

CPTC 2020 (ODD)**GROUP - A****1. Choose the most suitable answer from the following options :****1x20=20**

(i) The FILE structure is defined in which of the following files :

- (a) stdlib.h
- (b) stdio.c
- (c) io.h
- (d) stdio.h

Ans. : d

(ii) Which of the following can a format string of printf () function contain:

- (a) Characters, format specifications and escape sequences
- (b) Character, integers and floats
- (c) Strings, integers and escape sequences
- (d) Inverted commas, percentage sign and backslash character

Ans. : a

(iii) Given the statement, maruti. engine. bolts = 25; Which of the following is true ?

- (a) Structure bolts is nested within structure engine.
- (b) Structure engine is nested within structure maruti
- (c) Structure matuti is nested within structure engine
- (d) Structure maruti is nested within stucture bolts

Ans. : b

(iv) Struct time {int hours; int minutes; int seconds;}t; Struct time *pt; pt=&t; with reference to the above declarations, which of the following refers to seconds correctly

- (a) Pt. seconds
- (b) Pt → seconds
- (c) time. seconds
- (d) tiqe → seconds

Ans. : b

v) If int S[5] is a one-dimensional array of integers, which of the following refers to the third element in the array ?

- (a) * (S + 2)

(b) $*(S + 3)$

(c) $S + 3$

(d) $S + 2$

Ans. : d

(vi) Which you pass an array as an argument to a function, what actually gets passed?

(a) Address of the array

(b) Values of the elements of the array

(c) Address of the first element of the array

(d) Number of elements of the array

Ans. : a

(vii) An array is a collection of:

(a) Different data types scattered throughout memory

(b) The same data type scattered throughout memory

(c) The same data type placed next to each other in memory

(d) Different data types placed next to each other in memory

Ans. : c

(viii) What is the difference between the 5's in these two expressions? `int num[5]; num[5] = 11;`

select the correct answer:

(a) First is particular element, second is type

(b) First is array size, second is particular element

(c) First is particular elements, second is array size

(d) Both specify array size

Ans. : b

(ix) What will happen if you assign a value to an element of an array whose subscript exceeds the size of the array?

(a) The element will be set to 0

(b) Nothing is done all the time

(c) Other data may be overwritten

(d) Error message from the compiler

Ans. : c

(x) A preprocessor directive is :

- (a) A message from compiler to the programmer
- (b) A message from compiler to the linker
- (c) A message from programmer to the preprocessor
- (d) A message from programmer to the microprocessor

Ans. : b

(xi) A Header file is :

- (a) A file that contains standard library functions
- (b) A file that contains definitions and macros
- (c) A file that contains user defined functions
- (d) A file that is present in current working directory

Ans. : a

(xii) A do - while loop is useful when we want that the statements within the loop must be executed :

- (a) Only once
- (b) At least once
- (c) More than once
- (d) None of the above

Ans. : b

(xiii) In what sequence the initialization, testing and execution of body is done in a do- while loop?

- (a) Initialization, execution of body testing
- (b) Execution of body, initialization, testing
- (c) Initialization, testing, execution of body
- (d) None of the above

Ans. : a

(xiv) An expression contains relational operators, assignment operators and arithmetic operators and arithmetic operators. In the absence of parenthesis, they will be evaluated in which of the following order.

- (a) Assignment, relational, arithmetic
- (b) Arithmetic, relational, assignment
- (c) Relational, arithmetic, assignment
- (d) Assignment, arithmetic, relational

Ans. : b

(xv) Which of the following statement is used to take the control to the beginning of the loop?

- (a) Exit
- (b) Continue
- (c) Break
- (d) None of the above

Ans. : b

(xvi) Which of the following is odd one out?

- (a) +
- (b) -
- (c) /
- (d) **

Ans. : d

(xvii) In C, Arithmetic instruction cannot contain

- (a) Variables
- (b) Constants
- (c) Variables name on right side of =
- (d) Constants on left side of =

Ans. : d

(xviii) Which of the following is false in C?

- (a) Keywords can be used as variable names
- (b) Variable names can contain a digit
- (c) Variable names do not contain a blank space
- (d) Capital letters can be used in variable Names

Ans. : a

(xix) Hierarchy decides which operator

- (a) is most important
- (b) is used first
- (c) is fastest

(d) Operates on largest numbers

Ans. : b

(xx) The expression $X = 4 + 2\% - 8$ evaluates to

(a) -6

(b) 6

(c) 4

(d) None of the above

Ans. : b

GROUP B

Answer all Five Questions.

4x5=20

Q.2. What is a library function ? Mention any two library function in C.

