Loops

Course Code: CSC1102 &1103 Course Title: Introduction to Programming

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**4**	**Week No:**	**2 (1X1.5 hrs), 3 (1X1.5 hrs)**	**Semester:**	
Lecturer:	*Name & email*				

Lecture 4: Outline

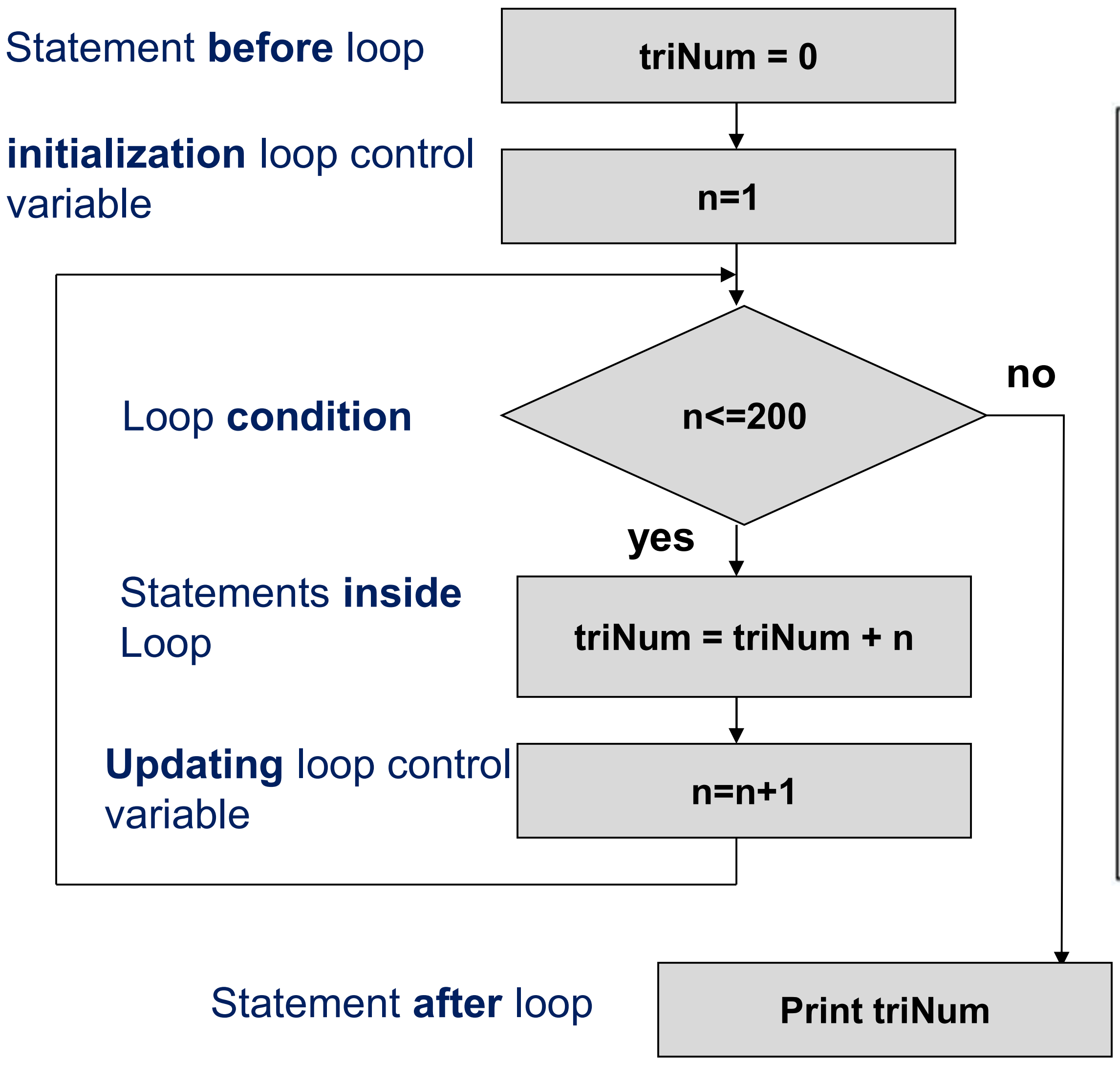
- ❑ `for` Loops
- ❑ Increment Operator
- ❑ Program Input
- ❑ `for` Loop Variants
- ❑ The `while` Statement
- ❑ The `do` Statement
- ❑ The `break` Statement
- ❑ The `continue` Statement

