**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 3**

Тема: Механизмы наследования в С++

Студент: Цыкин Иван Алексеевич

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения. Все классы должны поддерживать набор общих методов:

* Вычисление геометрического центра фигуры;
* Вывод в стандартный поток вывода std::cout координат вершин фигуры;
* Вычисление площади фигуры.

Создать программу, которая позволяет:

* Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.
* Сохранять созданные фигуры в динамический массив std::vector<Figure\*>.
* Вызывать для всего массива общие функции (1-3 см. выше).Т.е. распечатывать для каждой фигуры в массиве геометрический центр, координаты вершин и площадь.
* Необходимо уметь вычислять общую площадь фигур в массиве.
* Удалять из массива фигуру по индексу.

1. Описание программы

Программа состоит из 3 файлов: Figure.hpp – файл, где описан класс Figure со своим набором методов и 3 класса (Rectangle, Rhombus, Trapeze), которые в последствии наследуются от данного класса; Vector2D.hpp – класс, который описывает структуру вектора, где описаны методы работы с векторами (длина вектора, перпендикулярность векторов, параллельность); main.cpp – файл, который позволяет работать с классом фигур, в нем используется стандартная библиотека vector для удобного хранения фигур.

Меню для работы с фигурами:

1 - Add Figure - добавление фигуры

2 - Delete Figure - удалить фигуру по индексу

3 - Sum of areas - сумма площадей

4 - Working with a figure - посмотреть значения выбранной фигуры

5 - All figures - список всех фигур

0 - Stop - закончить работу

1. Набор тестов

* Работа с фигурами

PS C:\Users\itsyk\.vscode> & 'c:\Users\itsyk\.vscode\extensions\ms-vscode.cpptools-1.1.0-insiders2\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-sjoacwsn.cxa' '--stdout=Microsoft-MIEngine-Out-1g20vwvd.gis' '--stderr=Microsoft-MIEngine-Error-nnkx5b2s.4bm' '--pid=Microsoft-MIEngine-Pid-ogg5ei5i.kco' '--dbgExe=C:\MinGW\mingw32\bin\gdb.exe' '--interpreter=mi'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| |

| Menu |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| |

|1 - Add Figure |

|2 - Delete Figure |

|3 - Sum of areas |

|4 - Working with a figure |

|5 - All figures |

|0 - Stop |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

1

Enter the values of the x and y coordinate points

A: 1 1

B: 1 4

C: 4 4

D: 4 1

Figure was created: RECTANGLE

4

Enter figure number: 1

RECTANGLE

Point A: (1,1)

Point B: (1,4)

Point C: (4,4)

Point D: (4,1)

The center point: (2.5,2.5)

Area: 9

1

Enter the values of the x and y coordinate points

A: 3 3

B: 0 7

C: 3 11

D: 6 7

Figure was created: RHOMBUS

4

Enter figure number: 2

RHOMBUS

Point A: (3,3)

Point B: (0,7)

Point C: (3,11)

Point D: (6,7)

The center point: (3,7)

Area: 24

1

Enter the values of the x and y coordinate points

A: 3 1

B: 1 3

C: 1 5

D: 5 1

Figure was created: TRAPEZE

4

Enter figure number: 3

TRAPEZE

Point A: (3,1)

Point B: (1,3)

Point C: (1,5)

Point D: (5,1)

The center point: (2.5,2.5)

Area: 6

1

Enter the values of the x and y coordinate points

A: 4 7

B: 3 4

C: 2 2

D: 9 0

Error

* Работа с хранением фигур

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| |

| Menu |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| |

|1 - Add Figure |

|2 - Delete Figure |

|3 - Sum of areas |

|4 - Working with a figure |

|5 - All figures |

|0 - Stop |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

1

Enter the values of the x and y coordinate points

A: 1 1

B: 1 4

C: 4 4

D: 4 1

Figure was created: RECTANGLE

1

Enter the values of the x and y coordinate points

A: 3 3

B: 0 7

C: 3 11

D: 6 7

Figure was created: RHOMBUS

1

Enter the values of the x and y coordinate points

A: 1 3

B: 1 5

C: 5 1

D: 3 1

Figure was created: TRAPEZE

5

There are 3 figures

1.RECTANGLE

2.RHOMBUS

3.TRAPEZE

4

Enter figure number: 1

RECTANGLE

Point A: (1,1)

