**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №7

«Виртуальные функции и классы»

| Выполнил: |  | Проверил: |
| --- | --- | --- |
| студент группы ИУ5-24б |  | преподаватель каф. ИУ5 |
| Чернев Н.А. |  |  |
| Подпись и дата: 29.05.2024 |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

## Задание

## Цель работы

Целью лабораторной работы изучение механизмов динамического связывания. Студенты изучают понятия абстрактного класса, виртуальных функций и виртуальных классов. Они осваивают на практике по индивидуальным заданиям использование этих прием при программировании в среде С++.

## Задание

В соответствии с [указаниями по выполнению работы](https://iu5edu.ru/wiki/cpp2/docs/labs/lab7/Instructions/) реализовать классы, предусмотренные [вариантом](https://iu5edu.ru/wiki/cpp2/docs/labs/lab7/TaskVariants/).

# Указания по выполнению работы

1. Для определения иерархии классов связать отношением наследования классы, приведенные в приложении (для [заданного варианта](https://iu5edu.ru/wiki/cpp2/docs/labs/lab7/TaskVariants/)). Из перечисленных классов выбрать один, который будет стоять во главе иерархии. Это абстрактный класс.
2. Определить в классах все необходимые конструкторы и деструкторы. Не забыть в абстрактном классе определить виртуальный деструктор.
3. Компонентные данные класса специфицировать как protected.
4. Пример определения статических компонентов:

static person\* begin: // указатель на начало списка

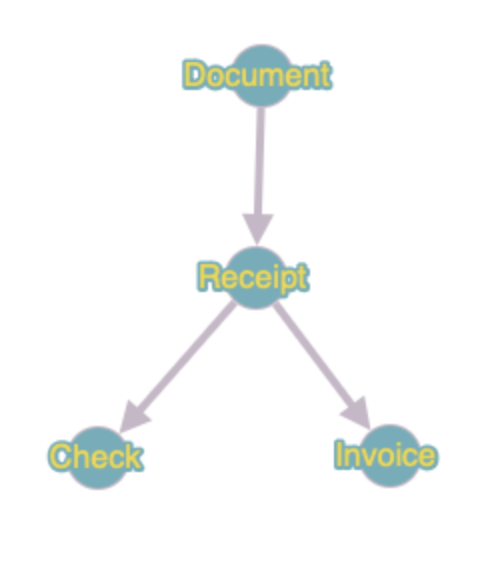
static void print(void); // просмотр списка

1. Статические компоненты класса необходимо определить в абстрактном классе.
2. Статическую компоненту - данное инициализировать вне определения класса, в глобальной области. Также в конце выполнения программы необходимо освободить память, выделенную на статические компоненты.
3. Для добавления объекта в список предусмотреть метод класса, т.е. объект сам добавляет себя в список (см. статические компоненты абстрактного класса). Например, a.Add() — объект адобавляет себя в список.  
   Включение объекта в список можно выполнять при создании объекта. т.е. поместить операторы включения в конструктор. В случае иерархии классов, включение объекта в список должен выполнять только конструктор базового класса. Вы должны продемонстрировать оба этих способа.  
   Необходимо предусмотреть изменения размера динамического массива, который выполняет роль списка.
4. Список просматривать путем вызова виртуального метода Show каждого объекта.
5. Статический метод просмотра списка вызывать не через объект, а через класс.
6. Определение классов, их реализацию, демонстрационную программу поместить в отдельные файлы. Данные файлы должны быть упакованы в отдельную статическую статическую библиотеку.

