CSCI-1411 Fundamentals Of Computing Lab

Anh Nguyen

Fall 2015

Lab 3: Expressions, Input, Output and Data Type Conversions

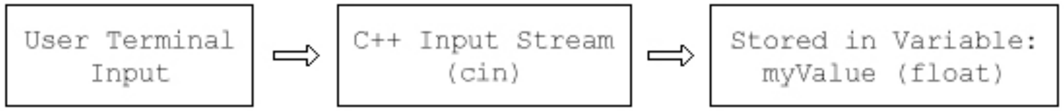
- Overview:
  - Lab 3 Components
    - Lab Sections (3.1, 3.2, 3.3, 3.4, 3.5, Design Document)
  - Lab 3 Concepts
    - User Input
    - Terminal (output) Formatting
  - C++ Standard Library Reference
    - http://www.cplusplus.com/
    - Utilize the search feature (ex. search for any standard function name)
  - Complete each Exercise
    - Turn in your source code after all changes have been made
    - Answer the questions from the exercises in a comment block

Lab 3: Expressions, Input, Output and Data Type Conversions

C++ Simple User Input

- Utilizes the built in `cin` stream with the `>>` (extraction) operator
- Example:

```
float myValue;
cin >> myValue;
// use myValue...
```



- Can be utilized with several data-types:
  - int, double, float
  - Strings are slightly different:
    - Will only parse or accept the first word or 'token' the user provides:

```
string name;
cin >> name;
```

    - At the terminal: What is your name? Bob Watson
    - Value of name = Bob

Lab 3: Expressions, Input, Output and Data Type Conversions

- C++ String Reading
  - C-string: an array of characters
    - char name[12]; // How many characters can be hold up?

      // Which character the last character must be reserved for?
  - Skip leading whitespaces
    - cin >> name;
  - To handle whitespaces (blank spaces, tabs, line breaks, etc.)
    - cin.getline(name, 12); // C-string
    - getline(cin, name); // string

Lab 3: Expressions, Input, Output and Data Type Conversions

- C++ Terminal Output formatting
  - Obviously spaces and tab characters can be utilized (`" "`, `"\t"`)
    - Tabs are not reliable (is a tab a character? 2 spaces, 4 spaces, 8 spaces?)
    - Inserting spaces becomes incredibly tedious
  - `setprecision(int n)`
    - Number of decimal places to display
  - `setw(int n)`
  - `fixed`
  - `showpoint`

  → include `<iomanip>` directive

Lab 3: Expressions, Input, Output and Data Type Conversions

- Data Type Conversion
  - Type coercion → implicitly
    - int count = 10.89;
    - cout << count; // What value is printed?
  - Type casting → explicitly
    - count = static_cast<int>(10.89);
  - Example:

```
int num_As = 10;
int totalgrade = 50;
float percent_As;
```

1. percent_As = num_As/totalgrade; // What value is printed?

2. percent_As = static_cast<float>(num_As)/totalgrade; // What value is printed?

Lab 3: Expressions, Input, Output and Data Type Conversions

- 3.1 Working with the `cin` Statement
  - (bill.cpp)
  - Answer questions asked in exercise 2 & 3

- 3.2 Formatting Output
  - (tabledata.cpp)

- 3.3 Arithmetic Operations and Math Functions
  - (righttrig.cpp)

- 3.4 Working with Type Casting
  - (batavg.cpp)
  - Answer questions asked in exercise 1 & 2

Lab 3: Expressions, Input, Output and Data Type Conversions

- 3.5 Develop your own Program
  - Choose 1of the 3 options
  - Name the source file: main.cpp
  - Include a design document for the option you choose
    - Includes algorithm description, input, output, diagrams, formulas, etc.

Lab 3: Expressions, Input, Output and Data Type Conversions

- Submission File Checklist
  - Submit all files on Canvas (One at a time or all of them in a single zip file). Be sure to include all source files and documents.

  - 3.1 bill.cpp
  - 3.2 tabledata.cpp
  - 3.3 righttrig.cpp
  - 3.4 batavg.cpp
  - 3.5 main.cpp (For any option you choose)
  - 3.5 Design Document

Lab 3: Customizing VIM

- Vi/Vim contains several more features than nano:
  - Line numbers
  - Syntax Highlighting
  - Powerful Shortcuts

- Vim can be customized to display all of these by default
  - Utilizing a shell script we can save these settings

1. Download the change vim.sh
2. Copy the file change vim.sh to your home directory:
   - `transues/changevim.sh`
3. Run the script using: `sh changevim.sh`