Overloading Functions & Command Line Use in C++

CS 16: Solving Problems with Computers I
Lecture #6

Ziad Matni
Dept. of Computer Science, UCSB

Announcements

- A reminder about Labs
  - Please make sure you READ the lab description BEFORE going to lab
  - Please make sure you understand the STYLING and REQUIREMENT parts of the lab
  - Please make sure to SIGN IN (or you will be counted as absent)

- Your 1st Midterm Exam is on THURSDAY (10/19)!!!
  - You didn't forget, did you?!

Lecture Outline

- Overloading Functions
- Command-line Arguments

- Midterm Review

MIDTERM #1 IS COMING! October 19th!

- Material: ***Everything*** we've done, **incl. up to Tue. 10/17**
  - Homework, Labs, Lectures, Textbook
- **Thursday, 10/19** in this classroom
- **Starts at 2:00pm **SHARP** (come early)**
- **Ends at 3:15pm **SHARP****
- **BRING YOUR STUDENT IDs WITH YOU!!!**
- Closed book: no calculators, no phones, no computers
- Only 1 sheet (single-sided) of written notes
  - Must be no bigger than 8.5" x 11"
  - **You have to turn it in with the exam**
- **You will write your answers on the exam sheet itself.**

What's on the Midterm#1? *From the Lectures, including…*

- Intro to Computers, Programming, and C++
- Variables and Assignments
- Boolean Expressions
  (comparison of variables)
- Input and Output on Standard Devices
  (cout, cin)
- Data Types, Escape Sequences,
  Formatting Decimal
- Arithmetic Operations and their Priorities
- Boolean Logic Operators
- Flow of Control & Conditional Statements
- Loops: for, while, do-while
- Types of Errors in Programming
- Multiway Branching and the switch command
- Generating Random Numbers
- Functions in C++:
  pre-defined, user-defined
  void functions, the main() function
  call-by-ref vs. call-by-value, overloading
- Command Line Inputs to C++ Programs
- *<u>No numerical conversions for Midterm 1!!!</u>*

A Note on Programming Style

When naming variables, functions, etc…

Styling Requirements for Labs
See announcement on Piazza

Not on Piazza yet?
MUST TELL ME ASAP!!!

- **Snake Case**: Using underscore character ('_')
  - Example: mortgage_amount function_fun()
  - Associated with C, C++ programmers
- **Camel Case**: Using upper-case letters to separate words
  - Example: MortgageAmount FunctionFun()
  - Associated with Java programmers
- For this class, YOU CAN USE EITHER!

Overloading Function Names

- C++ **allows more than one definition** for the **same function** name
  - Called “overloading”
  - Very convenient for situations in which the “*same*” function is needed for *different numbers* or *different types* of arguments

- ***Overloading a function name:***

  providing more than one declaration and definition
  using the same function name

Overloading Examples

BIG Pro-Tip:

Do not use the same function name for unrelated functions

```
double average(double n1, double n2)
{
            return ((n1 + n2) / 2);
}

double average(double n1, double n2, double n3)
{
     return (( n1 + n2 + n3) / 3);
}
```

- Compiler checks the number and types of arguments in the function call *and then decides which function to use automatically*!

- So, with a statement like: `cout << average( 10, 20, 30);`
  the compiler knows to use the ***second definition***

