**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: Чернобаев Андрей Александрович

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата: 9.11.2020

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:

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.

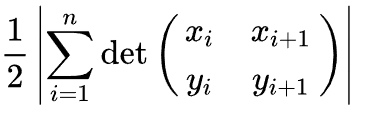
• Необходимо уметь вычислять общую площадь фигур в массиве.

• Удалять из массива фигуру по индексу.

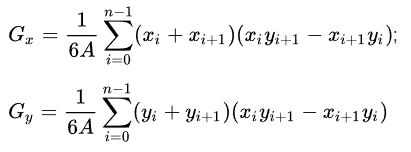
|  |  |  |  |
| --- | --- | --- | --- |
| **Вариант** | **Фигура №1** | **Фигура №2** | **Фигура №3** |
| 6. | 5-угольник | 6-угольник | 8-угольник |

1. Описание программы.

Площадь многоугольников вычисляется с помощью формулы площади Гаусса:



Геометрический центр (барицентр) многоугольника вычисляется следующим образом:



Программа состоит из двух файлов:

accessory.h - Заголовочный файл, содержит дополнительные функции программы, облегчающие ввод-вывод.

main.cpp - основная часть программы.

Файл accessory.h состоит из:

* 3-х лямбдно-функций.
* 2-ух функций, реализующих split.
* функция help.

Файл main.cpp состоит из:

* Класса Pair с перегруженными арифметическими операторами для хранения точек
* функции get\_det для вычисления детерминанта второго порядка
* функции gauss\_area для вычисления площади многоугольника по формуле Гаусса
* функции get\_centroid для вычисления центроида многоугольника
* базового класса Figure
* унаследованных классов Pentagon, Hexagon и Octagon

1. Набор тестов.

**из файла test\_01.txt:**

9.459 9.637 3.641 6.85 1.375 0.045 4.541 2.825 3.777 8.048 pentagon

7.991 2.683 8.738 4.904 2.531 5.495 6.077 8.981 6.288 0.363 9.728 7.885 hexagon

3.337 6.088 8.338 1.906 6.313 9.73 3.742 0.787 6.408 9.553 0.424 0.048 6.402 1.798 0.092 7.295 octagon

area

go

delete 0

9.251 4.554 4.228 2.259 4.57 8.39 3.275 1.01 5.919 7.915 4.804 9.958 hexagon

4.403 2.467 1.84 5.504 5.77 3.696 4.146 2.865 4.176 5.922 2.42 3.775 8.17 8.673 8.152 5.479 octagon

area

go

exit

**Из файла test\_02.txt:**

0.996 1.616 5.656 4.529 7.597 6.314 7.471 9.64 7.954 1.979 pentagon

4.526 4.185 7.221 2.085 9.394 9.438 9.648 3.4 3.525 0.368 8.18 8.396 hexagon

0.514 4.906 1.011 4.006 2.425 1.316 6.298 1.629 5.868 7.293 9.596 1.523 1.822 7.192 7.836 5.644 octagon

area

go

delete 1

3.599 2.061 9.535 0.612 1.173 6.095 7.754 8.708 7.223 8.919 pentagon

4.971 5.602 9.05 5.105 8.564 2.623 8.836 7.458 8.889 4.327 5.21 0.201 hexagon

area

go

exit

1. Результаты выполнения тестов.