Program Looping

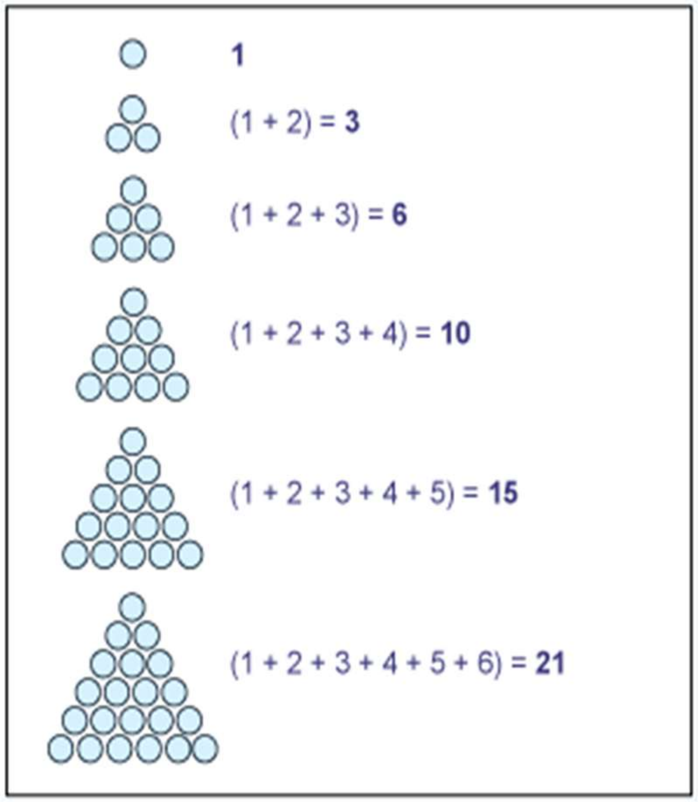
- Looping: doing one thing over and over
- Program loop: a set of statements that is executed repetitively for a number of times
- Simple example: displaying a message 100 times

Program looping: enables you to develop concise programs containing repetitive processes that could otherwise require many lines of code !

Example – 200th triangular number flowchart


Statement **before** loop
triNum = 0
initialization loop control variable
n=1
Loop **condition**
n<=200
no
yes
Statements **inside** Loop
triNum = triNum + n
Updating loop control variable
n=n+1
Statement **after** loop
Print triNum

