**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: Айрапетова Евгения Ашотовна

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

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;

Вариант 34:  
Квадрат

Прямоугольник

Трапеция

1. Описание программы

На вход от пользователя подается центр и вершина (вершина и угол между диагоналями для прямоугольника и две вершины и угол между диагоналями для трапеции). В зависимости от введенной команды пользователь может ввести или удалить фигуры из массива, получить площадь фигур, координаты вершин и центра.

1. Набор тестов

Тест 1:  
Квадрат:  
O: 0 0  
A: 2 2  
  
Прямоугольник:  
O: 0 0  
A: 2 2  
AOB: 90

Тест 2:  
Квадрат:  
O: 0 0  
A: 1 4.5  
  
Трапеция:  
O: 2 2  
A: 1 1  
C: 3 3  
AOB: 90

1. Результаты выполнения тестов

Тест 1:  
Квадрат:  
O: 0 0  
A: 2 2  
  
Прямоугольник:  
O: 0 0  
A: 2 2  
AOB: 90  
  
Для фигуры №1:  
O = [0, 0];  
A = [2, 2];  
B = [-2, 2];  
C = [-2, -2];  
D = [2, -2];  
Для фигуры №2:  
O = [0, 0];  
A = [2, 2];  
B = [-2, 2];  
C = [-2, -2];  
D = [2, -2];

Для фигуры №1: S = 16  
Для фигуры №2: S = 16  
  
Удаление всех элементов массива прошло успешно

Тест 2:  
Квадрат:  
O: 0 0  
A: 1 4.5  
  
Фигур в массиве: 1  
  
Трапеция:  
O: 2 2  
A: 1 1  
C: 3 3  
AOB: 90  
  
Фигур в массиве: 2  
  
Удаление фигуры номер: 1  
Удаление прошло успешно  
  
Для фигуры №1:  
O = [2, 2];  
A = [1, 1];  
B = [3, 1];  
C = [3, 3];  
D = [1, 3];

1. Листинг программы

/\*

Айрапетова Евгения M8o-206Б-19

34.

Квадрат

Прямоугольник

Трапеция

\*/

#include<iostream>

#include<vector>

#define PI 3.14159265

using namespace std;

struct Point {

double x = 0;

double y = 0;

};

Point operator+ (const Point& a, const Point& b) {

return { a.x + b.x, a.y + b.y };

}

Point operator- (const Point& a, const Point& b) {

return { a.x - b.x, a.y - b.y };

}

istream& operator>> (istream& input, Point& a) {

input >> a.x >> a.y;

return input;

}

ostream& operator<< (ostream& output, const Point& a) {

output << "[" << a.x << ", " << a.y << ']';

return output;

}

double sin\_(double deg) {

double rad = deg \* PI / 180.0;

return round(sin(rad) \* 1000) / 1000;

}

double cos\_(double deg) {

double rad = deg \* PI / 180.0;

return round(cos(rad) \* 1000) / 1000;

}

Point turn(Point A, Point O, double deg) {

Point a = { A.x - O.x, A.y - O.y };

double x = (a.x \* cos\_(deg)) - (a.y \* sin\_(deg));

double y = (a.y \* cos\_(deg)) + (a.x \* sin\_(deg));

return{ x + O.x, y + O.y };

}

class Figure {

protected:

Point A\_, B\_, C\_, D\_, O\_;

public:

Figure() {}

virtual void set(Point a, Point b) {}

virtual void set(Point a, Point b, double fi) {}

virtual void set(Point a, Point b, Point c, double fi) {}

virtual double S() { return 0; }

virtual Point A() { return { 0,0 }; }

virtual Point B() { return { 0,0 }; }

virtual Point C() { return { 0,0 }; }

virtual Point D() { return { 0,0 }; }

virtual Point O() { return { 0,0 }; }

/\*

virtual Point A() { return A\_; }

virtual Point B() { return B\_; }

virtual Point C() { return C\_; }

virtual Point D() { return D\_; }

\*/

};

class Square : public Figure {

public:

/\* Square() { // По умолчанию - квадрат с центром в точке 0 и стороной 2

O\_ = { 0,0 };

A\_ = { -1,-1 };

B\_ = { -1,1 };

C\_ = { 1,1 };

D\_ = { 1,-1 };

}

Square(Point o, Point a) {

O\_ = o;

