C PROGRAMMING :

DEVELOPMENT :

Types of applications :

Mobile applications : .apk ,need to install in Mobile phones

Websites : Run by URL(Uniform Resource Locator) : It runs on web-browser

Method to create -> Designing Language : HTML + CSS , BOOTSTRAP

Web Applications : Dynamic Websites : Runs by URL , It runs on web browser

Method to create :

* Designing : HTML +CSS / BOOTSTRAP
* Programming : C , C++, JAVA, C#, PYTHON , PHP, Javascript……………….
* Database : MYSQL, MSSQL, MSACCESS, ORACLE, MONGODB, ……………….

Console Applications : Run by Commands on blue/black screens

Desktop Applications : Run by .exe files , need to install

Standalone applications : Key based softwares , like Antiviruses

Languages : Languages is a medium to communicate with other person.

Programming Language : It is a medium to communicate with computer system.

By coding , you are able to perform any task by computer system.

**C programming : Standalone Applications : to apply logics**

**C is a high level programming language. It was developed by Dennis Ritchie in 1972 in A&T’s Bell Laboratory. C is a HLL(High level language) that means maximum words used in C programming is taken with the real life words.**

High level language that means a developer friendly language.

Computer->Language-> Machine language(Binary Language /Low level language/0&1)

C is also known as mother language just because syntaxes of maximum other programming language is taken from the C.

**Basic Structure of C programs :**

1. **Documentation Section : (Optional)**

Comments : Comments are those line of program , that is ignored by Compiler. It is just for the understanding of developer.

Single Line Comment : //Comments……………..

Multi-line comment : /\* …………………………………….

**……………………………………………………………………\*/**

1. **Linking Section : Each programming language provides some pre-defined codes, that is saved in the library of programming.**

**When we need to use those pre-defined codes firstly we need to add library in the programs. : (Mandatory)**

**C – Header Files**

**Java – Package**

**Python – Modules**

**C# - Namespace**

**Header File :** Header file is the collection of some pre-defined functions. Functions are kept in different header files based on the category.

**stdio.**h : Standard Input Output : printf(), scanf(), ……………………..

**conio.**h : Console Input Output : clrscr(), getch(), getche(), getchar(), putchar()……………………..

**math.**h : pow(), sqrt(), log(), ceil(), floor()………………….

**String.**h : strcpy(), strlwr(), strupr(), strrev(), strcat(), …………………………

**#include”header\_file\_name\_with\_extension”**

**#include< header\_file\_name\_with\_extension>**

# - Pre-processor directive

Lines written with # is executed first in the program.

include : include is a folder that contains all header files

1. **Global Declaration Section : Global declaration that means declaring some variables globally that is accessible in all function of programs : (Optional)**

**int a=5; //globally declared**

**void main() //pre-defined function + user-defined function**

**{**

**A=a+10;**

**}**

**Void techpile() //user-defined function**

**{**

**}**

1. **Main Function : Main function is a mandatory function of each C programs.**  Main function is declared with named main(). : (Mandatory)

**void main()**

**{**

**}**

**//void – Data type**

**Main() - Function**

1. **Other sub programs : You can define other user-defined function in C programs . It is optional. : (Optional)**

**//This is my First Program //comment lines**

#include<stdio.h> // Header-files

**void main() //it is the main part of program , execution starts from here**

**{**



**}**

**Keywords : Keywords** are some reserve words that has special meaning and functionality . In C program total 32 keywords are defined . In programming keywords are used at special places whenever it needs.

**Ex : if, else, do, int , while, float , double, long, struct, break, continue……..**

**Identifier :** Identifiers are used for naming like : user defined function, variable, constant.

Rules to declare identifier :

Identifier can not start with Number. It can not be NumericAlpha.

Ex : 2var : invalid

Identifier can not have space in name .

It can not contain some special symbols like & , \* etc

It can not be same as keywords

It can be Alpha-numeric like : var1, number1

Identifier : variables, constant , UDfunction

Data\_type variable\_name;

Data\_type constant­name;

Data\_type function\_name();

**Data Types : Data**  types are some reserve words that defines the type of value and total memory space. Data type defines which type of value , it can store and how much memory space it will occupy.

**There are mainly Four type of data type in C :**

1. **Primitive data type / Basic / Fundamental Data Type**

|  |  |  |  |
| --- | --- | --- | --- |
| Data type name | Format Specifier | Memory Space | Ex Values |
| Short : whole num | %d | 2 byte | -32768 to + 32767 |
| int : whole num | %d | 2 byte/ 4 byte | 3 |
| long int : whole num | %ld | 4 byte |  |
| long long int : whole num | %Ld | 8 byte |  |
| Float : decimal | %f | 4 byte | 3.5 |
| Double : decimal | %lf | 8 byte |  |
| long double : decimal | %Lf | 10 byte |  |
| char | %c | 1 byte |  |

1. **Non-primitive data type / Derived data type : Derived data type**  are created with the help of basic data types.

