جامعة الإمام عبد الرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

CS221 - Lab 6:

Structure

Objective(s)

1. Learn how to declare and use structure.
2. Passing structures to functions.
3. To use array of structures.

Tool(s)/Software DevC++.

Description

Refer to the handout in the blackboard.

Tasks/Assignments(s)

Task #1: Compute the area of Rectangle:

Compute the area of a rectangle using the width and height.

1. **Create the Rectangle struct.**

```
struct Rectangle
{       double width;
        double height;
        double area;
};
```

In the main function add the height and width by asking the user to enter them. Then print the area, after calling the function.

2. **Overload computeArea Function in two different ways:**

```
double computeArea(double w, double h)
```

```
Rectangle computeArea(Rectangle r)
```

3. **Change the computeArea to be called by reference .. what do you need to change?**

```
void computeArea(Rectangle &r)
```

Task #2: HW Add Two Fractions

Write a C++ program to add two fractions and display the result fraction. Your program will prompt the user to input fraction 1 and fraction 2. Each fraction is in the form of two numbers: numerator and denominator of each fraction.

- Use this definition of struct and function:

```
struct Fract

     {   int num;

         int deno;

      };

Fract sum(Fract,Fract);
```

- Sample Run:

```
Enter fraction 1(numerator denominator): 1 2

Enter fraction 2(numerator denominator): 2 5

Result: 9/10
```

Task #3: Students Records (Array of struct)

Write a C++ program to keep records and perform statistical analysis for a class of 20 students. The information of each student contains ID, Name, gender, total score, and grade.

- The student struct should be:

```
struct Student
{
string id;
string name;
char gender;
Date dob;
char grade ;
double score;
} st[20];
```

```
struct Date
{
int date;
int month;
int year;
};
```

The program will prompt the user to choose a task from a menu as shown below:

Menu:
1 - Add a student
2 - Delete a student
3 - Update a student score
4 - View all records
5 - Find the max score
6 - Find a student by ID
7 - Sort records by scores
8 - Show students by grade
9 - Show students by gender
10 - Show students born in month ##

Use separate function for each operation. For example:

```
        void delete_rec();          void add_record();
```

Also try using overloaded functions at least 2. For example:

```
int search();  //classic search by ID that returns the index if
                 found; -1 otherwise
void search(char gender);

void print_records();
void print_records(int month);
void print_records(char grade);
```

Solution:

Task1:

```
//Lab6- Task1

#include <iostream>
using namespace std;

struct Rectangle
{
    double width;
    double height;
    double area;
}; //end Rectangle

double computeArea(double w, double h) {  return w*h; }

void computeArea(Rectangle &r){  r.area=r.height*r.width; }

Rectangle computeAreaV(Rectangle r) {
    r.area=r.height*r.width;
    return r;
}// end computeAreaV

int main() {
    Rectangle r;
    cout<<" enter the height: ";
    cin>>r.height;
    cout<<" enter the width: ";
    cin>>r.width;

    /*it is not necassary to call all the three
     functions to calculate the area,
     however practice all of them for your benefit.
     */

    //using computeArea function with 2 parameters
    r.area=computeArea(r.height, r.width);

    //using computeArea function with referenced Rectangle parameter
    computeArea(r);
    //using computerAreaV function
    r=computeAreaV(r);

    cout<<r.area<<endl;
    return 0;
}//end of main
```

Task2:

```
#include <iostream>
using namespace std;
struct Fract
{  int num;
    int deno;
};
Fract sum(Fract f1,Fract f2){
    Fract r;
    r.num=(f1.num*f2.deno)+(f2.num*f1.deno);
    r.deno=(f1.deno*f2.deno);
    return r;
}
int main(){
    Fract x;
    Fract y;
    cout<<"enter Fraction 1 (numerator , denominator)\n";
    cin>>x.num>>x.deno;
    cout<<"enter Fraction 2 (numerator , denominator)\n";
    cin>>y.num>>y.deno;
    Fract z=sum(x,y);
    cout<<"result is ="<<z.num<<"/"<<z.deno;
}
```

