Royal Education Society's

College of Computer Science and Information Technology, Latur.

Department of Computer Science

Academic Year (2022-23)

Choice Based Credit System (CBCS Revised) Class: **B.Sc.(CS) FY SEM-I**

Name of Paper: **Programming in C (BCS- 102)** Prepared by: **Mr. Jogdand V.S.**

Instructions to the candidates:

1. All questions are Compulsory.

2. Figures to the right indicate full marks.

3. Assume suitable data, if required.

Q.1 Attempt any FIVE of the following (3 Marks each) 15

a) Explain Keywords.

b) Explain Algorithms.

c) What is if statements.

d) What is array.

e) Explain Machine Language.

f) Explain break statement.

g) Explain history of C.

Q. 2 Attempt any three of the following (5 Marks each) 15

a) Write a program to find factorial of given no.

b) Explain Operators

C) Which are Unformatted I/O statement's:

d) Write a program using array addition of two matrices.

e) Explain While loop.

Q. 3 Attempt any three of the following (5 Marks each) 15

a) Explain else-if-ladder statements.

b) Write a program to find given no. is odd or even.

C) Explain Passing arrays to function.

d) Explain structure of c program:

e) What is difference between Compilers and Interpreters:

Q. 4 Attempt any three of the following (5 Marks each) 15

a) Explain witch statements.

b) Write a program using array to find Larges Element in an array.

c) Explain formatted I/O Statements.

d) Explain do-while loop.

e) Write a sum of digits' program in C.

Q .5 Short notes on any three of the following (5 Marks each) 15

a) Variables

b) Data types:

c) Nested for loop.

d) go to statements.

e) High level languages.

Q.1 Attempt any FIVE of the following (3 Marks each) 15

a) Explain Keywords.

Answer:

1. **Definition**: "Keywords are the words whose meaning has already been explained to the C compiler and their meanings cannot be changed". Hence keywords are also called 'Reserved words'.
2. Keywords can be used only for their intended purpose.
3. Keywords serve as basic building blocks for program statements.
4. Keywords can't be used as programmer defined identifier.
5. The keywords can't be used as names for variables.
6. All keywords must be written in lowercase.
7. 32 keywords available in C.
8. auto, break, case, char, const, continue, default, int, float, const, double, char etc.
9. For example:
10. double and float: Both keywords double, as well as float, are needed for declaration of floating type variables.

b) Explain Algorithms.

Answer:

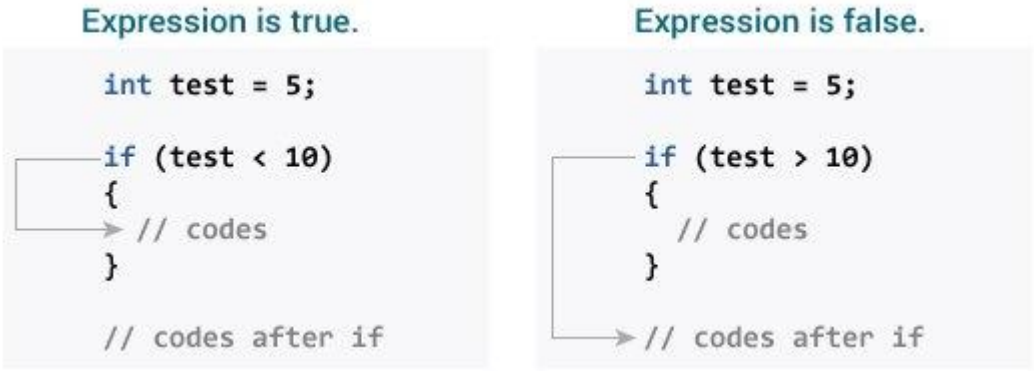
1. **Definition: a**n algorithm is a set of well-defined instructions to solve a particular problem. It takes a set of input and produces the desired output.
2. **Characteristics:**
   a) **Clear and Unambiguous**: The algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
   b) **Well-Defined Inputs**: If an algorithm says to take inputs, it should be well-defined inputs. It may or may not take input.
   c) **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well. It should take at least 1 output.
   d) **Finite-ness:** The algorithm must be finite, i.e. it should terminate after a finite time.
   e) **Feasible:** The algorithm must be simple, generic, and practical, such that it can be executed with the available resources. It must not contain some future technology or anything.
   f) **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.
3. Example: Add two numbers entered by the user

   Step 1: Start
   Step 2: Declare variables num1, num2 and sum.
   Step 3: Read values num1 and num2.
   Step 4: Add num1 and num2 and assign the result to sum.
   sum←num1+num2
   Step 5: Display sum
   Step 6: Stop

c) What is if statements.

Answer:

1. It is one of the powerful conditional statement.
2. If statement is responsible for modifying the flow of execution of a program.
3. If statement is always used with a condition.
4. If the test expression is evaluated to true, statements inside the body of **if** are executed.
5. If the test expression is evaluated to false, statements inside the body of **if** are not executed.
6. Syntax: I) if (condition)
   Instruction;
   II) if(condition)
   {
   Instruction;
   }
7. For example:



d) What is array.

