C++ goto statement with example

The goto statement in C++ is used to transfer control anywhere in the program unconditionally. because its jumps from one part to another part of the program, hence it is also called the jumping statement,

The goto-statement is different from other control statements. In other control statements where execution is performed under a condition, in goto-statement, we can transfer the control to another part of the program without any condition(as an unconditional transfer). Hence the goto statement is also called the conditional transfer statement.

However, using other control statements we can perform the conditional transfer. here we will perform both conditional and unconditional transfer.

There are two ways to use goto statement in the Program,

1st

```
goto label;

  ..........

  ..........
```

```
lable:

  ..........

  ..........
```

second

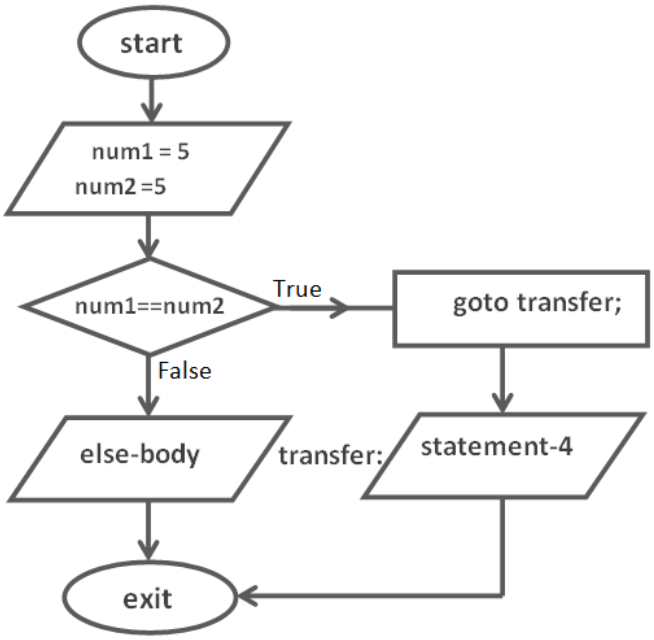
```
label:

  ...........

  ...........



goto label;

  ..........

  ..........
```

where **label** is an identifier.

here is the flow diagram of a `goto` statement,

C++ goto Flow Diagram

as you can see in the diagram after statement goto statement used no condition is defined.



compare the above diagram with `continue` and `break` statements in C++.

Here is the Example

Example of goto statement in C++

In the below Program we will find out whether a given number is equal or not.

Here, the goto statement will jump under a condition we can say a conditional jump will be performed here because `goto` is a body of an if-statement so, as soon as the condition becomes true in the if-statement `goto` statement will transfer the control to the other part of the Program.

```
#include<iostream>

using namespace std;
```

```
int main()

 {

    int num1,num2;

    cout<<"Enter Two number: ";

    cin>>num1>>num2;

      if(num1==num2)

           goto transfer;

      else

           cout<<"Given number are not Equal";



   transfer: {

       cout<<"\nGiven number are Equal "<<x;

     }

return 0;

}
```

OUTPUT

```
Enter Two number: 5  5
```

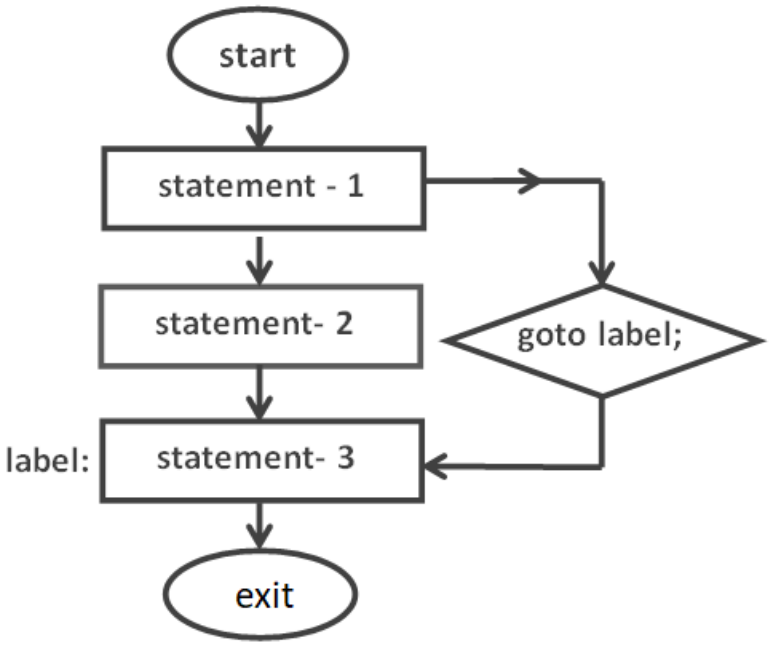
```
Given number are Equal
```

Explanation

Here we have given the label name transfer. In the program, we declared `goto` statement inside the if-body.

