Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Информационная безопасность»

ОТЧЕТ

по лабораторной работе № 3

по учебной дисциплине «Алгоритмические языки»

на тему: «Изучение возможностей наследования классов»

Вариант 18

Выполнил:

Студент 1 курса, гр. ИУ8-24

Ожогин Михаил

2024 г.

Цель работы:

Овладение навыками разработки программ на языке Си++, использующих возможности наследования классов для решения различных задач.

Задачи работы:

- Изучить необходимые учебные материалы, посвященные наследованию классов в языке Си++
- Разработать программу на языке Си++ для решения заданного варианта задания
- Отладить программу
- Представить результаты работы программы
- Подготовить отчет по лабораторной работе

Условие задачи:

1. **Описание класса Rectangle:**
    - Элементы класса:
        1. Поля, определяющие длины сторон прямоугольника (статус доступа protected).
        2. Конструктор для инициализации полей.
        3. Функция для вычисления площади прямоугольника.
        4. Функция для печати полей и значения площади.
2. **Создать производный класс "прямоугольный параллелепипед" на основе базового класса "прямоугольник".**
    - Элементы класса:
        1. Дополнительное поле, задающее высоту параллелепипеда.
        2. Конструктор для инициализации полей.

3. Переопределенная функция для вычисления объема параллелепипеда (вместо площади) (внутри переопределенной функции должна вызываться функция из базового класса).
4. Переопределенная функция для печати полей и значения объема.

3. **Создать по одному объекту каждого из классов.**
4. **Показать вызов созданных функций.**
    - При переопределении функций обеспечить и продемонстрировать два варианта: статический полиморфизм и динамический полиморфизм.

Выполнение работы:

Заголовочный файл класса Rectangle – **rectangle.hpp**:

```
#ifndef RECTANGLE_HPP_
#define RECTANGLE_HPP_

#include <cstddef>
#include <iostream>

class Rectangle {
  public:
  Rectangle(const size_t& _l, const size_t& _w);

  virtual size_t area();
  virtual void print();
  virtual void print_subdata();

  protected:
  size_t _length = 0;
  size_t _width = 0;
};
```

```
Rectangle::Rectangle(const size_t& _l, const size_t& _w) : _length(_l),
_width(_w) {};

size_t Rectangle::area() {
  return _length * _width;
}

void Rectangle::print() {
  std::cout << "Length: " << _length << ", Width: " << _width;
}

void Rectangle::print_subdata() {
  std::cout << "Area of the base: " << this->area();
}

#endif // RECTANGLE_HPP_
```

Заголовочный файл класса Parallelepiped — **parallelepiped.hpp**:

```
#ifndef PARALLELEPIPED_HPP_
#define PARALLELEPIPED_HPP_

#include "rectangle.hpp"

class Parallelepiped : public Rectangle {
  public:
  Parallelepiped(const size_t& _l, const size_t& _w, const size_t& _h);

  size_t volume();
  void print() override;
  void print_subdata() override;

  protected:
  size_t _height = 0;
};
```

```
Parallelepiped::Parallelepiped(const size_t& _l, const size_t& _w, const
size_t& _h) : Rectangle(_l, _w), _height(_h) {};

size_t Parallelepiped::volume() {
  return Rectangle::area() * _height;
}

void Parallelepiped::print() {
  Rectangle::print();
  std::cout << ", Height: " << _height;
}

void Parallelepiped::print_subdata() {
  std::cout << "Area of the base: " << this->area();
  std::cout << ", Volume: " << this->volume();
}

#endif // PARALLELEPIPED_HPP_
```

Файл реализации – **main.cpp**:

```
#include "../include/parallelepiped.hpp"

int main() {
  Rectangle rec(105, 13);
  Rectangle* pointer = &rec;

  std::cout << "Example of correct method-usage through pointer" <<
std::endl;
  pointer->print();
  std::cout << std::endl;
  pointer->print_subdata();
  std::cout << std::endl;

  Parallelepiped pip(120, 27, 18);
  pointer = &pip;
```

```
  std::cout << std::endl << "Example of dynamic polymorphism: " <<
std::endl;

  pointer->print();

  std::cout << std::endl;


  std::cout << std::endl << "Example of static polymorphism: " <<
std::endl;

  pointer->print_subdata();

  std::cout << std::endl;

}
```

Вывод:

В ходе выполнения лабораторной работы были изучены возможности наследования и статусы доступа наследования классов в языке C++. В процессе работы были изучены способы переопределения методов, а также принципы статического и динамического полиморфизма. Реализованы функции, демонстрирующие как статический, так и динамический полиморфизм.