**Министерство образования и науки Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ”**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**ОТЧЕТ**

**по Лабораторной работе № 9**

**«Решение задач с использованием НАСЛЕДОВАНИЯ»**

Специальность 09.02.07«Информационные системы и программирование»

Дисциплина «Прикладное программирование»

|  |  |
| --- | --- |
| Преподаватель:  Сорокин Д.Ю.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2019г.  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Выполнил:  студент группы Y2335  Котлицкий С.А. |

Санкт-Петербург

2018/2019

**Цель работы:** продемонстрировать работу с механизмом наследования в объектно-ориентированном программировании

**Практическое задание:** Реализовать дочерний класс из класса восьмой лабораторной работы. Объект описать минимально возможным количеством полей, остальные характеристики вычислять динамически с помощью методов. Класс должен содержать минимум 2 конструктора: конструктор по умолчанию и параметризованный. Перегрузить операторы записи и чтения потока, для обеспечения потокового ввода-вывода. Перегрузить минимум 2 стандартных оператора(унарный и бинарный).

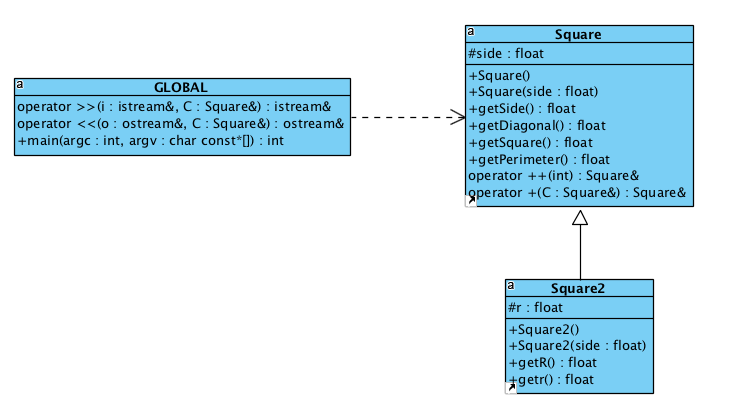
Написать программу, выполняющую:

1. создание 2-х экземпляров класса, с динамическим выделением памяти, первый с параметрами по умолчанию, второй с параметрами заданными пользователем;
2. вывод всех характеристик заданного экземпляра класса на экран;
3. демонстрацию работы перегруженных операторов;
4. освобождение выделенной памяти.

**Индивидуальное задание:**

1. Квадрат.

Диаграмма классов программы представлена ниже (Рисунок 1):



*Рисунок 1 Диаграмма классов программы*

Код программы:

#include <iostream>

#include <math.h>

using namespace std;

class Square {

protected:

float side;

public:

Square(){

side = 1;

}

Square(float side){

this -> side = side;

}

float getDiagonal(){

return sqrt(2 \* side \* side);

}

float getSquare(){

return side \* side;

}

float getPerimeter(){

return side \* 4;

}

Square&operator++(int){

side++;

return \*this;

}

Square&operator+(Square&C) {

Square T(0);

T.side = this -> side + C.side;

return T;

}

friend istream&operator>>(istream&i, Square&C);

friend ostream&operator<<(ostream&o, Square&C);

};

istream&operator>>(istream&i, Square&C) {

cout << "Введите длину стороны" << endl;

i >> C.side;

return i;

}

ostream&operator<<(ostream&o, Square&C) {

o << "side = " << C.side << endl;

return o;

}

class Square2: public Square{

protected:

float r;

public:

Square2(){

side = 1;

r = side/2;

}

Square2(float side) {

this -> side = side;

this -> r = side/2;

}

float getR(){

return r\*sqrt(2);

}

float getr(){

return r;

}

};

int main(int argc, const char \* argv[]) {

Square2 \*A = new Square2(5), \*B = new Square2();

cin>>\*B;

Square T = \*A+\*B;

cout<<\*A;

cout<<\*B;

cout << "Perimeter A " << A -> getPerimeter() << "\nPerimeter B " << B -> getPerimeter() << endl;

cout << "Diagonal A " << A -> getDiagonal() << "\nDiagonal B " << B -> getDiagonal() << endl;

cout << "Square A " << A -> getSquare() << "\nSquare B " << B -> getSquare() << endl;

cout << "r A " << A -> getr() << "\nr B " << B -> getr() << endl;

cout << "R A " << A -> getR() << "\nR B " << B -> getR() << endl;

cout << "T ";

cout<<T;

cout << "A ";

cout<<\*A;

delete A;

delete B;

return 0;

}

Протокол программы:

Введите длину стороны

4

side = 5

side = 4

Perimeter A 20

Perimeter B 16

Diagonal A 7.07107

Diagonal B 5.65685

Square A 25

Square B 16

r A 2.5

r B 0.5

R A 3.53553

R B 0.707107

T side = 9

A side = 5

Program ended with exit code: 0