Ans. : Library functions are built-in functions that are grouped together and placed in a common location called library. Each function here performs a specific operation. We can use this library functions to get the pre-defined output. All C standard library functions are declared by using many header files. These library functions are created at the time of designing the compilers. We include the header files in our C program by using **#include<filename.h>**. Whenever the program is run and executed, the related files are included in the C program.

scanf() Function

scanf() is a formatted input library function which allows to take one or more inputs through standard input device Keyboard. It converts the input data into the specified format and stores into the calculated addresses of the specified memory variable locations. Address of variable is calculated using dereferencing (address) operator '&' which precedes the specified variables. The string of specifiers must be enclosed within pair of double quotes. List of variables are separated by comma. Only string variables having %s or %[...] specifiers do not need dereferencing. Because string variables are itself dereferenced variables.

Syntax : **scanf("format specifier string", list of address of variables);**

In scanf() function, more than one format specifiers can be given separated by comma, space, or tab space. List of address of variables must be written after closing double quotes separated by comma. Each variable must be preceded with address '&' operator, except string variables.

Examples :

Scanf("%c,%d,%f,%o",&ch,&a,&x,&octnum);

Input: S,45,7.98,063 ↵

Scanf("%c %d %f",&ch,&a,&x);

Input: S 45 7.98 ↵

gets() : The gets() function accepts a string from the standard input device keyboard. The length of the string is limited by the declaration of the string variable. The input string may consist of multiple words. The input is

completed by pressing ↵<Enter> key. The input string data is transferred into the said string variable location with a NULL ('\0') character at the end of string.

Syntax : `char *gets(char *);`

Usage : `char name[30];`

`gets(name);`

M	a	h	e	n	d	r	a		S	i	n	g	h		D	h	o	n	i	\0
---	---	---	---	---	---	---	---	--	---	---	---	---	---	--	---	---	---	---	---	----

Action : Mahendra Singh Dhoni ↵

OR (अथवा)

Q.2. Write a c program to print "hello" message using while loop.

Ans. : //Program to print “hello” message using while loop

`#include<stdio.h>`

`#include<conio.h>`

`int main()`

`{`

`int c=1;`

`while(c<=10)`

`{`

`printf(“\nhello”);`

`c++;`

`}`

`getch();`

`return 0;`

`}`

M	a	h	e	n	d	r	a		S	i	n	g	h		D	h	o	n	i	\0
---	---	---	---	---	---	---	---	--	---	---	---	---	---	--	---	---	---	---	---	----

Q.3. Explain the different data types in C.

Ans. : There are 4 Data types in C:

- Basic
- Derived
- Void
- Enumeration

Most of the time, for small programs, we use the basic fundamental data types in C – int, char, float, and double. For more complex and huge amounts of data, we use derived types – array, structure, union, and pointer. Enumeration and void consist of enum and void, respectively. We will discuss these later in the article.

Basic Data Types

These are also termed as primary or fundamental data types. All the names mean the same thing. Suppose we have to store student details like name, id, group, avg_marks, interest_on_fees.

We can use basic data types to store each of these data:

`char name[25];`

```
int id;
char group;
float marks[5];
double interest;
```

int Data Type

Integer types can be signed (with negative values) or unsigned values (only positive). Int values are always signed unless specifically mentioned.

Float

The floating point data type allows the user to type decimal values. For example, average marks can be 97.665. if we use int data type, it will strip off the decimal part and print only 97. To print the exact value, we need 'float' data type.

Float is 4 bytes, and we can print the value using %f.

Double

Double is the double-precision data type. Long Double is treated the same as double by most compilers; however, it was made for quadruple data precision.

char

char stores a single character. Char consists of a single byte.

Derived Data Types

Array, pointers, struct, and union are the derived data types in C.

Array

Same as any other language, Array in C stores multiple values of the same data type. That means we can have an array of integers, chars, floats, doubles, etc

```
int numbers[] = ;
double marks[7];
float interest[5] = ;
```

The array needs to be either initialized, or the size needs to be specified during the declaration.

Pointers

Pointers are considered by many to be complex in C, but that is not the case. Simply put, a pointer is just a variable that stores the address of another variable. A pointer can store the address of variables of any data types. This allows for dynamic memory allocation in C. Pointers also help in passing variables by reference.

The pointer is defined by using a '*' operator. For example –

```
int *ptr;
```

This indicates ptr stores an address and not a value. To get the address of the variable, we use the dereference operator '&.' The size of a pointer is 2 bytes. Pointers cannot be added, multiplied, or divided.

Structs

A struct is a composite structure that can contain variables of different data types. For example, all the student data that we declared earlier in basic data types can be put under one structure. Instead of having the information scattered, when we give it a structure, it is easier to store information about more students.

```
typedef struct{
    char name[25];
    int id;
    char group;
    float marks[5];
    double interest;
}Student;
```

A structure can be created outside the main method as well as inside, just before creating the variable to use it.

```
struct student1, student[20];
```

Structure members can be accessed using the dot(.) operator.

Elements in structure can be accessed using pointers too. Instead of the dot operator, we are using '->' operator to fetch the values.

