AMA2222 Principles of Programming

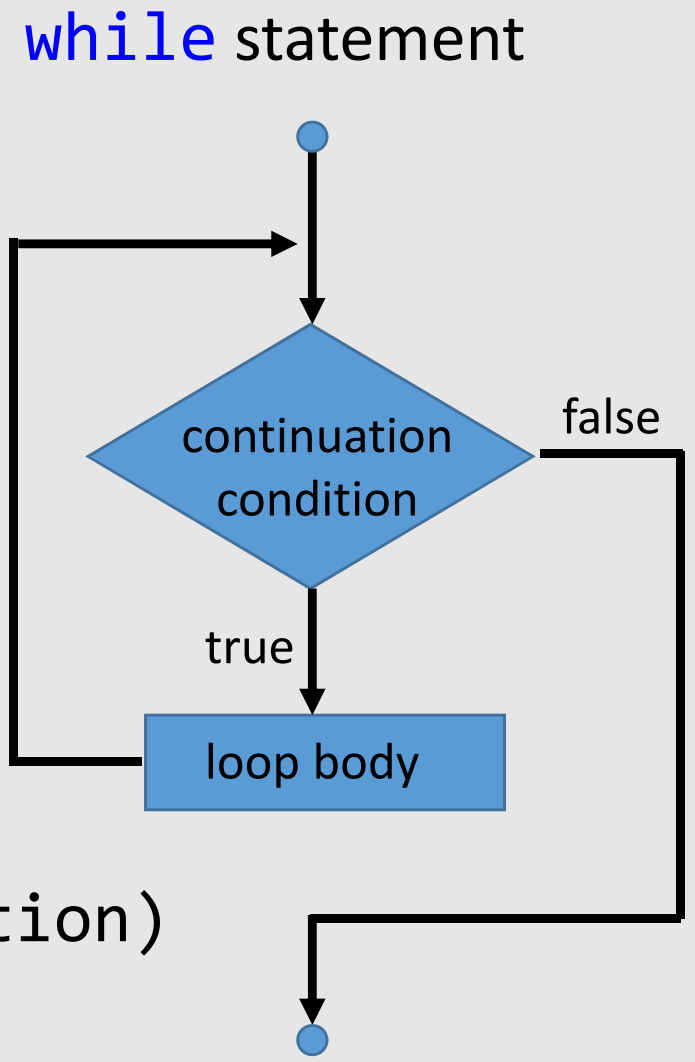
Leung Man Kin, Adam
Instructor

adam.leung@polyu.edu.hk
TU720

Chapter 3: Control statement II - looping
while loop, do-while loop, for loop,
file processing, nested loops

While loop

A repetition statement specifies that a program should repeat an action while some conditions remain true. The statements contained in the `while` repetition statement constitutes the body of the `while`, which can be a single statement or a block.



Syntax for while loop:

```
while (loop-continuation-condition)
{
   statement(s); // loop body
}
```

Be careful!

If the loop-continuation-condition remains true all the time, the loop body will be executed repeatedly without stop! You can use ctrl+c to kill the process.

A common way of controlling the number of iterations of the loop is by setting up a **counter**. This repetition is called **counter-controlled repetition**. It is also called **definite repetition** because the number of repetitions is known before the loop begins executing.

Example program 3.1

```
 1 // 3.1 While Loop
 2 #include <iostream>
 3 using namespace std;
 4 int main()
 5 {
 6          int i = 0;
 7          while (i < 10)
 8          {
 9                    cout << "The counter is " << i << endl;
10                    i++;
11          }
12          return 0;
13 }
```

Result:

```
The counter is 0
The counter is 1
The counter is 2
The counter is 3
The counter is 4
The counter is 5
The counter is 6
The counter is 7
The counter is 8
The counter is 9
```

