**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**ОТЧЕТ**

**по лабораторной работе № 1**

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

**Тема: «Работа с иерархией объектов: наследование и полиморфизм»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 3311 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Сапронов К. Д.  Землякова С. А. |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Преподаватель |  | Манирагена Валенс |

Санкт-Петербург

2025

**Текст индивидуального задания (вариант 40)**

По заданию требовалось доработать shape.cpp, добавив новую фигуру — комбинацию параллелограмма и креста (paralellogram + cross). Эти фигуры должны быть размещены в позициях 1, 2, 3 и 14 на рисунке. Также необходимо было определить их место в иерархии классов, реализовать необходимые методы и добавить их в итоговое изображение.

Иерархия классов

В ходе работы были добавлены три новых класса:

1. cross — крест, унаследован от rotatable, может поворачиваться.

2. paralellogram — параллелограмм, унаследован от rotatable и reflectable, умеет поворачиваться и отражаться.

3. new\_shape — объединённая фигура, наследует свойства cross и paralellogram.

Для новых классов были переопределены:

1. draw() — отвечает за отрисовку.

2. move(int a, int b) — перемещает фигуру.

3. resize(double d) — изменяет размер.

4. rotate\_right() и rotate\_left() — поворачивает фигуру.

5. flip\_horisontally() и flip\_vertically() — отражает фигуру (только для paralellogram).

Запрещён конструктор копирования для new\_shape, чтобы избежать ошибок при дублировании. Внутренние данные скрыты через protected, чтобы предотвратить их изменение извне.

**Контрольные тесты**

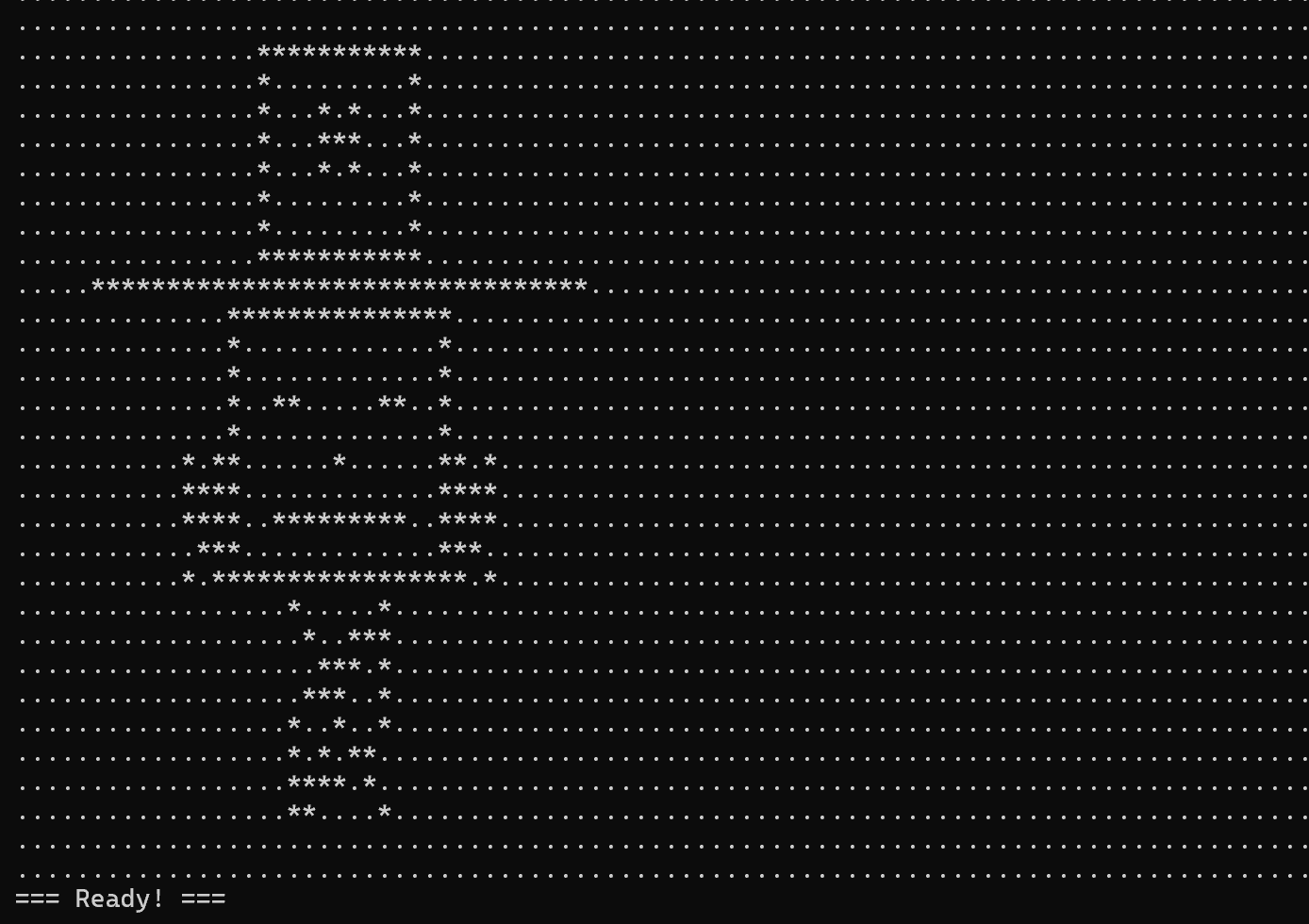
1. Начальный набор фигур.