**Ex : String , Array , Pointer**

1. **User-defined data type : User defined data type are declared by user. Which type of value it can store is decides by user.**

**Ex : Structure , Union**

1. **Empty data type: void is the empty data type. It is used to declare functions. It is not used to declare variables.**

**----------------------------------------------------------------------------------------**

**Input & Output function in C : Each programming language** some pre-defined input output function.

I/P Function : Input function is used to take input from the user at run time. Where you can store those values in a variable and can use anywhere in the program.

* Formatted Input Function :

scanf() :

scanf(“format\_specifier”,variable\_list\_with\_&); //int, float, long int , double, char , one word string “riya”

long int v;

scanf(“%ld”,&v);

* Un-formatted Input function :

getch() : used to input a single character. getch() allows user to input only a single character. Without showing the inputed character , program will execute to the next line .

This is unformatted function , specially made for the character input.

Syntax :

My\_variable=getch();

---------------------------------------------------------------------------------------------

getche() : used to input a single character from user

**getchar() : used to input a single character from user**

**gets() : used to input a multi-word string value from user**

**-----------------------------------------------------------------**

**Output Function : O/p functions are used to print message or value on the screen for user.**

* **Formatted Output function**

**printf() :**

**void main()**

**{**

**printf(“my variable is a”);**

**}**

* **Un-formatted output function**

**putchar()**

**puts()**

**-----------------------------------------------------------------------------------------**

**Basic Introduction of C**

**Basic structure of C program**

**Data type, keywords, variables, etc**

**------------------------------------------------------------------------------**

**Variable : Variables are some temporary storage area, which can hold a value and it’s value may be change any where during the execution of program.**

**Note :**

1. **Declaration of variable : Each variables in C, should be declare once at the top of program. Without declaration you can not use any variable in your program.**

**Syntax :** data\_type variable\_name;

1. **Definition of variable /** assign value to the variable
2. Direct Initialization / compile time initialization

var=50;

1. User Input / Run time initialization

scanf(“format\_specifiers”,variable\_list\_with\_&);

**scanf(“%f”,&var); - & - denotes the memory address of variable**

**int a; float f;**

**scanf(“%d”,&a);**

**scanf(“%f”,&f);**

**scanf(“%d %f”,&a,&f);**

1. **By expression / calculation**

**A=b+10;**

**How many Variable =?**

**Variable declare**

**Variable initialize – value**

**Int a;**

**A=30; //**

**Scanf(“%d”,&a); //**

**User input :**

**Numeric – scanf()**

**Char – getch()**

**Exercises & Task:**

**Coding - error**

Numeric value : whole number / real numbers – scanf

Character : getch(), getche(), getchar()

String : gets()

O/P FUNCTION :

Numeric value : whole number / real numbers – PRINTF()

SINGLE CHARCATER : PUTCHAR()

STRING : PUTS()

Character input special functions :

**getch() : getch() is used to input a single character from user. Getch() function reads the inputed value from the screen and saves the value in a character type variable.**

Working : getch() permits user to input only a single key on the output screen, next line of program executes without waiting to press enter key , as soon as user inputs a single character.



Syntax :

char ch;

ch=getch();

Character inputed by user, by using getch() function does not appear on output screen .

------------------------------------------------------------------------------------------



getche(): getche() also permits user to input only a single character and character inputed by user appears on the output screen and next line of program executes without waiting for enter key.

ex : //wap to input a character and print the inputed character

#include<stdio.h>

void main()

{

char ch;

printf("Please enter a single character : ");

ch=getche(); //?

printf("\nInputed character is : %c",ch);

}

----------------------------------------------------------------------------

**getchar() : getchar()**  function works same as the scanf() , but getchar() is a unformatted input function so you do not need to add any format specifier here.

Syntax :

char val;

val=getchar();

C source code -> Compile -> Object file -> converted into .exe file -> output

-----------------------------------------------------------------------------------------

Operator : Operators are some special symbols , pre-defined in library , that is used for special functionality.

Each operator has it’s own use, when ever we need this we can use the operators in any statement of program.

Operators are always used with operands .

Suppose a statement :

A=A+b : here a,b is the operand and + is the operator

= is also a operator.