Notice that `i++;` is equivalent to `i = i + 1;`

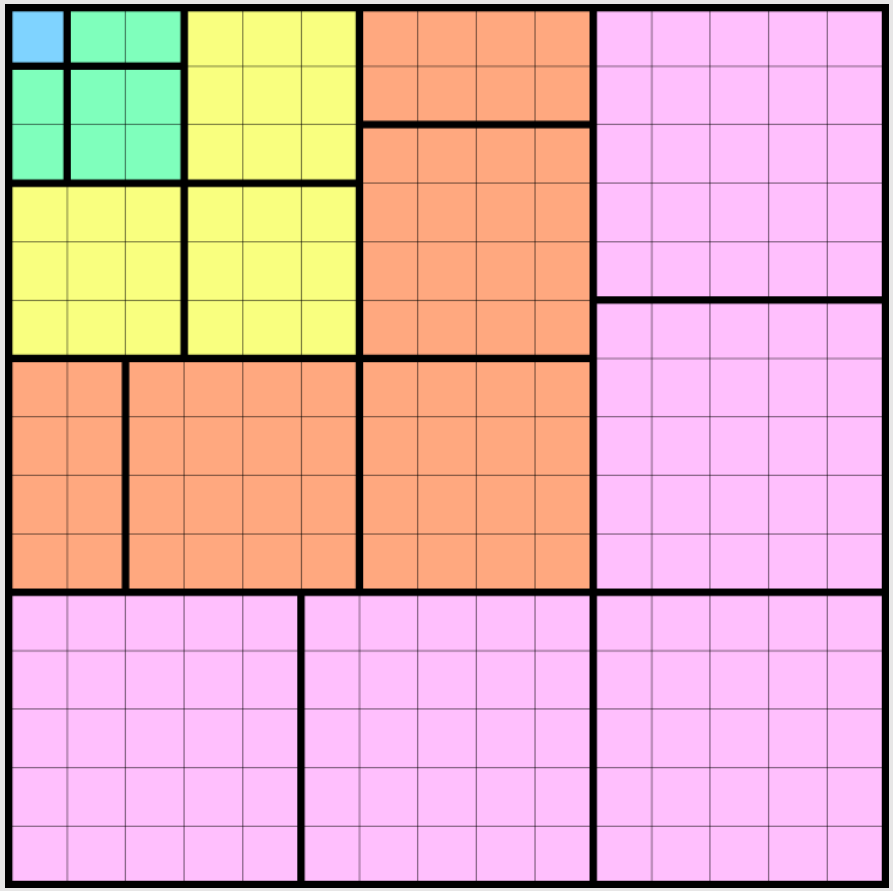
When the counter i is increased to 10, the loop-continuation-condition is not fulfilled. Therefore the loop ends.

Example program 3.2

How to sum up the cubes i.e.
$1^3 + 2^3 + 3^3 + \cdots + n^3$?

Mathematically, we can prove:
$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$

Now try to write a program that computes the series directly.



```
1 // 3.2a While Loop
2 #include <iostream>
3 #include <cmath>
4 using namespace std;
5 int main()
6 {
7           int n;
8           cout << "Enter the value of n " << endl;
9           cin >> n;
10          int sum = 0;
11          int i = 1;
12          while (i <= n)
13          {
14                    sum = sum + pow(i,3);
15                    i++;
16          }
17          cout << "The sum of cubes is " << sum << endl;
18          return 0;
19 }
```

n represents the total number of terms. i is the counter that goes from 1 to n. sum was initially set to 0 and added with each i^3.

A repetition is called **indefinite repetition** when the number of repetitions is not known in advance.

Example program 3.3

Consider the infinite series $1^3 + 2^3 + 3^3 + \cdots$. Starting from which term will the partial sum $1^3 + 2^3 + 3^3 + \cdots + n^3$ exceed a given value?

For example, the given value is 1000. We can observe that
$1^3 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3 + 7^3 = 784$
$1^3 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3 + 7^3 + 8^3 = 1296$
So the partial sum will exceed 1000 starting from the 8-th term.

```cpp
// 3.3 indefinite while Loop
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
        int x, sum = 0;
        cout << "Enter a number: ";
        cin >> x;
        int i = 0;
        while (sum <= x)
        {
                i++;
                sum = sum + pow(i,3);
        }
        cout << "Starting from the " << i << "-th term";
        return 0;
}
```

n represents the total number of terms. i is the counter that goes from 1 to n. sum was initially set to 0 and added with each i^3.

