CSC 128

TOPIC 4: REPETITION CONTROL STRUCTURE

By : MOHD SAIFULNIZAM ABU BAKAR

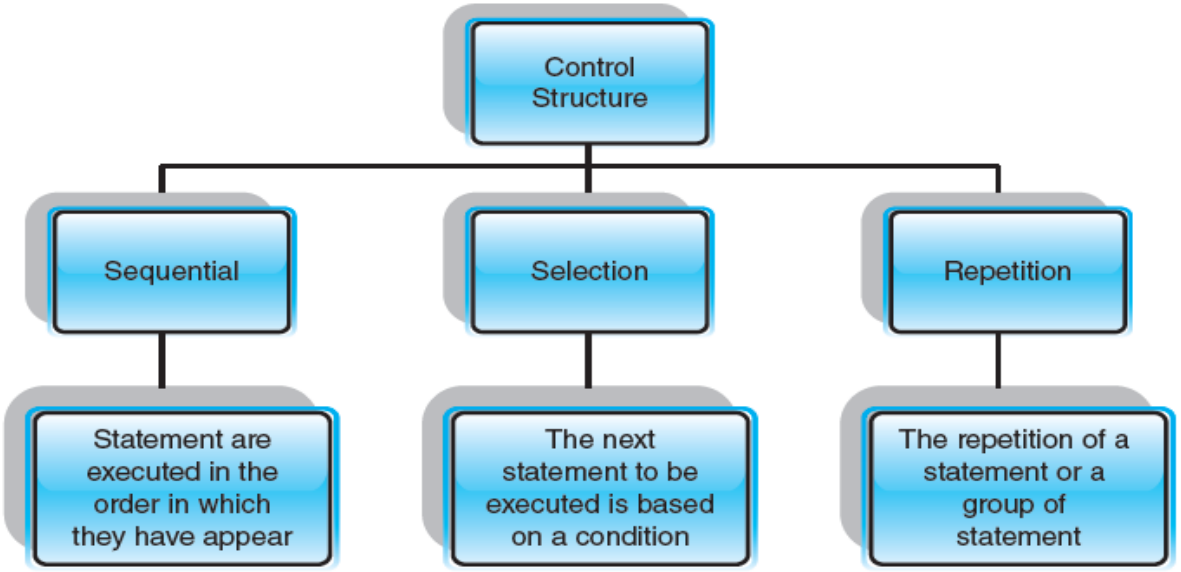
Learning Outcomes

At the end of this chapter, you should be able to:

- Differentiate between the three types of repetition control structure.
- Differentiate between counter-controlled structure, sentinel-controlled structure, and flag-controlled structure.
- Produce programs using repetition control structure.

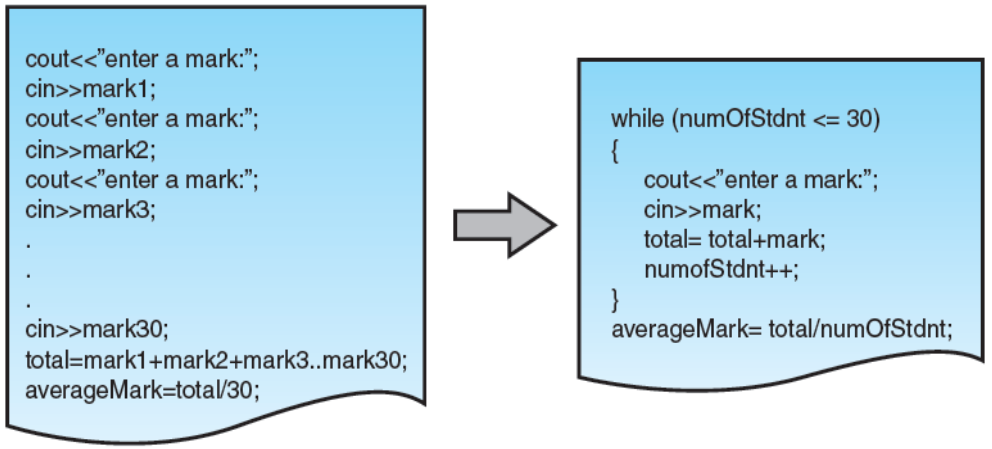
Control structure

- Before an algorithm is created, the three types of control structure should be understood first.
- A **control structure** is a pattern to control the flow of a program module.



Introduction

- There are three types of repetition control structure, and each of them can be written in another way as a set of instructions using sequential control structure and selection control structure.
- The three types of repetition in C++ are the `while` loop, the `do...while` loop and the `for` loop.
- These three types of statements are controlled by a control loop or Boolean test.



Sequential control structure

Repetition control structure

Requirements of Loop Control Variable (LCV) and Loop Condition (LC)

- **Loop control variable (LCV)** is a variable that is used to control the loop body, its initialization and execution.
- If the value loop control variable is not initialized, the program will assume any value.
- Repetition/iteration or loop is a control structure that allows a statement or group of statements to be executed repeatedly based on the Boolean test or loop condition.
- In order to write a program using repetition, three operations are needed—**initialization**, **evaluation** and **updating**.
- These three operations are required to make sure the program is repeated as we want it to, and to avoid an infinite loop.