Structs are simple to use and combine data in a neat way.

Union

With a union, you can store different data types in the same memory location. The union can have many members, but only one member can have a value at one time. Union, is thus, a special kind of data type in C.

The union is defined in the same way as a structure but with the keyword union.

```
union Student{
    char name[25];
```

```

int id;
char group;
float marks[5];
double interest;
}st1, st2;

```

When we assign values to union data, union allocates enough memory to accommodate the largest data type defined.

Enumeration

Enumeration data types enhance the readability of the code. If you have integer constants in the code that can be reused or clubbed together, we can use enums to define the constants. The most common example of this is the days of the week.

```

enum weekdays;
enum weekend;

```

Void

The void is just an empty data type used as a return type for functions. The absence of any other data type is void. When you declare a function as void, it doesn't have to return anything.

OR (अथवा)

Q.3. What is the general form of "do-while" statement ? Explain with example.

Ans. : `do...while()` is an **Exit-Controlled** Looping construct. In `do ... while ()` statement, it is not known in advance that how many times a statement will be executed. It is known that the statement will be executed at least once. In that case, there is no need to test the condition at the beginning. The condition is tested at the end followed by **semicolon (;)** to determine that should the body of the loop be executed again or not. On getting **TRUE** result of condition, control goes back to execute next pass to the body of loop. On getting **FALSE**, the loop terminates. The loop control variable must be initialized before initiating loop. The update expression to modify the value of loop control variable is placed inside the body of loop.

Syntax :

```

expr1; //Initialize Loop variable
do
{
    Statement-Block;
    expr3; // Looping variable update expression
} while ( expr2 ); // testing the condition

```

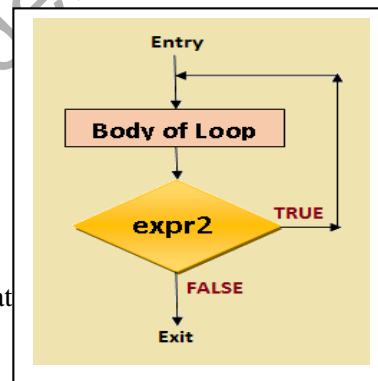
where `expr2` is a constant, a variable or an expression. The statement is executed repeatedly till the expression evaluates to a non-zero (True) value.

Example. :

```

int n = 1;
do
{
    printf("%5d", n);
    ++n;
} while (n<=100);

```



Q.4. How does X++ differ from ++X ?

Ans. : **Increment Operator** is an unary operator used to increase the value of a numeric operand by 1. Usually it operates on Integer and single precision float data. The symbol used for increment operator is `++`. It is accessed from Right to Left. There are two types of increment operators :

- i) pre-increment
- ii) post-increment

++x : When increment operator “++” is applied as prefix (i.e., just on left side) to the operand, it refers as Pre-increment operation. In this case, first the value of operand is incremented by 1 in memory and then the value of the operand is accessed (used).

Ex: `int x = 5;`

`printf(“%d”,++x);`

It will output the value 6 on screen.

X++ : When increment operator “++” is applied as postfix (i.e., just on right side) to the operand, it refers as Post-increment operation. In this case, first the value of operand is accessed (used) and then incremented by 1 in memory.

Ex: `int x = 5;`

`printf(“%d”,x++);`

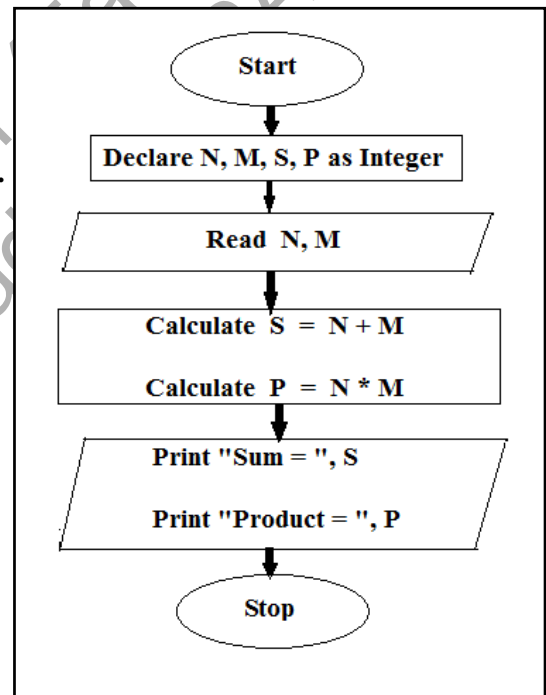
It will output the value 5 on screen.

OR (अथवा)

Q.4. Write the algorithm and draw the flowchart to find the sum and product of given two numbers.

