**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

14 ноября 2024 г**. Вариант № 24**

Факультет Кибербезопасности

Кафедра Инфокоммуникационных систем

# Отчет по лабораторной работе № 4

«Конструирование классов на основе принципа наследования»

**по дисциплине «Технологии и методы программирования»**

Выполнили студенты гр. ИКБ-32  
Кушалиев Карим Искандерович Цыплаков Никита Андреевич

Принял ст.преп.

Воронцова И. О.

Цель лабораторной работы

Изучить механизм открытого (public) наследования в C++, познакомиться с понятием «виртуальная функция», освоить технологию конструирования и способы документирования программы, включающей в себя классы­наследники, изучить возможности инструментальных сред разработки по автоматической генерации кода.

Задание

1. В соответствии с вариантом задания разработать базовый класс. В базовый класс следует включить свойства и методы, общие для заданных классов­наследников. Базовый класс должен включать в себя не менее двух свойств и двух методов, один из которых –

виртуальная функция.

2. Разработать классы, производные от базового класса (наследники). Классы­наследники

должны наследовать от базового класса хотя бы одно свойство, а также должны иметь хотя бы одно собственное свойство. В классы­наследники должны быть включены следующие

методы:

a.Метод, наследуемый от базового класса без переопределения.

b. Виртуальная функция базового класса, переопределённая в производном классе.

c. Собственные методы производного класса. В состав производного класса должен быть включён хотя бы один метод, изменяющий какое­либо свойство класса.

3. Разработать программу, выполняющую следующие действия:

a. Создание нескольких объектов на основе классов–наследников.

b. Объединение объектов в массив (массив указателей на базовый класс).

c. Отображение значений свойств объектов на экране в цикле.

d. Изменение свойств объектов по номеру элемента массива.

e. Вычисление заданного параметра.

f. Выход из программы.

4. Объявление и реализацию каждого класса поместить в отдельный модуль.

5. Действия над объектами(просмотр, изменение, вычисление параметра) должны быть доступны через меню; последовательность выполнения действий – произвольная, в цикле

Листинг программы

main.cpp

#include "include/Peripheral.h"

#include "include/Monitor.h"

#include "include/Printer.h"

#include <iostream>

#include <limits>

#include <array>

#include <span>

void print\_periferals**(**std**::**span**<**Peripheral **\*>** peripherals**);**

void get\_average\_lifespan**(**std**::**span**<**Peripheral **\*>** peripherals**);**

int main**()**

**{**

Monitor monitor1**(**"Samsung"**,** "S24C366EAU"**,** 22000**,** 5**,** "2560x1440"**);**

Monitor monitor2**(**"LG"**,** "LS27C330GAUXEN"**,** 15999.90**,** 7**,** "1920x1080"**);**

Printer printer1**(**"HP"**,** "LaserJet CP5225dn"**,** 74799**,** 6**,** 100**);**

Printer printer2**(**"Epson"**,** "L8050"**,** 45799**,** 5**,** 150**);**

std**::**array**<**Peripheral **\*,** 4**>** peripherals **=** **{&**monitor1**,** **&**monitor2**,** **&**printer1**,** **&**printer2**};**

int choice**;**

**do**

**{**

std**::**cout **<<** "Выберите действие: \n"**;**

std**::**cout **<<** "0. Выход\n"**;**

std**::**cout **<<** "1. Отображение объектов\n"**;**

std**::**cout **<<** "2. Редактирование объектов\n"**;**

std**::**cout **<<** "3. Найти среднюю продолжительность жизни периферии\n"**;**

std**::**cout **<<** "Ваш выбор: "**;**

std**::**cin**.**ignore**(**std**::**numeric\_limits**<**std**::**streamsize**>::**max**(),** '\n'**);** // Очистка буфера ввода

std**::**cin **>>** choice**;**

std**::**cout **<<** "\n"**;**

**switch** **(**choice**)**

**{**

**case** 1**:**

print\_periferals**(**peripherals**);**

**break;**

**case** 2**:**

**{**

int index**;**

std**::**cout **<<** "Введите номер периферии для редактирования (1-" **<<** peripherals**.**size**()** **<<** "): "**;**