A\_ = a;

C\_ = { A\_.x + 2 \* (O\_.x - A\_.x), A\_.y + 2 \* (O\_.y - A\_.y) };

B\_ = { O\_.x - A\_.y + O\_.y, O\_.y + A\_.x - O\_.x };

D\_ = { B\_.x + 2 \* (O\_.x - B\_.x), B\_.y + 2 \* (O\_.y - B\_.y) };

}

\*/

void set(Point o, Point a) {

O\_ = o;

A\_ = a;

C\_ = { A\_.x + 2 \* (O\_.x - A\_.x), A\_.y + 2 \* (O\_.y - A\_.y) };

B\_ = { O\_.x - A\_.y + O\_.y, O\_.y + A\_.x - O\_.x };

D\_ = { B\_.x + 2 \* (O\_.x - B\_.x), B\_.y + 2 \* (O\_.y - B\_.y) };

}

double S() { return pow(sqrt(pow(A\_.x - B\_.x, 2) + pow(A\_.y - B\_.y, 2)), 2); }

Point A() { return A\_; }

Point B() { return B\_; }

Point C() { return C\_; }

Point D() { return D\_; }

Point O() { return O\_; }

};

class Rectangle : public Figure {

public:

/\* Rectangle(Point a, Point o, double fi) { // fi - угол между диагоналями

A\_ = a;

O\_ = o;

B\_ = turn(A\_, O\_, fi);

C\_ = turn(B\_, O\_, 180.0 - fi);

D\_ = turn(C\_, O\_, fi);

}

\*/

void set(Point o, Point a, double fi) { // fi - угол между диагоналями

A\_ = a;

O\_ = o;

B\_ = turn(A\_, O\_, fi);

C\_ = turn(B\_, O\_, 180.0 - fi);

D\_ = turn(C\_, O\_, fi);

}

double S() { return (sqrt(pow(A\_.x - B\_.x, 2) + pow(A\_.y - B\_.y, 2)) \* sqrt(pow(C\_.x - B\_.x, 2) + pow(C\_.y - B\_.y, 2))); } //AB\*CB

Point A() { return A\_; }

Point B() { return B\_; }

Point C() { return C\_; }

Point D() { return D\_; }

Point O() { return O\_; }

};

class Trapeze : public Figure {

double F; // Угол между диагоналями

public:

/\*Trapeze(Point a, Point c, Point o, double fi) { // O - точка пересечения диагоналей, d - диагональ, fi - угол между диагоналями (AOB)

O\_ = o;

A\_ = a;

B\_ = turn(A\_, O\_, fi); // AB^2 = x^2 + y^2

C\_ = c;

D\_ = turn(C\_, O\_, fi);

F = fi;

}\*/

void set(Point o, Point a, Point c, double fi) { // O - точка пересечения диагоналей, d - диагональ, fi - угол между диагоналями (AOB)

O\_ = o;

A\_ = a;

B\_ = turn(A\_, O\_, fi); // AB^2 = x^2 + y^2

C\_ = c;

D\_ = turn(C\_, O\_, fi);

F = fi;

}

// AC = sqrt(pow(A\_.x + C\_.x, 2) + pow(A\_.y + B\_.y, 2)) - диагональ

double S() { return (pow(A\_.x + C\_.x, 2) + pow(A\_.y + B\_.y, 2) / 2) \* sin\_(F); }

Point A() { return A\_; }

Point B() { return B\_; }

Point C() { return C\_; }

Point D() { return D\_; }

Point O() { return O\_; }

};

void commands() {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

<< "|Команды: |\n"

<< "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n"

<< "|1. Добавить фигуру в массив |\n"

<< "|2. Удалить n-ю фигуру из массива |\n"

<< "|3. Очистить массив |\n"

<< "|4. Вывести координаты вершин и геометрического центра |\n"

<< "|5. Вывести площадь |\n"

<< "|6. Узнать количество фигур в массиве |\n"

<< "|7. Вычислить общую площадь фигур в массиве |\n"

<< "|8. Завершить работу программы |\n"

<< "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n"

<< endl;

}

int inputFigure() {

unsigned int var;

cout << "Список фигур:\n1. Квадрат\n2. Прямоугольник\n3. Трапеция\n"

<< "Введите номер фигуры: ";

if (cin >> var) { return var; }

return -1;