**Разработка алгоритма**

Библиотека docs:

* Абстрактный класс Document
  + Поля
    - int number; - номер документа
    - char \*name; - название документа
    - static Document \*\*docs; - массив указателей на документы
    - static int size; - кол-во элементов в массиве
    - static int capacity; - вместимость массива
  + Методы и функции:
    - Document(); - конструктор без параметров, инициализирует поля нулевыми значениями
    - Document(int \_number, const char \*\_name) - конструктор с параметрами, инициализирует поля переданными значениями
    - Document(const Document &doc) - конструктор копирования, инициализирует поля создаваемого объекта полями передаваемого
    - virtual ~Document() - виртуальный деструктор, удаляет динамически выделенную память
    - virtual void add() = 0; - объявление виртуального метода добавления документа в массив
    - virtual void show() = 0; - объявление виртуального метода вывода на экран информации о документе
    - Document& operator = (const Document &doc) - перегрузка оператора = для Document = Document
    - static void print() - вывод массива на экран
* Класс Receipt, наследник  класса Document (спец. доступа public)
  + Поля
    - int cost; - стоимость товара/услуги
    - char \*product\_name; - наименование товара/услуги
  + Методы
    - Receipt() - конструктор без параметров, инициализирует поля нулевыми значениями
    - Receipt(int \_number, const char \*\_name, int \_cost, const char \*\_product\_name) - конструктор с параметрами, инициализирует поля переданными значениями
    - Receipt(const Receipt &rec) - конструктор копирования, инициализирует поля создаваемого объекта полями передаваемого
    - Receipt& operator = (const Receipt &rec) - перегрузка оператора = для Receipt = Receipt
    - void add() - добавляет объект в массив документов
    - void show() - выводит на экран информацию о квитанции
    - ~Receipt() - деструктор, удаляет динамически выделенную память
* Класс Check, наследник класса Receipt (спец. доступа public)
  + Поля
    - int nds\_percent - процент НДС
  + Методы
    - Check() - конструктор без параметров, инициализирует поля нулевыми значениями
    - Check(int \_number, const char \*\_name, int \_cost, const char \*\_product\_name, int \_nds) - конструктор с параметрами, инициализирует поля переданными значениями
    - Check(const Check &ch) - ​​конструктор копирования, инициализирует поля создаваемого объекта полями передаваемого
    - Check& operator = (const Check &ch) - перегрузка оператора = для Check = Check
    - void show() - выводит на экран информацию о чеке
* Класс Invoice, наследник класса Receipt (спец. доступа public)
  + Поля
    - int delivery\_cost - стоимость доставки
  + Методы
    - Invoice() - конструктор без параметров, инициализирует поля нулевыми значениями
    - Invoice(int \_number, const char \*\_name, int \_cost, const char \*\_product\_name, int \_delivery\_cost) - конструктор с параметрами, инициализирует поля переданными значениями
    - Invoice(const Invoice &inv) - ​​конструктор копирования, инициализирует поля создаваемого объекта полями передаваемого
    - void show() - выводит на экран информацию о накладной
    - Invoice& operator = (const Invoice &inv) - перегрузка оператора = для Invoice = Invoice

**Иерархия классов**

**Текст программы**

Библиотека docs

docs.h

#pragma once

#include <iostream>

class Document{

protected:

int number;

char \*name;

public:

static Document \*\**docs*;

static int *size*;

static int *capacity*;

Document();

Document(int \_number, const char \*\_name);

Document(const Document &doc);

virtual ~Document() { delete[] name; }

virtual void add() = 0;

virtual void show() = 0;

Document& operator = (const Document &doc);

static void *print*() {

for (int ind = 0; ind < *size*; ind++) {

*docs*[ind]->show();

}

}

};

class Receipt: public Document{

protected:

int cost;

char \*product\_name;

public:

Receipt();

Receipt(int \_number, const char \*\_name, int \_cost, const char \*\_product\_name);

Receipt(const Receipt &rec);

Receipt& operator = (const Receipt &rec);

void add();

void show();

~Receipt() { delete[] product\_name; }

};

class Check: public Receipt{

int nds\_percent;

public:

Check() : Receipt(), nds\_percent(0) {}

Check(int \_number, const char \*\_name, int \_cost, const char \*\_product\_name, int \_nds) :

