Introduction

Review from Monday’s Class and from Lab

• The standard library
• Scope and the scope operator
• Expressions and statements
• Variables, objects, types
• Strings
• const

Overview of Today’s Class

Koenig and Moo (K&M) Chapter 2:

• Strings: subscripting and type declarations
• An example printing problem
• Loops and loop invariants
• Conditionals
• Assignments and lvalues
• Operators

More on strings

• Strings behave like arrays when using the subscript operator []:
  – This gives access to the individual characters in the string
  – Subscript 0 corresponds to the first character.
  – Examples will be given in class.

• Strings define a special type, which is the type returned by the string function size():
The :: notation means that size_type is defined within the scope of the string type.

- string::size_type is generally equivalent to unsigned int.
- You will have compiler warnings and potential compatibility problems if you use int.

Example Problem: Writing a name along a diagonal

Here’s how the program might behance:

What is your first name? Chuck

**********
*   *
* C   *
* h   *
*   u *
*   c *
*   k *
*     *
**********

Here are some things we need to think about:

- Making sure that we can put the characters in the right places on the right lines.
- Getting the asterisks in the right positions and getting the right number of blanks on each line.

We will look at two different ways of solving this.

Solution Approach #1

- Initial stuff: read the name, and output a blank line, as in an earlier example.
- Let’s think about the main output: think of the output region as a grid of rows and columns:
  - How big is this region?
What gets output where?

- This will lead to an implementation with two nested loops, and conditionals used to guide where characters should be printed.

Details of Solution 1

The implementation of this solution is attached to this handout (diagonal_name.cpp) and will be available on the web site. We will use this to review and discuss a number of aspects of C++ and programming:

- While loops
- For loops
- If-else statements
- Conditionals
- Operators, precedence, and associativity
  - Arithmetic
  - Relationship
  - Logical
  - A variety of others

See the list on page 32.

- Integral types: int, unsigned int, long, bool, string::size_type

Loop Invariants

- Definition: a loop invariant is a logical assertion that is true at the start of each iteration of a loop.
  - In for loops, the “start” is defined as coming after the initialization/increment and termination test.
- An invariant is stated in a comment; it is not part of the actual code.
- It helps determine:
  - The conditions that may be assumed to be true at the start of each iteration.
What must be done in each iteration.
What must be done at the end of each iteration to restore the invariant.

• Analyzing the code relative to the stated invariant also helps explain the code and think about its correctness.

A Second Solution

Here are ideas for a second solution:

• Think about what changes from one line to the next.
• Suppose we had a “blank line” string, containing only the beginning and ending asterisks and the intervening blanks.
• We could overwrite the appropriate blank character, output the string, and then restore the blank character.

The code to implement this is attached.

Thinking about problem solving

• We have approached a relatively simple problem in two different ways:
  – Thinking of the output as a two-dimensional grid and using logical operations to figure out what to output at each location.
  – Thinking of the output as a series of strings, one string per line, and then thinking about the differences between lines.

• There are often many ways to solve a programming problem. Sometimes you can think of several, while sometimes you struggle to come up with one.

• When you have finished a problem or when you are thinking about someone else’s solution, it is useful to think about the core idea (or ideas) that were used. If you can abstract and understand these ideas, you can later apply them to other problems.