**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 3**

Тема: Наследование, полиморфизм.

Студент: Курносов Максим Юрьевич

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

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:

1. Вычисление геометрического центра фигуры;

2. Вывод в стандартный поток вывода std::cout координат вершин фигуры;

3. 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.

Вариант задания №8:

|  |  |  |  |
| --- | --- | --- | --- |
| 8. | 8-угольник | Треугольник | Квадрат |

1. **Описание программы**

Программа в файле main.cpp с помощью методов класса работает с введёнными через std::cin данными, на вход подаётся несколько значений:

В начале номер команды из выведенной на стандартный вывод справки, затем в зависимости от запрошенной команды индекс фигуры в хранилище(векторе), первые две координаты фигуры или ничего не подаётся после выполнения команды программа вновь ждёт номера команды.

В результате выполнения команды появляется текст соответствующий запрошенной команде (вывод справки, показ введённых значений в вектор, отбивка об удалении элемента и т.д.).

Завершение программы обозначено отдельной командой.

Хранилище – std::vector<Figure \*>;

класс Figure описан в main.cpp ,как и его наследуемые классы.

Классы-наследники в основном пользуются методами родителя.

1. **Набор тестов**

Дан только для того, что бы показать как вводить значения для создания фигур, все вводы в программу осуществляются через std::in.

(Первые 2 значения – координаты первой вершины ,2ые – координаты второй, затем кол-во вершин фигуры).

For file test\_01.txt:

7 -2 11 -2

4

For file test\_02.txt:

1 8 9 -3 3

For file test\_03.txt:

55 66 7 8 8

1. **Результаты выполнения тестов**

**(Выполнялось с введением значений вручную)**

Для выполнения операции введите соответствующую цифру:

1 - Добавить фигуру в хранилище

2 - Удалить последнюю добавленную фигуру из хранилища

3 - Вывести Геом.Центр. для всех фигур

4 - Вывести Координаты всех вершин для всех фигур

5 - Вывести площади всех фигур

6 - Вычислить общую площадь всех фигур

7 - Удалить фигуру по индексу

8 - Закрыть программу

9 - Показать вектор

0 - Вывести справку

1

Введите слева-направо координаты двух нижних вершин вашей фигуры и кол-во вершин(на выбор: 3, 4, 8).

0

1

2

3

Введите кол-во вершин.

3

Эдемент вставлен.

1

Введите слева-направо координаты двух нижних вершин вашей фигуры и кол-во вершин(на выбор: 3, 4, 8).

5

6

7

8

Введите кол-во вершин.

8

Эдемент вставлен.

1

Введите слева-направо координаты двух нижних вершин вашей фигуры и кол-во вершин(на выбор: 3, 4, 8).

99

66

33333

1

Введите кол-во вершин.

4

Эдемент вставлен.

9

[0]3-угольник:

x = 0

y = 1

x = 2

y = 3

[1]8-угольник:

x = 5

y = 6

x = 7

y = 8

[2]4-угольник:

x = 99

y = 66

x = 33333

y = 1

2

Фигура удалена

Текущий размер хранилища :Size = 2

3

Геометрическим центром вашего 3-угольника является точка:

(0.666667, 2.666667)

Геометрическим центром вашего 8-угольника является точка:

(3.625000, 9.500000)

4

Фигура индекс №0

Вершина 1 = (0, 1);

Вершина 2 = (2, 3);

Вершина 3 = (-1, 4);

Фигура индекс №1

Вершина 1 = (5, 6);

Вершина 2 = (7, 8);

Вершина 3 = (7, 11);

Вершина 4 = (5, 13);

Вершина 5 = (2, 13);

Вершина 6 = (0, 11);

Вершина 7 = (0, 8);

Вершина 8 = (2, 6);

5

Площадь вашего 3-угольника = 3

Площадь вашего 8-угольника = 39

6

Общая площадь всех фигур = 42.1175

7

Введите номер удаляемого элемента в векторе

0

9

[0]8-угольник:

x = 5

y = 6

x = 7

y = 8

0

Для выполнения операции введите соответствующую цифру:

1 - Добавить фигуру в хранилище

2 - Удалить последнюю добавленную фигуру из хранилища

3 - Вывести Геом.Центр. для всех фигур

4 - Вывести Координаты всех вершин для всех фигур

5 - Вывести площади всех фигур

6 - Вычислить общую площадь всех фигур

7 - Удалить фигуру по индексу

8 - Закрыть программу

9 - Показать вектор

0 - Вывести справку

6

Общая площадь всех фигур = 80.7667

8

root@g8-RM:/mnt/c/Users/g7/Desktop/коды/2sem/oop/3laboop#

1. **Листинг программы**

For file main.cpp:

#include <iostream>

#include <cstdio>

#include <vector>

#include <cmath>

#include <unistd.h>

using namespace std;

typedef float T;

class Point

{

public:

T x;

T y;

Point()

{

this->x = 0;

this->y = 0;

}

Point(T x, T y)

{

this->x = x;

this->y = y;

}

void GetVal()

{

cout << "x = " << x << "\ny = " << y << endl;

}

T GetX()

{

return this->x;

}

T GetY()

{

return this->y;

}

};

class Figure

{

public:

int numb;

Point a;

Point b;

Figure()

{

Point u(0, 0);

this->a = u;

this->b = u;

}

Figure(T x1, T y1, T x2, T y2)

{

Point t1(x1, y1), t2(x2, y2);

this->a = t1;

this->b = t2;

}

void Show(){

this->a.GetVal();

this->b.GetVal();

}

Point GeomCenter()

{

/\* T angle = 3.14 / 180.0 \* (360 / this->numb);

T po = sqrt((this->a.GetX() - this->b.GetX())+(this->a.GetY() - this->b.GetY()));

T r = sin(angle)\*po/2; \*/

T r, angle = 3.14 / 180.0 \* (360 / this->numb);

T x1 = this->a.GetX();

T y1 = this->a.GetY();

T x2 = this->b.GetX();

T y2 = this->b.GetY();

T xc = abs(this->a.GetX() - this->b.GetX()) / 2;

T yc = r \* cos(angle);

for (int i = 0; i < this->numb; i++)

{

xc += round(x1);

yc += round(y1);

T nextX = ((x2 - x1) \* cos(angle) + (y2 - y1) \* -sin(angle)) + x2;

T nextY = ((x2 - x1) \* sin(angle) + (y2 - y1) \* cos(angle)) + y2;

x1 = x2;

y1 = y2;

x2 = nextX;

y2 = nextY;

}

xc = xc / this->numb;

yc = yc / this->numb;

Point center(xc, yc);

printf("Геометрическим центром вашего %d-угольника является точка:\n", this->numb);

printf("(%f, %f)\n", xc, yc);

return center;

}

void FindElsePoints()

{

T x1 = this->a.GetX();

T y1 = this->a.GetY();

T x2 = this->b.GetX();

T y2 = this->b.GetY();

T angle = 3.14 / 180.0 \* (360 / this->numb);

for (int i = 0; i < this->numb; i++)

{

printf("Вершина %d = (%.0f, %.0f);\n", i + 1, x1, y1);

T nextX = ((x2 - x1) \* cos(angle) + (y2 - y1) \* -sin(angle)) + x2;

T nextY = ((x2 - x1) \* sin(angle) + (y2 - y1) \* cos(angle)) + y2;

x1 = x2;

y1 = y2;

x2 = nextX;

y2 = nextY;

}

}

T SofFig()

{

T po = sqrt((this->a.GetX() - this->b.GetX()) \* (this->a.GetX() - this->b.GetX()) + (this->a.GetY() - this->b.GetY()) \* (this->a.GetY() - this->b.GetY()));

T angle = 3.14 / 180.0 \* (360 / this->numb);

T h = cos(angle / 2) / sin(angle / 2) \* (po / 2);

T s = (po \* h \* this->numb) / 2;

printf("Площадь вашего %d-угольника = %.0f \n", this->numb, s);

return s;

}

T S()

{

T po = sqrt((this->a.GetX() - this->b.GetX()) \* (this->a.GetX() - this->b.GetX()) + (this->a.GetY() - this->b.GetY()) \* (this->a.GetY() - this->b.GetY()));

T angle = 3.14 / 180.0 \* (360 / this->numb);

T h = cos(angle / 2) / sin(angle / 2) \* (po / 2);

T s = (po \* h \* this->numb) / 2;

return s;

}

};

class EightSquare : public Figure

{

public:

EightSquare() : Figure()

{

this->numb = 8;

};

EightSquare(T x1, T y1, T x2, T y2) : Figure(x1, y1, x2, y2)

{

this->numb = 8;

};

void FindElsePoints()

{

cout << "Фигура - Восьмиугольник" << endl;

Figure ::FindElsePoints();

}

};

class Square : public Figure

{

public:

Square() : Figure()

{

this->numb = 4;

};

Square(T x1, T y1, T x2, T y2) : Figure(x1, y1, x2, y2)

{

this->numb = 4;

};

void FindElsePoints()

{

cout << "Фигура - Квадрат" << endl;

Figure ::FindElsePoints();

}

};

class Triangle : public Figure

{

public:

Triangle() : Figure()

{

this->numb = 3;

};

Triangle(T x1, T y1, T x2, T y2) : Figure(x1, y1, x2, y2)

{

this->numb = 3;

};

void FindElsePoints()