Unary operator : used with one operand , like Increment – decrement operator

Binary operator : used with minimum 2 operands like : +,-, \*, > , < etc

Ternary Operator : used with minimum 3 operands like : conditional operator

Based on the working operators are divided into many category :

1. Arithmetic operator
2. Relational Operator
3. Logical operator
4. Assignment operator
5. Increment & Decrement operator
6. Conditional Operator
7. Bitwise operators
8. Arithmetic Operator : This operator is used for mathematical operations. It is used with numeric values. It is a binary operator that means it needs min. 2 operands to be used.

+ : addition

- : subtract

\* : multiply

/ : division

% : modular division

Precedence of operator : (Which one will execute first)

\*

/

%

+

-

=

Example :

#include<stdio.h>

void main()

{

int n1,n2,n3;

printf("enter three numbers : ");

scanf("%d%d%d",&n1,&n2,&n3);

printf("sum of numbers is : %d",(n1+n2+n3));

printf("\nsubtract of first & second number is : %d",(n1-n2));

printf("multipky result of last two values is : %d",n3\*(n1-n2));

}

**/** : it returns always a integer number if both operands are integer.

It returns a float value if any one operand is float.

/ : 20/3 = 6

20.0 /3 = 6.6666

**% : It**  is called as modulo operator. It returns remaining value after the division of two integer type values. It can not be applied on floating type value.

Int n1=40, n2=9;

N1/n2=4

N1%n2=division remainder

------------------------------------------------------------------------------------------------

Precedence of arithmetic operatr:

\*,/,%: same precedence :high:Associativity :left to right

+,-: same precedence :Low:Associativity :left to right

Int a=a-10\*b

**Realational operators**:relational operators are used to specify the relation between two operands.it is a binary operator.

Return type /result of relational statement is always true/false

Relational statement is a Boolean statement.

==:equal to

>:greater than

<:less than

>=:greater than equal to

<=less than equal o

!=:not equal to

Example.

A=a>10;

**Conditional Operators:** Conditional Operators are used to execute ane statement out of two statement based on a condition.

It is optional of if -else statement. When there is only line statement in if and else then you can use conditional statement in more easy way.

It is ternary operators, that means it executes with 3 operands.

? and :is condtional or ternary operators

Syntax-

Condition? true\_statement : false \_statement

**Condition is a statement which result is always true or false**

**Logical Operator:-Logical operator is also used in two operands .return value of logical operator is true or false.**

**&& : Logical AND**

**|| : Logical OR**

**! : Logical not / Logical Not**

!= : ignores a single value , a!=20,ch!=’m’

== : stands for a single value a==20,char ch;ch==’z’

|  |  |  |  |
| --- | --- | --- | --- |
| Statement | Output | Statement | Output |
| True && True | True/1 | True||True | True/1 |
| False && False | False/0 | True|| False | True/1 |
| True && False | False/0 | False||True | True/1 |
| False && False | False/0 | False||False | False/0 |

1&&1

20 && 50

20>40 && 50>60

Assignment Operator- = is the assignment operator .it is the used to assign right hand side calculation result to the left hand side variable.

=comes with the lower precedence so it is done after the execution of statement.

A=a>b;a=?,a=0;

a+b=20; //invalid statement

in left hand side there should be a single variable

* You can declare a variable same as global variable again as a local variable of a block.

**Constant in c:**Constant is a name that has a value that can not be change anywhere during the program.

In c programming you can declare constant in two ways-

1.By using #define

2.By using const keyword

#include<stdio.h>

#define pi 3.14 // Constant declaration by using #define :pi is also a macro

Int main()

{

}

1. By using #define:-

* # does not executes the line ,but replace a name with a value #define a 89.

That means all over the program a will be replace by 89.

* Do not put ; after constant declaration.
* We can not change constant value in the program.

#include<stdio.h>

#define pi 3.14 // Constant declaration by using #define :pi is also a macro

Int main()

{

Int 3.14=10; //Error : We cant not declare a local variable sam as macro.

Pi=pi+10; // Error : You can not change value of macro.

Printf(“%f”,3\*pi);

}

* We cant not declare a local variable sam as macro.
* You can not change value of macro.

#define a 50 //here a is macro

* You can define a function as a macro.

