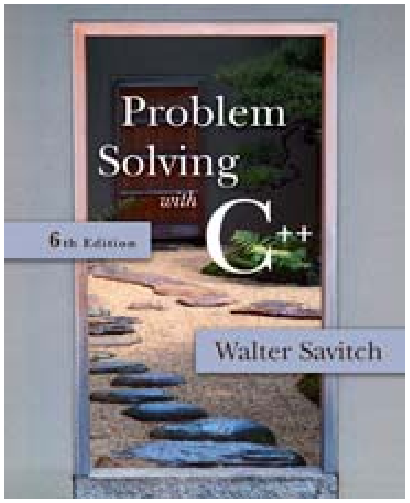
APS105: Lecture 7

Wael Aboelsaadat

wael@cs.toronto.edu

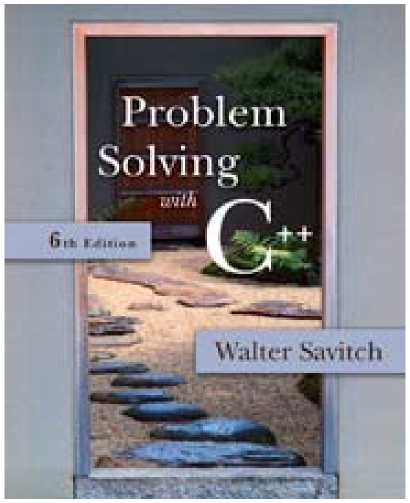
http://ccnet3.utoronto.ca/20079/aps105h1f/

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



2.4

Simple Flow of Control



Implementing the Branch

- if-else statement is used in C++ to perform a branch
  - if (hours > 40)
        gross_pay = rate * 40 + 1.5 * rate * (hours - 40);
  - else
  - gross_pay = rate * hours;

Display 2.7

Boolean Expressions

- Boolean expressions are expressions that are either true or false
- comparison operators such as '>' (greater than) are used to compare variables and/or numbers
    - (hours > 40)   Including the parentheses, is the boolean expression from the wages example
    - A few of the comparison operators that use two symbols (No spaces allowed between the symbols!)
        - >=   greater than or equal to
        - !=   not equal or inequality
        - = =   equal or equivalent

Display 2.8

if-else Flow Control (1)

- if (boolean expression)
      true statement
  else
      false statement
- When the boolean expression is true
  - Only the true statement is executed
- When the boolean expression is false
  - Only the false statement is executed

if-else Flow Control (2)

- if (boolean expression)
  ```
  {
          true statements
  }
  else
  {
  ```
-
  ```
          false statements
  }
  ```
- When the boolean expression is true
  - Only the true statements enclosed in { } are executed
- When the boolean expression is false
  - Only the false statements enclosed in { } are executed

AND

- Boolean expressions can be combined into more complex expressions with
  - &&   -- The AND operator
    - True if both expressions are true
- Syntax: (Comparison_1) && (Comparison_2)
- Example: if ((2 < x) && (x < 7))
  - True only if x is between 2 and 7
  - Inside parentheses are optional but enhance meaning

OR

- | | -- The OR operator (no space!)
  - True if either or both expressions are true

- Syntax: (Comparison_1) | | (Comparison_2)

- Example: if ((x = = 1) | | (x = = y))
  - True if x contains 1
  - True if x contains the same value as y
  - True if both comparisons are true

NOT

- ! -- negates any boolean expression
  - !(x < y)
    - True if x is NOT less than y
  - !(x = = y)
    - True if x is NOT equal to y
- ! Operator can make expressions difficult to understand…use only when appropriate

Inequalities

- Be careful translating inequalities to C++
- if x < y < z   translates as

if ((x < y) && (y < z))

NOT

~~if (x < y < z)~~

Pitfall: Using = or ==

- ' = ' is the assignment operator
  - Used to assign values to variables
  - Example: x = 3;
- '= = ' is the equality operator
  - Used to compare values
  - Example: if (x == 3)
- The compiler will accept this error:

  if (x = 3)

  but stores 3 in x instead of comparing x and 3
  - Since the result is 3 (non-zero), the expression is true

Compound Statements

- A compound statement is more than one statement enclosed in { }
- Branches of if-else statements often need to execute more that one statement
- Example:

```
if (boolean expression)
  {
        true statements
  }
 else
  {
            false statements
   }
```

-

Display 2.9

Branches Conclusion

- Can you
  - Write an if-else statement that outputs the word High if the value of the variable score is greater than 100 and Low if the value of score is at most 100? The variables are of type int.

