جامعة
القاهرة الجديدة
التكنولوجية

NCT

NEW CAIRO
TECHNOLOGICAL
UNIVERSITY

TUE – The Technological Universities in Egypt

NCTU – New Cairo Technological University

Faculty of Industry and Energy Technology

Information Technology Department

Second-Year

Course: Programming Essentials in C++

Lecture 5

Presented by

Dr. Ghada Maher

Contents:

- C++ loop types (iteration statements)
  - The **while** Statement
  - Terminating A Loop
  - The **do….while** Statement
  - The **for** Statement
  - The **break** Statement
  - The **continue** Statement
  - The **goto** Statement
  - Generating Pseudo-random Numbers

C++ loop types (iteration statements)

- Iteration is the repetition of a statement or block of statements in a program. C++ has three iteration statements:
  - The while statement.
  - The do..while statement.
  - The for statement.
- Iteration statements are also called *loops* because of their cyclic nature.

The while statement

The syntax for the while statement is

while (*condition***)** *statement*;

EXAMPLE 1 Using a while Loop to Compute a Sum of Consecutive Integers

```
#include <iostream>
using namespace std;
int main()
{
//This program computes the sum 1 + 2 + 3 + ··· + n, for an input integer n:
int n,i=1;
cout << "Enter a positive integer: ";
cin >> n;
long sum=0;
while (i <= n)
sum += i++;
cout << "The sum of the first " << n << " integers is " << sum;
return 0;
}
```

i	sum
0	0
1	1
2	3
3	6
4	10
5	15
6	21
7	28
8	36

Run:
Enter a positive integer: **8**
The sum of the first 8 integers is 36

While Loop to Repeat a Computation

- EXAMPLE 2 Using a while Loop to Repeat a Computation

```
#include <iostream>
using namespace std;
int main()
{ double x;
cout << "Enter a positive number: ";
cin >> x;
while (x > 0)
{ cout << "sqrt(" << x << ") = " << sqrt(x) << endl;
cout << "Enter another positive number (or 0 to quit): ";
cin >> x;
return 0;
}
```

Run:
Enter a positive number: 49
sqrt(49) = 7
Enter another positive number (or 0 to quit): 3.14159
sqrt(3.14159) = 1.77245
Enter another positive number (or 0 to quit): 100000
sqrt(100000) = 316.228
Enter another positive number (or 0 to quit): 0

Terminating A Loop

- We have already seen how the break statement is used to control the switch statement. Also, The break statement is also used to control loops.

EXAMPLE 3 Using a break Statement to Terminate a Loop

```
#include <iostream>
using namespace std;
int main()
{ int n, i=1;
cout << "Enter a positive integer: ";
cin >> n;
long sum=0;
while (true)
{ if (i > n) break; // terminates the loop immediately
sum += i++;
}
cout << "The sum of the first " << n << " integers is " << sum;
return 0;
}
```

Run:
Enter a positive integer: **100**
The sum of the first 100 integers is 5050

THE do....while STATEMENT

- The syntax for the do..while statement is
  do *statement* **while (***condition***);**

Example 4:

```
#include <iostream>
using namespace std;
int main()
{
int i=6;
do
{
cout<<i;
i=i+1;
}
while(i<=5);
Cout<<“Hi”;
return 0;
}
```

Run:
6
Hi

Example 5

```
#include <iostream>
using namespace std;
int main()
{
int i=6;
while(i<=5)
{
cout<<i;
i=i+1;
}
return 0;
}
```

Run:
Hi

The for Statement

- The syntax for the for statement is

  for (*initialization***;** *condition***;** *update***)** *statement***;**

- the sequence of events that generate the iteration are:
  1. evaluate the *initialization* expression;
  2. if the value of the *condition* expression is false, terminate the loop;
  3. execute the *statement*;
  4. evaluate the *update* expression;
  5. repeat steps 2–4.

Example 6:


```
#include <iostream>
using namespace std;
int main()
{ int n;
cout << "Enter a positive integer: ";
cin >> n;
long sum=0;
for (int i=1; i <= n; i++)
sum += i;
cout << "The sum of the first " << n << "
integers is " << sum;
return 0;
}
```

Run:
Enter a positive integer: **8**
The sum of the first 8 integers is 36

Example 7: More than One Control Variable in a for Loop


```
#include <iostream>
using namespace std;
int main()
{ for (int m=95, n=11; m%n > 0; m -= 3, n++)
cout << m << "%" << n << " = " << m%n << endl;
return 0;
}
```