3 OPERATION

1. **INITIALIZATION**
   - process of giving a variable an initial value

2. **EVALUATION**
   - verify how many loops will be done.

3. **UPDATING**
   - update the initial value, followed by an increase or decrease in the value, until the ending value is reached.

Operations: Initialization, Evaluation, Updating

- The first operation is **initialization** of the loop control variable (LCV).
- Initialization is the process of giving a variable an initial value. An initial value is important in the process of looping to count or sum the variable.
- Note that the initialization is the first value that will be evaluated. If the value is true, the loop will repeat the statement.
- Example: count = 0, start = 1, x = 10.

Operations: Initialization, Evaluation, Updating (cont.)

- The second operation is **evaluation**. It is used to verify how many loops will be done.
- For example: count <= 10, start < 30, x > 0.
- The program starts by testing the Boolean test.
- If the result of the Boolean test is true, the entire loop body is executed.
- Then, the process will be repeated as long as the condition is true.
- If the result of the Boolean test is false, the loop body will not be executed.
- The value true means run the loop body again and the value false means quit the loop.

Operations: Initialization, Evaluation, Updating (cont.)

- The third operation is **updating**.
- It is used to update the initial value, followed by an increase or decrease in the value, until the ending value is reached.
- For example: count++, start--, x+=5.

EXAMPLE 4.1

Initial value (Initialization)	Boolean Test (Evaluation)	Counter (Updating)	Remarks
sum = 1	sum<=10	sum++	The value of sum is given an initial value of 1. sum is then incremented by 1 ten times.
y=20	y>0	y--	The value of y is given an initial value of 10. y is then decremented by 1 twenty times.
x=0	x<=50	x+=5	The value of x is given an initial value of 0. x is then incremented by 5 ten times.

Repetition/Looping Control Structure

- There are three types of looping control in C++—counter-controlled structure, sentinel-controlled structure and flag-controlled structure.

1. A **counter-controlled structure** is a loop where we know how many times it will be executed.
2. A **sentinel-controlled structure** does not know how much data is to be read because it will be based on the value entered by the user.
3. A **flag-controlled structure** uses a `bool` (data type for true or false) variable to control the loop.

Repetition Using `while` Statement

- A **`while`** loop does not necessarily iterate a specified number of times. As long as its condition is true, a `while` loop will continue to iterate.
- A **`while`** loop is considered to be an indeterminate or indefinite loop because the number of times it will iterate can only be determined at run time.
- A `while` loop is also considered to be a top checking **(pre-test**) loop, since the control expression is located on the first line of code with the `while` keyword.
- If the control expression initially yields false, the loop will not iterate even once. If the condition is false, the loop is skipped. If the condition is true, the loop body is executed, and we then go back to testing the condition.

Repetition Using `while` Statement (cont.)

- The `while` statement has the form:

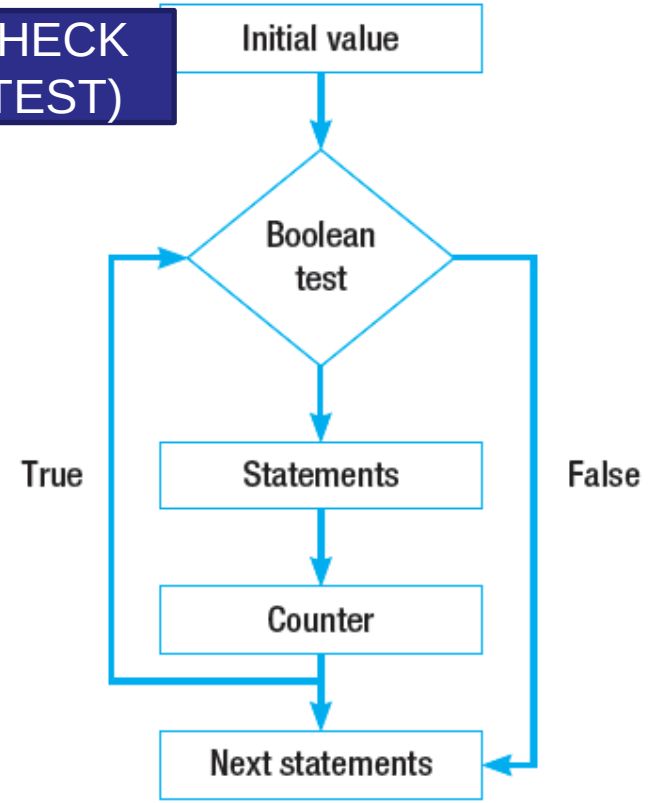
```
initial value;

while (condition/boolean test)

{
        statements;
        .. .. .. .. .. .. .. .;
        counter;

}
```

TOP CHECK (PRE TEST)



Repetition Using `while` Statement (cont.)