Some Triangular number examples

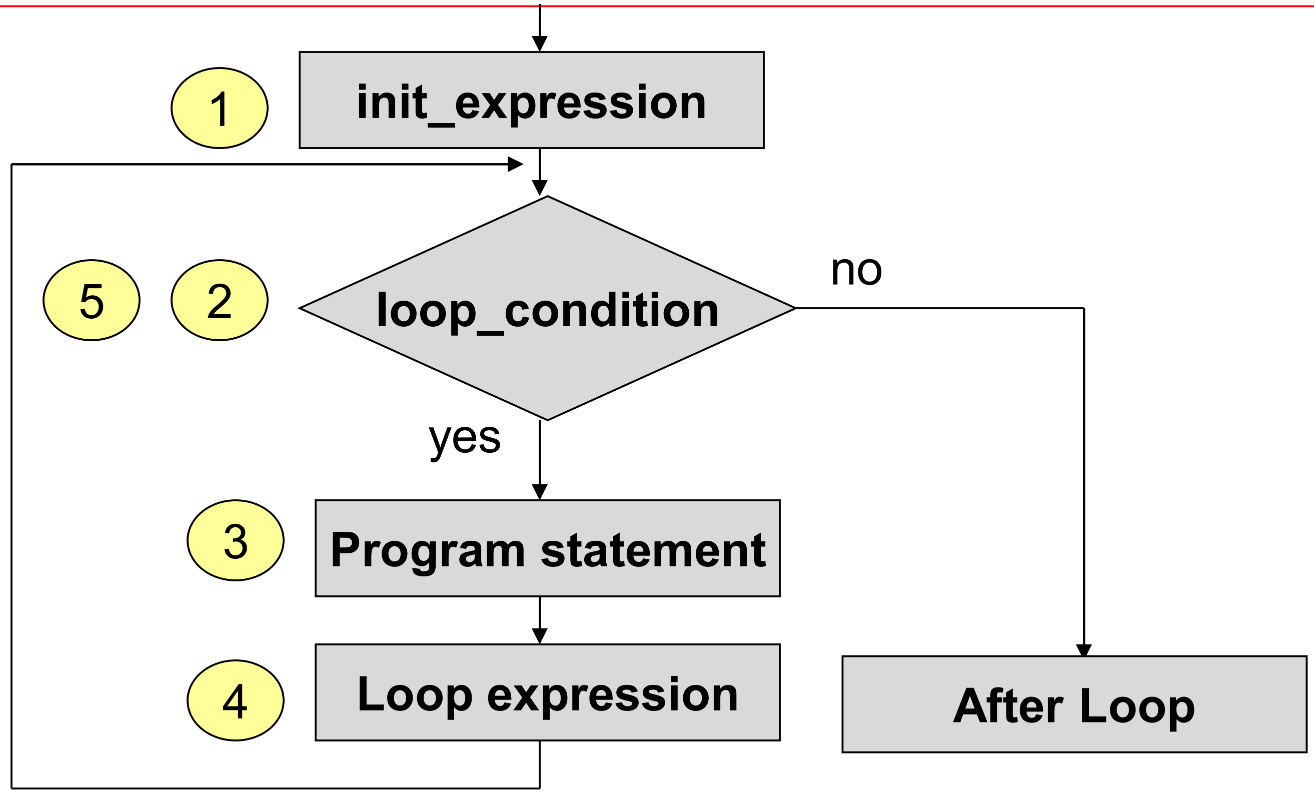


Example - `for`

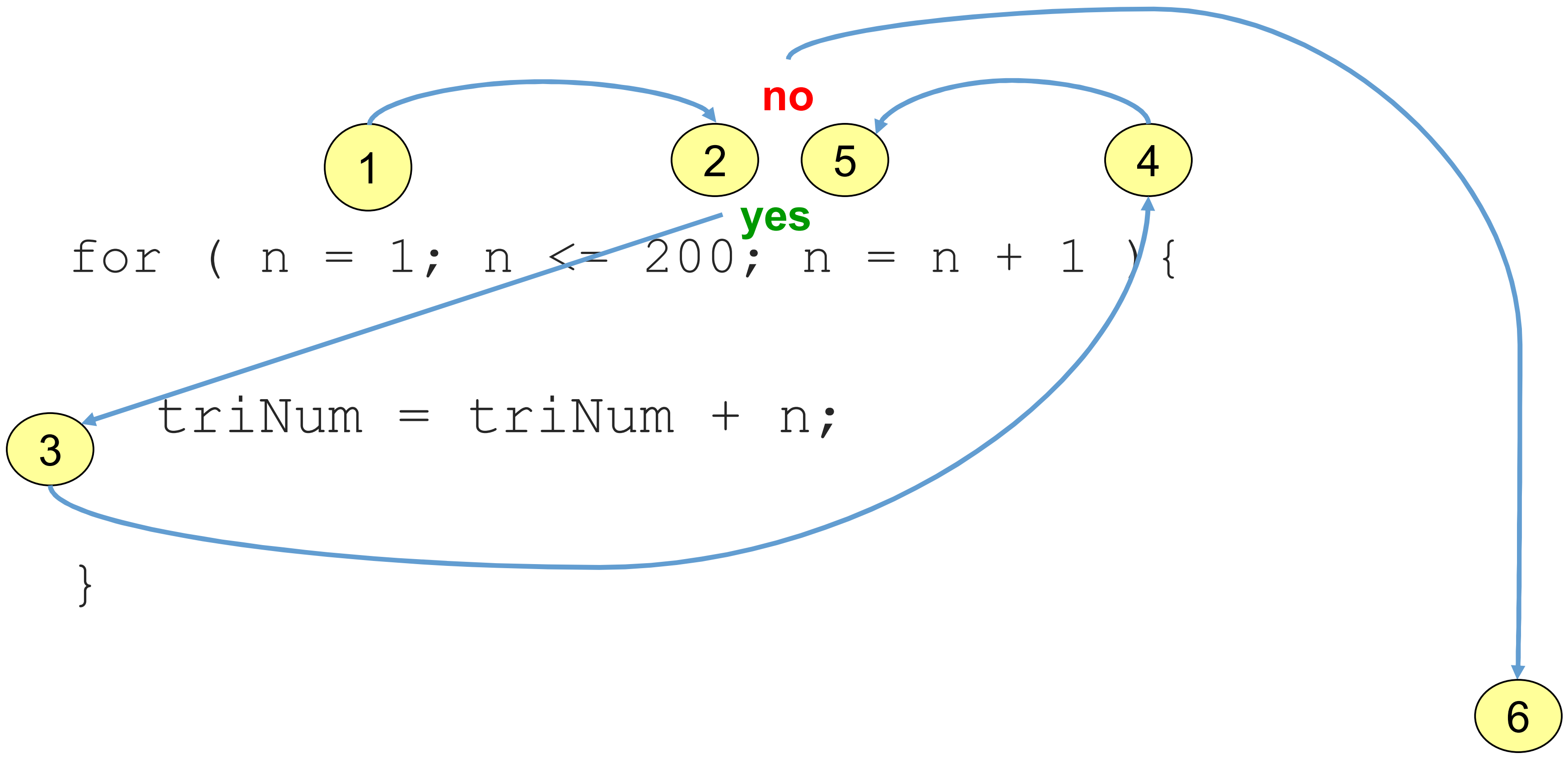
```
/* Program to calculate the 200th triangular number
Introduction of the for statement */

#include<iostream>
using namespace std;
int main ()
{
    int n, triNum;
    triNum = 0;

    for ( n = 1; n <= 200; n = n + 1 )
         triNum = triNum + n;

    cout<<"The 200th triangular number = " << triNum;
    return 0;
}
```

The `for` statement

```
for ( init_expression; loop_condition; loop_expression ){
                    program statements. . .
}
```



The `for` statement



```
for ( n = 1; n <= 200; n = n + 1 ){

    triNum = triNum + n;

}
```

How `for` works

- The execution of a for statement proceeds as follows:
  1. The initial expression is evaluated first. This expression usually sets a variable that will be used inside the loop, generally referred to as an *index* variable, to some initial value.
  2. The looping condition is evaluated. If the condition is not satisfied (the expression is false – has value 0), the loop is immediately terminated. Execution continues with the program statement that immediately follows the loop.
  3. The program statement that constitutes the body of the loop is executed.
  4. The looping expression is evaluated. This expression is generally used to change the value of the index variable
  5. Return to step 2.

