while ←

do – while

for ←

Fundamentals of Programming CS-114

Lecture 7

Break Statement Revisited

- **break statement can be used with Switch or any of the 3 looping structures**
- **it causes an immediate exit from the Switch, while, do-while, or for statement in which it appears**
- **if the break is inside nested structures, control exits only the innermost structure containing it**

Continue Statement

- **is valid only within loops**
- **terminates the current loop iteration, but not the entire loop**
- **in a for or while, continue causes the rest of the body statement to be skipped--in a for statement, the update is done**
- **in a do-while, the exit condition is tested, and if true, the next loop iteration is begun**

The break Statement

```
int j = 40;
while (j < 80){
    j += 10; // j=50,60,70
    if (j == 70)
        break;
    cout << "j is " << j<< '\n';
}
cout << "We are out of the loop as j=70.\n";
//see useBreak1.cpp
```

```
j is 50
j is 60
We are out of the loop as j=70.
```

The continue Statement

```
int j = 40;
while (j < 80){
    j += 10; //50 ,60, 70 ,80
   if (j == 70)
   continue; //skips the 70
   cout << "j is " << j<< '\n';
}//see UseContinue1.cpp
cout << "We are out of the loop" << endl;
```

```
j is 50
j is 60
j is 80

We are out of the loop.
```

break *and* continue

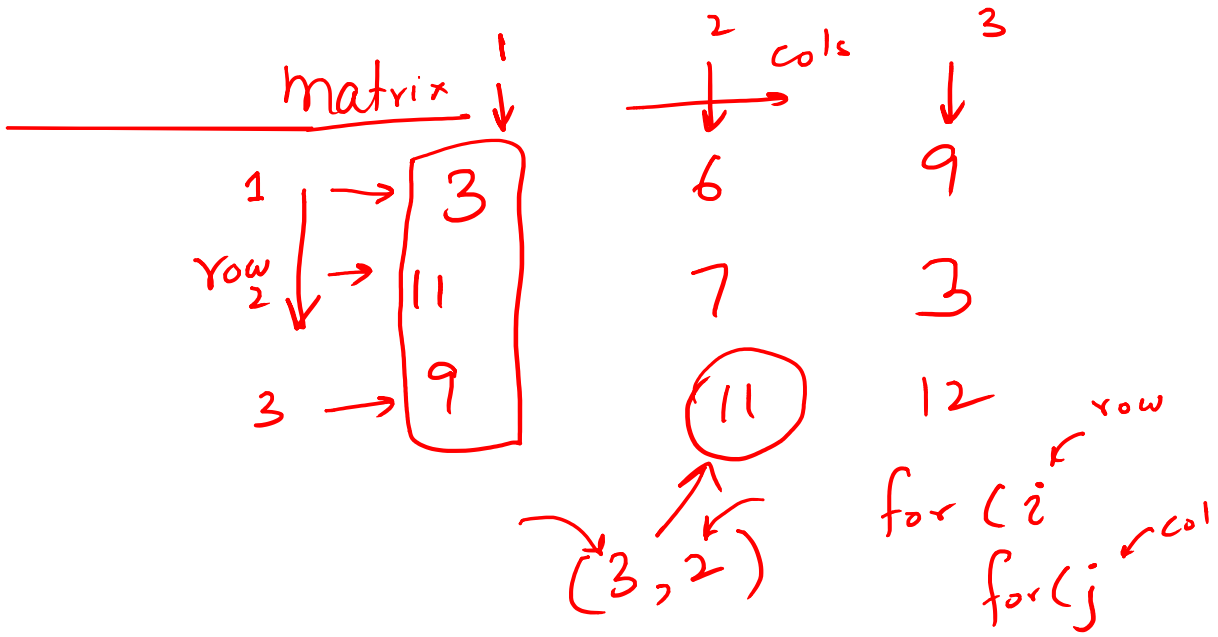
```
while ( - - - )
{
   statement 1;
   if( - - - )
        continue
   statement 2;
}
statement 3;
```

```
while ( - - - )
{
   statement 1;
   if( - - - )
        break
   statement 2;
}
statement 3;
```

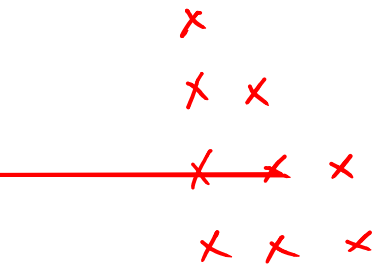
Nested Control Structures

- Suppose we want to create the following pattern

```
*
**
***
****
*****
```



- In the first line, we want to print one star, in the second line two stars and so on



```
int i,j,k,l;
while (i<1)
{
    cout << "*\n";
    while (j<2)
        cout << "*";
    while(k<3)
        cout<<"*
```

```
for (int i=0; i<5; i++)
{
    for (int j=0; j<i; j++)
    {
        cout << "*";
    }
    cout << "\n";
}
```

i=0
j= .....
i=1
j= ....
i=0
*
i=1
* *

→ i = 0

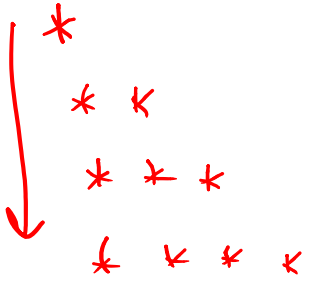
***

Conditional role

```
while (i<3)
{cout << "***\n";
while( i < 2)
{ cout << "**\n";
while ( i < 1)

        cout<< * \n";
        i++
}

i++
}
```



Nested Control Structures (continued)

