**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: Бутырев Даниил Вячеславович

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

<https://github.com/SteepturN/oop_exercise_03>

1. Постановка задачи

16 var: 8-угольник, Треугольник, Квадрат.

Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure.

Все классы должны поддерживать набор общих методов:

Вычисление геометрического центра фигуры.

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

Вычисление площади фигуры.

Программа должна иметь следующие возможности:

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.

вычислять общую площадь фигур в массиве.

удалять из массива фигуру по индексу.

2. Описание программы

Программа состоит из 6 файлов:

1. main.cpp - здесь находится пользовательский интерпретатор, происходят манипуляции с объектами класса Complex и их результаты отправляются на стандартный поток вывода, здесь все фигуры содержатся в массиве Figure\*, виртуально вызываются методы этого класса, которые переопределены в классах-наследниках
2. Read\_input.cpp - здесь происходит ввод данных,
3. Figure.cpp - класс, от которого наследуются все остальные
4. Triangle.cpp - класс-наследник
5. Square.cpp - класс-наследник
6. Octagon.cpp -класс-наследник

3. Набор тестов

p t 0 0 1 0 0 1 - добавляем треугольник с соответствующими координатами

p t 0 2 0 0 0 2 - добавляем треугольник с соответствующими координатами

p t 2 2 0 0 1 0 - добавляем треугольник с соответствующими координатами

p o 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 - добавляем восьмиугольник с соответствующими координатами

p o 0 0 1 1 2 4 5 2 0 -1 1 -1 2 -2 4 -1 - добавляем восьмиугольник с соответствующими координатами

p o 3 3 1 4 3 8 6 8 8 7 7 6 6 3 5 2 - добавляем восьмиугольник с соответствующими координатами

p s 1 1 1 2 2 2 2 2 - добавляем квадрат с соответствующими координатами

p s 3 3 4 4 1 -1 -1 1 - добавляем квадрат с соответствующими координатами

p s 4 4 3 -3 4 -3 8 -5 - добавляем квадрат с соответствующими координатами

coor 0 - показываем координаты 0-вой фигуры

coor 1 - показываем координаты 1-ой фигуры

coor 2 - показываем координаты 2-ой фигуры

coor 3 - показываем координаты 3-ей фигуры

coor 4 - показываем координаты 4-ой фигуры

coor 5 - показываем координаты 5-ой фигуры

coor 6 - показываем координаты 6-ой фигуры

coor 7 - показываем координаты 7-ой фигуры

coor 8 - показываем координаты 8-ой фигуры

cen 0 - показываем центр 0-ой фигуры

cen 1 - показываем центр 1-ой фигуры

cen 2 - показываем центр 2-ой фигуры

cen 3 - показываем центр 3-ей фигуры

cen 4 - показываем центр 4-ой фигуры

cen 5 - показываем центр 5-ой фигуры

cen 6 - показываем центр 6-ой фигуры

cen 7 - показываем центр 7-ой фигуры

cen 8 - показываем центр 8-ой фигуры

area 0 - показываем площадь 0-й фигуры

area 1 - показываем площадь 1-й фигуры

area 2 - показываем площадь 2-й фигуры

area 3 - показываем площадь 3-й фигуры

area 4 - показываем площадь 4-й фигуры

area 5 - показываем площадь 5-й фигуры

area 6 - показываем площадь 6-й фигуры

area 7 - показываем площадь 7-й фигуры

area 8 - показываем площадь 8-й фигуры

ars - показываем полную площадь

d 1 - удаляем текущий 1 элемент

d 4 - удаляем текущий 4 элемент

d 6 - удаляем текущий 6 элемент

ars - показываем полную площадь

exit - выходим

p t 0 0 1.5 0 0 1.199 - добавляем треугольник с соответствующими координатами

p t 0 2.41 0 0 0 -2.3415 - добавляем треугольник с соответствующими координатами

p t -2.5314 -2.1234 0 0 1.1234 0 - добавляем треугольник с соответствующими координатами

p o 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 - добавляем восьмиугольник с соответствующими координатами

p o 1 1 -2 2 4.1243 3 5 8.54 9 0.53 -5 5 4 4 -3 1 - добавляем восьмиугольник с соответствующими координатами

p o 1 2 -4.25 5 0 -9.25 -5 5 4 4 5 1 -9.25 1 0 -5 - добавляем восьмиугольник с соответствующими координатами

