

RASHTRIYA SANSKRIT SANSTHAN, LUCKNOW CAMPUS

SHATRI 2ND YEAR (4TH SEMESTER)

SUBJECT-COMPUTER (PAPER-7)

NOTES PRACTICE QUESTION BANK



**PROVIDED BY
MR.VINAY KUMAR DWIVEDI
GUEST TEACHER (COMPUTER)
RASHTRIYA SANSKRIT SANSTHAN,
LUCKNOW CAMPUS**

C Programming Language

- C language एक general-purpose programming language है जिसे सन् 1972 में **Dennis Ritchie** के द्वारा **Bell Labs.** में develop किया गया था। **UNIX operating system** को develop करने के लिए C Programming Language को बनाया गया।
- C programming language को System application create करने के लिए develop किया गया था जो की directly Hardware devices जैसे की **Kernels, drivers etc.** के साथ आसानी से interact कर सके।
- C programming language एक **procedural** और **structured programming** language होता है। अर्थात यह एक प्रकार का programming language होता है जो की एक ऐसा process / procedure (प्रक्रिया) provide करता है जिससे किसी भी program को अच्छे तरीके से लिखा जा सके। अच्छे से लिखे जाने को ही well structured steps कहा जाता है और इसी वजह से इसे Structured programming language भी कहा जाता है।

C language के कुछ महत्वपूर्ण तथ्य – Important facts of C

- C language का आविष्कार **UNIX operating system** को develop करने के लिए हुआ था।
- यह language आज सबसे ज्यादा popular programming language है।
- यह language **B language** का successor है। यह programming language को **B, BCPL** जैसे प्रोग्रामिंग लैंग्वेज में होने वाले प्रॉब्लम को दूर करने के लिए develop किया गया।
- **Linux OS** और **RDBMS MYSQL** C language में लिखे गए हैं।
- C low level language और high level language दोनों के ही features को include करता है इसलिए इसे Middle Level Language कहा जाता है।

C program basic components

जब भी कोई बड़ा चीज बनाया जाता है तो उसमें ढेर सारे छोटे छोटे parts (components) लगे हुए होते हैं। ठीक उसी प्रकार जब हम C program बनाते हैं तो उसमें ढेर सारे छोटे छोटे parts मिलकर एक complete program बनते हैं जो की compiler के द्वारा compile होकर के user को program के अनुसार output मिलते हैं। निम्नलिखित basic component C program में मौजूद होते हैं:

- Pre-processor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

सबसे पहले एक program को देखना होगा जिससे हमें बहुत ही आसानी से इन सभी topics के बारे में समझ सके ।

```
#include

int main()
{
    /* First C program */
    printf("Hello, This is First C Program");

    return 0;
}
```

इस program को step by step समझते हैं:

- सबसे पहला लाइन में **#include <stdio.h>** लिखा हुआ है यह एक **preprocessor command** है जो की compiler को बताता है की actual compilation से पहले **stdio.h** नाम के header files को program में include करना है। Preprocessor का मतलब होता है सबसे पहले process होना। कोई भी C program compile होने से पहले वह पहले से define किये गए program के द्वारा compile होता है उसे **preprocessor** कहा जाता है। Preprocessor command **hash (#)** symbol से start होता है। stdio का full form "**Standard Input Output**" होता है जिसके अन्दर input और output से related सारे

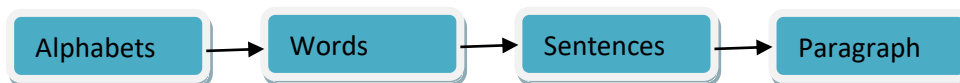
function को definition को define किया गया है। बहुत सी header files होती हैं ।

- दूसरी लाइन में **int main()** लिखा हुआ है। यहाँ पर **int return type** है जो यह बताता है की कैसा value return होगा। जबकि **main()** एक function है जहाँ से program का execution start होता है।
- तीसरी लाइन में **/* */** के अन्दर कुछ statement लिखे हुए हैं जिसे comments कहा जाता है। जब भी कोई program compile और execute होता है तो compiler comments को compile नहीं करता है। User अपने सुविधा के लिए यानि की किसी भी function को, variable को समझने के लिए comments लिखता है।
- Next line में **printf()** function है जिसके अन्दर कुछ message लिखे गए हैं। **printf() function** का इस्तेमाल screen पर output print करने के लिए होता है। इस function का definition **stdio.h** header files के अन्दर मौजूद होता है।
- उसके बाद **return 0;** लिखा गया है जो की **main()** function को 0 value return करता है और program के execution को terminate कर देता है।

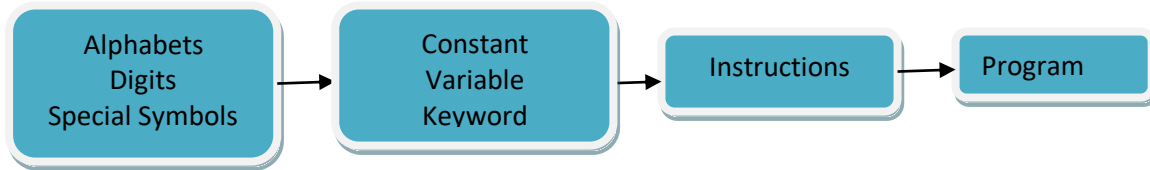
Note-source कोड से आउटपुट प्राप्त करने के लिए एक IDE (integrated development environment) जैसे turbo c/c++ आदि use किया जाता है । IDE में compiler भी होता है , जिसका कार्य source code से errors को दूर कर मशीन code में बदलना है ।

Steps in Learning C language-

Learning English Language Steps



Learning C Language Steps

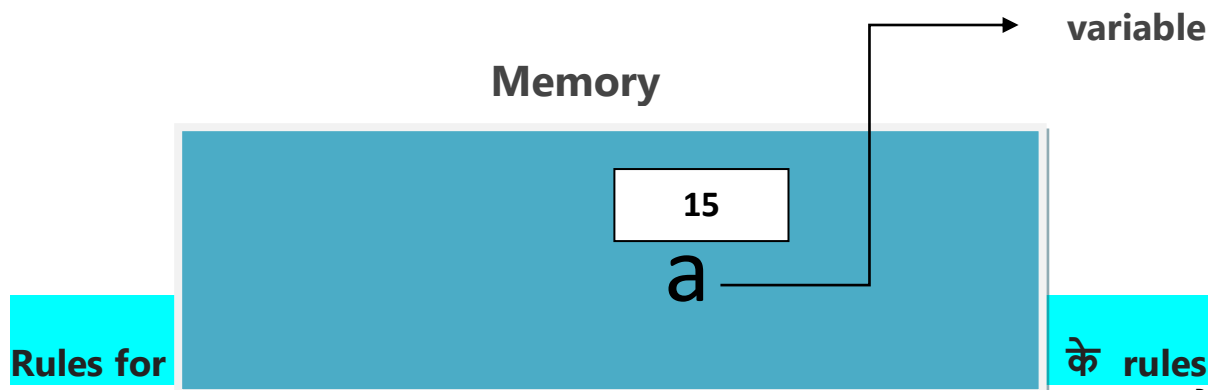


Variable in C-C programming में variable क्या होता है?