**Program:-**

*// you candeclare a function as a macro*

*#include<stdio.h>*

*#define add(x,y) x-y*

*#deine a 50*

*Void main()*

*{*

*Int var;*

*Var=add(50,20);*

*Printf(“variable =%d”,var);*

*Printf(“addition is =%f”,add(5.7,6.8));*

*}*

**\* You can add multiple line in #define by using \**

#include<stdio.h>

#define add(x,y) printf(“Addition is :%d”,(x+y)) \ Printf(“Substarction is:%d”,(x-y))

Void main()

{

Int a,b;

Printf(“Enter two values:”);

Scanf(“%d %d “,&a,&b); //50 40

Add(a,b);

}

* There is two pre-define macro for to get current date and time

\_\_DATE\_\_

\_\_Time\_\_

#include<stdio.h>

#define add(x,y) printf(“Addition is :%d”,(x+y)) \ Printf(“Substarction is:%d”,(x-y))

Void main()

{

Int a,b;

Printf(“Enter two values:”);

Scanf(“%d %d “,&a,&b); //50 40

Add(a,b);

Printf(“\n Current date is :%s”,\_\_DATE\_\_); //predefine macros

Printf(“\n Current time is :%s”,\_\_Time\_\_);//Pre-Define Macros

}

**How to make variable conastant:-**

**Const Int a,b;**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Date:-21-09-2021**

**Const keyword- const** is the keyword that is used to define constants within the block.

Const data\_type constant\_name=value;

**//wap to calculate simple interest value:(principle amount\*rate\*time)/100**

#include<stdio.h>

void main()

{

const int val=100;// val is the constant

float pa,rate,time;

printf("Enter principle amount:");

scanf("%f",&pa);

printf("Enter rate of interst:");

scanf("%f",&rate);

printf("Enter time of interest:");

scanf("%f",&time);

printf("Total Simple interest is:%.2f",(pa\*rate\*time)/100);

}

**Static Variables-**Static variables are a type of variable which life time does` not destroy even after the closing of block, static variable remains its value safe in the memory even block of variable has been completed

Normal variables destroy after the closing of block but static variable not.

Syntax for static variables:-

static data\_type variable\_name;

**programm-**

//Static variables

#include<stdio.h>

void main()

{

printValue();

printValue();

printValue();

}

void printValue()

{

static int a=0;

a=a+10;

printf("\n Value = %d",a);

}

**Bitwise Operator:-** Bitwise operators are used to performs some task on binary digits. Like 0&1.

&-this is bitwise and

|-This is bitwise or

!-This is bitwise not

^-Bitwise X-OR

BITWISE AND

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Bit1 | Bit 2 | Result |
| & | 1 | 1 | 1 |
| & | 1 | 0 | 0 |
| & | 0 | 1 | 0 |
| & | 0 | 0 | 0 |

BITWISE OR

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Bit1 | Bit 2 | Result |
| | | 1 | 1 | 1 |
| | | 1 | 0 | 1 |
| | | 0 | 1 | 1 |
| | | 0 | 0 | 0 |

BITWISE X-OR OPERATORS

|  |
| --- |
| 1^1=0 |
| 0^0=0 |
| 1^0=1 |
| 0^1=1 |

**Control statement:-** condituional statement are line of program that produce always as true or false.

Conditions can be applied by using relational or logical operators.

Relational or logical expression Always generates outputs in 0 or 1.

i.e. true or false

**1.Conditional statement**

**A.**If Statement:Where conditions are applied by using if keyword.

1. Simple If Statement:-where program has only a single pair of if without else is known as simple if programs.

**Syntax-**

If(condtion)

{

//block of if

}

>,<,>=,<=,==,!=,&&,||,!

**Program:-**

//wap a program to print cube of numbers if number is positive

#include<stdio.h>

int main()

{

int a;

printf("Enter a number:");

scanf("%d",&a);

if(a<0)

{

printf("can not be Cube this number");

}

if(a>0)

{ int b;

printf("Cube is:%d",b=a\*a\*a);

}

}



**Leap year Program:-**

//wap to check a year is a leap year or not.

//divide reminder bache 0 year%4==0,1,2,3

//only if greater than 2000

#include<stdio.h>

void main()

{

int year;

printf("Enter Your Year:");

scanf("%d",&year);

if(year>2000 && year%4==0)

{

printf("Yes it is leap year");

}

}

1. if-else statement:-if there is only one condition is given,and two diifent block of statementhas to execute .One if given condition is true and second if the given condtion is false.Then if-else statement is used.and second if the given condition is false.Then if-else statement is used.

* No semicolon should be after if or false
* No ther statement should be present between if and else block.
* No condition is written with else statement ,conditions always applied with if.
* Else statement cannot be used independently in program it is always used after if statement.

if(condition)

{

//true statement

}

else

{

//else statement

}

**Programm:**

//pass or fail with if else statement

#include<stdio.h>

void main()

{

float per;

printf("Enter Your Percenteg:");

scanf("%f",&per);

if(per>=33)

{

printf("Congratulation !! You are passed.");

}

else

{

printf("Soory! You are failed.");

}

}

**Second Programm:**

//pass or fail with if else statement

//only check if percatage is >=0 or <=100

#include<stdio.h>

void main()

{

float per;

printf("Enter Your Percenteg:");

scanf("%f",&per);

if(per>=0 && per<=100)

{

if(per>=33)

{

printf("Congratulation !! You are passed.");

}

else

{

printf("Soory! You are failed.");

}

}

else

{

printf("You Enterd Invalid input.");

}

}

1. **ladder if Statement**

When there is multiple conditions is given and diiferent statement are given based on the different conditions.

