Lab 12

LAB 12.1 Introduction to Files ("Revision")

This exercise is good for those needing a review of basic file operations, which was covered earlier.

Retrieve program `files.cpp` from the Lab 12 folder. The code is as follows:

```
// This program uses hours, pay rate, state tax and fed tax to determine gross
// and net pay.

// PLACE YOUR NAME HERE

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

int main()
{
   // Fill in the code to define payfile as an input file

   float gross;
   float net;
   float hours;
   float payRate;
   float stateTax;
   float fedTax;

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

   // Fill in the code to open payfile and attach it to the physical file
   // named payroll.dat


   // Fill in code to write a conditional statement to check if payfile
   // does not exist.
   {
      cout << "Error opening file. \n";
      cout << "It may not exist where indicated" << endl;
      return 1;
   }

   cout << "Payrate    Hours  Gross Pay    Net Pay"
        << endl << endl;

   // Fill in code to prime the read for the payfile file.

   // Fill in code to write a loop condition to run while payfile has more
   // data to process.
   {
      payfile >> payRate >> stateTax >> fedTax;

      gross = payRate * hours;

      net = gross - (gross * stateTax) - (gross * fedTax);

      cout << payRate << setw(15) << hours << setw(12) << gross
           << setw(12) << net << endl;

      payfile >> // Fill in the code to finish this with the appropriate
                 // variable to be input
```

```
    }

    payfile.close();

    return 0;
}
```

Exercise 1

Assume that the data file has `hours`, `payRate`, `stateTax`, and `fedTax` on one line for each employee. `stateTax` and `fedTax` are given as decimals (5% would be .05). Complete this program by filling in the code (places in bold).

Exercise 2

Run the program. Note: the data file does not exist so you should get the error message:

```
Error opening file.
It may not exist where indicated.
```

Exercise 3

Create a data file with the following information:

40	15.00	.05	.12
50	10	.05	.11
60	12.5	.05	.13

Save it in the same folder as the `.cpp` file. What should the data file name be?

Exercise 4

Run the program. Record the output here:

_____

_____

_____

Exercise 5

Change the program so that the output goes to an output file called `pay.out` and run the program. You can use any logical internal name you wish for the output file.

LAB 12.2 Files as Parameters and Character Data

Retrieve program `Grades.cpp` and the data file `graderoll.dat` from the Lab 12 folder. The code is as follows:

```
#include // FILL IN DIRECTIVE FOR FILES
#include <iostream>
#include <iomanip>
using namespace std;

// This program reads records from a file. The file contains the
// following: student's name, two test grades and final exam grade.
// It then prints this information to the screen.

const int NAMESIZE = 15;
const int MAXRECORDS = 50;
```

```
struct Grades    // declares a structure
{
   char name[NAMESIZE + 1];
   int test1;
   int test2;
   int final;
};

typedef Grades gradeType[MAXRECORDS];
// This makes gradeType a data type
// that holds MAXRECORDS
// Grades structures.

// FILL IN THE CODE FOR THE PROTOTYPE OF THE FUNCTION readIt
// WHERE THE FIRST ARGUMENT IS AN INPUT FILE, THE SECOND IS THE
// ARRAY OF RECORDS, AND THE THIRD WILL HOLD THE NUMBER OF RECORDS
// CURRENTLY IN THE ARRAY.

int main()
{
   ifstream indata;

   indata.open("graderoll.dat");

   int numRecord;      // number of records read in

   gradeType studentRecord;

   if (!indata)
   {
      cout << "Error opening file. \n";
      cout << "It may not exist where indicated" << endl;

      return 1;
   }

   // FILL IN THE CODE TO CALL THE FUNCTION readIt.

   // output the information
   for (int count = 0; count < numRecord; count++)
   {
      cout << studentRecord[count].name << setw(10)
           << studentRecord[count].test1
           << setw(10) << studentRecord[count].test2;
      cout << setw(10) << studentRecord[count].final << endl;
   }

   return 0;
}

//************************************************************
// readIt
//
// task:    This procedure reads records into an array of
//           records from an input file and keeps track of the
//           total number of records
// data in:  data file containing information to be placed in
//           the array
// data out: an array of records and the number of records
//
//************************************************************

void readIt// FILL IN THE CODE FOR THE FORMAL PARAMETERS AND THEIR
        // DATA TYPES.
        // inData, gradeRec and total are the formal parameters
        // total is passed by reference)
{
   total = 0;
```

```
   inData.get(gradeRec[total].name, NAMESIZE);

   while (inData)
   {
      // FILL IN THE CODE TO READ test1

      // FILL IN THE CODE TO READ test2

      // FILL IN THE CODE TO READ final

      total++;  // add one to total

      // FILL IN THE CODE TO CONSUME THE END OF LINE

      // FILL IN THE CODE TO READ name
   }

}
```

