Loops

Loops

REFERENCES

- Dr. Mohamed El-Desouki
  - While loop & Do while loop
  - For loop

- Adel Nasim
  - While loop & Do while loop
  - For loop

- Dr. Mostafa Saad
  - While loop
    - Video #1
    - Video #2
    - Video #3
  - For Loop
    - Video #1
    - Video #2
    - Video #3

Loops

Sometimes we need to repeat a specific course of actions either a specified number of times or until a particular condition is being satisfied.

- **This is called Repetition statements .**

So how can we choose specific statements to execute and other not?

- Here comes the answer

For example:

1- To calculate the Average grade for 10 students.

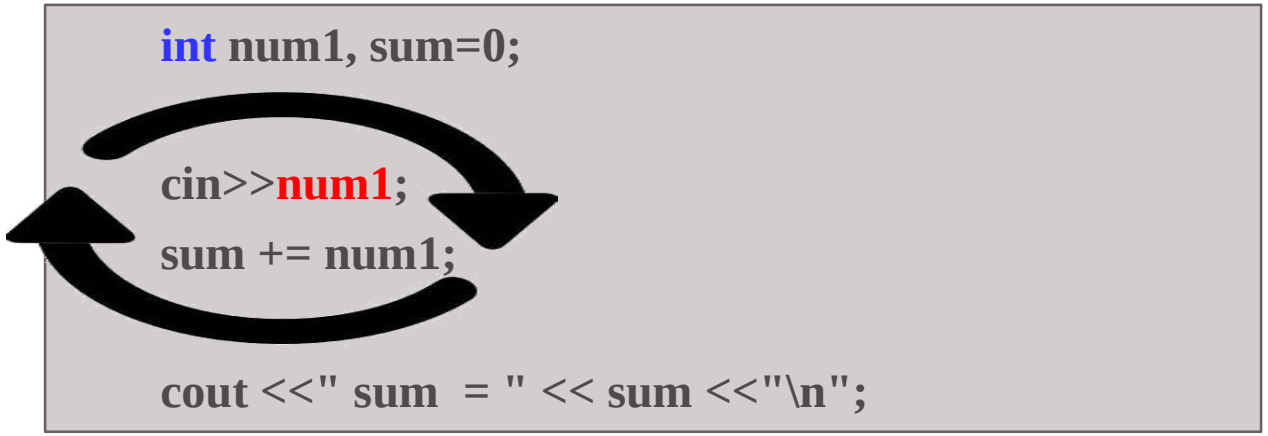
2- To calculate the bonus for 10 employees.

3- To sum the input numbers from the user as long as he/she enters positive numbers.

We will use the repetition statements to solve our problem.

Why Is Repetition Needed?

- Repetition allows you to efficiently use variables
- Can input, add, and average multiple numbers using a limited number of variables
- For example:
- to add five numbers:
  - Create a loop that reads a number into a variable and adds it to a variable that contains the sum of the numbers



Loops

- There are three methods by way of which we can repeat a part of a program. They are:

1. **While statement**
2. **Do-while statement**
3. **For statement**

While Loop

- The general form of the **`while`** statement is:

```
while (expression)
statement
```

➔ **`while`** is a reserved word.

- Statement can be simple or compound.
- Expression acts as a decision maker and is usually a logical expression.
- Statement is called the body of the loop.
- The parentheses and curly brackets are part of the syntax.

While Loop

`while` Looping (Repetition) Structure

```
initialise loop counter ;
while ( test loop counter using a condition )
{
    do this ;
    and this ;
    increment loop counter ;
}
```

- **loop counter is any numeric variable (int , float ,….).**
- **The initial value of the loop counter is up to you, but you have to increment it to avoid infinite loops.**

