**WHAT IS THE DIFFERENCE BETWEEN C AND C++?**

* C is structure/procedure oriented programming language whereas C++ is object oriented programming language.
* C language program design is top down approach whereas C++ is using bottom up approach.
* Polymorphism, virtual function, inheritance, Operator overloading, namespace concepts are not available in C programming language. Whereas C++ language supports all these concepts and features.
* C language gives importance to functions rather than data. Whereas C++ gives importance to data rather than functions.
* So, data and function mapping is difficult in C. But, data and function mapping is simple in C++ that can be done using objects.
* C language does not support user define data types. Whereas C++ supports user define data types.
* Exception handling is not present in C programming language. Whereas exception handling is present in C++ language.

**What is Macro? Why do we use macro?**

* Macro is a name which is given to a value or to a piece of code/block in a program. Instead of using the value, we can use macro which will replace the value in a program.
* The reason for using macro is, consider below example.
* You are using a person’s age as 50 in many places in your program. If you use direct value in all the places in your program, it is very difficult to change the age value in future if you want to change it to 60. If you use macro in your program, it is very simple to replace the value and we can change in only one place which will change the value in all places in your program.

#### Syntax: #define <MACRO\_NAME> VALUE Example: Original declaration – #define AGE 50 ****What is the use of sizeof() function in C?****

sizeof() function is used to find the memory space allocated for each data type in C.

#include <stdio.h>

int main() {

int a = 16;

printf("Size of variable a : %d\n",sizeof(a));

printf("Size of int data type : %d\n",sizeof(int));

return 0;

}

**Output : Size of variable a : 4**

**Size of int data type : 4**

**What is token in C?**

* C tokens are the basic buildings blocks in C language which are constructed together to write a C program.
* Each and every smallest individual unit in a C program are known as C tokens.

#### ****What is keyword in C?****

* Keywords are pre-defined words in a C compiler. Each keyword is meant to perform a specific function in a C program.
* Since keywords are referred names for compiler, they can’t be used as variable name.

#### ****What is “##” operator in C?****

## is a pre-processor macro in C. It is used to concatenate 2 tokens into one token.

#include<stdio.h>

#define concatination(a,b) a ## b

**int** main ()

{

**int** ab = 1000;

**printf**("The concatenated value is:%d \n",concatination(a,b));

**return** 0;

}

**Why is C called a mid-level programming language?**

## **C as a mid-level programming language :** C is considered as a middle-level language because it **supports the feature of both low-level and high-level languages**. C language program is converted into assembly code, it supports pointer arithmetic (low-level), but it is machine independent (a feature of high-level).

A **Low-level language** is specific to one machine, i.e., machine dependent. It is machine dependent, fast to run. But it is not easy to understand.

A **High-Level language** is not specific to one machine, i.e., machine independent. It is easy to understand.

**Why is C called a a procedural language?**

## **C as a procedural language :** A procedure is known as a function, method, routine, subroutine, etc. A procedural language **specifies a series of steps for the program to solve the problem**.A procedural language breaks the program into functions, data structures, etc. C is a procedural language. In C, variables and function prototypes must be declared before being used.

### **What is the use of printf() and scanf() functions?**

**printf():** The printf() function is used to print the integer, character, float and string values on to the screen.

Following are the format specifier:

* **%d**: It is a format specifier used to print an integer value.
* **%s**: It is a format specifier used to print a string.
* **%c**: It is a format specifier used to display a character value.
* **%f**: It is a format specifier used to display a floating point value.

