// Lab 01A source code provided by professor Tony

// Chapter 7, Programming Challenge 2: Rainfall Statistics

//author: tony chern

//comsc 200

//status: complete

#include<iostream>

#include <iomanip>

using namespace std;

// Constant for the number of months

const int NUM\_MONTHS = 12;

// Function prototypes

double getTotal(double [], int);

double getAverage(double [], int);

double getLargest(double [], int, int &);

double getSmallest(double [], int, int &);

int main()

{

// Array to hold the rainfall data

double rainFall[NUM\_MONTHS];

// Get the rainfall for each month.

for (int month = 0; month < NUM\_MONTHS; month++)

{

// Get this month's rainfall.

cout << "Enter the rainfall (in inches) for month #";

cout << (month + 1) << ": ";

cin >> rainFall[month];

// Validate the value entered.

while (rainFall[month] < 0)

{

cout << "Rainfall must be 0 or more.\n"

<< "Please re-enter: ";

cin >> rainFall[month];

}

}

// Set the numeric output formatting.

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

// Display the total rainfall.

cout << "The total rainfall for the year is ";

cout << getTotal(rainFall, NUM\_MONTHS)

<< " inches." << endl;

// Display the average rainfall.

cout << "The average rainfall for the year is ";

cout << getAverage(rainFall, NUM\_MONTHS)

<< " inches." << endl;

// Now display the largest & smallest amounts.

// The subscript variable will be passed by reference

// the the getLargest and getSmallets functions.

int subScript;

// Display the largest amount of rainfall.

cout << "The largest amount of rainfall was ";

cout << getLargest(rainFall, NUM\_MONTHS, subScript)

<< " inches in month ";

cout << (subScript + 1) << "." << endl;

// Display the smallest amount of rainfall.

cout << "The smallest amount of rainfall was ";

cout << getSmallest(rainFall, NUM\_MONTHS, subScript)

<< " inches in month ";

cout << (subScript + 1) << "." << endl << endl;

return 0;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The getTotal function calculates and returns the \*

// total of the values stored in the array parameter. \*

// The size parameter indicates the number of elements \*

// in the array. \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double getTotal(double values[], int size)

{

double total = 0.0; // Accumulator

// Step through the array and add each value

// to the total variable.

for (int count = 0; count < size; count++)

total += values[count];

return total;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The getAverage function returns the average of the \*

// values in the array parameter. The size parameter \*

// indicates the number of elements in the array. \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double getAverage(double values[], int size)

{

// Simpy get the total of the array values

// and divide by the number of elements.

double average = getTotal(values, size) / size;

return average;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The getLargest function returns the smallest value in \*

// the array parameter, and stores the subscript of the \*

// largest value in the element reference parameter. \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double getLargest(double values[], int size, int &element)

{

double largest; // To hold the largest value.

// Assume element 0 holds the largest value.

largest = values[0];

element = 0;

// Compare the value in largest to all the

// other values in the array.

for (int count = 1; count < size; count++)

{

if (values[count] > largest)

{

largest = values[count];

element = count;

}

}

return largest;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The getSmallest function returns the smallest value in \*

// the array parameter, and stores the subscript of the \*

// smallest value in the element reference parameter. \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double getSmallest(double values[], int size, int &element)

{

double smallest; // To hold the smallest value

// Assume element 0 holds the smallest value.

smallest = values[0];

element = 0;

// Compare the value in largest to all the

// other values in the array.

for (int count = 1; count < size; count++)

{

if (values[count] < smallest)

{

smallest = values[count];

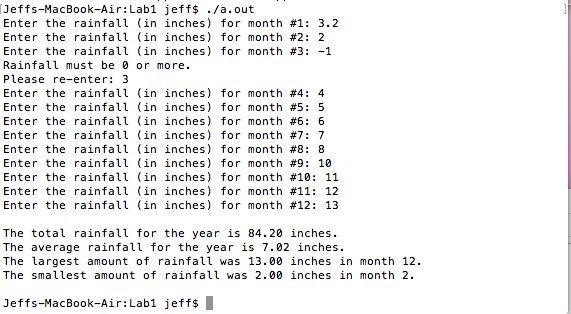
element = count;

}

}

return smallest;

}



//Lab 1 B

//comsc 200

//status: complete

// main.cpp

// lab1xcode

//

// Created by Jeff on 8/17/16.

// Copyright © 2016 Jeff. All rights reserved.

//

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void insertVec(vector<double>&); //decleare the parmeter vector here

void display(vector<double>&);

void Maxrain(vector<double>&);

void Small(vector<double>&);

int const Num\_vec = 12;

int main()

{

// int sum\_of\_elems = 0;

vector<double>myvector;

/\*

vector<int>myvector;

for(int i=0;i<3;i++)

{

cout<<"enter vector: " << endl;

int nums;

cin>>nums;

myvector.push\_back(nums);

}

for(vector<int>::iterator it = myvector.begin(); it != myvector.end(); ++it)

sum\_of\_elems += \*it;

cout << " this is standerd vector version total sum: " << sum\_of\_elems << endl;

\*/

//=====================================================================================

// parmeter verstion vector below

insertVec(myvector);

display(myvector);

Maxrain(myvector);

Small(myvector);

}

//insert vector using paremeter vector

void insertVec(vector<double>& newvector){

double value;

int Num=0;

for(int i=0;i<Num\_vec;i++){

cout<<"enter the rainfall for month # " <<Num+1 << " : ";

cin>>value;

newvector.push\_back(value);

Num++;

}

}

// display the sum of the vector using parameter vector

void display(vector<double>& myvector){

// cout<<"this is parmeter vector, sum of them : ";

