Difference Between `break` and `continue`	
break	**continue**
A `break` can appear in both `switch` and loop (`for`, `while`, `do`) statements.	A `continue` can appear only in loop (`for`, `while`, `do`) statements.
A `break` causes the `switch` or loop statements to terminate the moment it is executed. Loop or `switch` ends abruptly when break is encountered.	A `continue` doesn't terminate the loop, it causes the loop to go to the next iteration. All iterations of the loop are executed even if `continue` is encountered. The `continue` statement is used to skip statements in the loop that appear after the `continue`.
The `break` statement can be used in both `switch` and loop statements.	The `continue` statement can appear only in loops. You will get an error if this appears in switch statement.
When a `break` statement is encountered, it terminates the block and gets the control out of the `switch` or loop.	When a `continue` statement is encountered, it gets the control to the next iteration of the loop.
A `break` causes the innermost enclosing loop or `switch` to be exited immediately.	A `continue` inside a loop nested within a `switch` causes the next loop iteration.

Difference between If-else & switch

Basis for Comparison	if-else	switch
Basic	Which statement will be executed depend upon the output of the expression inside if statement.	Which statement will be executed is decided by user.
Expression	if-else statement uses multiple statement for multiple choices.	switch statement uses single expression for multiple choices.
Testing	if-else statement test for equality as well as for logical expression.	switch statement test only for equality.
Evaluation	if statement evaluates integer, character, pointer or floating-point type or boolean type.	switch statement evaluates only character or integer value.
Sequence of Execution	Either if statement will be executed or else statement is executed.	switch statement execute one case after another till a break statement is appeared or the end of switch statement is reached.
Default Execution	If the condition inside if statements is false, then by default the else statement is executed if created.	If the condition inside switch statements does not match with any of cases, for that instance the default statements is executed if created.
Editing	It is difficult to edit the if-else statement, if the nested if-else statement is used.	It is easy to edit switch cases as, they are recognized easily.

Structure vs. Union

Structure	Union
Struct keyword is used to define a structure.	Union keyword is used to define a union.
Members do not share memory in a structure.	Members share the memory space in a union.
Any member can be retrieved at any time in a structure.	Only one member can be accessed at a time in a union.
Several members of a structure can be initialized at once.	Only the first member can be initialized.
Size of the structure is equal to the sum of size of the each member.	Size of the union is equal to the size of the largest member.
Altering value of one member will not affect the value of another.	Change in value of one member will affect other member values.
Stores different values for all the members.	Stores same value for all the members.

Why c is called structured programing language??

- C is called a structured programming language because to solve a large problem, C programming language divides the problem into smaller modules called functions or procedures each of which handles a particular responsibility. The program which solves the entire problem is a collection of such functions.

Actual parameters: The parameters that appear in function calls.
Formal parameters: The parameters that appear in function declarations.
For example: We have a function declaration like this:

```
int sum(int a, int b);
The a and b parameters are formal parameters.
We are calling the function like this:
int s = sum(10, 20); //Here 10 and 20 are actual parameters or
int s = sum(n1, n2); //Here n1 and n2 are actual parameters
```

What is Function Call By Reference?

When we call a function by passing the addresses of actual parameters then this way of calling the function is known as call by reference. In call by reference, the operation performed on formal parameters, affects the value of actual parameters because all the operations performed on the value stored in the address of actual parameters.

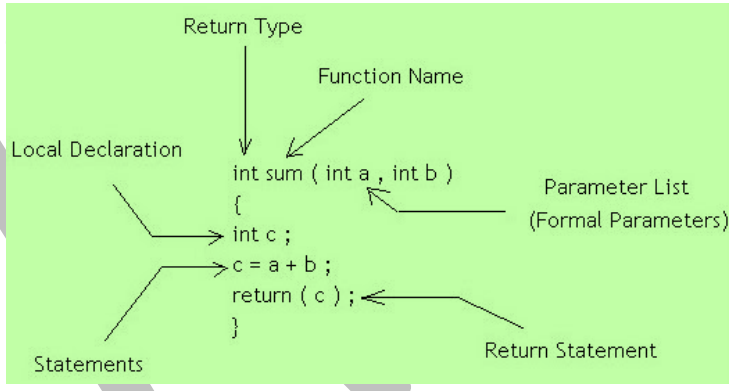
Difference between *call by value* and *call by reference*	
call by value	**call by reference**
In *call by value*, a copy of actual arguments is passed to formal arguments of the called function and any change made to the formal arguments in the called function have no effect on the values of actual arguments in the calling function.	In *call by reference*, the location (address) of actual arguments is passed to formal arguments of the called function. This means by accessing the addresses of actual arguments we can alter them within from the called function.
In call by value, actual arguments will remain safe, they cannot be modified accidentally.	In *call by reference*, alteration to actual arguments is possible within from called function; therefore the code must handle arguments carefully else you get unexpected results.

Components of a function

Return-type: This specifies the data type of the value being returned by the function. A function may or may not return a value. If the function does not return a value then the return type is void. In this case, the return value is an integer value c, which means the return type would be *int*.

Parameter list: The parameter list is the list of formal parameters being passed onto the function. In this case, there are two parameters of type *int* passed to the function.

Local variables or local declarations: The variables that are declared inside the function are called local variables. The scope of these variables lies within the function and they are not accessible outside the function.



Function body: A function body comprises of everything inside the curly brackets { and } following the return type, function name and the parameter list.

Function Name: A function name can be anything that you want. The standard is to make it something related to what it is supposed to do. The naming convention follows the same rule as that of variable naming convention in C.

Function declaration : A function declaration simply tells the compiler all about the function. These include the function's name, the return type and the number and types of parameters. The body of the function having the function definition can be defined somewhere else. A function declaration has the following parts:

```
return_type function_name( parameter list );
```

The example above can be declared as follows:

```
1. int sum(int num1, int num2);
```

or

```
1. int sum(int, int);
```

Types of function

Depending on whether a function is defined by the user or already included in C compilers, there are two types of functions in C programming

There are two types of function in C programming:

- Standard library functions
- User defined functions