Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе № 15.1**

Дисциплина: «Информатика»

Тема: ООП. Классы и объекты. Инкапсуляция

Вариант 12

Выполнила:

Студентка группы РИС-22-1б

Черкасова А.А.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь 2023**

**Содержание**

**Введение**

**Цель:**

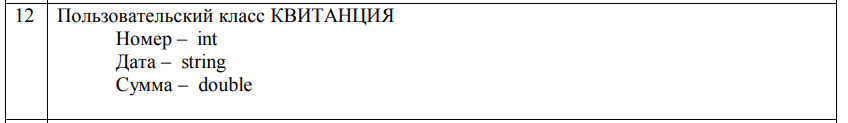
* Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
* Создание объектов с использованием конструктора.

Для достижения поставленной цели, необходимо решить следующие **задачи**:

* Провести анализ задачи
* Реализовать задачу на языке С++
* Составить блок-схему

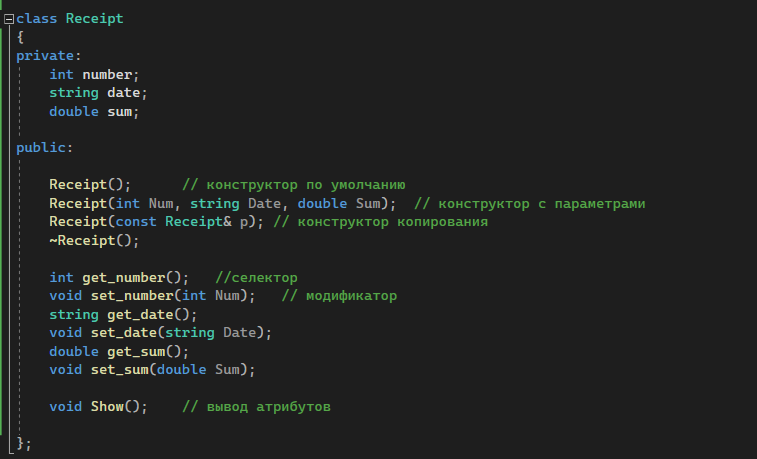
**Постановка задачи:**

1. Определить пользовательский класс
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора-копирования, вызов конструктора с параметрами и конструктора без параметров.

****

# **Анализ задачи**

1. Для решения задачи необходимо:
   1. Реализовать определение класса Receipt.



* 1. Реализовать определение конструкторов без параметра, с параметрами и копирования, деструктора, селекторов и модификаторов класса.
  2. Разработать функцию make\_Receipt для выделения памяти под новые данные в атрибутах объекта и обращения к конструктору с параметрами.
  3. Разработать функцию Print\_Receipt для обращения к методу вывода полей.
  4. Реализовать применение этих функций в главной функции.

1. В ходе работы были использованы типы данных:
   1. Для конструктора без параметров Receipt класса Receipt не используются аргументы:

Сам конструктор заполняет атрибуты объекта класса “пустотами”

* 1. Для конструктора с параметрами Receipt класса Receipt используются следующие аргументы:
  2. Тип int: номер квитанции.
  3. Тип string: дата.
  4. Тип double: сумма.

Сам конструктор заполняет атрибуты объекта класса параметрами.

* 1. Для конструктора копирования Receipt класса Receipt используются следующие аргументы:
  2. Адресация типа const Receipt: объект копирования.

Сам конструктор заполняет атрибуты объекта класса атрибутами объекта копирования.

* 1. Для деструктора Receipt класса Receipt не используются аргументы:

Сам деструктор удаляет атрибуты объекта класса.

* 1. Для селектора get\_number используются следующие аргументы:
  2. Тип int: номер квитанции.

Сам селектор имеет тип int и возвращает значение поля «номер квитанции»

* 1. Для селектора get\_date используются следующие аргументы:
  2. Тип string: дата.

Сам селектор имеет тип string и возвращает значение поля “дата”.

* 1. Для селектора get\_sum используются следующие аргументы:

1. Тип double: сумма.

Сам селектор имеет тип double и возвращает значение поля “сумма”.

* 1. Для модификатора set\_number используются следующие аргументы:

1. Тип int: номер квитанции.

Сам модификатор имеет тип void, поскольку при работе селектора не нужно возвращать значение.

* 1. Для модификатора set\_date используются следующие аргументы:

1. Тип string: дата.

Сам модификатор имеет тип void, поскольку при работе селектора не нужно возвращать значение.

* 1. Для модификатора get\_sum используются следующие аргументы:

1. Тип double: сумма.

Сам модификатор имеет тип void, поскольку при работе селектора не нужно возвращать значение.

* 1. Для метода Show не используются аргументы:

Сам метод имеет тип void, поскольку при работе функции не нужно возвращать значение.

* 1. Для функции make\_Receipt не используются аргументы:

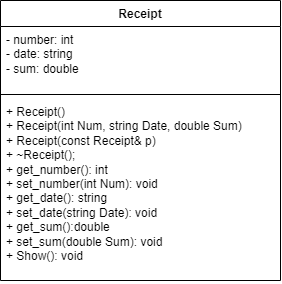
Сама функция имеет тип Receipt и возвращает объект с заполненными атрибутами.

