

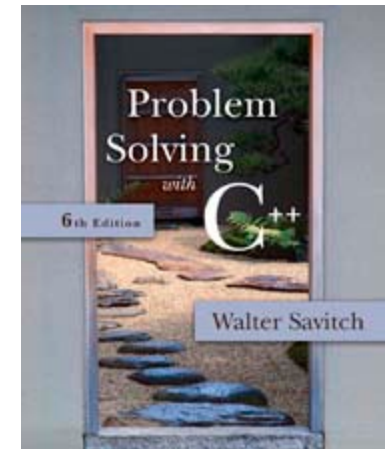
APS105: Lecture 12

Wael Aboelsaadat

wael@cs.toronto.edu

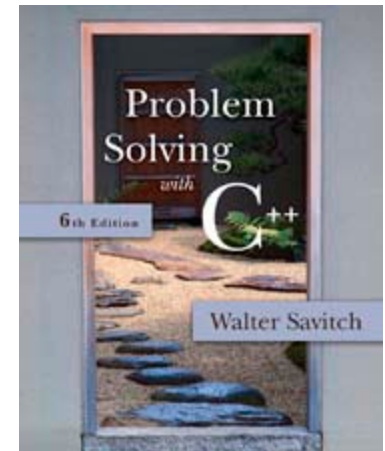
<http://ccnet3.utoronto.ca/20079/aps105h1f/>

Acknowledgement: These slides are a modified version of the text book slides as supplied by Addison Wesley



Chapter 3

More Flow of Control



The for-Statement

- A for-Statement (for-loop) is another loop mechanism in C++
 - Designed for common tasks such as adding numbers in a given range
 - Is sometimes more convenient to use than a while loop
 - Does not do anything a while loop cannot do

for/while Loop Comparison

- ```
sum = 0;
n = 1;
while(n <= 10) // add the numbers 1 - 10
{
 sum = sum + n;
 n++;
}
```
- ```
sum = 0;
for (n = 1; n <= 10; n++) //add the numbers 1 - 10
    sum = sum + n;
```

For Loop Dissection

- The for loop uses the same components as the while loop in a more compact form

- for (n = 1; n <= 10; n++)



for Loop Alternative

- A for loop can also include a variable declaration in the initialization action
 - `for (int n = 1; n <= 10; n++)`
This line means
 - Create a variable, `n`, of type `int` and initialize it with 1
 - Continue to iterate the body as long as `n <= 10`
 - Increment `n` by one after each iteration
- For-loop syntax and while loop comparison are found in

Display 3.11

for-loop Details

- Initialization and update actions of for-loops often contain more complex expressions

- Here are some samples

- `for (n = 1; n <= 10; n = n + 2)`

`for(n = 0 ; n > -100 ; n = n -7)`

- `for(double x = pow(y,3.0); x > 2.0; x = sqrt(x))`

The for-loop Body

- The body of a for-loop can be

- A single statement
- A compound statement enclosed in braces

- Example:

```
for(int number = 1; number >= 0; number--)  
{  
    // loop body statements  
}
```

- **Display 3.13** shows the syntax for a for-loop with a multi-statement body

The Empty Statement

- A semicolon creates a C++ statement
 - Placing a semicolon after `x++` creates the statement
`x++;`
 - Placing a semicolon after nothing creates an empty statement that compiles but does nothing

```
cout << "Hello" << endl;  
;  
cout << "Good Bye" << endl;
```

Extra Semicolon

- Placing a semicolon after the parentheses of a for loop creates an empty statement as the body of the loop
 - Example:

```
for(int count = 1; count <= 10; count++);  
cout << "Hello\n";
```

prints one "Hello", but not as part of the loop!

- The empty statement is the body of the loop
- `cout << "Hello\n";` is not part of the loop body!

Local Variable Standard

- ANSI C++ standard requires that a variable declared in the for-loop initialization section be local to the block of the for-loop
- Find out how your compiler treats these variables!
- If you want your code to be portable, do not depend on all compilers to treat these variables as local to the for-loop!

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 occasions 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

The break-Statement

- There are times to exit a loop before it ends
 - If the loop checks for invalid input that would ruin a calculation, it is often best to end the loop
- The break-statement can be used to exit a loop before normal termination
 - Be careful with nested loops! Using break only exits the loop in which the break-statement occurs