Overloading Rules in C++

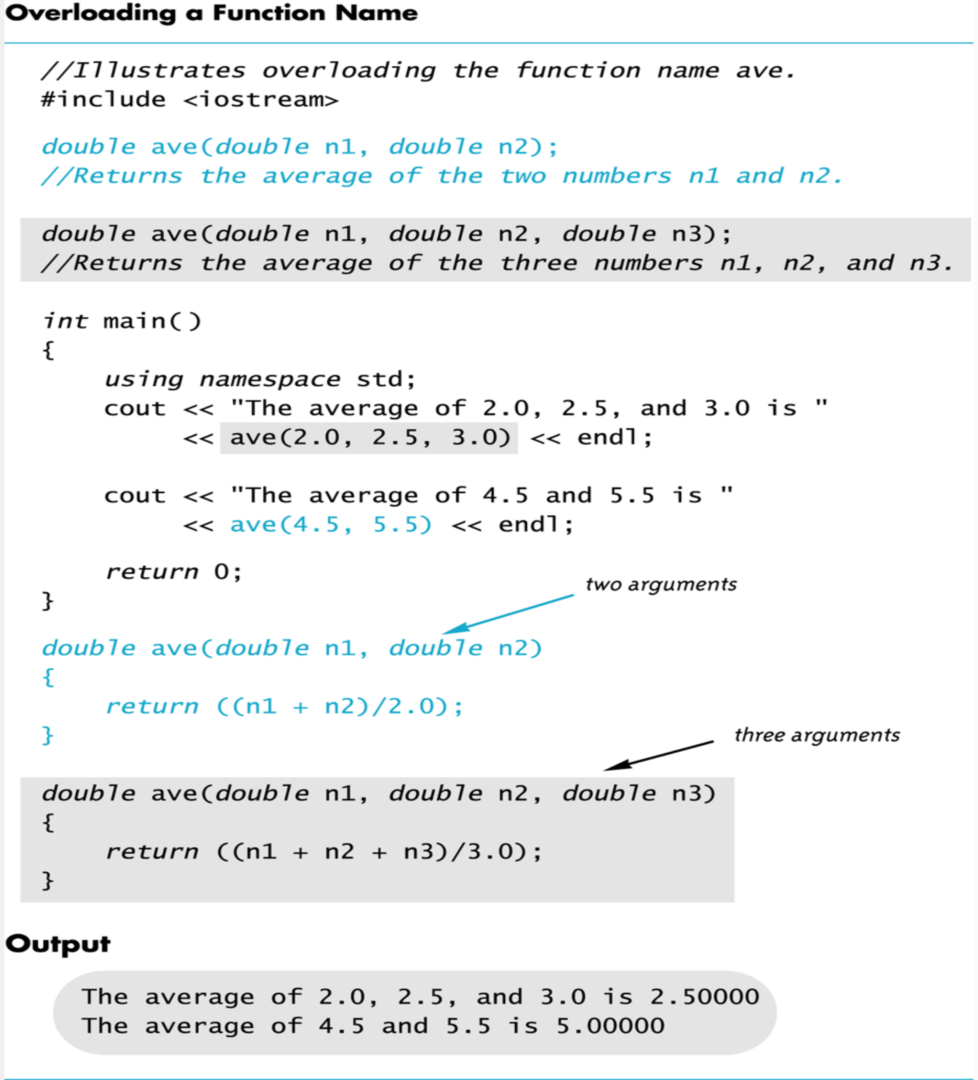
- Overloaded functions
  - Must have ***different numbers*** of formal parameters, **but must all be the same type**
    - e.g.: `double average(int a, int b)` *vs.* `double average(int a, int b, int c)`

OR

  - They can have the ***same number*** of parameters, **but must have at least one of them be of a *different type***
    - e.g.: `void print(int a)` *vs.* `void print(double a)` *vs.* `void print(char a)`

- You can not overload function declarations that differ *only* by return type.

Example from Textbook, Ch. 4



Overloading a Function Name

```
//Illustrates overloading the function name ave.
#include <iostream>

double ave(double n1, double n2);
//Returns the average of the two numbers n1 and n2.

double ave(double n1, double n2, double n3);
//Returns the average of the three numbers n1, n2, and n3.

int main()
{
    using namespace std;
    cout << "The average of 2.0, 2.5, and 3.0 is "
         << ave(2.0, 2.5, 3.0) << endl;

    cout << "The average of 4.5 and 5.5 is "
         << ave(4.5, 5.5) << endl;

    return 0;
}
```

two arguments

```
double ave(double n1, double n2)
{
    return ((n1 + n2)/2.0);
}
```

three arguments

```
double ave(double n1, double n2, double n3)
{
    return ((n1 + n2 + n3)/3.0);
}
```

Output

```
The average of 2.0, 2.5, and 3.0 is 2.50000
The average of 4.5 and 5.5 is 5.00000
```

Automatic Type Conversion

- C++ will automatically converts types between **int** and **double** in multiple examples
  - E.g. If I divide integers, I get integers: 13/2 = 6
  - E.g. If I make one of these a double, I get a double: 13/2.0 = 6.5
- It does the same with overloaded functions, for example, given the definition:

```
double mpg(double miles, double gallons) {
     return (miles / gallons);     }
```

  what will happen if **mpg** is called in this way?

```
cout << mpg(45, 2) << “ miles per gallon”;
```

