Life is Full of Alternatives

Last Time

- So far we have learnt the basic statements in C++. These are
  - Input/output statements
  - Assignment statements
  - 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

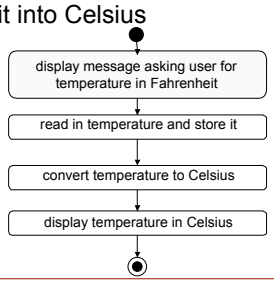
- Today we will begin examining how C++ statements can be executed out of sequence, and some statements could be skipped altogether
- Specifically, we will be looking at *selection structures*

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



`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

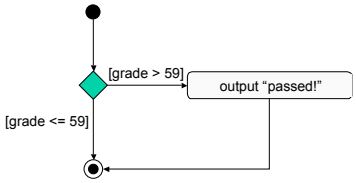
- If the condition in the `if` selection structure evaluates to false, then the statement following the `if` will be skipped

```
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!";
```



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.
- 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
  - If the order is $30 or less, shipping is $5
  - If the order is over $30 but less than $50, shipping is $3
  - If the order is over $50 then shipping is $2
- What would we need to change in the program?

Logical Operators

- If we want to check for more than one condition then we need to use logical operators
- These combine logical expressions (i.e. expressions that have a true/false value)
- There are three logical operators
  - && and
  - || or
  - ! Not

Examples of Logical Operators

- `if( ( x > 7 ) && ( x < 20 ) )`
- `if( ( temp > 90.0 ) && ( humidity > 0.9 ) )`
- `if( ( salary < minSalary ) || ( dependents > 5 ) )`

Problem

- Using logical expressions, how can we solve the bookstore problem
- The bookstore has now changed it's shipping policy so that
  - If the order is $30 or less, shipping is $5
  - If the order is over $30 but less than $50, shipping is $3
  - If the order is over $50 then shipping is $2

Evaluating Expressions: And &&

- **`(expr1) && (expr2)`**
- For the complete expression to be true, both expr1 and expr2 have to be true
- Example:

  `(temp > 90.0) && (humidity > 0.9)`
  - These are unbearable heat and humidity conditions
  - Both must be true for the entire expression to be true

Evaluating Expressions: Or ||

- (**`expr1 || expr2`**)
- The complete expression is true if either expr1 or expr2 is true
- Examples:
  - **`(salary < minSalary) || (dependents > 5)`**
  - To qualify for financial aid, salary has to be less than some minimum salary or the number of dependents is greater than 5
  - Only one condition has to be true

Evaluating Expressions: Not !

- `!expr`
- Unary operator
- Examples:
  - `!((salary < minSalary) && (dependents > 5))`
  - What makes this true? False?

Operator Precedence

- We have now added relational, equality and logical operators to the mathematical operators that were introduced last week
- Where do the new operators fit in the precedence table?

Operator Precedence & Associativity

`()`	`L->R`	Parentheses
`!, +, -`	`R->L`	Negation, Unary +, -
`*,/,%`	`L->R`	Mult, div, mod
`+, -`	`L->R`	Add, Subtract
`<<, >>`	`L->R`	Insertion/extraction
`<, <=, >, >=`	`L->R`	Relational
`==, !=`	`L->R`	Equality
`&&`	`L->R`	And
`||`	`L->R`	Or
`=`	`R->L`	Assignment

Expression Evaluation

- According to the operator precedence and associativity rules given on the previous slide, how will the following expressions be evaluated?
  - `x < min + max`
  - `min <= x && x <= max`
  - `!x == y + 2`
  - `x = a + b % 7 * 2`

`bool` Data Type

- `bool`: boolean
- Variables of type `bool` can be either `true` or `false`
  - They cannot be any other value
- Boolean variable names should start with b
  - See coding standards
- Example

```
bool bCanVote;
int age;
cin >> age;
bCanVote = age >= 18;
cout << bCanVote;
```


Data Types

- So far we have been introduced to the following C++ data types
  - int
  - double
  - char
  - string
  - bool

Examples

- Assume that
  - double x = 3.0;
  - double y = 4.0;
  - double z = 2.0;
  - bool bFlag = false;
- What is the value of the following expressions

```
!bFlag
x + y/z <= 3.5
!bFlag || (y + z >= x - z)
!(bFlag || (y + z >= x - z)
```


Summary

- In today's lecture we covered
  - UML activity diagrams
  - Simple **`if`** selection structure
  - Relational and equality operators
  - Logical operators
  - **`bool`** data type
- Readings
  - P. 34 - 39: simple **`if`**, equality and relational operators
  - P. 71 - 77: **`if`**, UML, **`bool`**
  - P. 124 - 128: logical operators, confusing = and ==