Q-1: When is a switch statement better than multiple if statements?

Ans-1: Switch statement is generally best to use when you have more than two conditional expressions based on a single variable of numeric type. For instance, rather than the code

if (x == 1)

printf(“x is equal to one.\n”);

else if (x == 2)

printf(“x is equal to two.\n”);

else if (x == 3)

printf(“x is equal to three.\n”);

else

printf(“x is not equal to one, two, or three.\n”);

the following code is easier to read and maintain:

switch (x)

{

case 1: printf(“x is equal to one.\n”);

break;

case 2: printf(“x is equal to two.\n”);

break;

case 3: printf(“x is equal to three.\n”);

break;

default: printf(“x is not equal to one, two, or three.\n”);

break;

}

Q-2: Is a default case necessary in a switch statement?

Ans-2: Not needed.If condition matches any one of the statement it executes.If does not match it executes default statement.

**Example with Default Statement**  
switch(1)  
{  
case 1:  
printf("ISC");  
break;

default:  
printf("Default");  
break;  
}  
In this program,ISC will be printed.  
  
  
switch(2)  
{  
case 1:  
printf("ISC");  
break;  
default:  
printf("Default");  
break;  
}  
In this program,Default will be printed.  
  
**Example without Default Statement**  
  
switch(1)  
{  
case 1:  
printf("ISC");  
break;  
}  
In this program,ISC will be printed.  
  
switch(2)  
{  
case 1:  
printf("ISC");   
break;  
}  
In this program,nothing will be printed.

Q-3: What are the differences between break and continue keyword?

Ans-3: The **break** statement allows you to exit a loop from any point within its body, bypassing its normal termination expression. When the **break** statement is encountered inside a loop, the loop is immediately terminated, and program control resumes at the next statement following the loop.

Example:

**#include <stdio.h>**

**int main()**

**{**

**int t ;**

**for ( ; ; ) {**

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

**if ( t==10 ) break ;**

**}**

**printf("End of an infinite loop...\n");**

**}**

The **continue** statement is somewhat the opposite of the **break** statement. It forces the next iteration of the loop to take place, skipping any code in between itself and the test condition of the loop.

Example:

**#include <stdio.h>**

**int main()**

**{**

**int x ;**

**for ( x=0 ; x<=100 ; x++) {**

**if (x%2) continue;**

**printf("%d\n" , x);**

**}**

**}**

**Q-4:** What are the differences between for, do while and while loop?

Ans-4:Do while loop: A form of programming loop in which the condition for termination is computed each time around the loop. There are several variants on this basic idea. For example,

while <condition> do  
  
  <statements>  
  
 end   
  
and also   
  
repeat  
  
 <statements>

untill<condition>

For loop: The initialization statement is executed only once at the beginning of the for loop. Then the test expression is checked by the program. If the test expression is false, for loop is terminated. But if test expression is true then the code/s inside body of for loop is executed and then update expression is updated. This process repeats until test expression is false.

for(initialization statement; test expression; update statement) {

code/s to be executed;

}

While loop: A **while** loop in C programming repeatedly executes a target statement as long as a given condition is true.

while(condition) {

statement(s);

}

Q-5: What are global and local variables?

Ans-5:Local: These variables only exist inside the specific function that creates them. They are unknown to other functions and to the main program. As such, they are normally implemented using a stack. Local variables cease to exist once the function that created them is completed.

Global: These variables can be accessed by any function comprising the program. They are implemented by associating memory locations with variable names. They do not get recreated if the function is recalled.

Q-6: What is pass by value in functions?

Ans-6:    
Recall that the variables in the formal parameter list are always *local variables* of a function

 [Consider this example program](http://www.cs.fsu.edu/~myers/c++/examples/func/byval.cpp)

* The  twice  function takes two integer parameters, and multiplies each by 2.
* Note that the original variables passed into the function from main() are **not** affected by the function call
* The local parameters a and b are *copies* of the original data sent in on the call

 This is known as **Pass By Value** - function parameters receive copies of the data sent in.

void Func1(int x, double y)

{

x = 12; // these lines will not affect the caller

y = 20.5; // they change LOCAL variables x and y

}

Q-7: What are the loop control statements?

Ans-7: With loop control statements, you can repeatedly execute a block of code. There are two types of loops:

* [for](http://www.mathworks.com/help/matlab/ref/for.html) statements loop a specific number of times, and keep track of each iteration with an incrementing index variable.

For example, preallocate a 10-element vector, and calculate five values:

x = ones(1,10);

for n = 2:6

x(n) = 2 \* x(n - 1);

end.

* [while](http://www.mathworks.com/help/matlab/ref/while.html) statements loop as long as a condition remains true.

For example, find the first integer n for which factorial(n) is a 100-digit number:

n = 1;

nFactorial = 1;

while nFactorial < 1e100

n = n + 1;

nFactorial = nFactorial \* n;

end.

Q-8: What are a finite and an infinite loop?

Ans-8:Infinite loop: An infinite loop is a piece of coding that lacks a functional exit so that it repeats indefinitely.

int main()

{

**for** (;;); *// or while (1);*

}

Finite loop: It's easy to write a loop which looks infinite but in fact completes quite quickly; for instance, in the C code-

for (int i = 1; i > 0; i++);

Q-9: Can one function call another function?

Ans-9: for example below I've defined a function and in it i called 'findFile' function,it works but i want to know is it basically right or i shouldn't call it in another function?

void employment::deleteRecord(std::vector<string>&lines2 ,string id3)

{

//...

findFile(lines2 , id4);//<HERE IS THE FUNCTION THAT I'VE CALLED IT

//...

}

**Q-10:** What is a constant?

Ans-10: Constant in C means the content whose value does not change at the time of execution of a program.

**Different Types of C Constants :**

| **Constant** | **Type of Value Stored** |
| --- | --- |
| Integer Constant | Constant which stores integer value |
| Floating Constant | Constant which stores float value |
| Character Constant | Constant which stores character value |
| String Constant | Constant which stores string value |