Ans. :

- | | |
|---------|--|
| Step-01 | Start |
| Step-02 | Rem Algo to find Sum and Product of two nos. |
| Step-03 | Declare N, M, S, P as Integer |
| Step-04 | Read N, M |
| Step-05 | Calculate $S = N + M$ |
| Step-06 | Calculate $P = N * M$ |
| Step-07 | Print “Sum =”, S |
| Step-08 | Print “Product =”, P |
| Step-09 | Stop |



Q.5. Write a c program to find the sum and average of given five numbers.

Ans. :

//Program to calculate average of given 5 nos

`#include<stdio.h>`

`int main()`

`{`

```

Enter any 5 integer nos :
56
45
89
23
49

Average of 5 nos = 52.000000
  
```

```

int n, i=0, sum=0;

float avg;

printf("\nEnter any 5 integer nos : ");

while(i<5)
{
    scanf("%d",&n);

    sum += n;

    i++;
}

avg = sum/i;

printf("\nAverage of 5 nos = %f", avg);

return 0;
}

```

OR (अथवा)

Q.5. Convert the following mathematical expressions into C expressions.

(i) $\sqrt{1+x} + \frac{\log \cos 2x}{1+|y|}$

(ii) $T = \sin a \cos b - |g - h| + \sqrt{ab}$

Ans. :

(i) `sqrt(1+x) + (log(cos(2*x)))/(1+mod(y))`

(ii) `T = sin(a) * cos(b) - mod(g-h) + sqrt(a*b)`

Q.6. Explain with examples the syntax of scanf () and printf () functions.

Ans. :

scanf() Function

`scanf()` is a formatted input library function which allows to take one or more inputs through standard input device Keyboard. It converts the input data into the specified format and stores into the calculated addresses of the specified memory variable locations. Address of variable is calculated using dereferencing (address) operator '&' which precedes the specified variables. The string of specifiers must be enclosed within pair of double quotes. List of variables are separated by comma. Only string variables having %s or %[...] specifiers do not need dereferencing. Because string variables are itself dereferenced variables.

Syntax : `scanf("format specifier string", list of address of variables);`

In scanf() function, more than one format specifiers can be given separated by comma, space, or tab space. List of address of variables must be written after closing double quotes separated by comma. Each variable must be preceded with address '&' operator, except string variables.

Input Variable Declaration:

```
int      a, b;;
unsigned int  num, hexnum, octnum;
long      dist;
float      x, y;
char      ch, name[30], msg[100];
```

Examples :

```
Scanf("%c,%d,%f,%o",&ch,&a,&x,&octnum);
```

```
Input: S,45,7.98,063 ↵
```

```
Scanf("%c %d %f",&ch,&a,&x);
```

```
Input: S 45 7.98 ↵
```

```
Scanf("%x,%u %ld",&hexnum,&num,&dist);
```

```
Input: 0x2b3,34545 789432 ↵
```

```
Scanf("%s",name);
```

```
Input: Ajay Kumar Singh ↵
```

```
Scanf("%[...]",msg);
```

```
Input: Everybody has to attained the class. ↵
```

printf() Function

printf() function is a formatted library output function which sends data from specified memory variable onto the standard output device, Monitor screen. It reads data from the variable location and convert it according to the specified format specifier. It displays the output data at the current cursor position. This function also permits to print string constants. Specifiers must be enclosed within pair of double quotes. The list of output variables must be separated by comma in the list. This function also allows to use escape sequence characters along with specifier string within the double quotes. It displays the output as the format string is customized. It replaces the format specifiers with the value of variables.

Syntax :

```
printf("string constant");
```

```
printf("format specifier string", list of variables);
```

In printf() function, is frequently used to interact with user of program through messages displayed on screen helping the user for required input operations. The output of the program can be displayed in customized format by placing messages along with format specifiers and the list of variables whose

output data are to be displayed on screen in the expected format. The format specifier string can have message as string constant including escape sequence characters, comma, space, or tab space. List of address of variables must be written after closing double quotes separated by comma.

Format Specifiers forms:

%[±] w.p type-specifier

String Specifier : %±w.ps

W : field width of output data in no. of column occupied on output screen, must be integer constant.

P : first column position of the text started to be displayed

+ : Right aligned text

- : Left aligned text

Ex:

```
printf("%d , %.2f , %s", num, avg, name);
```

OR (अथवा)**Q.6. What is a Pointer? Explain with the help of examples.**

Ans. : A Pointer is a variable that represents the location (rather than the value) of a data item within the memory, such as a variable or an array element. Pointers can be used to pass information back and forth between a function and its reference point. Pointers provide a way to return multiple data items from a function via function arguments. Pointers also permit references to other functions to be specified as arguments to a given function. This has the effect of passing functions as arguments to the given function. Every data item stored within memory. It occupies one or more contiguous memory cells (i.e., adjacent words or bytes). The number of memory cells required for storing data item depends on the type of data items. The data item can be accessed by its address within memory. This address of location can be determined by a variable preceded by '&' called address operator, that evaluates the address of its operand. Let us assign the address of a variable say v to another variable pv.

pv = &v will store the address of variable v into pv.