p s 1 1 1 2 -2.01 2 -2 2 - добавляем квадрат с соответствующими координатами

p s 3 3 4 -4.12 1 -1 -1 1 - добавляем квадрат с соответствующими координатами

p s 4 4 3 -3 -4.14 -3 8 -5 - добавляем квадрат с соответствующими координатами

coor 0 - показываем координаты 0-ой фигуры

coor 1 - показываем координаты 1-ой фигуры

coor 2 - показываем координаты 2-ой фигуры

coor 3 - показываем координаты 3-ей фигуры

coor 4 - показываем координаты 4-ой фигуры

coor 5 - показываем координаты 5-ой фигуры

coor 6 - показываем координаты 6-ой фигуры

coor 7 - показываем координаты 7-ой фигуры

coor 8 - показываем координаты 8-ой фигуры

cen 0 - показываем центр 0-ой фигуры

cen 1 - показываем центр 1-ой фигуры

cen 2 - показываем центр 2-ой фигуры

cen 3 - показываем центр 3-ой фигуры

cen 4 - показываем центр 4-ой фигуры

cen 5 - показываем центр 5-ой фигуры

cen 6 - показываем центр 6-ой фигуры

cen 7 - показываем центр 7-ой фигуры

cen 8 - показываем центр 8-ой фигуры

area 0 - показываем площадь 0-й фигуры

area 1 - показываем площадь 1-й фигуры

area 2 - показываем площадь 2-й фигуры

area 3 - показываем площадь 3-й фигуры

area 4 - показываем площадь 4-й фигуры

area 5 - показываем площадь 5-й фигуры

area 6 - показываем площадь 6-й фигуры

area 7 - показываем площадь 7-й фигуры

area 8 - показываем площадь 8-й фигуры

ars - показываем полную площадь

d 1 - удаляем текущий 1 элемент

d 4 - удаляем текущий 4 элемент

d 6 - удаляем текущий 6 элемент

ars - показываем полную площадь

exit - выходим

4. Результаты выполнения тестов

You can use

--put (triangle|square|octagon): p [t,s,o] [(point) n times, n(triangle)=3; n(square)=4; n(octagon)=8]

--center (figure number): cen (number of figure)

--coordinates (figure number): coor (number of figure)

--area (figure number): area (number of figure)

--area summ: ars

--delete (figure number): d (number of figure)

--exit

You've added t with this verteces: (0, 0)(1, 0)(0, 1)

You've added t with this verteces: (0, 2)(0, 0)(0, 2)

You've added t with this verteces: (2, 2)(0, 0)(1, 0)

You've added o with this verteces: (0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)

You've added o with this verteces: (0, 0)(1, 1)(2, 4)(5, 2)(0, -1)(1, -1)(2, -2)(4, -1)

You've added o with this verteces: (3, 3)(1, 4)(3, 8)(6, 8)(8, 7)(7, 6)(6, 3)(5, 2)

You've added s with this verteces: (1, 1)(1, 2)(2, 2)

You've added s with this verteces: (3, 3)(4, 4)(1, -1)

You've added s with this verteces: (4, 4)(3, -3)(4, -3)

0 coordinates are (0, 0)(1, 0)(0, 1)

1 coordinates are (0, 2)(0, 0)(0, 2)

2 coordinates are (2, 2)(0, 0)(1, 0)

3 coordinates are (0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)

4 coordinates are (0, 0)(1, 1)(2, 4)(5, 2)(0, -1)(1, -1)(2, -2)(4, -1)

5 coordinates are (3, 3)(1, 4)(3, 8)(6, 8)(8, 7)(7, 6)(6, 3)(5, 2)

6 coordinates are (1, 1)(1, 2)(2, 2)

7 coordinates are (3, 3)(4, 4)(1, -1)

8 coordinates are (4, 4)(3, -3)(4, -3)

0 center is: x = 0.333333, y = 0.333333

1 center is: x = 0, y = 1.33333

2 center is: x = 1, y = 0.666667

3 center is: x = 0, y = 0

4 center is: x = 1.875, y = 0.25

5 center is: x = 4.875, y = 5.125

6 center is: x = 1.33333, y = 1.66667

7 center is: x = 2.66667, y = 2

8 center is: x = 3.66667, y = -0.666667

0 area is 0.5

1 area is 0

2 area is 1

3 area is 0

4 area is 6

5 area is 25.5

6 area is 0.5

7 area is 1

8 area is 3.5

the whole area is 38

You've deleted figure #1

You've deleted figure #4

You've deleted figure #6

the whole area is 9

You can use

--put (triangle|square|octagon): p [t,s,o] [(point) n times, n(triangle)=3; n(square)=4; n(octagon)=8]