* 1. Для функции Print\_Receipt используются следующие аргументы:
  2. Тип Receipt: объект для вывода атрибутов.

Сама функция имеет тип void, поскольку при работе селектора не нужно возвращать значение.

1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы с данными используются атрибуты класса.
2. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных реализован с помощью оператора cin и getline, используемых при реализации функции make\_Receipt, в главной функции и при реализации конструкторов и деструктора.
   2. Вывод данных реализован с помощью оператора cout, используемого при реализации метода Show и функций make\_Receipt и Print\_Receipt.
3. Поставленные задачи решены следующими действиями:
   1. Определение класса Receipt было реализовано в заголовочном файле Receipt.h, определение конструкторов, деструктора, селекторов и модификаторов класса – в файле Receipt.cpp. Основной блок программы и функция make\_Staff описаны в главном файле main.cpp.
   2. При работе с объектами класса Receipt обращение к конструктору без параметра аналогичен выделению памяти под переменную. Обращение к конструктору с параметрами аналогичен обращению к методу Init. Обращение к конструктору копирования аналогичен оператору присваивания. Обращение к селекторам и модификаторам аналогичен обращению к методам или к полям структуры.

# **UML – диаграмма**



# **Приложение А**

**Листинг программы**

*A) Заголовочный файл Tariff.h*

#pragma once

#include <string>

using namespace std;

class Receipt

{

private:

int number;

string date;

double sum;

public:

Receipt(); // конструктор по умолчанию

Receipt(int Num, string Date, double Sum); // конструктор с параметрами

Receipt(const Receipt& p); // конструктор копирования

~Receipt();

int get\_number(); //селектор

void set\_number(int Num); // модификатор

string get\_date();

void set\_date(string Date);

double get\_sum();

void set\_sum(double Sum);

void Show(); // вывод атрибутов

};

*B) Файл с описанием методов класса Tariff.cpp*

#include "Receipt.h"

#include <iostream>

#include <string>

using namespace std;

Receipt::Receipt() // конструктор по умолчанию

{

number = 0;

date = " ";

sum = 0.00;

cout << "Вызван конструктор без параметров" << endl;

}

Receipt::Receipt(int Num, string Date, double Sum) // конструктор с параметрами

{

number = Num;

date = Date;

sum = Sum;

cout << "Вызван конструктор с параметрами" << endl;

}

Receipt::Receipt(const Receipt& p) // конструктор копирования

{

number = p.number;

date = p.date;

sum = p.sum;

cout << "Вызван конструктор копирования" << endl;

}

Receipt::~Receipt()

{

cout << "Вызван деструктор" << endl;

}

int Receipt::get\_number()

{

return number;

}

void Receipt::set\_number(int Num)

{

number = Num;

}

string Receipt::get\_date()

{

return date;

}

void Receipt::set\_date(string Date)

{

date = Date;

}

double Receipt::get\_sum()

{

return sum;

}

void Receipt::set\_sum(double Sum)

{

sum = Sum;

}

void Receipt::Show() // вывод атрибутов

{

cout << "Номер: " << number << endl;

cout << "Дата: " << date << endl;

cout << "Сумма: " << sum << endl;

}

*C) Файл с главной программой main.cpp*

#include <iostream>

#include <string>

#include "Receipt.h"

using namespace std;

Receipt make\_Receipt() // Функция создания объекта и заполнения его полей данными

{

int number;

string date;

double sum;

cout << "Номер квитанции: ";

cin >> number;

cout << "Введите дату: ";

getline(cin, date);

getline(cin, date);

cout << "Введите сумму: ";

cin >> sum;

while (sum < 0)

{

cout << "Неверно введена сумма! Попробуйте ещё раз: ";

cin >> sum;

}

cin.ignore();

Receipt Element(number, date, sum);

return Element;

}

void Print\_Receipt(Receipt Element) // Функция вывода данных объекта

{

Element.Show(); // Метод вывода полей объекта

}

int main()

{

system("chcp 1251>nul");

Receipt Elem1; // Вызов конструктора без параметра для 1-го объекта

Elem1.Show(); // Вызов метода вывода полей 1-го объекта

Receipt Elem2(8903784, "01.01.2001", 150.089); // Вызов конструктора с параметрами для 2-го объекта

Elem2.Show(); // Вызов метода вывода полей 2-го объекта

Receipt Elem3 = Elem2; // Вызов конструктора копирования для 3-го объекта (копирование полей 2-го объекта)