A special value called a **sentinel value** (also called a signal value, a dummy value or a flag value) can be used to indicate "end of data entry." The sentinel value must be chosen so that it's not confused with an acceptable input value.
Example program 3.4a

```
1  // 3.4a While Loop (with sentinel value)
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          cout << "Enter an integer (the input ends if it is 0): ";
7          int data;
8          cin >> data;
9          int sum = 0;
10         while (data != 0)
11         {
12                 sum += data;
13                 cout << "Enter an integer (the input ends if it is 0): ";
14                 cin >> data;
15         }
16         cout << "The sum is " << sum << endl;
17         return 0;
18 }
```

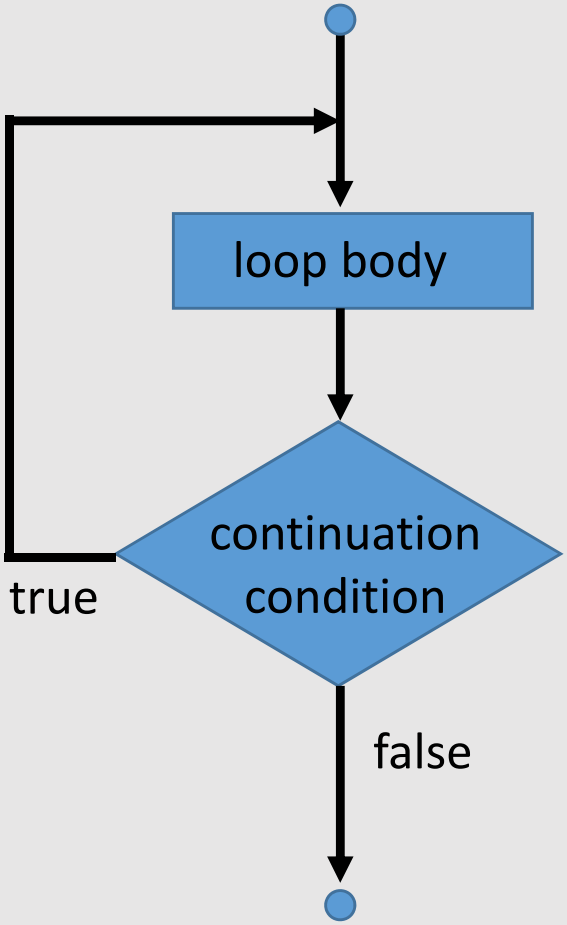
Do-while loop

A `do-while` loop is similar to a `while` loop except that it executes the loop body first and then checks the loop continuation condition.

Syntax for while loop:

```
do
{
    statement(s); // loop body
} while (loop-continuation-condition);
```