std**::**cin **>>** index**;**

**if** **(**index **>** 0 **&&** index **<=** peripherals**.**size**())**

**{**

peripherals**[**index **-** 1**]->**edit\_fields**();**

**}**

**else**

**{**

std**::**cout **<<** "Некорректный номер периферии." **<<** std**::**endl**;**

**}**

**break;**

**}**

**case** 3**:**

get\_average\_lifespan**(**peripherals**);**

**break;**

**default:**

std**::**cout **<<** "Некорректный выбор" **<<** std**::**endl**;**

**}**

**}** **while** **(**choice **!=** 0**);**

**return** 0**;**

**}**

void print\_periferals**(**std**::**span**<**Peripheral **\*>** peripherals**)**

**{**

**for** **(**Peripheral **\***peripheral **:** peripherals**)**

**{**

peripheral**->**print\_info**();**

**}**

**}**

void get\_average\_lifespan**(**std**::**span**<**Peripheral **\*>** peripherals**)**

**{**

int sum\_lifespan **=** 0**;**

**for** **(**Peripheral **\***peripheral **:** peripherals**)**

**{**

sum\_lifespan **+=** peripheral**->**get\_lifespan**();**

**}**

std**::**cout **<<** "Средний срок службы всех периферийных устройств: " **<<** **(**float**)**sum\_lifespan **/** peripherals**.**size**()** **<<** "\n\n"**;**

**}**

include/peripheral.h

#pragma once

#include <string>

#include <ctime>

class Peripheral

**{**

public**:**

Peripheral**(**std**::**string brand**,** std**::**string model**,** float price**,** int lifespan**);**

std**::**string get\_brand**();**

std**::**string get\_model**();**

int get\_lifespan**();**

float get\_price**();**

void set\_price**(**float price**);**

virtual void print\_info**();**

virtual void edit\_fields**();**

protected**:**

std**::**string brand**;**

std**::**string model**;**

int lifespan**;**

float price**;**

**};**

include/monitor.h

#pragma once

#include "Peripheral.h"

#include <string>

class Monitor **:** public Peripheral

**{**

public**:**

Monitor**(**std**::**string brand**,** std**::**string model**,** float price**,** int lifespan**,**

std**::**string resolution**);**

void print\_info**()** override**;**

void edit\_fields**()** override**;**

std**::**string get\_resolution**();**

protected**:**

std**::**string resolution**;**

**};**

include/printer.h

#pragma once

#include "Peripheral.h"

#include <string>

class Printer **:** public Peripheral

**{**

public**:**

Printer**(**std**::**string brand**,** std**::**string model**,** float price**,** int lifespan**,**

int paper\_capacity**);**

void print\_info**()** override**;**

void edit\_fields**()** override**;**

int get\_paper\_capacity**();**

protected**:**

int paper\_capacity**;**

**};**

classes/peripheral.cpp

#include "../include/Peripheral.h"

#include <iostream>

#include <string>

#include <limits>

Peripheral**::**Peripheral**(**std**::**string brand**,** std**::**string model**,** float price**,** int lifespan**)**

**{**

**this->**brand **=** brand**;**

**this->**model **=** model**;**

**this->**price **=** price**;**

**this->**lifespan **=** lifespan**;**

**};**

std**::**string Peripheral**::**get\_brand**()**

**{**

**return** **this->**brand**;**

**};**

std**::**string Peripheral**::**get\_model**()**

**{**

**return** **this->**model**;**

**};**

int Peripheral**::**get\_lifespan**()**

**{**

**return** **this->**lifespan**;**

**};**

float Peripheral**::**get\_price**()**

**{**

**return** **this->**price**;**

**};**

void Peripheral**::**set\_price**(**float price**)**

**{**

**if** **(**price **<** 0**)**

**{**

std**::**cout **<<** "Цена не может быть отрицательной!" **<<** std**::**endl**;**

**return;**

**}**

**this->**price **=** price**;**

**};**

void Peripheral**::**print\_info**()**

**{**

std**::**cout **<<** "1. Бренд: " **<<** **this->**brand **<<** std**::**endl**;**

std**::**cout **<<** "2. Модель: " **<<** **this->**model **<<** std**::**endl**;**

std**::**cout **<<** "3. Цена: " **<<** **this->**price **<<** std**::**endl**;**

std**::**cout **<<** "4. Срок службы: " **<<** **this->**lifespan **<<** std**::**endl**;**