  - Write an if-else statement that outputs the word Warning provided that either the value of the variable temperature is greater than or equal to 100, or the of the variable pressure is greater than or equal to 200, or both. Otherwise, the if_else sttement outputs the word OK. The variables are of type int.

Simple Loops

- When an action must be repeated, a loop is used
- C++ includes several ways to create loops
- We start with the while-loop
- Example:

```
while (count_down > 0)
  {
       cout << "Hello ";
        count_down  -= 1;
   }
```

- Output:

```
Hello Hello Hello
```

  when count_down starts at 3

Display 2.10

While Loop Operation

- First, the boolean expression is evaluated
  - If false, the program skips to the line following the while loop
  - If true, the body of the loop is executed
    - During execution, some item from the boolean expression is changed
  - After executing the loop body, the boolean expression is checked again repeating the process until the expression becomes false
- A while loop might not execute at all if the boolean expression is false on the first check

while Loop Syntax

- while (boolean expression is true)

  ```
  {
        statements to repeat
  }
  ```

  - Semi-colons are used only to end the statements within the loop
- While (boolean expression is true)

  ```
        statement to repeat
  ```

Display 2.11

An *if-else* Statement

```
#include <iostream>
using namespace std;
int main( )
{
    int hours;
    double gross_pay, rate;

    cout << "Enter the hourly rate of pay: $";
    cin >> rate;
    cout << "Enter the number of hours worked,\n"
         << "rounded to a whole number of hours: ";
    cin >> hours;

    if (hours > 40)
        gross_pay = rate*40 + 1.5*rate*(hours - 40);
    else
        gross_pay = rate*hours;

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "Hours = " << hours << endl;
    cout << "Hourly pay rate = $" << rate << endl;
    cout << "Gross pay = $" << gross_pay << endl;

    return 0;
}
```

Sample Dialogue 1

```
Enter the hourly rate of pay: $20.00
Enter the number of hours worked,
rounded to a whole number of hours: 30
Hours = 30
Hourly pay rate = $20.00
Gross pay = $600.00
```

Sample Dialogue 2

```
Enter the hourly rate of pay: $10.00
Enter the number of hours worked,
rounded to a whole number of hours: 41
Hours = 41
Hourly pay rate = $10.00
Gross pay = $415.00
```

Display 2.7

Back

Next

Display 2.8

Back Next

Syntax for an `if-else` Statement

A Single Statement for Each Alternative:

```
if (Boolean_Expression)
     Yes_Statement
else
     No_Statement
```

A Sequence of Statements for Each Alternative:

```
if (Boolean_Expression)
{
     Yes_Statement_1
     Yes_Statement_2
       . . .
     Yes_Statement_Last
}
else
{
     No_Statement_1
     No_Statement_2
       . . .
     No_Statement_Last
}
```

Display 2.9

Back Next

Comparison Operators

Math Symbol	**English**	**C++ Notation**	**C++ Sample**	**Math Equivalent**
$=$	equal to	`==`	`x + 7 == 2*y`	$x + 7 = 2y$
$\neq$	not equal to	`!=`	`ans != 'n'`	ans $\neq$ 'n'
$<$	less than	`<`	`count < m + 3`	count $<$ m $+$ 3
$\leq$	less than or equal to	`<=`	`time <= limit`	time $\leq$ limit
$>$	greater than	`>`	`time > limit`	time $>$ limit
$\geq$	greater than or equal to	`>=`	`age >= 21`	age $\geq$ 21

Display 2.10

Back Next

Compound Statements Used with *if-else*

```
if (my_score > your_score)
{
    cout << "I win!\n";
    wager = wager + 100;
}
else
{
    cout << "I wish these were golf scores.\n";
    wager = 0;
}
```

Display 2.11

Back Next

A *while* Loop

```
#include <iostream>
using namespace std;
int main( )
{
    int count_down;

    cout << "How many greetings do you want? ";
    cin >> count_down;

    while (count_down > 0)
    {
        cout << "Hello ";
        count_down = count_down - 1;
    }

    cout << endl;
    cout << "That's all!\n";

    return 0;
}
```

Sample Dialogue 1

```
How many greetings do you want? 3
Hello Hello Hello
That's all!
```

Sample Dialogue 2

```
How many greetings do you want? 1
Hello
That's all!
```

Sample Dialogue 3

```
How many greetings do you want? 0

That's all!
```

The loop body is executed zero times.