**scanf()**: The scanf() function is used to take input from the user.

### **What is the difference between the local variable and global variable in C?**

Following are the differences between a local variable and global variable:

|  |  |  |
| --- | --- | --- |
| **Basis for comparison** | **Local variable** | **Global variable** |
| Declaration | A variable which is declared inside function or block is known as a local variable. | A variable which is declared outside function or block is known as a global variable. |
| Scope | The scope of a variable is available within a function in which they are declared. | The scope of a variable is available throughout the program. |
| Access | Variables can be accessed only by those statements inside a function in which they are declared. | Any statement in the entire program can access variables. |
| Life | Life of a variable is created when the function block is entered and destroyed on its exit. | Life of a variable exists until the program is executing. |
| Storage | Variables are stored in a stack unless specified. | The compiler decides the storage location of a variable. |

### **What is the use of a static variable in C?**

Following are the uses of a static variable:

* A variable which is declared as static is known as a static variable. The static variable retains its value between multiple function calls.
* Static variables are used because the scope of the static variable is available in the entire program. So, we can access a static variable anywhere in the program.
* The static variable is initially initialized to zero. If we update the value of a variable, then the updated value is assigned.
* The static variable is used as a common value which is shared by all the methods.
* The static variable is initialized only once in the memory heap to reduce the memory usage.

### **What is the use of the function in C?**

**Uses of C function are:**

* C functions are used to avoid the rewriting the same code again and again in our program.
* C functions can be called any number of times from any place of our program.
* When a program is divided into functions, then any part of our program can easily be tracked.
* C functions provide the reusability concept, i.e., it breaks the big task into smaller tasks so that it makes the C program more understandable.

### **What is the difference between call by value and call by reference in C?**

**Following are the differences between a call by value and call by reference are:**

|  |  |  |
| --- | --- | --- |
|  | **Call by value** | **Call by reference** |
| Description | When a copy of the value is passed to the function, then the original value is not modified. | When a copy of the value is passed to the function, then the original value is modified. |
| Memory location | Actual arguments and formal arguments are created in separate memory locations. | Actual arguments and formal arguments are created in the same memory location. |
| Safety | In this case, actual arguments remain safe as they cannot be modified. | In this case, actual arguments are not reliable, as they are modified. |
| Arguments | The copies of the actual arguments are passed to the formal arguments. | The addresses of actual arguments are passed to their respective formal arguments. |

### **What is recursion in C?**

When a function calls itself, and this process is known as recursion. The function that calls itself is known as a recursive function. Recursive function comes in two phases:

1. Winding phase
2. Unwinding phase

**Winding phase**: When the recursive function calls itself, and this phase ends when the condition is reached.

**Unwinding phase**: Unwinding phase starts when the condition is reached, and the control returns to the original call.

**Example of recursion**

#include <stdio.h>

**int** calculate\_fact(**int**);

**int** main()

{

**int** n=5,f;

 f=calculate\_fact(n); // calling a function

 printf("factorial of a number is %d",f);

**return** 0;

}

**int** calculate\_fact(**int** a)

{

**if**(a==1)

  {

**return** 1;

  }

**else**

**return** a\*calculate\_fact(a-1); //calling a function recursively.

   }

### **What is an array in C?**

**Array :** An Array is a group of similar types of elements. It has a contiguous memory location. It makes the code optimized, easy to traverse and easy to sort. The size and type of arrays cannot be changed after its declaration.

**Arrays are of two types:**

* **One-dimensional array**: One-dimensional array is an array that stores the elements one after the another. **Syntax:** data\_type array\_name[size];
* **Multidimensional array**: Multidimensional array is an array that contains more than one array. **Syntax:** data\_type array\_name[size];

### **What is a pointer in C?**

A pointer is a variable that refers to the address of a value. It makes the code optimized and makes the performance fast. Whenever a variable is declared inside a program, then the system allocates some memory to a variable. The memory contains some address number. The variables that hold this address number is known as the pointer variable.

**For example:**

Data\_type \*p;

The above syntax tells that p is a pointer variable that holds the address number of a given data type value.

**Example of pointer**

#include <stdio.h>

**int** main()

{

**int** \*p; //pointer of type integer.

**int** a=5;

   p=&a;

   printf("Address value of 'a' variable is %u",p);

**return** 0;

}

## **Advantage of pointer**

1) Pointer **reduces the code** and **improves the performance**, it is used to retrieving strings, trees, etc. and used with arrays, structures, and functions.

2) We can **return multiple values from a function** using the pointer.

3) It makes you able to **access any memory location** in the computer's memory.