Exercise 1

Complete the program by filling in the code (areas in bold). This problem requires that you study very carefully the code and the data file already written to prepare you to complete the program. Notice that in the data file the names occupy no more than 15 characters. **Why?**

Exercise 2

Add another field called `letter` to the record which is a character that holds the letter grade of the student. This is based on the average of the grades as follows: `test1` and `test2` are each worth 30% of the grade while `final` is worth 40% of the grade. The letter grade is based on a 10 point spread. The code will have to be expanded to find the average.

90–100	A
80–89	B
70–79	C
60–69	D
0–59	F

LAB 12.3 Binary Files and the `read/write` Functions

Retrieve program `budget.cpp` from the Lab 12 folder. The code is as follows:

```
// This program reads in from the keyboard a record of financial information
// consisting of a person's name, income, rent, food costs, utilities and
// miscellaneous expenses. It then determines the net money
// (income minus all expenses) and places that information in a record
// which is then written to an output file.

// PLACE YOUR NAME HERE

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

const int NAMESIZE = 15;

struct budget   // declare a structure to hold name and financial information
{
   char name[NAMESIZE + 1];
   float income;              // person's monthly income
   float rent;                // person's monthly rent
```

```
   float food;                // person's monthly food bill
   float utilities;           // person's monthly utility bill
   float miscell;             // person's other bills
   float net;                 // person's net money after bills are paid
};

int main()
{
   fstream indata;
   ofstream outdata;          // output file of
                              // student.

   indata.open("income.dat", ios::out | ios::binary);    // open file as binary
                                                         // output.

   outdata.open("student.out");      // output file that we
                                     // will write student
                                     // information to.

   outdata << left << fixed << setprecision(2);   // left indicates left
                                                  // justified for fields

   budget person;      // defines person to be a record

   cout << "Enter the following information" << endl;

   cout << "Person's name: ";
   cin.getline(person.name, NAMESIZE);

   cout << "Income :";
   cin >> person.income;

   // FILL IN CODE TO READ IN THE REST OF THE FIELDS:
   // rent, food, utilities AND miscell TO THE person RECORD

   // find the net field
   person.net = // FILL IN CODE TO DETERMINE NET INCOME (income - expenses)

   // write this record to the file
   // Fill IN CODE TO WRITE THE RECORD TO THE FILE indata (one instruction)

   indata.close();

   // FILL IN THE CODE TO REOPEN THE indata FILE, NOW AS AN INPUT FILE.

   // FILL IN THE CODE TO READ THE RECORD FROM indata AND PLACE IT IN THE

   // write information to output file
   outdata << setw(20) << "Name" << setw(10) << "Income" << setw(10) << "Rent"
           << setw(10) << "Food" << setw(15) << "Utilities" << setw(15)
           << "Miscellaneous" << setw(10) << "Net Money" << endl << endl;

   // FILL IN CODE TO WRITE INDIVIDUAL FIELD INFORMATION OF THE RECORD TO
   // THE outdata FILE.(several instructions)

   return 0;
}
```

Exercise 1

This program reads in a record with fields `name`, `income`, `rent`, `food`, `utilities`, and `miscell` from the keyboard. The program computes the `net` (income minus the other fields) and stores this in the net field. The entire record is then written to a binary file (`indata`). This file is then closed and reopened as an input file. Fill in the code as indicated by the comments in bold.

Sample Run (user input underlined):

```
Enter the following information
Person's Name: Billy Berry
Income: 2500
Rent: 700
Food: 600
Utilities: 400
Miscellaneous: 500
```

The program should write the following text lines to the output file `student.out`.

```
Name          Income      Rent      Food     Utilities   Miscellaneous  Net Money
Billy Berry   2500.00     700.00    600.00   400.00      500.00         300.00
```

Exercise 2

Alter the program to include more than one record as input. Use an array of records.

