



**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» (ИУ)

**КАФЕДРА**

«Информационная безопасность» (ИУ8)

**Лабораторная работа № 3  
ПО КУРСУ  
«Алгоритмические языки»  
на тему «Наследование классов»**

**Студент**

**ИУ8-23**

(Группа)

**В. С. Ажгирей**

(И. О. Фамилия)

**Преподаватель:**

**М. В. Малахов**

(И.О. Фамилия)

## Введение

### Цели и задачи работы

Цель работы состоит в овладении навыками разработки программ на языке Си++, использующих возможности наследования классов для решения различных задач. Для достижения цели необходимо выполнить следующие задачи:

- изучить необходимые учебные материалы, посвященные наследованию классов в языке Си++;
- разработать программу на языке Си++ для решения заданного варианта задания;
- отладить программы;
- представить результаты работы программы;
- подготовить отчет по лабораторной работе.

### Условия для 1 варианта

Создать базовый класс «вектор на плоскости». Элементы класса: поля, задающие координаты точки (статус доступа *protected*), определяющей конец вектора (начало вектора находится в точке с координатами 0, 0); конструктор для инициализации полей; функция для вычисления длины вектора, функция для печати полей и длины вектора. Создать производный класс «вектор в трехмерном пространстве». Элементы класса: дополнительное поле, задающее дополнительную координату; конструктор для инициализации полей; переопределенная функция для вычисления длины вектора; переопределенная функция для печати полей и длины вектора. Создать по 1 объекту каждого из классов. Показать вызов созданных функций. При переопределении функций обеспечить и продемонстрировать два варианта: статический полиморфизм и динамический полиморфизм.

## Основная часть

Исходный текст программы (динамический полиморфизм):

Файл заголовка sources.hpp:

```
#pragma once
#include <iostream>
#include <math.h>

class VectorOnPlane
{
protected:
    int x, y;

public:
    VectorOnPlane();

    VectorOnPlane(int, int);

    virtual double length() const;

    virtual std::ostream& printDetails(std::ostream&) const;
};

class VectorInSpace : public VectorOnPlane
{
protected:
    int z;

public:
    VectorInSpace();

    VectorInSpace(int, int, int);

    double length() const override;

    std::ostream& printDetails(std::ostream&) const override;
};

std::ostream& operator<<(std::ostream&, const VectorOnPlane&);
```

Файл описания sources.cpp:

```
#include "sources.hpp"

VectorOnPlane::VectorOnPlane() : x(0), y(0) { std::cout << "VectorOnPlane
constructor" << std::endl; };

VectorOnPlane::VectorOnPlane(int x, int y) : x(x), y(y) {};

double VectorOnPlane::length() const
{
    return sqrt(pow(x, 2) + pow(y, 2));
}
```

```

std::ostream& VectorOnPlane::printDetails(std::ostream& output_stream) const
{
    output_stream << "vector coordinates: (" << x << ", " << y << ")" << std::endl;
    output_stream << "vector length: " << length() << std::endl;
    output_stream << "#####" << std::endl;

    return output_stream;
}

VectorInSpace::VectorInSpace() : VectorOnPlane(), z(0) { std::cout << "VectorInSpace
constructor" << std::endl; };

VectorInSpace::VectorInSpace(int x, int y, int z) : VectorOnPlane(x, y), z(z) {};

double VectorInSpace::length() const
{
    return sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
}

std::ostream& VectorInSpace::printDetails(std::ostream& output_stream) const
{
    output_stream << "vector coordinates: (" << x << ", " << y << ", " << z << ")"
<< std::endl;
    output_stream << "vector length: " << length() << std::endl;
    output_stream << "#####" << std::endl;

    return output_stream;
}

std::ostream& operator<<(std::ostream& output_stream, const VectorOnPlane& vector)
{
    return vector.printDetails(output_stream);
}

```

Исполняемый файл main.cpp:

```

#include "sources.hpp"

int main()
{
    VectorOnPlane xy;
    VectorInSpace xyz(1, 2, 3);

    std::cout << xy;
    std::cout << xyz;

    VectorOnPlane* vec1 = new VectorOnPlane(1, 3);
    VectorOnPlane* vec2 = new VectorInSpace(0, 7, 8);

    std::cout << vec1->length() << std::endl;
    std::cout << vec2->length();

    return 0;
}