- Variable memory location का एक नाम होता है जिसका इस्तेमाल मेमोरी location को identify करने के लिए किया जाता है। जब भी हम program लिखते हैं तो उसके सारे value मेमोरी में store होते हैं और उस मेमोरी location को identify करने के लिए हम उस मेमोरी location को एक **unique name** provide करते हैं जिसे **variable** कहा जाता है।
- C programming में प्रत्येक variable को एक data type के साथ declare किया जाता है ताकि उस variable का type पता चल सके की वह variable कितना मेमोरी space allocate करेगा, उस variable का कितना range होगा, वह variable किस प्रकार का value store करेगा etc.
- Program के execution के दौरान variable का value change किया जा सकता है। जैसे:

```
int a = 15;  
//here value of a is changing from 15 to 20  
a = 20;
```

यहाँ पर **a** नाम का एक variable बनाया गया है जिसका type **int** दिया गया है यह केवल integer type के value को store कर सकता है। सबसे पहले variable **a** में 15 value store किया गया है उसके बाद उसका value **20** store कर दिया गया है।



- C programming में variable का नाम दन स पहल आपका कुछ बाते जरूर जाननी चाहिए जिससे आप गलत variable name create करने से बच सकें। यदि आप C programming में दिए गए variable naming rules

को follow नहीं करते हैं तो आपका program execute नहीं होगा that means आपके program में error आ जायेंगे| **C programming में variable name create करने के कौन कौन से rules हैं -**

- Variable name हमेशा **letter** या **underscore** symbol से start होना चाहिए जैसे **first, _second, firstname** Variable name **letters, digits और underscore** का combination हो सकता है लेकिन पहला character हमेशा letter या underscore (_) symbol ही होना चाहिए|
- Variable name **upper case** और **lower case** का combination हो सकता है लेकिन अगर आप एक variable का नाम upper case letter में देते हैं तो उस program में हर जगह उस variable को upper case letter से ही identify करना होगा क्योंकि variable name **case sensitive** होते हैं| जैसे **First, first** यहाँ पर दो अलग अलग variable create किये गए हैं|
- Keyword का नाम और variable का नाम same नहीं होना चाहिए that means keywords variable name के जैसा इस्तेमाल नहीं हो सकता है|
- variable name में **blank space** इस्तेमाल नहीं होना चाहिए|
- Variable का नाम कितना लम्बा होना चाहिए इसका कोई रूल नहीं है लेकिन कुछ compiler 31 character से ज्यादा variable name को accept नहीं करते हैं|
- Variable name में आप special symbol का इस्तेमाल नहीं कर सकते जैसे की \$, #

Note: हमेशा variable का meaningful name देने की कोशिश करें इससे other person आसानी से समझ पाते हैं की यह variable किस लिए उपयोग किया गया है जैसे **first number** के लिए **firstnumber** variable name दें ना की **fn**.

C **strongly type** programming language है जो की सभी variable के **data type** को strong तरीके से check करता है| आप एक बार जिस type का variable define कर दिए उसको बाद में दुसरे type में redefine नहीं का सकते हैं| For example:

```
int a = 10;
```

```
double a = 25.7; //error
```

ऊपर दिए गए code में **a** नाम का एक **integer** type variable declare और initialize किया गया था| उसके बाद उस **a** variable को **double** data type के साथ redefine किया गया है जिसके बाद compiler error provide करता है|

Declaring and Initializing variable in C

C program में किसी भी variable को इस्तेमाल करने से पहले उसे **declare** करना होता है| Declare का मतलब घोषित करना that means compiler को यह बताना की यह variable name हमने इस type का declare

किया है ताकि compiler उस variable से related आगे का process कर सके।

variable declaration syntax:

data_type variablename;

example:

int a;

Initialization का मतलब variable में value को store करना होता है। जैसे

a = 10;

सबसे पहले हमने **a** नाम का variable declare किया उसके बाद उसमें value store किया that means variable को **initialize** किया। C programming में variable by default **0** से initialize होता है।

C programming में आप variable को declare और initialize दोनों एक साथ कर सकते हैं। For example:

int a = 10;

Introduction of Constants

- C language में Constants वो variables होते हैं | जिनकी value change नहीं होती है। जब हम कोई constant declare करते हैं | तो उसकी value fixed रहती है। यदि इसकी value change करने की कोशिश की जाती है | तो program में error आ जाती है। यानी constant variables की value fix होती है | जो program के run time में भी इसकी value change नहीं की जा सकती है | Constant किसी भी data type का हो सकता है।
- Constant Pointer भी होता है |
C language में Constant दो प्रकार के होते हैं |

- Constant Literals
- Constant Variables

• C Language – Constant Literals

C language में Constant literals ऐसी values होती हैं जिन्हें हम program में directly use कर सकते हैं। अब निचे दिए गए code को देखिये।

x=y+5;

ऊपर दिए गए code में 5 एक constant literal है। इसे program में directly use कर सकते हैं। जब हम program run करेंगे तो इसका value change नहीं होगा |

Constant literals को हम integer नंबर ही मानते हैं | क्योंकि इसकी value changeable नहीं होती है | इसीलिए Constant literals को कहीं कहीं use किया जाता है |

- **Constant variables** वो variables होते हैं | जिसे हम खुद declare करते हैं | और Constant variables का सबसे बड़ा benefit यह होता है | अगर Constant variables की value change करना हो तो बड़े ही आसानी से कर सकते हैं | Constant variables को हम दो प्रकार से declare कर सकते हैं। इसका details निचे दिया जा रहा है |

a) pre processor (#define)

b) const Keyword

a) pre processor (#define)

C language में #define एक pre processor है | इसका use करके हम constant variables declare कर सकते हैं। #define pre processor के द्वारा constant variables main function के पहले ही create किये जाते हैं

| pre processor के द्वारा define किये गए constant variables को program में कहीं भी use कर सकते हैं। और जरूरत पड़ने पर इसका value भी change कर सकते हैं |

```
# include <stdio.h>
# include <conio.h>
/* this is constant variable num */
#define num 20
int main()
{
printf("Your Result is : %d",num);
return 0;
}
```

b) Using const Keyword

C language में constant variables declare करने के लिए हम const keyword का भी use कर सकते हैं |आप जैसे ही const keyword का use करेंगे आपका constant variables create हो जायेगा और हा constant variables को हम function में भी create कर सकते हैं और उसे use भी कर सकते हैं

```
int main()
{
const int a=10;
const int b=10;
int c;
c = a+b;
printf("Your Result is : %d",c);
return 0;
}
```

Tokens in C – Basic Syntax

- C program में सबसे छोटे unit को या सबसे छोटे part को tokens के नाम से जाना जाता है। एक C program ढेर सारे छोटे छोटे code से मिलकर बना होता है जिसमें ढेर सारे syntax use होते हैं जिसे Tokens के नाम से जाना जाता है। जैसे Keywords, Identifier, constant, symbols.
- example :- एक घर को बनाने के लिए ढेर सारे material की आवश्यकता पड़ती है जैसे की पानी, सीमेंट, लकड़ी इत्यादि। ठीक उसी प्रकार एक C program को बनाने के लिए ढेर सारे छोटे छोटे code की आवश्यकता पड़ती है जिसे Token के नाम से जाना जाता है।
`printf("Hello World");`
ऊपर दिए गए example में printf, (), "Hello World" ये सभी tokens हैं।

C tokens कुल छः प्रकार के होते हैं:

1. Keywords (eg: do, while, float)
2. Identifier (eg: total, main)
3. String (eg. "HELLO")
4. Constant (eg. 5, 8, 10, 12)
5. Special Symbols (eg: { }, ())
6. Operators (eg: +, -, *, /)