Point B: (1,4)

Point C: (4,4)

Point D: (4,1)

The center point: (2.5,2.5)

Area: 9

4

Enter figure number: 2

RHOMBUS

Point A: (3,3)

Point B: (0,7)

Point C: (3,11)

Point D: (6,7)

The center point: (3,7)

Area: 24

4

Enter figure number: 3

TRAPEZE

Point A: (1,3)

Point B: (1,5)

Point C: (5,1)

Point D: (3,1)

The center point: (2.5,2.5)

Area: 6

3

S = 39

2

Enter figure number: 2

Figure was deleted!

5

There are 2 figures

1.RECTANGLE

2.TRAPEZE

3

S = 15

1. Листинг программы

* Vector2D.hpp

#ifndef VECTOR2D

#define VECTOR2D

#include <iostream>

#include <cmath>

using namespace std;

class Vector2D{

    public:

        double x, y;

        double l;

        void Set(double x1, double y1, double x2, double y2){

            x = x2 - x1;

            y = y2 - y1;

            l = sqrt(x\*x + y\*y);

        }

        bool in\_parallel(Vector2D v){

            if(x \* v.y == y \* v.x){

                return true;

            } else{

                return false;

            }

        }

        bool perpendicularity(Vector2D v){

            if(x \* v.x + y \* v.y){

                return false;

            } else{

                return true;

            }

        }

};

#endif //VECTOR2D

* Figure.hpp

#ifndef FIGURE\_LR

#define FIGURE\_LR

#include "Vector2D.hpp"

class Figure{

    public:

        double Ax, Ay;

        double Bx, By;

        double Cx, Cy;

        double Dx, Dy;

        double cenX, cenY;

        double S;

        virtual void name(){

            cout << "Non";

        }

        void print\_points(){

            cout << "Point A: (" << Ax << "," << Ay << ")" << endl;

            cout << "Point B: (" << Bx << "," << By << ")" << endl;

            cout << "Point C: (" << Cx << "," << Cy << ")" << endl;

            cout << "Point D: (" << Dx << "," << Dy << ")" << endl;

        }

        void print\_S(){

            cout << "Area: " << S << endl;

        }

        void print\_cen(){

            cout << "The center point: (" << cenX << "," << cenY << ")" << endl;

        }

        virtual void calS(){

            S = 0;

        }

        virtual void calC(){

            cenX = cenY = 0;

        }

};

class Rectangle: public Figure{

    public:

        Rectangle(double ax, double ay, double bx, double by, double cx, double cy, double dx, double dy){

            Ax = ax;

            Ay = ay;

            Bx = bx;

            By = by;

            Cx = cx;

            Cy = cy;

            Dx = dx;

            Dy = dy;

            calC();

            calS();

            cout << "Figure was created: ";

            cout << "RECTANGLE" << endl;

        }

        void name(){

            cout << "RECTANGLE" << endl;

        }

        void calS() override{

            Vector2D v1, v2;

            v1.Set(Ax, Ay, Bx, By);

            v2.Set(Bx, By, Cx, Cy);

            S = fabs(v1.l \* v2.l);

        }

        void calC(){

            cenX = (Ax+Cx)/2;

            cenY = (Ay+Cy)/2;

        }

};

class Rhombus: public Figure{

    public:

        Rhombus(double ax, double ay, double bx, double by, double cx, double cy, double dx, double dy){

            Ax = ax;

            Ay = ay;

