**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: «Р**абота с иерархией объектов:**

**Наследование и полиморфизм»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 7307 | Ефимчик Н.В. Петров В.М. |  |
| Преподаватель | Колинько П.Г. |  |

Санкт-Петербург

2019

Оглавление

[Цель работы 3](#__RefHeading___Toc188_4150899979)

[Задание 3](#__RefHeading___Toc190_4150899979)

[Иерархия классов 3](#__RefHeading___Toc192_4150899979)

[Добавленный код 4](#__RefHeading___Toc194_4150899979)

[Вывод 7](#__RefHeading___Toc196_4150899979)

# **Цель работы**

Изучить наследование и полиморфизм на примере программы рисующей различные фигуры.

# **Задание**

Доработать модуль shape.cpp: добавить в коллекцию ещё одну фигуру — параллелограмм с крестом. Для фигуры определить подходящее место в иерархии классов и написать необходимые функции - члены. Разработанной фигурой нужно дополнить картинку в позициях 1 и 14 (галстук и шишак). Фигура в позиции 14 должна быть отражена вверх,а в 1ой вниз.

# **Иерархия классов**

Были добавлены следующие классы:

* 1. 1) cross
  2. 2) parallelogram
  3. 3) parallelogramWithCross

1. Класс cross виртуально наследуется от shape, чтобы избежать неопределенности в дальнейшем. Класс parallelogram наследуется от rectangle и reflectable. Класс
   1. parallelogramWithCross наследуется от parallelogram и cross. От cross защищенно, так как необходим доступ к защищенным членам класса.
2. Пришлось переопределить некоторые функций: draw() и функции отвечающие за возврат крайних точек фигуры(west() и тд). Это было сделано, поскольку они работали не так как должны.
3. Все функции члены являются доступными.

# **Добавленный код**

**class** cross : **virtual** shape {

**protected** :

point n;

point w;

**public**:

cross(point a, point b) : n(a),w(b){ }

point north() **const** override {

**return** point(n);

}

point south() **const** override {

**return** point(n.x, n.y - 2 \*( (n.y - w.y) )); *// (n.y - w.y ) == расстояние от n до с*

}

point east() **const** override {

**return** point(w.x + 2 \* (n.x - w.x ),w.y); *// (n.y - w.y ) == расстояние от w до с*

}

point west() **const** override {

**return** point(w);

}

point neast() **const** override {

**return** point(east().x,n.y);

}

point seast() **const** override {

**return** point(east().x,south().y);

}

point nwest() **const** override {

**return** point(w.x,n.y);

}

point swest() **const** override {

**return** point(w.x,south().y);

}

**void** draw() override {

put\_line(cross::north(),cross::south());

put\_line(cross::west(),cross::east());

}

**void** move(**int** a, **int** b) override {

n.x += a;

n.y += b;

w.x += a;

w.y += b;

}

**void** resize(**int** d) override {

n.y += (n.y - w.y) \* (d - 1);

w.x -= (n.x - w.x) \* (d - 1);

}

};

**class** parallelogram : **public** rectangle, reflectable{

**protected**:

**int** h;

**bool** sideX;

**bool** horizontal;

**public**:

parallelogram(point a, point b ,**int** h, **bool** sideX) : rectangle(a,b) , h(h), sideX(sideX) {

horizontal = **true**;

}

**public**:

**void** draw() override {

point newNW(nwest().x + horizontal \* sideX \* h,nwest().y + sideX \* !horizontal \* h);

point newNE(ne.x - horizontal \* !sideX \* h,ne.y - !horizontal \* !sideX \* h);

point newSW(sw.x + horizontal \* !sideX \* h,sw.y + !horizontal \* !sideX \* h);

point newSE(seast().x - horizontal \* sideX \* h,seast().y - sideX \* !horizontal \* h);

put\_line(newNW,newNE);

put\_line(newNE, newSE);

put\_line(newSE, newSW);

put\_line(newSW, newNW);

}

**void** resize(**int** d) override {

rectangle::resize(d);

h = h \* d;

}

**void** rotate\_left() override {

rectangle::rotate\_left();

horizontal = !horizontal;

sideX = !sideX;

}

**void** rotate\_right() override {

rectangle::rotate\_right();

horizontal = !horizontal;

sideX = !sideX;

}

**void** flip\_horisontally() override {

sideX = !sideX;

}

**void** flip\_vertically() override {

flip\_horisontally();

}

};

**class** parallelogramWithCross : **public** parallelogram , **private** cross { *//*

**public**:

parallelogramWithCross(point a, point b ,**int** h, **bool** sideX) : parallelogram(a, b, h,sideX),

cross(point((parallelogram::west().x + parallelogram::east().x)/2,parallelogram::north().y),point(sw.x + h / 2,parallelogram::west().y)) {

}

**void** draw() override {

parallelogram::draw();

cross::draw();

}

**public**:

point north() **const** override {

**return** parallelogram::north();

}

point south() **const** override {

**return** parallelogram::south();

}

point east() **const** override {

**return** parallelogram::east();

}

point west() **const** override {

**return** parallelogram::west();

}

point neast() **const** override {

**return** parallelogram::neast();

}

point seast() **const** override {

**return** parallelogram::seast();

}

point nwest() **const** override {

**return** parallelogram::nwest();

}

point swest() **const** override {

**return** parallelogram::swest();

}

**void** flip\_horisontally() override {

parallelogram::flip\_horisontally();

}

**void** move(**int** a, **int** b) override {

parallelogram::move(a, b);

cross::move(a,b);

}

**void** resize(**int** d) override {

parallelogram::resize(d);

crossCoordinateOverride();

}

**void** rotate\_left() override {

parallelogram::rotate\_left();

crossCoordinateOverride();

}

**void** rotate\_right() override {

parallelogram::rotate\_right();

crossCoordinateOverride();

}

**private**:

**void** crossCoordinateOverride() {

*// Если параллелограмм в "горизонтальном состоянии"*

**if** (horizontal) {

cross::n.x = (parallelogram::west().x + parallelogram::east().x) / 2;

cross::n.y = parallelogram::north().y;

cross::w = point(sw.x + h / 2,parallelogram::west().y);

} **else** { *// Если параллелограмм в "вертикальном состоянии"*

cross::w = parallelogram::west();

cross::n.x = parallelogram::north().x;

cross::n.y = parallelogram::north().y - h / 2 ;

}

}

};

# **Результаты работы программы**

# 

# **Вывод**

В ходе работы мы ознакомились с понятиями наследование и полиморфизм. Благодаря наследованию удалось сократить количество кода при написании новых фигур. Было применено виртуальное наследование, чтобы не создавать лишних экземпляров родительского класса, которые в дальнейшем могли вызвать проблемы. Благодаря полиморфизму функции способны обрабатывать каждую фигуры единообразно, что, опять же сокращает количество кода. Считаю нужным отметить, что очень удачно создан класс parallelogram. Используя 2 булевых параметра можно очень легко чертить самые различные параллелограммы, менять ориентацию, переворачивать.