1. Keywords

C Compiler में पहले से define किये गए word को Keywords कहा जाता है। Keyword system के लिए reserved होता है जो की किसी specific काम के लिए बनाया गया होता है। Keywords को आप अपने variable name के तौर पर इस्तेमाल नहीं कर सकते हैं। C programming language में कुल 32 keywords होते हैं, जिनके list निचे दिए गए हैं।

auto	break	case	char	const	continue	default	Do
double	else	enum	extern	for	float	goto	If
Int	long	register	return	short	signed	sizeof	static
Struct	switch	typedef	union	unsigned	void	volatile	while

2. Identifier

किसी भी variable, function, array को identify करने के लिए उसका एक नाम दिया जाता है जिसे identifier कहा जाता है। जब हम C program में किसी भी variable,

function और array को कोई भी meaningful नाम provide करते हैं उसे Identifier कहा जाता है।

```
#include
int main()
{
int a, b, sum;
a = 10;
b = 20;
sum = a + b;
printf("Sum = %d",sum);
return 0;
}
```

ऊपर दिए गए program में main, a, b, sum, printf ये सब identifier हैं।

C programming में identifier name define करने के कुछ Rule :

- Identifier name हमेशा alphabet (a to z) या underscore (_) से शुरू होने चाहिए।
- identifier name में digit (0-9) भी रख सकते हैं लेकिन digit हमेशा alphabet या underscore symbol के बाद ही होने चाहिए। Identifier name के शुरू character में आप digit नहीं दे सकते हैं।
- Identifier name में कोई भी keywords इस्तेमाल नहीं होने चाहिए।
- Identifier name में space और white spaces नहीं होने चाहिए।
- identifier name में किसी भी प्रकार का special character और symbol नहीं हो सकते हैं केवल underscore (_) के अलावा
- Identifier name हमेशा case sensitive होने चाहिए जैसे की अगर आप किसी भी variable का नाम total declare करते हैं तो आप उसे Total लिख कर के access नहीं कर सकते हैं। यहाँ पर total और Total दो अलग अलग variable हैं।

कुछ identifier के नाम यहाँ पर दिए गए हैं पहले वाले टेबल में सही identifier के नाम हैं जबकि दूसरे वाले टेबल में wrong identifier के नाम हैं।

Total	sum_total	sum_9	abc99
abcname10	_sum	sumcheck	total200

Wrong Identifier Name

9total	@total	total@	total_123#
--------	--------	--------	------------

3. String

एक या एक से ज्यादा character के combination को जब double quote के अन्दर लिखा जाता है तब उसे string कहा जाता है। जबकि single letter (alphabets / digits) को जब single quote के अन्दर रखा जाता है तो उसे character कहा जाता है।

String example : "Ramesh" "Hi How are you". "123" यहाँ पर सभी example string है।

Character example: 'a', '2' यहाँ पर दोनों example character है।

4. Constant

Constant एक variable के जैसा ही treat होता है जिसमें एक fixed value store किया जाता है। यह value program के execution के समय कभी भी change नहीं होता है।

5. Special Symbol

Compiler के पास special symbol का special meaning होता है मतलब की सभी special symbol का अलग अलग definition होता है अलग अलग काम होता है। इसका उपयोग special task को perform करने के लिए किया जाता है।

Square brackets [], curly braces {}, parenthesis (), semicolon (;), comma (,) ये सभी special symbol हैं।

- **Brackets [] :** Opening Brackets और closing brackets का इस्तेमाल array element को define करने के लिए किया जाता है। Example: int arr[5];
- **Curly braces {} :** Curly braces का इस्तेमाल एक block का scope define करने के लिए किया जाता है।
- **parenthesis () :** Parenthesis का इस्तेमाल function को indicate करने के लिए किया जाता है। Parenthesis के अन्दर हम parameter भी भेज सकते हैं। eg: main()
- **Semicolon (;) :** Semicolon एक terminator होता है जिसका इस्तेमाल एक line को terminate करने के लिए होता है।
- **Comma (,) :** Comma एक separators होता है जिसका इस्तेमाल एक से ज्यादा statement को separate (अलग) करने के लिए किया जाता है। जैसे for loop में comma के द्वारा initialization, test condition और increment / decrement को separate किया जाता है।
- **Asterisk (*) :** यह एक special symbol होता है जिसका इस्तेमाल pointer variables को create करने के लिए किया जाता है। जैसे int *a;

6. Operators

Operator एक symbol होते हैं जो की किसी भी variables पर apply होने के बाद कुछ arithmetical और logical task को perform करते हैं। Operators को perform करने के लिए कुछ data या operands की जरूरत पड़ती है।

Operator को तीन categories में बाँटा गया है।

- **Unary operators:** Unary operators जैसे operators को कहा जाता है जो की single operand के साथ task perform करता है जैसे की increment decrement operator. Eg: a++
- **Binary Operators:** Binary operators जैसे operators को कहा जाता है जिसे task perform करने के लिए दो operand की आवश्यकता पड़ती है। जैसे: Arithmetic operator (+, -, *, /), Relational Operator (>, <, >=, <=) etc.
- **Ternary operators:** Ternary operators जैसे operators को कहा जाता है जिसे तीन operands की आवश्यकता पड़ती है। इसका इस्तेमाल decision making में किया जाता है। जैसे <condition> ? <true block> : <false block>;

Data types in C

C programming में variable और function को इस्तेमाल करने से पहले उसे declare करना होता है। Data type, variable और function के type और nature को specify करता है की ये किस प्रकार के data को store कर सकते हैं और कितना मेमोरी occupies करेंगे ।

Types of Data Type in C - C programming में Data Type निम्नलिखित प्रकार के होते हैं।

1. Basic data type
2. Derived data type
3. User defined data type

1) Basic Data Type

Basic data type के अंतर्गत integer, floating-point और char type आते हैं।

• **Integer (int)**

Integer data type whole number को store करने के लिए इस्तेमाल किया जाता है यह **zero, negative** और **positive values** को store कर सकता है जैसे की **0, 3, 6, -7, -9, -120**. यह data type decimal number (दशमलव संख्या) को (store नहीं करता है जैसे की **1.2, -2.3**

Integer variable को declare करने के लिए int **keyword** का उपयोग किया जाता है। For example:

```
int a;
```

यहाँ पर **a** एक variable name है जो की integer type का variable है। आप C programming में multiple variable को declare कर सकते हैं। Multiple variable को separate करने के लिए comma का इस्तेमाल किया जाता है। जैसे:

```
int x, y, z;
```

int data type का range **-32768 से 32767** या **-2147483648 से 2147483647** तक होता है। इसका range अलग अलग operating system के अनुसार होता है। जैसे **32-bit** operating system के लिए different range होता है और **64-bit** operating system के लिए different range होता है।

अलग अलग प्लेटफार्म पर यानि की अलग अलग operating system पर किसी भी data type और variable का size जानने के लिए आप **sizeof** operator का इस्तेमाल कर सकते हैं।

- **Floating-point types**

Floating-point types का उपयोग **real number** को store करने के लिए किया जाता है जैसे की **5.02, 5.364, -5.012** इस type के अंतर्गत **float, double** और **long double** data type आते हैं। Float data type का size **4 bytes** होता है, double data type का size **8 bytes** होता है जबकि long double data type का size **10 bytes** होता है। तीनों types अलग अलग प्रकार के precision को store करके रखते हैं। Precision का मतलब दशमलव के बाद अंक |

- **char**

char एक single character को store करने के लिए इस्तेमाल किया जाता है। Single character को single quote के अन्दर लिखा जाता है। character type के variable को define करने के लिए **char** keyword का उपयोग किया जाता है। Character variable का size **1 byte** होता है। जिसका range **-128 से 127 या 0 से 255** तक होता है।

declaration of character type variable:

char first = 'a';