            Bx = bx;

            By = by;

            Cx = cx;

            Cy = cy;

            Dx = dx;

            Dy = dy;

            calC();

            calS();

            cout << "Figure was created: ";

            cout << "RHOMBUS" << endl;

        }

        void name(){

            cout << "RHOMBUS" << endl;

        }

        void calS() override{

            Vector2D v1, v2;

            v1.Set(Ax, Ay, Cx, Cy);

            v2.Set(Bx, By, Dx, Dy);

            S = fabs(v1.l \* v2.l \* 0.5);

        }

        void calC(){

            cenX = (Ax+Cx)/2;

            cenY = (Ay+Cy)/2;

        }

};

class Trapeze: public Figure{

    public:

        Trapeze(double ax, double ay, double bx, double by, double cx, double cy, double dx, double dy){

            Ax = ax;

            Ay = ay;

            Bx = bx;

            By = by;

            Cx = cx;

            Cy = cy;

            Dx = dx;

            Dy = dy;

            calC();

            calS();

            cout << "Figure was created: ";

            cout << "TRAPEZE" << endl;

        }

        void name(){

            cout << "TRAPEZE" << endl;

        }

        void calS() override{

            Vector2D v1, v2, v3, v4;

            v1.Set(Ax, Ay, Bx, By);

            v2.Set(Cx, Cy, Dx, Dy);

            v3.Set(Ax, Ay, Dx, Dy);

            v4.Set(Ax, Ay, Cx, Cy);

            if(!v1.in\_parallel(v2)){

                v1.Set(Bx, By, Cx, Cy);

                v2.Set(Dx, Dy, Ax, Ay);

                v3.Set(Ax, Ay, Bx, By);

                v4.Set(Bx, By, Dx, Dy);

            }

            double polper = (v2.l + v3.l + v4.l)/2;

            double Str = sqrt(polper\*(polper-v2.l)\*(polper-v3.l)\*(polper-v4.l));

            double h = 2\*Str/v2.l;

            S = (0.5\*(v1.l + v2.l)\* h);

        }

        void calC(){

            double Jx, Jy, Gx, Gy;

            Vector2D v1, v2;

            v1.Set(Ax, Ay, Bx, By);

            v2.Set(Cx, Cy, Dx, Dy);

            if(!v1.in\_parallel(v2)){

                Jx = (Ax+Dx)/2;

                Jy = (Ay+Dy)/2;

                Gx = (Bx+Cx)/2;

                Gy = (By+Cy)/2;

                cenX = (Jx+Gx)/2;

                cenY = (Jy+Gy)/2;

            } else{

                Jx = (Ax+Bx)/2;

                Jy = (Ay+By)/2;

                Gx = (Dx+Cx)/2;

                Gy = (Dy+Cy)/2;

                cenX = (Jx+Gx)/2;

                cenY = (Jy+Gy)/2;

            }

        }

};

int sort(double a, double b, double c, double d, double e, double f, double g, double h){

    Vector2D v1, v2, v3, v4, v5, v6;

    v1.Set(a, b, c, d);

    v2.Set(c, d, e, f);

    v3.Set(e, f, g, h);

    v4.Set(g, h, a, b);

    v5.Set(a, b, e, f);

    v6.Set(c, d, g, h);

    if(v1.perpendicularity(v2) && v1.perpendicularity(v4) && v3.perpendicularity(v2) && v3.perpendicularity(v4)){

        return 1;

    } else if(v1.in\_parallel(v3) && v2.in\_parallel(v4) && v5.perpendicularity(v6)){

        return 2;

    } else if((v1.in\_parallel(v3) && !v2.in\_parallel(v4)) || (!v1.in\_parallel(v3) && v2.in\_parallel(v4))){

        return 3;

    } else{

        return 0;

    }

}

#endif //FIGURE\_LR

* main.cpp

#include "figure.hpp"