--center (figure number): cen (number of figure)

--coordinates (figure number): coor (number of figure)

--area (figure number): area (number of figure)

--area summ: ars

--delete (figure number): d (number of figure)

--exit

You've added t with this verteces: (0, 0)(1.5, 0)(0, 1.199)

You've added t with this verteces: (0, 2.41)(0, 0)(0, -2.3415)

You've added t with this verteces: (-2.5314, -2.1234)(0, 0)(1.1234, 0)

You've added o with this verteces: (0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)

You've added o with this verteces: (1, 1)(-2, 2)(4.1243, 3)(5, 8.54)(9, 0.53)(-5, 5)(4, 4)(-3, 1)

You've added o with this verteces: (1, 2)(-4.25, 5)(0, -9.25)(-5, 5)(4, 4)(5, 1)(-9.25, 1)(0, -5)

You've added s with this verteces: (1, 1)(1, 2)(-2.01, 2)

You've added s with this verteces: (3, 3)(4, -4.12)(1, -1)

You've added s with this verteces: (4, 4)(3, -3)(-4.14, -3)

0 coordinates are (0, 0)(1.5, 0)(0, 1.199)

1 coordinates are (0, 2.41)(0, 0)(0, -2.3415)

2 coordinates are (-2.5314, -2.1234)(0, 0)(1.1234, 0)

3 coordinates are (0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)(0, 0)

4 coordinates are (1, 1)(-2, 2)(4.1243, 3)(5, 8.54)(9, 0.53)(-5, 5)(4, 4)(-3, 1)

5 coordinates are (1, 2)(-4.25, 5)(0, -9.25)(-5, 5)(4, 4)(5, 1)(-9.25, 1)(0, -5)

6 coordinates are (1, 1)(1, 2)(-2.01, 2)

7 coordinates are (3, 3)(4, -4.12)(1, -1)

8 coordinates are (4, 4)(3, -3)(-4.14, -3)

0 center is: x = 0.5, y = 0.399667

1 center is: x = 0, y = 0.0228333

2 center is: x = -0.469333, y = -0.7078

3 center is: x = 0, y = 0

4 center is: x = 1.64054, y = 3.13375

5 center is: x = -1.0625, y = 0.46875

6 center is: x = -0.00333333, y = 1.66667

7 center is: x = 2.66667, y = -0.706667

8 center is: x = 0.953333, y = -0.666667

0 area is 0.89925

1 area is 0

2 area is 1.19271

3 area is 0

4 area is 22.2935

5 area is 8.03125

6 area is 1.505

7 area is 9.12

8 area is 24.99

the whole area is 68.0318

You've deleted figure #1

You've deleted figure #4

You've deleted figure #6

the whole area is 35.0105

5. Листинг программы

main.cpp:

#include <iostream>

#include "Square.cpp"

#include "Octagon.cpp"

#include "Figure.cpp"

#include "Triangle.cpp"

#include "Read\_input.cpp"

#include <cstdio>

#include <set>

int main(){ // можно вынести названия команд как константы

char a[] = "You can use\n\

--put (triangle|square|octagon): p [t,s,o] [(point) n times, n(triangle)=3; n(square)=4; n(octagon)=8]\n\

--center (figure number): cen (number of figure)\n\

--coordinates (figure number): coor (number of figure)\n\

--area (figure number): area (number of figure)\n\

--area summ: ars\n\

--delete (figure number): d (number of figure)\n\

--exit\n";

std::vector<Figure\*> figures;

char ch(' ');

char command[20];

std::set<std::string> valid\_commands = {"p", "cen", "coor", "area", "ars", "d", "exit"};

std::set<std::string> valid\_figures = {"t", "s", "o"};

std::cout << a;

do {

beginning\_of\_the\_entire\_input:

bool exit\_if\_successful\_input = false;

do{

read\_return\_t answer = get\_command(valid\_commands, command);

switch(answer) {

case END\_OF\_FILE: return 0;

case END\_OF\_LINE: continue;

case VALID\_INPUT: exit\_if\_successful\_input = true; break;

case INVALID\_INPUT:

do ch=getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

if(ch == EOF) return 0;

else break;