Task3:

```
/*
      Lab6-Task3 Students Records + Queries (statistical Analysis)
      Author: Ms. Mona Altassan

Menu:
      1  - Add a student
      2  - Delete a student
      3  - Update a student
      4  - View all records
      5  - Find the max score
      6  - Find a student by ID
      7  - Sort records by scores
      8  - Show students by grade
      9  - Show students by gender
      10 - Show students born in month ##

*/

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

//Global variables
const int SIZE=20; //global constant
int count=0;       //actual size of array

//Structures Declaration
struct Date
{
    int day;
    int month;
    int year;
};//end of struct Date

struct Student
{
      string id;
      string name;
      char gender;
      Date dob;
      double score;
      char grade;   //computed by an internal function
} st[SIZE]; //global array of struct Student

//Functions Prototypes
int menu();
char compute_grade(double score);

void add_record();
void delete_record();
void update_record();
void sort_records();
```

```
bool emptyArray();
int find_max();
int search();
void search(char gender);

void show_record(int i);
void print_records();
void print_records(int month);
void print_records(char grade);

int main()
{
      int choice;

  //Show a menu to the user
      do{
            choice=menu();

            switch (choice)
            {
                  case 1: //Add new student
                              if (count<SIZE)
                                 add_record();
                              else
                                 cout<<"\nERROR: The array is full.You need
to delete items first!";
                              break;

                  case 2: //delete a student
                              if (!emptyArray())
                                    delete_record();
                              break;

                  case 3: //update student's score
                              if (!emptyArray())
                                 update_record();
                              break;

                  case 4: //print all records
                              if (!emptyArray())
                                  print_records();
                              break;

                  case 5: //find the max score
                              if (!emptyArray())
                              {
                                    int i=find_max();
                                    show_record(i);
                              }//end if
                              break;

                  case 6: //find a student by ID
                              if (!emptyArray())
                            {
                              int i=search();
                              if (i>=0)
                                  show_record(i);
                              else
                                  cout<<"\nNot found!!\n";
                              }//end if
                              break;

                  case 7: //sort records by scores
                              if (!emptyArray())
                                  sort_records();
                              break;

                  case 8: //Show students by grade
                              if (!emptyArray())
                              {
                                    char grade;
                                  cout<<"\nPlease enter a grade (A/B/C/D/F):
";
                                    cin>>grade;  //add do while and check
later
                                  print_records(grade);
                            }//end if
                              break;

                  case 9: //Show students by gender
                              if (!emptyArray())
                              {
                                    char gender;
                                    //add do-while to ensure input either M
or F
                                    cout<<"\nPlease enter the gender (M/F):
";
                                    cin>>gender;
                                  search(gender);
                            }//end if
                              break;

                  case 10: //Show students born in the same month
                              if (!emptyArray())
                              {
                                    int month;
                                    //add do-while to ensure input in range
1 - 12
                                    cout<<"\nPlease enter a month (1 - 12):
";
                                    cin>>month;
                                  print_records(month);
                            }//end if
                              break;



                  case 11: cout<<"\nThanks for using our program!\n";
                               break;

                  default: cout<<"\nIncorrect menu option.";
            }//end switch
```

```
      }while(choice!=11);

      return 0;
}//end of main