|  |  |
| --- | --- |
| input | output |
| 9.459 9.637 3.641 6.85 1.375 0.045 4.541 2.825 3.777 8.048 pentagon  7.991 2.683 8.738 4.904 2.531 5.495 6.077 8.981 6.288 0.363 9.728 7.885 hexagon  3.337 6.088 8.338 1.906 6.313 9.73 3.742 0.787 6.408 9.553 0.424 0.048 6.402 1.798 0.092 7.295 octagon  area  go  delete 0  9.251 4.554 4.228 2.259 4.57 8.39 3.275 1.01 5.919 7.915 4.804 9.958 hexagon  4.403 2.467 1.84 5.504 5.77 3.696 4.146 2.865 4.176 5.922 2.42 3.775 8.17 8.673 8.152 5.479 octagon  area  go  exit | write exit for exit  write help for help  write area to calculate sum of areas  write index delete to remove figure by index  write go to show common function 1-3 for all array  input format x\_1 y\_1 x\_2 y\_2 ... x\_n y\_n figure  query 3: total area of all figures is 29.0722  query 4:  Pentagon  x = 9.459000 y = 9.637000  x = 3.641000 y = 6.850000  x = 1.375000 y = 0.045000  x = 4.541000 y = 2.825000  x = 3.777000 y = 8.048000  center: x = 1.747868 y = 0.495626  area: 5.13991  Hexagon  x = 7.991000 y = 2.683000  x = 8.738000 y = 4.904000  x = 2.531000 y = 5.495000  x = 6.077000 y = 8.981000  x = 6.288000 y = 0.363000  x = 9.728000 y = 7.885000  center: x = 1.099739 y = 13.658043  area: 2.22044  Octagon  x = 3.337000 y = 6.088000  x = 8.338000 y = 1.906000  x = 6.313000 y = 9.730000  x = 3.742000 y = 0.787000  x = 6.408000 y = 9.553000  x = 0.424000 y = 0.048000  x = 6.402000 y = 1.798000  x = 0.092000 y = 7.295000  center: x = 5.278542 y = 3.881495  area: 21.7118  query 5:  figure 0 was delete  query 8: total area of all figures is 51.5773  query 9:  Hexagon  x = 7.991000 y = 2.683000  x = 8.738000 y = 4.904000  x = 2.531000 y = 5.495000  x = 6.077000 y = 8.981000  x = 6.288000 y = 0.363000  x = 9.728000 y = 7.885000  center: x = 1.099739 y = 13.658043  area: 2.22044  Octagon  x = 3.337000 y = 6.088000  x = 8.338000 y = 1.906000  x = 6.313000 y = 9.730000  x = 3.742000 y = 0.787000  x = 6.408000 y = 9.553000  x = 0.424000 y = 0.048000  x = 6.402000 y = 1.798000  x = 0.092000 y = 7.295000  center: x = 5.278542 y = 3.881495  area: 21.7118  Hexagon  x = 9.251000 y = 4.554000  x = 4.228000 y = 2.259000  x = 4.570000 y = 8.390000  x = 3.275000 y = 1.010000  x = 5.919000 y = 7.915000  x = 4.804000 y = 9.958000  center: x = 6.752066 y = 5.283382  area: 12.7257  Octagon  x = 4.403000 y = 2.467000  x = 1.840000 y = 5.504000  x = 5.770000 y = 3.696000  x = 4.146000 y = 2.865000  x = 4.176000 y = 5.922000  x = 2.420000 y = 3.775000  x = 8.170000 y = 8.673000  x = 8.152000 y = 5.479000  center: x = 5.796588 y = 5.159007  area: 14.9193 |
| 0.996 1.616 5.656 4.529 7.597 6.314 7.471 9.64 7.954 1.979 pentagon  4.526 4.185 7.221 2.085 9.394 9.438 9.648 3.4 3.525 0.368 8.18 8.396 hexagon  0.514 4.906 1.011 4.006 2.425 1.316 6.298 1.629 5.868 7.293 9.596 1.523 1.822 7.192 7.836 5.644 octagon  area  go  delete 1  3.599 2.061 9.535 0.612 1.173 6.095 7.754 8.708 7.223 8.919 pentagon  4.971 5.602 9.05 5.105 8.564 2.623 8.836 7.458 8.889 4.327 5.21 0.201 hexagon  area  go  exit | write exit for exit  write help for help  write area to calculate sum of areas  write index delete to remove figure by index  write go to show common function 1-3 for all array  input format x\_1 y\_1 x\_2 y\_2 ... x\_n y\_n figure  query 3: total area of all figures is 30.3167  query 4:  Pentagon  x = 0.996000 y = 1.616000  x = 5.656000 y = 4.529000  x = 7.597000 y = 6.314000  x = 7.471000 y = 9.640000  x = 7.954000 y = 1.979000  center: x = 5.636802 y = 3.283564  area: 14.1349  Hexagon  x = 4.526000 y = 4.185000  x = 7.221000 y = 2.085000  x = 9.394000 y = 9.438000  x = 9.648000 y = 3.400000  x = 3.525000 y = 0.368000  x = 8.180000 y = 8.396000  center: x = 8.076607 y = 3.457405  area: 8.47598  Octagon  x = 0.514000 y = 4.906000  x = 1.011000 y = 4.006000  x = 2.425000 y = 1.316000  x = 6.298000 y = 1.629000  x = 5.868000 y = 7.293000  x = 9.596000 y = 1.523000  x = 1.822000 y = 7.192000  x = 7.836000 y = 5.644000  center: x = 0.768624 y = 1.382804  area: 7.70583  query 5:  figure 1 was delete  query 8: total area of all figures is 38.4854  query 9:  Pentagon  x = 0.996000 y = 1.616000  x = 5.656000 y = 4.529000  x = 7.597000 y = 6.314000  x = 7.471000 y = 9.640000  x = 7.954000 y = 1.979000  center: x = 5.636802 y = 3.283564  area: 14.1349  Octagon  x = 0.514000 y = 4.906000  x = 1.011000 y = 4.006000  x = 2.425000 y = 1.316000  x = 6.298000 y = 1.629000  x = 5.868000 y = 7.293000  x = 9.596000 y = 1.523000  x = 1.822000 y = 7.192000  x = 7.836000 y = 5.644000  center: x = 0.768624 y = 1.382804  area: 7.70583  Pentagon  x = 3.599000 y = 2.061000  x = 9.535000 y = 0.612000  x = 1.173000 y = 6.095000  x = 7.754000 y = 8.708000  x = 7.223000 y = 8.919000  center: x = 1.564813 y = 11.955119  area: 4.02504  Hexagon  x = 4.971000 y = 5.602000  x = 9.050000 y = 5.105000  x = 8.564000 y = 2.623000  x = 8.836000 y = 7.458000  x = 8.889000 y = 4.327000  x = 5.210000 y = 0.201000  center: x = 6.629403 y = 3.650526  area: 12.6196 |

