**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:

Группа:

Преподаватель:

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Вариант 16:

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | 8-угольник | Треугольник | Квадрат |

1. **Описание программы**

Программа содержит вектор геометрических фигур. Пользователь может добавлять фигуры в конец вектора, удалять фигуры по индексу, выводить геометрический центр, площадь, список вершин для конкретной фигуры или для всех фигур в векторе, а также посчитать сумму площадей всех фигур.

1. **Набор testcases**

|  |  |  |
| --- | --- | --- |
| № | Описание | Ввод |
| 1 | Демонстрация основных функций программы | 2  2  0 0  2  1.56  4  -1  5  -1  6  -1  7  2  1  10 10  6  0.1  4  -1  5  -1  5  0  5  1  6  -1  7  3  0  5  -1  6  -1  7  0 |

1. **Результаты выполнения тестов.**

*test\_01.txt*:

1. Show commands

2. Add figure

3. Delete figure

4. Center point

5. Print points

6. Size of figure

7. Total size

0. Exit

> 2

Choose type:

1 - Octagon

2 - Square

3 - Triangle

Type: 2

Coordinates of center: 0 0

Radius: 2

Angle: 1.56

> 4

Index (-1 to call for all figures): -1

0: (-5.96046e-08, -4.61005e-08)

> 5

Index (-1 to call for all figures): -1

0: (0.0215923, 1.99988), (-1.99988, 0.021592), (-0.0215927, -1.99988), (1.99988, -0.0215923)

> 6

Index (-1 to call for all figures): -1

0: 8

> 7

Total size of all figures: 8

> 2

Choose type:

1 - Octagon

2 - Square

3 - Triangle

Type: 1

Coordinates of center: 10 10

Radius: 6

Angle: 0.1

> 4

Index (-1 to call for all figures): -1

0: (-5.96046e-08, -4.61005e-08)

1: (10, 10)

> 5

Index (-1 to call for all figures): -1

0: (0.0215923, 1.99988), (-1.99988, 0.021592), (-0.0215927, -1.99988), (1.99988, -0.0215923)

1: (15.97, 10.599), (13.7979, 14.645), (9.401, 15.97), (5.355, 13.7979), (4.02997, 9.401), (6.20211, 5.355), (10.599, 4.02997), (14.645, 6.20211)

> 5

Index (-1 to call for all figures): 0

0: (0.0215923, 1.99988), (-1.99988, 0.021592), (-0.0215927, -1.99988), (1.99988, -0.0215923)

> 5

Index (-1 to call for all figures): 1

1: (15.97, 10.599), (13.7979, 14.645), (9.401, 15.97), (5.355, 13.7979), (4.02997, 9.401), (6.20211, 5.355), (10.599, 4.02997), (14.645, 6.20211)

> 6

Index (-1 to call for all figures): -1

0: 8

1: 101.823

> 7

Total size of all figures: 109.823

> 3

Index: 0

> 5

Index (-1 to call for all figures): -1

0: (15.97, 10.599), (13.7979, 14.645), (9.401, 15.97), (5.355, 13.7979), (4.02997, 9.401), (6.20211, 5.355), (10.599, 4.02997), (14.645, 6.20211)

> 6

Index (-1 to call for all figures): -1

0: 101.823

> 7

Total size of all figures: 101.823

> 0

1. **Листинг программы**

#include <iostream>

#include <vector>

#include <cmath>

#define PI 3.14159265f

//точка многоугольника

using point = std::pair<float, float>;

//родительский класс для всех фигур

class Figure

{

public:

//вычисление геометрического центра фигуры

point getCenter()

{

point center = std::make\_pair(0.0f, 0.0f);

for (point p : m\_points)

{

center.first += p.first;

center.second += p.second;

}

center.first /= m\_points.size();

center.second /= m\_points.size();

return center;

}

//вывод координат вершин фигуры

void print()

{

bool comma = false; //печатать запятую перед точкой или нет

for (point p : m\_points)

{

if (comma) std::cout << ", ";

comma = true;

std::cout << "(" << p.first << ", " << p.second << ")";

}

}

//вычисление площади

float size()

{

float S = 0.0f;

for (int i = 0; i < m\_points.size() - 1; i++)

S += triag(m\_points[0], m\_points[i], m\_points[i + 1]);

return S;

}

protected:

//точки многоугольника

std::vector<point> m\_points;

//площадь треугольника по координатам вершин

//S = 1/2 \* abs(det(x1 - x3, y1 - y3; x2 - x3, y2 - y3))

float triag(point& a, point& b, point& c)

{

return 0.5f \* abs((a.first - c.first) \* (b.second - c.second) -

(b.first - c.first) \* (a.second - c.second));

}

};

//Любую фигуру вращения можно задать координатами центра, радиусом описанной окружности и углом поворота

//8-угольник

class Octagon : public Figure

{

public:

