

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **технологий** | **Кафедра**  **информационных систем** |

**Основная образовательная программа 09.03.02  
«Информационные системы и технологии»**

**Отчет по дисциплине «Объектно-ориентированное программирование»**

**по лабораторной работе № 3**

**по теме: «Множественное наследование в C++»**

|  |  |
| --- | --- |
| Студент группы ИДБ-23-08 | Прохоров А.А. |
|  |  |
| Старший преподаватель | Варварюк А.В. |

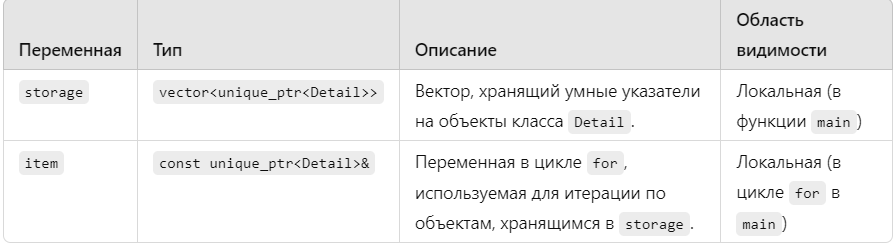
# Задание 1

**Описание**

Задать два иерархически связанных полиморфных класса: деталь и сборка (один из них базовый). Все конструкторы находятся в зоне protected (чтобы запретить явное создание). В main() следует создать несколько экземпляров указанных типов, помещая их в хранилище (vector), используя шаблонную дружественную функцию.

Дополнить программу всем необходимым для ее правильной работы.

**Использованные переменные**

****

**Программа**

#include <iostream>

#include <vector>

#include <memory>

#include <string>

using namespace std;

class Detail

{

protected:

Detail() {

cout << "Создание Детали\n";

}

public:

virtual void gg() = 0;

virtual ~Detail() = default; // Виртуальный деструктор

};

class Assembly : public Detail

{

protected:

Assembly() {

cout << "Создание сборки\n";

}

public:

void gg() override {

cout << "Это сборка";

}

static unique\_ptr<Detail> create() {

return std::unique\_ptr<Detail>(new Assembly());

}

};

int main() {

setlocale(LC\_ALL, "Russian");

vector<unique\_ptr<Detail>> storage;

storage.emplace\_back(Assembly::create());

for (const auto& item : storage) {

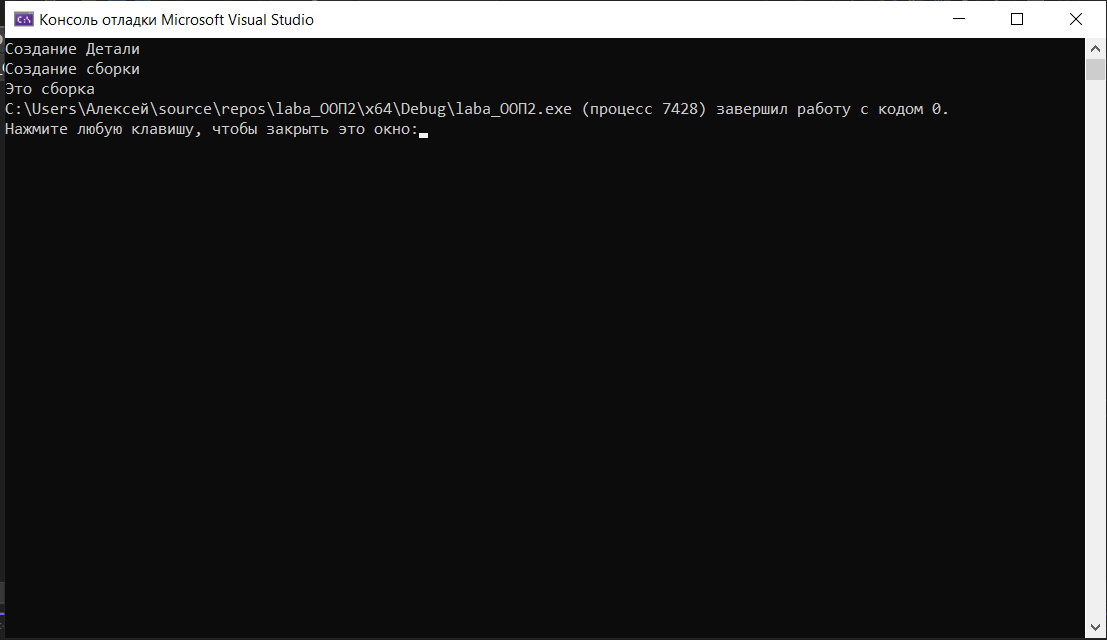
item->gg();

}

return 0;

