकता है। सबसे पहले कम्प्यूटर चालू करेंगे और ''सी'' भाषा के कोडों को लिख कर प्रोग्राम बनाएंगे। इसे **Source Program** कहते हैं। प्रोग्राम बनाने के बाद इसकी किसी भी प्रकार की व्याकरण सम्बंधी गलती को Edit Source Program Block में Edit करके सही करते हैं।

''सी'' कम्पाइलर द्वारा प्रोग्राम को कम्पाइल करते हैं, जिससे प्रोग्राम को कम्प्यूटर अपनी मशीनी भाषा में समझ सके। यदि इस प्रोग्राम में कोई अन्य वाक्य रचना सम्बंधी गलती हो, तो प्रोग्राम कंट्रोल पुनः सभी गलतियों के साथ Source Editing के लिए उसी Edit Source Program Block में चला जाता है। जब प्रोग्राम में किसी भी प्रकार की कोई व्याकरण सम्बंधी गलती नहीं रह जाती है, तब Program Control उन **System Library Files** को प्रोग्राम में लिंक करता है, जिनके Function Program में Use हुए हैं।

जैसे Input/Output के सारे Functions **stdio.h** नाम की Header File में Store रहते हैं, इसलिए I/O की सुविधा प्राप्त करने के लिए इस Header File को हर C Program में Include किया जाता है। जब Program Control सभी आवश्यक Header Files को Program से Link कर देता है। फिर अगली

Stage में यूजर से Data Input करवाया जाता है व प्रोग्राम Execute होता रहता है। अब यदि किसी प्रकार की तार्किक गलती हो तो वह गलती अगले प्रोसेस बॉक्स में पकड में आती है। यदि गलती है, तो प्रोग्राम Control पुनः Edit Source Program Block में पहुंच जाता है, और सारी की सारी प्रक्रिया पुनः प्रोग्राम को डिबग करने में अपनाई जाती है। लेकिन यदि प्रोग्राम में कोई Error नहीं हो तो प्रोग्राम Correct Output देता है और समाप्त हो जाता है। इस तरह पूरा प्रोग्राम Step-By-Step Execute होता है।

main() Function Section :

यह फंक्शन हर "सी" प्रोग्राम में होता है। कम्पाईल करते समय Program Control हमेशा main() Function को ही ढूंढता है। **हर "सी" प्रोग्राम में सिर्फ एक ही main() Function हो सकता है** व हर "सी" प्रोग्राम में main() Function का होना जरूरी होता है क्योंकि Program का Execution हमेंशा main() Function से ही शुरू होता है।

```
main()
{
        Function Body ;
}
```

यह किसी भी प्रोग्राम का एक अनिवार्य हिस्सा है। जब भी कोई प्रोग्राम कम्पाइल करते हैं तो कम्पाइलर सर्वप्रथम main() Function को ढूंढता है और इसके मंझले कोष्ठक से प्रोग्राम का Execution शुरू करता है। सभी Executables Code इन्ही मंझले कोष्ठकों के बीच लिखे जाते हैं।

किसी भी Function की शुरूआत व अन्त के Statements इन्हीं मंझले कोष्ठकों के बीच लिखे जाते हैं, फिर चाहे ये User Defined Functions हों या main() Function, Program के हर Statement का अन्त " ; " सेमीकॉलन के चिन्ह द्वारा ही होता है।

Compiler and Interpreter

Compiler व Interpreter भी High Level Program Codes को मशीनी भाषा में बदलने का काम करते हैं लेकिन दोनों के काम करने के तरीके में कुछ अन्तर हैं। Compiler पूरे प्रोग्राम को एक ही बार में मशीनी भाषा में बदल देता है व सभी Errors को Debug करने के बाद एक Executable Program File Provide करता है, जो कि एक Machine Language Code File होती है। इस Machine Language Code File को फिर से Compile करने की जरूरत नहीं होती है। जबकि Interpreter प्रोग्राम की हर लाइन को हर बार मशीनी कोड में बदलता है, जिससे एक Interpreted Program को हर बार Run करने के लिए Interpret करना जरूरी होता है। HTML Code File Interpreted Program का एक उदाहरण है, जिसे हर बार Run होने के लिए Web browser Interpreter की जरूरत होती है।

Assembler

Assembly Language में लिखे प्रोग्राम को मशीनी भाषा में बदलने का काम Assembler करता है। ये एक ऐसा Software होता है, जो किसी Text File में लिखे गए विभिन्न Assembly Codes को Computer की मशीनी भाषा में Convert करके Computer के CPU पर Process करता है। Computer का CPU उन Converted Codes को समझता है और हमें हमारा वांछित परिणाम उस भाषा में प्रदान करता है, जिस भाषा को हम समझ सकते हैं यानी CPU हमें सामान्य English भाषा में Processed Results प्रदान करता है।

Steps of Program:

1 (Problem Definition) प्राग्राम परिभाषण

इस चरण में उस समस्या को पूरी तरह से समझना होता है, जिसका प्रोग्राम बना कर कम्प्यूटर से समाधान प्राप्त करना हैं। यानी प्रोग्राम के द्वारा हमें क्या प्राप्त परिणाम करना है, यह निष्कर्ष निकालना होता है।

सारांश :– क्या परिणाम प्राप्त करना हैं ?

2 (Problem Design) प्रोग्राम डिजाइन

इस चरण में समस्या को कई भागों में बांट कर उसे बीजगणितीय एल्गोरिद्म के अनुसार लिख लिया जाता है। एल्गोरिद्म लिखने के लिए फ्लोचार्ट आदि को उपयोग में लिया जाता है।

सारांश :– कैसा परिणाम प्राप्त करना है ?

3 (Program Coding) कोडिंग

इस चरण में हाई लेवल भाषा के कोडों के अनुसार एल्गोरिद्म व फ्लोचार्ट की मदद से प्रोग्राम की कोडिंग की जाती है।

सारांश :– कब क्या होगा जब User इसे उपयोग में लेगा ?

4 (Program Execution) प्रोग्राम को **Execute** करना

इस चरण में बनाए गए प्रोग्राम को चलाया जाता है।

5 (Program Debugging) डीबगिंग

जब प्रोग्राम को बनाया जाता है, तब कई तरह की गलतियां रह जाती हैं। जिससे जब प्रोग्राम को चलाया जाता है तब या तो प्रोग्राम रन नहीं होता या फिर सही परिणाम प्राप्त नहीं होता है। जब प्रोग्राम को कम्पाइल किया जाता है तो कम्पायलर में एक डीबगर होता है, जो प्रोग्राम में जिस जगह पर गलती होती है, वहीं पर आकर रूक जाता है। हम वहां पर होने वाली बग को सही करके प्रोग्राम को पुनः रन करते हैं। प्रोग्राम मे होने वाली गलतियों को ढूंढना व उन्हे सही करना ही डीबगिंग कहलाता हैं।

सारांश :– प्रोग्राम की किसी भी तरह की व्याकरण सम्बंधी या तर्क सम्बंधी गलती को खोजना व उसे संसोधित करके प्रोग्राम को सही करना।

6 (Program Testing) प्रोग्राम टेस्टिंग

कई बार प्रोग्राम पूरी तरह सही रन होता है, लेकिन फिर भी उसमें गलती होती है। इसे तार्किक गलती कहते हैं। इस प्रकार की गलती से हमें वांछित सही परिणाम प्राप्त नहीं होता है। इसे सुधारने के लिए प्रोग्राम से ऐसी समस्याओं का हल मांगा जाता है, जिसका परिणाम हमें पहले से ही पता होता है। ऐसा करने से यदि प्रोग्राम में कहीं पर तार्किक कमी हो तो पता चल जाता है। इस प्रक्रिया को प्रोग्राम टेस्टिंग करना कहते हैं।

7 (Program Documentation) प्रोग्राम विवरण

कई बार प्रोग्राम इतने बडे व जटिल हो जाते हैं कि कब कहां और क्या होना है और कौनसा प्रोग्राम क्यों लिखा गया था इसका पता ही नहीं चल पाता है। इस तरह की समस्याओं से बचने के लिए प्रोग्राम में कई जगहों पर ऐसी टिप्पणीयां डाल दी जाती हैं, जिससे पता चल सके कि प्रोग्राम क्या है व वह प्रोग्राम किसलिए लिखा गया है।

C Language में दो तरह के Functions होते है:

1. **Pre-Defined** या **Built-In Functions:** जो Functions हमें Directly Use करने के लिए पहले से ही प्राप्त होते हैं, उन्हें **Pre-Defined** या **Built-In Functions** कहा जाता है। उदाहरण के लिए *printf(), clrscr(), getch()* आदि Functions हमें पहले से ही प्राप्त हैं। इन्हें Use करने के लिए हमें केवल उन Header Files को अपने Source Program में Include करना होता है, जिनमें इन Functions को Define किया गया होता है। जब हम किसी Predefined Function को अपने Source Program में Use करते हैं, तो इस प्रक्रिया को Function Call करना भी कहा जाता है।
2. **User-Defined Functions**: दूसरे प्रकार के Functions वे Functions होते हैं, जिन्हें Programmer अपनी जरूरत के आधार पर Develop करता है। जिन Functions को एक Programmer स्वयं Create करके Use करता है, उन Functions को **User-Defined Functions** कहते हैं। User-Defined Functions बनाना एक Programmer की इच्छा पर निर्भर करता है।
   main() Function भी एक Programmer किसी समस्या का समाधान प्राप्त करने के लिए बनाता है, इसलिए main() Program को भी User-Defined Function ही कहा जाता है। लेकिन ये एक ऐसा Function होता है, जिसे बनाना जरूरी होता है। यही वह Function होता है, जहां से Compiler Program को Execute करना शुरू करता है।

Preprocessor Directive

कई बार हमें ऐसी जरूरत होती है जिसमें हम चाहते हैं कि हमारा Source Program Compiler पर Compile होने के लिए Processor पर जाने से पहले कुछ काम करे। इस प्रकार के कामों को परिभाषित करने के लिए हम Preprocessor Directives का प्रयोग करते हैं। Preprocessor Directives की शुरूआत हमेंशा # से होती है और इन्हें हमेंशा Header Files को Include करने वाले Statement के Just नीचे लिखा जाता है।

Header File (#include<Header File>)

''सी'' भाषा में विभिन्न प्रकार के कामों को पूरा करने के लिए फंक्शनों की अपनी एक पूरी लाईब्रेरी है, जिसमें ढेर सारे **Built–In** Functions हैं। विभिन्न प्रकार के Functions को उनके काम करने की प्रकृति के आधार पर विभिन्न प्रकार की Files में Define या परिभाषित किया गया है। Functions की इन Files को **"C"** भाषा में **Header File** कहा जाता है। Header Files को Header File इसलिए कहा जाता है, क्योंकि ये Files किसी भी Source File के Head में यानी सबसे Top पर व सबसे पहले Include की जाती हैं। किसी भी Header File को प्रोग्राम में जोडने के लिए **#** के साथ **include** Keyword लगाया जाता है। फिर < > के चिन्हों के बीच में उस Header File का नाम लिखा जाता है, जिसे प्रोग्राम में जोडना होता है। इनको Declare करने का Syntax निम्नानुसार होता है–

Syntax : #include <header file name.h>

जैसे :– **#include <stdio.h>**
#include <conio.h>

"C" Language में जब हम किसी परिणाम को Computer की Screen यानी Output Device पर Display करना चाहते हैं, तब हमें "**stdio.h**" नाम की Header File में Define किए गए ***printf***() Function को Use करना होता है।

printf() Function

''सी'' भाषा में सभी I/O Functions **stdio.h** नाम की Header File में होते हैं। जब हमे कोई Message या किसी Variable में Stored मान को Screen पर Display करना होता है, तो हम **printf()** Function का प्रयोग करते हैं। इसका Syntax निम्नानुसार है–

printf(" Message CtrlStr1 CtrlStr2 CtrlStrN, Variable1, variable2, variableN);

मानलो कि हम एक ऐसा Program बनाना चाहते हैं, जिसे Run करने पर Monitor पर एक String Display हो। चूंकि हम हमारे इस Program में किसी प्रकार का कोई भी Input व Processing नहीं कर रहे हैं, इसलिए इस Program में केवल Output Section ही होगा। यदि हम इस Program का Algorithm बनाना चाहें, तो ये Algorithm निम्नानुसार बनेगा :

Algorithm

```
1 START [Algorithm Starts here.]
2 PRINT "Shastri Fourth Semester" [Print the message.]
3 END [Algorithm Ends here.]
```

यदि इस Algorithm के आधार पर हम यदि हम "C" Language में Program बनाना चाहें, तो उस Program का Source Code निम्नानुसार होगा :

```
#include<stdio.h>
void main()
{
        printf(" Shastri Fourth Semester ");
```

```
        getch();
}
```

1 हर प्रोग्राम में एक ***main***() Function होता है। main() Function एक Special Function होता है, क्योंकि जब हम "C" Language के किसी Program को Compile करते हैं, तो Compiler सबसे पहले Source Program में main() Function को ही खोजता है और Compiler को जहां पर main() Function मिलता है, Compiler वहीं से Program को Machine Language में Convert करना शुरू करता है।

2 { } (**Opening** व **Closing**) Curly Braces के बीच लिखे गए सभी Statements के समूह को **Statement Block** कहा जाता है और इन्हीं Statements का Execution होता है। चूंकि "C" Language में हर Function की शुरूआत एक Opening Curly Brace से व अन्त एक Closing Curly Brace पर होता है, इसलिए किसी भी Program के जितने भी Executable Instructions होते हैं, उन्हें main() Function के Statement Block में ही लिखा जाता है।

3 "C" Language में हर Statement का अन्त एक Semi Colon(;) द्वारा होता है और "C" में Double Quote (" ") के बीच लिखे जाने वाले Statements को **String** कहा जाता है।

4 printf() Function के " " (**Opening** and **Closing**) Double Quotes के बीच लिखा गया Statement Screen पर ज्यों का त्यों Print हो जाता है, क्योंकि ये एक Output Statement है जो किसी Message या मान को Screen पर Display करने का काम करता है।

इस Program को Run करने पर हमें निम्नानुसार Output प्राप्त होता है:

Output

Shastri Fourth Semester

Program Flow

जब इस Program को Run किया जाता है, तब:

1 यदि Program में किसी तरह की कोई Typing Mistake ना हो, तो "C" का Compiler सबसे पहले main() Function को खोजता है।

2 main() Function के मिल जाने के बाद Compiler main() Function के Statement Block में प्रवेश करता है और सबसे पहले ***clrscr***() Function को Execute करता है। ये Statement Output Screen को Clear कर देता है।

3 फिर Program का अगला Statement ***printf***() Function Execute होता है, जो Screen पर "***Shastri Fourth Semester***" Message को Display करता है।

4 अन्त में तीसरा Function ***getch***() Execute होता है। ये Function User से एक Key Press करने का इन्तजार करता है और जब तक User Key Press नहीं करता है, तब तक वह Output को Screen पर देख सकता है। जैसे ही User Keyboard से किसी Key को Press करता है, Program Terminate हो जाता है।

Basic Elements of "C"

''सी'' को शुरू करने से पहले इसके कुछ आधारभूत अवयवों को जान लेना बहुत जरूरी है। कुछ खास तरह की Statements को लिखने के लिए विभिन्न प्रकार के Operators व Expressions की जरूरत होती है। हर भाषा में कुछ खास Statements व उनको लिखने के कुछ खास तरीके होते हैं। ये ही बात ''सी'' भाषा पर भी लागू होती है। इस अध्याय में हम ''सी'' के आधारभूत अवयवों के बारे में जानेंगे।

"C" Characterset

प्रत्येक भाषा में चिन्हो, अंकों, अक्षरों का एक समूह होता है। इन चिन्हो, अंकों व अक्षरों को एक विशेष क्रम में रखने पर एक शब्द बनता है जिसका कि अपना एक खास अर्थ होता है। जैसे र् + अ + म् मिलकर राम शब्द बनाते हैं जिसका अपना एक अर्थ होता है।
इसी तरह ''सी'' में भी कुछ खास चिन्हो, अंको व अक्षरों को मान्यता दी गई है, जिनके मिलने से कुछ खास अर्थ निकलते हैं जिन्हे वास्तविक तौर पर सिर्फ कम्प्यूटर ही समझता है। इन चिन्हो, अंकों व अक्षरों के समूह को ''सी'' भाषा का ''सी'' केरेक्टर सेट कहा जाता है, जो कि निम्नानुसार होता है:

1 Uppercase (A-Z) and Lowercase (a-z) Alphabet
2 0 to 9 Digits
3 Whitespace Characters (Blank Space, H-Tab, V-Tab, Form Feed, New Line Character, Carriage Return)
4 Special Characters