Print\_Receipt(Elem3); // Функция вывода данных 3-го объекта

Elem3.set\_number(111111); // Вызов селектора 3-го объекта

Elem3.set\_date("12.06.2000"); // Вызов селектора поля 3-го объекта

Elem3.set\_sum(240.708); // Вызов селектора поля 3-го объекта

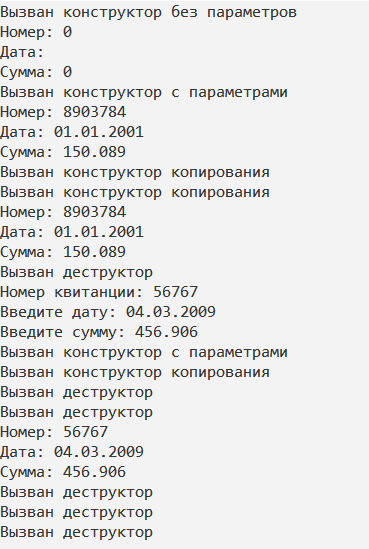
Elem1 = make\_Receipt(); // Вызов функции создания объекта и заполнения его полей данными (Происходит инициализация нового объекта и присваивание 1-му)

Elem1.Show(); // Метод вывода полей 1-го объекта

}

# **Приложение Б**

**Результаты выполнения программы**



**Ответы на вопросы**

1. Для чего нужен конструктор?

Конструктор – один из методов инициализации объекта созданного класса. Конструктор имеет одинаковое имя с классом.

1. Сколько типов конструкторов существует в С++?

Существует три типа конструкторов:

* Конструктор без параметров, для создания “пустого” объекта
* Конструктор с параметрами, для создания объекта с заданными значениями
* Конструктор копирования, для создания объекта с атрибутами ранее созданного.

1. Для чего используется деструктор? В каких случаях деструктор описывается явно?

Деструктор вызывается, когда нужно удалить объект из памяти. Описывать деструктор явно необходимо в случае, когда объект содержит указатели на память, выделяемую автоматически (иначе поля-указатели не буду свободными).

1. Для чего используется конструктор без параметров? Конструктор с параметрами? Конструктор копирования?

Конструктор без параметров используется для создания “пустого” объекта.

Конструктор с параметрами используется для инициализации объекта требуемыми значениями.

Конструктор копирования используется для создания нового объекта путём копирования существующего.

1. В каких случаях вызывается конструктор копирования?

Конструктор копирования вызывается:

* При описании нового объекта с инициализацией другим объектом
* При передаче объекта в функцию по значению
* При возврате объекта из функции

1. Перечислить свойства конструкторов.

* Не возвращает значение, нельзя получить указатель на конструктор
* В классе может быть несколько конструкторов (нужна перегрузка)
* Тип аргументов в конструкторе любой, кроме класса
* Конструктор без параметров – конструктор по умолчанию
* Не указан какой-либо конструктор – компилятор создает конструктор по умолчанию автоматически
* Не наследуются
* Нельзя описывать с модификаторами const, virtual, static

1. Перечислить свойства деструкторов.

* Нет аргументов и не возвращает значение
* Не наследуется
* Нельзя описывать с модификаторами const, static
* Можно описывать с модификатором virtual

1. К каким атрибутам имеют доступ методы класса?

Методы класса имеют доступ ко всем элементам класса, независимо от спецификаторов доступа и порядка объявления.

1. Что представляет собой указатель this?

Указатель this – указатель на объект, используемый в каком-либо из методов класса. Используется в качестве возвращающегося значения.

1. Какая разница между методами, определенными внутри класса и вне класса?

Разница между методами, определенными внутри класса и вне класса состоит лишь в их описании.

1. Какое значение возвращает конструктор?

Любой тип конструктора не возвращает значения.

1. Какие методы создаются по умолчанию?

По умолчанию создаются конструкторы и деструктор.

1. Какое значение возвращает деструктор?

Деструктор не возвращает значение.

1. Дано описание класса

class Student

{

string name;

int group;

public:

Student(string, int);

Student(const Student&);

~Student();

};

Какой метод отсутствует в описании класса?

В данном случае отсутствует конструктор без параметров.

1. Какой метод будет вызван при выполнении следующих операторов:

Student\* s;

s = new Student;

В данном случае будет вызван конструктор без параметров.

1. Какой метод будет вызван при выполнении следующих операторов:

Student s(“Ivanov”, 20);

В данном случае будет вызван конструктор с параметрами.

1. Какие методы будут вызваны при выполнении следующих операторов:

Student s1(“Ivanov”, 20);

Student s2 = s1;

В данном случае будут вызваны конструктор с параметрами и конструктор копирования соответственно.

1. Какие методы будут вызваны при выполнении следующих операторов:

Student s1(“Ivanov”, 20);

Student s2;

s2 = s1;

В данном случае будут вызваны конструктор с параметрами, конструктор без параметров и конструктор копирования соответственно.

1. Какой конструктор будет использоваться при передаче параметра в функцию print():

void print(student a)

{

a.show();

}

В данном случае будет вызван конструктор копирования.

1. Класс описан следующим образом:

class Student

{

string name;

int age;

public:

void set\_name(string);

void set\_age(int);

…….

};

Student p;

Каким образом можно присвоить новое значение атрибуту name объекта p?

Присвоение нового значения атрибуту name объекта p осуществляется путём обращения к модификатору, которое аналогично обращению к полям структуры:

String row = “Ivanov”; // строка, хранит данные для атрибута name

p.set\_name(string row);