**};**

void Peripheral**::**edit\_fields**()**

**{**

std**::**string field\_name**;**

std**::**cout **<<** "Доступные поля:\n"**;**

**this->**print\_info**();**

std**::**cout **<<** "Введите название поля или цифру для редактирования: "**;**

std**::**getline**(**std**::**cin**.**ignore**(**std**::**numeric\_limits**<**std**::**streamsize**>::**max**(),** '\n'**),** field\_name**);**

**if** **(**field\_name**.**compare**(**"Бренд"**)** **==** 0 **||** field\_name**.**compare**(**"1"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новый бренд: "**;**

std**::**getline**(**std**::**cin**,** **this->**brand**);**

**}**

**else** **if** **(**field\_name**.**compare**(**"Модель"**)** **==** 0 **||** field\_name**.**compare**(**"2"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новую модель: "**;**

std**::**getline**(**std**::**cin**,** **this->**model**);**

**}**

**else** **if** **(**field\_name**.**compare**(**"Цена"**)** **==** 0 **||** field\_name**.**compare**(**"3"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новую цену: "**;**

std**::**cin **>>** **this->**price**;**

**}**

**else** **if** **(**field\_name**.**compare**(**"Срок службы"**)** **==** 0 **||** field\_name**.**compare**(**"4"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новый срок службы: "**;**

std**::**cin **>>** **this->**lifespan**;**

**}**

**else**

**{**

std**::**cout **<<** "Такого поля нет!" **<<** std**::**endl**;**

**}**

**}**

classes/monitor.cpp

#include "../include/Monitor.h"

#include <iostream>

#include <string>

#include <limits>

Monitor**::**Monitor**(**std**::**string brand**,** std**::**string model**,** float price**,** int lifespan**,**

std**::**string resolution**)**

**:** Peripheral**(**brand**,** model**,** price**,** lifespan**)**

**{**

**this->**resolution **=** resolution**;**

**}**

std**::**string Monitor**::**get\_resolution**()**

**{**

**return** resolution**;**

**}**

void Monitor**::**print\_info**()**

**{**

Peripheral**::**print\_info**();**

std**::**cout **<<** "5. Разрешение: " **<<** **this->**get\_resolution**()** **<<** std**::**endl**;**

std**::**cout **<<** "\n"**;**

**}**

void Monitor**::**edit\_fields**()**

**{**

std**::**string field\_name**;**

std**::**cout **<<** "Доступные поля:\n"**;**

**this->**print\_info**();**

std**::**cout **<<** "Введите название поля или цифру для редактирования: "**;**

std**::**getline**(**std**::**cin**.**ignore**(**std**::**numeric\_limits**<**std**::**streamsize**>::**max**(),** '\n'**),** field\_name**);**

**if** **(**field\_name**.**compare**(**"Бренд"**)** **==** 0 **||** field\_name**.**compare**(**"1"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новый бренд: "**;**

std**::**getline**(**std**::**cin**,** **this->**brand**);**

**}**

**else** **if** **(**field\_name**.**compare**(**"Модель"**)** **==** 0 **||** field\_name**.**compare**(**"2"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новую модель: "**;**

std**::**getline**(**std**::**cin**,** **this->**model**);**

**}**

**else** **if** **(**field\_name**.**compare**(**"Цена"**)** **==** 0 **||** field\_name**.**compare**(**"3"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новую цену: "**;**

std**::**cin **>>** **this->**price**;**

**}**

**else** **if** **(**field\_name**.**compare**(**"Срок службы"**)** **==** 0 **||** field\_name**.**compare**(**"4"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новый срок службы: "**;**

std**::**cin **>>** **this->**lifespan**;**

**}**

**else** **if** **(**field\_name**.**compare**(**"Разрешение"**)** **==** 0 **||** field\_name**.**compare**(**"5"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новое разрешение: "**;**

std**::**cin **>>** **this->**lifespan**;**

**}**

**else**

**{**

std**::**cout **<<** "Такого поля нет!" **<<** std**::**endl**;**

**}**

**}**

classes/printer.cpp

#include "../include/Printer.h"