,	Comma	;	Semi Colon
:	Colon	?	Question Mark
.	Dot	'	Single Quote

"	Double Quote	|	V-Bar
$	Dollar Sign	#	Pound Sign
&	Ampersand	*	Asterisk
(	Left Parentheses	)	Right parentheses
[	Left Bracket	]	Right Bracket
{	Left Curly Brace	}	Right Curly Brace
<	Less Then Sign	>	Greater Then Sign
Blank	=	Equal to	
\	Back Slash	/	Slash
_	Under Score	%	Percent
~	Tilde	^	Upper Carat
+	Plus	-	Minus
!	Exclamation mark		

इस सारणी में हमने जितने भी Characters को दर्शाया हैं, उन सभी Characters को हम एक "C" Program में समय–समय पर व जरूरत के आधार पर Use कर सकते हैं।

"C" Tokens

जिस प्रकार से शब्द, किसी भी पैराग्राफ की वह लघुत्तम इकाई होती है, जिसमें एक विशेष अर्थ विधमान रहता है, ठीक इसी तरह इस भाषा में भी ऐसे ही कुछ शब्द, चिन्ह आदि हैं, जो स्वतंत्र रूप से अपना कुछ अर्थ रखते हैं। ''सी'' भाषा की वह लघुत्तम इकाई जो स्वतंत्र रूप से अपना कोई अर्थ रखती है, ''सी'' टोकन् कहलाती है। ''सी'' भाषा में पांच तरह के ''सी'' टोकनस् होते हैं, जिन्हे निम्नानुसार समझाया गया है:

1. Keywords या **Reserve Words:** ''सी'' भाषा के कुछ शब्दों को Reserve रखा गया है। इन शब्दों का C Compiler के लिए Special Meaning होता है, इसलिए इन्हें **Keyword** या **Reserve Words** कहते हैं। हर Reserve Word का अपना एक Special Meaning होता है और हर Reserve Word को किसी विशेष परिस्थिति में विशेष काम को पूरा करने के लिए ही Use किया जाता है। हम किसी Reserve Word को किसी सामान्य काम के लिए Use नहीं कर सकते हैं। C भाषा में निम्नानुसार 36 Keywords Define किए गए हैं। कुछ Compilers में इनकी संख्या 32 ही होती है तो कुछ Compilers में इनकी संख्या 36 से ज्यादा भी हो सकती है।

1	**auto**	2	**break**	3	**case**	4	**char**
5	**const**	6	**continue**	7	**default**	8	**do**
9	**double**	10	**else**	11	**enum**	12	**extern**
13	**float**	14	**for**	15	**goto**	16	**if**
17	**int**	18	**long**	19	**register**	20	**return**
21	**short**	22	**signed**	23	**static**	24	**struct**
25	**switch**	26	**typedef**	27	**union**	28	**unsigned**
29	**void**	30	**while**	31	**asm**	32	**fortran**
33	**pascal**	34	**huge**	35	**far**	36	**near**

2. Identifiers – Constant and Variable Name: जब हम Program Develop करते हैं, तब हमें विभिन्न प्रकार के Data को Computer की Memory में Input करके उस पर विभिन्न प्रकार की Processing करनी होती है। Computer में Data के साथ हम चाहे किसी भी प्रकार की प्रक्रिया करना चाहें, हमें हर Data को सबसे पहले Computer की Memory में Store करना जरूरी होता है। Computer की Memory में किसी Data को Store किए बिना हम उस Data के साथ किसी प्रकार की कोई प्रक्रिया नहीं कर सकते हैं।

Computer में Memory के हर Location का एक **Unique Address** होता है। जब हम Computer में किसी Data को Process करने के लिए Input करते हैं, तब वह Data Memory के किसी ना किसी Location पर जाकर Store हो जाता है। लेकिन हमें कभी भी सामान्य तरीके से ये पता नहीं चल सकता है कि हमारे द्वारा Input किया गया Data Computer की किस Memory Location पर Store हुआ है और ना ही हम स्वयं कभी ये तय कर सकते हैं कि हमारा Data किस Memory Location पर Store होगा, क्योंकि Data को Memory Allocate करने का काम अपनी सुविधानुसार हमारा Operating System स्वयं करता है।

जिस समय हमारे Data को Store करने के लिए Compiler Memory Reserve करता है, उसी समय हम उस Reserve होने वाली Memory Location का एक नाम Assign कर देते हैं। इस नाम के द्वारा ही हम हमारे Data को Computer की Memory में Identify कर सकते हैं। हमारे द्वारा किसी Data की Memory Location को दिए जाने वाले इस नाम को ही **Identifier** कहते हैं। हम किसी Memory Location का जो नाम Assign करते हैं, उन नामों को कुछ नियमों को ध्यान में रख कर परिभाषित करना होता हैं, क्योंकि ''सी'' कम्पाइलर उन विशेष प्रकार के नियमों के आधार पर परिभाषित किये गए नामों के साथ ही विभिन्न प्रकार की प्रक्रियाएं करता है। किसी Identifier को नाम देने के लिए हमें निम्न नियमों को Follow करना होता है, जिन्हें ***Identifier Naming Convention*** कहा जाता है:

• किसी भी Identifier के नाम में किसी भी **Upper Case** व **Lower Case Character** का प्रयोग किया जा सकता है।
• किसी भी Identifier के नाम में **Underscore (_)** का भी प्रयोग किया जा सकता है।
• किसी भी Identifier के नाम में यदि हम अंकों का प्रयोग करना चाहें, तो अंकों का प्रयोग करने से पहले कम से कम एक Character या Underscore का होना जरूरी होता है।
• इसके अलावा Identifier के नाम में किसी भी प्रकार के Special Symbol जैसे कि Period, Comma, Blank Space आदि का प्रयोग नहीं किया जा सकता है। साथ ही हम Identifier के नाम में किसी Reserve Word या किसी Built-In Function के नाम का प्रयोग भी नहीं कर सकते हैं।
• किसी भी नाम की शुरूआत किसी अंक से नहीं हो सकती है।
• ''सी'' एक **Case Sensitive Language** है, इसलिए इस भाषा में Capital Letters व Small Letters के नाम अलग–अलग माने जाते हैं। जैसे int Sum व int sum दो अलग–अलग **Variable Name** या **Identifies** होंगे ना कि समान।
किसी Variable Identifier या Constant Identifier का हम निम्न तरीके का कोई भी नाम रख सकते हैं, जो कि "C" के Naming Rules का पूरी तरह से पालन करते हैं:

number
number2
amount_of_sale
_amount
daysOfWeek

3. Constants and Variables:

Constants: Constants उस वेरिएबल को कहते हैं जिसकी वैल्यू पूरे प्रोग्राम के execution के टाइम में नहीं बदलती है| इन फिक्स वैल्यू को litrals भी कहते हैं| Contants किसी भी बेसिक डाटा टाइप के तरह के हो सकते हैं जैसे integer constants , character constant , float constants etc. Constants भी एक प्रकार के वेरिएबल ही हैं बस उनकी वैल्यू एक बार initialize होने के बाद दुबारा नहीं बदलती। **const keyword से** जब हम किसी वेरिएबल को ***const*** से डिफाइन करते हैं तो वह कंपाइल टाइम पर ऊपर वेरिएबल के डिफाइन स्टेटमेंट से उसकी वैल्यू ले लेता है ।

Example: const int num 5;

Variables: Program के वे मान जो पूरे Program में समय–समय पर आवश्यकतानुसार बदलते रहते हैं, Variables कहलाते हैं। Variables कभी भी किसी स्थिर मान को Represent करने के लिए Use नहीं किए जाते हैं। जब भी हमें किसी Constant को Program में Use करना होता है, तो उस Constant को Represent करने के लिए हमें Symbolic Constants की जरूरत होती है। इन Symbolic Constants को ही Literal भी कहा जाता है। किसी Calculation के Result को Store करने के लिए हमें हमारे Program में हमेंशा एक ऐसी Memory की जरूरत होती है, जिसमें विभिन्न प्रकार के बदलते हुए मान Store हो सकें।

Identifier Declaration: Program की जरूरत के आधार पर किसी Data को Store करने के लिए Computer की Memory में Space Reserve करने व उस Space का कोई Symbolic नाम देने की प्रक्रिया को **Identifier Declaration** कहते हैं। यदि Define किए जाने वाले Identifier का मान पूरे Program में स्थिर रहे, तो इस प्रक्रिया को **Constant Declaration** कहते हैं, जबकि यदि Define किए जाने वाले Identifier का मान पूरे Program में समय–समय पर Program की जरूरत के आधार पर बदलता रहे, तो इसे **Variable Declaration** कहते हैं।
Identifier Declaration के समय हमें हमेंशा दो बातें तय करनी होती हैं। पहली ये कि हमें किस प्रकार (***Data Type***)का Data Computer की Memory में Store करना है और दूसरी ये कि Reserve होने वाली Memory Location को क्या नाम (***Identifier Name***) देना है। Identifier Declare करने का General Syntax निम्नानुसार होता है:

Syntax:

	DataType	**V***ariableName*;
Example:	**int**	**age;**

DataType:
Syntax के इस शब्द के स्थान पर कुछ ऐसे Keywords का प्रयोग किया जाता हैं, जो ये तय करते हैं कि हम Computer की Memory में किस प्रकार के Data को Store करना चाहते हैं। उदाहरण के लिए यदि हमें केवल पूर्णांक संख्याओं को Store करने के लिए Memory Reserve करना हो, तो हम इस शब्द के स्थान पर ***int*** Keyword का प्रयोग करते हैं, जबकि यदि हमें किसी दसमलव वाली संख्या के लिए Memory Reserve करना हो, तो हमें इस शब्द के स्थान पर ***float*** Keyword को Use करना होता है।

Initialization
हम किसी भी Identifier को उसके Declaration के समय ही किसी ना किसी प्रकार का मान भी प्रदान कर सकते हैं। Identifier को उसके Declaration के समय ही कोई मान प्रदान करने की प्रक्रिया को **Value Initialization** करना कहते हैं। जैसे :–

```
int age   = 20;
int marks = 73;
```

हम एक ही समय में एक से अधिक Identifiers को जो कि समान प्रकार के Data Type के हों, Declare कर सकते हैं व किसी ना किसी मान से Initialize भी कर सकते हैं। जैसेः

int age, marks ;

OR

int age = 20, marks = 73 ;

Expressions:

जब दो या दो से अधिक **Operands** पर **Operators** की सहायता से कोई प्रक्रिया करके कोई परिणाम प्राप्त करना होता है, तो उस स्थिति में हम जो **Statement** लिखते हैं, उसे **Expression** कहते हैं। उदाहरण के लिए दो संख्याओं को जोड कर प्राप्त मान को किसी तीसरे Identifier में Store करने के लिए हम निम्न Statement लिखते हैंः

sum = digit1 + digit2

इस Statement को Expression कहा जाता है। उपरोक्त Statement एक Arithmetical Expression का उदाहरण है। इसी तरह विभिन्न प्रकार के Logical, Relational आदि Operators को Use करके विभिन्न प्रकार के Expressions बनाए जा सकते हैं।

Data and Data Types

मान या मानों के समूह Computer के लिए Data होता है। Real World में भी Data (मान या मानों का समूह [**Value** or a **Set of Values**]) कई प्रकार के होते हैं। जैसे किसी व्यक्ति की उम्र को हम संख्या के रूप में दिखाते हैं, जबकि उस व्यक्ति के नाम को Characters के समूह के रूप में परिभाषित करते हैं। इसी के आधार पर "C" Language में भी विभिन्न प्रकार के Data को Store करने के लिए विभिन्न प्रकार के Data Types के Keywords को Develop किया गया है। वास्तव में Data केवल दो तरह के ही होते हैं। या तो Data Numerical होता है, जिसमें केवल आंकिक मान होते हैं और इनके साथ किसी ना किसी प्रकार की Calculation किया जा सकता है या फिर Alphanumerical जो कि Characters का समूह होते हैं, जिनके साथ किसी प्रकार की किसी Calculation को Perform नहीं किया जा सकता।

"C" Language में भी Data को Store करने के लिए दो अलग तरह के Data Types में विभाजित किया गया है, जिन्हें क्रमशः Primary (Standard) Data Type व Secondary (Abstract or Derived) Data Type कहा जाता है। Primitive Data Type **Standard Data Type** होते हैं, जबकि Derived या Abstract Data Type Primitive Data Type पर आधारित होते हैं। फिर जरूरत के अनुसार इन दोनों Data Types को भी कई और भागों में बांटा गया है, जिन्हें हम निम्न चित्र द्वारा समझ सकते हैंः

"सी" Data Types and Variables

C में डाटा टाइप वेरिएबल डिफाइन और फंक्शन डिफाइन की टाइप डिफाइन करता है। इसको यदि हम सरल भाषा में कहें तो अगर हम integer पर काम काम करना चाहते हैं, तो वेरिएबल इन्टिजर डाटा टाइप का ही डिफाइन करना होगा। वेरिएबल की टाइप से हमें यह पता चलता है की इस वेरिएबल की वैल्यू स्टोर करने में कितनी जगह की आवश्यकता होगी और बिट से इसे किस पैटर्न में सेव किया जायेगा।

C के डाटा टाइप इस प्रकार हैं

1. **Basic Type** – C लैंग्वेज में मुख्य रूप से 5 तरह के बेसिक डाटा टाइप हैं।
   1) int – integer: a whole number. (2 or 4 Byte)
   2) float – floating point value: a number with a fractional part. (4 Byte)
   3) double – a double-precision floating point value. (8 Byte)
   4) char – a single character. (1 Byte)
2. **Enumerated types** – ये उस प्रकार के डाटा टाइप हैं जो ऐसे वेरिएबल को डिफाइन करते हैं जिनकी वैल्यू पूरे प्रोग्राम में कुछ फिक्स वैल्यू ही हो सकती हैं।
3. **The type void** – Void का अर्थ है कि कोई वैल्यू उपलब्ध नहीं है| यह तीन तरह की स्थितियों में होता है|
   a) जब किसी फंक्शन से कोई value return नहीं आ रही हो| जैसे void exit (int status);
   b) जब किसी function में कोई argument नहीं भेजना हो तब| जैसे int rand(void);
   c) void* टाइप का pointer जो कि pointer टाइप वेरिएबल के लिए एड्रेस तो दिखता है पर उसकी टाइप नहीं बताता है|
   जैसे void *malloc(size_t size);
4. **Derived types** – इसमें 5 सुब डाटा टाइप हैं (a) Pointer types, (b) Array types, (c) Structure types, (d) Union types and (e) Function types.