}

**Примеры работы программы**



# Задание 2

**Описание**

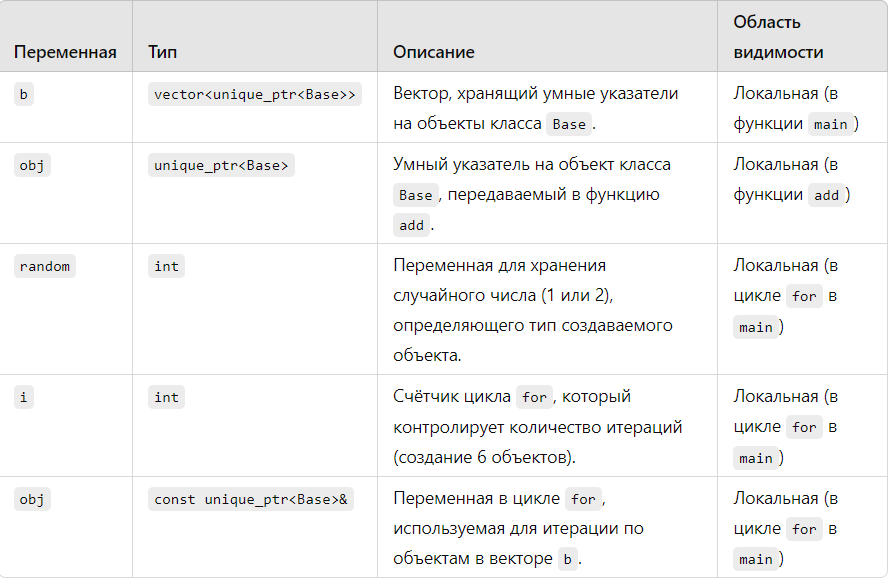
Задать два иерархически связанных полиморфных класса: Base и Derived.

В главной функции программы последовательно выполняется два действия. В цикле случайно (random) создаются несколько экземпляров любого из заданных типов (например, по четному случайному числу создается Base, а по нечетному - Derived). После создания объектов, их копии передаются в функцию add, которая помещает их в общее хранилище.

Затем в main необходимо показать для каждого из имеющихся в хранилище объектов разницу в поведении их типа, доставая их по очереди из хранилища.

В заключении, освободить все ресурсы.

**Использованные переменные**



**Программа**

#include <iostream>

#include <vector>

#include <memory>

#include <string>

#include <cstdlib>

using namespace std;

class Base

{

public:

virtual ~Base() = default;

virtual void gg() = 0;

};

class Derived1 : public Base

{

public:

void gg() override {

cout << "Это Dirived1\n";

}

};

class Derived2 : public Base

{

public:

void gg() override {

cout << "Это Derived2\n";

}

};

void add(vector<unique\_ptr<Base>>& b, unique\_ptr<Base> obj) {

b.push\_back(move(obj));

}

int main() {

setlocale(LC\_ALL, "ru");

srand(static\_cast<unsigned int>(time(nullptr)));

vector<unique\_ptr<Base>> b;

for (int i = 0; i < 6; i++) {

int random = 1 + rand() % 2;

if (random == 1) {

add(b, make\_unique<Derived1>());

}

else {

add(b, make\_unique<Derived2>());

}

}

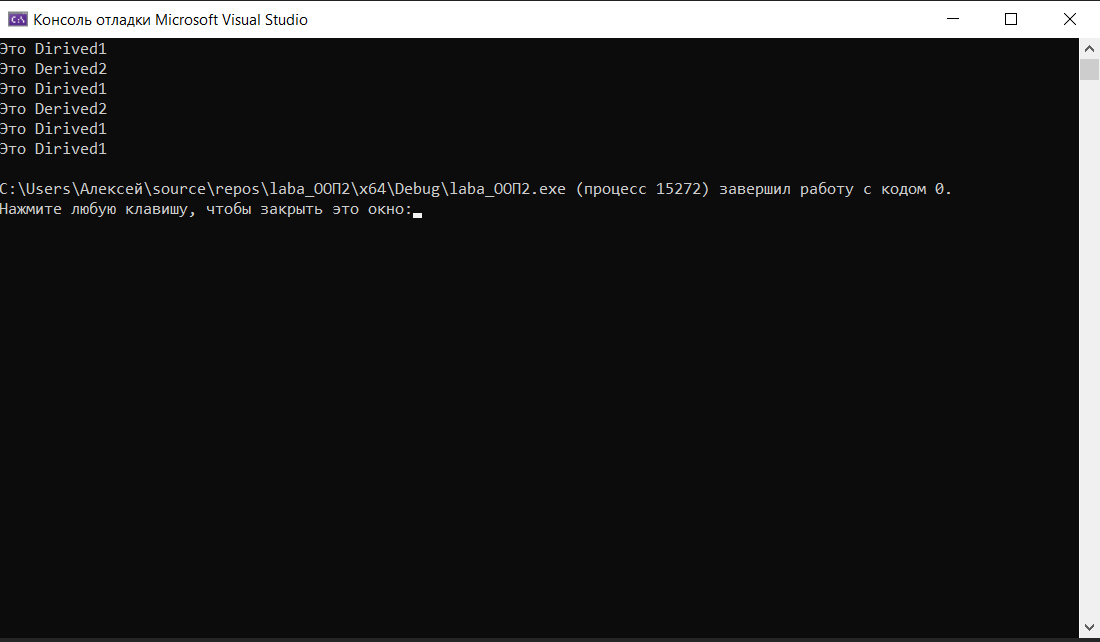
for (const auto& obj : b) {

obj->gg();

}

}

**Примеры работы программы**



**Вывод**

В данной лабораторной работе были реализованы две задачи, каждая из которых направлена на отработку принципов ООП, включая полиморфизм, работу с указателями и шаблонами.