## **What is a NULL pointer in C?**

## **NULL Pointer :**A pointer that is not assigned any value but NULL is known as the NULL pointer. If you don't have any address to be specified in the pointer at the time of declaration, you can assign NULL value. It will provide a better approach.

int \*p=NULL;

In the most libraries, the value of the pointer is 0 (zero).

### **What is a far pointer in C?**

A pointer which can access all the 16 segments (whole residence memory) of RAM is known as far pointer. A far pointer is a 32-bit pointer that obtains information outside the memory in a given section.

### **What is dangling pointer in C?**

* If a pointer is pointing any memory location, but meanwhile another pointer deletes the memory occupied by the first pointer while the first pointer still points to that memory location, the first pointer will be known as a dangling pointer. This problem is known as a dangling pointer problem.
* Dangling pointer arises when an object is deleted without modifying the value of the pointer. The pointer points to the deallocated memory.

**Let's see this through an example.**

1. #include<stdio.h>
2. **void** main()
3. {
4. **int** \*ptr = malloc(constant value); //allocating a memory space.
5. free(ptr); //ptr becomes a dangling pointer.
6. }

In the above example, initially memory is allocated to the pointer variable ptr, and then the memory is deallocated from the pointer variable. Now, pointer variable, i.e., ptr becomes a dangling pointer.

### **What is pointer to pointer in C?**

In case of a pointer to pointer concept, one pointer refers to the address of another pointer. The pointer to pointer is a chain of pointers. Generally, the pointer contains the address of a variable. The pointer to pointer contains the address of a first pointer. Let's understand this concept through an example:

#include <stdio.h>

**int** main()

{

**int** a=10;

**int** \*ptr,\*\*pptr; // \*ptr is a pointer and \*\*pptr is a double pointer.

    ptr=&a;

    pptr=&ptr;

    printf("value of a is:%d",a);

    printf("\n");

    printf("value of \*ptr is : %d",\*ptr);

    printf("\n");

    printf("value of \*\*pptr is : %d",\*\*pptr);

**return** 0;

}

In the above example, pptr is a double pointer pointing to the address of the ptr variable and ptr points to the address of 'a' variable.

### **What is static memory allocation?**

* In case of static memory allocation, memory is allocated at compile time, and memory can't be increased while executing the program. It is used in the array.
* The lifetime of a variable in static memory is the lifetime of a program.
* The static memory is allocated using static keyword.
* The static memory is implemented using stacks or heap.
* The pointer is required to access the variable present in the static memory.
* The static memory is faster than dynamic memory.
* In static memory, more memory space is required to store the variable.

1. For example:
2. **int** a[10];

### **What is dynamic memory allocation?**

* In case of dynamic memory allocation, memory is allocated at runtime and memory can be increased while executing the program. It is used in the linked list.
* The malloc() or calloc() function is required to allocate the memory at the runtime.
* An allocation or deallocation of memory is done at the execution time of a program.
* No dynamic pointers are required to access the memory.
* The dynamic memory is implemented using data segments.
* Less memory space is required to store the variable.

For example

**int** \*p= malloc(**sizeof**(**int**)\*10);

**What is the difference between malloc() and calloc()?**

|  |  |  |
| --- | --- | --- |
|  | **calloc()** | **malloc()** |
| Description | The malloc() function allocates a single block of requested memory. | The calloc() function allocates multiple blocks of requested memory. |
| Initialization | It initializes the content of the memory to zero. | It does not initialize the content of memory, so it carries the garbage value. |
| Number of arguments | It consists of two arguments. | It consists of only one argument. |
| Return value | It returns a pointer pointing to the allocated memory. | It returns a pointer pointing to the allocated memory. |

### **What is the structure?**

* The structure is a user-defined data type that allows storing multiple types of data in a single unit. It occupies the sum of the memory of all members.
* The structure members can be accessed only through structure variables.
* Structure variables accessing the same structure but the memory allocated for each variable will be different.

**Syntax of structure**

**struct** structure\_name

{

  Member\_variable1;

 Member\_variable2

.

