LESSON SET 1

Introduction to Programming and the Translation Process

OBJECTIVES FOR STUDENT

Lesson 1A:

1. To become familiar with the login process and the C++ environment used in the lab
2. To understand the basics of program design and algorithm development
3. To open, compile and run existing programs
4. To learn, recognize and correct both syntax and run time errors.

Lesson 1B:

5. To learn, recognize and correct logic errors
6. To independently develop a small program

ASSUMPTIONS

Lesson 1A:

1. Students have a minimal knowledge of the hardware (basic familiarity with I/O, disks, etc.)
2. Students will be somewhat familiar with the operating system.
   If this is not the case, the instructor may want to use the first lab as an introduction to the operating system environment used for the course, as well as teach some basic hardware concepts. This time is also given for students to learn the local system. Appendix A gives a very brief introduction to Visual C++ and Appendix B gives a brief introduction to UNIX. Teachers using different environments should prepare a lesson to introduce their system.

Lesson 1B:

1. Students have a basic knowledge of the environment. They can open, edit, compile and run simple programs.
2. For classes that introduce design early, it is assumed that students have been exposed to some design methodology to be used in developing their first simple program. If not, the design portion of the lab assignment should be skipped.

PRE-LAB WRITING ASSIGNMENT SOLUTIONS

1. syntax or grammar
2. logic
3. linker
4. Binary or Object
5. run time

LAB ASSIGNMENTS

Lesson 1A:

Lab 1.1: Opening, compiling and running your first program
Lab 1.2: Compiling a program with a syntax error
Lab 1.3: Running a program with a run time error

Lesson 1B:

Lab 1.4: Working with logic errors
Lab 1.5: Writing your first program (optional)

LESSON 1A

Lesson 1A is environmentally dependent. Use the following as a guide to assign the appropriate lab:

Visual C++ assign labs in Appendix A
Unix assign labs in Appendix B

If you are using an environment other than the two listed, you may have to orally walk the students through the environment and ask them to do Lab 1.1. You may want to also walk the students through the environment even if you are using one of the listed environments.

LAB 1.1: Opening, Compiling and Running Your First Program

This lab gives the student the experience of bringing in a program, compiling and running it.

LAB 1.2: Compiling a Program with a Syntax Error

This lab gives the student experience in bringing in a program and detecting a simple syntax error (missing semicolon).

A solution is found in `semiprobKey.cpp` in the instructor's folder for Lesson Set 1.

LAB 1.3: Running a Program with a Run Time Error

This lab gives the student experience in bringing in a program, compiling and detecting a run time error (dividing by 0).

A solution is found in `runprobKey.cpp` in the instructor's folder for Lesson Set 1.

LESSON 1B

LAB 1.4: Working with Logic Errors

Lab 1.4 creates the challenge of finding a logic error. This is the classic swap problem, which is unfamiliar to most students. This can be augmented by some discussion at the beginning of the lab on memory locations and the need for a temporary holding area. Even if the student is unable to correct the logical error, this is a valuable experience in demonstrating the complexity of such mistakes.

A solution is found in `logicprobKey.cpp` in the instructor's folder for Lesson Set 1.

LAB 1.5: Writing Your First Program (Optional)

Although many instructors will want the student generated code to be a vital component of all labs, this first set offers it as an optional exercise since some students may not yet be ready for this endeavor. Design methodologies and their introduction into a beginning course vary depending on the philosophy and time restraints of a particular institution. The design portion of this lab can be omitted, although it is recommended that future labs require some design for student generated code. This lab challenges the students to create their own code based on previously encountered software. As in Lab 1.4, valuable experience can be gained even if a successful program is not generated.

A solution is found in `kilotomilesKey.cpp` in the instructor's folder for Lesson Set 1.

Possible solutions to all labs are given in the instructor's folder for Lesson Set 1.

LESSON SET 2

Introduction to the C++ Programming Language

OBJECTIVES FOR STUDENT

Lesson 2A:

1. To learn the basic components of a C++ program
2. To gain a basic knowledge of how memory is used in programming
3. To understand the basic data types:
    a. Integer
    b. Character
    c. Float
    d. Boolean
    e. String (the `string` class is treated as a data type here)
4. To introduce the five fundamental instructions and to use the assign and output statements

Lesson 2B:

5. To develop a small program using simple C++ instructions
6. To work with characters and strings

ASSUMPTIONS

Lesson 2A:

1. Students have a basic knowledge of the programming environment. They can open, edit, compile and run simple programs.

Lesson 2B:

1. Students are familiar with the output and assignment statements in C++
2. Students are familiar with the general basic outline of a C++ program so that they can generate a simple program
3. Students are familiar with the basic data types including character and string (class treated as a data type)

PRE-LAB WRITING ASSIGNMENT SOLUTIONS

1. constant
2. Integer

3. Real or Floating point
4. Modulus or mod
5. output
6. Boolean
7. 8
8. comment
9. variable
10. string

LAB ASSIGNMENTS

Lesson 2A:

Lab 2.1: Working with the `cout` statement.
Lab 2.2: Working with constants, variables and arithmetic operators

Lesson 2B:

Lab 2.3: Rectangle area and perimeter
Lab 2.4 Working with characters and Strings

LESSON 2A

LAB 2.1: Working with the `cout` Statement

This is a simple lab that works with the `cout` statement.