नीचे दी गयी टेबल में C के स्टैंडर्ड डाटा टाइप और उनकी storage space और value range दी गयी हैं |

Type	Storage size	Value range
Char	1 byte	-128 to 127 or 0 to 255
Int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
Short	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,647

Type	Storage size	Value range	Precision
Float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
Double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Variable in C

वेरिएबल वास्तव में एक स्टोरेज स्पेस का नाम होता है जिसके इस्तेमाल हम अपने प्रोग्राम में कोई भी वैल्यू को स्टोर करने के लिए करते हैं| C का कोई भी वेरिएबल किसी स्पेसिफिक टाइप का ही हो सकता है, जो की उस वेरिएबल की स्टोरेज स्पेस की साइज या लेआउट के बारे में बताता है । डाटा टाइप से हम ये जान सकते हैं कि कोई भी वेरिएबल किस रेंज से किस रेंज तक की वैल्यू स्टोर कर सकता है तथा उस पर किस तरह के आपरेशन हो सकते हैं।

Example – int i ;

यदि i integer है तो i की स्टोरेज स्पेस 2 से 4 bytes होगी तथा उसमें -32,768 to 32,767 or -2,147,483,648 to 2,147,483,647 तक की वैल्यू स्टोर कर सकते हैं। और चूँकि i int है इसलिए उस पर सभी प्रकार मैथमेटिकल आपरेशन कर सकते हैं ।

वेरिएबल का नाम किसी भी letters , digits या अंडरस्कोर से मिला कर बनता है इसमें शुरुआत में करैक्टर या अंडरस्कोर ही हो सकता है| इसमें कैपिटल और स्माल letter अलग अलग माने जाते हैं इसलिए

```
int Age;
int age;
```

अलग अलग माना जायेगा । C में बेसिक डाटा टाइप नीचे लिस्ट में दिए गए हैं ।

C में वेरिएबल डिफाइन करना – C में वेरिएबल डिफाइन करते समय अपने कमांड के द्वारा हम कम्पाइलर को यह बताते हैं कि किसी वेरिएबल के लिए मेमोरी में कितनी स्पेस रखनी है। वेरिएबल डिफाइन करने के लिए किसी डाटा टाइप का नाम फिर जितने भी वेरिएबल उस टाइप के बनाने हैं उनके कॉमा के साथ लिखते हैं।

Example

```
int a, b,c;
char ch , na;
float salary;
```

वेरिएबल को initialize करना – वेरिएबल में जब कोई वैल्यू स्टोर (assign) की जाती है तो उसको वेरिएबल initialization कहते हैं |

```
Data_type variable_name = value;
```

जैसे `int a = 4; or float b = 5.5;`

Integer (int)

जब प्रोग्राम में सिर्फ पूर्णांक संख्याओं को Store करने के लिए ही Memory Reserve करनी होती है, तब Identifier को Integer प्रकार का Declare किया जाता है। इसमें भिन्नांक संख्याएं नहीं हो सकती है। किसी Variable को Integer प्रकार का Declare करने के लिए Identifier के नाम के साथ **int** Keyword का प्रयोग करके "C" Compiler को बताया जाता है कि वह Identifier केवल पूर्णांक संख्याओं को ही Memory में Store कर सकेगा। int प्रकार के Identifier में हम + व – दोनों तरह के मान रख सकते हैं।

यदि हम 16 – Bit Compiler का प्रयोग करते हैं तो इस प्रकार का Identifier Memory में दो Byte का Storage Space Reserve करता है जबकि यदि हम 32 – Bit का Compiler Use करते हैं तो इस प्रकार का Identifier Memory में 4 Bytes का Storage Space Reserve करता है।

long OR long int

जब हमें काफी बडी संख्या का प्रयोग करना होता है तब हम इस Data Type का चयन करते हैं। यह मेमोरी में 4 Byte की Storage Space Reserve करता है और -2,147,483,648 से 2,147,483,647 मान तक की संख्या को Store कर सकता है। इस Data Type का प्रयोग अक्सर वैज्ञानिक गणनाओं में किया जाता है जहां काफी बडी संख्याओं की गणना करनी होती है।

float

जब हमें प्रोग्राम में भिन्नात्मक व दशमलव वाली संख्याओं को Store करने के लिए Memory की जरूरत होती है, तब हम float प्रकार का Identifier Declare करते हैं। ये Identifier मेमोरी मे 4 Bytes की Storage Space Reserve करता है और भिन्न या घातांक रूपों में 3.4E-38 से 3.4E+38 मान तक की संख्या को Store कर सकता है। इस प्रकार के Identifier के साथ **unsigned, signed, short** या **long** किसी भी Modifier का प्रयोग नहीं किया जा सकता है।

fouble

जब हमें प्रोग्राम में इतनी बडी भिन्नात्मक या घातांक संख्या के साथ प्रक्रिया करनी होती है, जो की ***float*** की Range से भी ज्यादा हो, तब हम इस Data Type का प्रयोग करके Identifier Declare करते है। ये मेमोरी में 8 Byte की Storage Space Reserve करता है और 1.7E-308 से 1.7E+308 मान तक की संख्या को Store कर सकता है। इस प्रकार के Identifier को हम निम्नानुसार Declare कर सकते हैं:

double *d;*

long double: जब Double के साथ **long** Key word लगा दिया जाता है यानी जब **long double** प्रकार का Data Type Use करते है तब वह Identifier बडी से बडी संख्या को Store कर सकता है। यह मेमोरी में 10 Byte की Storage Space Reserve करता है और 3.4E-4932 से 3.4E+4932 मान तक की संख्या Store कर सकता है।

Character (char)

जब हमें Computer में ''सी'' Characterset के किसी Character को Store करने के लिए Memory को Reserve करना होता है, तब हम Character प्रकार के Data Type का प्रयोग करके Identifier Create करते हैं। इस प्रकार का Identifier Create करने के लिए हमें "C" Language के ***char*** Keyword का प्रयोग करना होता है। इस प्रकार का Identifier मेमोरी में 1 Byte की Space Reserve करता है। char प्रकार के Identifiers में हम केवल एक ही Character Store कर सकते हैं। हम char प्रकार के Identifier में संख्या भी Store कर सकते हैं। इस Data Type को भी दो भागों में बांटा गया है:
Computer की Memory में हम कभी भी किसी Character को Store नहीं करते हैं। यदि हम किसी Character को Store भी करते हैं, तो वह Character किसी ना किसी अंक के रूप में ही Computer में Store होता है। Computer में हर Character का एक ASCII Code होता है।

Data Types Modifiers

ये मानक डाटा टाइप की साईज बदल देते हैं यानी ये डाटा टाइप के आकार में परिवर्तन कर देते हैं। ये कुल चार प्रकार के होते हैं:

- **Signed**
- **Unsigned**
- **Short**
- **Long**

Control String

जिस तरह से हम "C" Language में विभिन्न प्रकार के Data को Store करने के लिए अलग–अलग Keywords का प्रयोग करके अलग–अलग **Limit** की Memory Location को Reserve किया जाता है, ठीक इसी तरह से अलग–अलग प्रकार के मानों को Access करने के लिए भी हमें अलग–अलग तरह के Control Strings का प्रयोग करना होता है। Control String कुछ ऐसे Characters होते हैं, जिन्हें **%** के साथ Use किया जाता है।
विभिन्न प्रकार के Data Type के Data को Screen पर Display करने के लिए **printf() Function** के साथ Use किए जाने वाले **Control String** को हम निम्न सारणी द्वारा समझ सकते हैं:

%d	Integer Data Type के मान को Display करने के लिए।
%c	Character Data Type के मान को Display करने के लिए।
%f	Float Data Type के मान को Display करने के लिए।
%lf	Double Data Type के मान को Display करने के लिए।
%s	String Data Type के मान को Display करने के लिए।

Input / Output Function:

1. printf() function:

printf() Function का प्रयोग हम किसी भी प्रकार के Numerical या Alphanumerical मान को Monitor पर Display करने के लिए करते हैं। इस Function में हमें जो भी Message Screen पर Display करना होता है, उस Message को हम String के रूप में Double Quotes (" ") के बीच में लिखते हैं। Double Quotes के बीच में लिखा गया Message ज्यों का त्यों Screen पर Display हो जाता है। उदाहरण के लिए यदि हमें Screen पर "**Hello World**" Print करना हो, तो हमें printf() Function में इस Message को निम्नानुसार लिखना होता है:

printf("Hello World");

इस Statement का Output हमें निम्नानुसार प्राप्त होता है:

Hello World

यानी printf() Statement में हम String को जिस Format में लिखते हैं, Output में हमें वह String उसी Format में दिखाई देता है। लेकिन विभिन्न प्रकार की Calculations के बाद प्राप्त होने वाले Result को Display करने के लिए भी हमें printf() Function का ही प्रयोग करना होता है।

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int Integer = 10;
        char Character = 'X';
        float Float = 13.2;
        double Double = 12365.599999;
        char String[] = "Hello World";
        printf("\n Integer = %d", Integer);
        printf("\n Character = %c", Character);
        printf("\n Float = %f", Float);
        printf("\n Double = %e", Double);
        printf("\n Double = %g", Double);
        printf("\n String = %s", String);
        getch();
}
```

2. scanf() Function

जब हम Keyboard से किसी Input को प्राप्त करना चाहते हैं, ***scanf***() Function Keyboard पर Press की गई Keys की Information को Keyboard के **Buffer** से प्राप्त करता है और उन Keys की Information को ***scanf***() Function में Specify किए गए Variable Identifier की Storage Location पर Store कर देता है। जब हम Keyboard से किसी Data को Input के रूप में प्राप्त करके किसी Memory Location पर Store करना चाहते हैं, तब जिस Data Type के Data को Keyboard से Receive करना चाहते हैं, उस Data Type के Control String को ***scanf***() Function में Specify करते हैं और Keyboard से आने वाले Data को Memory के जिस Storage Location पर Store करना चाहते हैं, ***scanf***() function में उस Storage Location के Variable Identifier का नाम Address Operator (**&**) के साथ Specify करते हैं।

printf() Function के साथ जो Control String जिस Data Type से Related होता है, scanf() Function में भी वह Control String उसी Data Type से Associated होता है। scanf() Function के साथ Use किए जा सकने वाले Control Strings निम्नानुसार हैं:

%d	Keyboard से Integer Data Type के मान को प्राप्त करने के लिए
%c	Keyboard से Character Data Type के मान को प्राप्त करने के लिए
%f	Keyboard से Floating Point Real Data Type के मान को प्राप्त करने के लिए
%lf	Keyboard से Double Data Type के मान को प्राप्त करने के लिए
%u	Keyboard से Unsigned Decimal Integer Data Type के मान को प्राप्त करने के लिए
%s	Keyboard से String Type के मान को प्राप्त करने के लिए

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

```
        int a, b, sum;
        printf("Enter First and Second Values ");
        scanf("%d%d", &a, &b);
        sum = a + b;
        printf("\n Total of %d and %d is = %d ", a, b, sum);
        getch();
}
```

& Operator को **Address Operator** कहा जाता है। ये एक Unary Operator है। ये Operator हमेंशा उस Identifier के Memory Location का Address Return करता है, जिसके साथ इसे Use किया जाता है।

Operators

किसी भी प्रोग्रामिंग भाषा में विभिन्न प्रकार के Results प्राप्त करने के लिए विभिन्न प्रकार के Mathematical व Logical Calculations करने पडते हैं। इन विभिन्न प्रकार के Mathematical व Logical Calculations को Perform करने के लिए कुछ Special Symbols का प्रयोग किया जाता है। ये Special Symbols कम्प्यूटर को विभिन्न प्रकार के Calculations करने के लिए निर्देशित करते हैं। विभिन्न प्रकार के Calculations को Perform करने के लिए Computer को निर्देशित करने वाले चिन्हों को **Operators** कहा जाता है। साथ ही Data को Refer करने वाले जिन Identifiers के साथ ये प्रक्रिया करते हैं, उन Identifiers को इन Operators का **Operand** कहा जाता है। Operators दो तरह के होते हैं:

Unary Operator

कुछ Operators ऐसे होते हैं, जिन्हें कोई Operation Perform करने के लिए केवल एक Operand की जरूरत होती है। ऐसे Operator **Unary Operator** कहलाते हैं। जैसे Minus **(-)** एक Unary Operator है। जिस किसी भी संख्या के साथ ये चिन्ह लगा दिया जाता है, उस संख्या का मान बदल जाता है। जैसे 8 के साथ - चिन्ह लगा देने से संख्या -8 हो जाती है। "C" Language में Support किए गए Unary Operators निम्नानुसार हैं।

&	Address Operator
*	Indirection Operator
+	Unary Plus
-	Unary Minus
~	Bit wise Operator
++	Unary Increment Operator
--	Unary Decrement Operator
!	Logical Operator

Binary Operators

जिन Operators को काम करने के लिए दो Operands की जरूरत होती है, उन्हें **Binary Operators** कहते हैं। जैसे 2 + 3 को जोडने के लिए Addition Operator (+) को दो **Operands** की जरूरत होती है, अतः Plus (+) एक Binary Operator भी है। ''सी'' Language में विभिन्न प्रकार के Operators को उनके काम के आधार पर निम्न Categories में बांटा गया है:

1. Arithmetic Operators

इनका उपयोग गणित के संख्यात्मक मानों की गणना करने के लिए किया जाता है। इन Operators की कुल संख्या पांच होती है, जो कि निम्नानुसार है:

a. Addition Operator (+)

ये Operator दो Operands को जोड कर उनका योगफल Return करता है। जैसे

C = 10 + 3

b. Subtraction Operator (-)

ये Operator पहले Operand के मान में से दूसरे Operands के मान को घटाने पर प्राप्त होने वाले घटान या घटाफल को Return करता है। जैसे

C = 10 - 3

c. Multiplication Operator (*)

ये Operator दोनों Operands के मानों को गुणा करके प्राप्त होने वाले गुणनफल को Return करता है। जैसे

C = 10 * 3

d. Division Operator (/)

ये Operator पहले Operands के मान में दूसरे Operand के मान का भाग देकर प्राप्त होने वाले भागफल को Return करता है। जैसे

C = 10 / 3

e. Modules OR Reminder Operator (%)

ये Operator पहले Operands के मान में दूसरे Operand के मान का भाग देकर प्राप्त होने वाले शेषफल को Return करता है। जैसे

C = 10 % 3

Result of C = **1**

Example:

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int A = 10, B = 3, C;
        C = A + B ;
        printf("\n Addition = %d", C);
        C = A - B ;
        printf("\n Subtraction = %d", C);
        C = A * B ;
        printf("\n Multiplication = %d", C);
        C = A / B ;
        printf("\n Division = %d", C);
        C = A % B ;
        printf("\n Modules|Reminder = %d", C);
        getch();
}
```

Output:

Addition = 13

Subtraction = 7

Multiplication = 30

Division = 3

Modules|Reminder = 1

2. Relational Operators

Relational Operators (रिलेशनल ऑपरेटर) उन ओपेरटरों को कहते हैं जो दो वेरिएबल के बीच का रिलेशन बताते हैं। जब Program में किसी Condition के आधार पर Execute होने वाले Statements का चुनाव करना होता है, तब Condition को Specify करने के लिए हम Relational Operators का प्रयोग करते हैं। किसी प्रोग्राम में इन Operators का प्रयोग करके हम ये पता लगाते हैं कि कोई Condition सही है या नहीं। यदि Statement सही (***True***) होती है, तो ये Operators **1** Return करते हैं और यदि Condition सही नहीं होती है (***False***) तो ये Operators **0** Return करते हैं। Relational Operators निम्न हैं:

Operator	**"C" Symbol**
Equal to	==
Not Equal to	!=
Less then	<
Greater then	>
Less then or Equal to	<=
Greater then or Equal to	>=

3. Logical Operator (लॉजिकल ऑपरेटर)

जब किसी Program में जिसमें दो या दो से अधिक Conditions के साथ प्रक्रिया करके परिणाम प्राप्त करना होता है, तब Logical Operators का उपयोग किया जाता है। नीचे दिए गए टेबल में वो सभी लॉजिकल ओपेरटर दिए गए हैं जिनका उपयोग C में होता है यहां पर हम ये मान लेते हैं कि A की वैल्यू 1 है और B की वैल्यू 0 है|

Operator	Description	Example
&& (AND)	जब Logical Operator के दोनों तरफ की Condition ***True*** होती है, तब ये AND Logical Operator ***True*** या **1** Return करता है। यदि AND Logical Operator के दोनों तरफ की Conditions में से किसी एक भी Condition द्वारा **0** या ***False*** Return हो रहा हो, तो ये Logical Operator भी ***False*** Return करता है।	(A && B) is false.
|| (OR)	इस Logical Operator के दोनों तरफ की Condition में से यदि किसी एक तरफ की Condition भी ***True*** होती है, तब भी ये Logical Operator ***True*** या 1 Return करता है। यदि Logical Operator केवल एक ही स्थिति में ***False*** Return करता है, जब इस Logical Operator के Left Hand Side व Right Hand Side दोनों तरफ की Conditions ***False*** होती हैं।	(A || B) is true.
! (NOT)	इसको नॉट लॉजिकल ऑपरेटर कहते हैं । यह किसी भी वेरिएबल या ऑपरेंड की लॉजिकल स्टेट को reverse करने के लिए इस्तेमाल किया जाता है जैसे यदि किसी की लॉजिकल स्टेट true है तो इस ऑपरेटर को इस्तेमाल करने के बाद इसकी वैल्यू false होगी|	!(A && B) is true

4. Bitwise operator (बिटवाइस ऑपरेटर)

"C" Language में कुछ ऐसे Operators भी Provide करता है, जिनका प्रयोग हम किसी Identifier की Bits पर कर सकते हैं। इस Operator का उपयोग सीधे ही किसी Identifier की Bits पर काम करने के लिए किया जाता है। ये Operator हमेंशा Integer प्रकार के Data Type के साथ ही Use होता है, यानी Bitwise Operators को केवल Integer प्रकार के Data Type के Identifier के साथ ही प्रक्रिया करने के लिए Use किया जा सकता है। "C" Language में इनकी कुल संख्या छः होती है:

&	Bitwise **AND** Operator
!	Bitwise **OR** Operator
^	Bitwise **Exclusive OR** Operator
<<	Bitwise **SHIFT LEFT** Operator
>>	Bitwise **SHIFT RIGHT** Operator
~	Bitwise **Ones Compliment** Operator

Bitwise operator बिट पर कार्य करता और यह bit-by-bit पर कार्य करता है

P	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

5. Miscellaneous operators (मिसलेनियस ऑपरेटर)

इस सीरीज में वो ऑपरेटर आते हैं जिनकी बात ऊपर नहीं की गयी फिर भी वो बहुत महत्वपूर्ण हैं ।

Operator	Description	Example
sizeof()	sizeof() इसमें ब्रैकेट में जिस वेरिएबल का नाम लिखते हैं ये ऑपरेटर उसकी साइज बता देता है	sizeof(a), where a is integer, will return 4.