EXAMPLE 4.2 Shows a program to display the statement "I LOVE C++" for ten times. This example is a type of counter-controlled loop because we know how many times the statement will be repeated.

```
#include<iostream>
using namespace std;
int main ( )

{
        int i=1;                    //Initialization LCV
        while (i<=10)               //Evaluation LCV
        {
        cout<<"I LOVE C++ n";              //Statement to repeat
        i++;        //Updating LCV
}
return 0;
}
```

TOP CHECK

```
int i=1;                    //Initialization LCV
while (i<=10)               //Evaluation LCV
{
cout<<"I LOVE C++ n";       //Statement to repeat
i++;        //Updating LCV
```

Initialization	Evaluation	Updating	
i	**while (i<=10)**	**cout<< I LOVE C++ \n**	**i++ (i = i+ 1)**
1	(1<=10) TRUE	I LOVE C++	2 = 1 + 1
2	(2<=10) TRUE	I LOVE C++	3 = 2 + 1
3	(3<=10) TRUE	I LOVE C++	4= 3+1
4	(4<=10) TRUE	I LOVE C++	5 = 4 + 1
…	….	….	….
10	(10<=10) **TRUE**	I LOVE C++	11=10+1
11	(11<=10) **FALSE**		

counter-controlled structure

Repetition Using `while` Statement (cont.)

- Consider the following segment of code. What will be displayed?

EXAMPLE 4.3

```
//the purpose of this program is to calculate the total of 3 numbers
#include <iostream>
using namespace std;
int main()
{
      int value = 1;            //Initialization LCV
      int total = 0;
   while( value<=3)             //Evaluation LCV
   {
            total = total + value;
            value++;            //Updating LCV
   }
   cout<<"The total is " <<total<<endl;
   return 0;
   }
```

TOP CHECK

counter-controlled structure

```
   int value = 1;              //Initialization LCV
   int total = 0;
while( value<=3)               //Evaluation LCV
{
         total = total + value;
         value++;              //Updating LCV
}
cout<<"The total is " <<total<<endl;
```

Initialization		Evaluation	Updating		
value	total	while (value<=3)	total=total+value	value++ (value=value+1)	
1	0	TRUE	1 = 0+1	2 = 1+1	
2	1	TRUE	3 = 1+2	3 = 2+1	
3	3	TRUE	**6** = 3 + 3	**4** = 3+1	
4	6	**(4<=3) FALSE**			

Repetition Using `while` Statement (cont.)

- Tracing the output of the program in Example 4.3

C++ Statement	value	Result	total
while(value<=3)	1	TRUE	0
total = total+value;			1
total++;	2		
while(value<=3)	2	TRUE	
total = total+value;			3
total++;	3		
while(value<=3)	3	TRUE	
total = total+value;			6
total++;	4		
while(value<=3)	4	FALSE	

Repetition Using `while` Statement (cont.)

- In Example 4.4, a program is written to calculate the total price of items purchased.
- The program should ask for the quantity of items, item names and the price for each item.

counter-controlled structure

EXAMPLE 4.4

```
#include<iostream>
#include<string>
using namespace std;
int main( )
{
    int   numOfItem;      //to store number of   items
    int   i=1;      //the loop control variable   (initialization)
    string   itemName;      //to store item name
    double   price;          //to store item price
    double   total=0;        //to store the total price of item

    cout<<"Enter number of items :";
    cin>>numOfItem;             //Input number of item

    while (i<=numOfItem)      //Boolean Test
    {
        cout<<"Enter item name :";
        cin>>ws;
        getline(cin, itemName);   //Input item name
        cout<"Enter item cost: RM";
        cin>>price;               //Input item price
        total = total + price;   //Calculate the total price
        i++;                      //update the value
    }
    //to set the number to two decimal places
    cout. setf(ios: :fixed);
    cout. precision (2);

    //display the total price
    cout<<"The total price is : RM"<<total<<endl;
}
```

Repetition Using `while` Statement (cont.)

TOP CHECK

EXAMPLE 4.5

```
#include<iostream>
using namespace std;
int main()
{
      int sentinel=-999;
      int num,sum=0,count=0;

      cout<<"Enter a number ending with:"<<sentinel<<endl;
      cin>>num;

      while(num!=sentinel)
      {
         sum = sum + num;
         count++;
         cin>>num;
      }
      cout<<"The sum of "<<count<<" numbers is "<<sum<<endl;

      return 0;
}
```

sentinel-controlled structure

```
int sentinel=-999;
int num,sum=0,count=0;
while(num!=sentinel)
{
    sum = sum + num;
    count++;
    cin>>num;
}
cout<<"The sum of "<<count<<" numbers is "<<sum<<endl;
```

Initialization		Evaluation	Updating		
sentinel	num	while (num!=sentinel)	Sum = sum + num	Count++ (count=count + 1)	Cin>> num
-999	23	23!=-999 TRUE	23 = 0 + 23	1 = 0+1	12
-999	12	12!=-999 TRUE	35 = 23 + 12	2=1 + 1	56
-999	56	56!=-999 TRUE	**91** = 35 + 56	3 =2+ 1	-999
-999	-999	**-999!=-999 FALSE**			

The sum of 3 number is 91

Repetition Using `while` Statement (cont.)