#include <iostream>

#include <limits>

Printer**::**Printer**(**std**::**string brand**,** std**::**string model**,** float price**,** int lifespan**,**

int paper\_capacity**)**

**:** Peripheral**(**brand**,** model**,** price**,** lifespan**)**

**{**

**this->**paper\_capacity **=** paper\_capacity**;**

**}**

int Printer**::**get\_paper\_capacity**()**

**{**

**return** **this->**paper\_capacity**;**

**}**

void Printer**::**print\_info**()**

**{**

Peripheral**::**print\_info**();**

std**::**cout **<<** "Максимальное кол-во бумаги: " **<<** **this->**paper\_capacity **<<** std**::**endl**;**

std**::**cout **<<** "\n"**;**

**}**

void Printer**::**edit\_fields**()**

**{**

std**::**string field\_name**;**

std**::**cout **<<** "Доступные поля:\n"**;**

**this->**print\_info**();**

std**::**cout **<<** "Введите название поля или цифрудля редактирования: "**;**

std**::**getline**(**std**::**cin**.**ignore**(**std**::**numeric\_limits**<**std**::**streamsize**>::**max**(),** '\n'**),** field\_name**);**

**if** **(**field\_name**.**compare**(**"Бренд"**)** **==** 0 **||** field\_name**.**compare**(**"1"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новый бренд: "**;**

std**::**getline**(**std**::**cin**,** **this->**brand**);**

**}**

**else** **if** **(**field\_name**.**compare**(**"Модель"**)** **==** 0 **||** field\_name**.**compare**(**"2"**)** **==** 0**)**

**{**

std**::**cout **<<** "Введите новую модель: "**;**

std::getline(std::cin, this->model);

}

else if (field\_name.compare("Цена") == 0 || field\_name.compare("3") == 0)

{

std::cout << "Введите новую цену: ";

std::cin >> this->price;

}

else if (field\_name.compare("Срок службы") == 0 || field\_name.compare("4") == 0)

{

std::cout << "Введите новый срок службы: ";

std::cin >> this->lifespan;

}

else if (field\_name.compare("Максимальное кол-во бумаги") == 0 || field\_name.compare("5") == 0)

{

std::cout << "Введите новое максимальное кол-во бумаги: ";

std::cin >> this->paper\_capacity;

}

else

{

std::cout << "\nТакого поля нет!" << std::endl;

}

}

Диаграмма классов

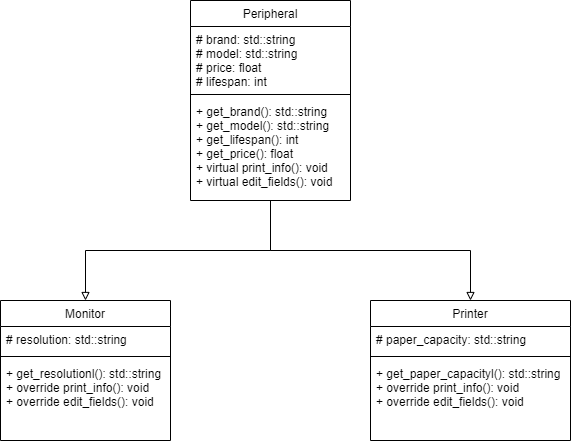
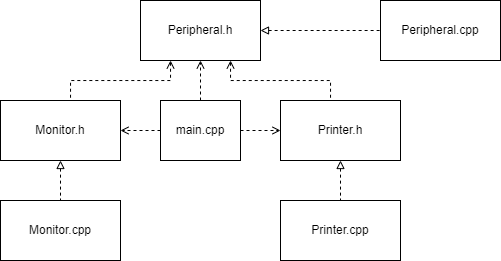


Диаграмма компонентов



Вывод

Была освоена работа с заголовками и классами, были изучены модификаторы *public*, *private*, *protected*, а также была проведена работа с виртуальными функциями. Цель достигнута. Задачи выполнены в полном объеме.