Answer:

1. **Definition:** An array is a collection of data items, all of the same type, accessed using a common name.
2. Explanation: All elements of array are stored in the contiguous memory locations.
3. The size of array must be a constant integral value.
4. Individual elements in an array can be accessed by the name of the array and an integer enclosed in square bracket called subscript/index variable like employee Salary [5].
5. Array is a random access data structure. you can access any element of array in just one statement.
6. The first element in an array is at index 0, whereas the last element is at index (size_of_array - 1).
7. Declaration of Array:
   a) Array Declaration by Specifying the Size
   // declare an array by specifying size in [].
   int my_array1[20];
   b) array Declaration by Initializing Elements:
   // initialize an array at the time of declaration.
   int my_array[] = {100, 200, 300, 400, 500}

8. Array Declaration by Specifying the Size and Initializing Elements

```
// declare an array by specifying size and

// initializing at the time of declaration

  //int my_array1[5] = {100, 200, 300, 400, 500};

// my_array1 = {100, 200, 300, 400, 500}

//  int my_array2[5] = {100, 200, 300};  // my_array2 = {100, 200, 300, 0, 0}
```

e) Explain Machine Language.

Answer:

1. Machine language is a low-level language made up of binary numbers or bits that a computer can understand.
2. The only language that the computer understands is machine language
3. When a specific task, even the smallest process executes, machine language is transported to the system processor. Computers are only able to understand binary data as they are digital devices.
4. In the computer, all data like videos, programs, pictures are represented in binary.
5. Machine language is a low-level language that machines understand but that humans can decipher using an assembler.
6. A compiler plays an important role between humans and computers as it converts machine language into other code or language that is understandable by humans.

7. Advantages:

a) High speed execution

b) The computer can have understood instructions immediately

c)No translation is needed.

8. Disadvantages:

a) Machine dependent
b) Programming is very difficult
c) Difficult to understand
d) Difficult to write bug free programs
e) Difficult to isolate an error

f) Explain break statement.

Answer:

1. **Definition**: The break statement terminates the execution of the nearest enclosing do, for, switch, or while statement in which it appears. Control passes to the statement that follows the terminated statement.
2. The break is a keyword in C which is used to bring the program control out of the loop.
3. The break statement is used inside loops or switch statement.
4. The break statement breaks the loop one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops

5.

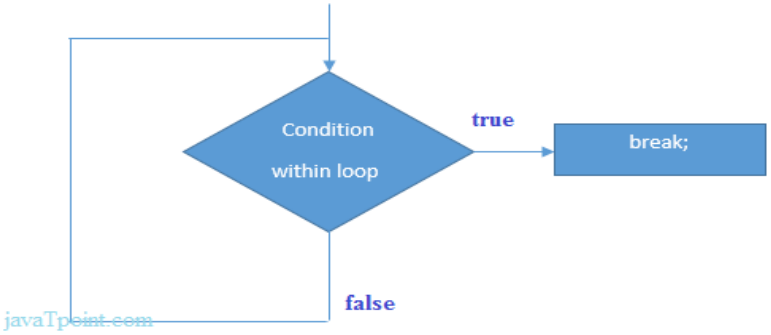


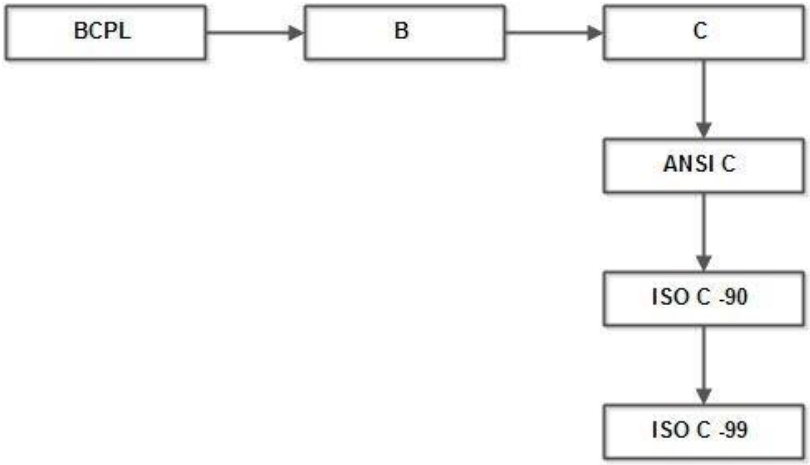
Figure: Flowchart of break statement

6. The break statement ends the loop immediately when it is encountered.
7. **Its syntax is:**
   break;

g) Explain history of C.

Answer:

1. C is a general-purpose language which has been closely associated with the UNIX operating system for which it was developed - since the system and most of the programs that run it are written in C.
2. Many of the important ideas of C stem from the language BCPL, developed by Martin Richards.
3. The influence of BCPL on C proceeded indirectly through the language B, which was written by Ken Thompson in 1970 at Bell Labs, for the first UNIX system on a DEC PDP7.
4. BCPL and B are "type less" languages whereas C provides a variety of data types.
5. In 1972 Dennis Ritchie at Bell Labs writes C and in 1978 the publication of The C Programming Language by Kernighan & Ritchie caused a revolution in the computing world. In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C.
6. The resulting definition, the ANSI standard, or "ANSI C", was completed late 1988.



7.

Q. 2 Attempt any three of the following (5 Marks each) 15

a) Write a program to find factorial of given no.