Receipt(\_number, \_name, \_cost, \_product\_name), nds\_percent(\_nds) {}

Check(const Check &ch) : Receipt(ch), nds\_percent(ch.nds\_percent) {}

Check& operator = (const Check &ch);

void show();

};

class Invoice: public Receipt{

int delivery\_cost;

public:

Invoice() : Receipt(), delivery\_cost(0) {}

Invoice(int \_number, const char \*\_name, int \_cost, const char \*\_product\_name, int \_delivery\_cost) :

Receipt(\_number, \_name, \_cost, \_product\_name), delivery\_cost(\_delivery\_cost) {}

Invoice(const Invoice &inv) : Receipt(inv), delivery\_cost(inv.delivery\_cost) {}

Invoice& operator = (const Invoice &inv);

void show();

};

docs.cpp

#include "docs.h"

Document::Document() {

number = 0;

name = new char[1];

name[0] = '\0';

}

Document::Document(int \_number, const char \*\_name) {

number = \_number;

name = new char[strlen(\_name) + 1];

strcpy(name, \_name);

}

Document::Document(const Document &doc) {

number = doc.number;

name = new char[strlen(doc.name) + 1];

strcpy(name, doc.name);

}

Document& Document::operator = (const Document &doc) {

delete[] name;

number = doc.number;

name = new char[strlen(doc.name) + 1];

strcpy(name, doc.name);

return \*this;

}

Receipt::Receipt() : Document(), cost(0) {

product\_name = new char[1];

product\_name[0] = '\0';

add();

}

Receipt::Receipt(int \_number, const char \*\_name, int \_cost, const char \*\_product\_name) : Document(\_number, \_name) {

cost = \_cost;

product\_name = new char[strlen(\_product\_name) + 1];

strcpy(product\_name, \_product\_name);

add();

}

Receipt::Receipt(const Receipt &rec) : Document(rec) {

cost = rec.cost;

product\_name = new char[strlen(rec.product\_name) + 1];

strcpy(product\_name, rec.product\_name);

add();

}

Receipt& Receipt::operator = (const Receipt &rec) {

delete[] name;

delete[] product\_name;

number = rec.number;

name = new char[strlen(rec.name) + 1];

strcpy(name, rec.name);

cost = rec.cost;

product\_name = new char[strlen(rec.product\_name) + 1];

strcpy(product\_name, rec.product\_name);

return \*this;

}

void Receipt::add() {

if (Document::*size* == Document::*capacity*) {

Document::*capacity* \*= 2;

Document \*\*tmp = new Document\*[Document::*capacity*];

for (int i = 0; i < Document::*size*; i++) {

tmp[i] = Document::*docs*[i];

}

tmp[Document::*size*++] = this;

Document::*docs* = tmp;

}

else {

Document::*docs*[Document::*size*++] = this;

}

}

void Receipt::show() {

std::cout << "Document type: Receipt\nDocument number: " << number << "\nDocument name: " << name << '\n';

std::cout << "Cost: " << cost << "\nProduct name: " << product\_name << "\n\n";

}

Check& Check::operator = (const Check &ch) {

delete[] name;

delete[] product\_name;

number = ch.number;

name = new char[strlen(ch.name) + 1];

strcpy(name, ch.name);

cost = ch.cost;

product\_name = new char[strlen(ch.product\_name) + 1];

strcpy(product\_name, ch.product\_name);

nds\_percent = ch.nds\_percent;

return \*this;

}

void Check::show() {

std::cout << "Document type: Check\nDocument number: " << number << "\nDocument name: " << name << '\n';

std::cout << "Cost: " << cost << "\nProduct name: " << product\_name << "\nValue added tax: " << nds\_percent << "%\n\n";

}

Invoice& Invoice::operator = (const Invoice &inv) {

delete[] name;

delete[] product\_name;

number = inv.number;

name = new char[strlen(inv.name) + 1];

strcpy(name, inv.name);

cost = inv.cost;

product\_name = new char[strlen(inv.product\_name) + 1];

strcpy(product\_name, inv.product\_name);

delivery\_cost = inv.delivery\_cost;

return \*this;