`while` Looping (Repetition) Structure

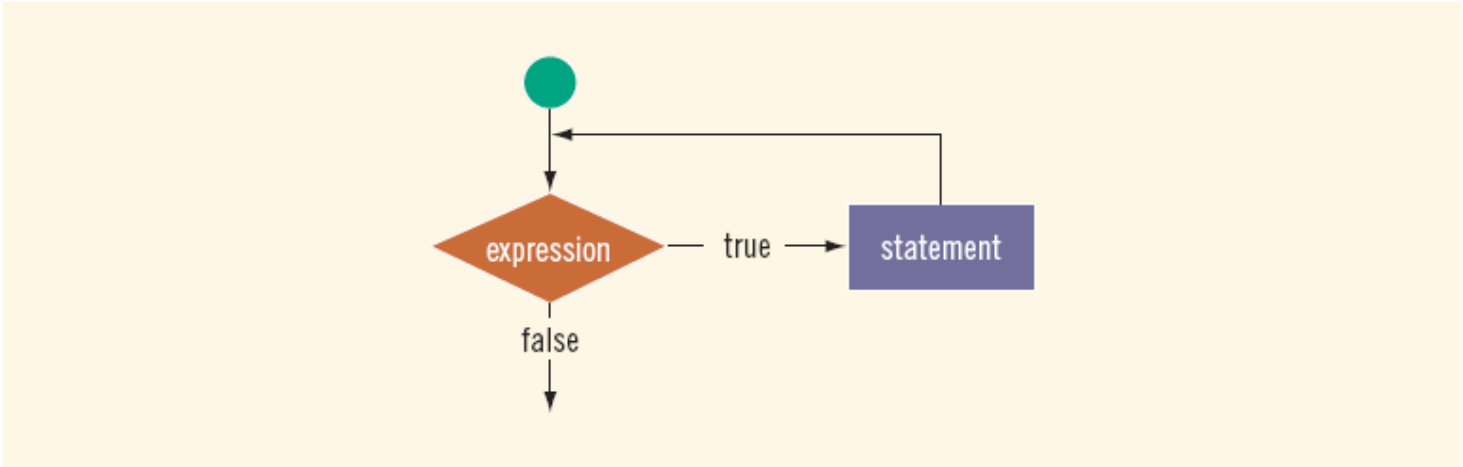


FIGURE 5-1 `while` loop

- Infinite loop: continues to execute endlessly.
  - Avoided by including statements in loop body that assure exit condition is eventually `false`

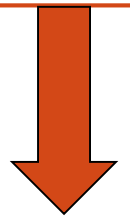
While Loop

`while` Looping (Repetition) Structure

- **Example** : Write a program that calculates and prints out the Average grade for 6 students .

```
int counter = 1;
int grade=0 , sum=0,num_of_students=6;
while (counter <=6)
{
cout<<"Enter grade for student no " << counter <<"\n";
cin>>grade;
sum += grade;
counter ++;
}
cout<<"Average Grade is " << float(sum)/num_of_students <<"\n";
```

The input: 10, 25, 30, 40, 50, 60



The output: 215/6 =35.83

HINT:

1- be careful with the initial value of counter, Here:
you can set it to Zero if you will use it during your Calculations
2- or simply divide the sum by the number of grades

`while` Looping (Repetition) Structure

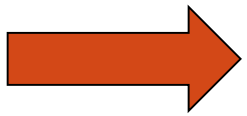
EXAMPLE 5-1

Consider the following C++ program segme

```
i = 0;                              //Line 1

while (i <= 20)                     //Line 2
{
    cout << i << " ";               //Line 3
    i = i + 5;                      //Line 4
}

cout << endl;
```



The output:

0 5 10 15 20

Designing `while` Loops

EXAMPLE 5-2

Consider the following C++ program segment:

```
i = 20;                          //Line 1
while (i < 20)                   //Line 2
{
    cout << i << " ";            //Line 3
    i = i + 5;                   //Line 4
}
cout << endl;                    //Line 5
```

It is easy to overlook the difference between this example and Example 5-1. In this example, in Line 1, `i` is set to 20. Because `i` is 20, the expression `i < 20` in the **`while`** statement (Line 2) evaluates to **`false`**. Because initially the loop entry condition, `i < 20`, is **`false`**, the body of the **`while`** loop never executes. Hence, no values are output and the value of `i` remains 20.

While Loop

Case 1: Counter-Controlled `while` Loops

- If you know exactly how many pieces of data need to be read, the `while` loop becomes a counter-controlled loop

```
counter = 0;              //initialize the loop control variable

while (counter < N) //test the loop control variable
{
      .
      .
      .
      counter++;          //update the loop control variable
      .
      .
      .
}
```

Case 2: Sentinel-Controlled `while` Loops

- Sentinel variable is tested in the condition and loop ends when sentinel is encountered

```
cin >> variable;                    //initialize the loop control variable

while (variable != sentinel)  //test the loop control variable
{
      .
      .
      .
      cin >> variable;        //update the loop control variable
      .
      .
      .
}
```

Case 3: Flag-Controlled `while` Loops

- A flag-controlled `while` loop uses a `bool` variable to control the loop
- The flag-controlled `while` loop takes the form:

```
found = false;          //initialize the loop control variable

while (!found)          //test the loop control variable
{
    .
    .
    .
    if (expression)
        found = true; //update the loop control variable
    .
    .
    .
}
```

While Loop

- `while` Looping (Repetition)

- **Try Yourself**

Write a program To print out the sum of the numbers entered from the user as long as he/she enters positive numbers.