&	जब हमें किसी वेरिएबल का मेमोरी एड्रेस जानना होता है तो हम & के साथ वेरिएबल नाम लिखते हैं ।	&a; returns the actual address of the variable.
*	किसी पॉइंटर के साथ * इस्तेमाल करके हम उस वेरिएबल की वैल्यू जान सकते हैं ।	*a;
? :	इसे हम सिंगल लाइन इफ स्टेटमेंट भी कह सकते हैं इसमें ? से पहले if statement या condition होती है ? के बाद true statement और : के बाद false statement लिखते हैं ।	If Condition is true ? then value X : otherwise value Y

Control-flow Statements

जब किसी प्रोग्राम में एक से अधिक टास्क में से किसी एक टास्क को चुनना होता है तो इसके लिए डिसिशन मेकिंग स्टेटमेंट लिखने की आवश्यकता होती है। C प्रोग्रामिंग लैंग्वेज में नॉन जीरो या non-null वैल्यू का अर्थ होता है true और जीरो या null वैल्यू का अर्थ होता है false।

Boolean Expression – बूलियन एक्सप्रेशन वो एक्सप्रेसशन होता है जिसका रिजल्ट true या false में ही हो सकता है ।

if statement:

if statement में सबसे पहले condition को चेक किया जाता है यदि दी गई condition true है तो if statement के अन्दर दिया गया statement execute हो जाता है|

example:

```
#include <stdio.h>
void main()
{
   int x = 20;
   int y = 22;
   if (x<y)
   {
      printf("Variable x is less than y");
   }
  }
```

if - else statement:

if statement में सबसे पहले condition को चेक किया जाता है यदि दी गई condition true है तो if statement के अन्दर दिया गया statement execute हो जाता है और यदि condition false है तो else statement statement execute हो जाता है .

if...else statement का syntax

```
if ( condition )
{
        statement block;
}
else
{
        statement - outside;
}
```

Example:

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

```
    int x = 20;
    int y = 22;
    if (x<y)
    {
       printf("Variable x is less than y");
    }
    else
    {
       printf("Variable x is greater than y");
    }
}
```

nested if statements

जब हम एक if या else या if और else दोनों के अंदर दूसरा या अधिक if या if else या else if स्टेटमेंट लिखते हैं। तो उसे नेस्टेड इफ स्टेटमेंट कहते हैं ।

switch statement

स्विच स्टेटमेंट में एक कंडीशन होती है और और इस कंडीशन के रिजल्ट के आधार पर ढेर सारे टास्क में से कोई एक ही रन होता है । मानलो हमें ऐसी programming करनी है जिसमे हम अपनी जरुरत के अनुसार जिस statement को चाहे वही execute हो (लिखे गये statement में से) तो इस स्थिति में switch case statement better है| switch case statement भी एक decision making statement है| switch statement में expression एक integer है| इसमें एक – एक करके सभी statement को चेक किया जाता है जब कोई भी case दी गई expression से मैच करता है तो वो statement execute हो जाता है|

Example:

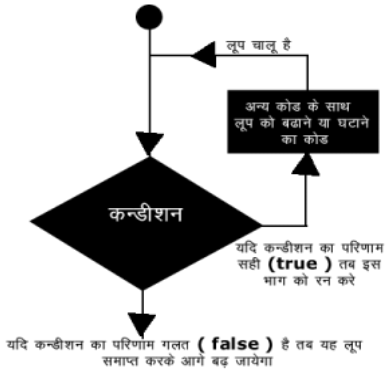
```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        printf("\nEnter a number: \n);
        scanf("%d",&n);
        switch(n)
        {
        case 1:
                printf("\n One");
                break;
        case 2:
                printf("\n Two");
                break;
        case 3:
                printf("\n Three");
                break;
        default:
                printf("\n Not match");
        }
}
```

Iteration (Loops) लूप

प्रोग्रामिंग में बहुत बार ऐसी स्थिति आती है जब कोड में किसी लाइन को या कुछ लाईन को बार बार रन कराने की आवश्यकता पड़ती है । ऐसी स्थिति में हम लूप का इस्तेमाल करते हैं । लूप भी एक तरह का प्रोग्रामिंग स्टेटमेंट है । लूप लगते समय कुछ बातें ध्यान देने की होती हैं । जैसे लूप में किन लाइन को बार बार रन करना है और कितनी बार रन करना है ।

लूप के लिए C में for , while , do – while प्रकार के लूप स्टेटमेंट इस्तेमाल किये जाते हैं ।

किसी भी लूप स्टेटमेंट के तीन भाग होते हैं और लूप काउंटर इस संख्या को निर्धारित करता है की लूप कितनी बार चलेगा । पहले भाग में हम लूप काउंटर को इनिशियलाइज़ करते हैं और इसके दूसरे भाग में हम चेक करते हैं की लूप अपनी अंतिम या लूप ख़तम करने कंडीशन में पहुँच गया है की नहीं और यदि नहीं तो हम लूप काउंटर की वैल्यू को थोड़ा बढ़ा या घटा (लूप के तीसरे पार्ट के अनुसार) देते हैं ।



1. for loop:

इस लूप में लूप वेरिएबल होता है, जो की यह तय करता है की लूप के अंदर का कोड कितनी बार रन करेगा हर बार लूप चलने से पहले लूप इस वेरिएबल की वैल्यू चेक कर लेता है|

for loop example:

```
#include <stdio.h>
void main ()
{
        int i;
        for(i=1 ;i<=10 ;i++ )
        {
                printf("This loop will run 10 times.\n");
        }
 }
```

2. while loop:

इस लूप में एक स्टेटमेंट या एक साथ कई स्टेटमेंट बार बार रन होते हैं जब तक की while में दी गयी कंडीशन true रहती है | यह हर बार लूप चलने से पहले कंडीशन चेक करता है ।

while loop example:

```
#include <stdio.h>
void main ()
{
        int i=0;
        while(i<=10 )
        {
```

```
                printf("This loop will run 10 times.\n");
                i=i+1;
        }
}
```

3. do…while loop:

इस लूप में एक स्टेटमेंट या एक साथ कई स्टेटमेंट बार बार रन होते हैं जब तक की while में दी गयी कंडीशन true रहती है | यह हर बार लूप चलने के बाद कंडीशन चेक करता है ।

do – while example:

```
#include <stdio.h>
void main ()
{
        int i=0;
        do {
                printf("This loop will run 10 times.\n");
                i=i+1;
          }while(i<=10 );
}
```

nested loops

इस प्रकार का लूप वह होता है जिसमें एक लूप के अंदर दूसरा लूप स्टेटमेंट होता है तो उसे नेस्टेड लूप स्टेटमेंट कहते हैं|

Loop Control Statements:

C में break, continue और goto command हैं जो की लूप को कण्ट्रोल करते हैं ।

S.N.	Control Statement & Description

1	**break statement:** यह लूप को तुरंत समाप्त करके कण्ट्रोल को लूप के बाहर अगले स्टेटमेंट पर पहुंच देता है । example:
	for(i=0;i<5;i++){
	if(i==2) {
	break; }
	}

2	**continue statement:** यह लूप में कॉन के नीचे लिखे स्टेटमेंट को छोड़ कर लूप को कंडीशन चेक करने के लिए और लूप को आगे बढ़ने के लिए फिर से ऊपर लूप के स्टेटमेंट पर भेज देता है ।
	for(i=0;i<5;i++){
	if(i==2) {
	continue; }
	}

3	**goto statement:** यह स्टेटमेंट वास्तव में दो पार्ट में होता है पहले पार्ट में किसी लेबल का नाम दिया जाता है और दूसरे पार्ट में goto के बाद उस लेबल का नाम दिया जाता है जिससे लूप के अंदर जब goto स्टेटमेंट मिलता है तो वह उस लूप को समाप्त कर उस लेबल पर कंट्रोल को पहुंचा देता है ।
	for(i=0;i<5;i++){
	if(i==2) {
	goto x; }

```
}
x:  printf("bye");
```

Function in 'C' (फंक्शन)

C में एक या अधिक स्टेटमेंट जो कि सम्मिलित रूप से किसी टास्क को करते हैं को फंक्शन कहा जाता है । किसी भी C प्रोग्राम में कम से कम एक फंक्शन main() होता ही है इसके अलावा अन्य प्रोग्राम में और भी फंक्शन होते हैं। अपने कोड को कई फंक्शन में बाँट कर रख सकते हैं। फंक्शन बना कर coding करने से कई लाभ होते हैं, जैसे एक ही प्रकार के टास्क के लिए हमें बार बार कोडिंग नहीं करनी पड़ती बस हमें उस फंक्शन को ही हर बार कॉल करना होता है| बड़े प्रोग्राम में एरर की बहुत सम्भावना होती है पर जब हम इससे कई छोटे छोटे फंक्शन में बाँट कर लिखते हैं तब हमारी पकड़ बहुत अच्छी होती है और गलतियों की सम्भावना बहुत काम होती है ।

1. Built-in Function or Predefined Function

हमें बहुत सारे फंक्शन कि के साथ भी मिलते हैं जिन्हें हम बिल्ट इन फंक्शन कहते हैं| इनको अलग अलग केटेगरी के हिसाब से अलग अलग लाइब्रेरी फाइल में डिफाइन किया गया है जब जिस प्रकार के फंक्शन को इस्तेमाल करना होता है उस कटेगरी की हेडर फाइल को अपने कोड में include करके हम उन फंक्शन का इस्तेमाल कर सकते हैं । जैसे input और output से सम्बंधित फंक्शन इस्तेमाल करने के लिए हमें standard input and output header file include करनी पड़ती है इसलिए हम #include <stdio.h> लिख कर इस फाइल को अपने कोड में ऐड कर लेते हैं।

Example: printf(), scanf(), getch() function.

2. User Defined Function

जो फंक्शन हम अपने प्रोग्राम में अपनी जरूरत के अनुसार डिफाइन करते हैं या बनाते हैं उन्हें यूजर डिफाइंड फंक्शन कहते हैं । जब हम फंक्शन डिफाइन की बात करते हैं तो हम मूल रूप से फंक्शन हेडर और फंक्शन बॉडी की बात करते हैं। जैसे Return type, function name, parameter और function body.

function prototype – यह पार्ट सबसे ऊपर लिखा जाता है इस पार्ट से कम्पाइलर को मालूम पड़ता है कि इस नाम का फंक्शन नीचे डिफाइन किया गया है ।

function define – इस पार्ट में फंक्शन में क्या होगा इसके लिए कोडिंग की जाती है |

function call – इस पार्ट में फंक्शन को कॉल किया जाता है या हम ये भी कह सकते हैं की इसी स्टेटमेंट के बाद ही फंक्शन कार्य करना शुरू करता है।

Example:

```
int add(int ,int );      //function prototype

int add(int a,int b) {              // function definition
        int c=0;
        c = a + b;
        return (c) ;
}

int sum = add(10,15);               // function calling
```

इसमें add function name है और उसके बाद ब्रैकेट में दो बार int लिखा गया है जिसका अर्थ है की इस function में हमें दो int value parameter या argument के रूप में भेजनी पड़ेंगी। लाइन में सबसे पहले भी int लिखा गया है इसका अर्थ है कि यह function run होने के बाद एक int value return करेगा ।

Function Call: फंक्शन को कॉल करते समय हम उसे value या argument दो प्रकार से दे सकते हैं |

Call by value:

इस प्रकार से जब हम किसी वेरिएबल की वैल्यू function में भेजते हैं तो उस वेरिएबल की वास्तविक value की copy भेजते हैं इससे वेरिएबल की वास्तविक वैल्यू पर कोई असर नहीं पड़ता है।

Example:

```
#include<stdio.h>
int add(int a,int b);                    //function decleration or function prototype
void main(){
        int a=10, b=20, c;
        c = add(a, b);                   //function call
        print("\n Sum =  %d",c);
}
int add(int a,int b){                    //function definition or function body
    int c = a+b;
    return (c);
}
```

output -> Sum = 30

Call by reference:

जब हम function call करते हैं तो हम variable का address भेजते हैं तो फंक्शन में कार्य उस variable की वास्तविक वैल्यू पर होता है।

Example:

```
#include<stdio.h>
int add(int *a, int *b);                 //function decleration or function prototype
void main(){
        int a=10, b=20, c;
        c = add(&a, &b);                 //function call
        print("\n Sum =  %d",c);
}
int add(int *a,int *b){                  //function definition or function body
    int c = *a + *b;
    return (c);
}
```

output -> Sum = 30

Recursion:

Recursive एक प्रोग्रामिंग तकनीक है जो प्रोग्रामर को स्वयं के संदर्भ में कार्य करने की permission देता है| सी में, यह एक ऐसे function का रूप लेता है जो स्वयं कॉल करता है| recursive function जहां निर्देशों में से एक "प्रक्रिया दोहराना" है। Recursive कुछ मायनों में लूपिंग के समान है दूसरी ओर, recursion उन विचारों को व्यक्त करना में आसान बनाता है जिसमें recursive कॉल का परिणाम कार्य को पूरा करने के लिए आवश्यक है।

Array

C language में एक प्रकार के data समूह को array कहा जाता है| यह data कंप्यूटर की menory में क्रमबद्ध तरीके से एक साथ रखा जाता है Array की एक इकाई को एलिमेंट कहा जाता है किसी भी ऐरे में सभी data एलिमेंट एक ही प्रकार के होना चाहिए यदि ऐरे intiger संख्या का है तो उसमे सिर्फ int value हो स्टोर हो सकती है एक ऐरे में कुछ एलिमेंट int ,कुछ float और कुछ char नहो हो सकते है ऐरे के किसी भी अवयव को प्रदर्शित करने के लिए ऐरे की नाम के बाद कोष्टक में index लिखते है| उदहारण: यदि किसी ऐरे का नाम marks है तो ऐरे marks[5] use ऐरे में पाचवे एलिमेंट को प्रदर्शित करेगे|

ऐरे का use प्रोग्रामिंग में बहुत लाभ दायक होत है क्योकि इसके मध्य में एक समूह के हजार अवयव के विषय में data भरा जाता है, संगणना की जाती है और परिणाम को भी मुद्रित किया जाता है| परीक्षाफल और अंकतालिका ऐरे के द्वारा ही तयार की जाती है| ऐरे की index 0 से start होती है|

Array एक ऐसा data structure है जो की fixed number के same data type elements को store कर सकता है। वैसे तो array बहुत सारे data को store करने के काम आता है लेकिन इसको same type के variable का collection भी कहा जा सकता है । "An Array is refers to a group of data item, where data items must be of similar type." Array एक ऐसा data type है जिसमे केवल similar type की value ही store की जा सकती है|

example – अगर array integer value के लिए declare किया गया है तो उसमे केवल integer value ही store की जा सकती है और यदि character type का declare किया गया है तो उसमे केवल character type की value ही store की जा सकती है और float type का है तो only float type की value स्टोर कर सकते है| दुसरे शब्दों में, एक array में only similar type की value की स्टोर कर सकते है|

किसी Array के अंदर जितने भी Value जमा होते है उन सबका एक index number होता है | ये index number 0 से start होता है मतलब Array के अंदर जो पहला Value जमा होता है उसका index number 0 होता है और दूसरे Value का index number 1 होता है और इसी तरह आगे के Value का index number 2 , 3 … होते जाता है | इसी लिए अगर हमे किसी Array के अंदर के किसी Value को output में दिखाना है तो हमे उस Array के नाम के साथ उसका index number लिखना होता है |

एक one dimensional Array का use साधरण लिस्ट इत्यादि बनाने में किया जाता है तथा एक ही index variable का use किया जाता है इसको single - subscripted variable भी कहते है

यदि ऐरे का नाम x है तो उसके n अवयव इस प्रकार प्रदर्शित कर सकते है: x[0],x[1],x[2]....x[n] इत्यादि

one dimensional Array का syntax निम्लिखित है:

datatype variablename [size];

यह type ऐरे के अवयव का डेटा type प्रदर्शित करता है जेसे int , float ,char आदि variable name ऐरे का नाम है जिसे ऐरे का memory में शुरुआती एड्रेस (base Address) भी कहते है और size ऐरे का size प्रदर्शित कर रही है size के अनुसार ही ऐरे के लिए memory में सुरक्षित होगी|