EXAMPLE 4.6

```
#include<iostream>
using namespace std;
int main( )

{
      bool guess;
      guess = false;
      int value = 18;
      int num;

      while (!guess)
      {
            cout<"Enter an integer between 0 and 20:";
            cin>>num;
            if(num==value)
            {
                  cout<<"\nYou guessed a correct number.\n";
                  guess=true;
            }
            else if(num<value)
                  cout<<"\nYour guess is too low.\nGuess again!\n";
            else
            {
                  cout<<"\nYour guess is too high.";
                  cout<<"\nGuess again!\n";
            }
      }           return 0;
}
```

TOP CHECK

flag-controlled structure

Repetition Using `do...while` Statement

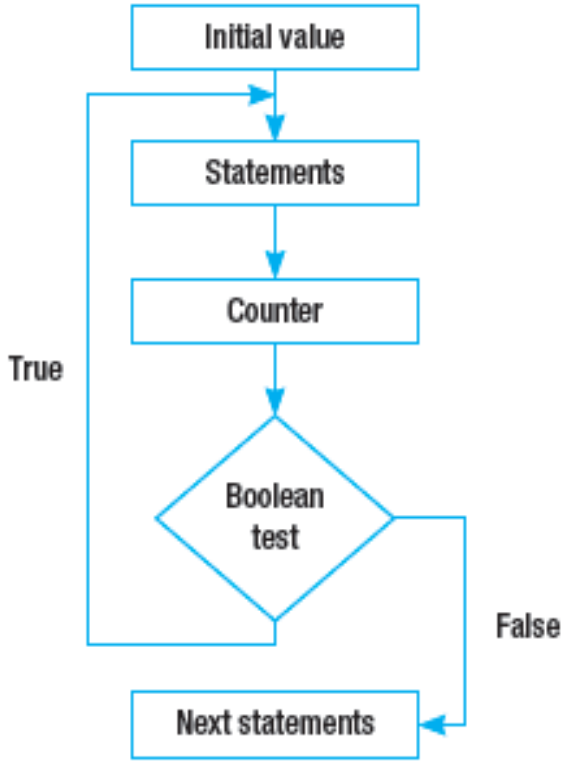
- A `do. . .while` statement is similar to the `while` loop, except that the evaluation operation occurs at the end of the loop.
- It is guaranteed that a `do. . .while` loop will iterate at least once because its condition is placed at the end of the loop.
- A `do. . . while` loop is considered to be a **bottom-checking** (**posttest**) loop, since the control expression is located after the body of the loop after the `while` keyword.
- A `do. . .while` loop is guaranteed to iterate at least once even if the condition evaluated yields false.
- It is useful for tasks that need to be executed once before a test is made to continue, such as a test that is dependent upon the results of the loop.

Repetition Using `do...while` Statement (cont.)

- The do. . .while statement has the form:

```
initial value;
do
{
        statements;
        counter;
}while (Boolean test);
```

BOTTOM-CHECKING (POSTTEST)



Repetition Using `do...while` Statement (cont.)

EXAMPLE 4.7

```
#include<iostream>
using namespace std;
int main()
{
      int i=1;//Initialization LCV
      do
      {
            cout<<"I'm a good student\n";//Statement to repeat
            i++;//Updating LCV
      }while (i<=10);//Evaluation LCV
      return 0;
}
```

BOTTOM-CHECKING (POSTTEST)

```
int i=1; //Initialization LCV
do
{
      cout<<"I'm a good student\n"; //Statement to repeat
      i++; //Updating LCV
}while (i<=10); //Evaluation LCV
```

Initialization	Updating		Evaluation
i	cout<< (I'm a good student)	i++ (i = i+ 1)	while (i<=10)
1	I'm a good student	2 = 1 + 1	(2<=10) TRUE
2	I'm a good student	3 = 2 + 1	(3<=10) TRUE
3	I'm a good student	4= 3+1	(4<=10) TRUE
4	I'm a good student	5 = 4 + 1	(5<=10) TRUE
…	….	….	….
9	I'm a good student	10=9+1	(10<=10) TRUE
10	I'm a good student	11=10+1	(11<=10) **FALSE**

counter-controlled structure

Repetition Using `do...while` Statement (cont.)

EXAMPLE 4.8

```
#include<iostream>
using namespace std;
int main( )
{
      int i=1;      //Initialization LCV
      do
      {
            cout<"I'm a good student\n";   //Statement to repeat
            i++;   //Updating LCV
      }while (i>=10);   //Evaluation LCV
      return 0;
}
```

BOTTOM-CHECKING (POSTTEST)

```
int i=1;      //Initialization LCV
do
{
      cout<"I'm a good student\n";   //Statement to repeat
      i++;   //Updating LCV
}while (i>=10);   //Evaluation LCV
return 0;
```

Initialization	Updating		Evaluation
i	**cout<< (I'm a good student)**	**i++ (i = i+ 1)**	**while (i>=10)**
1	I'm a good student	2 = 1 + 1	(2>=10) **FALSE**

counter-controlled structure

Repetition Using `do...while` Statement (cont.)