Where **&** is a **Unary Operator** always printed by '%u' format string. Pointer variables must be declared before use. When a pointer variable is declared, the variable name must be preceded by an **Asterisk (*)**. This identifies the fact that the variable is a pointer. The data type that appears in the declaration refers to the Object of the pointer, i.e., the item that is stored in the address represented by the pointer, rather than the pointer itself.

Syntax : **datatype *pointer_variable_name;**

Ex.:- **int *ptr; int x, *px;**

px = &x; // Storing address of x into pointer variable px

GROUP - C

Answer all Five Questions.

6x5=30

Q.7. Write a C program to add two M x N matrices and store the results in third matrix.

The value of M and N are given by user.

Ans. :

//Addition of two matrices

#include<stdio.h>

#include<conio.h>

int main()

{

int arr[3][4],mat[3][4],sum[3][4],i,j;

```

printf("\n\n\tEnter 12 numbers to first Matrix:\n");

for(i=0 ; i<3 ; i++) //for ROW

    for(j=0 ; j<4 ; j++) // for COL

        scanf("%d",&arr[i][j]);

printf("\n\n\tEnter 12 numbers to Second Matrix:\n");

for(i=0 ; i<3 ; i++) //for ROW

    for(j=0 ; j<4 ; j++) // for COL

        scanf("%d",&mat[i][j]);

//Adding the two matrices

for(i=0 ; i<3 ; i++) //for ROW

    for(j=0 ; j<4 ; j++) // for COL

        sum[i][j] = arr[i][j] + mat[i][j];

printf("\n\n\tData from first Source Matrix:\n\n\t");

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

{

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

        printf("%4d",arr[i][j]);

    printf("\n\n\t");

}

printf("\n\n\tData from second Source Matrix:\n\n\t");

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

{

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

        printf("%4d",arr[i][j]);

    printf("\n\n\t");

}

printf("\n\n\tData from resultant Matrix:\n\n\t");

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

{

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

```

```

Enter 12 numbers to first Matrix:
10
11
12
13
14
15
16
17
18
19
21
22

Enter 12 numbers to Second Matrix:
11
22
33
44
55
66
77
88
99
111
222
333

Data from first Source Matrix:
10 11 12 13
14 15 16 17
18 19 21 22

Data from second Source Matrix:
10 11 12 13
14 15 16 17
18 19 21 22

Data from resultant Matrix:
21 33 45 57
69 81 93 105
117 130 243 355

```

```
printf("%4d",sum[i][j]);

printf("\n\n\t");

}

getch();

return 0;

}
```

OR (अथवा)

Q.7. Define the following terms:

- (a) Source program
- (b) Object program
- (c) Executable program

Ans. :

(a) Source program :

Source code is the fundamental component of a computer program that is created by a programmer. It can be read and easily understood by a human being. When a programmer types a sequence of C programming language statements into Windows Notepad, for example, and saves the sequence as a text file, the text file is said to contain the source code. Source code and object code are sometimes referred to as the "before" and "after" versions of a compiled computer program. For script (noncompiled or interpreted) program languages, such as JavaScript, the terms source code and object code do not apply, since there is only one form of the code. Programmers can use a text editor. In large program development environments, there are often management systems that help programmers separate and keep track of different states and levels of source code files. Beyond providing the foundation for software creation, source code has other important purposes, as well. For example, skilled users who have access to source code can more easily customize software installations, if needed. Access to source code also allows programmers to contribute to their community, either through sharing code for learning purposes or by recycling portions of it for other applications.

(b) Object program :

Object code is the **output of a compiler** after it processes the **source code**. The object code is usually a **machine code**, also called a **machine language**, which can be **understood directly by a specific** type of CPU (central processing unit), such as x86 (i.e., Intel-compatible) or PowerPC. However, some compilers are designed to convert source code into an **assembly language** or some other another programming language. An assembly language is a human-readable notation using the **mnemonics** (mnemonics is a **symbolic** name for a single executable machine language instruction called an **opcode**) in the ISA (Instruction Set Architecture) of that particular CPU .

(c) Executable program :

Software in a form that can be run in the computer. Executable code generally refers to machine language, which is the set of native instructions the computer carries out in hardware. Executable files in the DOS/Windows world use .EXE and .COM file extensions, while executable files in Mac, Linux and Unix do not require specific

extensions. They are identified by their file structure.

Interpreted Languages Require One More Step

Executable code may also refer to programs written in interpreted languages that require additional software to actually execute. Some interpreted languages remain in their source code form, such as JavaScript and VBScript, while others are compiled into an intermediate language, such as Java and Visual Basic. Interpreted languages require software "runtime engines" to convert the program into executable instructions for a particular CPU family that the hardware executes.

