**LESSON SET 11**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Structures and Abstract Data Types**

**OBJECTIVES FOR THE STUDENT**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lesson 11A:**

1. To introduce the concept of an abstract data type

2. To introduce the concept of a structure

3. To develop and manipulate an array of structures

**Lesson 11B:**

4. To use structures as parameters

5. To use hierarchical (nested) structures

**ASSUMPTIONS**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This lab assumes that the student has a basic understanding of arrays, functions,

data types, and nested logic. It is not assumed that students know anything about

object-oriented programming or abstract data types at this point. However, we do

introduce this lesson as a precursor to the object-oriented paradigm.

**Lesson 11A**:

1. Students have been introduced to the basic concept of a structure

2. Students have a solid background in arrays, thus enabling them to work with array of

structures

**Lesson 11B:**

1. Students have a solid background with functions and nested logic

**PRE-LAB WRITING ASSIGNMENT SOLUTIONS**

1. tag

2. different

3. nested structure

4. members

5. uninitialized

6. abstract data type

7. subscript or index

8. dot operator

9. structure declaration

10. function

**LAB ASSIGNMENTS**

**Lesson 11A:**

Lab 11.1: Working with basic structures

Lab 11.2: Initializing structures

Lab 11.3: Arrays of structures

**Lesson 11B:**

Lab 11.4: Nested structures

Lab 11.5: Student generated code assignments

**LESSON 11A**

**LAB 11.1: Working with Basic Structures**

Lab 11.1 requires the student to use program rect\_struct.cpp from the Lab 11

folder. This program uses a structure to hold data about a rectangle. For Exercise

1 students are asked to fill in missing code segments. In particular, students are

asked to declare a structure named rectangle with five members, define a

structure variable of rectangle type, read certain quantities into structure members,

and then output some structure members. In Exercise 2 students are asked

to modify the program so that it determines whether or not the rectangle input

by the user is a square. Of course, all that is really required is the comparison of

two structure members, coupled with an if statement and the appropriate output

message.

A solution program can be found in rect\_structKey.cpp in the instructor’s folder for Lesson Set 11.

**LAB 11.2: Initializing Structures**

Lab 11.2 requires the student to use program init\_struct.cpp from the Lab 11

folder. This program demonstrates partially initialized structure variables. Students

are asked to fill in missing code segments. They will need to be able to initialize

the first three structure members of two structure variables while leaving the

last two members un-initialized. They will also be required to add read statements

involving one of the structure members. If a student has successfully completed

Lab 11.1, Lab 11.2 should be straightforward and easy to complete.

A solution program can be found in init\_structKEY.cpp in the instructor’s folder for

Lesson Set 11.

**LAB 11.3: Arrays of Structures**

Lab 11.3 requires the student to use program array\_struct.cpp from the Lab 11

folder. Like the first two labs, students are asked to fill in missing code involving

a structure declaration. However, in this lab they are also required to define an

array of structures. In addition, there is a looping statement that must be added

at the end of the program. In the second exercise students are asked to address

an indexing issue regarding arrays ( i.e., if we have a list indexed by 1,2,3, . . .

then we have to compensate for the fact that the array holding this information

must start with 0). Of course, this is something that has been addressed repeatedly

in earlier lessons. However, it is useful for students to recall this point several

times throughout the semester.

A solution program can be found in array\_structKey.cpp in the instructor’s folder for Lesson Set 11.

**LESSON 11B**

**LAB 11.4: Nested Structures**

Lab 11.4 requires the student to use program nestedRect\_struct.cpp from the

Lab 11 folder. Like the first three labs, students are asked to fill in missing code

involving structure declarations. The first is a structure named dimensions that

contains two members of type float. The second is a structure named rectangle

that contains two float members and a third member of type dimensions.

In other words, they should nest a dimensions structure inside of a rectangle

structure. In the second exercise students are asked to modify the program so that

the rectangle structure has structure variables for both of its members. The

third exercise requires structure variables to be passed as arguments to a pair of

functions.

A solution program can be found in nestedRect\_structKey.cpp in the instructor’s folder for Lesson Set 11.

**LAB 11.5: Student Generated Code Assignments**

*Option 1:* The first option is for students to re-write Sample Program 11.2 so

that it works for an array of structures. The program should compare 6

circles rather than 2. In addition to testing their knowledge on arrays of

structures, this program makes students come up with a way of determining

which circle’s center is closest to the origin. Sample Program 11.2 does

this for two circles. However, students will need to add a slightly different

algorithm to make this work for 6 circles.

A solution program can be found in circleopt1KEY.cpp in the instructor’s folder for Lesson Set 11.

*Option 2:* The second option asks the student to write a program from scratch. In particular, they need to write a program that uses a structure to store the following information for a particular month at the local airport:

Total number of planes that landed

Total number of planes that departed

Greatest number of planes that landed in a given day that month

Least number of planes that landed in a given day that month

The program should have an array of twelve structures to hold travel information

for the entire year. It should prompt the user to enter data for each month. Once

all data is entered, the program should calculate and output the average monthly

number of landing planes, the average monthly number of departing planes,

the total number of landing and departing planes for the year, and the greatest

and least number of planes that landed on any one day (and which month it

occurred in). This program may take most students more than 30 minutes to

complete. Instructors should allow students time outside of the lab to complete

this program if assigned. If it is important that the student be able to complete the

assignment during the lab time, the instructor may wish to assign Option 1rather

than2.

A solution program can be found in airportKEY.cpp in the instructor’s folder for Lesson Set 11.