- **Void Data Type in C**

void data type किसी भी प्रकार का value return नहीं करता है मतलब की यह no value available को specify करता है। जब हमें function से किसी भी प्रकार का value return नहीं करवाना होता है तो हम उस function का type void specify कर सकते हैं।

2) Derived data type in C

Derived data type का इस्तेमाल primary data type की मदद के द्वारा किया जाता है इसलिए इसे derived data type कहा जाता है। जैसे -

- Array type
- Pointer Type
- Structure type
- Union type

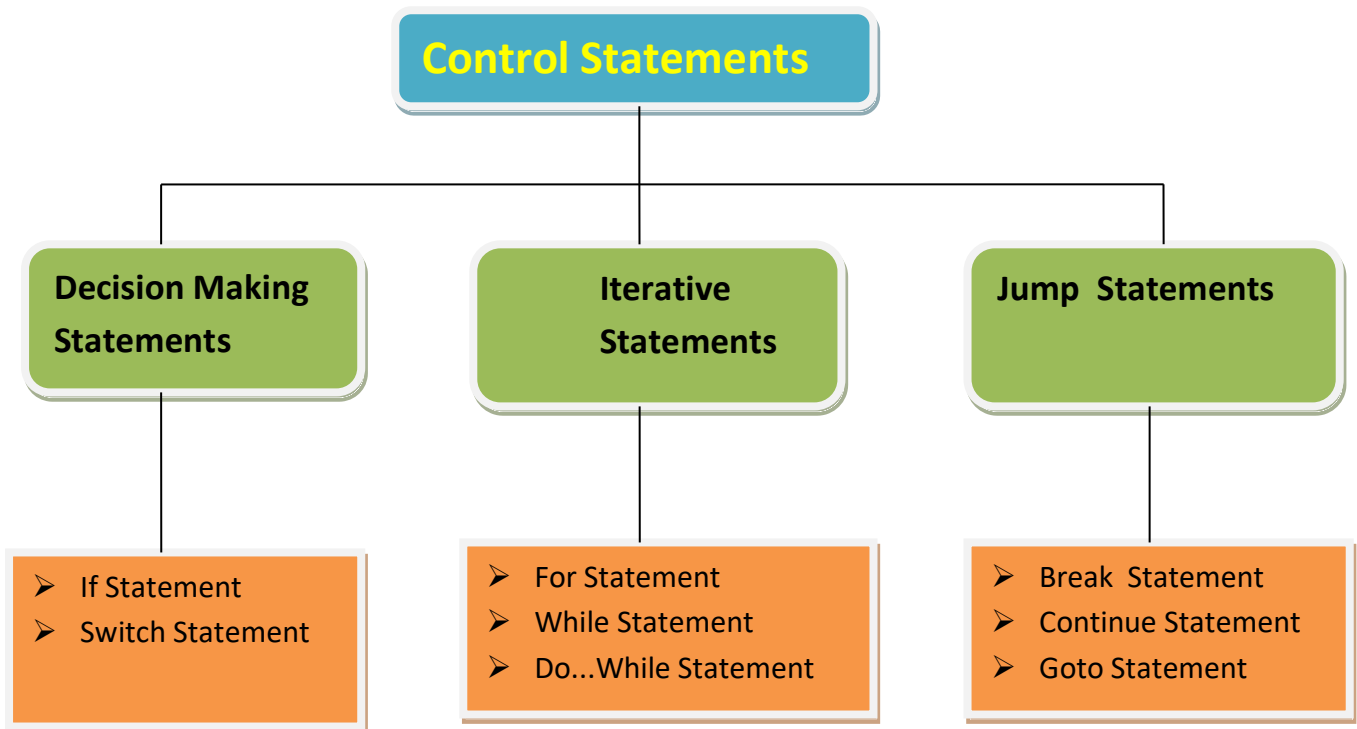
3) User defined data type

इसमें मुख्य रूप से दो डाटा टाइप आते हैं-

- typedef
- enumerated data type

control statements(structures)(कण्ट्रोल स्टेटमेंट्स क्या है):-

प्रोग्रामिंग भाषा में, प्रोग्राम के execution के flow को नियंत्रित करने के लिए जो स्टेटमेंट्स या structures प्रयोग किये जाते हैं उन्हें control statements(structures) कहते हैं.



Decision Making Statements

जब हम program की किसी statement को condition के आधार पर execute करना चाहते हैं, decision making statement use की जाती है | इसे conditional statement भी कहते हैं | decision making statement निम्नलिखित हैं |

- if statement
- if else statement
- else if statement
- nested if else
- switch

➤ If Statement

C language में if statement के द्वारा conditional statement की operation को perform कर सकते हैं | if statement में define किये गये condition true होते हैं | तो program execute होता है |

```
1 . #include <stdio.h>
2 . #include <conio.h>
3 . void main()
4 . {
5 . int num1 = 40;
6 . int num2 = 50;
7 . if(num1<num2)
8 . {
9 . printf("variable is num2 : ",num2);
10 . }
11 . return 0;
12 . }
```

OUT PUT

```
1 . variable is num2 :
```

> If-Else Statement

C language में If-Else Statement दोनों ही if statement के part हैं | लेकिन यहाँ पर condition के साथ साथ दो statement होते हैं | यानी अगर condition true होगा तो if statement का code execute होगा और condition false होने पर else statement का code execute हो जाता है |

```
#include<conio.h>
void main()
{
    int number;
    printf("enter a number:");
    scanf("%d",&number);
    if(number%2==0)
    {
        printf("%d is even number",number);
    }
    else
    {
        printf("%d is odd number",number);
    }
    getch();
}
```

➤ if else -if ladder statement

अगर प्रोग्राम में multiple condition के true या false होने पर code execute करना चाहते हैं तो ,इस type की statement use की जाती है।

Example of If else-if ladder Statement

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int number;
1   printf("enter a number:");
2   scanf("%d",&number);
3   if(number==25)
4   {
5       printf("number is equals to 25");
6   }
7   else if(number==50)
8   {
9       printf("number is equal to 50");
10  }
11  else if(number==75)
12  {
13      printf("number is equal to 75");
14  }
15  else
16  {
17      printf("number is not equal to 25, 50 or 75");
18  }
19  getch();
20  }
```

➤ Nested if else

ये statement उस टाइम use की जाती है जब हमे condition के अन्दर कोई और condition दी जाती है,जब if else के block के अन्दर फिर से if else लिखा जाता है तो उसे nested if else कहते हैं ।

Program -Greatest ऑफ़ three numbers

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

```

void main()
{
    int a,b,c, big;
    printf("Enter Three Numbers:- ");
    scanf("%d%d%d",&a ,&b, &c);
    if(a>b)
    {
        if(a>c)
        {
            big=a;
        }
        else
        {
            big=c;
        }
    }
    else
    {
        if(b>c)
        {
            big=b;
        }
        else
        {
            big=c;
        }
    }
    printf("Biggest Number Is %d", big);
    getch();
}

```

➤ Switch Statement

- switch statement भी if statement की तरह ही होता है | पर switch statement में condition के जगह case चेक किया जाता है |
- आप switch statement का case अपने अनुसार लिख सकते हैं | और अपने अनुसार code भी execute कर सकते हैं |
- switch statement का case character भी हो सकता है | और numbers भी हो सकता है | और symbols भी हो सकता है | character,numbers,symbols इन तीनों category का use case के लिए किया जाता है |

- Case एक variable से match किया जाता है। जो case variable से match हो जाता है वही case execute हो जाता है | अगर case match नहीं होता है | तो default case execute हो जाता है |