.

}[structure variables];

### **What is a union?**

* The union is a user-defined data type that allows storing multiple types of data in a single unit. However, it doesn't occupy the sum of the memory of all members. It holds the memory of the largest member only.
* In union, we can access only one variable at a time as it allocates one common space for all the members of a union.

**Syntax of union**

**union** union\_name

{

Member\_variable1;

Member\_variable2;

.

.

Member\_variable n;

}[**union** variables];

**What is an auto keyword in C?**In C, every local variable of a function is known as an automatic (auto) variable. Variables which are declared inside the function block are known as a local variable. The local variables are also known as an auto variable. It is optional to use an auto keyword before the data type of a variable. If no value is stored in the local variable, then it consists of a garbage value.

### **What is the purpose of sprintf() function?**

The sprintf() stands for "string print." The sprintf() function does not print the output on the console screen. It transfers the data to the buffer. It returns the total number of characters present in the string.

**Syntax**

1. **int** sprintf ( **char** \* str, **const** **char** \* format, ... );

**Let's see a simple example**

 #include<stdio.h>

**int** main()

{

**char** a[20];

**int** n=sprintf(a,"javaToint");

 printf("value of n is %d",n);

**return** 0;}

**Output:**

value of n is 9

**What is command line argument?**

The argument passed to the main() function while executing the program is known as command line argument. For example:

main(**int** count, **char** \*args[]){

//code to  be executed

}

### **What is the difference between getch() and getche()?**

The **getch()** function reads a single character from the keyboard. It doesn't use any buffer, so entered data will not be displayed on the output screen.

The **getche()** function reads a single character from the keyword, but data is displayed on the output screen. Press Alt+f5 to see the entered character.

**Let's see a simple example**

#include<stdio.h>

#include<conio.h>

**int** main()

{

**char** ch;

 printf("Enter a character ");

 ch=getch(); // taking an user input without printing the value.

 printf("\nvalue of ch is %c",ch);

 printf("\nEnter a character again ");

 ch=getche(); // taking an user input and then displaying it on the screen.

  printf("\nvalue of ch is %c",ch);

**return** 0;

}

### **What is the newline escape sequence?**

The new line escape sequence is represented by "\n". It inserts a new line on the output screen.

### **What is typecasting?**

The typecasting is a process of converting one data type into another is known as typecasting. If we want to store the floating type value to an int type, then we will convert the data type into another data type explicitly.

**Syntax**

(type\_name) expression;

### **Write a program to print "hello world" without using a semicolon?**

#include<stdio.h>

**void** main(){

**if**(printf("hello world")){} // It prints the ?hello world? on the screen.

}

**What is the use of a ‘\0’ character?**

It is referred to as a terminating null character, and is used primarily to show the end of a string value.

What is the difference between the = symbol and == symbol?

The = symbol is often used in mathematical operations. It is used to assign a value to a given variable. On the other hand, the == symbol, also known as “equal to” or “equivalent to”, is a relational operator that is used to compare two values.

**What is the modulus operator?**

The modulus operator outputs the remainder of a division. It makes use of the percentage (%) symbol. For example: 10 % 3 = 1, meaning when you divide 10 by 3, the remainder is 1.

**What are header files and what are its uses in C programming?**

Header files are also known as library files. They contain two essential things: the definitions and prototypes of functions being used in a program. Simply put, commands that you use in C programming are actually functions that are defined from within each header files. Each header file contains a set of functions. For example: stdio.h is a header file that contains definition and prototypes of commands like printf and scanf.

**What is syntax error?**

Syntax errors are associated with mistakes in the use of a programming language. It maybe a command that was misspelled or a command that must was entered in lowercase mode but was instead entered with an upper case character. A misplaced symbol, or lack of symbol, somewhere within a line of code can also lead to syntax error.

**What is the advantage of an array over individual variables?**

When storing multiple related data, it is a good idea to use arrays. This is because arrays are named using only 1 word followed by an element number. For example: to store the 10 test results of 1 student, one can use 10 different variable names (grade1, grade2, grade3… grade10). With arrays, only 1 name is used, the rest are accessible through the index name (grade[0], grade[1], grade[2]… grade[9]).