//заполняем вектор вершин

Octagon(float x, float y, float r, float a)

{

for (int i = 0; i < 8; i++)

{

float phi = a + i \* PI / 4.0f;

m\_points.push\_back(std::make\_pair(r \* cosf(phi) + x, r \* sinf(phi) + y));

}

}

};

//Квадрат

class Square : public Figure

{

public:

//заполняем вектор вершин

Square(float x, float y, float r, float a)

{

for (int i = 0; i < 4; i++)

{

float phi = a + i \* PI / 2.0f;

m\_points.push\_back(std::make\_pair(r \* cosf(phi) + x, r \* sinf(phi) + y));

}

}

};

//Треугольник

class Triangle : public Figure

{

public:

//заполняем вектор вершин

Triangle(float x, float y, float r, float a)

{

for (int i = 0; i < 3; i++)

{

float phi = a + i \* PI / 1.5f;

m\_points.push\_back(std::make\_pair(r \* cosf(phi) + x, r \* sinf(phi) + y));

}

}

};

//список команд

void showCommands()

{

std::cout <<

"1. Show commands" << std::endl <<

"2. Add figure" << std::endl <<

"3. Delete figure" << std::endl <<

"4. Center point" << std::endl <<

"5. Print points" << std::endl <<

"6. Size of figure" << std::endl <<

"7. Total size" << std::endl <<

"0. Exit" << std::endl;

}

int main()

{

//вектор фигур

std::vector<Figure\*> figures;

showCommands();

//цикл программы

bool loop = true;

while (loop)

{

//читаем введённую команду

std::cout << "> ";

int command;

std::cin >> command;

switch (command)

{

case 0:

loop = false;

break;

case 1:

showCommands();

break;

case 2:

{

std::cout << "Choose type:" << std::endl <<

"1 - Octagon" << std::endl <<

"2 - Square" << std::endl <<

"3 - Triangle" << std::endl <<

"Type: ";

int type;

std::cin >> type;

if (type < 1 || type > 3)

std::cout << "Unknown type" << std::endl;

else

{

float x, y, r, a;

std::cout << "Coordinates of center: ";

std::cin >> x >> y;

std::cout << "Radius: ";

std::cin >> r;

std::cout << "Angle: ";

std::cin >> a;

switch (type)

{

case 1:

figures.push\_back(new Octagon(x, y, r, a));

break;

case 2:

figures.push\_back(new Square(x, y, r, a));

break;

case 3:

figures.push\_back(new Triangle(x, y, r, a));

break;

}

}

break;

}

case 3:

{

std::cout << "Index: ";

int index;

std::cin >> index;

if (index < 0 || index >= figures.size())

std::cout << "Index out of bounds" << std::endl;

else

{

delete figures[index];

figures.erase(figures.begin() + index);

}

break;

}

case 4:

{

std::cout << "Index (-1 to call for all figures): ";

int index;

std::cin >> index;

if (index == -1)

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

{

point center = figures[i]->getCenter();

std::cout << i << ": (" << center.first << ", " << center.second << ")" << std::endl;

}

else if (index < -1 || index >= figures.size())

std::cout << "Index out of bounds" << std::endl;

else

{

point center = figures[index]->getCenter();

std::cout << index << ": (" << center.first << ", " << center.second << ")" << std::endl;

}

break;

}

case 5:

{

std::cout << "Index (-1 to call for all figures): ";

int index;

std::cin >> index;

if (index == -1)

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

{

std::cout << i << ": ";

figures[i]->print();

std::cout << std::endl;

}

else if (index < -1 || index >= figures.size())

std::cout << "Index out of bounds" << std::endl;

else

{

std::cout << index << ": ";

figures[index]->print();

std::cout << std::endl;

}

break;

}

case 6:

{

std::cout << "Index (-1 to call for all figures): ";

int index;

std::cin >> index;

if (index == -1)

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

{

std::cout << i << ": " << figures[i]->size() << std::endl;

}

else if (index < -1 || index >= figures.size())

std::cout << "Index out of bounds" << std::endl;

else

{

std::cout << index << ": " << figures[index]->size() << std::endl;

}

break;

}

case 7:

{

float total = 0.0f;

for (Figure\* f : figures)

total += f->size();

std::cout << "Total size of all figures: " << total << std::endl;

break;

}

default:

std::cout << "Unknown command" << std::endl;

break;

}

}

return 0;

}

1. **Выводы:**

Изучены механизмы работы с наследованием в С++. Разработана программа на языке C++, использующая данные механизмы для работы с вектором, содержащим различные виды фигур.

**СПИСОК ЛИТЕРАТУРЫ**

1. Наследование в C++ [электронный ресурс]. URL: <https://habr.com/ru/post/445948/>
2. std::vector [электронный ресурс]. URL: <https://ru.cppreference.com/w/cpp/container/vector>