Infinite loops

- It's the task of the programmer to design correctly the algorithms so that loops end at some moment !

```
// Program to count 1+2+3+4+5

int main (void)

{
    int  i, n = 5, sum =0;
    for ( i = 1; i <= n; n = n + 1 ){
        sum = sum + i;
      cout<<sum<<endl;
    }
    return 0;

}
```

What is wrong here ?

Does the loop end?

Relational operators

Operator	Meaning
==	Is equal to
!=	Is not equal to
<	Is less than
<=	Is less or equal
>	Is greater than
>=	Is greater or equal

The relational operators have lower precedence than all arithmetic operators:
`a < b + c` is evaluated as `a < (b + c)`

ATTENTION ! Do not confuse:
the "is equal to" operator == and the "assignment" operator =

ATTENTION when comparing floating-point values !
Only < and > comparisons make sense !

Increment operator

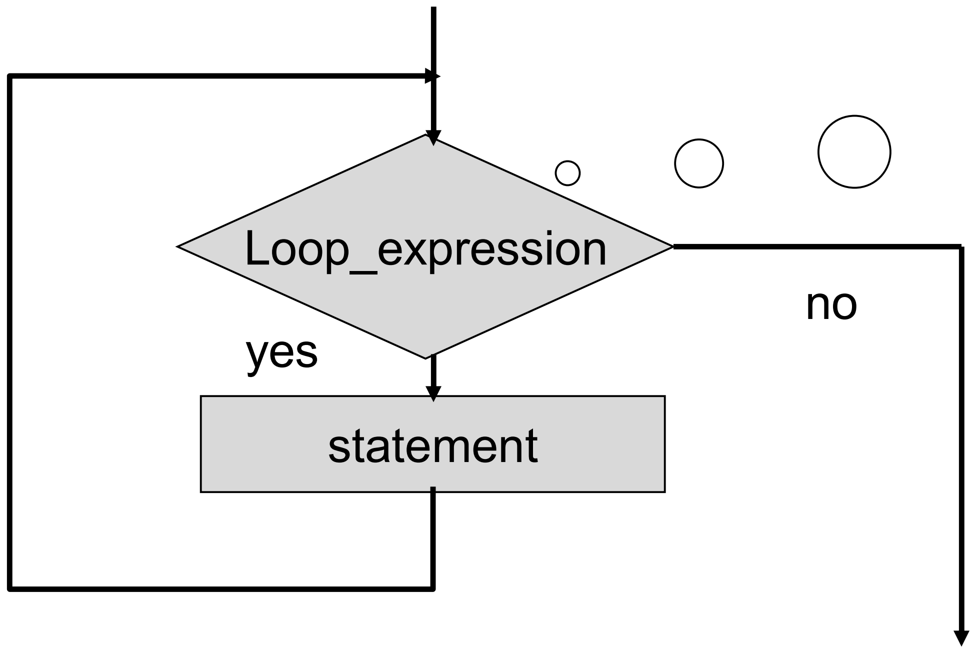
- Because addition by 1 is a very common operation in programs, a special operator was created in C for this.
- *Increment operator*: the expression ++n is equivalent to the expression n = n + 1.
- *Decrement operator*: the expression --n is equivalent to the expression n = n – 1
- Increment and decrement operators can be placed in front (*prefix*) or after (postfix) their operand.
- The difference between prefix and postfix:
- Example: if n=4:
    - a=n++ leads to a=4, n=5
    - a=++n leads to a=5, n=5

The `while` statement

while (*expression*){
        program statements . . .
}

```
while ( number <= 0 )
{
   cout<<"The number must be >0"<<endl;
   cout<<"Give a new number:  "<<endl;
   cin>>number;
}
```

The `while` statement



Loop with the test in the beginning !
Body might never be executed !

