STQS1313 PENGATURCARAAN KOMPUTER. Nota 8.

Adapted from C++ Programming: From Problem Analysis to Program Design, 8th edition by D. S. Malik

CONTROL STRUCTURES 2 (REPETITION)

1) Let's start with a question. How do you write a C++ code that accepts the marks of 10 students, and then calculates the average of these numbers?
2) The most direct way could be to declare 10 variables to store those 10 numbers, for example, `num1, num2, num3, … , num10`, and using `cin` to get these numbers. For example:

```
#include<iostream>
using namespace std;
int main()
{
    double sum, average;
    double num1, num2, num3, num4, num5, num6, num7, num8, num9, num10;

    cout << "Please enter 10 numbers: ";
    cin >> num1>> num2>> num3>> num4>> num5>> num6>> num7>> num8>> num9>> num10;

    sum = num1+num2+num3+num4+num5+num6+num7+num8+num9+num10;
    average = sum/10;

    cout << "\nThe mean value is " << average << endl;

    return 0;
}
```

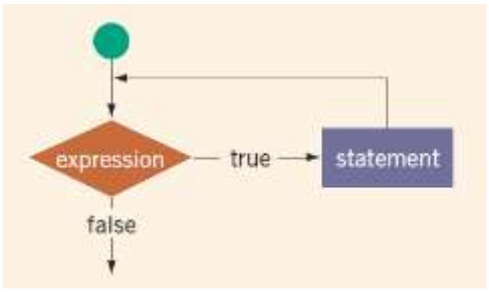
3) This way of writing a code will take quite a lot of time, especially when the input data is much larger than 10.
4) Imagine now that you want to find the average of 30 numbers. Here, based on the way the code is written, you would need to declare 20 more variables and hence add more lines in your code. Or even, if you want to find the average of 5 numbers, then the code will need to be modified.
5) There are many other situations in which it is necessary to repeat a set of statements.
6) In C++, the repetition can be done using a while loop, a for loop, or a do...while loop.

A) while loop

1) This loop takes the form

```
while (expression)
    statement
```

2) while is a reserved word.
3) The parentheses around the expression are part of the syntax (compulsory).
4) The statement can be either a simple or compound statement, and is called the **body** of the loop.
5) The expression acts as a decision maker, and is usually a logical expression.
6) The expression provides an entry condition to the loop.
7) Mechanism:
    - if expression evaluates to true, the statement executes.
    - the expression is then reevaluated. If it again evaluates to true, the statement executes again.
    - the statement continues to execute until the expression is no longer true.



8) A loop that continues to execute endlessly is called an **infinite loop** → usually we want to avoid.
9) This has enabled us to change the above code to something simpler, and can be used for any number of N:

```
#include<iostream>
using namespace std;
int main()
{
    double sum=0, average, num;
    int i=1, N=10;
    cout << "Please enter N numbers: ";
    while (i<=N)
    {
      cin >> num;
      sum = sum + num;
      i = i+1;
    }
    average=sum/N;
    cout << "\nThe mean value is " << average << endl;
    return 0;
}
```

Assume that you type 2, 1, 2, 3, 4, 2, 3, 4, 6, 7 as inputs to the code. The output is: The mean value is 3.4.

In the background, this is how the code processes the input:

Iteration	Value of i	Expression in while (i<=N)	Value of sum
1	i = 1	i <= 10 is true	sum = 0 + 2 = 2
2	i = 2	i <= 10 is true	sum = 2 + 1 = 3
3	i = 3	i <= 10 is true	sum = 3 + 2 = 5
4	i = 4	i <= 10 is true	sum = 5 + 3 = 8
5	i = 5	i <= 10 is true	sum = 8 + 4 = 12
6	i = 6	i <= 10 is true	sum = 12 + 2 = 14
7	i = 7	i <= 10 is true	sum = 14 + 3 = 17
8	i = 8	i <= 10 is true	sum = 17 + 4 = 21
9	i = 9	i <= 10 is true	sum = 21 + 6 = 27
10	i = 10	i <= 10 is true	sum = 27 + 7 = 34
11	i = 11	i <= 10 is false	The loop terminates