#include <vector>

void menu(){//меню для работы

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

    cout << "|                            |" << endl;

    cout << "|            Menu            |" << endl;

    cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

    cout << "|                            |" << endl;

    cout << "|1  - Add Figure             |" << endl;

    cout << "|2  - Delete Figure          |" << endl;

    cout << "|3  - Sum of areas           |" << endl;

    cout << "|4  - Working with a figure  |" << endl;

    cout << "|5  - All figures            |" << endl;

    cout << "|0  - Stop                   |" << endl;

    cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

}

int main(){

    int option1, option2, i;

    vector <Figure\*> vec;

    double a, b, c, d, e, f, g, h;

    menu();

    for(;;){

        cin >> option1;

        switch(option1){

            case 1:{

                cout << "Enter the values of the x and y coordinate points" << endl;

                cout << "A: ";

                cin >> a >> b;

                cout << "B: ";

                cin >> c >> d;

                cout << "C: ";

                cin >> e >> f;

                cout << "D: ";

                cin >> g >> h;

                if(sort(a, b, c, d, e, f, g, h) == 1){

                    vec.push\_back(new Rectangle(a, b, c, d, e, f, g, h));

                }else if(sort(a, b, c, d, e, f, g, h) == 2){

                    vec.push\_back(new Rhombus(a, b, c, d, e, f, g, h));

                }else if(sort(a, b, c, d, e, f, g, h) == 3){

                    vec.push\_back(new Trapeze(a, b, c, d, e, f, g, h));

                } else{

                    cout << "Error" << endl;

                    break;

                }

                break;

            }

            case 2:{

                cout << "Enter figure number: ";

                cin >> i;

                auto iter = vec.cbegin();

                if(i > vec.size() && i>=0){

                    cout << "There are " << vec.size() << " figures" << endl;

                    cout << "Try again!" << endl;

                    break;

                }

                vec.erase(iter + i - 1);

                cout << "Figure was deleted!" << endl;

                break;

            }

            case 3:{

                if(vec.size() == 0){

                    cout << "S = 0" << endl;

                    break;

                } else{

                    double S = 0;

                    for(int i=0; i < vec.size(); ++i){

                        S+=vec[i]->S;

                    }

                    cout << "S = " << S << endl;

                }

                break;

            }

            case 4:{

                cout << "Enter figure number: ";

                cin >> i;

                if(i > vec.size() && i >= 0){

                    cout << "There are " << vec.size() << " figures" << endl;

                    cout << "Try again!" << endl;

                    break;

                }

                vec[i-1]->name();

                vec[i-1]->print\_points();

                vec[i-1]->print\_cen();

                vec[i-1]->print\_S();

                break;

            }

            case 5:{

                if(!vec.size()){

                    cout << "There are " << vec.size() << " figures" << endl;

                    break;

                }

                cout << "There are " << vec.size() << " figures" << endl;

                for(int n = 0; n < vec.size(); ++n){

                    cout << n+1 << ".";

                    vec[n]->name();

                    cout << endl;

                }

                break;

            }

            default:

                cout << "Try again!" << endl;

                break;

            case 0:

                vec.clear();

                return 0;

        }

    }

    return 0;

}

1. Вывод

Во время лабораторной работы мной была написан программа, описывающая класс фигур. Убедился, наследование классов — очень мощная возможность в объектно ориентированном программировании. Оно позволяет создавать производные классы (классы наследники), взяв за основу все методы и элементы базового класса (класса родителя). Таким образом экономится масса времени на написание и отладку кода новой программы. Вспомнил как динамически выделяется память и вспомнил о удобном способе хранения данных, с использованием структуры вектор.

Ссылки

1. <https://github.com/youngtommypickles/oop_exercise_003> - github
2. <http://cppstudio.com/post/10103> - наследование классов
3. <https://ravesli.com/urok-85-dinamicheskoe-vydelenie-pamyati-operatory-new-i-delete/> - выделение памяти