**What is debugging?**

Debugging is the process of identifying errors within a program. During program compilation, errors that are found will stop the program from executing completely. At this state, the programmer would look into the possible portions where the error occurred. Debugging ensures the removal of errors, and plays an important role in ensuring that the expected program output is met.

**What does the format %10.2 mean when included in a printf statement?**

This format is used for two things: to set the number of spaces allotted for the output number and to set the number of decimal places. The number before the decimal point is for the allotted space, in this case it would allot 10 spaces for the output number. If the number of space occupied by the output number is less than 10, addition space characters will be inserted before the actual output number. The number after the decimal point sets the number of decimal places, in this case, it’s 2 decimal spaces.

**What are logical errors and how does it differ from syntax errors?**

Program that contains logical errors tend to pass the compilation process, but the resulting output may not be the expected one. This happens when a wrong formula was inserted into the code, or a wrong sequence of commands was performed. Syntax errors, on the other hand, deal with incorrect commands that are misspelled or not recognized by the compiler.

**What are preprocessor directives?**

Preprocessor directives are placed at the beginning of every C program. This is where library files are specified, which would depend on what functions are to be used in the program. Another use of preprocessor directives is the declaration of constants.Preprocessor directives begin with the # symbol.

**What are reserved words?**

Reserved words are words that are part of the standard C language library. This means that reserved words have special meaning and therefore cannot be used for purposes other than what it is originally intended for. Examples of reserved words are int, void, and return.

**What is the difference between the expression “++a”  and “a++”?**

In the first expression, the increment would happen first on variable a, and the resulting value will be the one to be used. This is also known as a prefix increment. In the second expression, the current value of variable a would the one to be used in an operation, before the value of a itself is incremented. This is also known as postfix increment.

**What would happen to X in this expression: X += 15;  (assuming the value of X is 5)**

X +=15 is a short method of writing X = X + 15, so if the initial value of X is 5, then 5 + 15 = 20.

**What are actual arguments?**

When you create and use functions that need to perform an action on some given values, you need to pass these given values to that function. The values that are being passed into the called function are referred to as actual arguments.

**What is a newline escape sequence?**

A newline escape sequence is represented by the \n character. This is used to insert a new line when displaying data in the output screen. More spaces can be added by inserting more \n characters. For example, \n\n would insert two spaces. A newline escape sequence can be placed before the actual output expression or after.

**What are run-time errors?**

These are errors that occur while the program is being executed. One common instance wherein run-time errors can happen is when you are trying to divide a number by zero. When run-time errors occur, program execution will pause, showing which program line caused the error.

**What is the difference between functions abs() and fabs()?**

These 2 functions basically perform the same action, which is to get the absolute value of the given value. Abs() is used for integer values, while fabs() is used for floating type numbers. Also, the prototype for abs() is under <stdlib.h>, while fabs() is under <math.h>.

**When is a “switch” statement preferable over an “if” statement?**

The switch statement is best used when dealing with selections based on a single variable or expression. However, switch statements can only evaluate integer and character data types.

**What are enumerated types?**

Enumerated types allow the programmer to use more meaningful words as values to a variable. Each item in the enumerated type variable is actually associated with a numeric code. For example, one can create an enumerated type variable named DAYS whose values are Monday, Tuesday… Sunday.

**What does the characters “r” and “w” mean when writing programs that will make use of files?**

“r” means “read” and will open a file as input wherein data is to be retrieved. “w” means “write”, and will open a file for output. Previous data that was stored on that file will be erased.

**What is the difference between text files and binary files?**

Text files contain data that can easily be understood by humans. It includes letters, numbers and other characters. On the other hand, binary files contain 1s and 0s that only computers can interpret.

**is it possible to create your own header files?**

Yes, it is possible to create a customized header file. Just include in it the function prototypes that you want to use in your program, and use the #include directive followed by the name of your header file.