- The values of the arguments will automatically be converted to type double (45.0 and 2.0): The answer will be: “22.5 miles per gallon”

Command Line Arguments with C++

- In C++ you can accept **command line arguments**
  - That is, when you execute your code, you can pass input values at the same time

- These are arguments that are passed *into* the program
  from the OS command line

- To use command line arguments in your program, you must add **2 special arguments** in the **main( )** function
  - Argument #1 is the number of elements that you are passing in: **argc** (reserved name)
  - Argument #2 is a full list of all of the command line arguments: ***argv[ ]** (reserved name, too)
    - This is an array pointer … more on those in a later class…

Command Line Arguments with C++

- The main() function should be written as:

```
    int main(int argc, char* argv[]) { … }
```

- In the OS, to execute the program, the command line form should be:

```
$ program_name  argument1 argument2 … argumentn
```

example:

```
$ sum_of_squares 4 5 6
```

Demo!

```
int main ( int argc, char *argv[] )
{
   cout << "There are " << argc << " arguments here:" << endl;

   for (int i = 0; i < argc; i++)
      cout << "argv[" << i << "] is : " << argv[i] << endl;

   return 0;
}
```

argv[*n*] Is Always a Character Type!

- While **argc** is always an int (it's calculated by the compiler)…
- …all you get from the command-line is **character arrays**
  - This is a hold-out from the early days of C
  - So, the data type of argument being passed is always an *array of characters* (a.k.a. a *C-string*)

- To treat an argument as another type (like a number, for instance), you have to first ***convert it inside your program***

- **<cstdlib>** library has pre-defined functions to help!

What If I Want an Argument That’s a Number?

- **<cstdlib>** library has pre-defined functions to help!
- Examples: **atoi( )**, **atol( )**, and **atof( )**
  Convert a **character array** into **int**, **long**, and **double**, respectively.

Example:

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main(int argc, char *argv[]) {
   for(int i = 1; i < argc; i++)
        cout << atoi(argv[i]) << endl;
   return 0;  }
```

Midterm Review

What's on the Midterm#1?
From the Lectures, including…

- Intro to Computers, Programming, and C++
- Variables and Assignments
- Boolean Expressions (comparison of variables)
- Input and Output on Standard Devices (cout, cin)
- Data Types, Escape Sequences, Formatting Decimal
- Arithmetic Operations and their Priorities
- Boolean Logic Operators
- Flow of Control & Conditional Statements
- Loops: for, while, do-while
- Types of Errors in Programming
- Multiway Branching and the switch command
- Generating Random Numbers
- Functions in C++:
  pre-defined, user-defined
  void functions, the main() function
  call-by-ref vs. call-by-value, overloading
- Command Line Inputs to C++ Programs
- *<u>No numerical conversions for Midterm 1!!!</u>*

Midterm Prep

1. Lecture slides
2. Homework problems
3. Lab programs
4. Book chapters 1 thru 5*

**check the lecture slides (from lectures 1 thru 6) with it!!*

Sample Question
Multiple Choice

Complete the following C++ code that is supposed to print the numbers 2, 3, 4, 5, 6:

```
int c = 0;
while (_____________) {
 cout << c+2 << “ ”;
 c++; }
```

A. `c < 7`

B. `c > 5`

C. `(c + 2)  < 6`

D. `(c + 2) != 6`

E. `c < 5`

Sample Question
Multiple Choice

What is the exact output of this C++ code?