EXAMPLE 4.9

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
      int menu;
      int areaR, length, width;
      double areaT,base,height;
      double areaC,radius;
      const double PI = 3.142;

      do
      {
            cout<<"\n\tGeometry Calculator: \n";
            cout<<"1: Calculate the Area of Rectangle\n";
            cout<<"2: Calculate the Area of Triangle\n";
            cout<<"3: Calculate the Area of  Circle\n";
            cout<<"4: Quit\n";
            cout<<"Enter your choice:";
            cin>>menu;

          if(menu<1 || menu > 4)
            cout<<"\nPlease enter 1,2,3 or 4 :";
          else if (menu == 1)
            {
                  cout<<"Enter length and width of rectangle :";
                  cin>>length>>width;
                  areaR = length * width;
                  cout<<"\nThe area of Rectangle is :"<<areaR<<endl;
            }
            else if (menu==2)
            {
                  cout<<"Enter base and height of Triangle :";
                  cin>>base>>height;
                  areaT= 1.0/2.0 * base * height;
                  cout<<"\nThe area of Triangle is :"<<areaT<<endl;

            }
            else if (menu==3)
            {
                  cout<<"Enter radius of circle :";
                  cin>>radius;
                  areaC=PI * pow(radius,2);
                  cout<<"\nThe area of circle is :"<<areaC<<endl;
            }


      }while (menu != 4);


      return 0;
}
```

Repetition Using `for` Statement

- The `for` loop is a loop that will **repeatedly execute a block of statements a fixed number of times.**
- This type of loop is used when we **know the number of times** to repeat the statement.
- When we have a group of statements to repeat, enclose them with curly braces.
- The general form is as follows:

