

EECS168/169-Lab2

Input/Output, Variables, if-else statement

University of Kansas
Spring 2018

Announcements

- **Please read the wiki page's detailed instructions for each lab carefully and before doing any other task.**
 - Slides are just a summary of the lab.
- **Please read the rubric carefully.**
- **Please upload your assignments on Blackboard.**
- Grades and feedback will be posted on Blackboard.
- This lab is due one week from the start of the lab.
- Your submission will be checked according to your Blackboard submission time.
- We have an in-class assignment on debuggers worth 10 points (5 for 169).
 - Please participate in the exercise for receiving these points.

If you are **SSHing** (remote access to lab cycles)

""IMPORTANT NOTE""

- *cycle1.eecs.ku is NOT a mirror of the lab machines
- *it is running a different version of linux
- *I do not recommend using cycle1 for checking your work

Objectives

- 1. Input and output
 - `std::cout`
 - `std::cin`
- 2. Use primitive variables
 - Data Type
 - Assignment Statements
 - Using Named Constants
 - Convention for naming variables
- 3. Numeric Operators
- 4. Code Comments
- 5. if-else statement
- 6. Debuggers

Input and output syntax


- Adding line in your main.cpp

- `using namespace std;`



cout is name for screen


- `cout << "Hello world";`



- `cout << kuid;`



cin is name for keyboard



- `cin >> kuid;`

- Think of:

- **cout** as name for screen – **output** device
 - **cin** as name for keyboard – **input** device
 - `<<` and `>>` point in direction data moves.

Data Type

- Type of a value. E.g.: Numbers, alphabets(characters)
- Integers:
 - 0
 - -1, -2, -3, ..
 - 1, 2, 3, ..
- Characters:
 - a,b,c,.. A, B, C,.. #,@,\$,.. \n,\t,\0,..
- Double/Float:
 - Numbers with decimal point. -1.5, 3.14,..
- Boolean:
 - 0 and 1 – Only two values
- Defined in C++ as: `int`, `char`, `double/float`, `bool`

Variables

- Variables in a program are used to store data/value.
- Value stored can be changed later.
- Variable declaration
 - Telling the compiler what will be the data type of the value stored in it.
 - Type name(int, char) & name(conventions)
 - Where to declare:
 - At the start of the program
 - Just before where they are used
 - Multiple variables of same type can be declared in one line.

```
int exam_score;  
int kuid;  
char grade;  
double percentage;
```

```
int exam_score, kuid;
```

- Variable initialization.
 - Store some initial value while declaring.
 - Risky not to!

```
int exam_score = 0, kuid = 0;  
char grade = '\0';  
double percentage = 0.0;
```

Assignment Statements

- The most direct way to change the value of a variable.

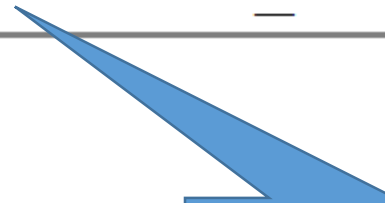
```
exam_score = 95;  
grade = 'a';  
percentage = 85.45;
```

- Syntax:

Variable = Expression;

```
int exam1_score, exam2_score, total_score;  
exam1_score = 95;  
exam1_score = 88;  
total_score = exam1_score + exam2_score;
```

```
cout << total_score;
```



Right hand side is
evaluated First

The convention for naming variables

```
float a, b;  
a = 0.0;  
b = 0.0;  
  
cin >> a;  
b = 3.14 * a * a;  
cout << b;
```

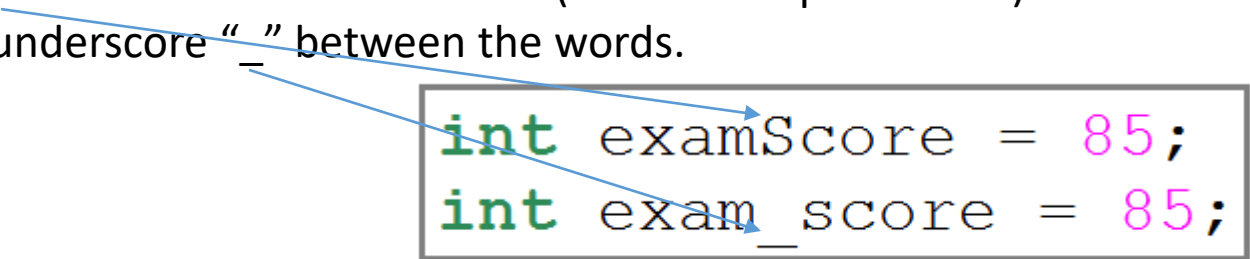
What is this calculation doing?