Answer:

```
#include<stdio.h>
#include<conio.h>
int main()
{
 int i,fact=1,number;
printf("Enter a number: ");
  scanf("%d",&number);
 for(i=1;i<=number;i++){
 fact=fact*i;
 }
 printf("Factorial of %d is: %d",number,fact);
return 0;
}
```

Output:

```
Enter a number: 5

  Factorial of 5 is: 120
```

b) Explain Operators

Answer:

1. **Operators:** A sign or Symbol used to perform arithmetic or logical operation is called as operators.
2. There are 8 types of operators:

C language offers many types of operators. They are,

a. Arithmetic operators
b) Assignment operators
c) Relational operators
d) Logical operators
e) Bit wise operators
f) Conditional operators (ternary operators)
g) Increment/decrement operators
h) Special operators

a) ARITHMETIC OPERATORS IN C

C Arithmetic operators are used to perform mathematical calculations like addition,

subtraction, multiplication, division and modulus in C programs.

Arithmetic Operators Operation Example

1 + Addition A+B

2 – Subtraction A-B

3 * multiplication A*B

4 / Division A/B

Eg: add = a+b;

b) ASSIGNMENT OPERATORS IN C

In C programs, values for the variables are assigned using assignment operators.

For example, if the value ―10‖ is to be assigned for the variable ―sum‖, it can be assigned as

―sum = 10;‖

Operators Example Explanation

Simple assignment operator = sum = 10

10 is assigned to variable sum

Compound assignment operators

sum += 10

This is same as sum = sum + 10

c) RELATIONAL OPERATORS IN C

Relational operators are used to find the relation between two variables. i.e. to compare the values of two variables in a C program.

EXAMPLE PROGRAM FOR RELATIONAL OPERATORS IN C

In this program, relational operator (==) is used to compare 2 values whether they are equal are not.

If both values are equal, output is displayed as ‖ values are equal‖. Else, output is displayed as ―values are not equal‖.

Note: double equal sign (==) should be used to compare 2 values. We should not single equal sign (=).

d) Logical Operators

-These operators are used to perform logical operations on the given expressions.
-There are 3 logical operators in C language. They are, logical AND (&&), logical OR (||) and logical NOT (!).

S.no	Operators	Name	Example	Description

1	&&	logical	AND	(x>5)&&(y<5) It returns true when both conditions are true
2	||	logical	OR	(x>=10)||(y>=10) It returns true when at-least one of the condition is true.
3	!	logical	NOT	!((x>5)&&(y<5)) It reverses the state of the operand

e) BIT WISE OPERATORS IN C

These operators are used to perform bit operations. Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.

Bit wise operators in C language are & (bitwise AND), | (bitwise OR), ~ (bitwise OR), ^ (XOR), << (left shift) and >> (right shift).

f) CONDITIONAL OR TERNARY OPERATORS IN C

Conditional operators return one value if condition is true and returns another value is condition is false.

This operator is also called as ternary operator.

Syntax : (Condition? true_value: false_value);

Example : (A > 100 ? 0 : 1);

g) Increment/decrement Operators

Increment operators are used to increase the value of the variable by one and decrement operators are used to decrease the value of the variable by one in C programs.

Syntax:

Increment operator: ++var_name ;(or) var_name++;

Decrement operator: – -var_name; (or) var_name – -;

Example:

Increment operator : ++ i ; i ++ ;

Decrement operator : – – i ; i – – ;

C) Which are Unformatted I/O statement's:

Answer: Unformatted input/output functions

1. Unformatted console input/output functions are used to read a single input from the user at console and it also allows us to display the value in the output to the user at the console.
2. While calling any of the unformatted console input/output functions**," we do not have to use any format specifiers in them, to read or display a value".** Hence, these functions are named unformatted console I/O functions.
3. Some of the most important formatted console input/output functions are –
   a) getchar() : - It reads a character from the keyboard.
   The syntax of getchar() function is as follows
   Variablename=getchar();

For example,

```
Char a;
a = getchar();
```

b) gets() : It reads a string from the keyboard.

The syntax for gets() function is as follows − gets(variablename);

c) putchar(): It displays a character on the monitor.

The syntax for putchar() function is as follows – Putchar(variablename);

For example,

Putchar('a');

d) puts() : It displays a string on the monitor.

The syntax for puts() function is as follows − puts(variablename);

For example,

puts("tutorial");

e) getche() : Reads a single character from the user at the console, and echoing it.
f) getch() : Reads a single character from the user at the console, without echoing it.
g) putch() : Displays a single character value at the console.

d) Write a program using array addition of two matrices.