Then ladder else -if statement is used. it ensure that out of all statement only one statement will be executed at the time.

**Syntax:-**

**If(condition)**

**{**

**}**

**else if()**

**{**

**}**

**else if**

**{**

**}**

**else**

**{**

**}**

wap to check result status of student based on the percentage but result should not be pass or fail.

0-32 :fail

33-34:Third division

45-59-second division

60-100-First Division

**Programm:**

//ladder if else program

/\*

wap to check result status of student based on the percentage but result should not be pass or fail.

0-32 :fail

33-44:Third division

45-59-second division

60-100-First Division

\*/

#include<stdio.h>

void main()

{

float per;

printf("Enter Your Percentage:");

scanf("%f",&per);

if(per>=0 && per<=32)

{

printf("! You are failed ! \n Samhal jao nhi to papa ko phone kar dunga.");

}

else if(per>=33 && per<=34)

{

printf("! You are passed with third division \Please Focus on your study !");

}

else if(per>=45 && per<=59)

{

printf("! You are Second division !\ Work Hard beta");

}

else if(per>=60 && per<=100)

{

printf("Congratulation ! You are First division !");

}

else

{

printf("Ye kya Input Kar rhe ho ye sab yha nhi chalta :{ Dusra Number input karo.");

}

}

|  |  |
| --- | --- |
| Char Type Value |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. Nested if Statement

**B.**Switch statement

**2.Iteration statement/looping statement:** Looping statement are some keyword like for while do-while that is used to execute statement multiple time in program.

Type sof loop

There is mainly two type of loop

**Entry Control Loop:-** Where Given condition checked first then block is executed only if given condition is true.

1.**For loop:-** For loop is entry control loop that means given condition with loop is checked first and then statement is executed by compiler.

**Syntax**:

for(initialization ;condition ; updation)

{

}

Initialization: a=5,a=b+2,

Conditions=:a>5 ,a==5,a%2==5%3

Updation :a++,a=a+10,a+=10,a=a%2

Int a;

for(i=1,i==3,i++) // here program does not executes because here value of i=1 is not equal to 3.so program terminate here.

{

//statement of for

}

**Impostant:**

For(i=1;i<21;i+=2)

{

}

**2.While Loop:-**while loop is also a entry control loop where given condition with loop is checked in the starting of loop if the given condition is true then block will be executed or not.

Only condition is written with the while keyword and updation of initialized variable can be done anywhere withing the block of statement.

First condition then statement .This is the concept of entry control loop.

**Exit control loop:** where statement executes first ,then conditions is checked .next time statement will execute or not depends on the condition.

1.**do while loop:** Do while loop is a exit control loop that means firstly block of do will be executed and then after condition will checked.

In the case the given condition is false then also loop will executes for 1 time.Minimum execution time for do while loop is 1 and for while and for loopi is 0.

**Syntax**:

Initialization:

do{

//statement

//updation

}//while(condition);

Each loop has three important parts.

1.**Initializations –** starting of loop

**2.Condition:-**Stopping Point of loop/ending of loop

**3. Updations:-**increment /decrement value on each step

**3.Branching Statement/Transfer Statement/jumping Statement:**

**Fflush:**fflush(stdin);: fflush is the typically out stream function.it is used to move output buffer data to the disk memory.So that output buffer can store next inputed value.

If you will get input of any character/string type value from user, then output buffer saves this data and there is no space next inputed value. So firstly we have to flush or clear the output buffer by using fflush(stdin) method.

Syntax:

fflush(stdin);

//wap to input first character of gender in(m/f) of 10 student and count how many males /females

//fflush ka use karne se koi input skip nhi hota hai

#include<stdio.h>

void main()

{

char gen;

int i,m=0,f=0;

for(i=1;i<=10;i++)

{

printf("enter gender first character(m/f):");

scanf("%c",&gen);

fflush(stdin);

if(gen=='m')

m=m+1;

else if(gen=='f')

f=f+1;

}

}

**Branching Statement:-**Branching statement are used to transfer control from one place to another in the program.in branching statement mainly three keyword are used to transfer control.

1.Break

2.Continue

3.goto

Break a continue statement is used only within the block of loop.

**1.Break:** Break can be used with switch statement or within block of looping statement .

**Syntax:**

Break;

-------------------------

#include<stdio.h>

void main()

{

int i;

for(i=1;i<=10;i++)

{

printf("%d\n",i);

if(i==3)

{

break;

}

}

}

**Switch Statement:** Switch statement is optional of else-if ladder. Switch statement is used to executes blocks based on a single value. when you have to applied multiple conditions based on the single value than instead of else if ladder you can use switch statement.