While Loop

- Answer

```
int number=0 , sum=0;
cout <<" Enter Positive numbers to sum \n";
cin>>number;
while (number >= 0)
{
sum += number;
cout <<" Enter Positive numbers to sum \n";
cin>>number;
}
```

While Loop

- **Example** : Write a program that calculates and prints out the Average grade for 5 students or ends the program by entering -1.

```
int grade=0, counter =1, sum=0;
cout <<" Enter 5 grades or -1 To Exit \n";

while (counter <=5 && grade !=-1)
{
    cin>>grade;
    sum += grade;
    counter ++;
}

cout <<"The sum of grades is " << sum <<"\n";
```

Do...while Looping (Repetition) Structure

- General form of a do...while:

```
do
      statement
while (expression);
```

- The statement executes first, and then the expression is evaluated
- To avoid an infinite loop, body must contain a statement that makes the expression false
- The statement can be simple or compound
- Loop always iterates at least once

Do ... while Loop

Do...while Looping (Repetition) Structure

```
do
{
      this ;
      and this ;
      and this ;
      and this ;
} while ( this condition is true ) ;
```

```
while ( this condition is true )
{
      this ;
      and this ;
      and this ;
      and this ;
}
```

1- loop counter is any numeric variable (int , float ,....).

2- The initial value of the loop counter is up to you, but you have to increment it to avoid endless loops.

3- The Loop Body is Executed at least one Time.

Do ... while Loop

Do...while Looping (Repetition) Structure

EXAMPLE 5-15

```
i = 0;

do
{
    cout << i << " ";
    i = i + 5;
}
while (i <= 20);
```

The output:

0 5 10 15 20

Do … while Loop

- **Guess the OUTPUT**

A)

EXAMPLE 5-16

Consider the following two loops:

a.
```
i = 11;
while (i <= 10)
{
    cout << i << " ";
    i = i + 5;
}
cout << endl;
```

B)

b.
```
i = 11;
do
{
    cout << i << " ";
    i = i + 5;
}
while (i <= 10);

cout << endl;
```

Do … while Loop

➢ Answer

(A) **No Output !!**

As i=11 which is > 10
So, while loop body will not executed.

(B) **Output : 11**

Then change the value of i from 11 to 16
Via
Executing i=i+5

Do … while Loop

➢ Try Yourself

- Write a program that calculates and prints out the Average grade for 6 students .

Do … while Loop

➢ Answer

```
int counter = 1;
int grade=0 , sum=0;
do
{
cout <<"Enter grade for student no " << counter <<"\n";
cin >>grade;
sum += grade;
counter ++;
}
while (counter <=6) ;
cout <<"Average Grade is " << float(sum)/6<<"\n";    // or
cout <<"Average Grade is " << float(sum)/(counter-1)<<"\n";
```

For Looping (Repetition) Structure

- The general form of the **for** statement is:

```
for ( initial statement ; loop condition ; update statement )
      statement
```

➔ **for** is a reserved word.

- Statement can be simple or compound.
- ***The initial statement, loop condition, and update statement*** are called for loop control statements.
  - initial statement usually initializes a variable (called the for **loop control**, or for **indexed**, **variable**)
- Statement is called the body of the loop.
- The parentheses and are part of the syntax.

For Looping (Repetition) Structure

- For Loop is probably the most popular looping instruction.
- The general form of the **for** statement is:

```
for ( initialise  counter ; test  counter ; increment  counter )
{
      do this ;
      and this ;
      and this ;
}
```

- The for allows us to specify three things about a loop in a single line:
  - Setting a loop counter to an initial value.
  - Testing the loop counter to detect whether its value reached the number of repetitions desired.
  - Increasing the value of loop counter each time the program segment within the loop has been executed.

For Looping (Repetition) Structure

EXAMPLE 5-7

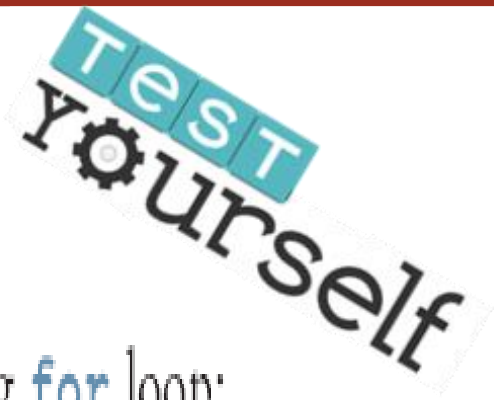
The following for loop prints the first 10 non negative integers:

```
for (i = 0; i < 10; i++)
    cout << i << " ";
cout << endl;
```

The output:

0 1 2 3 4 5 6 7 8 9

For Loop



- **Guess the OUTPUT**

A)

EXAMPLE 5-8

1. The following `for` loop outputs `Hello!` and a star (on separate lines) five times:

```
for (i = 1; i <= 5; i++)
{
    cout << "Hello!" << endl;
    cout << "*" << endl;
}
```

B)

2. Consider the following `for` loop:

```
for (i = 1; i <= 5; i++)
    cout << "Hello!" << endl;
    cout << "*" << endl;
```

For Loop

Answer

(A)

Hello!

