Life is Full of Alternatives

Problem

- Write a C++ program that allows the user the ability to enter their name and the number of nickels and pennies they have. You are then to print the number of dollars and change that corresponds to. The change should be in the form of nickels and pennies

Last Time

- So far we have learned the basic statements in C++. These are
  - o Input/output statements
  - o Assignment statements
  - o Arithmetic statements
- When we compiled and ran our programs all the statements were executed in sequential order. In other words, the statements were executed in order, one after the other.

These are called *sequential structures*

Today

- Today we will begin examining how C++ statements can be executed out of sequence, and some statements could be skipped altogether
- Specifically, we will be looking at *selection structures*

UML Activity Diagrams

- UML: Unified Modelling Language
- Used to represent algorithms that will later be translated into code
- Give the programmer a visual representation of the solution to a problem
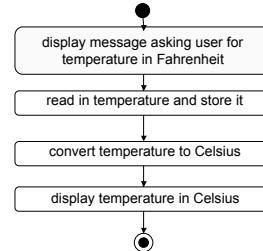- Can also help the programmer see a solution to a problem

Example

- Write a program that will read in the current temperature in degrees Fahrenheit and convert it into Celsius

1

`if` Selection Structure

- The `if` selection structure allows a program to make a decision based on the truth or falsity of some condition

- The format is

  `if (some condition is true)`

  `execute some statement(s)`

- Example

  `if (weight > 100.0)`

  `shipCost = 10.00;`

`if` Selection Structure

- If the condition in the `if` selection structure evaluates to false, then the statement following the `if` will be skipped

  `if( grade > 59 )`

  `cout << "passed!";`

- "passed!" will only be output if the grade is greater than 59. If the grade is 59 or less the passed will not be output and the program will continue
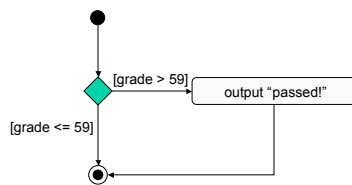
UML Activity Diagram

`if( grade > 59 )`

`cout << "passed!";`

[grade > 59]

output "passed!"

[grade <= 59]

Equality and Relational Operators

- Conditions in `if` selection structures are formed using equality and relational operators

`<`	less than	relational
`>`	greater than	relational
`<=`	less than or equal to	relational
`>=`	greater than or equal to	relational
`==`	equal to	equality
`!=`	equal to	equality

Examples

- `if( height >= 6.2 )`

- `if( age == 18 )`

- Notice the difference between == and =
  - == is only used to compare two values
  - = is used to assign the value to right into the variable on the left

Example

- Your local bookstore has asked you to write a program to help them determine the cost of shipping of customers orders. If the order is $30 or less then shipping will cost $5, if the order is over $30 then shipping will be $3.

- 7.1: Write an algorithm to solve this problem

Solution

```
#include <iostream>
#include "stdafx.h"

using namespace std;

int main()
{
  double order, shipping;

  cout << "Enter the total amount of the order: ";
  cin >> order;

  if( order <= 30 )
    shipping = 5.00;

  if( order > 30 )
    shipping = 3.00;

  cout << "The cost of shipping is $" << shipping
        << endl;

  return 0;
}
```

Problem

- The bookstore has now changed it's shipping policy so that
  o If the order is $30 or less, shipping is $5
  o If the order is over $30 but less than $50, shipping is $3
  o If the order is over $50 then shipping is $2

- 7.2: What would we need to change in the program?

Summary

- In today's lecture we learnt
  o UML activity diagrams
  o Simple **if** selection structure
  o Relational and equality operators

- Readings from Chapter 2
  o P. 34 - 39: simple **if**, equality and relational operators
  o P. 71 - 77: **if**, UML, **bool**