Switch statement executes blocks base on value not on a conditions.

The syntax of switch statement is:

**Switch(value/exepression)**

{

**Case value:**

//Statement;

Break;

**Case value:**

//Statement;

Break;

}

**Example:**

**// wap to input a number of day and print name of day.**

**#include<stdio.h>**

**void main()**

**{**

**int day;**

**printf("Enter a number of day:");**

**scanf("%d",&day);**

**switch(day)**

**{**

**case(1):**

**printf("This is Monday");**

**break;**

**case(2):**

**printf("This is Tuesday");**

**break;**

**case(3):**

**printf("This is Wednesday");**

**break;**

**case(4):**

**printf("This is Thrusday");**

**break;**

**default:**

**printf("No Day in a week");**

**}**

**}**

**Switch only performed with integer or character type value. Not work with float or double and string type value.**

**Continue Statement:**

Continue statement can only be used withing looping block. continue is used to break the current iteration and continue the loop from the next iteration.

Continue breaks only one iteration but executes for it’s complete time.

**Syntax:**

**Continue;**

**Goto statement:**goto statement is used to transfer control to a pre define label.

Label is identifier that is used to define a position into program.

**Syntax:**

main()

{

Point://labe

-------

//statement

Goto Point;

}

**Task**

**//Wap to input multiple value from the user find sum of all values until user inputs a negative value.**

**//wap to input numbers and print factorial of numbers until user inputes 0 or 1.**

**//wap to find table of all numbers from 1 to 5**

**Example 1 2 3 4 5 6 7 8….**

**2 4 6 8 10…….**

**3 6 9 12 15……**

**//wap to print all prime numbers between 1 to 50**

**Nested loop:-**

whenever a loop block is contained inside another loop block then the set

of blocks looping all together are called as nested loops.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Important point of the nested loops.

1.Every nested block insides a loop statement must be same family

2.Every nested block must executes atleast once ,then only we can say it is nested loops other wise

they are just nested blocks written in a manner that they appear to be nested.

**for(i=0;i<10;i++)**

**{**

**for(j=0;j<10;j++)**

**{**

**}**

**}**

Explaination:-Both syntax appears to be nested in form of loop but they cannot be reagarded as

nested loops.Why,beacuse the condition for the outer loop become false ,hence no of iterations

for the outer loop becomes=0

inner loop iteration=0

**Syntax of nested loops:**

Outer\_loop(<min\_condition-1>)

{

//some statement

Inner\_loop(<min\_condition-2>)

{

//some statement

}

//some statements

}

# - alternate-%:

{}: - <% %>

//prime number from 1 to 100 without using loop

//squre root without using method.h not use square root function loop,operator() no if else not ternary it is possible

Odd Loops:

Odd loops are user defined loop which are subjected to ,run according to user defined condition.

**Why not any loop cannot be used as odd loops**

**So ,the reason is any program can have only minimum one no of transaction.**

Now if the user wants to run any specific no of times he can executes the program.

In such case ,we use menu driven program so that we can build a better program Practice.

Now again there is one doubt sir ,what are menu driven programs.

So let build a menu driven program.

**Types of Loops:**

**1)Uncntroled Loops/Infinite Loops**

**2)Controled loops/Finite loops**

**a)Entry control loop**

Conditions will be checked before hand

For ,while

**b)Exit control loop**

Condition will be checked afterwards

do while,goto

**c)Odd loops**

This is also a finite loops it runs according to user defined input.

Odd Loops are used for menu driven program.(menu driven program means project programs)

Nested Loops:-

for(j=1;j<=

1111

3333

5555

For(i=1;i<=6;i=i+2)

{

For(j=1;j<=4;j++)

{

J=?,i=?

}

}

**Array:-**Array is collection of similar data type when we need to store multiple values of same data type then instead of declaring of all variables separately you can declare a array with fix size.

An array can store multiple values on different index.Indexing of array always starts from 0 and the last element index is size-1

Size of array is a integer value ,it is assign at the compile time.

Types of array:

1.One dimensional Array

2.Two dimensional Array

Int s1,s2,s3,s4,s5;

Int s[5];//declaration of array.

1D Array: 1D Array used to store multiple values sequentially .ID array initialize sequential memory for all indexes of array.

Syntax of declaration is

A[n]:n is size of array

&A[n]: storing data

A[n]:fetch data

**Indexing of arrya is always start with 0.**

2D Array: Two dimensional arrya is a memory organization that stores value in form of rows and columns.Two dimensional arrya is used to store any values of similar data types in form of rows and columns.