```

1. #include <stdio.h>
2. #include <conio.h>
3. int main()
4. {
5. int number;
6. printf("Enter a number : ");
7. scanf("%d",&number);
8. switch(number)
9. {
10. case 1:
11. printf("First Case");
12. break;
13. case 2:
14. printf("Second Case");
15. break;
16. default:
17. printf("Default Case");
18. break;
19. }
20. return 0;
21. }

```

ऊपर दिये गये code के अनुसार हम ने 1 number enter किया है | और 1 number enter करने के बाद जो result आया है |

OUT PUT

1. First Case

Operators (operators क्या है):-

- Operators वे symbols होते हैं जो compiler को किसी भी mathematical और logical operation को perform करने के लिए निर्देश देते है.
- Programming में operators का इस्तेमाल Data और variable को manipulate करने के लिए किया जाता है।

- Operators एक या एक से अधिक **variable, constants** या **operands** के साथ मिलकर काम करते हैं। **Variable, constant**, operands, function और operators इन सभी एक को एक साथ मिला देने से एक **expression** बनता है।

Example- $a+b*9+6/3$ एक **expression** है ।

Types of operators (operators के प्रकार) :-

'C' programming language में operators के निम्नलिखित प्रकार होते हैं :-

1. Arithmetic Operators
2. Assignment Operators
3. Relational Operators
4. Logical Operators
5. Conditional Operators (Ternary Operators)
6. Bitwise Operators
7. Increment / Decrement Operators
8. Special Operators

1. Arithmetic Operators

arithmetic operators का प्रयोग आंकिक गणनाओं के लिए किया जाता है. 'सी' में arithmetic ऑपरेटर + का प्रयोग जोड़ (addition) के लिए, - का प्रयोग घटाने (subtraction) के लिए, * का प्रयोग गुणा (multiply) के लिए, / का प्रयोग भाग (divide) तथा % का प्रयोग भाग-अवशेष (modulo division) के लिए किया जाता है.

2. Assignment operator

जब किसी वेरिएबल को मान प्रदान किया जाता है, तो असाइनमेंट operator का प्रयोग किया जाता है. 'सी' भाषा में यह ऑपरेटर (=) है।

3. Relational Operator

Relational Operator का इस्तेमाल दो operands के value को compare करने के लिए किया जाता है इसलिए इसे Comparison operator भी कहा जाता है। =, <, >, <=, >=, != **Relational Operator** हैं।

4. logical operators

'सी' में लॉजिकल ऑपरेटर का प्रयोग variables में लॉजिकल ऑपरेशन करने के लिए किया जाता है.

operators	Example/Description
&& (logical AND)	(a>6)&&(b<6) यह true दिखाता है यदि दोनों कंडीशन सत्य (true) हो तो.
(logical OR)	(a>=12) (b>=12) यह true दिखाता है यदि एक कंडीशन सत्य हो तो.
! (logical NOT)	!((a>6)&&(b<6)) यह true return करता है जब conditions satisfy नहीं होती है तो

5. Conditional Operators (Ternary Operators)

Conditional operators का इस्तेमाल condition को check करने के लिए किया जाता है। इस operator के पास केवल दो ही option होते हैं **TRUE** और **FALSE**. अगर दिया गया condition satisfy करता है तो TRUE return होगा अन्यथा FALSE return होगा।

Syntax of conditional operators:-

Condition ? True Expression : False Expression;

यह वास्तव में if condition ही होता है लेकिन इसको थोड़ा modify करके एक short form बनाया गया है जो की केवल दो ही option provide कर सकता है। **TRUE और FALSE**

6. Bitwise Operator

Bitwise operator का इस्तेमाल **BIT level** के data को manipulate करने के लिए किया जाता है। इस operator का इस्तेमाल right से left और left से right bit shifting के लिए किया जाता है। Bitwise operator **float और double** data type पर apply नहीं होता है।

Operator	Description
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR
<<	Left Shift
>>	Right Shift

7. Increment / Decrement Operator

Increment / Decrement operator का इस्तेमाल variable के value को एक बार increase और Decrease करने के लिए किया जाता है। जैसे यदि पहले से

variable में 5 value store है तो increment operator के द्वारा इसका value 6 हो जायेगा और decrement operator के द्वारा इसका value 4 हो जायेगा।

Increment / decrement operator दो प्रकार के होते हैं।

- Pre increment / decrement
- Post increment / decrement
- **Pre increment / decrement operator**

Pre increment / decrement operator में सबसे पहले value increase और decrease होता है उसके बाद आगे का calculation perform होता है।

Syntax of pre increment / decrement operator

++ (variable name), -- (variable name)

- **Post increment / decrement operator**

Post increment / decrement operator में सबसे पहले calculation perform हो जाता है उसके बाद value change होता है। Post का मतलब होता है बाद में। इसमें कैलकुलेशन perform होने के बाद में value increase और decrease होता है।

Syntax of post increment / decrement operator

(variable name) ++, (variable name) --

8. Special Operator

Special operator का इस्तेमाल special work को perform करने के लिए किया जाता है। C programming में ढेर सारे special operator है जिनके list यहाँ दिए गये हैं।

Operator	Description	Example
sizeof()	यह operator किसी भी variable का size उसके data type के अनुसार return करता है।	जैसे a एक integer variable है तो यह 4 या 2 return करेगा। अलग अलग operating system के अनुसार
&	यह operator variable के address को return करता है।	&a; यह a के address को return करेगा।

*	Pointer variable को denote करने के लिए इस्तेमाल किया जाता है।	*a
---	---	----

What is Loop (loop क्या है) :-

जब कोई प्रोग्राम बनाते हैं, अगर उसमें कोई एक ही चीज बार-बार प्रिंट करानी है या किसी भी वर्ड या गिनती को एक के बाद एक को प्रिंट करना चाहते हैं। या किसी भी नाम को एक से अधिक बार प्रिंट करवाना चाहते हैं। तो हमें जितनी बार नाम को प्रिंट करवाना है, उतनी बार हमें printf() का प्रयोग करना पड़ेगा, इस कारण दो समस्या होती है।

1. प्रोग्राम काफी लम्बा हो जायेगा ।
2. मेमोरी भी काफी waste होगी।

उपरोक्त समस्या से बचने के लिए C लैंग्वेज में loop की सुविधा दी गयी जिससे आप आसानी से loop की मदद से एक ही Statement को बार-बार आसानी से execute कर सकते हैं।

C में अलग-अलग प्रकार के loop की व्यवस्था है, और हर तरह के loop में block होता है, जिसमें वह Statement लिखा जाता है। जिन्हें आप प्रोग्राम में एक से ज्यादा बार execute करवाना चाहते हैं।

loop तीन चीजों से मिलकर बना होता है।

1. Initial Variable :-

यह एक counter variable होता है, जहाँ से loop को स्टार्ट करते हैं। यह एक integer variable होता है, इस Variable को तब तक चेक किया जाता है। जब तक की loop के अंदर दी गयी Condition false ना हो जाये, इस Variable को loop की Condition में include किया जाता है।

2. Condition :-

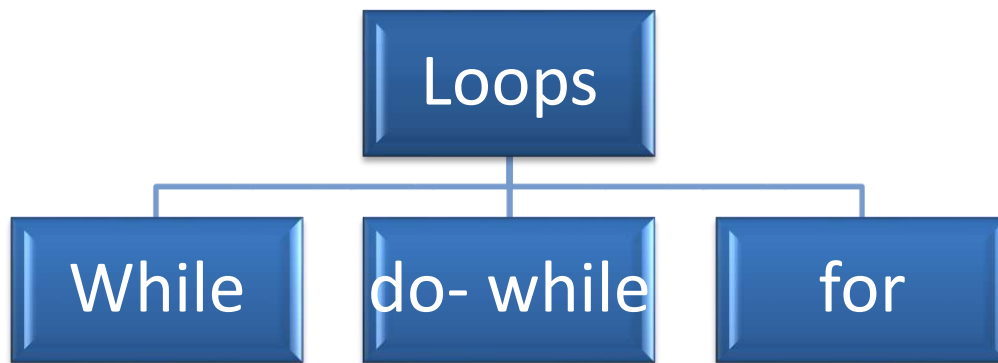
यह वह Condition होती है, जो loop को कंट्रोल करती है। जब तक यह Condition true रहती है, loop execute होता रहता है। जैसे ही Condition false होती है control loop के बाहर हो जाता है।

3. Increment/Decrement :-

इस पार्ट में counter variable में प्रत्येक बार कितना increment/decrement करना है decide किया जाता है ।

Types of Loop (loop के प्रकार) :-

Loop निम्नलिखित प्रकार के होते हैं।



1. While Loop :-

- **While Loop एक simple loop** होता है।
- यह जबतक **Condition true** रहती है, तबतक **execute** होता है। **Condition के false** होने पर यह **loop terminate** हो जाता है। यदि **Condition पहली बार** में ही **false** हो तो **कम्पाइलर loop** में एंटर ही नहीं होता है। इसीलिए **while loop** को **Entry controlled loop** कहते हैं ।

Syntax:-

