



**Al-mustaqbal University collage**  
**Biomedical Engineering Department**  
**Class: First**  
**Subject: Computer Skills & Programming**

***Lecture 6 part2: Jump statements in C++ and Functions***

**BY**  
**IT. Zahraa Abdzaid AbdelAbbas**  
**Supervised by: ASS.T. Hala Fadel Alaiwi**

# Jump statements in C++

Jump statements are used to manipulate the flow of the program if some conditions are met. It is used to terminating or continues the loop inside a program or to stop the execution of a function. In C++ there is four jump statement: continue, break, return, and goto.

## 1-Continue Statement in C++

Continue is also a loop control statement just like the break statement. continue statement is opposite to that of break statement, instead of terminating the loop, it forces to execute the next iteration of the loop.

### Syntax:

`continue;`

Ex:write a program which prints number from 1 to 10 and but not 6.

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

int main() {
    // loop from 1 to 10
    for (int i = 1; i <= 10; i++) {

        // If i is equals to 6,
        // continue to next iteration
        // without printing
        if (i == 6)
            continue;

        else
            // otherwise print the value of i
            cout<<("%d ", i);
    }

    getch();}
```

### Output:

1 2 3 4 6 7 8 9

## 2-Break Statement in C++

We have already seen the break statement used in the switch statement. It is also used in while, do...while, and for loops. When it is executed, it terminates the loop, “breaking out” of the iteration at that point.

Syntax:

```
break;
```

Ex: Write a program to find the sum of the series (the sum doesn't exceed 50000):

$$1^3 + 3^3 + 5^5 + 7^7 + \dots$$

```
# include <iostream.h>
#include<conio.h>
# include <math.h>
int main() {
int i =1,sum=0 ;
int n;
cin>>n;
while ( i>0) {
sum = sum + pow(i,i) ;
if (sum>50000) break; i+= 2 ;
} cout<<" The sum of the series = "<<sum<<endl; getch();}
```

## 3-goto statement in C++

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

Syntax1		Syntax2
goto label;		label:
.		.
.		.
.		.
label:		goto label;

**Ex: write C++ program to check if a number is even or not using (goto statement).**

```
#include <iostream>
#include<conio.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:
    cout << num << " is even";
    // return if even
    return;
odd:
    cout << num << " is odd";
}

// Driver program to test above function
int main()
{
    int num = 26;
    checkEvenOrNot(num);
    getch();
}
```

**Output:**

26 is even

## **The switch STATEMENT (Multiple Choice Statement)**

When a multiple selection is required we may use switch statement which is illustrated

below:

```
switch (expression or variable )
{
case value1 : statement1; break;
case value2 : statement2; break;
.....
case value n : statement n; break;
default : statement; }
```

During execution of the program, the expression is evaluated and compared with the values mentioned in different cases of switch expression. If the value matches a value of a particular case, the statements in that case are executed. If no case-value matches with the value of the expression the program goes to the last statement which is a default statement as shown in figure below:

Note The word break means exit from switch statement.

Ex: Write a program to receive an arithmetic operator and two integers, the program performs the arithmetic operation on the two numbers (use switch statement).

```
#include<iostream.h>
#include<conio.h>
int main()
{ char ch;
int x,y;
cout<<"Enter the arithmetic operator : "<<endl;
cin>>ch;
cout<<"Enter the two numbers : "<<endl;
cin>>x>>y;
switch(ch)
{ case '+' : cout<<x + y ; break;
case '-' : cout<<x - y ; break;
case '*' : cout<<x * y; break;
case '/' : cout<<x / y; break;
case '%' : cout<<x % y; break;
default : cout<<" Error try again";
} getch();}
```

## **FUNCTIONS**

A function is a group of statements used to perform a certain operation. It is called from some point of the main program or another function

.There are several advantages of using functions:

- Functions allow for breaking down the program into discrete units.
- Programs that use functions are easier to design, program, debug and maintain.

Functions are of two types:

1. Functions return a value to the main program end with return statement.
2. Functions do not return a value defined by the word void.

### **1. FUNCTIONS RETURN A VALUE:**

The function definition is illustrated below:

```
type function_name (type parameter1, type parameter2, ..... )
{
statements ;
return value;
}
```

In the above definition the first word is the type of the function, it is the type of data it returns. The second item is the name of the function. (type parameter1, type parameter2, ..... ) are called arguments of the function. For example:

```
int sum(int x, int y)
{
int sum=x+y;
return sum;
}
```

## **ACCESSING A FUNCTION (The Call of the Function)**

The main program calls the function by declaring a variable followed by the function name and its arguments, as follows:

```
type variable_name=function_name(arguments)
```

For example:

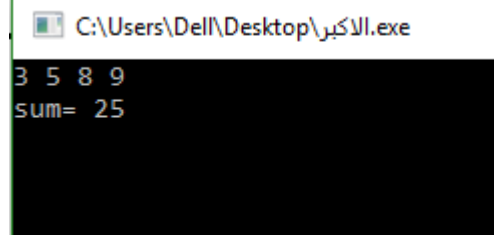
```
int s1=sum(a, b) ;
```

Ex: Write a function finds the sum of two numbers, the main program calls the function to find the sum of four numbers.

```
#include<iostream.h>
#include<conio>
float sum(int x, int y)
{
    float z=x+y;
    return z;
}
main( ) {
    float a,b,c,d;
    cin>>a>>b>>c>>d;
    float s1=sum(a,b);
    float s2=sum(c,d);
    float s=sum(s1,s2);
    cout<<"sum= "<<s;
    getch();
}
```

float s=sum(sum(a,b),sum(c,d));

## Output



```
C:\Users\Dell\Desktop\الاكبر.exe
3 5 8 9
sum= 25
```

## 2. VOID FUNCTIONS

A function does not return a value to the main program and is known as a procedure or a subroutine. In C++, such a function is identified simply by placing the word void before the name of the function as shown below:

```
void function_name (arguments)
```

For example :

```
void AA(int x, int y)
{
    int sum;
    sum=x+y;
    cout<<sum;
}
```

This type of functions is called by its name without using any variable as shown:

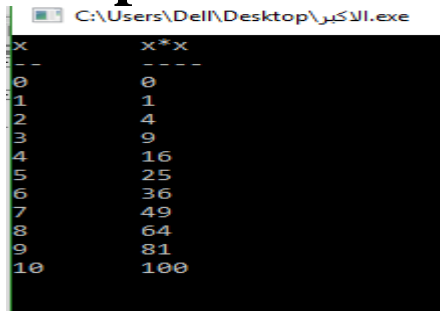
```
AA(a, b);
```

Ex: Write a function that find and print the square of an integer, the main program calls this function to find the squares of 0-10.

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

void square(int x)
{
    int z;
    z=0;
    cout<<x<<"\t"<< x*x <<endl;
}
main( )
{
    cout<<"x\t"<<"x*x"<<endl;
    cout<<"--\t"<<"----"<<endl;
    int i;
    for(i=0; i<=10; i++)
        square(i);
    getch();
}
```

## Output



The screenshot shows a Windows command prompt window titled "C:\Users\Dell\Desktop\الاكبر.exe". The output is as follows:

x	x*x
0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

The following function does not receive arguments form the main program and does not return value to the main program. The definition is as shown below:

```
void AA(void)
{
    cout<<"this is a C++ program";
}
}
```

The call of this function is as shown below:

```
AA( );
```



Thank you