Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.2**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Классы и объекты. Инкапсуляция.

Вариант 12

Выполнил:

студент группы РИС-20-2б

Пономарев Егор Витальевич

Проверила:

доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов.
2. Создание объектов с использованием конструкторов.

**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 (селекторы и модификаторы).

5. Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора: конструктора копирования, конструктора с параметрами и конструктора без параметров.

**Задание варианта:**

Пользовательский класс КВИТАНЦИЯ

Номер- int

Дата- string

Сумма- double

**Анализ задачи**

1. Определение класса:

class Tovar

{

string data;

int num;

double sum;

public:

Tovar();

Tovar(string, int, double);

Tovar(const Tovar&);

~Tovar();

string get\_data();

void set\_data(string);

int get\_num();

void set\_num(int);

double get\_sum();

void set\_sum(double);

void print();

};

1. Определение компонентных функций:

Tovar::Tovar()

{

data = "";

num = 0;

sum = 0;

cout << "конструктор без параметров для обЪекта" << this<<endl;

}

Tovar::Tovar(string d, int n, double s)

{

data = d;

num = n;

sum = s;

cout << "конструктор с параметрами для обЪекта" << this << endl;

}

Tovar::Tovar(const Tovar &t)

{

data = t.data;

num = t.num;

sum = t.sum;

cout << "конструктор копирования для обЪекта" << this << endl;

}

Tovar::~Tovar()

{

cout << "Деструктор для обЪекта" << this << endl;

}

string Tovar::get\_data()

{

return data;

}

int Tovar::get\_num()

{

return num;

}

double Tovar::get\_sum()

{

return sum;

}

void Tovar::set\_data(string d)

{

data = d;

}

void Tovar::set\_num(int n)

{

num=n;

}

void Tovar::set\_sum(double s)

{

sum = s;

}

void Tovar::print()

{

cout << "Дата: " << data << endl;

cout << "Номер: " << num << endl;

cout << "Сумма: " << sum << endl;

}

1. Определение функции make:

Tovar make\_tovar()

{

string s;

int i;

double d;

cout << "Введите дату : ";

cin >> s;

cout << "Введите номер: ";

cin >> i;

cout << "Введите сумму: ";

cin >> d;

Tovar t(s, i, d);

return t;

}

1. определение функции print:

void print\_Tovar(Tovar t)

{

t.print();

}

1. Функция main():

void main()

{

setlocale(0, "");

Tovar t1;

t1.print();

Tovar t2("14.05.2007",1,70000);

t2.print();

Tovar t3 = t2;

t3.set\_data("12.01.2002");

t3.set\_num(1);

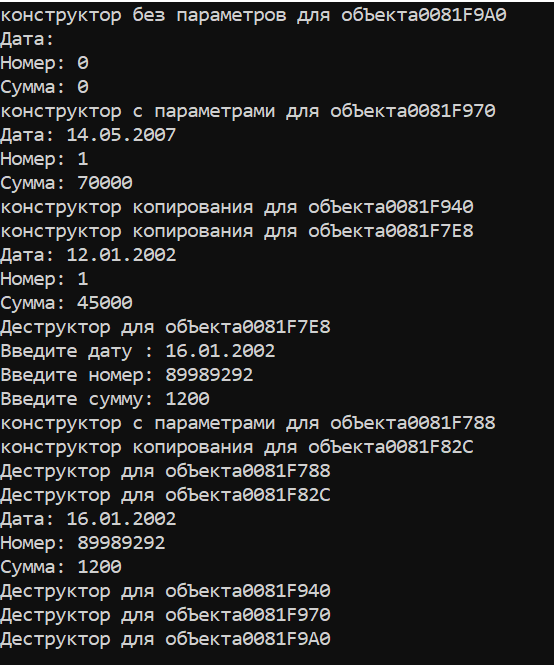
t3.set\_sum(45000.0);

print\_Tovar(t3);

t1 = make\_tovar();

t1.print();

**Результат работы программы**



**Ответы на вопросы**

1. Класс – это конструкция, объединяющая данные и функции, которые в основном обрабатывают эти данные.
2. Объект (экземпляр класса) – это переменная класса.
3. Поля класса называются атрибутами класса.
4. Функции класса называются методами.
5. Спецификаторы доступа используются для указания возможности доступа к данным (это применяется с целью сокрытия данных, для того, чтобы уберечь данные от несанкционированного доступа или случайного изменения).
6. Спецификатор public используется для открытия доступа к полям и методам экземпляра данного класса.
7. Спецификатор private используется для сокрытия полей и методов экземпляра класса.
8. Для классов умолчанию используется спецификатор доступа private.
9. Для структур по умолчанию используется спецификатор доступа public.
10. Для описания интерфейса класса должен использоваться спецификатор public, чтобы пользователь имел возможность взаимодействовать с методами класса.
11. Изменить значения атрибутов класса можно с помощью методов класса, называемых модификаторами.
12. Получить значения атрибутов класса можно с помощью методов класса, называемых селекторами.
13. s->name;
14. s.name;
15. С помощью функции-модификатора, если она имеется.
16. С помощью функции-модификатора, если она имеется.
17. s->name;