`do-while` statement



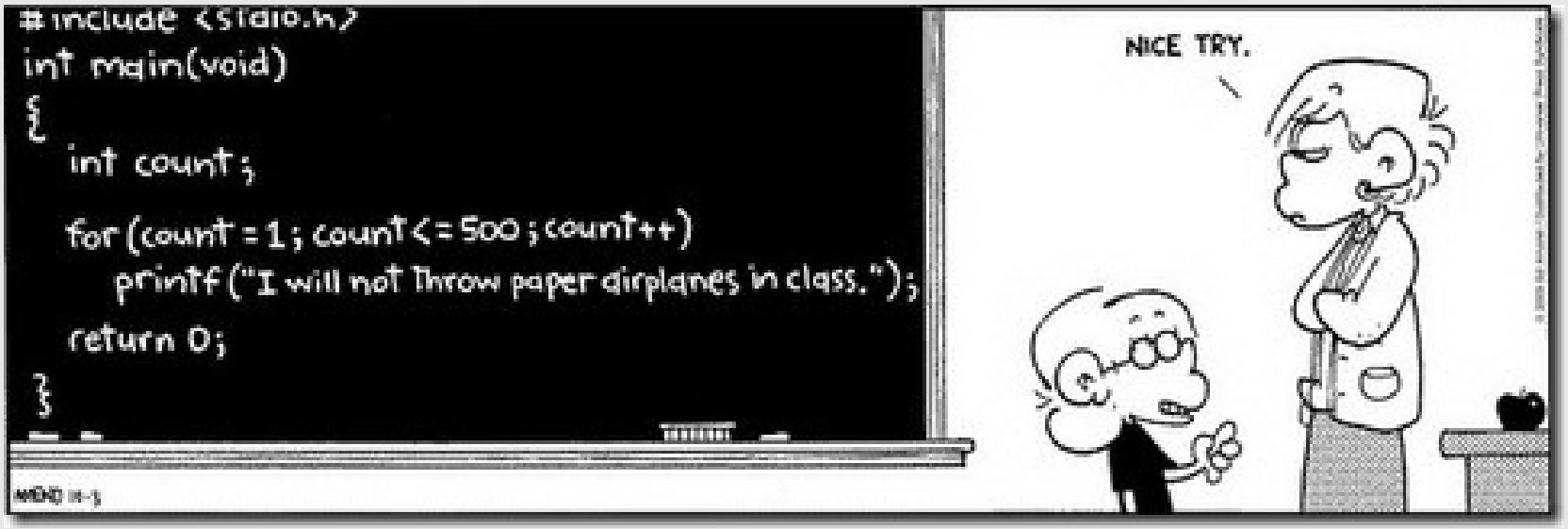
In the previous example, we will at least ask the user to input for once. Let's try to rewrite the program using do-while loop.

Example program 3.4b

```
 1 // 4.4b do-while loop
 2 #include <iostream>
 3 using namespace std;
 4 int main()
 5 {
 6           int sum = 0;
 7           int data = 0;
 8           do
 9           {
10                     sum += data;
11                     cout << "Enter an integer (the input ends if it is 0): ";
12                     cin >> data;
13           } while (data != 0);
14           cout << "The sum is " << sum << endl;
15           return 0;
16 }
```

For loop

A `for` loop is similar to a counter-controlled `while` loop. It executes the statements inside with a repetition of an exactly number of times according to a predefined counter.



```
//Syntax for for loop:
for (int i = initialValue; i < stopValue; i++)
{
    statement(s); // loop body
}

//An equivalent counter-controlled while loop:
int i = initialValue;
while (i < stopValue)
{
    statement(s); // loop body
    i++;
}
```

** notice that the i < stopValue can be replaced by other Boolean condition, the i++ can be replaced by other operations.

The last stored value of i is called exit value. However, it is a local variable and will be destroyed once the loop is done.

Classwork exercise 3.1

For how many times will the codes below print "Hello" ? For what values of i the execution was made? What is the exiting value of i?

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

0,1,2,3,4,5,6,7,8,9 (10 times)
exit value: 10

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

0,2,4,6,8,10 (6 times)
exit value: 12

```
for (int i=10; i>=1 i--)
{
    cout << "Hello" << endl;
}
```

10,9,8,7,6,5,4,3,2,1 (10 times)
exit value: 0

```
for (int i=1; i<=10; i+=3)
{
    cout << "Hello" << endl;
}
```

1,4,7,10 (4 times)
exit value: 13

Revisit example program 3.2 (sum of cubes) using for loop.

```
 1 // 3.2b For Loop
 2 #include <iostream>
 3 #include <cmath>
 4 using namespace std;
 5 int main()
 6 {
 7          int n;
 8          cout << "Enter the value of n " << endl;
 9          cin >> n;
10          int sum = 0;
11          for (int i = 1; i <= n; i++)
12          {
13                    sum = sum + pow(i,3);
14          }
15          cout << "The sum of cubes is " << sum << endl;
16          return 0;
17 }
```

Example program 3.5a Display all factors of an input integer.

```
 1 // 3.5a Factors
 2 #include <iostream>
 3 using namespace std;
 4 int main()
 5 {
 6           int n;
 7           cout << "Enter the value of n: ";
 8           cin >> n;
 9           cout << "The factors of " << n << " are:" << endl;
10           for (int i = 1; i <= n; i++)
11           {
12                     if (n % i == 0)
13                     {
14                               cout << i << endl;
15                     }
16           }
17           return 0;
18 }
```

Example program 3.5b Show if an integer is prime number or not.