```
int prod(1);
for (int m = 1; m < 7; m+=2) prod *= m;
cout << "Total product is: " << prod << endl;
```

A. Total product is: 720
B. Total product is: 105
C. Total product is: 48
D. Total product is: 15
E. Total product is: 1

Sample Question
Multiple Choice

Assuming precision is set to 1, the command **cout << static_cast<double>(5/2)** returns _____ to the display:

A. 5.0

B. 5.2

C. 2.0

D. 2 ½

E. 2.5

Sample Question
Multiple Choice

If a command line is used as follows ('**$**' is the command prompt):

```
$ myProgram 6 0 JokersWild
```

Then what is the value of **argv[0]**?

A. 6
B. 0
C. 4
D. "myProgram"
E. "JokersWild"

Sample Question
Short-Answer Coding

Write C++ code showing a function definition of **`distance( )`** which takes 4 **int** values `x1, x0, y1,` and `y0` and returns a **double** data type that's equal to

$\sqrt{(x_1-x_0)^2 + (y_1-y_0)^2}$.

Assume that the **cmath** lib has been imported.

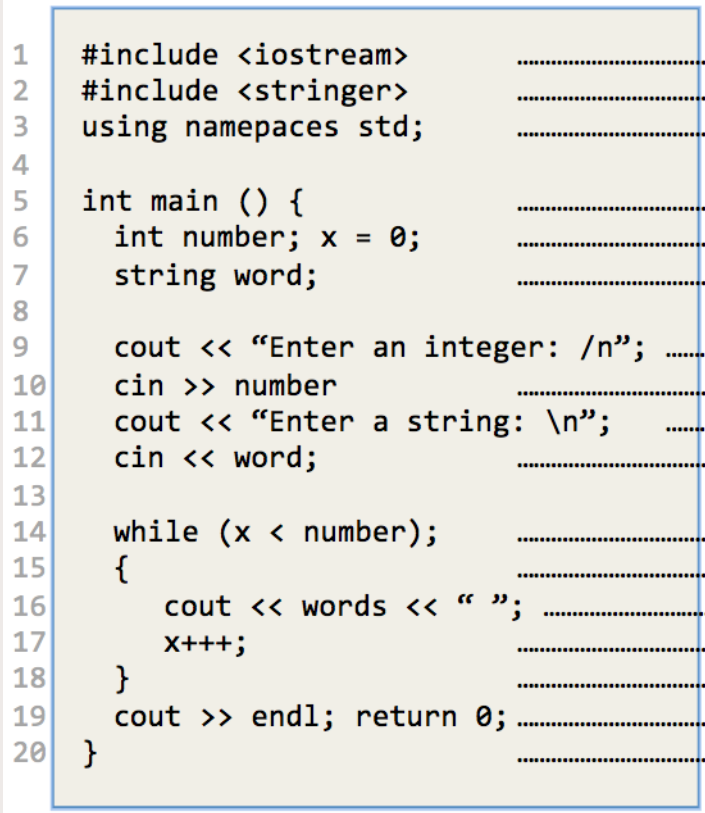
```
double distance(int x1, int x0, int y1, int y0)
{
    double a = pow(x1 – x0, 2);
    double b = pow(y1 – y0, 2);
    double z = sqrt(a + b);
    return z;
}
```

Note:
When I ask for "code", that means not a complete program.
Otherwise I'd ask for a "program". Also, this would be clear from the question.

That is, just look for syntax and logic errors

Sample Question

Find at Least 10 Mistakes *(ignore styling conventions)*



```
1  #include <iostream>
2  #include <stringer>
3  using namepaces std;
4
5  int main () {
6    int number; x = 0;
7    string word;
8
9    cout << “Enter an integer: /n”;
10   cin >> number
11   cout << “Enter a string: \n”;
12   cin << word;
13
14   while (x < number);
15   {
16      cout << words << “ ”;
17      x+++;
18   }
19   cout >> endl; return 0;
20 }
```

2: Should be: **<string>**

3: Should be: **using namespace std;**

6: Should be: **int number, x = 0;**

9: Should be: **\n**

10: Missing **;** at the end

11: Should be: **cin >> word;**

14: Must remove the **;** at the end

16: Should be: **cout << word << “ “;**

17: Should be: **x++**

19: Should be: **cout << endl; return 0;**

YOUR TO-DOs

- ❑ **STUDY FOR YOUR MIDTERM!!**
- ❑ Turn in HW3 on Thursday

- ❑ Lab 3 is NOT DUE UNTIL **MONDAY 10/23**!!!!
- ❑ Lab 4 follows the usual schedule. Starts Mon. 10/23; Due Fri. 10/27

- ❑ HW4 will be released on Thursday, will be due in 1 week.
- ❑ Visit Prof's and TAs' office hours if you need help!

- ❑ Good Luck!!!!

</LECTURE>