```

## Исходный текст программы (динамический полиморфизм):

Файл заголовка sources.hpp:

```
#pragma once
#include <iostream>
#include <math.h>

class VectorOnPlane
{
protected:
    int x, y;

public:
    VectorOnPlane();

    VectorOnPlane(int, int);

    double length() const;

    std::ostream& printDetails(std::ostream&) const;
};

class VectorInSpace : public VectorOnPlane
{
protected:
    int z;

public:
    VectorInSpace();

    VectorInSpace(int, int, int);

    double length() const;

    std::ostream& printDetails(std::ostream&) const;
};

std::ostream& operator<<(std::ostream&, const VectorOnPlane&);
```

Файл описания sources.cpp:

```
#include "sources.hpp"

VectorOnPlane::VectorOnPlane() : x(0), y(0) { std::cout << "VectorOnPlane
constructor" << std::endl; };

VectorOnPlane::VectorOnPlane(int x, int y) : x(x), y(y) {};

double VectorOnPlane::length() const
{
    return sqrt(pow(x, 2) + pow(y, 2));
}

std::ostream& VectorOnPlane::printDetails(std::ostream& output_stream) const
{
    output_stream << "vector coordinates: (" << x << ", " << y << ")" << std::endl;
    output_stream << "vector length: " << length() << std::endl;
    output_stream << "#####" << std::endl;

    return output_stream;
}
```

```

VectorInSpace::VectorInSpace() : VectorOnPlane(), z(0) { std::cout << "VectorInSpace
constructor" << std::endl; };

VectorInSpace::VectorInSpace(int x, int y, int z) : VectorOnPlane(x, y), z(z) {};

double VectorInSpace::length() const
{
    return sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
}

std::ostream& VectorInSpace::printDetails(std::ostream& output_stream) const
{
    output_stream << "vector coordinates: (" << x << ", " << y << ", " << z << ")"
<< std::endl;
    output_stream << "vector length: " << length() << std::endl;
    output_stream << "#####" << std::endl;

    return output_stream;
}

std::ostream& operator<<(std::ostream& output_stream, const VectorOnPlane& vector)
{
    return vector.printDetails(output_stream);
}

```

Исполняемый файл main.cpp:

```

#include "sources.hpp"

int main()
{
    VectorOnPlane xy;
    VectorInSpace xyz(1, 2, 3);

    std::cout << xy;
    std::cout << xyz;

    VectorOnPlane* vec1 = new VectorOnPlane(1, 3);
    VectorOnPlane* vec2 = new VectorInSpace(0, 7, 8);

    std::cout << vec1->length() << std::endl;
    std::cout << vec2->length();

    return 0;
}

```

## Снимки выполнения работы программы

```
VectorOnPlane xy;  
VectorInSpace xyz(1, 2, 3);
```

Рисунок 1 – Входные данные

```
VectorOnPlane constructor  
vector coordinates: (0, 0)  
vector length: 0  
#####  
vector coordinates: (1, 2, 3)  
vector length: 3.74166  
#####
```

Рисунок 2 – Метод печати (динамический полиморфизм)

```
VectorOnPlane constructor  
vector coordinates: (0, 0)  
vector length: 0  
#####  
vector coordinates: (1, 2)  
vector length: 2.23607  
#####
```

Рисунок 3 – Метод печати (статический полиморфизм)

```
VectorOnPlane* vec1 = new VectorOnPlane(1, 3);  
VectorOnPlane* vec2 = new VectorInSpace(0, 7, 8);
```

Рисунок 4 – Входные данные

```
3.16228  
10.6301
```

Рисунок 5 – Метод вычисления длины (динамический полиморфизм)

```
3.16228  
7
```

Рисунок 6 – Метод вычисления длин (статический полиморфизм)

## Заключение

Задачи лабораторной работы были решены, результаты проверены. Изучено на практике наследование классов в C++. Также изучены статический и динамический полиморфизм.