}

void Invoice::show() {

std::cout << "Document type: Invoice\nDocument number: " << number << "\nDocument name: " << name << '\n';

std::cout << "Cost: " << cost << "\nProduct name: " << product\_name << "\nDelivery cost: " << delivery\_cost << "\n\n";

}

CMakeLists.txt библиотеки docs

cmake\_minimum\_required(VERSION 3.23)

set(project "docs")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

docs.cpp)

set(${project}\_HEADERS

docs.h)

set(${project}\_SOURCE\_LIST

${${project}\_SOURCES}

${${project}\_HEADERS})

add\_library(${project}

STATIC

${${project}\_SOURCE\_LIST})

main.cpp

#include "docs/docs.h"

Document \*\*Document::docs = new Document\*[1];

int Document::size = 0;

int Document::capacity = 1;

int main() {

Receipt r(123, "receipt mem", 120, "Coca-cola");

Check c(1, "check pepsi", 80, "Pepsi-cola", 20);

Invoice i(5, "invoice for delivery TV", 50000, "LG 4K", 3000);

Document::print();

delete[] Document::docs;

return 0;

}

CMakeLists.txt проекта

cmake\_minimum\_required(VERSION 3.23)

set(project "lab07")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

main.cpp)

add\_subdirectory(docs)

set(${project}\_SOURCE\_LIST

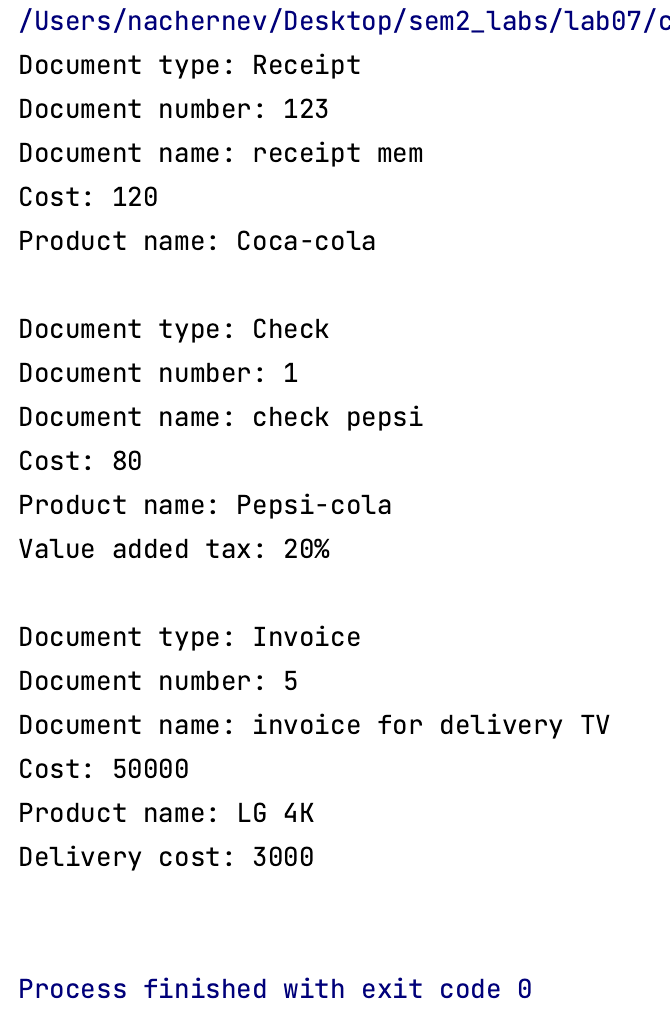
${${project}\_SOURCES})

add\_executable(${project}

${${project}\_SOURCE\_LIST})

target\_link\_libraries(${project} docs)

**Анализ результатов**

****