Run:
95%11 = 7
92%12 = 8
89%13 = 11
86%14 = 2
83%15 = 8

Example 8: Nesting for Loops

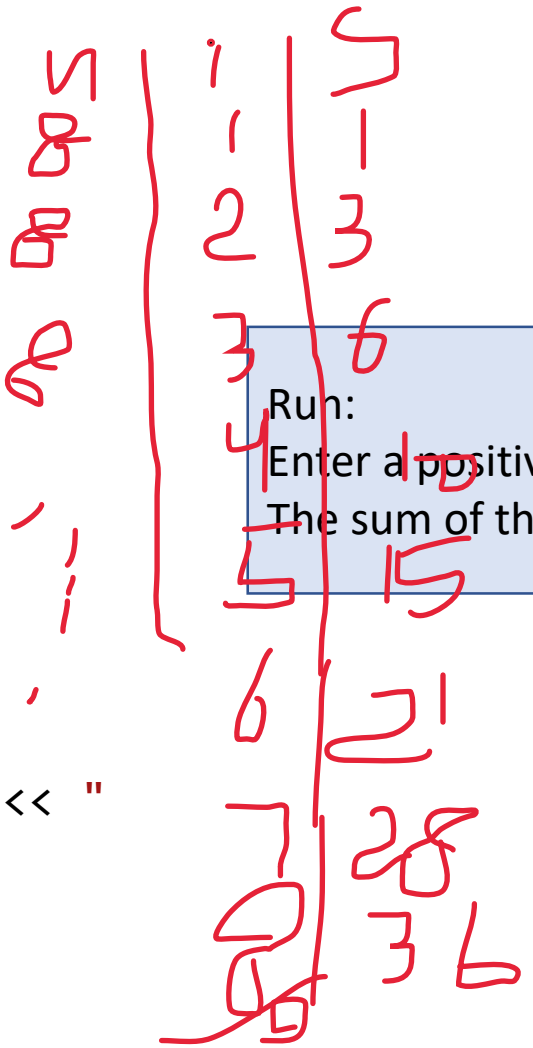

```
//This program prints a multiplication table:
#include <iomanip> // defines setw()
#include <iostream> // defines cout
using namespace std;
int main()
{
for (int x=1; x <= 12; x++)
{ for (int y=1; y <= 12; y++)
cout << setw(4) << x*y;
cout << endl;
}
return 0;
}
```

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

Example 9: Using a break Statement to Terminate a Loop

```
#include <iostream> // defines cout
using namespace std;

int main()
{ int n,i=1;
cout << "Enter a positive integer: ";
cin >> n;
long sum=0;
while (true)
{
if (i > n) break;
sum += i++;
}
cout << "The sum of the first " << n << "
integers is " << sum;
return 0;
}
```



Run:
Enter a positive integer: 8
The sum of the first 8 integers is 36

Example 10: Using a break Statement wit Nested Loops


```
#include <iostream> // defines cout
using namespace std;
int main()
{ for (int x=1; x <= 12; x++)
{ for (int y=1; y <= 12; y++)
if (y > x) break;
else cout << setw(4) << x*y;
cout << endl;
}
return 0;
}
```

```
 1
 2   4
 3   6   9
 4   8  12  16
 5  10  15  20  25
 6  12  18  24  30  36
 7  14  21  28  35  42  49
 8  16  24  32  40  48  56  64
 9  18  27  36  45  54  63  72  81
10  20  30  40  50  60  70  80  90 100
11  22  33  44  55  66  77  88  99 110 121
12  24  36  48  60  72  84  96 108 120 132 144
```

The continue statement

The **break** statement skips the rest of the statements in the loop's block, jumping immediately to the next statement outside of the loop. The **continue** statement is similar. It also skips the rest of the statements in the loop's block, but instead of terminating the loop, it transfers execution to the next iteration of the loop. It continues the loop after skipping the remaining statements in its current iteration.

Example 11: Using continue and break Statements


```
#include <iostream> // defines cout
using namespace std;
int main()
{ int n;
for (;;)
{ cout << "Enter int: "; cin >> n;
if (n%2 == 0) continue;
if (n%3 == 0) break;
cout << "\tBottom of loop.\n";
}
cout << "\tOutside of loop.\n";
return 0;
}
```

Run:
Enter int: 7
Bottom of loop.
Enter int: 4
Enter int: 9
Outside of loop.

The goto statement

- The **goto** statement is another kind of jump statement. Its destination is specified by a label within the statement. Example 12:

```
#include <iostream>
using namespace std;
int main()
{
    const int N=5;
    for (int i=0; i<N; i++)
    {
        for (int j=0; j<N; j++)
        {
            for (int k=0; k<N; k++)
                if (i+j+k>N) goto esc;
                else cout << i+j+k << " ";
                cout << "* ";
        }
esc: cout << "." << endl; // inside the i loop,outside the j loop
    }
return 0;
}
```


```
Run:
0 1 2 3 4 * 1 2 3 4 5 * 2 3 4 5 .
1 2 3 4 5 * 2 3 4 5 .
2 3 4 5 .
3 4 5 .
4 5 .
```

Example 13: Example : Write a program that prints out the following

*

**

***

****

*****

```cpp
#include <iostream>
using namespace std;
int main()
{
for (int i =1 ; i <= 5;i++)
{
for (int j = 1 ; j<= i;j++)
cout<< "*" ;
cout<< endl ;
}
return 0;
}
```

References:

John R. Hubbard, "*Schaum's outline of* theory and problems of programming with C++", Second Edition, Schaum's Outline Series, McGRAW-HILL, *New York San Francisco Washington.*