- Since five lines are to be printed, we start with the following for statement

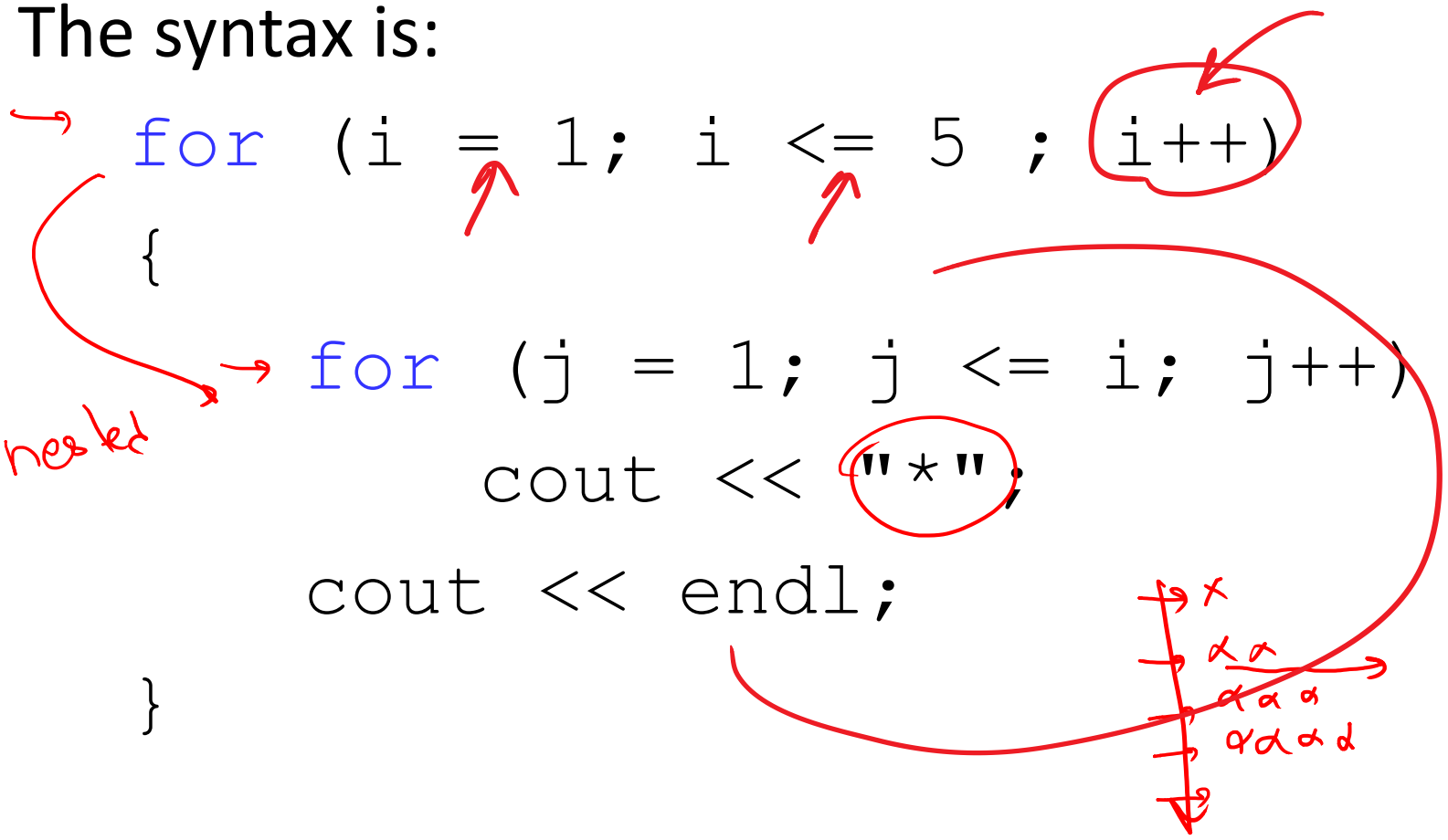
```
for (i = 1; i <= 5 ; i++)
```

- The value of `i` in the first iteration is `1`, in the second iteration it is `2`, and so on
- Can use the value of `i` as limit condition in another for loop nested within this loop to control the number of starts in a line

Nested Control Structures (continued)

- The syntax is:

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



Nested loops

- A loop can be nested inside of another loop. C++ allows at least 256 levels of nesting.

```
#include <iostream>
using namespace std;
int main()
{
int size;
cout << "How large a triangle do you want? ";
cin >> size;
for (int r = 1; r <= size; r++)
{
    for (int c = 1; c <= r; c++)
     {
       cout << "*";
     }
cout << endl;
}
return 0;
}
```

The result when the entered number is 7:

```
*
**
***
****
*****
******
*******
```

Nested Control Structures (continued)

- What pattern does the code produce if we replace the first for statement with the following?

```
for (i = 5; i >= 1; i--)
```

- Answer:

```
*****
****
***
**
*
```

Which Loop To Use?

- Choose the type of loop late in the design process
  - First design the loop using pseudocode
  - Translate the pseudocode into C++
  - The translation generally makes the choice of an appropriate loop clear
  - While-loops are used for all other loops when there might be occassions when the loop should not run
  - Do-while loops are used for all other loops when the loop must always run at least once

Choosing a for-loop

- for-loops are typically selected when doing numeric calculations, especially when using a variable changed by equal amounts each time the loop iterates

Choosing a while-loop

- A while-loop is typically used

  - When a for-loop is not appropriate

  - When there are circumstances for which the loop body should not be executed at all

Choosing a do-while Loop

- A do-while-loop is typically used

  - When a for-loop is not appropriate

  - When the loop body must be executed at least once