**What is gets() function?**

The gets() function allows a full line data entry from the user. When the user presses the enter key to end the input, the entire line of characters is stored to a string variable. Note that the enter key is not included in the variable, but instead a null terminator \0 is placed after the last character.

**The % symbol has a special use in a printf statement. How would you place this character as part of the output on the screen?**

You can do this by using %% in the printf statement. For example, you can write printf(“10%%”) to have the output appear as 10% on the screen.

**What is the use of a semicolon (;) at the end of every program statement?**

It has to do with the parsing process and compilation of the code. A semicolon acts as a delimiter, so that the compiler knows where each statement ends, and can proceed to divide the statement into smaller elements for syntax checking.

**What are local static variables? What is their use?**  
**Ans:**A local static variable is a variable whose lifetime doesn’t end with a function call where it is declared. It extends for the lifetime of complete program. All calls to the function share the same copy of local static variables. Static variables can be used to count the number of times a function is called. Also, static variables get the default value as 0. For example, the following program prints “0 1”.

|  |
| --- |
| #include <stdio.h>  **void** fun()  {      // static variables get the default value as 0.  **static** **int** x;  **printf**("%d ", x);      x = x + 1;  }    **int** main()  {      fun();      fun();  **return** 0;  }  // Output: 0 1 |

**What is ternary operater ?**

The ternary operator is used to execute code based on the result of a binary condition.

It takes in a binary condition as input, which makes it similar to an 'if-else' control flow block. It also, however, returns a value, behaving similar to a function.

##### **Syntax :**result = binaryCondition ? valueReturnedIfTrue : valueReturnedIfFalse;

#### ****What is const pointer in C?****

Const pointer is a pointer that can’t change the address of the variable that is pointing to.  
Once const pointer is made to point one variable, we can’t change this pointer to point to any other variable. This pointer is called const pointer.

**What is a flowchart?**

A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution to a givenproblem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. Flowcharts are used in designing and documenting complex processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help the people to understand a process, and perhaps also find flaws, bottlenecks, and other less-obvious features within it. There are many different types of flowcharts, and each type has its own repertoire of boxes and notational conventions. The two most common types of boxes in a flowchart are:

a processing step, usually called activity, and denoted as a rectangular box  a decision, usually denoted as a diamond.

**What is the use of typedef in c ?**

typedef is a keyword in the C and C++ programming languages. The purpose of typedef is to assign alternative names to existing types, most often those whose standard declaration is cumbersome, potentially confusing, or likely to vary from one implementation to another.

**Explain in detail High level, Machine language and Assembly language.**

Low Level Languages Low level computer languages are machine codes or close to it. Computer cannot understand instructions given in high level languages or in English. It can only understand and execute instructions given in the form of machine language i.e. language of 0 and 1. There are two types of low level languages:

Machine Language.

Assembly Language

**Machine Language:** It is the lowest and most elementary level of Programming language and was the first type of programming language to be Developed. Machine Language is basically the only language which computer Can understand. In fact, a manufacturer designs a computer to obey just one Language, its machine code, which is represented inside the computer by a String of binary digits (bits) 0 and 1. The symbol 0 stands for the absence of Electric pulse and 1 for the presence of an electric pulse . Since a computer is Capable of recognizing electric signals, therefore, it understand machine Language.

**Advantages of Machine Language** i) It makes fast and efficient use of the computer.

ii) It requires no translator to translate the code i.e.Directly understood by the computer

Disadvantages of Machine Language:

i) All operation codes have to be remembered

ii) All memory addresses have to be remembered.

iii) It is hard to amend or find errors in a program written

In the machine language

iv) These languages are machine dependent i.e. a particular

Machine language can be used on only one type of computer

**Assembly Language** :It was developed to overcome some of the many inconveniences of machine language. This is another low level but a very important language in which operation codes and operands are given in the form of alphanumeric symbols instead of 0‟s and l‟s. These alphanumeric symbols will be known as mnemonic codes and can have maximum up to 5 letter combination e.g. ADD for addition, SUB for subtraction, START,LABEL etc. Because of this feature it is also known as „Symbolic Programming Language‟. This language is also very difficult and needs a lot of practice to master it because very small English support is given to this language. The language mainly helps in compiler orientations. The instructions of the Assembly language will also be converted to machine codes by language translator to be executed by the computer.