int menu()
{
      int choice;
      cout<<"\nPlease choose a task:"
            <<"\n1  - Add a student"
            <<"\n2  - Delete a student"
            <<"\n3  - Update a student"
            <<"\n4  - View all records"
            <<"\n5  - Find the max score"
            <<"\n6  - Find a student by ID"
            <<"\n7  - Sort records by scores "
            <<"\n8  - Show students by grade"
            <<"\n9  - Show students by gender"
            <<"\n10 - Show students born in month ##"
            <<"\n11 - Exit"
            <<"\n*********************************"
            <<"\n>> ";
      cin>>choice;
      return choice;
}//end of menu

char compute_grade(double score)
{
      if (score>=90.0)
          return 'A';
      else if (score>=80.0)
          return 'B';
      else if (score>=70.0)
          return 'C';
      else if (score>=60.0)
          return 'D';
      else
          return 'F';
}//end of compute_grade

void add_record()
{
      cout<<"Please enter student info: ";
      cout<<"\nID: ";      cin>>st[count].id;
      cout<<"Name: ";      cin>>st[count].name;

      cout<<"Gender (M/F): ";
      cin>>st[count].gender;
      //check_gender(st[count].gender);  //by ref

      cout<<"Date of birth (DD-MM-YYYY): ";
      cout<<"\nDay: ";     cin>>st[count].dob.day;
      cout<<"Month: ";     cin>>st[count].dob.month;
      cout<<"Year: " ;     cin>>st[count].dob.year;
      //check_dob(st[count].dob);  //by ref

      cout<<"Score: ";     cin>>st[count].score;
      //check_score(st[count].score);
      st[count].grade=compute_grade(st[count].score);
      count++;
}//end of add_record

void print_records()
{
      for (int i=0;i<count;i++)
           show_record(i);
}//end of print_records

void show_record(int i)
{
      cout<<"\nStudent "<<(i+1)
          <<":\n\tID    :"<<st[i].id
          <<"\n\tName  : "<<st[i].name
            <<"\n\tGender:"<<st[i].gender
            <<"\n\tScore :"<<st[i].score<<" ("<<st[i].grade<<")"
            <<"\n\tDate of Birth (DD-MM-YYYY): "<<st[i].dob.day<<"-
"<<st[i].dob.month<<"-"<<st[i].dob.year
            <<"\n-------------------------------------------------";
}//end of show_record

void print_records(int month)
{
      for (int i=0;i<count;i++)
            if (st[i].dob.month==month)
                  show_record(i);
}//end of print records by month

void print_records(char grade)
{
      for (int i=0;i<count;i++)
           if (st[i].grade==grade)
               show_record(i);
}//end of print_records by grade

void delete_record()
{
      int index=search();
      if (index<0)
      {
            cout<<"\nNot found!!\n";
            return;  //exit from the function
      }//end if

      //delete if found i.e. index>=0
      for (int j=index; j< count; j++) //shifting up
            st[j]=st[j+1];

      count--;
      cout<<"\nStudent deleted successfully. \n";
}//end of delete_record
```

```
void update_record()
{//update student's score only
      int index=search();
      if (index<0)
      {
            cout<<"\nNot found!!\n";
            return;  //exit from the function
      }//end if

      //update if found i.e. index>=0
      cout<<"\nOld score: "<<st[index].score;
      cout<<"\nNew score: ";
      cin>>st[index].score;
      //check_score(st[index].score);  //later
      st[index].grade=compute_grade(st[index].score);  //update student's
grade
      cout<<"\nStudent score updated successfully. \n";
}//end of update_record

bool emptyArray()
{
      if (count==0)
      {
            cout<<"\nERROR: Array is empty!!\n";
            return true;
      }//end if

      return false;
}//end of empty Array

int find_max()
{
      double max=st[0].score;
      int max_indx=0;

      for (int i=1;i<count;i++)
            if (st[i].score>max)
            {
                  max=st[i].score;
                  max_indx=i;
          }//end if

      return max_indx;
}//end of find_max

int search()
{//classic search by ID
      string id;

      cout<<"\nPlease enter a student ID:";
      cin>>id;

      for (int i=0;i<count;i++)
            if (st[i].id==id)
                  return i;
      return -1;  //i.e. not found
}//end of search by ID

void search(char gender)
{
      for (int i=0;i<count;i++)
           if (st[i].gender==gender)
               show_record(i);
}//end of search by gender

void sort_records()
{//sort by scores descendingly max-to-min
      bool ordered=false;
      Student temp;

      if (count < 2)
            cout<<"\nNothing to sort!!";

    for(int i = 0; i < count-1 && ordered==false ; ++i)
    {
            ordered=true;
            for(int j = 0; j < count-1;++j)
            if(st[j].score < st[j+1].score)
            {
                  ordered=false;
                  temp = st[j];
                  st[j] = st[j+1];
                  st[j+1] = temp;
            }//end if
      }//end for

      cout<<"\nScores in descending order:";
      print_records();
}//end of sort_records
```