```
for (initial_value; boolean_test; counter)
{
        //loop body
}
```

counter-controlled structure

Repetition Using `for` Statement (cont.)

The following is the step-by-step guide on program execution for the `for` loop:

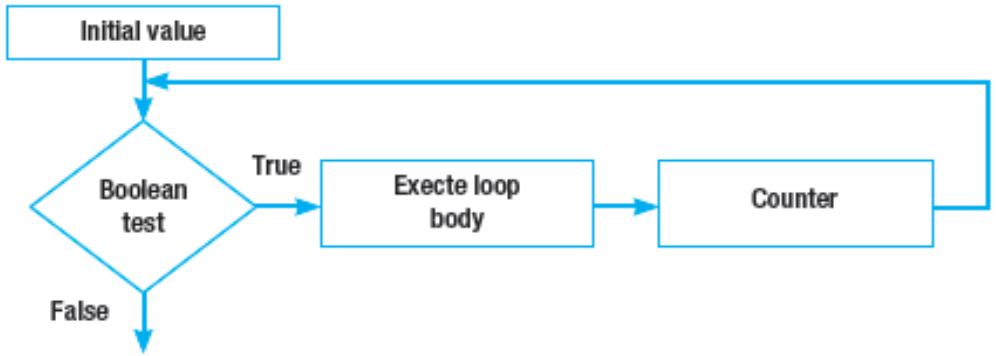
Step 1: `initial_value`
Sets up the initial value of the loop counter.

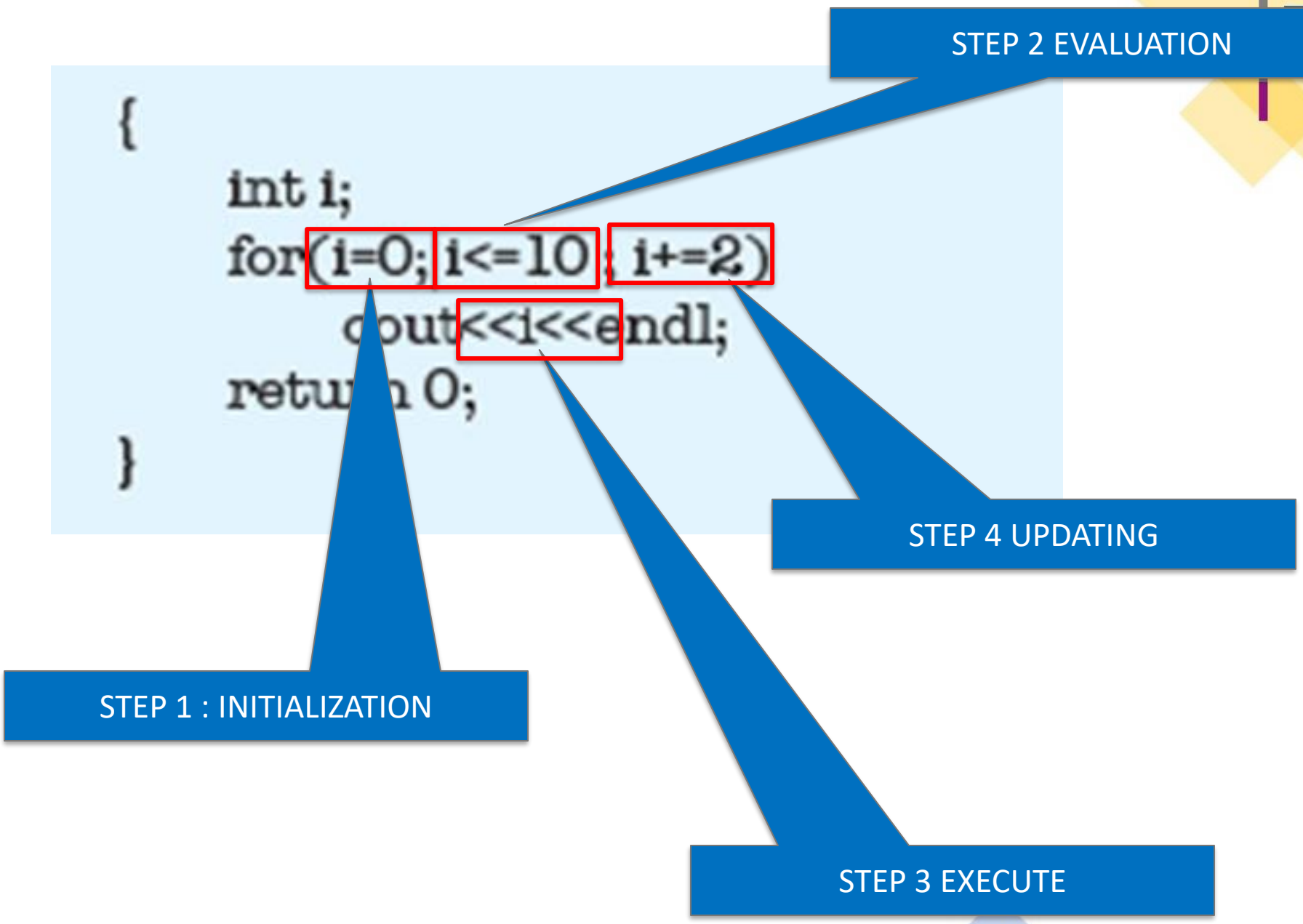
Step 2: `boolean_test`
The condition that is tested to determine if the loop should be executed again. If it is true, go to step 3.

Step 3: Execute the body of the loop

Step 4: `counter`
This describes how the initial value is changed on each execution of the loop then go back to step 2.

The flow of the program can be represented as a flow chart as follows:




{
int i;
for(i=0; i<=10 ; i+=2)
cout<<i<<endl;
retu n 0;
}
STEP 2 EVALUATION
STEP 4 UPDATING
STEP 1 : INITIALIZATION
STEP 3 EXECUTE

Repetition Using `for` Statement (cont.)

In Example 4.11, we want to display the even numbers from 0 to 10. We can do it in the following way:

EXAMPLE 4.11

```
#include <iostream>
using namespace std;
int main()

{
      int i;
      for(i=0; i<=10 ; i+=2)
            cout<<i<<endl;
      return 0;
}
```

output:

```
 0
 2
 4
 6
 8
10
```

In Example 4.11, we want to display the even numbers from 0 to 10. We can do it in the following way:

```
{
    int i;
    for(i=0; i<=10 ; i+=2)
        cout<<i<<endl;
    return 0;
}
```

counter-controlled structure

Initialization	Evaluation	Execute	Updating
i=0	**(i<=10)**	**cout<<i;**	**i+=2 (i=i+2)**
0	TRUE	0	2 = 0+2
2	TRUE	2	4 = 2 + 2
4	TRUE	4	6 = 4 + 2
6	TRUE	6	8 = 6 + 2
8	TRUE	8	10 = 8 + 2
10	TRUE	10	12 = 10+2
12	**(12<=10) FALSE**		

output:
0
2
4
6
8
10

Repetition Using `for` Statement (cont.)

In Example 4.12, we write a program to display the integer range between 1 to 10 in descending order.

EXAMPLE 4.12

```
#include <iostream>
using namespace std;
int main( )
{
      int i;
      for(i= 10; i>0 ; i--)

            cout<<i<<" ";
      return 0;
}
```

output:

```
10 9 8 7 6 5 4 3 2 1
```

In Example 4.12, we write a program to display the integer range between 1 to 10 in descending order.