The idea is to check is n has any factor rather than 1 and n itself. Once that happens, n is not a prime number. Otherwise n is prime.

```
 1 // 3.5b Prime
 2 #include <iostream>
 3 using namespace std;
 4 int main()
 5 {
 6          int n;
 7          bool flag = true;
 8          cout << "Enter the value of n: ";
 9          cin >> n;
10          for (int i = 2; i < n; i++)
11                    if (n % i == 0)
12                              flag = false;
13          if (flag)
14                    cout << n << " is a prime number." << endl;
15          else      cout << n << " is not a prime number." << endl;
16          return 0;
17 }
```

Break and continue

In a for-loop or while-loop, we can use `break` to terminate the loop and `continue` to terminate an iteration.

When `break` is executed in a loop, it causes immediate exit from that loop. Common uses of the break statement are to escape early from a loop and to skip the remaining iterations or to avoid infinite looping.

When `continue` is executed in a loop, the remaining line in the current iteration of the loop are skipped and proceeds with the next iteration. In a while-loop, the loop-continuation test evaluates immediately after the continue statement executes. In a for-loop, the loop variable will be assigned to the next value in the list after the continue statement executes.

Classwork exercise 3.2

Evaluate the output of the following programs.

```
for (int i = 0; i < 6; i++)
{
    if (i==3) break;
    cout << i << endl;
}
```

```
0
1
2
```

```
for (int i = 0; i < 6; i++)
{
    if (i==3) continue;
    cout << i << endl;
}
```

```
0
1
2
4
5
```

File processing

We can use a `for` loop or a `while` loop to read and process data from a file. The function `.eof()` in `<fstream>` is a boolean function which returns true when the end of the file has been read until its end.

If the number of entries is known, we can use for loop to read that number of entries. If not, we might set up a while loop using `while(!filename.eof())` to read until the end of file.

Example 3.6 Evaluation of midterm marks.
The file "midterm_marks.txt" contains the mark (integers 0-100) of an uncertain number of students. Write a program to read the marks and evaluate the mean, maximum and minimum marks.

```
67 58 99 37 48 89 75 78 93 56
```

```
There are 10 entries.
The average is 70 marks.
The maximum is 99 marks.
The minimum is 37 marks.
```

```cpp
// 3.6 File processing
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
        int count=0;
        double mark, mean_mark=0;
        double max_mark=0, min_mark=100;
        ifstream file;
        file.open("midterm_marks.txt");
        while (!file.eof())
        {
                file >> mark;
                if (mark > max_mark) max_mark = mark;
                if (mark < min_mark) min_mark = mark;
                mean_mark += mark;
                count++;
        }
        file.close();
        mean_mark /= count;
        cout << "There are " << count << " entries."  << endl;
        cout << "The average is " << mean_mark << " marks."  << endl;
        cout << "The maximum is " << max_mark << " marks."  << endl;
        cout << "The minimum is " << min_mark << " marks."  << endl;
        return 0;
}
```

update the maximum/minimum mark
if a bigger/smaller mark is observed

```
12        while (!file.eof())
13        {
14              file >> mark;
15              if (mark > max_mark) max_mark = mark;
16              if (mark < min_mark) min_mark = mark;
17              mean_mark += mark;
18              count++;
19        }
20        file.close();
21        mean_mark /= count;
```

sum up the total mark and
the total number of entries

calculate the mean mark based on total
mark and total number of students

Nested for loop

We can nest a `for` loop inside another `for` loop to perform repetition of more than one dimension.