{

cout << "Фигура - Треугольник" << endl;

Figure ::FindElsePoints();

}

};

void help()

{

printf("Для выполнения операции введите соответствующую цифру:\n ");

printf("\t 1 - Добавить фигуру в хранилище\n ");

printf("\t 2 - Удалить последнюю добавленную фигуру из хранилища\n ");

printf("\t 3 - Вывести Геом.Центр. для всех фигур\n ");

printf("\t 4 - Вывести Координаты всех вершин для всех фигур\n ");

printf("\t 5 - Вывести площади всех фигур\n ");

printf("\t 6 - Вычислить общую площадь всех фигур\n ");

printf("\t 7 - Удалить фигуру по индексу\n ");

printf("\t 8 - Закрыть программу\n ");

printf("\t 9 - Показать вектор\n ");

printf("\t 0 - Вывести справку\n ");

}

int main()

{

//freopen("test\_02.txt", "r", stdin);

char enter = -1;

int c;

T xl, xr, yl, yr;

vector<Figure\*> vec;

help();

cin >> enter;

while (isdigit(enter))

{

//help();

switch (enter-48)

{

case 1:

{

cout << "Введите слева-направо координаты двух нижних вершин вашей фигуры и кол-во вершин(на выбор: 3, 4, 8)." << endl;

cin >> xl >> yl >> xr >> yr;

cout << "Введите кол-во вершин." << endl;

cin >> c;

switch (c)

{

case 3:

{

Triangle tr(xl, yl, xr, yr);

Triangle\* ptr = new Triangle;

\*ptr = tr;

vec.push\_back(ptr);

ptr = NULL;

delete ptr;

break;

}

case 4:

{

Square a(xl, yl, xr, yr);

Square\* pa = new Square;

\*pa = a;

vec.push\_back(pa);

pa = NULL;

delete pa;

break;

}

case 8:

{

EightSquare e(xl, yl, xr, yr);

EightSquare\* pe = new EightSquare;

\*pe = e;

vec.push\_back(pe);

pe = NULL;

delete pe;

break;

}

default:

{

cout << "Введено неприемлимое кол-во вершин." << endl;

sleep(1);

cout << "Удаление диска через " << endl;

for (int i = 3; i > 0; --i)

{

sleep(1);

cout << i << endl;

}

sleep(1);

cout << " Шутка =)" << endl;

sleep(1);

return 0;

}

}

cout << "Эдемент вставлен." << endl;

break;

}

case 2:

{

vec.pop\_back();

cout << "Фигура удалена\n Текущий размер хранилища :Size = " << vec.size() << endl;

break;

}

case 3:

{

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

{

vec[i]->GeomCenter();

}

break;

}

case 4:

{

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

{

cout << "Фигура индекс №" << i << endl;

vec[i]->FindElsePoints();

}

break;

}

case 5:

{

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

{

vec[i]->SofFig();

}

break;

}

case 6:

{

T x;

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

{

x += vec[i]->S();

}

cout << "Общая площадь всех фигур = " << x <<endl;

break;

}

case 7:

{

int del;

cout << "Введите номер удаляемого элемента в векторе"<<endl;

cin >> del;

vec.erase(vec.begin() + del);

break;

}

case 8:

{

return 0;

}

case 9:

{

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

{

printf("[%d]%d-угольник:\n", i,vec[i]->numb);

(\*vec[i]).a.GetVal();

(\*vec[i]).b.GetVal();

}

break;

}

case 0:

{

help();

break;

}

default:

{

cout << "Некоректный ввод\n выход из программы..." << endl;

return 0;

}

}

cin >> enter;

}

cout << "Похоже введена буква, ну в другой раз не промахнётесь по клавиатуре \n\n" <<endl;

return 0;

}

For file CMakeLists.txt:

all: oop\_exercise\_03

oop\_exercise\_03: main.o

g++ main.o -o oop\_exercise\_03

main.o: main.cpp

g++ -c main.cpp

clean:

rm -rf \*.o oop\_exercise\_03

1. **Вывод**

В процессе выполнения лабораторной работы были закреплены навыки создания классов, работы с классами, а также приобретены навыки в области работы с наследованием классов и стандартной библиотекой векторов С++ .

1. **Источники информации:**

1.Бьярне Страуструп. Программирование: принципы и практика использования C++, исправленное издание = Programming: Principles and Practice Using C++. — М.: «Вильямс», 2011. — С. 1248. — ISBN 978-5-8459-1705-8.

2.Канал #SimpleCode YouTube https://www.youtube.com/watch?v=O7ruEWCa7zc

3. Сайт cppstudio.com

<http://cppstudio.com/post/10103/>

4. Сайт cppreference.com

<https://en.cppreference.com/w/cpp/container/vector>