```
int i;
for(i=10; i>0 ; i--)

      cout<<i<<" ";
return 0;
```

counter-controlled structure

Initialization	Evaluation	Execute	Updating
i=10	**(i>0)**	**cout<<i" ";**	**i-- (i=i-1)**
10	TRUE	10	9 = 10 - 1
9	TRUE	9	8 = 9 - 1
8	TRUE	8	7 = 8 - 1
7	TRUE	7	6 = 7 - 1
…	…	…	….
1	TRUE	1	0 = 1-1
0	**(0<0) FALSE**		

Repetition Using `for` Statement (cont.)

EXAMPLE 4.13

```
#include <iostream>
#include<math.h>
using namespace std;
int main()
{
      int j,number,result;
      cout<"Enter number :";
      cin>>number;
      cout<"Number t Number Squared n";
      cout<<"==========n";
      for(j=1; j<=number ; j++)
          {
                result=pow(j,2);
                cout<<j <<"tt"<< result <<"n";
          }

      return 0;
}
```

output:

```
Enter number: 7
Number          Number Squared
==============================
1                       1
2                       4
3                       9
4                       16
5                       25
6                       36
7                       49
```

```
for(j=1; j<=number ; j++)
    {
            result=pow(j,2);
            cout<<j <<"tt"<< result <<"n";
    }
```

counter-controlled structure

Initialization	Evaluation	Execute		Updating
j=1	**=7 (j<=7)**	result=pow(j,2)	**cout<<j"\t\t "result;**	**j++ (j=j+1)**
1	1<=7 TRUE	1 = pow(1,2)	1 1	2 = 1 + 1
2	2<=7 TRUE	4 = pow(2,2)	2 4	3 = 2 + 1
3	3<=7 TRUE	9 = pow(3,2)	3 9	4 = 3 + 1
4	4<=7 TRUE	16 = pow(4,2)	4 16	5 = 4+1
..	..	..	..	..
7	7<=7 TRUE	49 = pow(7,2)	7 49	8 = 7+1
8	**(8<=7) FALSE**			

The Difference Between `for`, `while` And `do...while` Loop

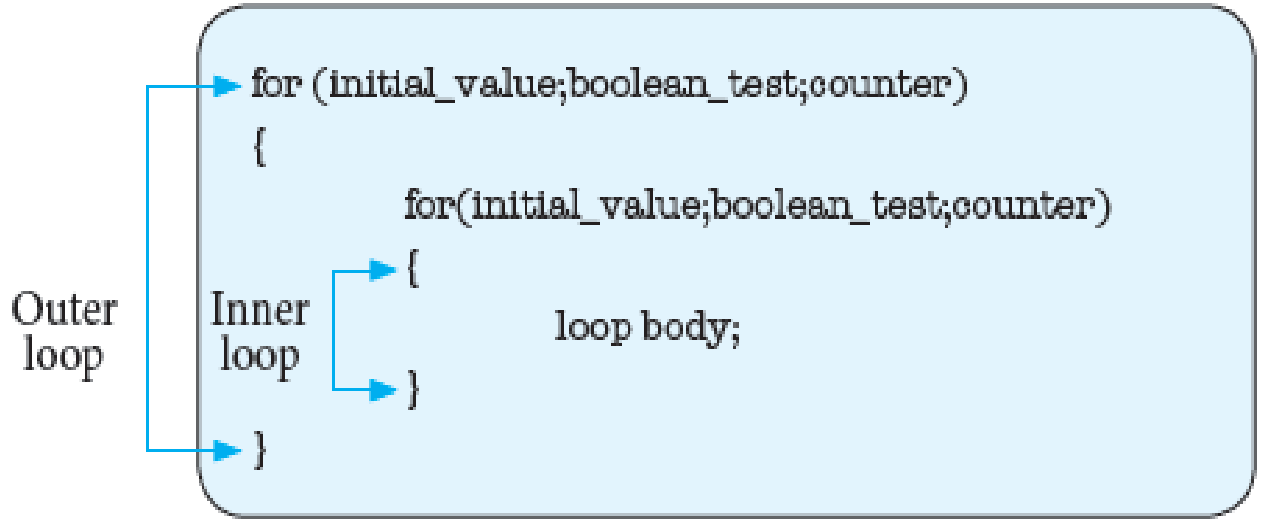
The three types of looping are used to repeat certain statements. The differences between for, while and do...while loop can been seen in Table 4.2.

Type	While loop	Do...while loop	For loop
Controlled loop	• Counter • Sentinel • Flag	• Counter • Sentinel • Flag	• Counter
Evaluation	***Pre test*** If the first condition evaluated yield false, the loop will never be executed	***Post test*** The statement is executed, then the condition is evaluated. At least the loop will be executed once	***Pre test*** If the first condition evaluated yields false, the loop will never be executed

Example	`int x=1;`
	`while( x<=5)`
	`{`
	`  cout<<x<<endl;`
	`  x++;`
	`}`
		`int x=1;`
		`do`
		`{`
		`  cout<<x<<endl;`
		`  x++;`
		`} while( x<=5);`
			`int x;`
			`for (x=1; x<=5; x++)`
			`{`
			`  cout<<x<<endl;`
			`}`

Nested Loop

- The placing of one loop inside the body of another loop is called nesting.
- When you “nest” two loops, the outer loop takes control of the number of complete repetitions of the inner loop.
- While all types of loops may be nested, the most commonly nested loops are `for` loops.



Nested Loop (cont.)

EXAMPLE 4.15