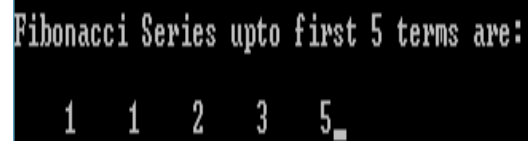
Q.8. Write a C program to compute fibonacci series up to five terms.

Ans. : //Fibonacci Series upto first 5 terms

```
#include<stdio.h>

#include<conio.h>

main()
{
    int a=1,b=1,c,count;
    printf("\n\nFibonacci Series upto first 5 terms are:\n\n");
    printf("%5d%5d",a,b);
    for(count=3;count<=5;count++)
    {
        c=a+b;
        printf("%5d",c);
        a=b;
        b=c;
    }
    getch();
    return 0;
}
```



Fibonacci Series upto first 5 terms are:
1 1 2 3 5

OR (अथवा)

Q.8. Write a C program to find the roots of quadratic equation $ax^2 + bx + c = 0$ for all possible combinations of a, b, c.

Ans. :

/* To calculate the Roots of a Quadratic Equation $A \cdot X^2 + B \cdot X + C = 0$ */

```

#include<stdio.h>

#include<conio.h>

#include<math.h>

main()
{
    float a,b,c,d,z,r1,r2;

    printf("\n\nEnter Values of A B C : ");

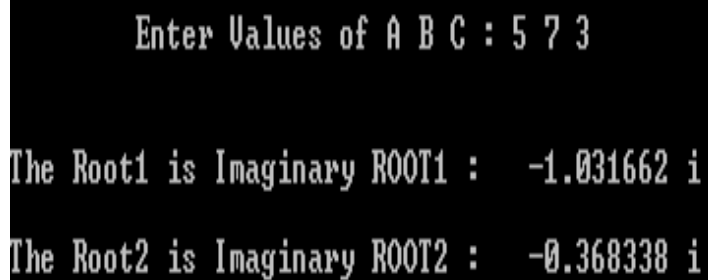
    scanf("%f %f %f",&a,&b,&c);

    d = b*b - 4*a*c;

    z = sqrt(abs(d));

    r1 = (-b - z) / ( 2 * a );
    r2 = (-b + z) / ( 2 * a );
    if (d<0)
    {
        printf("\n\nThe Root1 is Imaginary ROOT1 : %10.6f i",r1);
        printf("\n\nThe Root2 is Imaginary ROOT2 : %10.6f i",r2);
    }
    else
    {
        printf("\n\nThe Root1 is Real ROOT1 : %10.6f",r1);
        printf("\n\nThe Root2 is Real ROOT2 : %10.6f",r2);
    }
    getch();
    return 0;
}

```



```

Enter Values of A B C : 5 7 3

The Root1 is Imaginary ROOT1 :  -1.031662 i
The Root2 is Imaginary ROOT2 :  -0.368338 i

```

Q.9. Define a structure with the following three numbers: roll number, name and total marks of student. Write a C program to read and display the details of student.

Ans. :

//Program to print student record


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

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

```
struct student
```

```
{
```

```
    int roll;
```

```
    char name[30];
```

```
    int total_marks;
```

```
};
```

```
int main()
```

```
{
```

```
    struct student s;
```

```
    printf("\n\tEnter Roll : ");
```

```
    scanf("%d",&s.roll);
```

```
    printf("\n\tEnter Name of Student : ");
```

```
    fflush(stdin);
```

```
    gets(s.name);
```

```
    printf("\n\tEnter Total Marks obtained : ");
```

```
    scanf("%d",&s.total_marks);
```

```
    printf("\n\n\tThe data from structure :\n\n");
```

```
    printf("\n\tROLL No. : %d",s.roll);
```

```
    printf("\n\tNAME : %s",s.name);
```

```
    printf("\n\tTotal Marks Obtained : %d",s.total_marks);
```

```
    printf("\n\n\tThe Size of Structure STUDENT : %d",sizeof(s));
```

```
    printf("\n\tThe Address of Structure Student : %u",&s);
```

```
    printf("\n\tThe Address of ROLL in Structure Student : %u",&s.roll);
```

```
    printf("\n\tThe Address of NAME in Structure Student : %u",&s.name);
```

```
    printf("\n\tThe Address of Total Marks in Structure Student : %u",&s.total_marks);
```

```
    getch();
```

```
    return 0;
```

```
}
```

```
Enter Roll : 102
Enter Name of Student : Ajay Kumar Singh
Enter Total Marks obtained : 467

The data from structure :

ROLL No. : 102
NAME : Ajay Kumar Singh
Total Marks Obtained : 467

The Size of Structure STUDENT : 40
The Address of Structure Student : 2358816
The Address of ROLL in Structure Student : 2358816
The Address of NAME in Structure Student : 2358820
The Address of Total Marks in Structure Student : 2358852_
```

