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 №18.7**

Дисциплина: «Основы теории алгоритмов и структуры данных»  
Тема: Шаблоны классов

Вариант 12

Выполнил:

студент группы РИС-20-2б

Пономарев Егор Витальевич

Проверила:

доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2021

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов.

2. Реализация шаблона класса-контейнера.

**Постановка задачи**

1. Определить шаблон класса-контейнера.

2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.

3. Перегрузить необходимые операции.

4. Инстанцировать шаблон для стандартных типов данных.

5. Реализовать пользовательский класс .

6. Перегрузить для пользовательского класса операции ввода-вывода.

7. Перегрузить операции, необходимые для выполнения операций контейнерного класса.

8. Инстанцировать шаблон для пользовательского класса.

9. Написать тестирующую программу, демонстрирующую выполнение указанных заданий.

**Задание варианта:**

Кдасс-контейнер СПИСОК с ключевыми значениями типа int.

Реализовать операции:

[]-доступа по индексу;

()-определение размера вектора;

+число-добавляет константу ко всем элементам вектора;

Пользовательский класс Pair(пара чисел). Пара должна быть представлена двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием.

**Анализ задачи**

1. Описание структуры:

struct fraction {

double first;

int second;

void Init(double, int); // метод для инициализации полей

void Read(); // метод для чтения значений полей

void Show(); // метод для вывода значений полей

double Summa(); // вычисление степени

};

void fraction::Init(double F, int S) {

while (F < 0 || S < 0) {

cin >> F;

cin >> S;

}

first = F;

second = S;

}

void fraction::Read() {

cout << "\nfirst? ";

cin >> first;

while (first < 0) {

cin >> first;

}

cout << "\nsecond? ";

cin >> second;

while (second < 0) {

cin >> first;

}

}

void fraction::Show() {

cout << "\nfirst = " << first;

cout << "\nsecond = " << second;

cout << endl;

}

double fraction::Summa() {

return first \* second;

}

fraction make\_fraction(double F, int S) {

fraction t;

t.Init(F, S);

return t;

}

int main()

{

fraction A;

fraction B;

A.Init(3.0, 2);

B.Read();

A.Show();

B.Show();

cout << "A.Summa(" << A.first << ", " << A.second << ") = " << A.Summa() << endl;

cout << "B.Summa(" << B.first << ", " << B.second << ") = " << B.Summa() << endl;

fraction\* X = new fraction;

X->Init(2.0, 5);

X->Show();

X->Summa();

cout << "X.Summa(" << X->first << ", " << X->second << ") = " << X->Summa() << endl;

fraction mas[3];

for (int i = 0; i < 3; i++)

mas[i].Read();

for (int i = 0; i < 3; i++)

mas[i].Show();

for (int i = 0; i < 3; i++) {

mas[i].Summa();

cout << "mas[" << i << "].Summa(" << mas[i].first << ", " << mas[i].second << ") = ";

cout << mas[i].Summa() << endl;

}

fraction\* p\_mas = new fraction[3];

for (int i = 0; i < 3; i++)

p\_mas[i].Read();

for (int i = 0; i < 3; i++)

p\_mas[i].Show();

for (int i = 0; i < 3; i++) {

p\_mas[i].Summa();

cout << "p\_mas[" << i << "].Summa(" << p\_mas[i].first << ", " << p\_mas[i].second << ") = ";

cout << p\_mas[i].Summa() << endl;

}

double y;

int z;

cout << "first? ";

cin >> y;

cout << "second? ";

cin >> z;

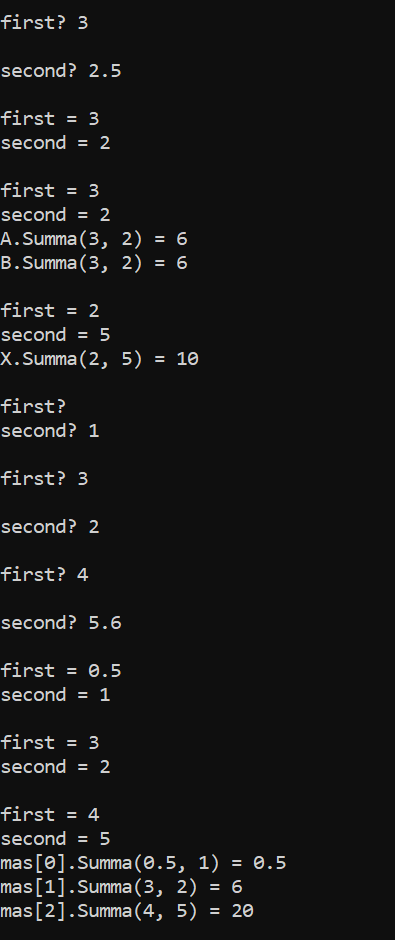
fraction F = make\_fraction(y, z);

F.Show();

return 0;

}

**Результаты работы программы**



**Ответы на вопросы**

1. Шаблоны используются, чтобы не создавать множество однотипных функций или классов для разных типов данных. Шаблоны позволяют сэкономить время написания кода и уменьшить его размер.
2. template <параметры\_шаблона> тип\_функции имя\_функции {}
3. template <параметры\_шаблона> class имя\_класса {};
4. Это параметры неопределённого типа, которыми оперирует функция.
5. При вызове функции компилятор определяет тип значения, переданного в функцию в качестве аргумента и устанавливает тип данных параметру шаблона функции. Габлоны функций не могут быть виртуальными.
6. Template <class T> (T – имя параметризуемого типа).
7. Можно.
8. Шаблоны классов могут содержать статические элементы, дружественные функции и классы. Шаблоны могут быть производными как от шаблонов, так и от обычных классов, а также являться базовыми и для шаблонов и для обычных классов.
9. Нет.
10. Не все.
11. Да.
12. С использованием синтаксиса объявления шаблона перед функцией.
13. Процесс генерации компилятором определения конкретного класса по шаблону класса и аргументам шаблона.
14. При создании экземпляра класса с указанием типа данных в качестве параметра.