//Syntax for a nested `for` loop with two counters:

```
for (int i = initialValue1; i < stopValue1; i++)
{
    for (int j = initialValue2; j < stopValue2; j++)
    {
        statement(s); // loop body
    }
}
```

For example, if we want to repeat a task for outer counter i going from 1 to 10 and inner counter j going from 1 to 5:

```
for (int i = 1; i <= 10; i++)
{
    for (int j = 1; j <= 5; j++)
    {
        statement(s); // loop body
    }
}
```

Notice that under a value of the outer counter, the inner counter will be looped completely before going to the next value of the outer counter. The process will be like:

i = 1, j = 1, 2, 3, 4, 5

i = 2, j = 1, 2, 3, 4, 5

…

i = 10, j = 1, 2, 3, 4, 5

Example program 3.7 Multiplication table

A multiplication table involves two dimensions, the row number and the column number with both valued 1 to 9. The product of these numbers is shown in each entry.

To display this table, we need a nested for-loop with two indexes: i for the row number and j for the column number.

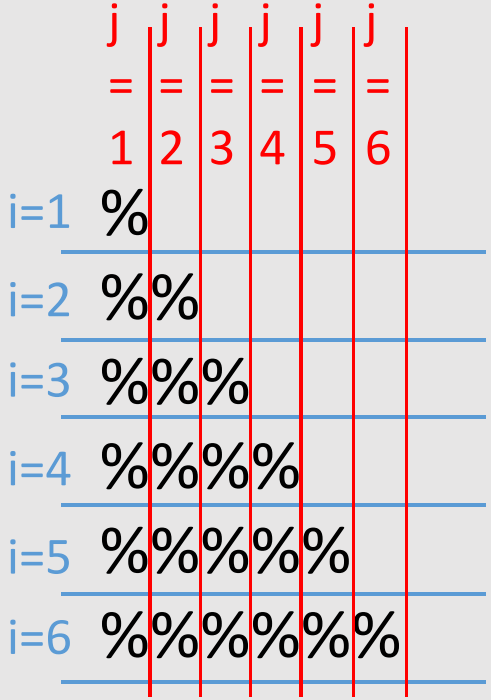
```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81

--------------------------------
Process exited after 0.05301 seconds with return value 0
Press any key to continue . . .
```

Example program 3.7 Multiplication table

```
 1 // 3.7 Nested for Loop
 2 #include <iostream>
 3 #include <iomanip>
 4 using namespace std;
 5 int main()
 6 {
 7          for (int i = 1; i <= 9; i++)
 8          {
 9                   for (int j = 1; j <= 9; j++)
10                   {
11                            cout << setw(3) << i * j;
12                   }
13                   cout << endl;
14          }
15          return 0;
16 }
```

Example program 3.8 Triangle

In this program, we aim to build a triangular pattern with any character and any number of layers, for example:



Using nested for-loop, we easily come up with i being the row number, and it goes from 1 to n (n=6 here). However, the range of values of the column number j is dependent to i. This is because different layers got different widths.

We can consider, i goes from 1 to n and j goes from 1 to i. Notice that in a nested for-loop, an inner index can depends on an outer index.

Example program 3.8 Triangle

```
 1 // 4.9 Nested for Loop
 2 #include <iostream>
 3 #include <cmath>
 4 using namespace std;
 5 int main()
 6 {
 7           int n;
 8           cout << "Enter the number of layers: ";
 9           cin >> n;
10           char ch;
11           cout << "Enter the character: ";
12           cin >> ch;
13           for (int i = 1; i <= n; i++)
14           {
15                     for (int j = 1; j <= i; j++)
16                     {
17                               cout << ch;
18                     }
19                     cout << endl;
20           }
21           return 0;
22 }
```

Classwork 3.3

Write a program that prompts the user to enter an positive integer n and produce a checker pattern on a nxn square.

```
Enter the size: 10
0X0X0X0X0X
X0X0X0X0X0
0X0X0X0X0X
X0X0X0X0X0
0X0X0X0X0X
X0X0X0X0X0
0X0X0X0X0X
X0X0X0X0X0
0X0X0X0X0X
X0X0X0X0X0
```