// Specification file for the Rectangle class. This version has a constructor.

/\*#pragma once

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

class Rectangle

{

private:

double width;

double length;

public:

Rectangle(); // Constructor

void setWidth(double);

void setLength(double);

double getWidth() const

{

return width;

}

double getLength() const

{

return length;

}

double getArea() const

{

return width \* length;

}

};

#endif\*/

#pragma once

#ifndef Degree\_H

#define Degree\_H

#include <string>

using namespace std;

class Degree

{

private:

string fname;

string lname;

string subject;

char grade;

public:

Degree(); // Constructor

void setfName(string);

void setlName(string);

void setSubject(string);

void setGrade(char);

string getfName() const

{

return fname;

}

string getlName() const

{

return lname;

}

string getSubject() const

{

return subject;

}

char getGrade() const

{

return grade;

}

};

#endif

// Implementation file for the Rectangle class. This version has a constructor.

/\*#include "Header.h" // Needed for the Rectangle class

#include <iostream> // Needed for cout

#include <cstdlib> // Needed for the exit function

using namespace std;

// The constructor intializes width and length to 0.0.

Rectangle::Rectangle()

{

width = 0.0;

length = 0.0;

}

// setWidth sets the value of the member variable width

void Rectangle::setWidth(double w)

{

if (w >= 0)

width = w;

else

{

cout << "Invalid width\n";

exit(EXIT\_FAILURE);

}

}

// setLength sets the value of the member variable length

void Rectangle::setLength(double len)

{

if (len >= 0)

length = len;

else

{

cout << "Invalid length\n";

exit(EXIT\_FAILURE);

}

}

// This program uses the Rectangle class's constructor.

int main()

{

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

// Display the rectangle's data

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

cout << "Width: " << box.getWidth() << endl;

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

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

return 0;

}

#include "Header.h" // Needed for the Degree class

#include <iostream> // Needed for cout

using namespace std;

// The constructor intializes width and length to 0.0.

Degree::Degree()

{

fname = "";

lname = "";

subject = "";

grade = 'F';

}

// setfName sets the value of the member variable fname

void Degree::setfName(string f)

{

fname = f;

}

// setlName sets the value of the member variable lname

void Degree::setlName(string l)

{

lname = l;

}

// setSubject sets the value of the member variable subject

void Degree::setSubject(string s)

{

subject = s;

}

// setGrade sets the value of the member variable grade

void Degree::setGrade(char g)

{

grade = g;

}

// This program uses the Rectangle class's constructor.

int main()

{

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

string f;

string l;

string s;

char g;

cout << "What is the first name? ";

cin >> f;

cout << "What is the last name? ";

cin >> l;

cout << "What is the subject? ";

cin >> s;

cout << "What is the grade? ";

cin >> g;

// Store the data

box.setfName(f);

box.setlName(l);

box.setSubject(s);

box.setGrade(g);

// Display the degree's data

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

cout << "First Name: " << box.getfName() << endl;

cout << "Last Name: " << box.getlName() << endl;

cout << "Subject: " << box.getSubject() << endl;

cout << "Grade: " << box.getGrade() << endl;

return 0;

}

// This program stores the address of a variable in a pointer.

#include <iostream>

using namespace std;

int main()

{

int x = 25; // int variable

int \*ptr = nullptr; // pointer variable, can point to an int

ptr = &x; // store the address of x in ptr

cout << "The value in x is " << x << endl;

cout << "The address of x is " << ptr << endl;

return 0;

}\*/

// This program demonstrates the use of the indirection operator.

#include <iostream>

using namespace std;

int main()

{

int x = 25; // int variable

int \*ptr = nullptr; // pointer variable, can point to an int

ptr = &x; // store the address of x in ptr

// use both x and ptr to display the value in x

cout << "Here is the value in x, printed twice:\n";

cout << x << endl; // displays the contents of x

cout << \*ptr << endl; // displays the contents of x

// assign 100 to the location pointed to by ptr. this will actually assign

// 100 to x

\*ptr = 100;

// use both x and ptr to display the value in x

cout << "Once again, here is the value in x:\n";

cout << x << endl; // displays the contents of x

cout << x << endl; // displays the contents of x

return 0;

}