Intermediate Programming / C
CSCI 2212

Course Facts and Mechanics
Review of CS 110

August 24, 2015

Course Facts and Mechanics

C Programming style.

The Course Website

eliza.newhaven.edu/cprog2/

This website has several essential pages:

Homepage: This defines our contract; read it carefully.

Lectures: I will post the lecture before and corrections after class.

Homework: Pencil-and-paper homework will prepare you for exams.

Programs: The programming assignments are necessary to pass.
Tools, instructions and data files will be posted here.

Chapters: Links to the C textbook are on this page.

C++ Reference: The C++ topics will be integrated with C topics.

Course Facts and Rules (home page)

Course description: Prerequisites, books, outcomes.

Requirements: Expectations, exams, grades.

Policies: Communication, submitting work, attendance, late work.

Programs: The programming assignments are necessary to pass.

Academic honesty: The UNH policy and my own policy.

Getting the work done: Systems and assistance.

UNH Support Policies: Help, non-discrimination, special needs.

Reading due Friday (Homework page)

We want you to review the last few chapters you covered in the C book. We will also cover some of these topics in class. Do this review prior to doing the diagnostic test.

The new Freshmen who are not familiar with C I/O should also look at Chapters 3 and 4. Read the programs and make notes about anything you do not understand. Bring your notes to the session with your Tutor, Kevin.

Homework for Friday (Homework page)

We ask you to do a diagnostic test. Maybe you know everything on it – maybe not. However it is VERY important for you and for the teaching staff to know how much catching up you need to do. We have great TA's to tutor you. They need to know your specific needs.

So, PLEASE do this assessment paper promptly, and do it without help. Don't spend a lot of time searching references, but feel free to consult them for small questions.

The most important thing is to turn this in ON TIME, on Friday.

Program due Friday (Programming page)

Hello CSCI 2212

The purpose is for you to get started with some computer system capable of running both C and C++. If you have any problems, one of the TA's or Mark Morton, our department technician, will be able to help you.

Turn this in ON TIME, on Friday. Please, no excuses.

Example: Making Change.

Specifications
Test plan
Starting a new program
Coding style

Specifications

To write a program, start with a specification that is as precise as possible. Lists the information and requirements that your client (or teacher) has set out. The parts of a specification are:

- Goal: a brief statement in English about the general job you are trying to do.
- Inputs: what data will the user supply? What types? Are there minimum and maximum allowable values?
- Outputs: What answers must you compute, and how should they be displayed? How many decimal places are needed?
- Constants and formulas: mathematical equations needed to derive the output from the input, and constant values used in those formulas, if any.
- Other: Any other requirements the client provides.

Test Plan

Before you write even one line of code, construct a test plan. This is a list of inputs with the expected output or error comment that should be given for each. The inputs on this list must cover the following categories:

- Easy to process; you can figure out the answer in your head.
- The minimal and maximal acceptable inputs, if those are defined.
- Error inputs (too small, too large). Each error outcome should be tested independently.
- Typical inputs for a real situation. This is used to get your output formatting right.

The Top-Down Idea

Now you are ready to start designing a solution. Here is the general procedure, which will be expanded below.

- Start a new project folder and a new main program file.
- In it, write down the obvious parts of the program.
- Inside main, write comments for each major phase of the program.
- If a program phase is brief (less than 3 lines), write the code.
- As you work on main, write declarations for any variables you need.
- Invent and name a function for anything that requires more than a couple of lines of work.

The Top-Down Idea, continued

A function stub is a copy of the function's prototype, with nothing in the body except a trace comment and a return statement, if necessary.

- After the end of main, write a stub for every function you have named.
- Compile and debug your stubby program.
- Choose the easiest function. Write a real definition for it, with appropriate local variable declarations, and debug again.
- Now or later, you may find that you guessed wrong about an initial function prototype. You may find you need more parameters. If so, fix the prototype and the function's top line.
- Continue writing one function at a time and debug it using your easy test cases until the program is done.

The Top-Down Idea, finished

- Now pay attention to the program heading and your input prompts. Are they clear? Fix any problems.
- Look at your output. Is it neat and readable? Fix any problems you see.
- Perform a full test on the finished program.
- Use the mouse to capture the output and paste it into a file. Make sure the output is in the same folder as the program.
- Put your code through your IDE's re-indent process. Make sure the layout looks professional.