*

Hello!

*

Hello!

*

Hello!

*

Hello!

*

(B)

Hello!

Hello!

Hello!

Hello!

Hello!

*

For Loop

➢ GUESS THE OUTPUT

EXAMPLE 5-10

You can count backward using a `for` loop if the `for` loop control expressions are set correctly.

For example, consider the following `for` loop:

```
for (i = 10; i >= 1; i--)
    cout << " " << i;
cout << endl;
```

The output:

10 9 8 7 6 5 4 3 2 1

For Loop

➢ GUESS THE OUTPUT

EXAMPLE 5-11

You can increment (or decrement) the loop control variable by any fixed number. In the following `for` loop, the variable is initialized to 1; at the end of the `for` loop, `i` is incremented by 2. This `for` loop outputs the first 10 positive odd integers.

```
for (i = 1; i <= 20; i = i + 2)
    cout << " " << i;
cout << endl;
```

The output:

1 3 5 7 9 11 13 15 17 19

For Loop

➢ Try Yourself

- Write a program that calculates and prints out the Average grade for 6 students .

For Loop

- Answer

```
int grade=0, sum=0;

for (int counter =1 ; counter<=6 ; counter ++ )
{
  cout <<" Enter grade \n";
  cin>>grade;
  sum += grade;
}

cout<<"The Average grades is " << float(sum)/6 <<"\n";
```

For Loop

For Looping (Repetition) Structure

```
int i= 1 ;
for ( ; i <= 10 ; i ++ )
      cout<< i;
```

```
int i;
for ( i = 0 ; i++ < 10 ; )
      cout<< i;
```

The output of all of them :

1 2 3 4 5 6 7 8 9 10

```
int i= 1 ;
for ( ; i<= 10 ; )
{
      cout<< i;
      i++;
}
```

```
int i;
for ( i = 0 ; ++ i < 10 ; )
      cout<< i;
```

For Loop

➢ Try Yourself

- Write a program that calculates the Factorial for any given positive number.
- Ex: **Factorial (5) = 5 * 4 * 3 * 2 * 1**

For Loop

- Answer

```
int number, factorial=1;
cout<<"Enter a positive number\n";
cin>> number;
if (number < 0 )
cout<<" Enter Positive Numbers only\n";
else
      for (int i= 1 ; i<=number ; i++)
            factorial = factorial * i;

cout<<" Factorial =" << factorial <<"\n";
```

Choosing the Right Looping Structure

- All three loops have their place in C++
  - If you know or can determine in advance the number of repetitions needed, the for loop is the correct choice
  - If you do not know and cannot determine in advance the number of repetitions needed, and it could be zero, use a while loop
  - If you do not know and cannot determine in advance the number of repetitions needed, and it is at least one, use a do…while loop

For Looping (Repetition) Structure

- The following is a legal for loop:

```
for( ; ; )
cout << "Hello" << endl;
```

!!!! It's legal but it will be Infinite loop.

- The following is a legal for loop:

```
for( ; ; )
{
      cout << "Hello" << endl;
      break;
}
```

!!!! It's legal but what is break statement?!!!

Break & Continue

break and continue statements

- break and continue alter the flow of control
- break statement is used for two purposes:
  - To exit early from a loop
    - Can eliminate the use of certain (flag) variables
- After the break statement executes, the program continues with the first statement after the structure

➔ **`break`** and **`continue`** are reserved words.

- continue is used in while, for, and do…while structures
  - When executed in a loop
    - It skips remaining statements and proceeds with the next iteration of the loop

Nested Loops

Nested Control Structures

To create the following pattern:

```
*

**

***

****

*****
```

```
int i,j;
for(i = 1; i <= 5 ; i++)
{
      for(j = 1; j <= i; j++)
            cout << "*";
      cout << endl;
}
```

Nested Loops

➢ GUESS THE OUTPUT

```
int i,j;
for(i = 5; i >= 1 ; i--)
{
      for(j = 1; j <= i; j++)
            cout << "*";
      cout << endl;
}
```

OUTPUT:

*****

****

***

**

*

Nested Loops

➢ Trv Yourself

- Write a program that calculates the Factorial for numbers from 1 to 10

Nested Loops

- Answer

```
int factorial;
for ( int number=1; number<=10 ; number++)
{
      factorial=1;
      for ( int i= 1 ; i<=number ; i++)
      {
            factorial = factorial * i;
      }
      cout<<" Factorila of " << number <<"=" << factorial <<"\n";
}
```

THANK YOU