Answer:

```
#include <stdio.h>
int main()
{
   int rowCount, columnCount, i, j;
   int firstMatrix[10][10], secondMatrix[10][10], resultMatrix[10][10];

   printf("Number of rows of matrices to be added : ");
   scanf("%d", &rowCount);
   printf("Number of columns matrices to be added : ");
   scanf("%d", &columnCount);
   printf("Elements of first matrix : \n");
   for (i = 0; i < rowCount; i++){
      for (j = 0; j < columnCount; j++){
```

```
            scanf("%d", &firstMatrix[i][j]); }
    printf("Elements of second matrix : \n");  }
    for (i = 0; i < rowCount; i++) {
        for (j = 0; j < columnCount; j++) {
            scanf("%d", &secondMatrix[i][j]);}
                {
    printf("Sum of entered matrices : \n");
}
    for (i = 0; i < rowCount; i++)
    {
        for (j = 0; j < columnCount; j++)
        {
            resultMatrix[i][j] = firstMatrix[i][j] + secondMatrix[i][j];
            printf("%d\t",resultMatrix[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Sample Output:

Number of rows of matrices to be added : 2

Number of columns matrices to be added : 2

Elements of first matrix:

5

5

5

5

Elements of second matrix:

1

1

1

1

Difference of entered matrices:

6 6

6 6

e) Explain While loop.

Answer:

1. While loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given condition.
2. It can be viewed as a repeating if statement. The while loop is mostly used in the case where the number of iterations is not known in advance.
3. While-loop is entry control loop.
4. Syntax of while loop in C language

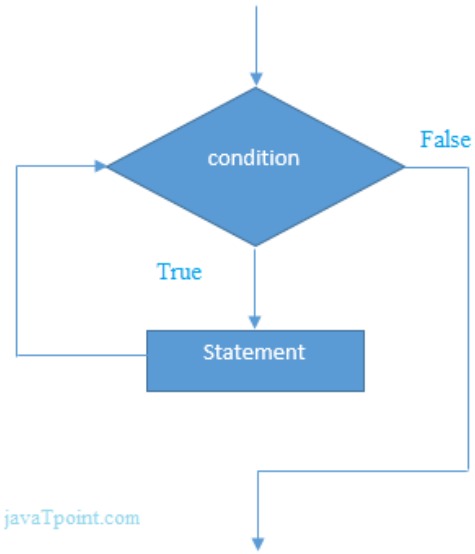
   The syntax of while loop in c language is given below:

   while(condition){

   //code to be executed

   }

5. Flowchart of while loop in C:



5. Example of the while loop in C language

   Let's see the simple program of while loop that prints table of 1.

   #include<stdio.h>

```
int main(){
int i=1;
while(i<=10){
printf("%d \n",i);
i++;
}
return 0;
}
```

Output

1

2

3

4

5

6

7

8

9

10

Q. 3 Attempt any three of the following (5 Marks each) 15

a) Explain else-if-ladder statements.

Answer:

6. In programming, if-else-if statement is also known as if-else-if ladder. It is used when there are more than two possible action based on different conditions.
7. Syntax

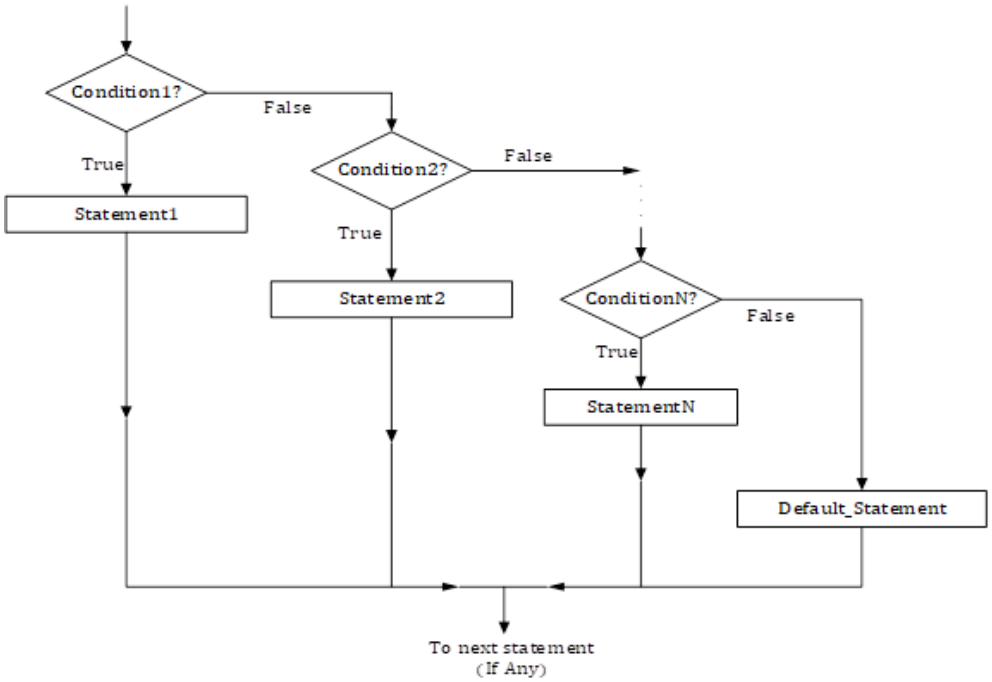
General syntax for if-else-if statement is:

```
if (Condition1)
{
  Statement1;
}
else if(Condition2)
{
 Statement2;
}
else if(ConditionN)
{
   StatementN;
}
else
{
  Default_Statement;
}
```

3. Working

In the above syntax of if-else-if, if the Condition1 is TRUE then the Statement1 will be executed and control goes to next statement in the program following if-else-if ladder.

4. If Condition1 is FALSE then Condition2 will be checked, if Condition2 is TRUE then Statement2 will be executed and control goes to next statement in the program following if-else-if ladder.
5. Similarly, if Condition2 is FALSE then next condition will be checked and the process continues.
6. If all the conditions in the if-else-if ladder are evaluated to FALSE, then Default_Statement will be executed.
7. The working of if-else-if ladder can be illustrated by following flowchart:
8. Flowchart



9. Programming Examples

Example 1: C program to find largest from three numbers given by user to explain working of if-else-if statement or ladder

```
#include<stdio.h>

int main()

  {          int a,b,c;

   printf("Enter three numbers: \n");

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

   if(a>b && a>c)

   {         printf("Largest = %d", a);

   }

   else if(b>a && b>c)

   {         printf("Largest = %d", b);

   }

   else

   {         printf("Largest = %d", c);

   }

   return(0);

   }
```

Output

Enter three numbers:

12

33

-17

Largest = 33

b) Write a program to find given no. is odd or even.

Answer:

```
#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: 9

9 is odd.

