Introduction

Discussion of Syllabus

- Instructor, TAs, office hours

- A new CS II:
  - Early in the semester, we will emphasize a high-level of abstraction in problem solving: writing programs using the standard library, its container classes, and its functions
  - We will focus on implementation of container classes (data structures) later in the semester

- Overall expectations

- Prerequisites

- Two textbooks

- Class lectures

- Requirements:
  - Labs (15%),
  - Assignments (35%),
  - Tests (50%)

- Late policy

- Academic integrity

- Schedule of material

Overview of Today’s Class

- Koenig and Moo (K&M) Chapter 0: Getting Started

- Koenig and Moo (K&M) Chapter 1: Working with Strings

- Read these two chapters before Wednesday’s lab — not everything that is important will be covered in class.
Hello World

Here is the standard introductory program in almost any programming language:

```cpp
// a small C++ program
#include <iostream>

int main()
{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Small though it is, it may be used to illustrate a number of important points.

The standard library

- The standard library is not a part of the core C++ language. Instead it contains types and functions that are important extensions.
  - In our programming we will use the standard library to such a great extent that it will “feel” like part of the C++ core language.

- streams are the first component of the standard library that we see.
- std is a namespace that contains the standard library
- std::cout and std::endl are defined in the standard library

Expressions

- Each expression has a value and 0 or more side effects.
- An expression followed by a semi-colon is a statement. The semi-colon tells the computer to “throw away” the value of the expression.
- We will look carefully at
  ```cpp
  std::cout << "Hello, world!" << std::endl;
  ```
  which is really two expressions and one statement.
- "Hello, world!" is a string literal.
Scope

- The scope of a name is the part of the program in which it has meaning.

- The hello world program already illustrates two ways of establishing scope:
  - Curly braces, `{ }`
  - The operator `::`

Chapter 1: Strings

This chapter introduces an important object type from the standard library. Our discussion will focus on the example used in the text.

```cpp
#include <iostream>
#include <string>

int main()
{
    std::cout << "Please enter your first name: ";
    std::string name;
    std::cin >> name;

    // build the message that we intend to write
    const std::string greeting = "Hello, " + name + "!";

    // build the second and fourth lines of the output
    const std::string spaces( greeting.size(), ' ' );
    const std::string second = "* " + spaces + " *";

    // build the first and fifth lines of the output
    const std::string first(second.size(), '*');

    // write it all
    std::cout << std::endl;
    std::cout << first << std::endl;
    std::cout << second << std::endl;
    std::cout << greeting << std::endl;
    std::cout << second << std::endl;
    std::cout << std::endl;
    std::cout << std::endl;
    std::cout << std::endl;
    std::cout << std::endl;
```
std::cout << first << std::endl;

    return 0;
}

Variables, Objects and Types

• A variable is an object with a name

• An object is computer memory that has a type

• When a variable name goes out of scope, the name is “destroyed” and the memory used by the object associated with the name is returned for reuse.

string objects

• A string is an object type defined in the standard library to contain a sequence of characters.

• The string type, like all types, defines an interface, which includes construction (initialization), operations, functions (methods), and even other types(!).

• The greeting example code exhibits three ways of “constructing” string objects.

  – The notation

  greeting.size()

  is a call to a function size that is defined as a member function of the string class.

• Input to string objects through streams includes the following steps:

  1. The computer inputs and discards white-space characters, one at a time, until a non-white-space character is found.

  2. A sequence of non-white-space characters is input and stored in the string. This overwrites anything that was already in the string.

  3. Reading stops either at the end of the input or upon reaching the next white-space character (without reading it in).
• The (overloaded) operator ‘+’ is defined on strings. It concatenates two strings to create a third string.

• The assignment operation ‘=’ on strings overwrites the current contents of the string.

Other Notable C++ Ideas in Chapter 1

• Use const for a type whenever the object should not be changed after it is initialized.

• Understanding output buffering is important for efficiency and for debugging.

Scope Examples

To help reinforce your understanding of scope, here are two important program examples taken from the exercises at the end of Chapter 1 (with a small modification on the second). It is important that you understand why these programs are legal and why they behave the way they do.

```c++
#include <iostream>
#include <string>

int main()
{
    { const std::string s = "a string";
      std::cout << s << std::endl; }

    { const std::string s = "another string";
      std::cout << s << std::endl; }

    return 0;
}
```

```
#include <iostream>
#include <string>

int main()
{

```
```cpp
{ const std::string s = "a string";
  std::cout << s << std::endl;

  { const std::string s = "another string";
    std::cout << s << std::endl; }
  std::cout << s << std::endl; // what’s output here?
}

return 0;
}
```