Sample Run (user input underlined):

```
Enter the following information
Person's Name: Billy Berry
Income: 2500
Rent: 700
Food: 600
Utilities: 400
Miscellaneous: 500

Enter a Y if you would like to input more data
Y

Enter the following information
Person's Name: Terry Bounds
Income: 3000
Rent: 750
Food: 650
Utilities: 300
Miscellaneous: 400

Enter a Y if you would like to input more data
n
That's all the information
```

The output program `student.out` should then have the following lines of text written to it.

```
Name          Income      Rent      Food     Utilities   Miscellaneous  Net Money
Billy Berry   2500.00     700.00    600.00   400.00      500.00         300.00
Terry Bounds  3000.00     750.00    650.00   300.00      400.00         900.00
```

LAB 12.4 Random Access Files

Retrieve program `randomAccess.cpp` and the data file `proverb.txt` from the Lab 12 folder. The code is as follows:

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

// PLACE YOUR NAME HERE
```

```
int main()
{
   fstream inFile("proverb.txt", ios::in);
   long offset;
   char ch;
   char more;

   do
   {
      // Fill in the code to write to the screen
      // the current read position (with label)

      cout << "Enter an offset from the current read position: ";
      cin >> offset;

      //  Fill in the code to move the read position "offset" bytes
      //  from the CURRENT read position.

      //  Fill in the code to get one byte of information from the file
      //  and place it in the variable "ch".

      cout << "The character read is " << ch << endl;
      cout << "If you would like to input another offset enter a Y"
           << endl;
      cin >> more;

      // Fill in the code to clear the eof flag.

   } while (toupper(more) == 'Y');

   inFile.close();

   return 0;
}
```

Exercise 1

Fill in the code as indicated by the comments in bold.

The file `proverb.txt` contains the following information:

Now Is The Time fOr All GoOd Men to come to the aid of their Family

Sample Run (user input underlined):

The read position is currently at byte 0
Enter an offset from the current position: 4
The character read is I
If you would like to input another offset enter a Y y

The read position is currently at byte 5
Enter an offset from the current position: 2
The character read is T
If you would like to input another offset enter a Y y

The read position is currently at byte 8
Enter an offset from the current position: 6
The character read is e
If you would like to input another offset enter a Y y

The read position is currently at byte 15
Enter an offset from the current position: 44
The character read is r
If you would like to input another offset enter a Y y

```
The read position is currently at byte 60
Enter an offset from the current position: 8
The character read is r
If you would like to input another offset enter a Y n
```

Exercise 2
Why do you think that the character printed at the last run was another r? What would you have to do to get a different letter after the position is beyond the `eof` marker?

Exercise 3
Change the program so that the read position is calculated from the end of the file. What type of offsets would you need to enter to get characters from the proverb? Do several sample runs with different numbers to test your program.

LAB 12.5 Student Generated Code Assignments

Option 1:

Write a program that will read the radii of circles. Use an array of records where each record will have the radius of the circle read from the keyboard and the diameter and area of the circle will be calculated by the program. This information (radius, diameter and area) is stored in a binary file. The information in the binary file is then read back into the records and stored in a text output file. Left justify the information for each field. Each record will consist of the following fields:

radius	float
diameter	float
area	float

You may assume a maximum of 20 records. You may want to include the `cmath` library.
You need to know the formulas for finding the area and circumference of a circle.

This assignment is very similar to the program in Lab 12.3.

Sample Run (user input underlined):

```
Enter the following information:
Radius of circle: 5
Enter a Y if you would like to input more data
Y

Enter the following information:
Radius of circle: 4
Enter a Y if you would like to input more data
Y

Enter the following information:
Radius of circle: 7
Enter a Y if you would like to input more data
n

That's all the information.
```

The output file contains the following:

```
Radius      Area        Circumference
5.00        78.54       31.42
4.00        50.27       25.13
7.00        153.94      43.98
```

Option 2:

Bring in the file `employee.in` from Lab 12 folder. Write a program that will read records from this file and store them in a binary file. That file will then be used as input to create an output file of the information. The data file contains employee information consisting of name, social security, department ID, years employed, and salary. In addition to displaying the information of each record, the program will also calculate the average salary and years employed of all the records. This additional information is stored in the same output file.

Sample Data File:

Bill Tarpon	182460678	789	8	30600
Fred Caldron	456905434	789	10	40700
Sally Bender	203932239	790	8	50000
David Kemp	568903493	790	9	60000

The output file should look like this:

```
Name              Social Security   Department ID     Years Employed    Salary
Bill Tarpon       182460678         789               8                 30600
Fred Caldron      456905434         789               10                40700
Sally Bender      203932239         790               8                 50000
David Kemp        568903493         790               9                 60000

The average number of years employed is 8

The average salary is $45325.00
```