उदहारण : int group [10];

उपरोक्त उदहारण में एक group नाम का ऐरे लिया गया है जिसका type int है इसका अर्थ यह है की ऐरे में अभी एलिमेंट int type के स्टोर होगे|

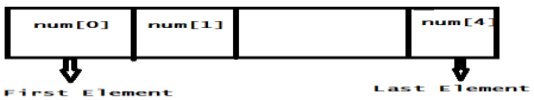
ऐरे को एक बार declare करने के बाद इसका size and data type change नहीं किया जा सकता।

Array declaration: int number[5];

ऊपर data type "int" है, "number" variable का नाम है एंड Brackets मे 5 का मतलब है की इसमें 5 int values को store किया जा सकता है।

Array मे कुछ store करना या access करना – ऊपर declare किये गए array मे 5 values store की जा सकती है।

First value access or refer के लिए : number[0] ;	First ऐरे मे value store करने के लिए : number[0] = 5;
second value access or refer के लिए : number[1] ;	Second ऐरे मे value store करने के लिए : number[1] = 10;
third value access or refer के लिए : number[2] ;	Third ऐरे मे value store करने के लिए : number[2] = 15;
Fourth value access or refer के लिए : number[3] ;	Fourth ऐरे मे value store करने के लिए : number[3] = 20;
Fifth value access or refer के लिए : number[4] ;	Fifth ऐरे मे value store करने के लिए : number[4] = 30;



Example 1:

```
#include<stdio.h>
 void main()
{
        int i;
        int arr[5] = {10, 20, 30, 40, 50};
        for (i=0; i<5; i++)
        {
                printf("\n %d ",arr[i]);
        }
 }
```

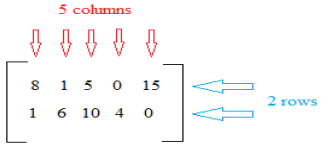
Example 2:

```
#include<stdio.h>
void main(){
        int arr[5];                    /* 1 - D array declaration*/
        int i, j;
        for(i=0; i<5; i++) {
                printf("Enter value in array: ");
                scanf("%d", &arr[i]);
                }
        //Displaying array elements
        printf("One Dimensional array elements:\n");
        for(i=0; i<5; i++) {
                printf(" %d ", arr[i]);
                }
  }
```

Two - Dimensional Array (2-D Array):-

two dimensional Array मेट्रिक्स या टेबल को प्रदर्शित करने के लिए use किया जाता है इसमें दो subscript use होते है एक रो की संख्या बताने के लिए और दूसरी कोलम की संख्या बताने के लिए इसके अवयव निर्दिष्ट करते समय पहले रो की संख्या लिखी जाति है|

Two Dimensional Array में data को row और column के अनुसार जमा किया जाता है| इस array में दो sub - script होती है। इसका सबसे बड़ा उदाहरण है Matrix.



Declaration of Two Dimensional Array in C Programming:- C Programming में Two Dimensional Array को declare करने का तरीका निम्नलिखित है |

Data_Type Array_Name[Row_Size][Column_Size];
int arr[2][3];

1. Data_Type :- सबसे पहले Array का Data Type लिखा जाता है | (example: int)
2. Array_Name :- इसके बाद Array का नाम लिखा जाता है | (example: arr)
3. Row_Size :- जिस Data को Array में जमा किया जा रहा है उसके पंक्तियों या Row की संख्या को यहाँ लिखा जाता है | (example: 2)
4. Column_Size :- जिस Data को Array में जमा किया जा रहा है उसके स्तंभ या Column की संख्या को यहाँ लिखा जाता है |(example: 3)

Example:

```
#include<stdio.h>
void main(){
        int arr[2][3];                 /* 2D array declaration*/
```

```
        int i, j;
        for(i=0; i<2; i++) {
                for(j=0;j<3;j++) {
                        printf("Enter value in array: ");
                        scanf("%d", &arr[i][j]);
                }
        }
        //Displaying array elements
        printf("Two Dimensional array elements:\n");
        for(i=0; i<2; i++) {
                for(j=0;j<3;j++) {
                        printf("%d ", arr[i][j]);
                }
        }
}
```

Pointers

पॉइंटर एक वेरिएबल है जो दूसरे variable का address संग्रहीत करता है| एक pointer एक अन्य पॉइंटर के address को भी स्टोर कर सकता है जिसे pointer की श्रृंखला कहा जाता है और वेरिएबल के मान को पॉइंटर के माध्यम से पढ़ा जा सकता है।मेमोरी के प्रत्येक संग्रहण स्थान का एक valied address है, address एक संख्या है जो क्रमिक रूप से बढ़ता है मेमोरी में रखे गए हर प्रोग्राम के लिए, प्रोग्राम में प्रत्येक वेरिएबल या फ़ंक्शन का valied address होता है। Pointers एक बहुत powerful प्रोग्रामिंग टूल हैं इसका उपयोग करके कार्यक्रम (Program) की efficiency में सुधार कर सकते है| यह unlimited मात्रा में डेटा को संभालने में भी आपकी मदद करता है।

Pointers का उपयोग व्यापक रूप से प्रोग्रामिंग में किए जाता हैं| वे variable identifier का उपयोग किए बिना किसी अन्य variable के स्मृति स्थान को referenced करने के लिए उपयोग में लाये जाते है।

Pointers का उपयोग:

- Pointers कार्य करने के लिए एक से अधिक मान वापस करने का एक तरीका प्रदान करते हैं।
- Pointers का उपयोग arrays और structures को संभालने के लिए किए जाता हैं क्योंकि ये अधिक कुशल होते हैं।
- Pointers complex डेटा संरचनाओं जैसे कि linked सूची, queues, पेड़, ग्राफ़ आदि का निर्माण करने में हमारी मदद करते हैं।
- Pointers कार्यक्रम के execution समय को कम करता है।

Syntax:

data_type *pointer_variable_name;
int *p;

उदाहरण: char*p; जहां, * यह दर्शाता है कि "p" पॉइंटर char है और एक सामान्य char नहीं है।

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a=10;
        int *p=&a;
        printf("\n a=%d",a);
        printf("\nAddress of a = %d",p);
        printf("\nValue of location p=%d",*p);
        getch();
}
```

Strings

C language में करैक्टर (Character) या अंको (Number) और संकेतो (Symbol) के समूह को string कहते है जिसे quotation mark (" ") के अन्दर लिखा जाता है| एक string को char के ऐरे (Array) के रूप में देखते है जिसे string का ऐरे भी कहते है| हर string का अंतिम char null केरेक्टर (\0) होता है जिससे string की समाप्ति का पता चलता है, इसलिए null केरेक्टर को स्टोर करने के लिए एक करैक्टर ऐरे का size (n+1) होना चाहिए क्योकि अंतिम जगह पर null ("\0") स्टोर किया जाता है|

Example :

char s[]={'i','n','d','i','a','\0'};

किसी भी string का प्रत्येक char menory में 1 बाइट लेता है और string का अंतिम char हमेशा null होत है जिसे compiler द्वारा string के अंत में जोडा जाता हो|

C language में string librery function निम्लिखित है

1. strcat()
2. strcpy()
3. strlen()
4. strcmp()

strcat() Function: इस function के द्वारा दो स्ट्रींग को एक साथ जोडा जाता है| इसका syntax निम्लिखित है:

strcat(s1,s1);

strcpy() Function: यह function एक string के अवयव को दूसरी string में कॉपी करता है इस function में स्त्रोत तथा लक्ष्य string के base address का argument में रूप में पास करते है| इसका syntax निम्नलिखित है:

strcpy(s1,s2);

strlen() Function: इस function के द्वारा string के char गिने जाते है जिसे string की लम्बाई कहते है इसका syntax निम्लिखित है

n =strlen(string);

n एक int variable है जिसे string की लम्बाई कहते है यह function पहले null char तक की string के char को गिनता है|

strcmp () Function: strcmp() दो string की तुलना करने के लिए use होता है यह तुलना दोनों string के एक एक char के ASCII मान के आधार पर होती है जेसे ही किन्ही तो char के ASCII मान में अंतर प्रप्त होता है तुलना वही रोक दी जाती है एसी स्तथी में दोनों char के ASCII मनो का अंतर function द्वारा लोटा दिया जाता है|

Syntax:

strcmp(s1,s2);

gets() function:

इस function का use इनपुट string function को पढने के लिए किया जाता| gets function में वाक्य अथवा एक या एक से अधिक शब्दों को इनपुट किया जाता है| इनपुट string में रिक्त स्थन और tab का use भी किया जा सकता है| इसका syntax निम्लिखित है:

gets(string);

puts() function:

puts() function का use string को आउटपुट के रूप में प्रदर्शित करने के लिए किया जाता है यह function भी एक समय में एक ही करैक्टर या variable का मान प्रदर्शित करने के लिए use किया जा सकता है| जेसे :

puts("Hello");

सी में एक स्ट्रिंग वास्तव में एक character array है एक individual character variable के रूप में केवल एक अक्षर को संग्रहीत करते है हमें स्ट्रिंग को स्टोर करने के लिए characters array की आवश्यकता है| प्रत्येक वर्ण एक array में एक स्थान पर रहता है अंतिम character के बाद रिक्त वर्ण '\ 0' को रखा गया है ऐसा इसलिए किया जाता है ताकि प्रोग्राम यह बता सके कि स्ट्रिंग के अंत में कब तक पहुंचा जा सकेगा ।

सी programming भाषा में स्ट्रिंग को घोषित करने के दो तरीके हैं-

1. **Char array:**
   char ch[10]={'u', 's', 'v', 'v', '\0'};
2. **String literal:**

char ch[]="usvv";

ऐसे केस में '\ 0' को कंपाइलर द्वारा स्ट्रिंग के अंत में जोड़ दिया जाता है।

Example:

```
#include <stdio.h>
void main ()
{
  char ch[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
  printf("Message: %s", ch );
  getch();
}
```

Structures

C भाषा में विभिन्न प्रकार के डेटा को स्टोर करने के लिए structures का उपयोग किया जाता है| c language में ऐरे का use एक ही प्रकार के data के लिए किया जाता है| ये data int float या char हो सकता है परन्तु एक array के सभी एलिमेंट एक ही प्रकार के data type के होते है ऐरे का data अमिश्रित होता है जबकि structure मिश्रित होता है जो यूजर द्वारा परिभषित होता है, अत: structure में एक साथ कई भिन्न प्रकार के data type का use किया जा सकता है|

Example: एक लाइब्रेरी में विभिन पुस्तको का रिकॉर्ड रखना है तो उसके लिए पुस्तक का नाम प्रष्ट क्रमांक व पुस्तक का मूल्य इस्यादी का data एक साथ तेयार करना होगा| पुस्तक का नाम तथा लेखक का नाम char data है जबकि प्रष्ट क्रमांक एक int data है और पुस्तक का मूल्य float data है|

Structure का उपयोग करके हम एक record को तैयार कर सकते है| Structure ये एक अलग-अलग data types का कलेक्शन होता है| Structure का इस्तेमाल करने के लिए 'struct' keyword का इस्तेमाल किया जाता है| एक संरचना के प्रत्येक element को एक member कहा जाता है|

C Language में different type का data स्टोर करने के लिए structure का use करते है| एक structure का syntax निम्लिखित है

```
struct structureName
{
  //member definitions
};
```

Example:

```
struct book
{
    char bookname[20];
    char authorname[20];
    int page;
    float price;
};
```

C Program for structure:

```
#include<stdio.h>
#include <string.h>
struct employee
{  int id;
   char name[50];
}e1;  //declaring e1 variable for structure
void main( )
{
  //store first employee information
  e1.id=105;
  strcpy(e1.name, "Sushil Chamoli");//copying string into char array
```

```
  //printing first employee information
  printf( "employee 1 id : %d\n", e1.id);
  printf( "employee 1 name : %s\n", e1.name);
}
```

Union: यूनियन भी c language का एक यूजर डिफाइन data type है जिसका syntax structure के सामान होता है परन्तु structure के हर सदस्य की स्टोरेज location अलग – अलग होती है जबकि यूनियन के सभी सदस्यों की स्टोरेज location एक होती है इसका अर्थ है की structure में अनेक प्रकार के data type के सदस्य हो सकते है किन्तु उनकी स्टोरेज location एक होने के कारण एक समय पर उसमे से किसी एक का use एक बार में किया जा सकता है

Example :

```
union abc
{
int a;
char c;
float d;
}code;
```

Dynamic Memory Allocation

malloc () function in C

इस function का use कर memory को रन टाइम पर allocate किया जाता है यह function बताए गई size का memory ब्लाक सुरक्षित रखता है और void type का pointer return करता है लेकिन जरूरत के अनुसार इसका type Change कर सकते है

इसका syntax निम्लिखित है :

```
ptr=(cast type*)malloc(size);
```

यह पर ptr cast type का pointer है|

calloc () function In C

यह एक अन्य memory एलोकेशन function है इसका use प्रोग्राम रन करने के दोरान diraived data type जेसे ऐरे स्ट्रक्चर आदि द्वारा मांगे गए memory space को allocate करने के लिए किया जाता है यह पर स्टोरेज के multipal ब्लाक को allocate करता है और फिर बाइट को 0 पर set कर देता है| इसका syntax निम्लिखित है

```
*ptr=(cast type*)calloc(n,elem-size);
```

relloc () function In C

इस function का use पूर्व में allocate memory space को परिवर्तित करने के लिए किया जाता है इसका syntax निम्लिखित है

```
ptr= relloc(ptr,size);
```

free () function In C

इस function का use रन करते समय allocate की गई memory को रिलीस करने के लिए किया जाता है इसका syntax निम्लिखित है