C) Explain Passing arrays to function.

Answer:

1. Passing arrays to functions in C programming: Like other values of variables, arrays can be passed to a function.. Both one-dimensional arrays and multidimensional arrays can be passed as function arguments.
2. Passing one-dimensional array to the function : While passing one-dimensional array to the function name of the array is passed as actual arguments and array variable with subscript is passed as formal arguments.
3. Ways of passing arrays as formal arguments.

   1: Passing formal parameter as pointer

   void func_name(int *name)

{ .....

}

2: Passing formal parameters as sized array

void func_name(int name[10])

{ .....

}

3: Passing formal parameters as unsized array

void func_name(int name[])

{ .....

}

4. Example: C program to pass an array to the function that contains marks obtained by the student. Then display the total marks obtained by the student

```
#include <stdio.h>

int total_marks(int a[]);

int main()

{

  int total = 0, marks[]={40,80,75,90,88};

  printf("Total marks = %d",total_marks(marks));

  return 0;

}

int total_marks(int a[])

{

  int sum=0,i;

  for (i=0;i<5;i++)

     sum=sum+a[i];

  return sum;

}
```

Output

Total marks = 373

5. **Explanation:**

Here, the total_marks( ) function is used to calculate total marks. The for loop is used to access the array elements.

6. Passing Multidimensional array to the function

Like simple arrays, multidimensional arrays can be passed to the function. Two-dimensional array is indicated by two sets of brackets with array variable. The size of second dimension must be specified.

7. Example: C program to display the 3*3 matrix of given elements.

```
#include <stdio.h>
void matrix( int m[][3] )
{
  int i,j;
  printf("Matrix is :\n");
  for ( i=0;i<3;i++ )
  {
    for ( j=0;j<3;j++)
    {
      printf("%d",m[i][j]);
     }
    printf("\n");
  }
}
int main()
{
  int m[3][3] = {1,2,3,4,5,6,7,8,9};
  matrix(m);
  return 0;
}
```

Output

```
Matrix is :
1 2 3
4 5 6
7 8 9
```

d) Explain structure of c program:

Answer:

1. Documentation section: The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.

2. Link section: The link section provides instructions to the compiler to link functions

from the system library such as using the #include directive.

3. Definition section: The definition section defines all symbolic constants such using

the #define directive.

4. Global declaration section: There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user-defined functions.

5. main () function section: Every C program must have one main function section. This section contains two parts; declaration part and executable part

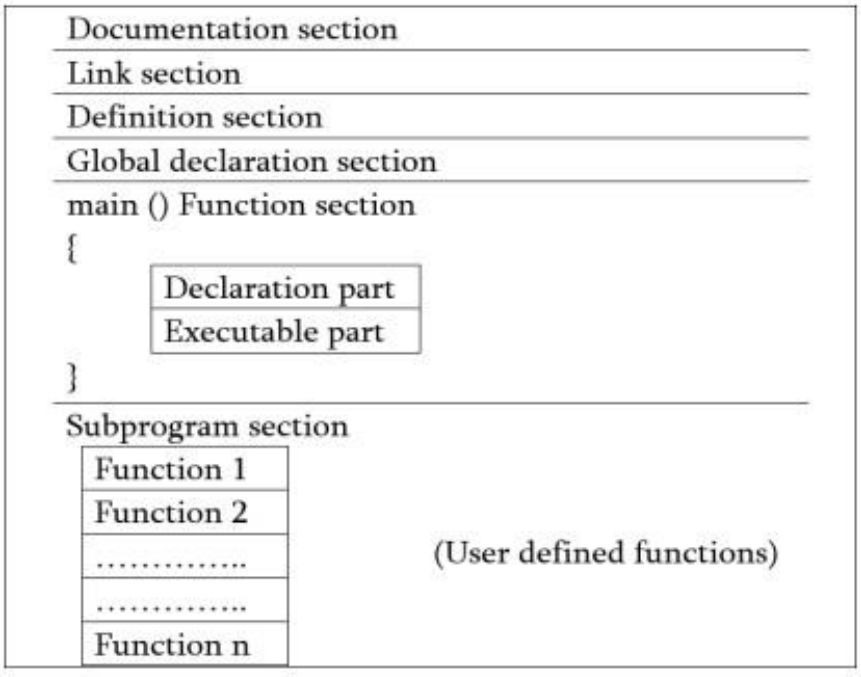
1. Declaration part: The declaration part declares all the variables used in the executable part.

2. Executable part: There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. The program execution begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a semicolon.

6. Subprogram section: If the program is a multi-function program then the subprogram section contains all the user-defined functions that are called in the main () function. User-defined functions are generally placed immediately after the main () function, although they may appear in any order.

7.