A solution is found in `nameKey.cpp` in the instructor's folder for Lesson Set 2.

LAB 2.2: Working with Constants, Variables and Arithmetic Operators

This is a simple lab that continues to work with the `cout` statement and introduces the assignment statement.

A solution is found in `circleareaKey.cpp` in the instructor's folder for Lesson Set 2.

LESSON 2B

LAB 2.3: Rectangle Area and Perimeter

Although Lab 2.3 asks students to create a program from scratch, it is not labeled as optional since it is so similar to Lab 2.2 that most students should not find it too difficult.

A solution is found in `rectangleKey.cpp` in the instructor's folder for Lesson Set 2.

LAB 2.3: Working with characters and Strings

This lab introduces characters and the `string` class which is treated as a data type. The distinction of the `string` class from true data types is not explained until the student is

introduced to arrays of characters later in the manual.

A solution is found in `stringcharKey.cpp` in the instructor's folder for Lesson Set 2.

Possible solutions to all labs are given in the instructor's folder for Lesson Set 2.

LESSON SET 3

Expressions, Input, Output and Data Type Conversions

OBJECTIVES FOR STUDENT

Lesson 3A:

1. To work with input statements
2. To learn input and formatted output statements
3. To work with constants and mathematical functions

Lesson 3B:

4. To learn data type conversions (coercion and casting)
   These lessons introduce a variety of several key concepts in programming and are thus more comprehensive in content than most.

ASSUMPTIONS

Lesson 3A:

1. Students have a good understanding of the basic data types in C++
2. Students know the output, input and assignment statements in C++

Lesson 3B:

1. Students know the difference between constants and variables
2. Students know the concept of pre-defined functions in C++
3. Students know some basic concepts of file manipulation

PRE-LAB WRITING ASSIGNMENT SOLUTIONS

1. –20
2. `2*x+pow(3,4)`
3. coercion
4. casting
5. `#include <iostream>`
6. whitespace
7. insertion

8. `iomanip`
9. causes a carriage return (end of line)

LAB ASSIGNMENTS

As always, instructors should pick labs best suited for their particular class. All three labs of Lesson 3A cover the basics of the chapter and thus should be completed. Lab 3.5 is the student generated code component.

Lesson 3A:

Lab 3.1: Working with the `cin` statement
Lab 3.2: Formatting output
Lab 3.3: Arithmetic operations and math functions

Lesson 3B:

Lab 3.4: Working with type casting
Lab 3.5: Student generated code assignments

LESSON 3A

LAB 3.1: Working with the `cin` Statement

Lab 3.1 introduces the `cin` statement and shows the importance of prompts for interactive input.

Students may be confused about when to use the `<<` and `>>` operators. The difference should be reinforced.

Students may have to experiment for awhile to realize that at times a `cout` statement may need to begin with `cout << endl <<...` This is sometimes true when a `cout` statement comes right after a `cin` statement.

A solution is found in `billKEY1.cpp` in the instructor's folder for Lesson Set 3.

Possible Answers to the Exercise Questions

Exercise 2:

1. Rerun the program with the same data given in Exercise 1 above and record your results.
   4e+002 (or something to this effect). It puts the answer in scientific notation.

2. What do you think the fixed attribute in the `cout` statement does?
   It puts the answer in standard decimal notation instead of scientific notation.

Exercise 3:

1. Rerun the program with the same data given in Exercise 1 and record your results:
   $241.5600

2. What do you think the `setprecision( )` attribute in the `cout` statement does?
   It determines how many digits to the right of the decimal point are shown.

Exercise 4:
This is an optional section that introduces the `string` class (used as a data type). A solution is found in billKEY2.cpp in the instructor's folder for Lesson Set 3.

LAB 3.2: Formatting Output

Lab 3.2 works with formatted output and follows very closely the material learned in Lab 3.1. Thus it is important that 3.1 and 3.2 are done in sequence.

A solution is found in `tabledataKey.cpp` in the instructor's folder for Lesson Set 3.

LAB 3.3: Arithmetic Operations and Math Functions

This lab has the students convert the Pythagorean formula into a C++ expression. It requires the use of the `pow` and `sqrt` math functions.

A solution for Exercise 1 is found in `righttrigKey1`.cpp in the instructor's folder for Lesson Set 3.

In Exercise 2 of this lab the students need to alter the program by including the directive `<iomanip>`. They also need to include the statement:

```
cout << fixed << showpoint << setprecision(2);
```

just before printing the length of the hypotenuse.

A solution for Exercise 2 is found in `righttrigKEY2.cpp` in the instructor's folder for Lesson Set 3.

LESSON 3B

Lab 3.4: Working with Type Casting

This lab works with the classic batting average problem. Since at bats will always be less than hits (unless we really have a superstar), the average obtained by dividing by these two integers will always be 0. This lab demonstrates both type casting and coercion.

A solution is found in `batavgKEY.cpp` in the instructor's folder for Lesson Set 3.

Possible Answers to the Exercise Questions

Exercise 1:
Run this program and record the results.
The batting average is 0.

Exercise 2:
There is a logical error in this program centering around data types. Does changing the data type of `batavg` from `int` to `float` solve the problem?
No. It also needs type casting.

Make that change and run the program again and record the result.
The batting average is 0.292162.