```
free(ptr);
```

References:

1. Let-Us-C by Yashavant-Kanetkar.
2. C In Depth by Deepali Srivastava, S.K Srivastava
3. C in Hindi by Kuldeep Chand.
4. https://www.web3tutorial.com/c/
5. http://notesandprojects.com/
6. https://programming-tutorial-hindi.blogspot.in/p/index.html
7. *http://engineersworld.in/Programming-in-c-Tutorial-in-Hindi*
8. *https://www.tutorialspoint.com/cprogramming/index.htm*
9. *https://www.cprogramming.com/tutorial/c-tutorial.html*
10. *https://www.studytonight.com/c/*

डेटाबेस मैनेजमेंट सिस्टम (DATABASE MANAGEMENT SYSTEM)

Database जानकारी (Data) का एक संग्रह है | डेटाबेस मैनेजमेंट सिस्टम(Database Management System) एक सॉफ्टवेर (Software) है जिसका उपयोग डेटाबेस को बनाने(creating) और डेटाबेस को संभालने(managing) के लिए किया जाता है |DBMS वह सॉफ्टवेर है, जिससे हम एक नया Database को बना सकते है|DBMS अपने उपयोगकर्ताओं (users) और प्रोग्रामर (programmers) को एक व्यवस्थित तरीके के साथ डाटा को बनाने(create) , संभालने (manage) और update करने की सुविधा प्रदान करता है । Database में जानकारियों को इस प्रकार से इकट्ठा किया जाता है कि जानकारियों को बड़ी आसानी से संभाला जा सके और जरूरत पड़ने पर किसी भी जानकारी को बड़ी आसानी से प्राप्त किया जा सके |
उदाहरण के लिए किसी स्कूल में पढ़ने वाले सभी विद्यार्थियों की जानकारी को Student Database (स्टूडेंट डाटाबेस) के अंदर इकट्ठा करके रखा जा सकता है जिसमें की विद्यार्थियों का नाम, पता, कक्षा रोल नंबर, इत्यादि से संबंधित जानकारियां हो सकती है|
DBMS के कुछ उदाहरण :- MySQL, PostgreSQL, Microsoft Access(माइक्रोसॉफ्ट एक्सेस), Oracle(ओरेकल) इत्यादि।

डेटाबेस मैनेजमेंट सिस्टम के फायदे (Advantages of Database Management System)

1. **Controlling Data Redundancy (डेटा रिडंडंसी को नियंत्रित करना) :-** File Based System में प्रत्येक एप्लिकेशन प्रोग्राम की अपनी निजी फाइल होती है इस स्थिति में, कई स्थानों पर एक ही डेटा की डुप्लिकेट files बनाई जाती हैं। DBMS में, एक संगठन (organization) के सभी डेटा को एक डेटाबेस फ़ाइल में एकीकृत किया जाता है मतलब की डेटा डाटाबेस में केवल एक स्थान पर दर्ज किया जाता है और इसे दोहराया नहीं जाता है।
2. **Sharing of Data (डेटा साझा करना) :-** DBMS में, organization के authorized users (अधिकृत उपयोगकर्ताओं) द्वारा डेटा साझा किया जा सकता है। डाटाबेस एडमिनिस्ट्रेटर डेटा को नियंत्रित करता है और डेटा को access करने के लिए उपयोगकर्ताओं को अधिकार देता है । कई उपयोगकर्ताओं को एक साथ जानकारी के समान टुकड़े तक पहुंचने का अधिकार दिया जा सकता है जा सकता है remote users भी समान डेटा साझा कर सकते हैं। इसी तरह, एक ही डाटाबेस के डेटा को अलग-अलग एप्लीकेशन प्रोग्राम के बीच साझा किया जा सकता है।
3. **Data Consistency (डाटा स्थिरता) :-** डेटा रिडंडंसी (Data Redundancy) को नियंत्रित करके, डाटा स्थिरता प्राप्त की जाती है। मतलब की डाटाबेस में एक ही प्रकार के डेटा को बार-बार इन जमा होने से रोका जाता है
4. **Integration of Data (डेटा का एकीकरण):-** DBMS में, डेटाबेस में डेटा tables (तालिका) में संग्रहित होता है। एक डेटाबेस में एक से अधिक tables होते हैं और तालिकाओं (या संबंधित डेटा संस्थाओं) के बीच रिश्तों को बनाया जा सकता है। इससे डेटा को पुनः प्राप्त करना और अपडेट करना आसान हो जाता है
5. **Data Security(डाटा सुरक्षा) :-** DBMS में डाटा को पूरी तरह से Database Administrator (एडमिनिस्ट्रेटर) द्वारा नियंत्रित किया जाता है और डाटा बेस एडमिनिस्ट्रेटर ही यह सुनिश्चित करता है कि किस User को कितना Database के कितने हिस्से पर Access देना है या नहीं देना है इससे डेटाबेस कि सिक्योरिटी बहुत अधिक बढ़ जाती है |
6. **Recovery Procedures(डाटा रिकवरी) :-** कंप्यूटर एक मशीन है इसलिए यह संभव है कि कभी भी कंप्यूटर में कोई हार्डवेयर या सॉफ्टवेयर संबंधित समस्या उत्पन्न हो जाए ऐसे में यह बहुत जरूरी है की कंप्यूटर में किसी प्रकार की समस्या उत्पन्न होने पर उसमें रखें डेटाबेस को हम Recover कर पाएं DBMS में यह काम बड़ी आसानी से किया जा सकता है |

डाटाबेस सिस्टम के नुकसान (Disadvantages of Database Management System)

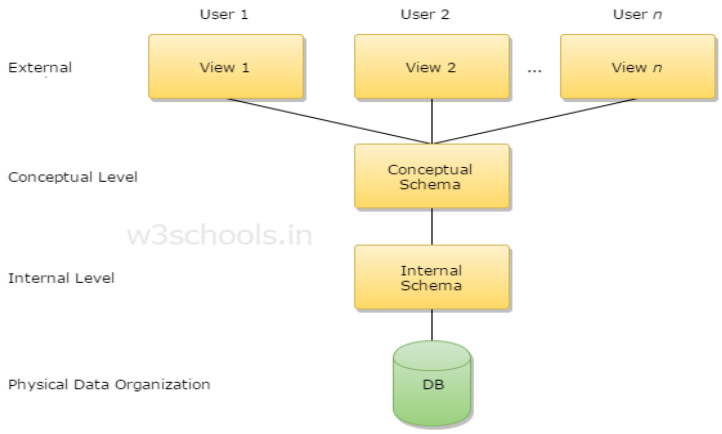
1. **Cost of implementing :-** डाटाबेस सिस्टम को implement(कार्यान्वयन) करने में जो लागत आती है वह काफी ज्यादा हो सकता है जिसमें काफी रुपए खर्च हो सकते हैं |
2. **Effort to transfer data :–** मौजूदा सिस्टम से डाटाबेस में डेटा को transfer करने के लिए काफी मुश्किलों का सामना करना पड़ सकता है और इसमें बहुत अधिक समय भी लग सकता है |

3. **Risk Of database fails :-** अगर डेटाबेस विफल हो जाता है भले ही अपेक्षाकृत कम अवधि के लिए तो संपूर्ण कंपनी पर भी असर पड़ेगा और कंपनी को कई प्रकार के नुकसान उठाने पड़ेंगे |

DATABASE ARCHITECTURE:-

Information Technology में, डेटा Database Architecture नीतियों (policies), नियमों (rules) या मानकों (standards) से बना होता है जो कि यह नियंत्रित करता है कि Database में डाटा कैसे संग्रहित हो रहा है उस Data का किस प्रकार से उपयोग हो रहा है और उस Data को किस प्रकार से संभाला जा रहा है | Data को वास्तव में bits, या संख्याओं और strings के रूप में संग्रहित किया जाता है, लेकिन इस स्तर पर डेटा के साथ काम करना मुश्किल है।

Database Architecture का उद्देश्य डेटाबेस के प्रत्येक Users (उपयोगकर्ता) को भौतिक रुप से Database का उपयोग करने से रोकता है किसी User को भौतिक रूप से Database का उपयोग करने से रोकने के कई सारे कारण है | जैसे कि जिस User को Database के जिस हिस्से का उपयोग करना है उस यूजर के समक्ष Database का वही हिस्सा प्रदर्शित हो और बाकी का डेटाबेस को वह यूजर Access ना कर पाएं जिससे डेटाबेस की सिक्योरिटी पर कोई खतरा पैदा ना हो जाए | दो अलग अलग यूजर एक ही डाटा को देखना चाहता है तो वह दोनों अलग-अलग उस एक ही डाटा को देख सकता है लेकिन अगर वह दोनों डेटाबेस के दिखने के तरीके को बदलना चाहे तो बदल सकता है लेकिन इस बदलाव का असर सिर्फ उस यूजर के लिए हो डेटाबेस पर इस बदलाव का कोई असर ना हो | डेटाबेस का User सिर्फ इतना जानता है कि Database में क्या जानकारी उपलब्ध है लेकिन वह यह कभी भी नहीं जान पाए कि Database में वह जानकारी किस प्रकार से जमा की गई है | Database Administrator (DBA) उपयोगकर्ता को प्रभावित किए बिना डेटाबेस संग्रहण संरचनाओं (database storage structures) को बदलने में सक्षम होना चाहिए |



Database Architecture को दो प्रकारों में बांटा गया है-

1. **Two-tier Client / Server architecture:** Two-tier Client / Server architecture के दो मुख्य स्तर या परत हैं: –
   (i) Client
   (ii) Server Database.
   Two-tier Client – Server architecture User और Server पर आधारित है। ग्राहक और सर्वर के बीच सीधे संचार होता है ग्राहक और सर्वर के बीच कोई मध्यवर्ती नहीं है | Two-tier architecture User इंटरफ़ेस क्लाइंट पर चलता है और Database Server पर संग्रहीत रहता है।

2. **Three-Tier Client / Server Architecture:** American National Standards Institute (ANSI) या अमेरिकी राष्ट्रीय मानक संस्थान और Standards Planning and Requirements Committee (SPARC) ने 1975 में Database Architecture का उत्पादन किया।
   Three-Tier DBMS architecture के तीन स्तर या परत निम्न हैं: –

i. **External Level (बाहरी स्तर):**- यूज़र इस स्तर पर कार्य करते हैं और वे इस स्तर से परे डेटाबेस के किसी भी अस्तित्व के बारे में कुछ नहीं जानते हैं। इस परत पर, डेटाबेस के कई आवेदन द्वारा Database में जमा डाटा को देख सकते है | User डाटाबेस को कैसे देखना

चाहता है ये पूरी तरह से उसी पर निर्भर करता है पर यूजर को क्या क्या देखना है और क्या नहीं ये DBA तय करता है | User का इसपर कोई control नहीं होता है |

ii. **Conceptual Level (संकल्पनात्मक स्तर):**- यह स्तर External Level और Internal Level के बीच में स्थित होता है | User Conceptual से परे डेटाबेस के किसी भी अस्तित्व से अनजान हैं। दूसरे छोर पर, Internal Level किसी भी उपयोगकर्ता के बारे में जानकारी से अनजान हैं। इसलिए, Conceptual Level मध्य में बैठता है और External Level और Internal/ physical Level के मध्य मध्यस्थ के रूप में कार्य करता है।

iii. **Internal/ physical Level (आंतरिक/ भौतिक स्तर):**– इस परत में वास्तविक डाटाबेस तस्वीर में आता है। पूरी डेटाबेस इसी स्तर पर physical जमा होती है |

DATABASE MODEL (डेटाबेस मॉडल)

Database मॉडल एक प्रकार का डेटा मॉडल है जो डेटाबेस के logical ढांचे को निर्धारित करता है और मूल रूप से यह निर्धारित करता है कि किस तरह से संग्रहीत और उपयोग किया जा सकता है। Database model Data के विभिन्न हिस्सों के बीच संबंधों का वर्णन करता है। Database का मॉडल ही यह तय करता है कि database का डिजाइन किस प्रकार का होगा, मतलब की डेटाबेस के मॉडल को आधार बनाकर ही डेटाबेस का नक्शा तैयार किया जाता है|Data base model यह define करता है, कि data को database में किस प्रकार organize किया गया है। हर database model कुछ नियम(Rule) तथा standard को follow करते हैं।

डेटाबेस मॉडल के प्रकार :-

1. Relational model
2. Hierarchical database model
3. Network model
4. Object-oriented database model
5. Entity-relationship model

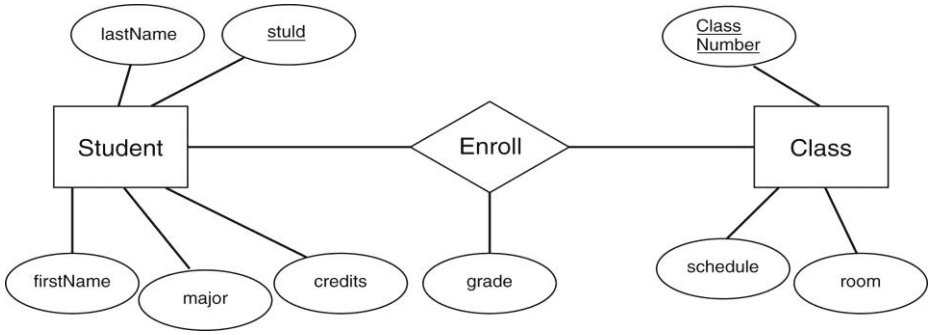
ऊपर लिखे डाटाबेस मॉडल में से आप किस मॉडल का चुनाव करेंगे यह बात कई चीजों पर निर्भर करता है लेकिन किसी मॉडल को चुनने का सबसे प्रमुख कारण है आपके Software की आवश्यकता मतलब की आपके Software की आवश्यकता ही यह सुनिश्चित करती है कि आपको किस database मॉडल को अपने Software के लिए चुनाना है | इसके साथ ही आप database model को अपने प्रोजेक्ट में शामिल करना चाहते हैं उस डेटाबेस मॉडल की गति और उस डाटाबेस मॉडल को आपके प्रोजेक्ट में शामिल करने से आपके प्रोजेक्ट में खर्च होने वाले रुपए और समय भी किसी database model को चुनने और ना चुनने का कारण बन सकता है | Database model में सबसे ज्यादा उपयोग में आने वाला database model रिलेशनल मॉडल (Relational model) है| जैसे कि MS SQLServer, IBM DB2, Oracle, MySQL, and Microsoft Access.

1. Relational model (रिलेशनल मॉडल):

Relational model को E.F Codd द्वारा पेश किए गए था| रिलेशनल मॉडल सबसे सामान्य मॉडल है| रिलेशनल मॉडल में डाटा को Table (तालिकाओं) में जमा किया जाता है| हर Table (तालिकाओं) एक दूसरे table से संबंधित होता है | जिसे Relation के रूप में भी जाना जाता है |हर टेबल columns और rows में विभाजित होता है| हर columns में Attribute को जमा किया जाता है और Rows में एक विशिष्ट उदाहरण के बारे में डेटा शामिल करता है |

इसमें डाटा को Tables के संग्रह के रूप में व्यवस्थित है किया जता है। टेबल्स में Row तथा Columns होते हैं। इन Tables को रिलेशन भी कहते हैं। टेबल या रिलेशन का Row Tupples कहलाता है जबकी Columns को Attributes कहा जाता है। इसमें trued Query Language के सॉफ्टवेयर का प्रयोग किया जाता है।

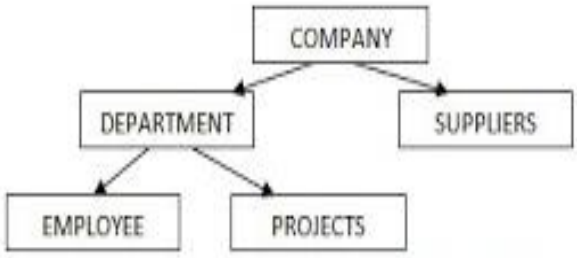
Relational database में data को two dimensional table में organized किया गया हैं। Relational database में हर row किसी record को दर्शायेगी। जिसे table कहा जाता हैं। टेबल की column जिन field को दर्शाती है उसे attribute कहा जाता है। Relational database में row को एक unique value दी जाती है। जिसे primary key कहते हैं।



Relational Data Model

2. Hierarchical database model (हिरार्चिकल डेटाबेस मॉडल):-

इस मॉडल में भी डाटा को रिकॉर्ड के संग्रह के रूप में व्यवस्थित किया जाता है, परंतु सभी रिकॉर्ड आपस में पदानुक्रम (Tree like structure) से जुड़े होते हैं। यह सबसे पुराना तथा लोकप्रिय डाटा मॉडल है।इसमें डाटा Data Manipulation Language सॉफ्टवेयर का प्रयोग किया जाता है। यह database का सबसे पुराना रूप है। जिसका निर्माण IBM ने अपने information management system के लिए तैयार किया था। यह model data को tree structure में organize करता है। जिसमें की एक parent node अन्य child node को बताती है। इस database में one - two - many relationship लगायी जा सकती है।



Hierarchical database model

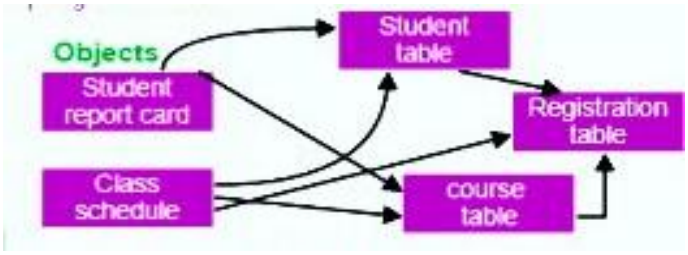
3. Network database model (नेटवर्क डेटाबेस मॉडल):-

इस मॉडल में डाटा को Records के ग्रुप के रूप में व्यवस्थित किया जाता है। रिकॉर्ड आपस में Links के द्वारा जुड़े होते हैं। यह model को database task group ने सन 1971 में बनाया। यह model हमे one to one , one to many , व many to many relationship को दर्शाने में कारगर है।

Network Database model

4. Object oriented database model (ऑब्जेक्ट ओरिएंटेड डेटाबेस मॉडल):-

Relational database में एक database को कई सारे server's पर distributes नही किया जा सकता है। जिसके कारण object oriented database management system का निर्माण किया गया। इस model में user अपने अपने data access के methode define कर सकते हैं। यह future database model है।



Object oriented database model

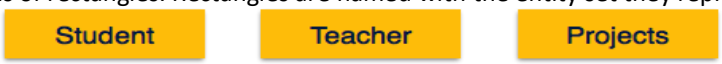
Entity Relationship Diagram (E-R Model/Diagram)

The Entity Relationship (E-R) data model is based on concept of real world that consists of a collection of basic objects called 'entities' and relationship among the object. The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

1. Entity

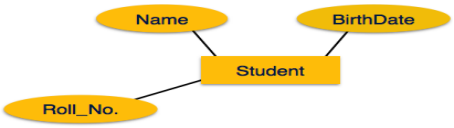
An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity. An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.