10) Consider another example here:

```
#include<iostream>
using namespace std;
int main()
{
    int i=0;
    while (i<=20)
    {
      cout << i << " ";
      i = i+5;
    }
    return 0;
}
```

The output is: 0 5 10 15 20
In the background, this is how the code works:

Iteration	Value of i	Expression in while (i<=N)	Value shown by cout
1	i = 0	i <= 20 is true	0
2	i = 5	i <= 20 is true	5
3	i = 10	i <= 20 is true	10
4	i = 12	i <= 20 is true	15
5	i = 20	i <= 20 is true	20
6	i = 25	i <= 20 is false	The loop terminates

11) Consider another example here:

```
#include<iostream>
using namespace std;
int main()
{
    int i=20;
    while (i>=0)
    {
      cout << i << " ";
      i = i-5;
    }
    return 0;
}
```

The output is: 20 15 10 5 0

In the background, this is how the code works:

Iteration	Value of i	Expression in while (i<=N)	Value shown by cout
1	i = 20	i >= 0 is true	20
2	i = 15	i >= 0 is true	15
3	i = 10	i >= 0 is true	10
4	i = 5	i >= 0 is true	5
5	i = 0	i >= 0 is true	0
6	i = -5	i >= 0 is false	The loop terminates

i) Counter-controlled while loops

1) The examples shown above are **counter-controlled** while loops, since the variable i (or can be any valid identifier name) acts as a counter. The value of i will be updated each time the loop executes, and the loop terminates when the value of i in the expression is larger or smaller certain values.
2) In this case, we know exactly the number of times the loop will be executed.
3) This is useful when we know the amount of input data available.
4) Let's consider another example below, where the user determines the number of time the loop will be executed.

```
#include<iostream>
using namespace std;
int main()
{
    double sum=0, average, num;
    int i=1, N;
    cout << "How many numbers do you want to insert: ";
    cin >> N;
    cout << "Please enter " << N << " numbers: ";
    while (i<=N)
    {
      cin >> num;
      sum = sum + num;
      i = i+1;
    }
    average=sum/N;
    cout << "\nThe sum is " << sum << endl;
    cout << "\nThe mean value is " << average << endl;
    return 0;
}
```

ii) Sentinel-Controlled while loops

1) You do not always know how many pieces of data (or entries) need to be read, but you may know that the last entry is a special value, called a **sentinel**, that will tell the loop to stop.
2) The while loop will continue to execute as long as the program has not read the sentinel.
3) This loop has the following form:

```
cin >> variable;
while (variable != sentinel)
{
    .
    .
    cin >> variable;
    .
    .
}
```

4) Consider the following code:

```
#include<iostream>
using namespace std;
const int SENTINEL = -1;
int main()
{
    double sum=0, average, num;
    int N=0;
    cout << "Please enter positive numbers, and hit -1 to finish: ";
    cin >> num;
    while (num!=SENTINEL)
    {
      sum = sum + num;
      N = N+1;
      cin >> num;
    }
    average=sum/N;
    cout << "\nThe sum is " << sum << endl;
    cout << "\nThe mean value is " << average << endl;
    return 0;
}
```

Assume that you type 4, 2, 3, 1, -1 as inputs to the code. This means the loop will stop once -1 in entered.

In the background this is what is happening.

Iteration	Value of num	Expression in while (i<=N)	Value in sum
1	num = 4	num != -1 is true	4
2	num = 2	num != -1 is true	6
3	num= 3	num != -1 is true	9
4	num = 1	num != -1 is true	10
5	num = -1	num != -1 is false	The loop terminates

iii) Flag-controlled and EOF-controlled while loops
This will not be covered in this course.

B) for loop
1) The for loop is a specialized form of the while loop.
2) Its primary purpose is to simplify the writing of counter-controlled loops.
3) For this reason, the for loop is typically called a counter or indexed for loop.
4) The general form of the for statement is:

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

5) The statement can be simple or compound statement.
6) Example:

```
#include<iostream>
using namespace std;
int main()
{
    double sum=0, average, num;
    int i, N;
    cout << "How many numbers do you want to insert: ";
    cin >> N;
    cout << "Please enter " << N << " numbers: ";
    for (i=1; i<=N; i=i+1)
    {
      cin >> num;
      sum = sum + num;
    }

    average=sum/N;
    cout << "\nThe sum is " << sum << endl;
    cout << "\nThe mean value is " << average << endl;
    return 0;
}
```

Here you can change the statement `i=i+1` to `i++` or `i+=1`.

