Table of Contents

Chapter 1 - intro
Objects
Terminology - Classes
Terminology - Classes (cont’d)
Terminology - Definition & Declaration
Terminology - Scope
Terminology - Functions
C++ Features & Differences from C
Reference Parameters
Stream I/O

Assignment 0:
Chapter 1 Exercise 2
Due 8/4/00 11:59 pm
Objectives of Course

- advanced programing & language - C++
- command line operating system - Unix
- problem solving with advanced computer software

If you master these objectives your ability to programing will be significantly enhanced
CS 250 C++ Programming

Objects

⇒ paradigm of computing
⇒ code & data - results

The object is the basic programming structure in C++
class student

// data - referred to as member data
long id;
char lastName[30];
char firstName[30];

// functions - referred to as methods or member functions
void Input();
void Print();
float calcGrade();  

}; // end class student;
Terminology - Classes (cont’d)

⇒ classes can be linked
  ⇒ referring to member data
  ⇒ by calling class functions (methods)
  ⇒ by embedding - class can be a member of a class
  ⇒ by inheritance - derived from a base class
  ⇒ polymorphism - using different names for the same entity.

Inheritance Example

- mail
- letter
- package
Terminology - Definition & Declaration

⇒ definition - creates program entity which can be referred to by name (identifier)

Note: C has standard variable definitions - int, float, etc.

⇒ declaration - creates an instance of that entity

Definition:
struct z {
  float r;
} ZVar;

Declaration:
ZVar myZVar;

Note: Definitions do not use storage - int sum(int *Arry);
Terminology - Scope

⇒ “lifetime” or “visibility” of a declaration usually a variable

⇒ scope operator - “::” resolves to global scope variable

```cpp
int i, j; // has file scope
void sub(void) {
    int j, k; // block scope - note same a function scope
    auto int l; // block scope
    static int m; // block scope - maintains value between calls
    j = 3; // refers to local j
    ::j = 5; // refers to file scope j - “::” scope operator
    // rest of subroutine
} // end of sub
```
Terminology - Functions

⇒ in C++ functions are unique by their signature
⇒ function signature - both name and parameters

// In C error - redefinition because of identical names
// In C++ no problem - distinct function signatures
// prototypes

int Sum(int* theArray, int cnt);
float Sum(float* theArray, int cnt);

// Note that return value is not part of function signature
int Sum(float* theArray, int cnt);  // error in C or C++

Note: Operators act like functions lacking parentheses
C++ Features & Differences from C

- many new key words (see text page 9)
- some introduced in CS 150
- others will be intro’d as needed
- some may not be discussed in course
- underscore variables (don’t use) - int __sysName;
- casting, linking to C functions
- named constants (const qualification), enumerated constants
- reference variables
Reference Parameters

⇒ in functions - replaces using pointers
e.g., in function calls

```cpp
void swap(int& x, int& y) {
    int temp = x; // local var. temp set to x value
    x = y;       // the ref.’d x variable is set to y value
    y = temp;    // the ref.’d y is set to the temp value
}
```
Stream I/O

- `<<` - write operator, `>>` - read operator
- Standard files (as in C) - stdin, stdout, stderr
- `cin >> “variable name”` - reads variable value from standard input file named “cin”
- `cout << “var. name”` - writes variable value to standard output file named “cout”
- Variable name can be any variable type where an operator function exits
- Built in for all standard variables - int, float, etc.
Problem Solving - steps

1. State the problem clearly - this may be the hardest step
2. Describe the input and output
3. Work example problem by hand (or with calculator) to get check case
4. Develop and program solution in C++
5. Debug and test solution using 3

Step 5 may be the hardest part. Don’t just test for output, examine the program step by step when debugging.

Edison - “I know 92 ways not to make a light bulb.”