What is this calculation doing?

```
float radius, area_circle;  
radius = 0.0;  
area_circle = 0.0;  
  
cin >> radius;  
area_circle = 3.14 * radius * radius;  
cout << area_circle;
```

The convention for naming variables

- Start the variable name with a lowercase letter – a,b,c,..
- More than one word then separate the words by:
 - 1. Capital first letter of second word(each subsequent word)
 - 2. Use underscore “_” between the words.



```
int examScore = 85;  
int exam_score = 85;
```

The diagram illustrates the naming convention for variables. It shows two examples of variable declarations. The first example, `int examScore = 85;`, uses camel case where the first letter of each word is capitalized. The second example, `int exam_score = 85;`, uses snake case where words are separated by underscores. Blue arrows point from the list items to these examples: one arrow points from 'Capital first letter of second word' to the 'S' in 'examScore', and another arrow points from 'Use underscore' to the underscore in 'exam_score'.

Named Constants

- The value of $\pi=3.14$ is a constant.
- Use keyword `const` to declare.
- Initialized once and never changed in the program. Compile error if you try.
- Syntax:
 - `const <DataType> <CONSTANTNAME> = VALUE;`

```
const float PI = 3.14;
```

Numeric Operators

- To perform arithmetic operations. Operators as per precedence.
 - $()$ - Parenthesis
 - Exponent
 - $*$: Multiplication
 - $/$: Division
 - $+$: Addition
 - $-$: Subtraction

Code Comments

- Comments are descriptions that the creator of the code uses to describe what the code is doing.

```
// This is a single line comment

/* This is a single line comment */

/* This is a multiple
   * line comment
   */

/** Notice that we have to put an asterisk
   * at the front of each line
   */
```

Compile vs Debug

- Compile is the act of turning human-readable code into code the machine can understand and execute.
- Debug is the act of finding out where in the code the application is going wrong (debug = get rid of bugs.)
- In other words, A **compiler** converts programs written in a language a human being can understand into the only language a computer understands (binary code). So to execute your code, a **compiler** is the only thing you need. A **debugger** on the other hand is a tool that helps you identify bugs (errors) in your program.

GDB Debugger

- “GNU Debugger” A debugger for several languages, including C and C++ It
 - allows you to inspect what the program is doing at a certain point during execution.
 - Errors like segmentation faults may be easier to find with the help of gdb.
 - In Make file we had: `g++ main.cpp -g -Wall -o HelloWorld`
 - -g for enable built-in debugging for gdb
 - GDB gives ability to examine the program while it is executing statement by statement.
 - for setting a break point and checking the condition of the program before running all of it
 - Links for detailed tutorials for both debuggers (gdb and ddd) is available in the lab instructions (wiki)

Debugging using gdb

- GDB
- `$gdb HelloWorld` // Runs gdb with HelloWorld program to debug
- **(gdb)** `list` //Shows the source code lines where program has temporarily **halted**
- `(gdb) run` //Starts executing the HelloWorld program from first line till breakpoint or end of program
- `(gdb) run` //After program has finished executing last line. It can be run again and again
- `(gdb) break 50` //Set breakpoint in line 50 of the **current file**
- `(gdb) info breakpoint` //Lists all breakpoints
- `(gdb) run` //Executing **halts** at line 50 and shows the line in terminal
- `(gdb) list` //Shows code around line 50 where program hit breakpoint
- `(gdb) print i` //Shows the **value stored in variable i**. **Most common usage for you in these labs/homework**
- `(gdb) next` //Go to next line and stop. Shows line where program halted again.
- `(gdb) next` // Can keep on issuing next till end of program. Any output to terminal in program gets printed.
// Also any input to variable is taken from keyboard as in normal run. Program will not advance unless input is made.
- `(gdb) print i` //Can print any variable value
- `(gdb) clear 50` //Remove breakpoint set
- `(gdb) info breakpoint` //Lists all breakpoints
- `(gdb) quit` //quits gdb