2. Применение трансформаций (повороты, отражения, изменения размеров).



3. Итоговое изображение с новыми фигурами.



Выводы:

Временная сложность:

1. Отрисовка (draw()) — O(n), где n — число линий.
2. Поворот и отражение — O(1), так как меняется только состояние объекта.
3. Перемещение (move()) — O(1), так как обновляются только координаты.

**Контрольные вопросы**

*6. В чём смысл объявления функций в базовом классе как виртуальных?*  
Виртуальные функции позволяют использовать динамическое связывание. Это значит, что вызов метода будет зависеть не от типа указателя, а от фактического типа объекта. В коде это видно на примере shape::draw(), где в shape\_refresh() вызывается p->draw(), и благодаря виртуальности вызываются конкретные реализации draw() у line, rectangle, new\_shape и других наследников.

*7. Нужно ли объявлять виртуальной функцию в производном классе, если в базовом она уже объявлена таковой?*Нет, не нужно — если функция уже виртуальная в базовом классе, она остаётся виртуальной в производных, но её можно переопределить с override, как сделано в draw() у cross и paralellogram.

*8. Можно ли потребовать от компилятора проверить корректность* объявления виртуальной функции в производном классе и как это сделать?  
Да, можно. Для этого используется override в определении функции в производном классе. Если базовый класс не имеет такой виртуальной функции, компилятор выдаст ошибку. В коде, например, draw() в cross объявлена с override, что гарантирует соответствие сигнатуре базового класса.

*9. Можно ли запретить виртуальную функцию в классах-наследниках?*Да, можно. Для этого в производном классе надо объявить её с final. Например:

class Derived : public Base {

void draw() override final;

};

После этого дальнейшие наследники Derived уже не смогут переопределять draw().

*10. Что такое «чисто виртуальная функция»?*Это виртуальная функция, у которой нет реализации в базовом классе, и её необходимо определить в производных классах. Записывается так:

class shape {

virtual void draw() = 0; // Чисто виртуальная функция

};

В коде shape является абстрактным классом, потому что в нём есть несколько чисто виртуальных методов, например draw().

*11. Обязательно ли переопределять все функции-члены базового класса в производном классе?*Нет, только чисто виртуальные. Например, если класс shape имеет draw() как чисто виртуальную, то cross, paralellogram и new\_shape обязательно должны её переопределить. Если функция не чисто виртуальная, её можно оставить без изменений.

*12. Зачем может понадобиться создание набора (массива или списка) указателей на разные типы объектов в пределах некоторой иерархии?*  
Чтобы можно было работать с объектами разных классов через единый интерфейс. Например, в коде есть list<shape\*> shapes;, который хранит указатели на разные фигуры. Это позволяет вызвать shape\_refresh(), и для каждого объекта в списке корректно отработает его версия draw().

*13. Как запретить для объекта вызов конструктора по умолчанию?*Чтобы запретить для объекта вызов конструктора по умолчанию, достаточно объявить его = delete:

class Example {

Example() = delete;

};

В коде это не использовано, но можно применить для new\_shape, если не нужно создавать пустые экземпляры.

*14. Как запретить вызов конструктора для использования в качестве неявного преобразователя типа?*  
Использовать explicit:

class Example {

public:

explicit Example(int x);

};

Тогда Example obj = 5; вызовет ошибку, а Example obj(5); — нет. В коде это не применено, но можно использовать для фигур, если требуется явное создание объектов.