```
Initial variable declare;
while (condition)
{

//statement;

//increment /decrement;

}
```

Program-

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

```

{
int num=1;
  while(num<=5)
  {
    printf("Loop number %d\n",num);
    num++;
  }
return 0;
}

```

Output-

```

Loop number 1
Loop number 2
Loop number 3
Loop number 4
Loop number 5

```

2. **Do-While Loop :-**

- **Do-While Loop** भी while loop की तरह होता है,लेकिन इसमें **पहले Condition** चेक होने की बजाय,**पहले Statement execute** होते हैं। इसीलिए do while loop को exit controlled loop कहते हैं | अर्थात condition लास्ट में चेक होती है |

Syntax :-

```

do
{
//statement;
//increment/decrement ;

}while(condition);

```

जैसे ही कम्पाइलर **do** सेक्शन में जाता है,तो **do block** के सभी **Statement execute** कर दिए जाते हैं। और इनिशियल Variable को **increment/decrement** कर दिया जाता है,इसके बाद **कम्पाइलर while Condition** को चेक करता है। यदि **Condition true** होती है,तो **do block** को वापस से **execute** कर दिया जाता है। लेकिन अगर **Condition false** है,तो **loop terminate** करके **कम्पाइलर आगे बढ़** जाता है। **while** के अंदर की **Condition** चाहे **true** हो या **false** **do block** के **Statement** एक बार जरूर **execute** होंगे।

Program-

```
#include <stdio.h>
#include <conio.h>
int main()
{
int num=0;
do
{
printf("Loop Number %d\n",num);
num++;
}while(num<5);
return 0;
}
```

Output-

```
Loop Number 1
Loop Number 2
Loop Number 3
Loop Number 4
Loop Number 5
```

3. For Loop :-

- यह loop सबसे ज्यादा C प्रोग्रामिंग में यूज़ होता है, यह loop बहुत ही easy होता है। और यह एक सिंगल Statement में डिफाइन हो जाता है।
- for loop को Entry controlled loop कहते हैं , क्योंकि condition starting में चेक होती है ।

Syntax :-

```
for(initial variable; condition; increment/decrement)
{
//statement;
}
```

Program-

```
#include <stdio.h>
#include <conio.h>
int main()
{
int num;
for(num=1; num<=5; num++)
{
printf("For Loop %d\n",num);
}
```

```
    }  
    return 0;  
}
```

Output-

```
For Loop 1  
For Loop 2  
For Loop 3  
For Loop 4  
For Loop 5
```

Nested Loop :

- वास्तव में इसे loop का प्रकार नहीं माना जाता क्योंकि इस लूप में for, while or do..while लूप ही execute किये जाते हैं।
- इसमें एक लूप के अंदर दूसरे लूप को execute किया जा सकता है।

Example -

```
for (initialization; condition; increment/decrement)  
{  
    statement(s);  
    for (initialization; condition; increment/decrement)  
    {  
        statement(s);  
        ... ..  
    }  
}
```

Loop Control Statement-

- इन statement से आप loop को control कर सकते हो। अगर आप किसी loop को loop पूरा होने से पहले रोकना चाहते हैं या बीच में किसी और statement को execute करना चाहते हैं तो इसके लिए आप Loop Control Statement का use कर सकते हैं

Loop Control Statement Types-

1. break statement
2. continue statement
3. goto statement

1. break Statement

- अगर loop को पूरा होने से पहले रोकना चाहते हो तो आप **break statement** का use कर सकते हो। जब भी आप loop अंदर break statement को लिखते हो तो जब break statement execute होता है तो उसी time loop रुक जाता है और उसके बाद जो भी statement होगा वो execute होगा इसका example नीचे दिया गया है।
- Switch case statement से बाहर आने के लिए भी break statement का use किया जाता है ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num;
    for(num=1;num<=10;num++)
    {
        if(num==5)
        {
            printf("Loop Stopped");
            break;
        }
        printf("%d\n",num);
    }
}
```

Output-

```
1
2
3
4
Loop Stopped
```

2. continue Statement

Continue statement का use से आप loop की iteration को skip कर सकते है अर्थात यदि किसी भी statement को 10 times print करवा रहे है और आप चाहते की 5th number statement न print हो तो वहां पर आप Continue Statement use कर सकते है ।

Example-

```
void main()
{
    int num;
    for(num=1;num<=10;num++)
    {
        if(num==5)
        {
            printf("Fifth Number Not Print");
            continue;
        }
        printf("%d\n",num);
    }
}
```

Output -

```
1
2
3
4
Fifth Number Not Print
6
7
8
9
10
```

3. goto Statement

goto statement का use करके एक **block of code** से दूसरे **block of code** में जा सकते हो जैसे ही loop में भी होता है अगर किसी loop के iteration में किसी दूसरे block of code को execute करवाना चाहते है तो goto का use कर सकते है नीचे गए example को देखिये

```
#include <stdio.h>
int main()
{
    int i,sum=0;
    for(i = 0; i<=10; i++)
    {
```

```

sum = sum+i;
if(i==5)
{
    goto addition;
}
}
addition:
printf("sum=%d", sum);
return 0;

}