DDD

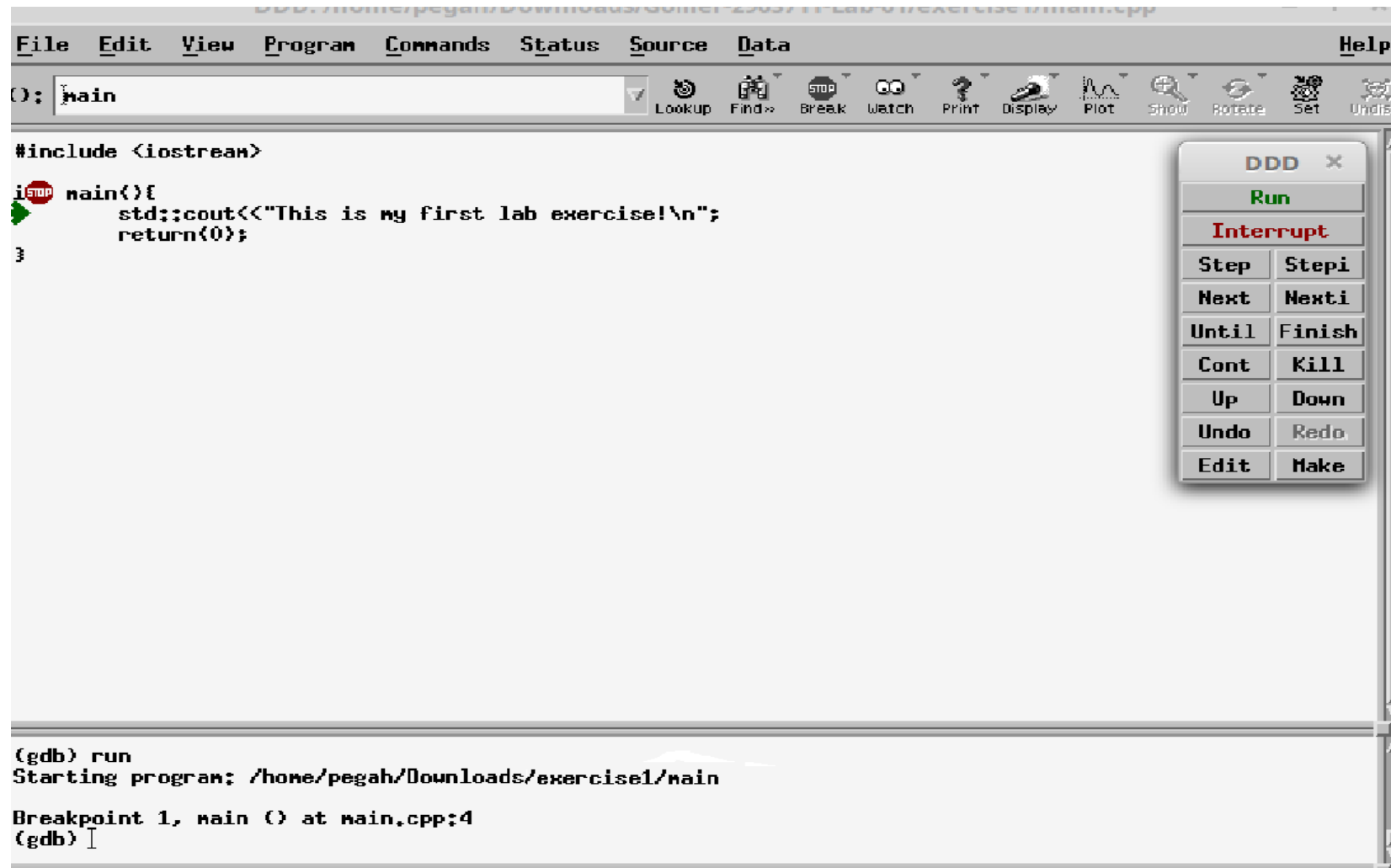
- Linux includes several debuggers in their distributions.
- However, they are all command-line debuggers which aren't very nice to use.
- Thankfully, the DDD (Data Display Debugger) is a front-end to all of these debuggers
- Or DDD is a GUI front-end to gdb (and other debuggers)