2. Attributes

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes. Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



Entity-Set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set. For example, the roll_number of a student makes him/her identifiable among students.

Primary Key – A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Super Key – A set of attributes (one or more) that collectively identifies an entity in an entity set.

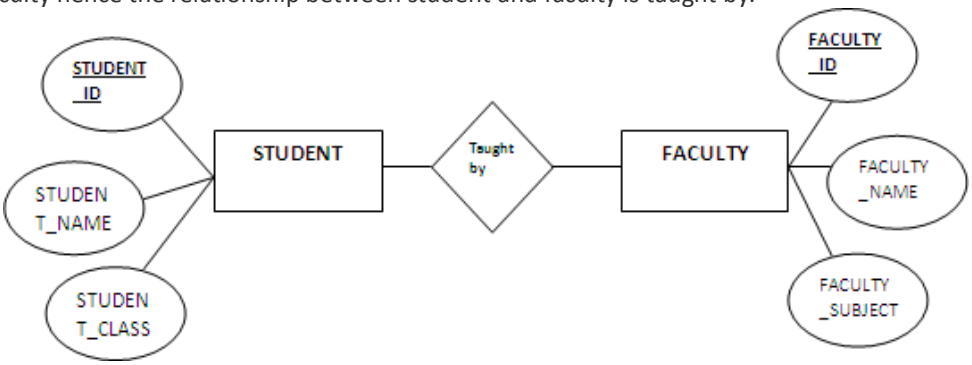
Candidate Key – A minimal super key is called a candidate key. An entity set may have more than one candidate key.

3. Relationship

The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships. Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

Example of ER Diagram:

In this example student and faculty are entity. Studentid, student_name, student_class are attributes of student entity. In this studentid is primary key. In faculty entity facultyid, faculty_name and faculty_subject are attributes and facultyid is primary key. Students taught by faculty hence the relationship between student and faculty is taught by.



Student faculty ER Diagram

Normalization in DBMS

Normalization को सर्वप्रथम 1972 में कोड द्वारा प्रतिपादित किया गया था, Database normalization रिलेशनल स्कीमा को उनके फंक्शनल डिपेंडेंसी और प्राइमरी- की के आधार पर विशलेषण करने की प्रक्रिया है|

Normalization Benefits in Database - इसका मुख्य उद्देश्य निम्न्न विशेषताओ को प्राप्त करना है

- Redundancy Minimizing
- Performance Improvement
- Query Optimization
- Minimizing Insertion, Deletion, Modification Anomalies

Edgar F. Codd ने तीन नॉर्मल फॉर्म को प्रतिपादित किया जो 1NF, 2NF and 3NF कहलाते है, 3NF (Third Normal Form) की ठोस परिभाषा को – Boyce-Codd Normal Form (BCNF) Boyce और Codd द्वारा 1974 में प्रतिपादित किया गया था| डिजाइनिंग डाटाबेस, सिस्टम

एनालिसिस और डिजाइन तकनीक के भाग की तरह डाटा एनालिसिस को सम्मिलित किया गया है, डाटा एनालिसिस एक प्रोसेस का यूज़ करती है जो Normalization कहलाती है जो एन्टिटी और सिस्टम से प्रचुरता को कम करती है, अत डाटा संघटन की भविष्य की आवश्यकताओ के लिए डाटा मॉडल को बनाती है|

Normalization एक एसी तकनीक है जो Transactional Database and Data warehousing के लिए सारणी के घटकों को संगठित करती है|नॉर्मल फॉर्म का उपयोग के सुनिश्चित करने के लिए किया जाता है की अनियमितता के विभिन्न प्रकार और असंगतता डाटाबेस में प्रदर्शित नही की जा सकती है|

Functional Dependency

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute. If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X->Y, which specifies Y is functionally dependent on X. Here X is a determinant set and Y is a dependent attribute. Each value of X is associated with precisely one Y value. Functional dependency in a database serves as a constraint between two sets of attributes. Defining functional dependency is an important part of relational database design and contributes to aspect normalization.

References:

1. "An Introduction to Database Systems" by Bipin Desai
2. "Fundamentals of Database Systems" by R Elmasri and S Navathe
3. "Database Management Systems" by Raghu Ramakrishnan
4. ***https://www.web3tutorial.com***
5. ***https://hi.wikipedia.org/wiki***
6. ***https://computerhindinotes.com/data-base-management-system/***
7. *https://www.tutorialspoint.com/dbms*

Microsoft Access (MS Access)

माइक्रोसॉफ्ट एक्सेस (Microsoft Acess) एक **डेटाबेस प्रबन्धन प्रणाली** है, जिसकी रचना और वितरण **माईक्रोसॉफ्ट** ने अपनी **माइक्रोसॉफ्ट विन्डोज़** और मैक ओएस एक्स के लिए किया है। यह अनुप्रयोग **माईक्रोसॉफ्ट ऑफिस** का एक भाग है। इसका उपयोग **डेटाबेस** बनाने, डेटाबेस को मैनेज करने, विश्लेषण करने एवं रिपोर्ट बनाने में किया जाता है।

MS Access एक डेटाबेस सॉफ्टवेर है, जिसकी मदद से हम डाटा एंट्री का अपना खुद का प्रोग्राम बना सकते हैं| डाटाबेस डेवलपमेंट के लिए MS Access एक महत्वपूर्ण और शक्तिशाली प्लेटफार्म है| MS Access माइक्रोसॉफ्ट कंपनी द्वारा तैयार किया गया एक डेटाबेस एप्लीकेशन सॉफ्टवेयर है जिसके माध्यम से किसी भी कंपनी, संस्था आदि के बारे में हम ज्यादा से ज्यादा सूचनाओं को इकट्ठा कर सकते है तथा बाद में कोई भी विशिष्ट जानकारी जो की उस संस्था से सम्बंधित हो, उसे प्राप्त कर सकते है |

डाटा बेस (Database)

किसी विशेष उद्देश्य के लिए एक स्थान पर संग्रहित एक समान तथा आपस में संबद्ध डाटा को डाटाबेस कहा जाता है। डाटा को व्यवस्थित कर उससे विभिन्न सूचनाएं प्राप्त की जा सकती हैं तथा उनका आवश्यकतानुसार उपयोग किया जा सकता है। एक ही डाटाबेस का उपयोग एक से अधिक उद्देश्यों के लिए किया जा सकता है।

Microsoft Access file extensions:

Access Database (2003 and earlier) is .mdb. For example: ba.mdb
Access Database (2007, 2010, 2013, 2016) is .accdb. For example: shastri.accdb

डाटा प्रोसेसिंग (Data Processing)

किसी इनपुट को उपयोगी आउटपुट में बदलने के लिए गतिविधियों और प्रक्रियाओं का क्रम प्रोसेसिंग कहलाता है। डाटा प्रोसेसिंग डाटा (इनपुट)को उपयोगी सूचना (आउटपुट)में बदलता है।

MS Access को हम "collection of database " भी कह सकते है| **.accdb** – Access 2007 Database File के लिये उपयोगी होता है। **.mdb** – Microsoft Access Database के लिये उपयोगी होता है| इसके अंतर्गत निम्न ऑब्जेक्ट्स आते है|

Objects of MS Access

1. Table:

किसी भी RDBMS के अंतर्गत टेबल एक महत्वपूर्ण ऑब्जेक्ट होता है| इसके अंतर्गत डाटा को रो और कॉलम में अरेंज करते है| विभिन्न प्रकार के रिकार्ड्स के कलेक्शन को टेबल कहते है| "Table is a Collection of records "

2. Query:

किसी जानकारी को प्राप्त करने के लिए पूछा गया प्रश्न क्वेरी कहलाता है| एक डेटाबेस के अन्दर एक से अधिक टेबल्स होती है, इन टेबल्स के मध्य रिलेशनशिप create करके सूचना को प्राप्त किया जा सकता है| टेबल के अन्दर कुछ सूचनाये इस तरह की भी होती है जिनकी आवश्यकता हमे बार -बार होती है, इसके लिए हमे उस टेबल को बार -बार प्रयोग करना होता है| MS Access में किसी टाइप के अन्दर से सिलेक्टेड फ़ील्ड्स की सूचना को हम अलग एक टेबल के रूप में रख सकते है, तथा जब भी हमे उस सूचना की आवश्यकता होती है हम अलग से टेबल के रूप में रखी गई उस इनफार्मेशन का प्रयोग कर सकते है | MS Access में यह queries के द्वारा implement किया जाता है |

3. Form:

यह MS Access का तीसरा और महत्वपूर्ण ऑब्जेक्ट है इसके द्वारा हम डाटा को डेटाबेस में स्टोर करा सकते है, प्रिंट करा सकते है लेकिन मुख्यतः MS Access मे form का प्रयोग इनपुट के लिए किया जाता है।

4. Report :

MS Access में report एक ऑब्जेक्ट है जिसका प्रयोग आउटपुट को प्राप्त करने के लिए किया जाता है | रिपोर्ट डेटा को संगठित तरीके से display और print करने के लिए उपयोग की जाती है। रिपोर्ट काफी लचीली होती है इसमें हम एक सामान डेटाबेस को कई लेआउट और फॉर्मेटिग में देख सकते है |

5. Macros:

मैक्रो एक टूल है जो कार्यों को स्वचालित करता है, और फॉर्म, रिपोर्ट और कंट्रोल्स में कार्यक्षमता जोड़ने की अनुमति देता है। उदाहरण के लिए, यदि आप किसी फॉर्म में एक कमांड बटन जोड़ते हैं, तो आप बटन के ऑनक्लिक ईवेंट को मैक्रो में जोड़ते हैं| मैक्रो में उन कमांड्स को शामिल किया जाता है, जिन्हें आप हर बार क्लिक करने के दौरान शामिल करना चाहते है। इस तरह एक बटन पर क्लिक करते ही ही सारे कमांड रन हो जाएगे| मैक्रो हम किसी भी फॉर्म,टेबल ,रिपोर्ट ,आदि में ओपन कर सकते है |

6. Page:

Page का आशय डेटाबेस को वेबपेज फॉर्मेट में अरेंज कर के देखने के लिए होता है इसका संबंध डेटाबेस मैनेजमेंट से नहीं होता |

7. Module:

मॉड्यूल, यूजर द्वारा डिफाइन किये गए functions, subroutines, और VBA code में लिखे गए global variables का समूह है। इन ऑब्जेक्ट का उपयोग एक्सेस डाटाबेस में कहीं से भी किया जा सकता है।

Database Design in MS Access

डेटाबेस डिजाइन एक डेटाबेस के विस्तृत डेटा मॉडल को प्रोड्यूस करने की प्रक्रिया है| डेटाबेस डिजाइन टर्म एक संपूर्ण डेटाबेस सिस्टम के डिजाइन के बहुत सारे भिन्न भागों का वर्णन करने के लिए प्रयोग की जाती है डेटाबेस डिजाइन के मुख्य टाइप निम्नलिखित हैं

1. Gathering and Analysis Design
2. Conceptual design
3. Logical design
4. Physical design

1. Gathering and analysis Design

डेटाबेस डिजाईन प्रक्रिया का सबसे मुख्य भाग आवश्यकता अनुसार जानकारी को इकट्ठा करना हैं जो डेटाबेस तैयार करना हैं उसके बारे में जानकारी को ढूढना फिर उन जानकारियों को इकट्ठा करना ही डेटाबेस डिजाईन का पहला चरण हैं

i. जहाँ का भी डेटाबेस तैयार करना हैं वहां खड़े होकर observe करना अर्थात यह देखना कि वहां क्या क्या हो रहा हैं किस तरह कार्य किया जा रहा है कार्य को करने की क्या प्रक्रिया हैं आदि |

ii. Observe करने के बाद उस व्यक्ति का Interview लेना जिससे जानकारी प्राप्त करना हैं |

iii. आवश्यकताओं का संकलन साधारण: संगठन के यूजरों से पूछताछ करके किया जाता हैं बड़ी संस्थाओ में यह कार्य काफी समय लगाने व थकाने वाला होता हैं इसलिए सिर्फ एक ही व्यक्ति से पूछताछ की जाती हैं परन्तु इससे डाटा का जानकारी अधूरी प्राप्त हो पाती हैं|

iv. जानकारी को इकट्ठा करने का एक दूसरा तरीका प्रश्नावलियों का उपयोग हैं अनुभव द्वारा प्रभावशाली प्रश्नवली बनाकर उन्हें लोगों में बाँट दिया जाता हैं उनके उत्तर फॉर्म के रूप में एकत्र कर लिए जाते हैं

2. Conceptual design

डेटाबेस से सम्बंधित जानकारों को एकत्रित करने के बाद उस डाटा पार विचार किया जाता हैं कि जो जानकारी एकत्रित की गई हैं वह सही हैं या नहीं| एक बार डेटाबेस डिजाइनर उस डाटा के बारे में जानकारी प्राप्त कर लेता है जो डेटाबेस में स्टोर होनी है तब वह यह निर्धारित करता है कि कहां पर डाटा में निर्भरता है कभी-कभी जब data बदल जाता है तो वे अपने आप दूसरे डेटा को बदल सकते हैं जो उचित नहीं होता है जैसे नाम तथा एड्रेस की

एक लिस्ट में एक स्थिति की कल्पना करते हैं जहां एक से अधिक व्यक्ति एक जैसे एड्रेस को रख सकते हैं लेकिन एक व्यक्ति एक से अधिक एड्रेस को नहीं रख सकता नाम एड्रेस पर निर्भर करता है क्योंकि यदि एड्रेस भरना है तो उससे संबंधित नाम भी भिन्न होता है अतः एक एट्रिब्यूट बदला जा सकता है दूसरा नहीं बदला जा सकता|

Conceptual design के टूल्स:-

i. ER Diagram
ii. Relationship
iii. Normalization
iv. Anomaly check

3. Logical Design (लॉजिकल डिजाइन)

डेटाबेस का दूसरा चरण लॉजिकल डिजाइन होता है डेटाबेस डिजाइन करते समय आवश्यक जानकारियों को इकट्ठा करने के बाद उस इंफॉर्मेशन को अलग-अलग बांट कर Data का लॉजिकल स्ट्रक्चर तैयार किया जाता है| लॉजिकल डिजाइन डेटाबेस के विस्तृत वर्णन के बारे में सोच है जैसे डेटाबेस में क्या इंफॉर्मेशन स्टोर होगी किस टाइप की इंफॉर्मेशन स्टोर होगी आदि|

4. Physical Design (फिजिकल डिजाइन)

डेटाबेस का तीसरा चरण डेटाबेस डिजाइन को फिजिकली डिजाइन करना है अर्थात आवश्यक सूचनाओं को इकट्ठा करने उनके बारे में विस्तृत जानकारी निकालने के बाद उस डाटा को भौतिक रूप से प्रयोग में लाया जाता है| डेटाबेस की फिजिकल डिजाइन स्टोरेज मीडिया पर डेटाबेस के फिजिकल कॉन्फ़िगरेशन को specify करता है फिजिकल डिजाइन डेटाबेस के सबसे नीचे के लेवल के डेटा का वर्णन करने के लिए प्रयोग किया जाता है फिजिकल डिजाइन के अंतर्गत डेटाबेस के वास्तविक स्ट्रक्चर को डिजाइन किया जाता है|

Data Types of MS Access

डाटा की प्रकृति के आधार पर डाटा कई प्रकार का होता है। एम एस एक्सेस मे डाटा टाईप निम्न प्रकार के होते है-

1. Text – इस प्रकार के डाटा मे (mathematical calculation) गणितीय गणनायें नही की जा सकती है। इसकी रेंज 0 To 255 अक्षर की होती है। अर्थात इस डाटा टाईप के फील्ड मे अधिकतम 255 अक्षर लिखे जा सकते है। उदाहरण- name, city, address etc.

2. Number – इस प्रकार के डाटा मे (mathematical calculation) गणितीये गणनायें की जा सकती है। इसके फील्ड मे नंबर को स्टोर किया जाता है। इसको निम्न भागो मे बाॅटा गया है। जैसे Mark, Principle (मूल्य घन), Rate (दर), Time (समय) etc.

3. Date and Time – इसमे डेट एवं समय को स्टोर किया जाता है। इसके निम्न प्रकार के फार्मेट हेाते है।

Date Type	Date Format	Example
General	MM/DD/YYYY Time	11/4/2017 7:18:11 PM
Long Date	MM/DD/yyyy, month ,day, yyyy	Friday, November 04, 2017
Medium Date	dd-Month-yy	4-Nov-17
Short	MM/DD/YYYY	11/4/2017
Time Format		

Long Time	HH:MM:SS PM /AM	7:19:15 PM
Medium Time	HH:MM:SS	7:19:29
Short Time	HH:SS	7:19

4. Currency – इस प्रकार के डाटा मे करंसी,पैसा केा स्टोर किया जाता है। इसमे गणतीये गणनाये भी कर सकते है। प्रत्येक देश की अपनी करेंसी होती है। एवं उसका एक चिन्ह होता है। जैसे.4500, 520000 etc.

5. Memo – यह एक विशेष प्रकार का डाटा टाईप है। इसके टैक्ट को स्टोर करने की कोई सीमा नही होती है। इसका प्रयोग तब किया जाता है। जब किसी के बारे मे ज्यादा जानकारी स्टोर करनी हो।

6. OLE Object – इसका पूरा नाम Object linking embedding है। इसके किसी भी फाईल को लिंक कराया जा सकता है। जिस पर क्लिक करके खोला जा सकता है। OLE object मे लिंक करना.

Insert menu→ Object→ Insert object dialog box→ create form file→ select file→ ok

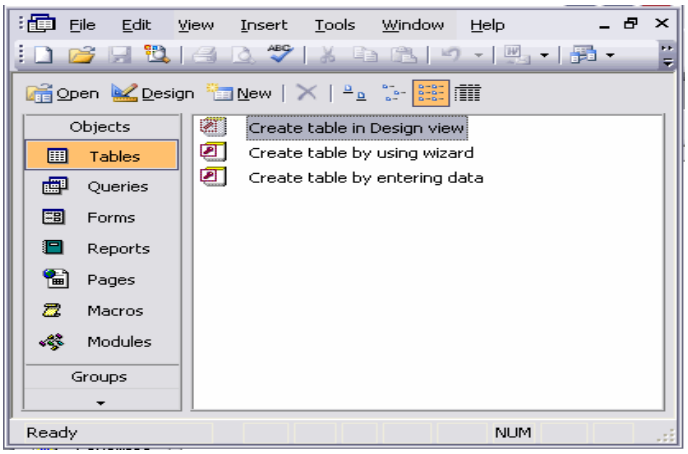
7. Logical – इसमे लाॅजिकल डाटा को स्टोर किया जाता है। जिसके केवल दो आॅपशन होते है।

Yes/No
True/false
No/Off