```
for (value1=0;value1<5;value1++)
{
        for(value2=0;value2<4;value2++)
        {
                cout<<value1<<""<<value2<<endl;
        }
}
```

Outer loop

Inner loop

Memory		Output
value1 (outer)	value2 (inner)	
0	0 1 2 3 4 end loop	0 0 0 1 0 2 0 3
1	0 1 2 3 4 end loop	1 0 1 1 1 2 1 3
2	0 1 2 3 4 end loop	2 0 2 1 2 2 2 3
3	0 1 2 3 4 end loop	3 0 3 1 3 2 3 3
4	0 1 2 3 4 end loop	4 0 4 1 4 2 4 3
5 end of loop		

Nested Loop (cont.)

- Example 4.16 is a program to display a multiplication table.

EXAMPLE 4.16

```
#include<iostream>
using namespace std;
int main ()
{
      int x, total,num;
      int sentinel = -1;
      cout<"Enter number:";
      cin>>num;
      while(num!=sentinel)
        {
        for(x=1;x<=12;x=x+1)
        {
           total=x*num;
                cout<<"\n"<<x<<"*"<<num<<"="<<total<<endl;
           }
        cout<<"\nEnter number:";
        cin>>num;
      }
        return 0;
}
```

Nested Loop (cont.)

- Example 4.17 is a program written to calculate the average marks of three tests for each student.

EXAMPLE 4.17

```
#include <iostream>
using namespace std;
int main()
{
    int marks, sum=0;
        char ans;
        double average;
        do
        {
            for(int i=1;i<=3;i++)
            {
                cout<<"Enter test "<<i<<" marks:";
                cin>>marks;
                sum=sum+marks;
            }

            average=sum/3;
            cout<"The average marks is:"<<average<<endl;
            cout<<"Do you want to repeat?:";
            cin>>ans;

    }while(ans=='Y' || ans=='y');
    cout<<"Thank you"<<endl;
  return 0;
}
```

Conclusion

- The repetition/looping control structure is important to repeat the same group of statements.
- There are three looping control structures in C++—the `while` loop, the `do. . .while` loop and the `for` loop.
- The difference between each of the three types of loop is how they control the repetition.
- These three types of loop require a loop control variable (LCV) to control the loop.
- There are three elements or operations needed in looping—initialization LCV, Evaluation LCV and updating LCV.
- The loop control in looping structure can be divided into three types—counter-controlled loop, sentinel-controlled loop and flag-controlled loop.

Conclusion

- The repetition/looping control structure is important to repeat the same group of statements.
- There are three looping control structures in C++—the `while` loop, the `do. . .while` loop and the `for` loop.
- The difference between each of the three types of loop is how they control the repetition.
- These three types of loop require a loop control variable (LCV) to control the loop.
- There are three elements or operations needed in looping—initialization LCV, Evaluation LCV and updating LCV.
- The loop control in looping structure can be divided into three types—counter-controlled loop, sentinel-controlled loop and flag-controlled loop.

Investment

- Deposit **RM1000 in bank saving**
- Bank Dividend **10%** each year on current balance
- How much money after 10 years ? With no transaction happen.

Annual Schedule

Annual Schedule | Monthly Schedule

	start principal	start balance	interest	end balance
1	$1,000.00	$1,000.00	$100.00	$1,100.00
2	$1,000.00	$1,100.00	$109.99	$1,210.00
3	$1,000.00	$1,210.00	$121.00	$1,331.00
4	$1,000.00	$1,331.00	$133.09	$1,464.10
5	$1,000.00	$1,464.10	$146.40	$1,610.51
6	$1,000.00	$1,610.51	$161.03	$1,771.56
7	$1,000.00	$1,771.56	$177.15	$1,948.72
8	$1,000.00	$1,948.72	$194.87	$2,143.59
9	$1,000.00	$2,143.59	$214.37	$2,357.95
10	$1,000.00	$2,357.95	$235.80	$2,593.74

Investment

- Deposit RM1000
- Dividend 10% each year on current balance
- How much money after 10 years?

Analysis - OPERATION

1. **Initialization – money = 1000, year = 0**
2. **Evaluation – year<=10**
3. **Updating - money = money + dividend**

   - year++ (year=year+1)

Investment

```
double money=1000.00,div;
int year=0;

while ( year<=10)
{
    div=money*0.1;
    money=money+div;
    year++;
}
```

Initialization

Evaluation

Updating

counter-controlled structure

Investment

Initialization		Evaluation	Updating		
money	year	while (year<=10)	div=money*0.1	money=money +div	year++ (year=year+1)
1000	0	TRUE	100	1100=1000+ 100	1=0+1
1100	1	TRUE	110.00	1,210.00	2
1,210.00	2	TRUE	121.00	1,331.00	3
1,331.00	3	TRUE	133.09	1,464.10	4
1,464.10	4	TRUE	146.40	1,610.51	5
…	…	….	….	….	….
2,357.95	10	TRUE	235.80	**2,593.74**	**11**
2,593.74	11	FALSE			