• Exercise:

ddd

Or search ddd in applications

Example



if-else statement

- Simple program flow control.
- // Checking multiple conditions

```
if (Boolean expression 1)
{
    Runs if the boolean expression is true
}
else if (Boolean expression 2)
{
    Runs if the boolean expression 1 is false and boolean expression 2 is true
}
else
{
    Runs if all previous boolean expressions are false
}
```

Pitfall:

Using = in place of ==

```
int exam_score;
cin >> exam_score;
if (exam_score = 100)
    cout << "Full score!";
```

Comparison Operators

Math Symbol	English	C++ Notation
=	equal to	==
≠	not equal to	!=
<	less than	<
≤	less than or equal to	<=
>	greater than	>
≥	greater than or equal to	>=

File header comment

- All your program files must contain this header at the very beginning of main.cpp

```
/* -----  
*  
* File Name:  main.cpp  
* Author: Your name  
* Assignment:  EECS-168/169 Lab 2  
* Description: This program will convert the input temperature from Fahrenheit to  
               Celsius.  
* Date: date the program was last modified  
*  
----- */  
//Start your program
```

Exercise 1

- In this exercise we will use the Math class to perform square root calculations.

```
#include <math.h>
```

```
double answer = 0.0;  
answer = sqrt( 9.0 );  
cout << answer; //prints 3.0
```

- Create a program to solve for the hypotenuse of a right triangle. The formula is simple, $a^2 + b^2 = c^2$. (Remember, there is no ^ operator).
- You will obtain a value for a and b then solve for c. You will need to use the sqrt() function. Make your output look like the sample output below:
- Will use the formula $a^2 + b^2 = c^2$ to solve for the hypotenuse of a triangle.
- Input a value for a: 8.0
- Input a value for b: 10.5
- The hypotenuse is : 13.2004

Exercise 2

- Create a program that solves the quadratic equation
- a , b , and c from the user

You will obtain an a , b , and c from the user

You may assume the user will give values for a , b , and c that are valid.

You must verify that the discriminant is positive **before** taking the square root → If it is not positive, give an error message

- Sample output:
 - Input a : 1.5
 - Input b : 6.5
 - Input c : 3.0
 - root 1: -0.5252
 - root 2: -3.8081

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Exercise 3

Create a new directory called exercise1

Create a program to accomplish the tasks below

Next, you will write a program that the user enters the temperature in Fahrenheit and program converts the temperature from Fahrenheit to Celsius. This program should:

Obtain a character from the user

Assume the user will input either capital 'F' or capital 'C'

Obtain a temperature

Read the floating-point numbers from the keyboard.

If the original was in Fahrenheit (F) then convert to Celsius

$$(^{\circ}\text{F} - 32) \times 5/9 = ^{\circ}\text{C}$$

Beware of integer division!

If the original was in Celsius (C) then convert to Fahrenheit

I'll leave it to you to do the algebra to solve for this conversion

Display the temperatures.

The result should look like below:

Enter the temperature in Fahrenheit: 55.5

55.5 degrees Fahrenheit = 13.0556 degrees Celsius.

Integer Division

$$\begin{array}{r} 4 \leftarrow 12/3 \\ 3 \overline{) 12} \\ \underline{12} \\ 0 \leftarrow 12\%3 \end{array}$$

$$\begin{array}{r} 4 \leftarrow 14/3 \\ 3 \overline{) 14} \\ \underline{12} \\ 2 \leftarrow 14\%3 \end{array}$$

Exercise 4

- This exercise you'll see how to cast a character to an int.
- **ASCII** (American Standard Code for Information Interchange) is the most common format for text files in computers and on the Internet. In an **ASCII** file, each alphabetic, numeric, or special character is represented with a 7-bit binary number (a string of seven 0s or 1s). 128 possible characters are defined.
- The table and values assigned to alphabetic, numeric, or special characters are predefined by the standard. You just have find the ASCII value for user's input in the predefined ASCII table.
 - **Hint: `int(your_char)` will be the ASCII value**

- Example output:

Input a character: a

The ASCII value for 'a' is: 97

Goodbye!