BASIC STRUCTURE OF C PROGRAMMING



e) What is difference between Compilers and Interpreters:

Answer:

Sr.no	Compiler	Interpreter
1.	The compiler scans the whole program in one go.	Translates the program one statement at a time.
2.	As it scans the code in one go, the errors (if any) are shown at the end together.	Considering it scans code one line at a time, errors are shown line by line.
3.	The main advantage of compilers is its execution time.	Due to interpreters being slow in executing the object code, it is preferred less.
4.	It converts the source code into object code.	It does not convert source code into object code instead it scans it line by line
5	It does not require source code for later execution.	It requires source code for later execution.
6	Execution of the program takes place only after the whole program is compiled.	Execution of the program happens after every line is checked or evaluated.
7	The machine code is stored in the disk storage.	Machine code is nowhere stored.
8	Compilers more often take a large amount of time for analyzing the source code.	In comparison, Interpreters take less time for analyzing the source code.
9.	It is more efficient.	It is less efficient.
10.	CPU utilization is more.	CPU utilization is less.
Eg.	C, C++, C#, etc are programming languages that are compiler-based.	Python, Ruby, Perl, SNOBOL, MATLAB, etc are programming languages that are interpreter-based.

Q. 4 Attempt any three of the following (5 Marks each) 15

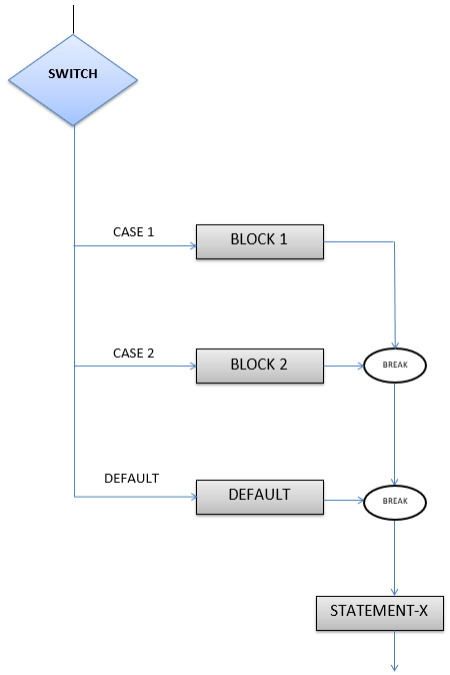
a) Explain witch statements.

Answer:

1. Switch statement in C tests the value of a variable and compares it with multiple cases.
2. Once the case match is found, a block of statements associated with that particular case is executed.
3. Each case in a block of a switch has a different name/number which is referred to as an identifier.
4. The value provided by the user is compared with all the cases inside the switch block until the match is found.
5. If a case match is NOT found, then the default statement is executed, and the control goes out of the switch block.
6. Syntax:
   Switch Case Syntax
   A general syntax of how switch-case is implemented in a 'C' program is as follows:

```
switch( expression )
{
    case value-1:
                Block-1;
                Break;
    case value-2:
                Block-2;
                Break;
    case value-n:
                Block-n;
                Break;
    default:
                Block-1;
                Break;
}
Statement-x;
```

7. Flow chart:



8. Program:

```
#include <stdio.h>
   int main() {
      int num = 8;
      switch (num) {
         case 7:
            printf("Value is 7");
            break;
         case 8:
            printf("Value is 8");
            break;
         case 9:
            printf("Value is 9");
            break;
         default:
            printf("Out of range");
            break;
      }
      return 0;
   }
```

Output:

Value is 8

9.**Explaination of program:** In the given program we have explain initialized a variable num with value A switch construct is used to compare the value stored in variable num and execute the block of statements associated with the matched case. In this program, since the value stored in variable num is eight, a switch will execute the case whose case-label is 8. After executing the case, the control will fall out of the switch and program will be terminated with the successful result by printing the value on the output screen.

b) Write a program using array to find Larges Element in an array.

Answer:

```
 include <stdio.h>
int main()
{
 int n;
 double arr[100];
  printf("Enter the number of elements (1 to 100): ");
  scanf("%d", &n);
  for (int i = 0; i < n; ++i) {
 printf("Enter number%d: ", i + 1);
 scanf("%lf", &arr[i]);
            }
  // storing the largest number to arr[0]
  for (int i = 1; i < n; ++i) {
   if (arr[0] < arr[i]) {
              arr[0] = arr[i];
                        }
                                 }
            printf("Largest element = %.2lf", arr[0]);
  return 0;
}
```

Output

Enter the number of elements (1 to 100): 5

Enter number1: 34

Enter number2: 2

Enter number3: -35

Enter number4: 38

Enter number5: 24

Largest element = 38

c) Explain formatted I/O Statements.

Answer:

1. **Definition**: It is called as formatted I/O statements because it uses the format specifiers in these functions.
2. We use the formatted input and output functions in the C language for taking single or multiple inputs from the programmer/user at the console.
3. These functions also allow a programmer to display single or multiple values, in the form of output, to the users present in the console.
4. Functions Used and their Description of the Function
5. printf() We use this function for displaying a single or multiple values in the form of output for the user end at the console.
6. scanf() We use this function for reading a single or multiple values in the form of input from the user end at the console.
7. sprintf() We use this function for reading the values that are stored in various variables, and for storing these values into the array of characters.
   Syntax: sprint(array name,"format specifier", variable name)
8. sscanf() We use this function for reading the characters available in the string, and then storing them into the available variables.
9. Syntax: sprint(array name,"format specifier", &variable name)

10. scanf()

    We use the scanf() function for getting the formatted inputs or standard inputs so that the printf() function can provide the program with numerous options of conversion.