Program to calculate the 200th triangular number Using while Loop

```
#include<iostream>
using namespace std;

int main ()
{
   //Statements before loop
    int n, triNum;
    triNum = 0;

    n = 1; //initialization loop control variable

    while (n <= 200){//while loop condition

        triNum = triNum + n; //Statements inside loop
        n = n + 1;
    }
    //Statements after loop
    cout<<"The 200th triangular number = " << triNum;
    return 0;
}
```

Example:

- A program to find the greatest common divisor of two nonnegative integer values …

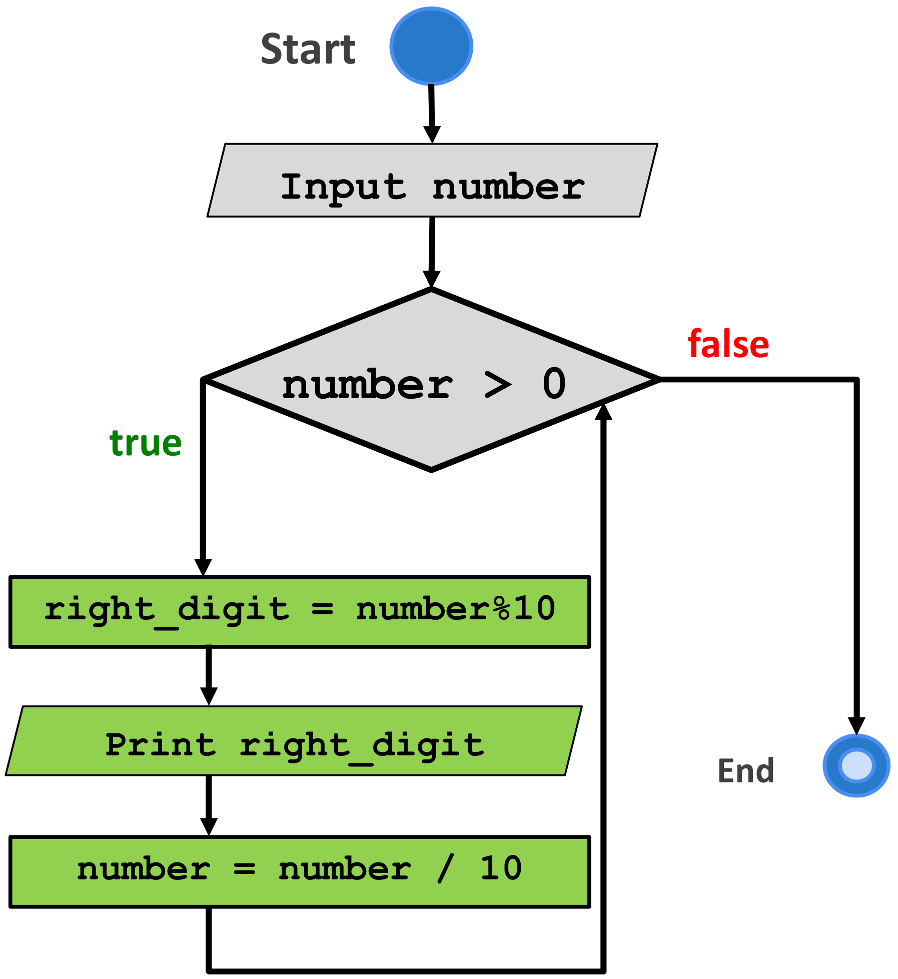
Example - `while`

```
/* Program to find the greatest common divisor
of two nonnegative integer values */
#include <iostream>
using namespace std;
int main (void)
{
int u, v, temp;
cout<<"Please type in two nonnegative integers."<<endl;
cin>>u>>v;
while ( v != 0 ) {
temp = u % v;
u = v;
v = temp;
}
cout<<"Their greatest common divisor is "<< u;
return 0;
}
```

Example:

- A program to print out the digits of a number in reverse order …

Example – Flowchart for Printing the digits of a number in reverse order.