// for(vector<int>::iterator it = myVector.begin(); it != myVector.end();++it)

int total = 0 ;

for(auto item:myvector){

total+=item;

//cout<< "total1: "<< total << ' '<<endl;

}

cout<<"total rainfall: "<< total <<' '<<endl;

cout <<"average rainfall: " << total/Num\_vec << ' '<<endl;

}

void Maxrain(vector<double>&myvector){

double max=0;

int month=0;

for(auto item:myvector)

{

if(myvector[max]<myvector[month])

max = month;

month++;

}

cout<<"max rainfall is :"<<myvector[max] << ' ' << "in "<< max+1 << " month"<< endl;

}

void Small(vector<double>&myvector){

int small=0;

int month=0;

for(auto item:myvector){

if(myvector[small]>myvector[month])

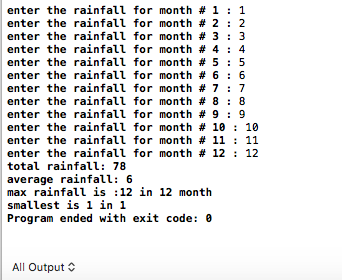
small = month;

month++;

}

cout<<"smallest is " <<myvector[small]<<' '<< "in "<<small+1<<endl;

}



//Lab 1 C

//comsc 200

//status: complete

#include <iostream>

#include "Rectangle.h" // Needed for Rectangle class

using namespace std;

int main()

{

Rectangle box; // Define an instance of the Rectangle class

double rectWidth; // Local variable for width

double rectLength; // Local variable for length

// Get the rectangle's width and length from the user.

cout << "This program will calculate the area of a\n";

cout << "rectangle. What is the width? ";

cin >> rectWidth; //这里的值还在本地 需要转到class

cout << "What is the length? ";

cin >> rectLength; //这里的值还在本地 需要转到class

// Store the width and length of the rectangle

// in the box object.

box.setWidth(rectWidth); //这里把值 送到 class， 但不能反回 name.class(varibale) 只有进去没有回来

box.setLength(rectLength);

// Display the rectangle's data.

cout << "Here is the rectangle's data:\n";

cout << "Width: " << box.getWidth() << endl; //这里可以返回 name.class() 没有进去，只有返回。

cout << "Length: " << box.getLength() << endl;

cout << "Area: " << box.getArea() << endl;

return 0;

}

Header file ---------------------------------

//

// Rectangle.h

// lab2

//

// Created by Jeff on 8/17/16.

// Copyright © 2016 Jeff. All rights reserved.

//

#ifndef Rectangle\_h

#define Rectangle\_h

class Rectangle{

private:

double width; //从CPP 那边访问不到这里 只是方便header内部使用的varibale

double length;

public:

//constructors

Rectangle(){width =1; length=1;}; //default construct , nothing knows here

Rectangle(double w , double l) {width=w; length=l;};

//getters

void setWidth(double w) {width=w;};

void setLength(double l){length =l ;};

//setters

double getWidth()const {return width;};

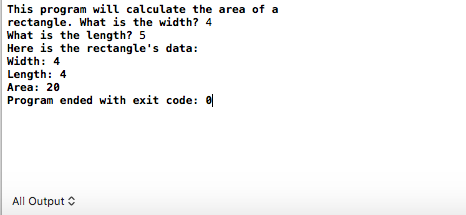
double getLength()const {return width;};

//feature functions(methods)

double getArea() {return width\*length;};

};

#endif /\* Rectangle\_h \*/



//Lab 1 D

//comsc 200

//status: complete

// Chapter 13, Programming Challenge 1: Date Class

#include <iostream>

#include <string>

#include "Date.h"

using namespace std;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function main \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{

// Create a Date object and initialize it

// using the overloaded constructor.

Date today(12, 25, 2012);

// Show the date in form #1.

today.showData1();

// Store a new month, day, and year

// in the object.

today.setMonth(6);

today.setDay(16);

today.setYear(2012);

// Show the date in form #2.

today.showData2();

// Show the date in form #3.

today.showData3();

return 0;

}

Header file --------------------------------------------------------

//

// Date.h

// lab1d

//

// Created by Jeff on 8/17/16.

// Copyright © 2016 Jeff. All rights reserved.

//

#ifndef Date\_h

#define Date\_h

const int NUM\_MONTHS=12;

class Date{

private:

int day;

int month;

int year;

std:: string names[NUM\_MONTHS];

//private helper

void setNames(){

names[0] = "Jan";

names[1] = "Feb";

names[2] = "Mar";

names[3] = "Apr";

names[4] = "May";

names[5] = "Jun";

names[6] = "Jul";

names[7] = "Aug";

names[8] = "Sept";

names[9] = "Oct";

names[10] = "Nov";

names[11] = "Dec";

};

public:

//Date(){day=1;month=1;year=2016;};

Date(int d,int m, int y){

setDay(d);

setMonth(m);

setYear(y);

setNames();

}

//setters

void setDay(int d){day=d;};

void setMonth(int m){month=m;};

void setYear(int y){year=y;};

//getters

int getDay()const{return day;};

int getMonth()const{return month;};

int getYear()const{return year;};

//feature methods

//1/1/2016

void showData1(){

std::cout<<month <<"/" <<day << "/" <<year << std::endl;

}

//january 1,2016

void showData2(){

std::cout<<names[month-1] <<" "<<day<<" , "<< year <<std::endl;

}

//1 january,2016

void showData3(){

std::cout<<day<<" "<<names[month-1]<<" , " <<year <<std::endl;

}

};

#endif /\* Date\_h \*/