    Syntax for scanf()

    scanf (format_specifier, &data_a, &data_b,……); // Here, & refers to the address operator

11. The purpose of the scanf() function is to read the characters that we get from the standard input, convert them according to the string of format specification, and then store the available inputs in the memory slots that the other arguments represent.
12. Example of scanf()

scanf("%d %c", &info_a,&info_b);

When we consider the string data names in a program, the '&' character does not prefix the data name.

13. printf() : We use the printf() function for generating the formatted outputs or standard outputs in accordance with a format specification. The output data and the format specification string act as the parameters of the function printf().

Syntax for printf()

printf (format_specifiers, info_a, info_b,……..);

Example of printf()

printf("%d %c", info_a, info_b);

d) Explain do-while loop.

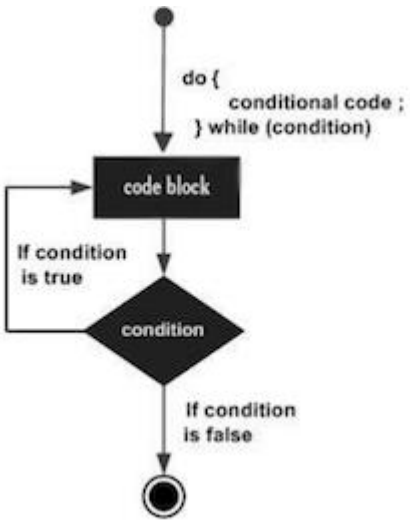
Answer:

1. The do while loop is a post tested loop. Using the do-while loop, we can repeat the execution of several parts of the statements.
2. The do-while loop is mainly used in the case where we need to execute the loop at least once.
3. The do-while loop is mostly used in menu-driven programs where the termination condition depends upon the end user.
4. do while loop syntax

The syntax of the C language do-while loop is given below:

```
do{
//code to be executed
}while(condition);
```

5. Flow chart



6.
```
#include<stdio.h>
#include<conio.h>
int main()
{
int num=1;          //initializing the variable
do          //do-while loop
{
            printf("%d\n",2*num);
            num++;              //incrementing operation
}while(num<=10);
return 0;
}
```

Output:

```
2
4
6
8
10
12
14
16
18
20
```

e) Write a sum of digits' program in C.

Answer:

```
 #include<stdio.h>
 int main()
{
int n,sum=0,m;
printf("Enter a number:");
scanf("%d",&n);
while(n>0)
{
m=n%10;
sum=sum+m;
n=n/10;
}
printf("Sum is=%d",sum);
return 0;
}
```

Output:

Enter a number:654

Sum is=15

Q .5 Short notes on any three of the following (5 Marks each) 15

a) Variables

Answer:

Variables in C:

- A variable is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times.
- It is a way to represent memory location through symbol so that it can be easily identified.

Syntax:

type variable_list;

1. The example of declaring the variable is given below:

```
int a;
float b;
char c;
```

2. Here, a, b, c are variables. The int, float, char are the data types.

3. We can also provide values while declaring the variables as given below:

```
int a=10,b=20;//declaring 2 variable of integer type
float f=20.8;
char c='A';
```

4. Rules for defining variables
   - A variable can have alphabets, digits, and underscore.
   - A variable name can start with the alphabet, and underscore only. It can't start with a digit.
   - No whitespace is allowed within the variable name.
   - A variable name must not be any reserved word or keyword, e.g. int, float, etc.

5. Valid variable names:

```
int a;
int _ab;
int a30;
```

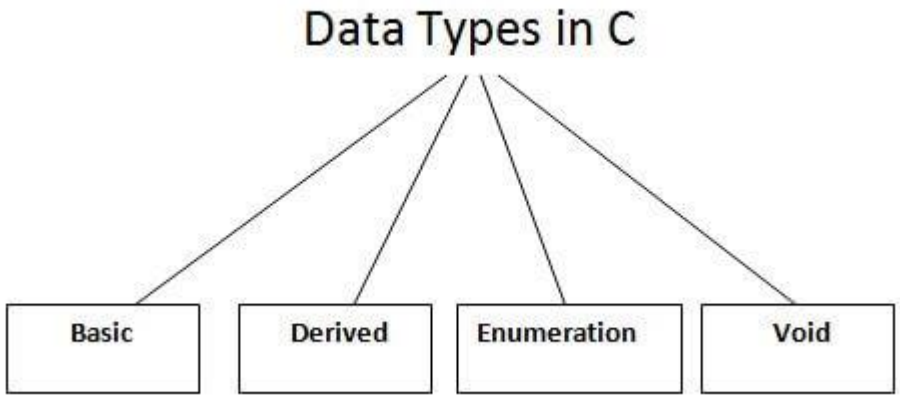
6. Invalid variable names:

```
int 2;
int a b;
int long;
```

b) Data types :

Answer:

1. Each variable in C has an associated data type.
2. Each data type requires different amounts of memory and has some specific operations which can be performed over it.
3. It specifies the type of data that the variable can store like integer, character, floating, double, etc.
4. The data type is a collection of data with values having fixed values, meaning as well as its characteristics.
5. There are four types of datatype:



6. Basic Data Type includes int, char, float, double

int

- Integers are whole numbers that can have both zero, positive and negative values but no decimal values. For example, 0, -5, 10
- We can use int for declaring an integer variable.
  int id;
  Here, id is a variable of type integer.
- You can declare multiple variables at once in C programming. For example,
  int id, age;
- The size of int is usually 4 bytes (32 bits). And, it can take 232 distinct states from -2147483648 to 2147483647.

float and double

- float and double are used to hold real numbers.
  float salary;
  double price;
- floating-point numbers can also be represented in exponential. For example,
  float normalizationFactor = 22.442e2;
- difference between float and double
- The size of float (single precision float data type) is 4 bytes. And the size of double (double precision float data type) is 8 bytes.

char

- Keyword char is used for declaring character type variables. For example,
- char test = 'h';
- The size of the character variable is 1 byte.

