**11.3**

#include <iostream>

using namespace std;

// function that returns a new array that doubles the size of the parameter list

int\* doubleCapacity(const int\* list, int size)

{

int newSize = 2 \* size;

int\* newList = new int[newSize];

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

\*(newList + i) = list[i];

delete[] list;

return newList;

}

int main()

{

// prompt the user to enter originial array size

cout << "Enter the size of your array: ";

int s;

cin >> s;

// prompt the user to enter the inputs for the array depending on the size inputted

int\* list = new int[s];

int value;

cout << "Enter the values for the array: ";

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

{

cin >> value;

\*(list + i) = value;

}

// use the doubleCapacity function

// and prompts the user to enter inputs for the remaining array size

list = doubleCapacity(list, s);

int newS = 2 \* s;

cout << "Enter " << newS - s << " values: ";

for (int i = s; i < newS; i++)

{

int integer;

cin >> integer;

\*(list + i) = integer;

}

cout << "The new array is ";

for (int j = 0; j < newS; j++)

{

cout << \*(list + j) << " ";

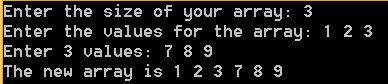
}

cout << endl;

system("pause");

return 0;

}

****

**11.5**

#include <iostream>

using namespace std;

// function to find the smallest integer in the array

int findSmallest(const int\* x, int size)

{

int smallest = \*x; // or x[0];

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

{

if (smallest > \*(x + i))

smallest = \*(x + i);

}

return smallest;

}

int main()

{

// using the function above to get the smallest integer using the given array

int a[] = { 1,2,4,5,10,100,2,-22 };

cout << "The smallest of the array is " << findSmallest(a, 8) << endl;

system("pause");

return 0;

}

****

**11.8**

**UML Diagram**

|  |
| --- |
| **Circle2D** |
| - x : double  - y : double  - radius : double |
| + Circle2D()  + Circle2D(r : double)  + Circle2D(x : double, y : double, r : double)  + getX() : double const  + getY(): double const  + getRadius(): double const  + getArea(): double const  + getPerimeter(): double const  + contains(x : double, y : double) : bool const  + contains(const Circle2D& circle) : bool const  + overlaps(const Circle2D& circle) : bool const |

**Header**

#ifndef CIRCLE2D\_H

#define CIRCLE2D\_H

class Circle2D

{

public:

Circle2D();

Circle2D(double r);

Circle2D(double x, double y, double r);

double getX() const;

double getY() const;

double getRadius() const;

double getArea() const;

double getPerimeter() const;

bool contains(double x, double y) const;

bool contains(const Circle2D& circle) const;

bool overlaps(const Circle2D& circle) const;

private:

double x;

double y;

double radius;

};

#endif

**Implementation**

#include "Circle2D.h"

#include <iostream>

using namespace std;

Circle2D::Circle2D()

{

x = 0;

y = 0;

radius = 1;

}

Circle2D::Circle2D(double r)

{

x = 0;

y = 0;

radius = r;

}

Circle2D::Circle2D(double x, double y, double r)

{

this->x = x; // this is a pointer to the implicit object making the call

this->y = y;

radius = r;

}

double Circle2D::getX() const

{

return x;

}

double Circle2D::getY() const

{

return y;

}

double Circle2D::getRadius() const

{

return radius;

}

double Circle2D::getArea() const

{

return 3.14159 \* pow(radius, 2);

}

double Circle2D::getPerimeter() const

{

return 2 \* 3.14159 \* radius;

}

bool Circle2D::contains(double x, double y) const

{

if (pow((this->x - x), 2) + pow((this->y - y), 2) <= pow(radius, 2))

return true;

else

return false;

}

bool Circle2D::contains(const Circle2D& circle) const

{

if (contains(circle.x, circle.y) && circle.radius <= radius)

return true;

else

return false;

}

bool Circle2D::overlaps(const Circle2D& circle) const

{

if ((pow((x - circle.x), 2) + pow((y - circle.y), 2)) >= pow((radius + circle.radius), 2)