Exercise 5

- Create a program (without using if statements or loops) that can make change for the user. The user will give a number of pennies they have. Your program will tell them how many quarters, dimes, nickels, and pennies you can convert that to.

Example output:

How many pennies do you have? 117

Quarters: 4

Dimes: 1

Nickels: 1

Pennies: 2

Exercise -169

- Create a program that let's the user place a food order. They will get to choose from a list of foods. You'll then print a receipt with a total price before tax (subtotal), and a total price after tax.
- Taco stand requirements
- have three foods for sale, each at a different price (see examples below for names and prices)
- each item is selectable by a single letter, but the upper and lowercase should be considered valid
- the first item must have 3 optional toppings and an option for no toppings(see below)
 - only display the toppings menu if this item is chosen
 - There will be 3 toppings each at a unique price and an option for no toppings
- The user will only choose 1 item from the list then give a quantity of that item
- Assume the correct types of input, but have an error message if they don't choose a valid menu item
- Don't worry about...
 - formatting the output of the currency in this lab, meaning you can have values like "\$52.4849"
 - plurals or non-plurals based on the quantity (e.g. Salads VS Salad)
- Sample output can be found on the lab wiki page

Rubric will be strictly followed for grading.

You may assume valid input from the user.

- Submitted properly (5pts)
 - tarball name
 - email subject (if applicable)
 - tarball only contains relevant folders and files (e.g. don't include empty folders or your vacation pictures)

168

- [10pts] Exercise 1: Hypotenuse
 - [7pts] Calculations
 - [3pts] Well organized output
- [25pts] Exercise 2: Quadratic Equation
 - [10pts] Calculations
 - [10pts] Only performs calculations if the discriminant is positive (print error otherwise)
 - [5pts] Well organized output
- [25pts] Exercise 3: Temperatures
 - [10pts] Correct detection of units given by user
 - [10pts] Calculations
 - [5pts] Well organized output
- [10pts] Exercise 4: Casting ints and chars
 - [7pts] Proper conversions
 - [3pts] Well organized output
- [15pts] Comments, coding styling, and formatting
 - [5pts] Top of every .cpp file
 - [5pts] Sensical indentation
 - [5pts] Meaningful variable names
- [10pts] Debugger demonstration
 - Participate in an exercise or show your TA your now how to use the debugger

169

- [5pts] Exercise 1: Temperatures
 - [3pts] Calculations
 - [2pts] Well organized output
- [5pts] Exercise 2: Hypotenuse
 - [3pts] Calculations
 - [2pts] Well organized output
- [10pts] Exercise 3: Quadratic Equation
 - [5pts] Calculations
 - [2pts] Well organized output
 - [3pts] Error message printed when discriminant is negative
- [10pts] Exercise 4: Casting ints and chars
 - [7pts] Proper conversions
 - [3pts] Well organized output
- [15pts] Exercise 5: Change Maker
 - [10pts] Correct calculations, but must not use loops or if statements
 - [5pts] Well organized output
- [40pts] Exercise 6: Food stand
 - [10pts] Menu detects choices correctly
 - [10pts] Toppings menu activates only when first option selected
 - [10pts] Total and subtotals calculated correctly
 - [10pts] Output matches given example
- [5pts] Comments, coding styling, formatting
- [5pts] Debugger demonstration
 - Participate in an exercise or show your TA your now how to use the debugger

<http://www.sanfoundry.com/c-programming-examples/>

<https://www.programiz.com/c-programming/examples/vowel-consonant>

<http://www.thegeekstuff.com/2013/01/control-conditions-in-c/>

<http://www.techcrashcourse.com/2015/05/c-programming-if-else-statement.html>

<http://www.java-samples.com/showtutorial.php?tutorialid=251>

<http://www.learncpp.com/cpp-tutorial/52-if-statements/>

<https://www.quora.com/What-is-difference-between-a-debugger-and-a-compiler>