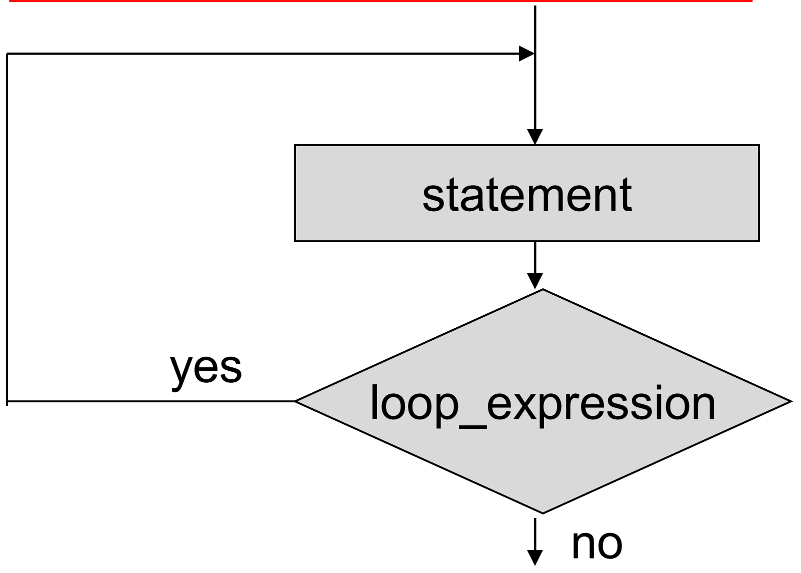


Example - while

```
// Program to reverse the digits of a number
#include <iostream>
using namespace std;

int main ()
{
        int number, right_digit;
        cout<<"Enter your number"<<endl;
        cin>>number;

while ( number != 0 ){

        right_digit = number % 10;
        cout<< right_digit;
        number = number / 10;
}
return 0;
}
```

The `do{}while()` statement

```
do{
        program statements . . .
}
while ( loop_expression );
```



Loop with the test at the end !
Body is executed at least once !

Which loop to choose ?

- Criteria: Who determines looping
  - Entry-condition loop -> for, while
  - Exit-condition loop -> do

- Criteria: Number of repetitions:
  - Indefinite loops ->while
  - Counting loops -> for

- You can actually rewrite any while as a for and viceversa !

The `break` Statement

- Can be used in order to immediately exiting from a loop

- After a break, following statements in the loop body are skipped and execution continues with the first statement after the loop

- If a break is executed from within nested loops, only the innermost loop is terminated

The `break` statement

//Programming style: don’t abuse break !!!

```
...
while ( number != 0 ) {
    // Statements to do something in loop
    cout<<"Stop, answer 1:” <<endl;
   cin>>answer;
    if(answer == 1)
        break; // very bad idea to do this
}
```