Display 3.14

Section 3.3 Conclusion

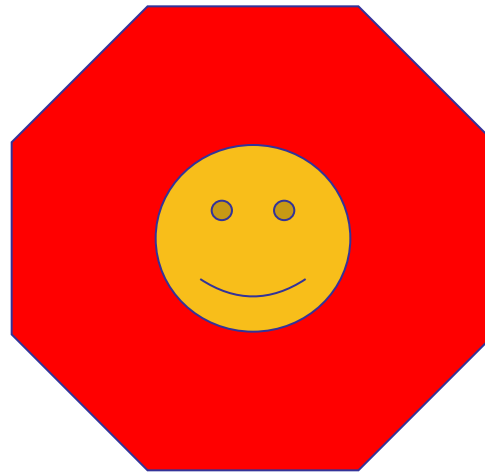
- Can you
 - Determine the output of the following?
for(int count = 1; count < 5; count++)
 cout << (2 * count) << " ";
 - Determine which type of loop is likely to be best for
 - Summing a series such as $1/2 + 1/3 + 1/4 + \dots + 1/10$?
 - Reading a list of exam scores for one student?
 - Testing a function to see how it performs with different values of its arguments

Nested Loops

- The body of a loop may contain any kind of statement, including another loop
 - When loops are nested, all iterations of the inner loop are executed for each iteration of the outer loop
 - Give serious consideration to making the inner loop a function call to make it easier to read your program
- Display 3.15 show two versions of a program with nested loops

Display 3.15

Chapter 3 -- End



for Statement

Syntax

```
for (Initialization_Action; Boolean_Expression; Update_Action)  
  Body_Statement
```

Example

```
for (number = 100; number >= 0; number--)  
  cout << number  
  << " bottles of beer on the shelf.\n";
```

Equivalent while loop

Equivalent Syntax

```
Initialization_Action;  
while (Boolean_Expression)  
{  
  Body_Statement  
  Update_Action;  
}
```

Equivalent Example

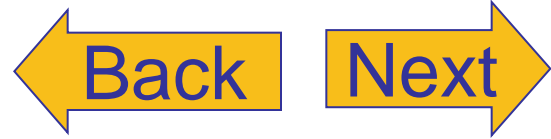
```
number = 100;  
while (number >= 0)  
{  
  cout << number  
  << " bottles of beer on the shelf.\n";  
  number--;  
}
```

Output

```
100 bottles of beer on the shelf.  
99 bottles of beer on the shelf.  
.  
.  
.  
0 bottles of beer on the shelf.
```



Display 3.12



A for Statement

```
//Illustrates a for loop.
#include <iostream>
using namespace std;

int main()
{
    int sum = 0;

    for (int n = 1; n <= 10; n++)
        sum = sum + n;

    cout << "The sum of the numbers 1 to 10 is "
         << sum << endl;
    return 0;
}
```

Initializing action

Repeat the loop as long as this is true.

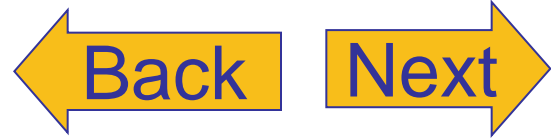
Done after each loop body iteration

//Note that the variable n is a local variable of the body of the for loop!

Output

The sum of the numbers 1 to 10 is 55

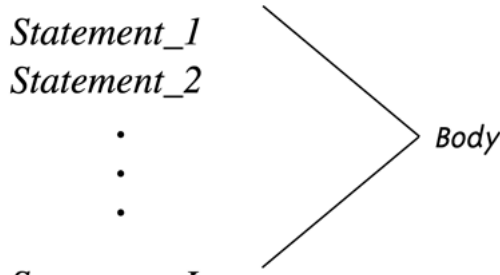
Display 3.13



for Loop with a Multistatement Body

Syntax

```
for (Initialization_Action; Boolean_Expression; Update_Action)  
{  
    Statement_1  
    Statement_2  
    .  
    .  
    .  
    Statement_Last  
}
```

A diagram consisting of a large right-pointing chevron shape. The word 'Body' is written to the right of the chevron. Lines connect the top and bottom of the chevron to the word 'Body'. The top line starts from the right side of 'Statement_1' and the bottom line starts from the right side of 'Statement_Last'.

Example

```
for (int number = 100; number >= 0; number--)  
{  
    cout << number  
        << " bottles of beer on the shelf.\n";  
    if (number > 0)  
        cout << "Take one down and pass it around.\n";  
}
```

A break Statement in a Loop

```
//Sums a list of ten negative numbers.
#include <iostream>
using namespace std;

int main()
{
    int number, sum = 0, count = 0;
    cout << "Enter 10 negative numbers:\n";

    while (++count <= 10)
    {
        cin >> number;

        if (number >= 0)
        {
            cout << "ERROR: positive number"
                << " or zero was entered as the\n"
                << count << "th number! Input ends "
                << "with the " << count << "th number.\n"
                << count << "th number was not added in.\n";
            break;
        }

        sum = sum + number;
    }

    cout << sum << " is the sum of the first "
        << (count - 1) << " numbers.\n";

    return 0;
}
```

Sample Dialogue

```
Enter 10 negative numbers:
-1 -2 -3 4 -5 -6 -7 -8 -9 -10
ERROR: positive number or zero was entered as the
4th number! Input ends with the 4th number.
4th number was not added in.
-6 is the sum of the first 3 numbers.
```

Display 3.14