so as soon as the condition becomes True in the **if-statement**, the control is transferred outside the **if-statement** and if the condition becomes False than **else-body** is executed.

the above program will execute as follows,



now we performed an unconditional jump in the following Program,

goto statement with break statement in C++

In the below Program, we add two number given by the user in a different way, here we use goto with break statement within a condition to make a loop both these statements will create a loop without using any looping statement.

```
#include<iostream.h>

using namespace std;

 int main()

  {

   int num1,num2,sum;

    start:

      cout<<"\nEnter two number: ";

      cin>>num1>>num2;

       if(num1== 0 || num2==0)

           break;

      sum = num1+num2;

      cout<<"Total: "<<sum;

      goto start;

  cout<<"this is next statement";

 }
```

OUTPUT

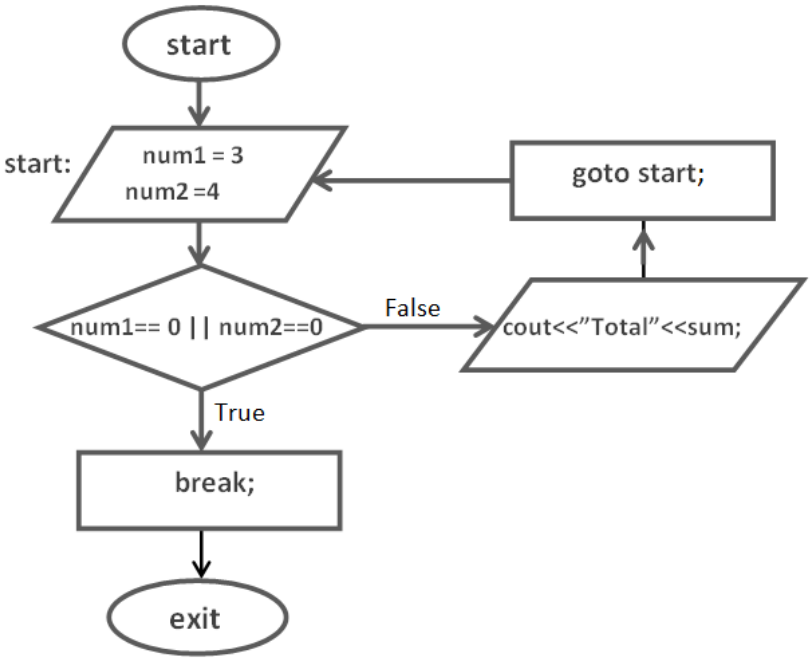
```
Enter two number: 3 4

Total: 7

Enter two number: 6 3

Total : 9

Enter two number: 0 4
```

Explanation

In the Program, we declare a **break-statement** inside the **if-statement** within a condition and the condition is that the program will remain in execution until the value of one of the variables is 0.

The program will keep adding numbers until the value of one of the variables becomes 0. As soon as the value of either one is 0, the **if-statement** will be executed, which will terminate the `goto` statement. And the execution will go to the next statement.

above program execute as follows,



goto statement with continue

In the following Program, we will print even number from a series.

```
#include<iostream.h>

using namespce std;

 int main()

   {

     for(int i=0; i<=20; i++)

      {

         if(i%2==0)
```

```
            goto even; // even number

         else

            continue; // odd number skip

        even:

          cout<<i<<"\t";

      }

return 0;

  }
```

OUTPUT

```
0  2  4  6  8  10  12  14  16  18  20
```

Thing to know

The goto-statement transfers execution anywhere in the program and the function can also perform similar tasks. But where the function can return the control the **goto-statement** does not. This means that `goto` does only one-way transfer.

the goto-statement is an important role in improving programming skill. it is good practice to improve logics. as stated **goto** transfer the control anywhere in the program unconditionally so it may occur bugs in the large codes and also it is harmful in security so a developer does not consider goto as a good programming practice.

however, using control statements with each other you can develop your ability to create logics, so a beginner can go with a goto-statement. later you automatically know where `goto` statement should be used and where not.