Syntax to declare a 2d array:

Data\_type\_var\_name[rows][column];

Int arr[2][3];

0 1 3

|  |  |  |
| --- | --- | --- |
| 0 ar[0][0] | Ar[0][1] | Ar[0][2] |
| 1 ar[1][0 ] | Ar[1][1] | Ar[1][2] |

Total number of elements in two dimensional array is :number of rows\* number of column

Ar[3][3]=9 unique elements of array

Indexing of rows start from 0 and max index of row is row\_size-1.

In same way indexing of column also starts from 0 and max index is column\_size-1

**UDF(User Defined Function):-**User defined function are a block of statement that is written by developer for the self use. Functions are mainly with two types

1.Pre Defined Function

2.User Defined Function

Functions are nothing, it just a name with a block of statement that is used to perform a specific task.

The use of pre defined function are already defined in compiler, we just need to call and use when even we need it.

**User Defined Function: UDF** provides facility of code reusability you can declare a UDF for any specific task and you can call this function multiple time when even you need this code in the program.

There is syntax to define UDF:

Return\_type function\_name (Formal\_Argument\_List)

{

//local

//statement

//return value;

}

**Return Type :**Return type of function is a data type that is always used before the name of function.This return type defines the value that a function returns to the caller.

**Void func1()**//void data type before the function name defines that this function does not return any value to the caller**.**

{

//…………

}

**Int fun1() //** this function will return a integer value to the caller**.**

**{**

**Return**

**}**

**Passing Argument :**Arguments are some local variable of function which is defined with the top parenthesis of function.

**Example:**

Void func1(int a,int b)

{

}

Here func1() is the udf with two integer type parameters .

**Note:**This arguments is known as formal parameters.

When this function will call by caller then caller has to pass value to the function same in length,data\_type and number of arguments.

**Note: Value of arguments variables are assigned by caller.**

**Void add()**

**{**

**Int a,b;**

**Printf(“Enter two number:”);**

**Scanf(“%d %d”,&a,&b);**

**Printf(“Result is=%d”,(a+b));**

**}**

Void add(int a,int b)

{

Printf(“result:%d”,a+b);  
}

A UDF with parameter and without parameter difference:

The main difference is that when we make function without parameter function takes input and then performe task.

In case of without parameter function already know the value of parameter .(man leta hai ki value pahle se assign hui hai).

**Based on the structer of UDF ,UDF can be defined in four different way:**

1.No return Type and no passing arguments

2.No Return Type with passing arguments

3.Return type with no passing arguments

4.Return Type with passing arguments

**Passing Argument:**

**No Passing Argument:**

**Void addition (int a,int b): int a,int b is the formal parameters**

**{**

**}**

**Calling:addition(20,40); - 20,40 is the actual parameters**

**Actual parameters and formal parameters should be same in number.data type.**

**-----------------------------------------------------------------------------------------------**

**Return Type: A UDF with void return type is known as no return type UDF.if a UDF has any other return type except void then this function will return a value to the caller.**

**A single function can return a single value at a time.**

**Return type function must have return keyword within the block.**

UDF:definition,

**String:** String is the collection of characters.in c directly string data type

Is not supported so we have to declare a character type array to hold a string type value.

Mostly used data type of programming is string. In c string data type input and output function are different.

Syntax to declare a string type variable.

Char var\_name[size];

Ex:-

Char name[50];

This name variable is string type variable which can store a value of maximum function of string.

Scanf():which can read only one word string from the user.

Gets(): which can read multi-word or multiline string from user.

Gets(name);

**Need:-String data** type need-to store multi character value String:

**Direct Initialization of string type variable:**

Char str[]=”Techpile”;

Char atr[]={‘T’,’e’,’c’,’h’,’p’,’i’,’l’,’e’};

To input &output of string type value thare is a formate specifier:%s

printf(“%s”,str); //Techpile

**User Input to String type variable:**

**Char str[50];**

Scanf(“50”);

Scanf(“%s”,str);//one-word string

Gets(str);//multi-word string-terminate by enter key

Int arr[5];

To store value on each index ,we executes a loop

But in array of character type(string):there is %s .By using %s,you can store and fetch all index value of array at one time.

Char name[20];

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s | U | R | a | j | \0 |  |  |  |  |

Automatic

0 termination

**//wap to input a string and show input as output**

#include<stdio.h>

void main()

{

char str[50];

printf("Enter your name:");

gets(str);

printf("Hello:%s",str);

}

Length divya rai o/p:8

String Pre-define Function: there are some pre define function in c that makes our task easy. So all string type function are store in a library the library is **string.h** users need to know the syntax of function and you can directly call it whenever you need it.