OR (अथवा)**Q.9. What do you mean by machine language ? How it differs from high level language?****What are the advantage of high level language?**

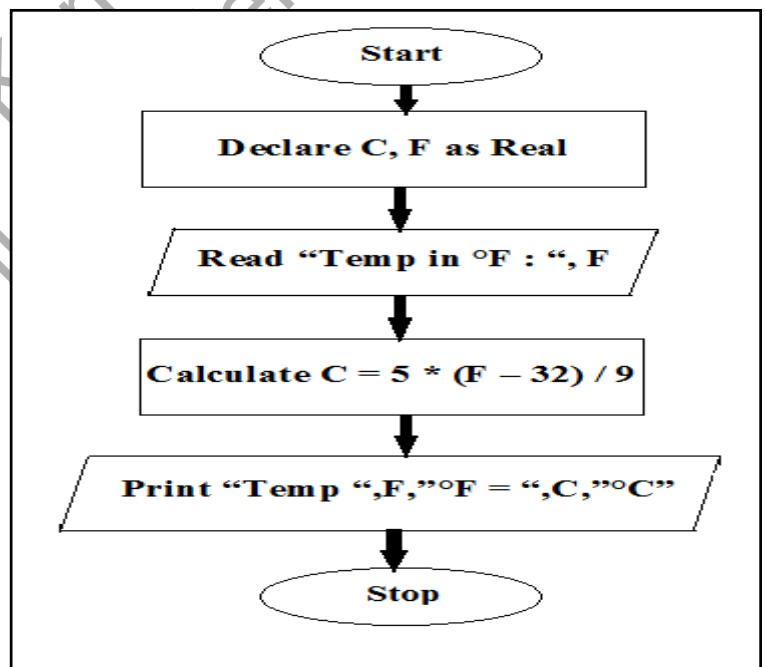
Ans. : Machine Language : The first language was binary, also known as machine language, which was used in the earliest computers and machines. We know that computers are digital devices, which have only two states, ON and OFF (1 and 0). Hence, computers can understand only two binary codes. Therefore, every instruction and data should be written using 0s and 1s. Machine language is also known as the computer's 'native' language as this system of codes is directly understood by the computer. Instruction in machine language consists of two parts. The first part is an operation, which tells the computer what functions are to be performed. The second part of the instruction is the operand, which tells the computer where to find or store the data on which the desired operation is to be performed. A binary program is a long list of instructions that are executed by the CPU. Normally, instructions are executed one after the other, but program flow may be influenced by special jump instructions that transfer execution to an instruction other than the following one. Each computer has its own set of instructions based on its architecture. Hence, machine language may differ from computer to computer.

High Level Language : High-level languages are similar to English language. Programs written using these languages can be machine independent. A single high-level statement can substitute several instructions in machine or assembly language. Unlike assembly and machine programs, high-level programs may be used with different types of computers with little or no modification, thus reducing the re-programming time. In high-level language, programs are written in a sequence of statements to solve a problem.

Q.10. Write the algorithm and draw the flow chart to convert the temperature in °F to °C using the formula $^{\circ}\text{C} = 5 (F - 32) / 9$.

Ans. :

- | | |
|---------|------------------------------------|
| Step-01 | Start |
| Step-02 | Declare C, F as Real |
| Step-03 | Read "Temp in °F : ", F |
| Step-04 | Calculate $C = 5 * (F - 32) / 9$ |
| Step-05 | Print "Temp ", F, "°F = ", C, "°C" |
| Step-06 | Stop |

**OR (अथवा)****Q.10. Define function. Explain prototype of function,**

Ans. : Function : A Function is a self-contained program segment that carries out some specific, well-defined task. Every 'C' program consists of one or more functions. One of these functions must

be called main. If a program contains multiple functions, their definitions may appear in any order, though they must be independent of one another. That is, one function definition cannot be embedded within another. A function will carry out its intended action whenever it is accessed (i.e., Called) from some other portion of the program. Once the function has been carried out its intended action, control will be returned to the point from the function was accessed. Generally, a function will process information passed to it from the calling portion of the program, and return a single value. Information will be passed to the function through special identifier called Arguments or Parameters and returned through return statement.

Function Prototypes : A function PROTOTYPE is a function declaration that specifies the return type and the data types of the arguments. The main purpose of the function prototyping is to prevent errors by data type mismatches between the values function is expecting. In particular, the proposed ANSI standard permits each of the argument data types within function declaration to be followed by an argument name, that is ,

Syntax :

data_type function_name (type1 arg1, type2 arg2, , type-n arg-n) ;

Where arg1, arg2, arg3, , arg-n refer to the first argument, the second argument and so on. Function declaration written in this form is called Function Prototype. Their use is not mandatory in 'C'.