7) Another example

```
#include<iostream>
using namespace std;
int main()
{
    int i=0;
    for (i=0; i<=20; i = i+5)
      cout << i << " ";
    return 0;
}
```

8) And another example:

```
#include<iostream>
using namespace std;
int main()
{
    int i=0;
    for (i=20; i>0; i = i-5)
      cout << i << " ";
    return 0;
}
```

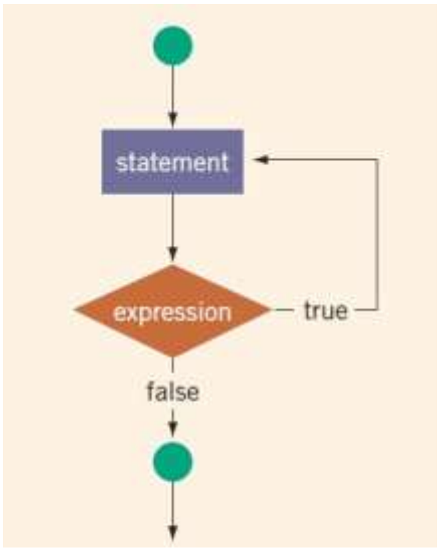
9) You can change the example in 7) to, for example, `i--` and etc to see that the decrement (and decrement) can comfortably be used in the for loop.

C) do…while loop

1) The general form is

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

2) The statement can be simple or compound statement.
3) The loop will run at least once.
4) The statement executes first, and then the expression is evaluated. If the expression evaluated to true, the statement executes again. As long as the expression is true, the statement will continuously execute.



5) Example:

```
#include<iostream>
using namespace std;
int main()
{
    double sum=0, average, num;
    int i=1, N;
    cout << "How many numbers do you want to insert: ";
    cin >> N;
    cout << "Please enter " << N << " numbers: ";
    do
    {
      cin >> num;
      sum = sum + num;
      i = i+1;
    } while (i<=N);

    average=sum/N;
    cout << "\nThe sum is " << sum << endl;
    cout << "\nThe mean value is " << average << endl;
    return 0;
}
```

whose output has no difference with those produced by the while loop.

6) Example: Comparing while and do...while loops

```
#include<iostream>
using namespace std;
int main()
{
    int i=11;
    while (i<=10)
    {
      cout << i << " ";
      i = i+1;
    }

    return 0;
}
```

and

```
#include<iostream>
using namespace std;
int main()
{
    int i=11;
    do
    {
      cout << i << " ";
      i = i+1;
    }while (i<=10);

    return 0;
}
```

Here the do...while produces an output, but not the while loop.

7) You can also use a sentinel in a do...while loop. For example:

```
#include<iostream>
using namespace std;
const int SENTINEL = -1;
int main()
{
    double sum=0, average, num;
    int N=0;
    cout << "Please enter positive numbers, and hit -1 to finish: ";
    do
    {
      cin >> num;
      sum = sum + num;
      N = N+1;
    }while (num!=SENTINEL);

    average=sum/N;
    cout << "\nThe sum is " << sum << endl;
    cout << "\nThe mean value is " << average << endl;
    return 0;
}
```

However, if you enter -1 as your first entry (which means there is no positive number), it will be saved in num. This is not correct. Hence for loop is preferable in this case (please try the code below).

```
#include<iostream>
using namespace std;
const int SENTINEL = -1;
int main()
{
    double sum=0, average, num;
    int N=0;
    cout << "Please enter positive numbers, and hit -1 to finish: ";
    cin >> num;
    while (num!=SENTINEL)
    {
      sum = sum + num;
      N = N+1;
      cin >> num;
    }

    average=sum/N;
    cout << "\nThe sum is " << sum << endl;
    cout << "\nThe mean value is " << average << endl;
    return 0;
}
```