Strlen(): strlen is a function used to find out the length of the string. It return a unsigned integer. It takes a string type argument.

Int len=strlen(string\_val)

Strcpy():-strcpy() is a function used to copy a string to another string.

It requires two string type argument.

Ex.

char name[100];

name=”divya”//wrong

Strcpy(name,”divya”);

Strcpy(name,firstname);//right way

Strncpy()

Strcmp():-strcmp() is used to compare string that it is equal or not.

If both string are equal this function will return 0 either any positive or negative value it may return.

Int val=Strcmp(string\_1,string\_2);

Val==0 if string\_1 and string\_2 is equal.

Same string different in cases are not equal by strcmp() function.

------------------------------------------------------------------------------------------------

Strcmpi():- it is same as strcmp() but it ignore the cases of characters of both string.only character of string should be same.

It also takes two string type arguments and return a integer type value.

-----------------------------------------------------------------------------------------

Strncmp():it is same as strcmp() so it also compares that two given string are equal or not.

But it compare both string till a given length .it need 3 arguments ,first two of string type and last one of integer type and return a integer type value same as strcmp().

Strnicmp():-strnicmp() is same as strncmp() but it ignores the cases used in both strings.

So it compares two string till n number of characters and ignores the cases of string value.

Syntax:

Int res=strnicmp(striung\_1,string\_2,n)

N=number of character

Ex. int result = strnicmp(“techpile”,”technology”,4);

------------------------------------------------------------------------------------------------

Strlwr():-strlwr() is used to convert all character of string to the lowercase.

It accepts one arguments an set the argument value to the lowercase.

Char name[]=”TECHPILE”;

Syntax:strlwr(name);

//name=”techpile”

Strupr():-strupr() is used to convert all character of string to the upercase.

Its accept only one arguments.

Strcat():-strcat function is used to concatenate two string type value and assign into first arguments.

Syntax:strcat(string\_1,string\_2);

It takes 2 string type value as argument and return nothing.

Strncat():-it is same as strcat but takes 3 arguments.First two argument

Is a string type value to be concatenated and third is a integer type value which shows how many characters of second string will be concatenated to the first string.

Strncat(string\_1,string\_2,6);

Strrev():its reverse the value of a string.it takes only one arguments and after the execution of function string value will be reverse of previous value.

Syntax:strrev(string\_1);

Ex:char name[100]=”name”;

Printf(“%s”,name);//tech

Strrev(name);

Printf(“%s”,name);//hcet

------------------------------------------------------------------------------------------------

**Tasks**

Complete all given task by using Pre-defined functions of C:

WAP to check a given string is palindrome or not.

WAP to input username and password from user and print a message “Welcome To Techpile” if Username is “Techpileuser” and password is “Techpile”.

Note : Ingnore case

WAP to input first name and last name of user and print Full name

WAP to input userid from the user and print “Welcome to techpile” if first 8 characters of string is techpile.

WAP to input name from user and print name with hello . all username letters should be small case.

TECHpile

If(name[i]>=97 && name[i]<=122)

Flag=1;

Else

Flag=0;

Break;

C program to toggle each character of a string. Techpile

In this C program, we are going to learn how to toggle each character of a string? Here, all uppercase characters will be converted into lowercase and all lowercase character will be converted into uppercase.

C program to count digits, spaces, special characters, alphabets in a string.

**Pointers:** Pointer is also a variable that points to the memory address of any variable.Pointer is also a variable declare with \* ,but this type of variable can hold memory address of another variable.

Mainly two symbols used with pointer.

&-& is a symbol used to denote the memory address of a variable.

\*-\* is a symbol used to denote the value of present at a memory address.

Int a;// this is a normal integer type variable.

Int \*a;// this is pointer type variable .Which can hold memory address of a integer type variable.

Declaration of pointer

Int a=8;\*j;

J=&a;

\*j

**Recursion:** Recursion is concept when a function called by itself. When calling of a function is done within the block of same function then this concept is know as recursion.

**Example:**

#include<stdio.h>

void main()

{

printf("this is main.");

main()

}

Structer can hold multiple values of different data type.That means we can say that a structer is the collection of heterogeneous elements.

Mainly structer is a data type ,you can declare multiple variables of struct type.

And one variable would store which type of value is declare by user at the declaration time of structer.

Syntax to declare a structer.

Struct structer\_name;

{

Data\_type variable\_name;

* - - - - - - -
* - - - - - - -

};

Struct book

{

Bid int;

Char bname[100];

Char aname[100];

Float price;

}

**Global Structer:**

**whi**

Struct stru{

Int a;

Char name[20];

}//total size of union=size of element with largest size =20 byte