&& (pow((x - circle.x), 2) + pow((y - circle.y), 2)) >= pow((radius - circle.radius), 2))

return true;

else

return false;

}

**Main**

#include <iostream>

#include "Circle2D.h"

using namespace std;

int main()

{

Circle2D c1(2, 2, 5.5);

cout << "The area of c1 is " << c1.getArea() << endl;

cout<< "The perimeter of c1 is " << c1.getPerimeter() << endl;

cout << endl;

if (c1.contains(3, 3))

cout << "Point (3,3) is inside the circle c1(2, 2, 5.5)\n";

else

cout << "Point (3,3) is outside the circle c1(2, 2, 5.5)\n";

Circle2D c2(2, 2, 5.5);

if (c1.contains(c2))

cout << "Circle c2(2, 2, 5.5) is inside the circle c1(2, 2, 5.5)\n";

else

cout << "Circle c2(2, 2, 5.5) is outside the circle c1(2, 2, 5.5)\n";

Circle2D c3(4, 5, 10.5);

if (c1.overlaps(c3))

cout << "Circle c3(4, 5, 10.5) overlaps the circle c1(2, 2, 5.5)\n";

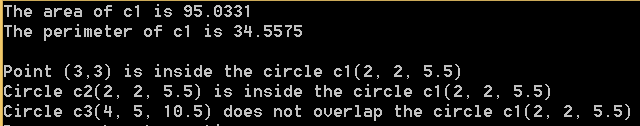
else

cout << "Circle c3(4, 5, 10.5) does not overlap the circle c1(2, 2, 5.5)\n";

system("pause");

return 0;

}

****

**11.13**

**Header**

#ifndef COURSE\_H

#define COURSE\_H

#include <string>

using namespace std;

class Course

{

public:

Course(const string& courseName, int capacity);

~Course();

Course(const Course&);

string getCourseName() const;

void addStudent(const string& name);

void dropStudent(const string& name);

string\* getStudents() const;

int getNumberOfStudents() const;

void clear();

private:

string courseName;

string\* students;

int numberOfStudents;

int capacity;

};

#endif

**Implementation**

#include <iostream>

#include "Course.h"

using namespace std;

Course::Course(const string& courseName, int capacity)

{

numberOfStudents = 0;

this->courseName = courseName;

this->capacity = capacity;

students = new string[capacity];

}

Course::~Course()

{

delete[] students;

}

Course::Course(const Course& course)

{

courseName = course.courseName;

numberOfStudents = course.numberOfStudents;

capacity = course.capacity;

students = new string[capacity];

}

string Course::getCourseName() const

{

return courseName;

}

void Course::addStudent(const string& name)

{

if (numberOfStudents == capacity)

{

capacity = 2 \* capacity;

string\* oldList = students;

students = new string[capacity];

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

\*(students + i) = \*(oldList + i);

delete[] oldList;

}

students[numberOfStudents] = name;

numberOfStudents++;

}

void Course::dropStudent(const string& name)

{

int count = 0;

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

{

if (\*(students + i) == name)

{

for (int j = i + 1; j < numberOfStudents; j++)

\*(students + i) = \*(students + j);

numberOfStudents--;

count++;

}

}

if (count == 0)

{

cout << name << " was not enrolled in the course.\n";

}

}

string\* Course::getStudents() const

{

return students;

}

int Course::getNumberOfStudents() const

{

return numberOfStudents;

}

void Course::clear()

{

delete[] students;

numberOfStudents = 0;

students = new string[capacity];

}

**Main**

#include <iostream>

#include "Course.h"

#include <string>

using namespace std;

int main()

{

Course c1("CSC 252", 2);

c1.addStudent("James");

c1.addStudent("Jenny");

c1.addStudent("Jessica");

c1.dropStudent("Jenny");

cout << "Number of students in " << c1.getCourseName()

<< ": " << c1.getNumberOfStudents() << endl;

string\* students = c1.getStudents();

for (int i = 0; i < c1.getNumberOfStudents(); i++)

cout << students[i] << ", ";

cout << endl;

system("pause");

return 0;

}

****