1. Листинг программы.

Файл accessory.h:

#ifndef TASK3\_ACCESSORY\_H

#define TASK3\_ACCESSORY\_H

#include <string>

#include <vector>

#include <sstream>

auto string\_to\_int = [](const std::string& s) {return std::stoi(s);};

auto string\_to\_double = [](const std::string& s) {return std::stod(s);};

auto int\_to\_string = [](const int i) {return std::to\_string(i);};

template <typename Out>

void split(const std::string &s, char delim, Out result) {

std::istringstream iss(s);

std::string item;

while (std::getline(iss, item, delim)) {

\*result++ = item;

}

}

std::vector<std::string> split(const std::string &s, char delim) {

std::vector<std::string> elems;

split(s, delim, std::back\_inserter(elems));

return elems;

}

void help(){

std::cout << "write exit for exit" << std::endl;

std::cout << "write help for help" << std::endl;

std::cout << "write area to calculate sum of areas" << std::endl;

std::cout << "write index delete to remove figure by index" << std::endl;

std::cout << "write go to show common function 1-3 for all array" << std::endl;

std::cout << "input format x\_1 y\_1 x\_2 y\_2 ... x\_n y\_n figure" << std::endl;

std::cout << "" << std::endl;

}

#endif //TASK3\_ACCESSORY\_H

Файл main.cpp:

#include <iostream>

#include <vector>

#include <string>

#include <ctime>

#include <cmath>

#include "accessory.h"

/\*

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 buffer 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.

var 6

5-угольник

6-угольник

8-угольник

\*/

class Pair{

public:

double x;

double y;

Pair() = default;

Pair(double x, double y){

this->x = x;

this->y = y;

}

friend std::ostream& operator<<(std::ostream& os, const Pair& pair){

os << "x = " + std::to\_string(pair.x) + " y = " + std::to\_string(pair.y) << std::endl;

return os;

}

Pair operator+(const Pair& rhs) const{

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

}

Pair operator-(const Pair& rhs) const{

return {x - rhs.x,y - rhs.y};

}

Pair operator/(double d) const{

return {x / d,y / d};

}

Pair& operator+=(const Pair& rhs) {

x += rhs.x;

y += rhs.y;

return \*this;

}

};

double get\_det(const double x1, const double y1,

const double x2, const double y2){

return x1 \* y2 - x2 \* y1;

}

double gauss\_area(const std::vector<Pair>& nodes){

double out = 0.0;

auto count = nodes.size();

for(size\_t i = 0; i < (count - 1); i++){

out += get\_det(nodes[i].x,nodes[i].y,nodes[i+1].x,nodes[i+1].y);

}

out += get\_det(nodes[count-1].x,nodes[count-1].y,nodes[0].x,nodes[0].y);

return fabs(0.5 \* out);

}

Pair get\_centroid(const std::vector<Pair>& nodes){

Pair centroid = {0, 0};

double signedArea = 0.0;

double x0 = 0.0;

double y0 = 0.0;

double x1 = 0.0;

double y1 = 0.0;

double a = 0.0;

int i = 0;

for (i = 0; i < nodes.size() - 1; ++i) {

x0 = nodes[i].x;

y0 = nodes[i].y;

x1 = nodes[i+1].x;

y1 = nodes[i+1].y;

a = get\_det(x0,y0,x1,y1);

signedArea += a;

centroid.x += (x0 + x1)\*a;

centroid.y += (y0 + y1)\*a;

}

x0 = nodes[i].x;

y0 = nodes[i].y;

x1 = nodes[0].x;

y1 = nodes[0].y;

a = get\_det(x0,y0,x1,y1);

signedArea += a;

centroid.x += (x0 + x1)\*a;

centroid.y += (y0 + y1)\*a;

signedArea \*= 0.5;

centroid.x /= (6.0\*signedArea);

centroid.y /= (6.0\*signedArea);

return centroid;