}

int main() {

setlocale(LC\_ALL, "rus");

vector<Figure\*> f;

int i = 0; // Количество фигур в массиве

unsigned int input = 0;

commands();

while (input != 8) {

cout << "Введите команду: ";

if (!(cin >> input)) {

cout << "Команда введена некорректно.\n";

commands();

break;

}

double allS = 0;

int numb;

Point O, A, C;

double AOB;

switch (input) {

case 1:

numb = inputFigure();

switch (numb) {

case 1:

f.push\_back(new Square());

// Point O, A;

cout << "Введите координаты центра (точки пересечения диагоналей) O и одной из вершин квадрата A:\n"

<< "O: ";

cin >> O;

cout << "A: ";

cin >> A;

f[i]->set(O, A);

++i;

break;

case 2:

f.push\_back(new Rectangle());

// Point O, A; double AOB;

cout << "Введите координаты центра (точки пересечения диагоналей) O, одной из вершин\nпрямоугольника A и угол между диагоналями AOB:\n"

<< "O: ";

cin >> O;

cout << "A: ";

cin >> A;

cout << "AOB: ";

cin >> AOB;

f[i]->set(O, A, AOB);

++i;

break;

case 3:

f.push\_back(new Trapeze());

// Point O, A, C; double AOB;

cout << "Введите координаты центра (точки пересечения диагоналей) O, две вершины, лежащие на одной\nдиагонали A и C, а также угол между диагоналями AOB:\n"

<< "O: ";

cin >> O;

cout << "A: ";

cin >> A;

cout << "C: ";

cin >> C;

cout << "AOB: ";

cin >> AOB;

f[i]->set(O, A, C, AOB);

++i;

break;

default:

cout << "Ошибка: Неверный номер фигуры\n";

break;

}

break;

case 2:

int n;

if (i == 0) {

cout << "Нечего удалять. В массиве нет ни одной фигуры\n";

break;

}

cout << "Введите номер фигуры, которую вы хотите удалить: ";

cin >> n;

--n;

if (n >= i) {

cout << "Невозможно удалить фигуру №" << n << ". Всего в массиве фигур: " << i << '\n';

break;

}

for (n; n < i - 1; ++n) {

f[n] = f[n + 1];

}

f.pop\_back();

--i;

cout << "Удаление прошло успешно\n";

break;

case 3:

if (i == 0) {

cout << "Нечего удалять. В массиве нет ни одной фигуры\n";

break;

}

for (int n = 0; n < i; ++n) {

f.pop\_back();

}

cout << "Удаление всех элементов массива прошло успешно\n";

i = 0;

break;

case 4:

if (i == 0) {

cout << "В массиве нет ни одной фигуры\n";

break;

}

for (int var = 0; var < i; ++var) {

cout << "Для " << "фигуры №" << var + 1 << ":\n"

<< "O = " << f[var]->O() << ";\n"

<< "A = " << f[var]->A() << ";\n"

<< "B = " << f[var]->B() << ";\n"

<< "C = " << f[var]->C() << ";\n"

<< "D = " << f[var]->D() << ";\n";

}

break;

case 5:

if (i == 0) {

cout << "В массиве нет ни одной фигуры\n";

break;

}

for (int var = 0; var < i; ++var) {

cout << "Для фигуры №" << var + 1 << ": S = " << f[var]->S() << '\n';

}

break;

case 6:

cout << "Фигур в массиве: " << i << '\n';

break;

case 7:

for (int n = 0; n < i; ++n) {

allS += f[n]->S();

}

cout << "Общая площадь всех фигур в массиве: " << allS << '\n';

case 8: break;

default:

cout << "Команда введена некорректно.\n";

commands();

break;

}

cout << '\n';

}

return 0;

}

1. Вывод:

В процессе выполнения лабораторной работы мною был изучен vector, а также такой инструмент объектно-ориентированного программирования, как наследование. Также я освежила в памяти свои знания об указателях.

Список литературы:

1. Наследование классов в C++ [Электронный ресурс]. URL: <http://cppstudio.com/post/10103/> (дата обращения: 10.10.2020)
2. Vector C++ [Электронный ресурс]. URL: <https://codelessons.ru/cplusplus/vektory-v-c-dlya-nachinayushhix.html> (дата обращения: 13.10.2020)