Lab 7: Structs

"C++ allows you to group several variables together under a single item known as a structure." -Tony Gaddis, Starting Out With C++

```
#include <iostream>
using namespace std;

// Struct Definition (Think blueprint)
struct StudentRecord
{
    // Member Variables
    int studentID;
    string firstName;
    string lastName;
    double gpa;
};  // the semicolon is required
```

```
int main()
{
   // Variable Declaration (or more accurately object declaration)
   //
   // Like "string", "StudentRecord" is an abstract data type,
   // and so we must create an instance of it when we want to use one.
   // Here, "StudentRecord" is our abstract data type, and "student" is
   // the name of our newly created object.
   //
   // An object is  similar to a variable, but while a variable contains
   // only a single piece of data, an object is comprised of multiple
   // pieces of data with often different data types (later you will learn
   // structures also can contain functions).

   StudentRecord student;

   // Accessing an Object's Members
   //
   // We use the dot operator (.) to access member variables.
   // syntax : objectName.memberName

   student.studentID = 12321;
   student.firstName = "John";
   student.lastName = "Snow";
   student.gpa = 3.76;

   cout << student.studentID << endl
        << student.firstName << endl
        << student.lastName << endl
        << student.gpa << endl;

   return 0;
}
```

Initializing a Struct

Declaring and Assigning at the same time

```
#include <iostream>
using namespace std;

// Struct Definition
struct Time
{
   int hour;
   int minutes;
   int seconds;
};

int main()
{
   // Struct Object Initialization
   Time myTime = {12, 32, 45};
   cout << myTime.hour << ":" << myTime.minutes << ":" << myTime.seconds;
}
```

Structs Example 1

```
#include <iostream>
using namespace std;

// Struct Definition
struct Time
{
   int hour;
   int minutes;
   int seconds;
};

int main()
{
   // Struct Object Declaration
   Time start_time;
   Time end_time;

   // Struct Object Assignment of values
   cout << "What is the start time (H M S)?" << endl;
   cin >> start_time.hour;
   cin >> start_time.minutes;
   cin >> start_time.seconds;

   // Struct Object Assignment of values
   cout << "What is the end time (H M S)?" << endl;
   cin >> end_time.hour;
   cin >> end_time.minutes;
   cin >> end_time.seconds;

   cout << "\nStart: " << start_time.hour << ":" << start_time.minutes
        << "." << start_time.seconds << endl;
   cout << "End  : " << end_time.hour << ":" << end_time.minutes << "."
        << end_time.seconds;
   return 0;
}
```

Arrays of Structs

```
#include <iostream>
using namespace std;

// Struct Definition
struct StudentRecord
{
  // Members
  int studentID;
  string firstName;
  string lastName;
  double gpa;
};


int main()
{
   // Array of Objects Declaration
   StudentRecord student [10000];

   // Accessing members of an element in an array of structs
   student[0].firstName = "Cersei";
   student[0].lastName = "Lannister";
   student[0].studentID = 543345;
   student[0].gpa = 3.9;

   cout << student[0].studentID << endl
        << student[0].firstName << endl
        << student[0].lastName << endl
        << student[0].gpa << endl;

 return 0;
}
```

Structs With Member Arrays

```
#include <iostream>
using namespace std;

// Struct Definition
struct StudentRecord
{
  // Members
  string firstName;
  string lastName;
  int testGrades[4]; // We can have members that are arrays
};


int main()
{
   // Array of Objects Declaration
   StudentRecord student [10000];

   cout << "Enter student's first name" << endl;
   cin >> student[0].firstName;

   cout << "Enter student's last name" << endl;
   cin >> student[0].lastName;

   cout << "Enter the student's 4 test grades" << endl;

   // Here we use a FOR loop to get each grade for student "0".
   // Note that this will only fill the array for student "0".
   for(int i = 0; i < 4; i++)
   {
      cin >> student[0].testGrades[i];
   }


   return 0;
}

// Note that this code only assigns values to 1 of the 10000 StudentRecord
// array elements that we created. To fill the other values, we would have to
// use an outer loop around the given FOR loop to repeat this code for the
// other students.
```

Structs Example 2

```
// Reads in students from user (keyboard) into an array
// of structs and outputs them to a file.

#include <iostream>
#include <fstream>
using namespace std;

// Struct Definition
struct StudentRecord
{
   int studentID;
   string firstName;
   string lastName;
   double gpa;
};


int main()
{
    // Variable Array Declaration
    StudentRecord student [1000];
    int count;
    char userChoice;

    ofstream fout;
    // We can set our ofstream to append to the end of the file so we do not
    // lose previously entered data by using std::ofstream::app as demonstrated.
    // This is just for your knowledge and will not be tested over.
    fout.open("StudentData.txt", std::ofstream::app);

    count = 0;
   //continues on next page
```

```
// Example 2 continued
// Take student info from user and put it into the array of Structs.
    do
    {
        cout << "Enter Student ID: ";
        cin >> student[count].studentID;

        cout << "Enter Student's First Name: ";
        cin >> student[count].firstName;

        cout << "Enter Student's Last Name: ";
        cin >> student[count].lastName;

        cout << "Enter Student's GPA: ";
        cin >> student[count].gpa;

        cout << "Are you finished? (Y or y)" << endl;
        cin >> userChoice;
        cout << endl;

        count++;
    }
    while(userChoice != 'Y' && userChoice != 'y');


    for(int i=0; i<count; i++)
    {
        fout << student[i].studentID << " " << student[i].firstName
             << " " << student[i].lastName << " " << student[i].gpa
             << endl;
    }

    fout.close();

    return 0;
}
```