}

class Figure{

public:

std::vector<Pair> nodes;

virtual Pair get\_center() = 0;

virtual double get\_area() = 0;

std::string name;

friend std::ostream& operator<<(std::ostream& os, const Figure& figure){

os << figure.name << std::endl;

for(auto node : figure.nodes){

os << node;

}

return os;

}

};

class Pentagon : public Figure{

public:

explicit Pentagon(std::vector<Pair>&& nodes\_){

nodes = nodes\_;

name = "Pentagon";

if (nodes.size() != 5){

std::string error = std::string("size error in Pentagon") + "size is " + int\_to\_string(nodes\_.size());

perror(error.c\_str());

}

}

Pair get\_center() override{

return ::get\_centroid(nodes);

}

double get\_area() override{

return gauss\_area(nodes);

}

};

class Hexagon : public Figure{

public:

explicit Hexagon(std::vector<Pair>&& nodes\_){

nodes = nodes\_;

name = "Hexagon";

if (nodes.size() != 6){

std::string error = std::string("size error in Hexagon") + "size is " + int\_to\_string(nodes\_.size());

perror(error.c\_str());

}

}

Pair get\_center() override{

return ::get\_centroid(nodes);

}

double get\_area() override{

return gauss\_area(nodes);

}

};

class Octagon : public Figure{

public:

explicit Octagon(std::vector<Pair>&& nodes\_){

nodes = nodes\_;

name = "Octagon";

if (nodes.size() != 8){

std::string error = std::string("size error in Octagon") + "size is " + int\_to\_string(nodes\_.size());

perror(error.c\_str());

}

}

Pair get\_center() override{

return ::get\_centroid(nodes);

}

double get\_area() override{

return gauss\_area(nodes);

}

};

int main() {

std::vector<Figure\*> figures;

std::string cmd;

std::vector<std::string> out;

std::stringstream buffer;

int count = 0;

help();

while (true){

getline(std::cin, cmd);

std::string prefix = std::string("query ") + int\_to\_string(count) + ": ";

if (cmd == "exit"){

break;

}

else if (cmd == "help"){

help();

}

else if (cmd == "area"){

if (figures.empty()){

buffer << prefix << "no figures in array" << std::endl;

}

else {

double sum = 0.0;

for(auto el : figures){

sum += el->get\_area();

}

buffer << prefix << "total area of all figures is " << sum << std::endl;

}

}

else if (cmd.find("delete") != std::string::npos) {

int index = string\_to\_int(split(cmd,' ')[1]);

if (index >= figures.size()){

buffer << prefix << "index > figures size" << std::endl;

}

else {

buffer << prefix << std::endl;

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

buffer << "figure " + int\_to\_string(index) + " was delete" << std::endl;

}

}

else if (cmd == "go"){

buffer << prefix << std::endl;

for(auto figure : figures){

buffer << \*figure;

buffer << "center: " << figure->get\_center();

buffer << "area: " << figure->get\_area() <<std::endl;

}

}

else {

std::vector<std::string> array = split(cmd,' ');

std::vector<Pair> numbers;

numbers.resize((array.size() - 1) / 2);

int j = 0;

for(size\_t i = 0;i < numbers.size() \* 2 ; i += 2){

numbers[j] = {string\_to\_double(array[i]),string\_to\_double(array[i+1])};

j++;

}

if (array.back() == "pentagon"){

figures.emplace\_back(new Pentagon(std::move(numbers)));

}

else if (array.back() == "hexagon"){

figures.emplace\_back(new Hexagon(std::move(numbers)));

}

else if (array.back() == "octagon"){

figures.emplace\_back(new Octagon(std::move(numbers)));

}

}

count++;

}

std::cout << buffer.str() << std::endl;

return 0;

}

1. Выводы.

Я научился реализовывать механизмы наследования в C++.

1. Список использованных источников.

1 C++ Inheritance [Электронный ресурс].

URL: <https://www.tutorialspoint.com/cplusplus/cpp_inheritance.htm>

2 Barycenter [Электронный ресурс].

URL: <https://en.wikipedia.org/wiki/Barycenter>

3 std::move [Электронный ресурс].

URL: <https://en.cppreference.com/w/cpp/utility/move>

4 Shoelace formula [Электронный ресурс].

URL: <https://en.wikipedia.org/wiki/Shoelace_formula>

5 std::string::find [Электронный ресурс].

URL: http://www.cplusplus.com/reference/string/string/find/