}

} while(!exit\_if\_successful\_input);

std::string&& command\_string = static\_cast<std::string>(command);

if(command\_string == "exit") return 0;

if(command\_string == "p") {

char type\_of\_figure[10];

read\_return\_t answer = get\_command(valid\_figures, type\_of\_figure);

switch(answer) {

case END\_OF\_FILE: return 0;

case END\_OF\_LINE: continue;

case VALID\_INPUT: break;

case INVALID\_INPUT:

do ch=getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

if(ch == EOF) return 0;

else goto beginning\_of\_the\_entire\_input;

}

std::string&& figure\_type\_string = static\_cast<std::string>(type\_of\_figure);

int num\_of\_verteces(0);

std::pair<double, double> vertex;

if(figure\_type\_string == "t") {

num\_of\_verteces = 3; //можно лучше сделать - объявить константами

figures.push\_back(new Triangle);

} else if(figure\_type\_string == "s") {

num\_of\_verteces = 4;

figures.push\_back(new Square);

} else if(figure\_type\_string == "o") {

num\_of\_verteces = 8;

figures.push\_back(new Octagon);

}

for(int i = 0; i < num\_of\_verteces; i++) { //add usage of \n?

if((get\_value<double>(vertex.first) != VALID\_INPUT) || (get\_value<double>(vertex.second) != VALID\_INPUT)) {

do ch=getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

figures.pop\_back();

if(ch == EOF) return 0;

else goto beginning\_of\_the\_entire\_input;

}

figures[figures.size()-1]->verteces.push\_back(vertex);

}

std::cout << "You've added " << figure\_type\_string << " with this verteces: ";

figures[figures.size()-1]->coordinates();

} else if(command\_string == "ars") {

long double area(0);

for(long unsigned int i = 0; i< figures.size(); i++) area+=figures[i]->area();

std::cout << "the whole area is " << area;

} else {

unsigned int input\_figure\_number(0);

if((get\_value<unsigned int>(input\_figure\_number) != VALID\_INPUT) || (input\_figure\_number >= figures.size())) {

do ch=getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

if(ch == EOF) return 0;

else continue;

} else if(command\_string == "cen") {

auto center = figures[input\_figure\_number]->center();

std::cout << input\_figure\_number << " center is: x = " << center.first << ", y = " << center.second;

} else if(command\_string == "coor") {

std::cout << input\_figure\_number << " coordinates are ";

figures[input\_figure\_number]->coordinates();

} else if(command\_string == "area") {

std::cout << input\_figure\_number << " area is " << figures[input\_figure\_number]->area();

} else if(command\_string == "d") {

figures.erase(figures.begin()+input\_figure\_number);

std::cout << "You've deleted figure #" << input\_figure\_number;

}

}

do ch = getchar(); while((ch != '\n') && (ch != EOF));

std::cout << std::endl;

if(ch == EOF) return 0;

} while(true);

return 0;

}

Figure.cpp, Triangle.cpp, Square.cpp, Octagon.cpp:

#include <utility>

#ifndef \_\_FIGURE\_\_

#define \_\_FIGURE\_\_

class Figure {

public:

virtual std::pair<double, double> center() const noexcept = 0;

virtual void coordinates() const noexcept = 0;

virtual double area() const noexcept = 0;

std::vector<std::pair<double, double>> verteces;

};

#endif

#include <vector>

#include "Figure.cpp"

#include <utility>

#include <iostream>

#define NUM\_OF\_VERTECES\_OF\_TRIANGLE 3

class Triangle : public Figure {

public:

std::pair<double, double> center() const noexcept override {

std::pair<double, double> \_center(verteces[0]);

for(int i = 1; i < NUM\_OF\_VERTECES\_OF\_TRIANGLE; ++i) {

\_center.first+=verteces[i].first;

\_center.second+=verteces[i].second;

}

\_center.first/=NUM\_OF\_VERTECES\_OF\_TRIANGLE;

\_center.second/=NUM\_OF\_VERTECES\_OF\_TRIANGLE;

return \_center;

}

void coordinates() const noexcept override{

for(int i = 0; i < NUM\_OF\_VERTECES\_OF\_TRIANGLE; ++i)

std::cout << '(' << verteces[i].first << ", " << verteces[i].second << ')';