**Advantages of Assembly Language** i) It is easier to understand and use as compared to machine language.

ii)It is easy to locate and correct errors.

iii) It is modified easily

Disadvantages of Assembly Language i) Like machine language it is also machine dependent.

ii) Since it is machine dependent therefore programmer Should have the knowledge of the hardware also.

**High Level Languages** :High level computer languages give formats close to English language and the purpose of developing high level languages is to enable people to write programs easily and in their own native language environment (English). High-level languages are basically symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes. Each instruction in the high level language is translated into many machine language instructions thus showing one-to-many translation.

**Advantages of High Level Language** : Following are the advantages of a high level language:

User-friendly  Similar to English with vocabulary of words and symbols  Therefore it is easier to learn.  They require less time to write.  They are easier to maintain.  Problem oriented rather than 'machine' based.  Program written in a high-level language can be translated into many machine language and therefore can run on any computer for which there exists an appropriate translator.  It is independent of the machine on which it is used i.e.Programs developed in high level language can be run on any Computer

**What is the use of jumping statement.**

jump statements are used to interrupt the normal flow of program. Types of Jump Statements

* Break
* Continue
* GoTo

**Break Statement :** The break statement is used inside loop or switch statement. When compiler finds the break statement inside a loop, compiler will abort the loop and continue to execute statements followed by loop.

#include<stdio.h>

void main()

{ int a=1;

while(a<=10)

{ if(a==5)

break;

printf("Statement %d.",a);

a++;

}

printf("\nEnd of Program.");

}

**Output :** Statement 1. Statement 2. Statement 3. Statement 4. End of Program.

[**Why we use 'include stdio.h'?**](https://www.urbanpro.com/c-language/why-we-use-include-stdio-h)

STDIO.H is a file which contain declaration of many functions and Macros which required to get input from input devices and show output on output screen of C Program. It is not compulsory to include "stdio.h" as we can provide input to c program using any other program like DOS and can store output to any FILE.

**What are the difference between while and do while loop.**

|  |  |  |
| --- | --- | --- |
| **Basis** | **While** | **Do While** |
| **Definition** | The loop which continues till the assertion holds true and repeats always. | The loop which holds true for particular directions. |
| **Statement** | Only one assertion for all of the package deal to work | Requires separate assertion for all of the while circumstances. |
| **Expression** | While (situation) . | Do while (situation); |
| **Execution** | System crashes as a result of speedy and steady repetitions. | Only executes the assertion for the actual time interval of the particular worth. |
| **Nature** | Takes much less time to execute however and the code is shorter. | Takes extra time to execute and code turns into longer. |

**What are formatted and unformatted input output functions?**

**Formatted and Unformatted Input/Output :** Unformatted Input/Output is the most basic form of input/output. Unformatted input/output transfers the internal binary representation of the data directly between memory and the file. Formatted output converts the internal binary representation of the data to ASCII characters which are written to the output file. Formatted input reads characters from the input file and converts them to internal form. Formatted I/O can be either "Free" format or "Explicit" format.

**What are the difference Macro and Function ?**

|  |  |
| --- | --- |
| **Macro** | **Function** |
| Macro is Preprocessed | Function is Compiled |
| No Type Checking is done in Macro | Type Checking is Done in Function |
| Using Macro increases the code length | Using Function keeps the code length unaffected |
| Use of macro can lead to side effect at later stages | Functions do not lead to any side effect in any case |
| Speed of Execution using Macro is Faster | Speed of Execution using Function is Slower |
| Before Compilation, macro name is replaced by macro value | During function call, transfer of control takes place |
| Macros are useful when small code is repeated many times | Functions are useful when large code is to be written |
| Macro does not check any Compile-Time Errors | Function checks Compile-Time Errors |