7. Derived Data Type includes array, pointer, structure, union

Derived Data Types

- Data types that are derived from fundamental data types are derived types. For example: arrays, pointers, function types, structures, etc.

8. Enumeration Data Type includes enum
9. Void Data Type include void

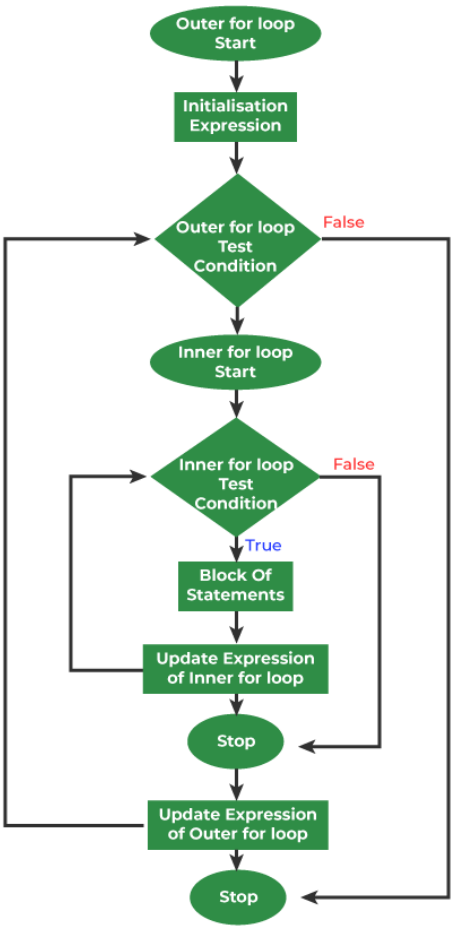
void

- void is an incomplete type. It means "nothing" or "no type". You can think of void as absent.
- For example, if a function is not returning anything, its return type should be void.
- cannot create variables of void type.

c) Nested for loop.

Answer:

1. Nested for Loop:
   Nested for loop refers to any type of loop that is defined inside a 'for' loop.
2. Below is the equivalent flow diagram for nested 'for' loops:



3. Syntax:

```
for ( initialization; condition; increment ) {
  for ( initialization; condition; increment ) {
  // statement of inside loop
            }
 // statement of outer loop
            }
```

4. Example:

```
//The following program uses a nested for loop to find the prime numbers from 2 to 20 −
#include <stdio.h>
int main () {
  /* local variable definition */
  int i, j;

  for(i = 2; i<100; i++) {
  for(j = 2; j <= (i/j); j++)

   if(!(i%j)) break; // if factor found, not prime

       if(j > (i/j)) printf("%d is prime\n", i);
            }
              return 0;
           }
```

Output:

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
```

d) go to statements.

Answer:

1. The goto statement is a jump statement which is sometimes also referred to as unconditional jump statement.
2. The goto statement can be used to jump from anywhere to anywhere within a function.

3. Syntax:

```
Syntax1      |  Syntax2
-------------------------
goto label;  |   label:
    .        |     .
    .        |     .
    .        |     .
         label:     |  goto label;
```

4. In the above syntax, the first line tells the compiler to go to or jump to the statement marked as a label.
5. Here label is a user-defined identifier which indicates the target statement.
6. The statement immediately followed after ‘label:’ is the destination statement.
7. The ‘label:’ can also appear before the ‘goto label;’ statement in the above syntax.

8. CODE:

```
// even or not using goto statement
#include <stdio.h>
// function to check even or not
void checkEvenOrNot(int num)
{
    if (num % 2 == 0)
            // jump to even
            goto even;
    else
            // jump to odd
            goto odd;
even:
    printf("%d is even", num);
    // return if even
    return;
odd:
    printf("%d is odd", num);
}
int main() {
    int num = 26;
    checkEvenOrNot(num);
    return 0;
}
```

OUTPUT:

26 is even

e) High level languages.
Answer:

High-Level Languages:

1. The symbolic languages greatly improved programming efficiency they still required programmers to concentrate on the hardware that they were using working with symbolic languages was also very tedious because each machine instruction had to be individually coded.
2. The desire to improve programmer efficiency and to change the focus from the computer to the problems being solved led to the development of high-level languages.
3. High-level languages are portable to many different computer allowing the programmer to concentrate on the application problem at hand rather than the intricacies of the computer.

C	A systems implementation Language
C++	C with object oriented enhancements
JAVA	Object oriented language for internet and general applications using basic C syntax

4. Advantages:

a) Easy to write and understand
b) Easy to isolate an error
c) Machine independent language
d) Easy to maintain
e) Better readability
f) Low Development cost
g) Easier to document
h) Portable

5. Disadvantages:

a) Needs translator
b) Requires high execution time
c) Poor control on hardware
d) Less efficient

6. Example: C language

```
#include<stdio.h>
void main()
{
int a,b,c;
 scanf("%d%d%",&a,&b);

c=a+b;
printf("%d",c);
}
```