8. Auto number- इस डाटा टाईप से सीरियल नंबर अपने आप आते है। इसका प्रयोग सीरियल नंबर को स्टोर करने के लिये किया जाता है।

Create a table in MS Access (एम एस एक्सेस में टेबल का निर्माण करना)

MS Access मे डाटा को स्टोर करने के लिये टेबिल का निर्माण करना होता है। टेबिल डाटाबेस फाईल के अंदर हेाती है। एक डाटाबेस फाईल के अंदर एक से अधिक टेबिल हो सकती है। टेबिल का निर्माण रो एवं काॅलम से मिलकर हेाता है। फील्ड मे डाटा टाईप को सेट किया जाता है।



First step-

Go to file menu→ new→ click on blank database→ insert file name→ click on create button

Second step-

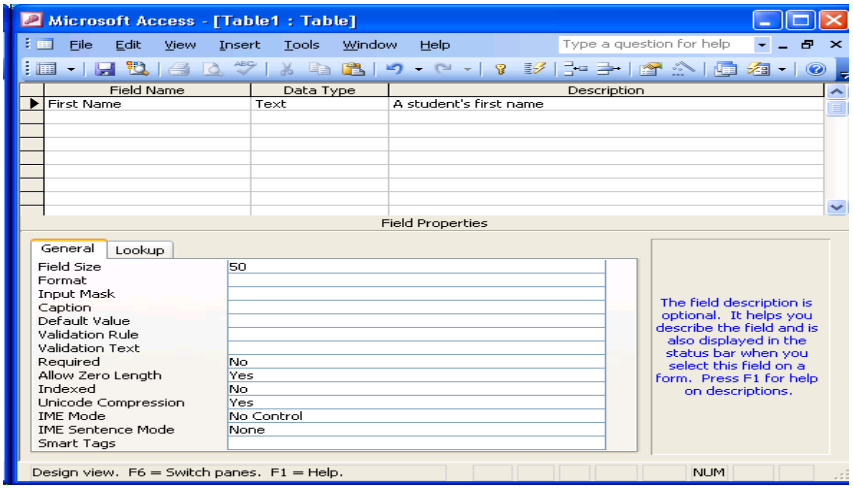
select table object इसमे तीन प्रकार से टेबिल को बनाया जा सकता है।

1. Create a table in design view:-

इसमे टेबिल को यूजर के द्वारा डिजाइन किया जाता है। इसमे फील्ड का नाम देते है। और उसके डाटा टाईप को सिलेक्ट करते है और उस फील्ड की प्रॉपर्टी को सेट करते है।

Design view मे चार प्रकार से जा सकते है।

1. डिजाईन व्यू आप्शन पर डबल क्लिक करके Design view मे जा सकते है।
2. सिलेक्ट करके Design Button पर क्लिक करके Design view मे जा सकते है।
3. Right click on design view → design view मे जा सकते है।
4. Open Option पर क्लिक करके Design view मे जा सकते है।



Working with Table in MS Access

MS Access में हम एक टेबल में निम्नलिखित कार्य कर सकते है

- Add Record
- Delete Record
- Edit Record
- Sort Record
- Find and Replace
- Filter & Select

1. Add Record in a Table:- पहले से बनी हुयी Table में नया Record जोडने के लिए उस Table को Open करते हैं, और रिकॉर्ड को जोड़ते है| टेबल में नया रिकॉर्ड जोड़ने के लिए निम्नलिखित स्टेप्स को फॉलो करेंगे |

- यदि Table में नए Record को जोड़ना हैं तो सबसे पहले उस Table को Select करते हैं,जिसमे रिकॉर्ड जोड़ना है |
- अब Table Screen पर Display हो जाएगी। इसमें Last record पर Mouse के Curser को Point करते हैं तथा Mouse का Right Button Click करते हैं|
- जिससे एक Popup Menu Display होता हैं इसमें Add Record Option पर Click करते हैं तो Curser Last Record के First Field में पहुँच जाता हैं अब इसमें हम नए Record को Add कर सकते हैं।

2. Delete a Record:- किसी Table में यदि किसी Record की आवश्यकता नही होती हैं अर्थात जो अनावश्यक Record होते है उन्हें डिलीट कर दिया जाता है |अगर किसी रिकॉर्ड को टेबल से हटाना है तो निम्नलिखित स्टेप्स को फॉलो करेंगे |

- सबसे पहले जिस Record को Delete करना हैं उस उस Record को select करते हैं तथा Mouse का Right Button Click करते हैं
- इसके Delete Record Option पर Click करते हैं ऐसा करने पर एक Message Display होता हैं।
- यह Message हमसे Record को Delete करने के बारे में पूछता हैं। कि आप इसे Delete करने के लिए तैयार है या नही।
- जब हम इस Message Box में Ok पर Click करते हैं तो Select किया हुआ Record Delete हो जाता हैं।

3. Edit Record:- यदि हम Table में हम किसी Record को संशोधित करना चाहते है तो इस कार्य को आसानी से कर सकते हैं।

Example:- यदि हमारे पास एक Employee नाम की Table है और उसमे हम किसी Employee के Record में Employee Name में Editing करन चाहते हैं तो निम्नलिखित स्टेप्स को फॉलो करेंगे |

- सबसे पहले वह टेबल ओपन करे जिसमे सुधार करना है |
- इसके बाद Table में Record को Select करते हैं ,इसके बाद उस Field को Select करते हैं जिस Field में सुधार करना हैं।
- इसके बाद उस Field में संशोधन करके उस Table को Save कर देते हैं।

4. Sort Record:- हम Table में किसी भी Field को Sort कर सकते हैं। Sorting से तात्पर्य किसी Field को Ascending या Descending Order में arrange करना होता हैं। रिकॉर्ड को Ascending या Descending Order में arrange करने के लिए निम्नलिखित स्टेप्स को फॉलो करेंगे |

- किसी Field को Sorting करने के लिए उस Field को Select करते हैं |
- इसके बाद Record Menu पर Click करते हैं |
- इसके बाद Sort Sub Menu पर Click करते हैं |
- यहाँ Sorting order (Ascending or Descending) को चुनते हैं। ऐसा करने पर Select की गयी Field उस Sorting order में व्यवस्थित हो जाती हैं जिसे आपने चुना है ।

5. Find and Replace:- Table में किसी Particular Field में किसी विशिष्ट Value का पता लगाने के लिए Find Option का उपयोग करते हैं।

Example:- Employee Table में हजारों की संख्या में Record हैं उनमें से किसी Particular Employee के नाम का पता लगाना हैं तो इसके लिए Find Option का Use करते हैं। रिकॉर्ड को Find करने के लिए निम्नलिखित स्टेप्स को फॉलो करेंगे |

- Table में किसी Field में Particular Value पता करने के लिए सबसे पहले Table को Open करते हैं |
- इसके बाद Edit Menu में Find Menu पर Click करते हैं |
- इससे Find & Replace Window Display होगी |
- इसमें Find What Box में वह Value Type करते हैं जो Find करनी हैं। Curser उस Field की Value पर Highlight हो जाता हैं |
- यदि Same Text फिर से Find करना हो तो Find Next Button पर Click करते हैं अन्यथा Find & Replace Window को बंद कर देते हैं।

6. Replace:- किसी Record की Value को नयी value से बदलने के लिए Replace Option का Use करते हैं। Replace करने के लिए निम्न स्टेप्स को फॉलो करेंगे |

- Table में किसी Field में कोई Value Replace करने के लिए Replace Sub Menu पर Click करते हैं |
- Find & Replace Window Open हो जाती हैं इस Window में Find वाले Box में Source Name (जिस नाम को बदलना हैं) लिखते हैं। तथा Replace With Box में Destination Name (जो नाम लिखना हैं) लिखते हैं तथा Replace Button पर Click करते हैं तो पुराना नाम नए नाम के साथ Replace हो जाता हैं |
- यदि Find वाले Box में लिखे गए नाम की जगह पूरी Table में Replace With Box में लिखे गए नाम को लिखना हैं तो Replace All Button पर Click करते हैं।

Data Backup (बैकअप)

बैकअप का अर्थ होता है अपनी फाईलो की कॉपी अपने मुख्य कंप्यूटर की हार्ड डिस्क के अतिरिक्त किसी अन्य संग्रहण इकाई जैसे मैग्नेटिक टेप ,पेनड्राइव या नेटवर्क में किसी अन्य कंप्यूटर में भी रखने से है ताकि अगर दुर्घटनावश हमारी मूल फाईले नष्ट हो जाये तो उन्हें इन अतिरिक्त प्रतियों द्वारा पुनः प्राप्त किया जा सके | हमारी फाईलो की ये अतिरिक्त प्रातियाँ बैकअप कहलाता है बैकअप हमारी फाईलो की सुरक्षा का अंतिम उपाय है |

Recovery Procedures (डाटा रिकवरी)

कंप्यूटर एक मशीन है इसलिए यह संभव है कि कभी भी कंप्यूटर में कोई हार्डवेयर या सॉफ्टवेयर संबंधित समस्या उत्पन्न हो जाए ऐसे में यह बहुत जरूरी है की कंप्यूटर में किसी प्रकार की समस्या उत्पन्न होने पर उसमें रखें डेटाबेस को हम Recover कर पाएं DBMS में यह काम बड़ी आसानी से किया जा सकता है | डेटा रिकवरी याने नष्ट हुआ डेटा रिस्टोर करने की प्रोसेस है| यह डेटा रिकवरी प्रोसेस डेटा नष्ट होने की परिस्थितियों के आधार पर भिन्न हो सकती हैं|

Accidentally delete of File or folder:

जब आप कोई फाइल को डिलीट करते है, तो इसे ड्राइव से तुरंत हटाया नही जाता, लेकिन डिलीट का मार्क लगा दिया जाता है और वे ड्राइव में ही रहते है जबतक किसी अन्य फाइल व्दारा ओवरराइट नही किया जाता| इस दौरान मूल फाइल कई बार डिस्कनेक्टिड फ्रैग्मन्ट के रुप में फाइल वैसे ही रहती है और रिकवर हो सकती है|

Damage File system format:

फाइल सिस्टम डिस्क या पार्टीशियन में फाइल्स का ट्रैक रखने के लिए ऑपरेटींग सिस्टम की एक मेथड और डेटा स्ट्रक्चर है| इसे वायरस या गलत निर्देश से नुकसान पहंच सकता है| सक्षम डेटा रिकवरी सॉफ्टवेयर क्रैश फाइल सिस्टम से डैमज्ड पार्टिशन से डेटा रिकवर कर सकते है, जो उपलब्ध अलोकेशन जानकारी और क्षति की स्थिती पर निर्भय है| कई बार जब वही फाइल सिस्टम से फॉरमॅट किया हो तो डेटा रिकवरी की संभावना अधिक होती है|

Referential Integrity Rule

Relationship Window में Referential Integrity का Option होता हैं इसका अर्थ हैं कि यदि दो Table के मध्य Referential Integrity rule स्थापित हैं तो प्रथम Table में यदि किसी Record में संशोधन या Updationकरते हैं तो इससे संबंधित Table में स्वत: ही संशोधन हो जाता हैं इसी प्रकार यदि प्रथम table में से किसी record को delete करते हैं तो इससे संबंधित table में से वह record delete हो जाता हैं। Table की Relationship Create करते समय Edit Relationship Window में Referential Integrity के तीन Option होते हैं।

Data integrity

Data integrity is the maintenance of and the assurance of the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data. It is at times used as a proxy term for data quality, while data validation is a pre-requisite for data integrity. Data integrity aims to prevent unintentional changes to information. Data integrity is normally enforced in a database system by a series of integrity constraints or rules. Three types of integrity constraints are an inherent part of the relational data model: entity integrity, referential integrity and domain integrity:

1. **Entity integrity:** concerns the concept of a primary key. Entity integrity is an integrity rule which states that every table must have a primary key and that the column or columns chosen to be the primary key should be unique and not null.
2. **Referential integrity:** concerns the concept of a foreign key. The referential integrity rule states that any foreign-key value can only be in one of two states. The usual state of affairs is that the foreign-key value refers to a primary key value of some table in the database. Occasionally, and this will depend on the rules of the data owner, a foreign-key value can be null. In this case, we are explicitly saying that either there is no relationship between the objects represented in the database or that this relationship is unknown.

3. **Domain integrity:** specifies that all columns in a relational database must be declared upon a defined domain. The primary unit of data in the relational data model is the data item. Such data items are said to be non-decomposable or atomic. A domain is a set of values of the same type. Domains are therefore pools of values from which actual values appearing in the columns of a table are drawn.
4. **User-defined integrity:** refers to a set of rules specified by a user, which do not belong to the entity, domain and referential integrity categories.

Data Security (डाटा सुरक्षा)

DBMS में डाटा को पूरी तरह से Database Administrator (एडमिनिस्ट्रेटर) द्वारा नियंत्रित किया जाता है और डाटा बेस एडमिनिस्ट्रेटर ही यह सुनिश्चित करता है कि किस User को कितना Database के कितने हिस्से पर Access देना है या नहीं देना है इससे डेटाबेस कि सिक्योरिटी बहुत अधिक बढ़ जाती है |

Manage user-level security for an earlier-format database file

Note: Do not convert your database to one of the new file formats if you want to continue using user-level security. The user-level security features work only in databases that use an earlier Access file format, such as .mdb files.

- Open the database that has user-level security settings that you want to manage.
- On the **Database Tools** tab, in the **Administer** group, click **Users and Permissions**.
- Click one of the following commands:
- **User and Group Permissions** Use this to grant or revoke user or group permissions, or to change the owner of database objects.
- **User and Group Accounts** Use this to create or delete a user or a group, to change the password or the group membership of a user, or to change the database Logon password.
- **User-level Security Wizard** Use this to start the Security Wizard, which makes an unsecured backup copy of your database and guides you through the process of implementing user-level security features.

Other security features

For better security, consider using one or more of the following features:

1. **Encryption** The encryption tool makes your data unreadable by other programs or tools, and it forces users to enter a password to use the database. The encryption tool is available only in databases that use one of the new file formats. In Access, click **File** > **Encrypt with Password**.
2. **Database server** Store your data on a database server that manages user security, such as Microsoft SQL Server. Then, use Access to build queries, forms, and reports by linking to the data on the server. You can use this technique on a database saved in any Access file format.
3. **SharePoint site** SharePoint provides user security and other useful features, such as working offline. There are a variety of implementation options, depending on which SharePoint product you use. Some SharePoint integration features are available only in databases that use one of the new file formats.

References:

1. Microsoft Office Access 2007: The Complete Reference (Complete Reference Series) Paperback by Virginia Andersen
2. Microsoft Access Data Analysis by Michael Alexander
3. *https://hi.wikipedia.org/wiki/*
4. ***https://computerhindinotes.com/database-design-in-ms-access/***
5. *https://computerhindinotes.com/create-a-table-in-ms-access/*