}

double area() const noexcept override{

double area(0);

for(int i = 0; i< NUM\_OF\_VERTECES\_OF\_TRIANGLE; ++i) {

area+= verteces[i].first \* verteces[(i+1)%NUM\_OF\_VERTECES\_OF\_TRIANGLE].second - \

verteces[(i+1)%NUM\_OF\_VERTECES\_OF\_TRIANGLE].first \* verteces[i].second;

}

area/=2;

return area<0?-area:area;

}

};

#include <vector>

#include "Figure.cpp"

#include <utility>

#include <iostream>

#define NUM\_OF\_VERTECES\_OF\_SQUARE 3

class Square : public Figure {

public:

std::pair<double, double> center() const noexcept override {

std::pair<double, double> \_center(verteces[0]);

for(int i = 1; i < NUM\_OF\_VERTECES\_OF\_SQUARE; ++i) {

\_center.first+=verteces[i].first;

\_center.second+=verteces[i].second;

}

\_center.first/=NUM\_OF\_VERTECES\_OF\_SQUARE;

\_center.second/=NUM\_OF\_VERTECES\_OF\_SQUARE;

return \_center;

}

void coordinates() const noexcept override{

for(int i = 0; i < NUM\_OF\_VERTECES\_OF\_SQUARE; ++i)

std::cout << '(' << verteces[i].first << ", " << verteces[i].second << ')';

}

double area() const noexcept override{

double area(0);

for(int i = 0; i< NUM\_OF\_VERTECES\_OF\_SQUARE; ++i) {

area+= verteces[i].first \* verteces[(i+1)%NUM\_OF\_VERTECES\_OF\_SQUARE].second - \

verteces[(i+1)%NUM\_OF\_VERTECES\_OF\_SQUARE].first \* verteces[i].second;

}

area/=2;

return area<0?-area:area;

}

};

#include <vector>

#include "Figure.cpp"

#include <utility>

#include <iostream>

#define NUM\_OF\_VERTECES\_OF\_OCTAGON 8

class Octagon : public Figure {

public:

std::pair<double, double> center() const noexcept override {

std::pair<double, double> \_center(verteces[0]);

for(int i = 1; i < NUM\_OF\_VERTECES\_OF\_OCTAGON; ++i) {

\_center.first+=verteces[i].first;

\_center.second+=verteces[i].second;

}

\_center.first/=NUM\_OF\_VERTECES\_OF\_OCTAGON;

\_center.second/=NUM\_OF\_VERTECES\_OF\_OCTAGON;

return \_center;

}

void coordinates() const noexcept override{

for(int i = 0; i < NUM\_OF\_VERTECES\_OF\_OCTAGON; ++i)

std::cout << '(' << verteces[i].first << ", " << verteces[i].second << ')';

}

double area() const noexcept override{

double area(0);

for(int i = 0; i< NUM\_OF\_VERTECES\_OF\_OCTAGON; ++i) {

area+= verteces[i].first \* verteces[(i+1)%NUM\_OF\_VERTECES\_OF\_OCTAGON].second - \

verteces[(i+1)%NUM\_OF\_VERTECES\_OF\_OCTAGON].first \* verteces[i].second;

}

area/=2;

return area<0?-area:area;

}

};

Read\_input.cpp:

#include <iostream>

#include <string>

#include <cstdio>

#include <set>

typedef enum {

END\_OF\_FILE,

VALID\_INPUT,

INVALID\_INPUT,

END\_OF\_LINE,

} read\_return\_t;

read\_return\_t get\_command(std::set<std::string>& valid\_commands, char\* command)

{

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

std::cin >> command;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

if(valid\_commands.count(command)) return VALID\_INPUT;

else return INVALID\_INPUT;

}

template <typename T>

read\_return\_t get\_value(T& d)

{

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

template <>

read\_return\_t get\_value (unsigned int& d) {

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

if(ch == '-') return INVALID\_INPUT;

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

template <>

read\_return\_t get\_value (unsigned long long& d) {

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

if(ch == '-') return INVALID\_INPUT;

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

6. Вывод:

Виртуальные функции являются полезным инструментом для хранения массива объектов наследников.

ЛИТЕРАТУРА

Б. Страуструп "Язык программирования С++", четвёртое издание, 2013.