Ex. :-- int add (int a , int b);
 long int factorial (int n);
 void reverse (void);
 float process (float radius);
 float largest (float x , float y , float z);

Q.11. Explain the following terms:

- (a) Break
- (b) Continue
- (c) Exit

Ans. :

(a) Break : The break statement transfers the control out of the block, where it is used. It is also used to exit from a switch or terminate loops. It can be used within a while(), do...while(), for() or a switch() statements.

- A break statement can be used to terminate or to come out from the loop or conditional statement unconditionally.
- It can be used in switch statement to break and come out from the switch statement after each case expression.
- Whenever, break statement is encounter within the program then it will break the current loop or block.
- A break statement is normally used with if statement.
- When certain condition becomes true to terminate the loop then break statement can be used.

Syntax : **break ;**

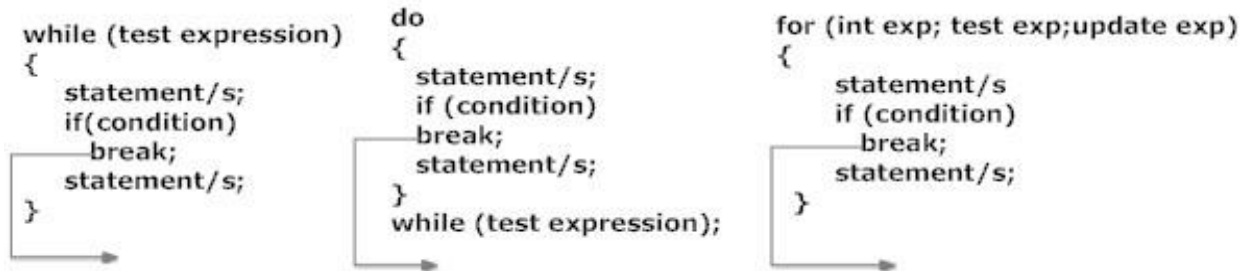
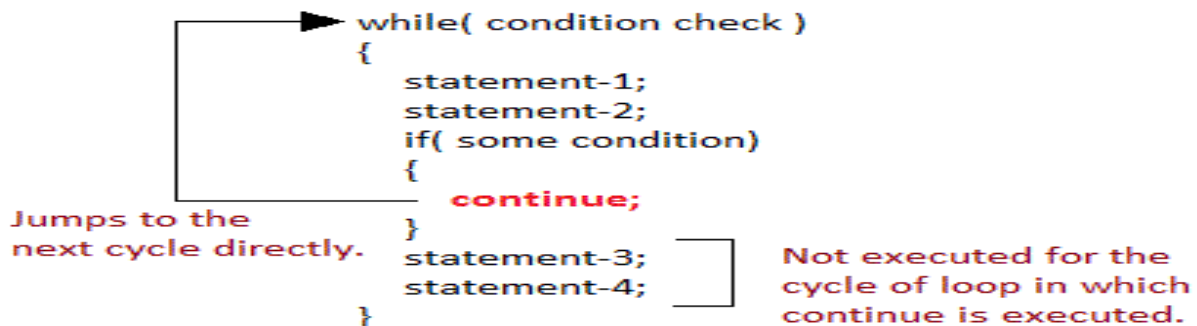


Fig: Working of break statement in different loops

(b) Continue : The continue statement is used to bypass the remainder of the current pass through a loop. The loop does not terminate when a continue statement is encountered. Rather, the remainder loop statements are skipped and the computation proceeds directly to the next pass through the loop.

- A *continue* statement can be used into the loop when we want to skip some statement to be executed and continue the execution of above statement based on some specific condition.
- Similar to break statement, *continue* is also used with if statement.
- When compiler encounters continue, statements after continue are skip and control transfers to the statement above continue.

Syntax : **continue ;**



(c) Exit :

- An *exit* statement is used to terminate the current execution flow.
- It is an in-built system function it is available in ***process.h*** header file.
- As soon as exit statement is found, it will terminate the program.
- It has single argument of zero. For example: ***exit(0);***
- the following program demonstrates the use of both *break* and *exit* statements.

OR (अथवा)

Q.11. Write down the general syntax of switch statement and explain with the help of example.

Ans. :

Switch () Case Statement

The **switch** statement causes a particular group of statement to be chosen from several available groups expressed under **case** followed by a **value**. The selection is based upon the current

value of an expression that is included within the switch statement and control is transferred in that section of the matching **case value**.

Syntax :

Where expression takes any given value from val-1, val-2, , val-N, the control is transferred to that appropriate case.

In each case, the statements are executed and then the break statement transfers the control out of switch statement.

The default keyword is, usually mentioned at the end of switch statement, used if the value of the expression does not match any of the case values.

Solution of CPTC 2020 (ODD)
By: Ajay Kumar (Faculty)
Polytechnic Academy, PATNA