```

Output-
sum=15

ARRAY

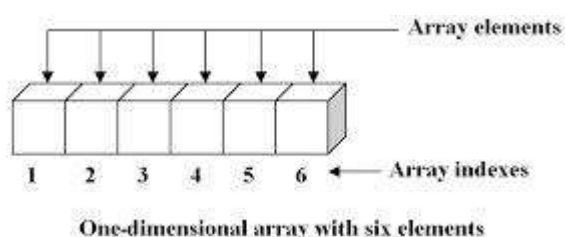
- जब हमें किसी Program में एक ही प्रकार के बहुत सारे Data को Store करके Access व Manipulate करना होता है, तब हम इन समान प्रकार के Data को Store करने के लिए बहुत सारे Variables Declare करने के स्थान पर Array Declare करते हैं।
- **Array एक similar data type की values का collection होता है। Similar data type का मतलब एक ही तरह के data store किये जाते है | जैसे int, float, char आदि ।**
- Array एक प्रकार का Reference Type होता है, जो समान प्रकार के Data Items को Store करने के लिए Memory के Heap Area का प्रयोग करता है और Array के हर Data Item को Store करने के लिए Continues Memory Locations का प्रयोग करता है। इसलिए Array के सभी Data Items Memory में एक निश्चित क्रम में Store होते हैं और हर Element को Uniquely Identify करने के लिए उसके साथ एक Zero Based Unique Index Number Specified रहता है। साथ ही Array एक Fixed Data Structure होता है। इसलिए जब एक बार Array की Size को Define कर दिया जाता है, तो उसके बाद Array की Size को Modify नहीं किया जा सकता।

Array के दो प्रकार के है |

1. Single/One Dimensional Array
2. Two Dimensional Array
3. Multi Dimensional Array

1:- Single/One dimensional(1-D) arrays:-

वह arrays जिसमें सिर्फ एक subscript होती है उसे one dimensional arrays कहते हैं। इसका प्रयोग linear रूप में डेटा को स्टोर करने के लिए किया जाता है।



Create a Arrays

C language में Array एक structured data type होता है। जो एक ही types के data को store करता है | अब वह data integer भी हो सकता है | या नाम भी हो सकता है |

data-type array-name[size];

C language में जब भी array create करते हैं | तो हमें array का data type और array का नाम और उस array में कितनी values store करने हैं | यह define करते हैं। size में हम array का length define करते हैं | की हमारा array कितना value store करेगा |

आप 10 number store करना चाहते हैं | तो आप array create कर सकते हैं |

```
int num[10];
```

उपर दिए गए उदाहरण में num एक array है | और इस array में कोई भी 10 integer values store कर सकते हैं।

Initialize of Arrays

जब भी हम array create करते हैं | तो साथ में उस array का index भी create हो जाता है | और उसी index को point करके हम values भी store कर सकते हैं |

किसी भी array का first index 0 होता है | अब वह array int data store करता हो या char data store करता है | दोनों में उसका index 0 ही होगा |

जब भी हम array create करते है | तो जितनी उसकी size होती है | उतनी ही memory array में allocate हो जाती है। और उतने ही index number allocate होती है।

Array के नाम और index number से हम value store कर सकते है | और बाद में उस value को access भी कर सकते है।

array_name [index no.] = value;

Eg-num[5] = 600;

हम चाहे तो एक ही साथ value assign कर सकते है |

Example-

```
int num[10] = {100,200,300,400,500,600,700,800,900,1000};
```

अगर आप user के द्वारा value assign करना चाहते है | तो उसका भी example निचे दिया जा रहा है |

```
1. #include<stdio.h>
2. #include<conio.h>
3. void main()
4. {
5. int num[5];
6. int i;
7. for(i=0; i<=4; i++)
8. {
9. printf("Enter your no : \n");
10. scanf("%d",&num[i]);
11. }
12. for(i=0; i<=4; i++)
13. {
14. printf("[%d] - %d\n",i,num[i]);
15. }
16. return 0;
17. }
```

OUT PUT

```
1. Enter your no :
2. 1
3. Enter your no :
4. 2
5. Enter your no :
```

6. 3
7. Enter your no :
8. 4
9. Enter your no :
- 10.5
- 11.[0] - 1
- 12.[1] - 2
- 13.[2] - 3
- 14.[3] - 4
- 15.[4] - 5

Access of Array Elements

array के elements को कैसे access करते है | इसका example निचे दिया जा रहा है |

1. #include<stdio.h>
2. #include<conio.h>
3. int main()
4. {
5. int num[5]={ 10,20,30,40,50 };
6. printf("Your no is %d",num[3]);
7. return 0;
8. }

OUT PUT

1. Your no is 40

2:- Two dimensional(2-D) arrays:-

वह arrays जिसमें दो subscript होती है उसे two dimensional array कहते है। two dimensional arrays को matrix तथा table भी कहते है।

Syntax for Two Dimensional Array Declaration

```
data_type array_name[ row_size ][ column_size ];
```

Syntax for Two Dimensional Array Initialization

for eg.

```
int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12}; OR
```

```
int arr[3][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

Example for Two Dimensional Array

```
#include <stdio.h>

int main() {

int arr[3][4] = { {1,2,3,4}, {5,6,7,8}, {9,10,11,12}
};
int i, j;

    for ( i = 0; i < 3; i++ ){

        for ( j = 0; j < 4; j++ ){

            printf("[%d][%d]=%d\n",i,j,arr[i][j]);

        }
    }
return 0;
}
```

Output :

```
[0][0]=1
[0][1]=2
[0][2]=3
[0][3]=4
[1][0]=5
[1][1]=6
[1][2]=7
[1][3]=8
[2][0]=9
[2][1]=10
[2][2]=11
[2][3]=12
```

3. Multi Dimensional Array

ऐसे array जिसमे एक से जायदा subscripts होती हैं उन्हें multi-dimensional array कहते हैं।

Some Important Program of C language Program

1. C "Hello, World!" Program

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

    printf("Hello, World!");
    return 0;
}
```

Output

Hello, World!

2. Program to Print an Integer

```
#include <stdio.h>
int main() {
    int number;
    printf("Enter an integer: ");
    scanf("%d", &number)
    printf("You entered: %d", number);

    return 0;
}
```

Output

Enter an integer: 25
You entered: 25

3. Program to Add Two Integers

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

    int number1, number2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    // calculating sum
    sum = number1 + number2;

    printf("%d + %d = %d", number1, number2,
sum);
    return 0;
}
```

Output

```
Enter two integers: 12
11
12 + 11 = 23
```

4. Program to Check Even or Odd

```
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);

    // True if num is perfectly divisible by 2
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```

Output

Enter an integer: -7
-7 is odd.

5. Program to Check Leap Year

```
#include <stdio.h>
int main() {
    int year;
    printf("Enter a year: ");
    scanf("%d", &year);
    if (year % 4 == 0) {
        if (year % 100 == 0) {
            if (year % 400 == 0)
                printf("%d is a leap year.", year);
            else
                printf("%d is not a leap year.", year);
        } else
            printf("%d is a leap year.", year);
    } else
        printf("%d is not a leap year.", year);

    return 0;
}
```

Output 1

Enter a year: 1900
1900 is not a leap year.