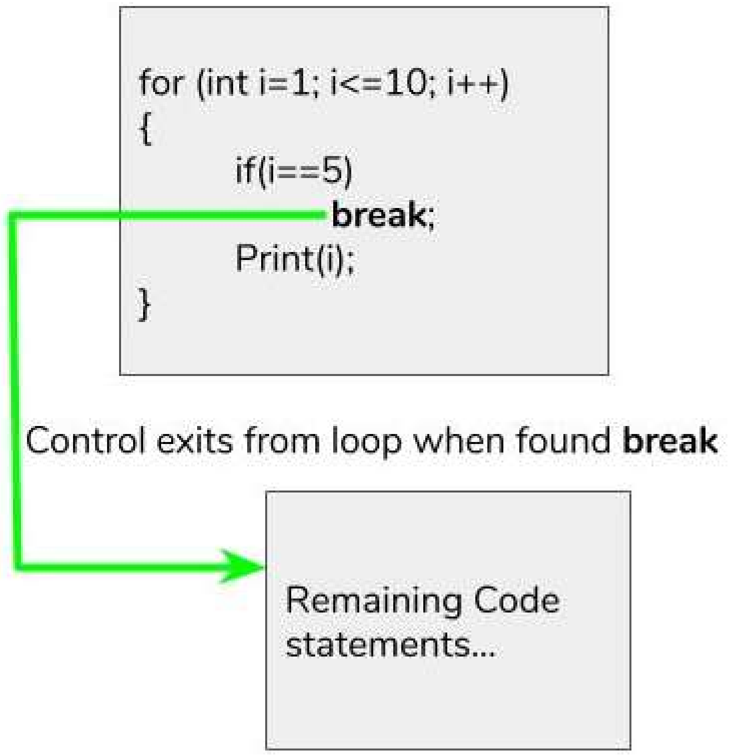
The `continue` statement

- Similar to the break statement, but it does not make the loop terminate, just skips to the next iteration

Example - `continue` statement

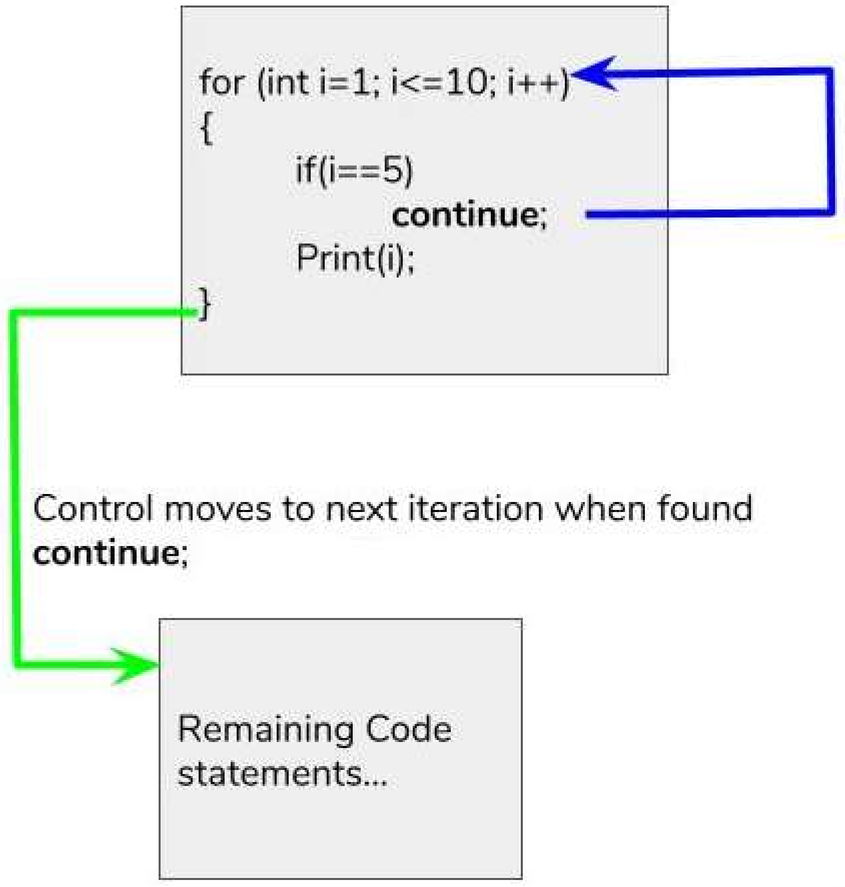
```
#include <iostream>
using namespace std;
int main()
{
    // loop from 1 to 10
    for (int i = 1; i <= 10; i++) {
    /* If i is equals to 6, continue to next iteration
    without printing */
        if (i == 6)
                continue;
        else
        cout << i << " "; // otherwise print the value of i
    }
    return 0;
}
```

Difference between break and continue



```
for (int i=1; i<=10; i++)
{
        if(i==5)
                break;
        Print(i);
}
```

Control exits from loop when found **break**
Remaining Code statements...



```
for (int i=1; i<=10; i++)
{
        if(i==5)
                continue;
        Print(i);
}
```

Control moves to next iteration when found **continue**;
Remaining Code statements...