6. Check Positive or Negative Using Nested if...else

```
#include <stdio.h>
int main() {
    double num;
    printf("Enter a number: ");
    scanf("%lf", &num);

    if (num < 0.0)
        printf("You entered a negative number.");
    else if (num > 0.0)
        printf("You entered a positive number.");
    else
        printf("You entered 0.");

    return 0;
}
```

Output 1

```
Enter a number: 12.3
You entered a positive number.
```

7. Multiplication Table Up to 10

```
#include <stdio.h>
int main() {
    int n, i;
    printf("Enter an integer: ");
    scanf("%d", &n);
    for (i = 1; i <= 10; ++i) {
        printf("%d * %d = %d \n", n, i, n * i);
    }
    return 0;
}
```

Output

```
Enter an integer: 9
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90
```

8. Sum of Natural Numbers Using for Loop

```
#include <stdio.h>
int main() {
    int n, i, sum = 0;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

    for (i = 1; i <= n; ++i) {
        sum += i;
    }

    printf("Sum = %d", sum);
    return 0;
}
```

Output

```
Enter a positive integer: 100
Sum = 5050
```

9. Reverse an Integer

```
#include <stdio.h>
int main() {
    int n, rev = 0, remainder;
    printf("Enter an integer: ");
    scanf("%d", &n);
    while (n != 0) {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n /= 10;
    }
    printf("Reversed number = %d", rev);
    return 0;
}
```

Output

```
Enter an integer: 2345
Reversed number = 5432
```

10. Check Armstrong Number of n digits

A positive integer is called an Armstrong number (of order n) if

$$abcd\dots = a^n + b^n + c^n + d^n + \dots$$

In the case of an Armstrong number of 3 digits, the sum of cubes of each digit is equal to the number itself. For example, 153 is an Armstrong number because

$$153 = 1*1*1 + 5*5*5 + 3*3*3$$

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

int main() {
    int num, originalNum, remainder, n = 0,
        result = 0, power;
    printf("Enter an integer: ");
    scanf("%d", &num);
    originalNum = num;
    while (originalNum != 0) {
        originalNum /= 10;
        ++n;
    }
    originalNum = num;
    while (originalNum != 0) {
        remainder = originalNum % 10;
        power = round(pow(remainder, n));
        result += power;
        originalNum /= 10;
    }

    if (result == num)
        printf("%d is an Armstrong number.", num);
    else
        printf("%d is not an Armstrong number.",
num);
    return 0;
}
```

Output

```
Enter an integer: 1634
1634 is an Armstrong number.
```

11. Program to Check Palindrome

An integer is a palindrome if the reverse of that number is equal to the original number.

```
#include <stdio.h>
int main() {
    int n, reversedN = 0, remainder, originalN;
    printf("Enter an integer: ");
    scanf("%d", &n);
    originalN = n;

    // reversed integer is stored in reversedN
    while (n != 0) {
        remainder = n % 10;
        reversedN = reversedN * 10 + remainder;
        n /= 10;
    }

    // palindrome if originalN and reversedN are equal
    if (originalN == reversedN)
        printf("%d is a palindrome.", originalN);
    else
        printf("%d is not a palindrome.", originalN);

    return 0;
}
```

Output

```
Enter an integer: 1001
1001 is a palindrome.
```

12. Program to Check Prime Number

A prime number is a positive integer that is divisible only by 1 and itself. For example: 2, 3, 5, 7, 11, 13, 17

```
#include <stdio.h>
int main() {
    int n, i, flag = 0;
    printf("Enter a positive integer: ");
    scanf("%d", &n);

    for (i = 2; i <= n / 2; ++i) {

        // condition for non-prime
        if (n % i == 0) {
            flag = 1;
            break;
        }
    }

    if (n == 1) {
        printf("1 is neither prime nor composite.");
    }
    else {
        if (flag == 0)
            printf("%d is a prime number.", n);
        else
            printf("%d is not a prime number.", n);
    }

    return 0;
}
```

Output

```
Enter a positive integer: 29
29 is a prime number.
```

13. Store Numbers and Calculate Average Using Arrays

```
#include <stdio.h>
int main() {
    int n, i;
    float num[100], sum = 0.0, avg;

    printf("Enter the numbers of elements: ");
    scanf("%d", &n);

    while (n > 100 || n < 1) {
        printf("Error! number should in range of (1
to 100).\n");
        printf("Enter the number again: ");
        scanf("%d", &n);
    }

    for (i = 0; i < n; ++i) {
        printf("%d. Enter number: ", i + 1);
        scanf("%f", &num[i]);
        sum += num[i];
    }

    avg = sum / n;
    printf("Average = %.2f", avg);
    return 0;
}
```

Output

```
Enter the numbers of elements: 6
1. Enter number: 45.3
2. Enter number: 67.5
3. Enter number: -45.6
4. Enter number: 20.34
5. Enter number: 33
6. Enter number: 45.6
Average = 27.69
```

14. Find the Largest Element in an array

```
#include <stdio.h>
int main() {
    int i, n;
    float arr[100];
    printf("Enter the number of elements (1 to 100): ");
    scanf("%d", &n);

    for (i =0; i< n; ++i) {
        printf("Enter number%d: ", i + 1);
        scanf("%f", &arr[i]);
    }

    // storing the largest number to arr[0]
    for (i = 1; i < n; ++i) {
        if (arr[0] < arr[i])
            arr[0] = arr[i];
    }

    printf("Largest element = %.2f", arr[0]);

    return 0;
}
```

Output

```
Enter the number of elements (1 to 100): 5
Enter number1: 34.5
Enter number2: 2.4
Enter number3: -35.5
Enter number4: 38.7
Enter number5: 24.5
Largest element = 38.70
```

15. Program to print half pyramid using *

```
*  
* *  
* * *  
* * * *  
* * * * *
```

Source Code-

```
#include<stdio.h>  
int main() {  
    int i, j, rows;  
    printf("Enter number of rows: ");  
    scanf("%d", &rows);  
    for (i=1; i<=rows; ++i) {  
        for (j=1; j<=i; ++j)  
            { printf("* "); }  
        printf("\n");  
    }  
    return 0;  
}
```

16. Program to print half pyramid a using numbers

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Source Code

```
#include<stdio.h>
int main() {
    int i,j,rows;
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    for (i=1; i<=rows; ++i) {
        for (j=1; j<=i; ++j)
            { printf("%d ",j); }
        printf("\n");
    }
    return 0;
}
```

17. Program to print full pyramid using *

```
      *
     * * *
    * * * * *
   * * * * * * *
  * * * * * * * *
```

Source Code

```
#include<stdio.h>
int main() {
    int i, space, rows, k=0;
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    for (i=1; i<=rows; ++i,k=0) {
        for (space=1; space<=rows-i; ++space)
            printf(" ");
        while (k!=2*i-1) {
            printf("* ");
            ++k;
        }
        printf("\n");
    }
    return 0;
}
```

18. Print Floyd's Triangle.

```
1
2 3
4 5 6
7 8 9 10
```

Source Code-

```
#include<stdio.h>
int main() {
    int rows, i, j, number= 1;
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    for (i=1; i<=rows; i++) {
        for (j=1; j<=i; ++j)
            { printf("%d ", number);
              ++number;
            }
        printf("\n");
    }
    return 0;
}
```

Following are some of the questions according to your syllabus, only for practice and self assessment purpose

1. What is C language?
2. Who developed C language?
3. Describe about history of C programming language.
4. What are ASCII codes?
5. What is the extension of a source file in C?
6. Where is C programming language used or uses of C language?
7. What is the difference between top down approach and bottom up approach in programming languages?
8. What is meant by programming language and give some examples?
9. What is the difference between assembler, compiler and interpreter?
10. What is printf()?
11. What is scanf()?
12. What is clrscr()?
13. What is getch()?
14. Every statement in C language ends with a.....symbol.
15. Execution of a C program starts from which function?
16. What is the use of main() function in C?
17. What are all the sections that a C program may have and must have?
18. What is header file in C language?
19. Is C language case sensitive?
20. What is data type in C?
21. What are Constants, Variables and Keywords? Give examples.
22. What are static variables?
23. What is operator in C?
24. What are all loop control statements in C?
25. Write the difference between GoTo, Exit, Continue and Break statements.
26. What is the difference between while and do-while loops in C?
27. What are nested loops?
28. What is the difference between array and string in C?
29. What are nested if-else statements?
30. What is the range of integer variable in 32 bit operating system.
31. What is identifier in C.
32. What is token in C.
33. Explain different types of operators in C.
34. Explain different types of decision making statement in C language.
35. What is relational and conditional operator in C.?