МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа № 7:**

"Шаблоны классов."

Выполнила: студентка гр.РИС-23-3Б

Мокроусова Ангелина Андреевна

Проверила: доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2024

**Постановка задачи:**

Общая часть:

1. Определить шаблон класса-контейнер (см. лабораторная работа №6)
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции указанные в варианте.
4. Инстанцировать шаблон для стандартных типов данных(int, float, double).
5. Написать тестирующую программу, иллюстрирующую выполнение операций контейнера, содержащего элементы стандартных типов.
6. Реализовать пользовательский класс (см. лабораторную работу №3).
7. Перегрузить для пользовательского класса операции ввода-вывода.
8. Перегрузить операции необходимые для выполнения операций контейнерного класса.
9. Инстанцировать шаблон для пользовательского класса.
10. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы пользовательского класса.

Задание варианта:

Класс-контейнер СПИСОК с ключевыми значениями типа int.

Реализовать операции:

[] – доступа по индексу

+вектор – сложение элементов списков a[i] + b[i]

+число – добавляет константу ко всем элементам списка

-- – переход к предыдущему элементу

Создать класс Pair (пара чисел). Пара должна быть представлено 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:**

Класс Pair:

* Имеет две переменные-члена для хранения первого и второго элементов пары.
* Предоставляет геттеры и сеттеры для доступа к элементам.
* Имеет конструкторы по умолчанию, с параметрами и с копированием.
* Реализует операторы сравнения, префиксного декремента, сложения, ввода и вывода.

Класс List:

* Имеет массив пар и переменную-член для хранения количества элементов в списке.
* Обеспечивает доступ к элементам списка с помощью оператора индексации.
* Предоставляет функцию для получения ключа указанного элемента.
* Перегружает операторы сложения для суммирования списков попарно или с целым числом.
* Также имеет конструкторы по умолчанию, с копированием и деструктор.
* Реализует операторы ввода и вывода.

Основная функция main:

* Вводит размеры и значения для списков различных типов данных (int, double, float).
* Выводит результаты выполнения операций сложения, суммирования со скалярным значением и префиксного декремента.

**Код на С++ с комментариями:**

// Включаем стандартную библиотеку ввода-вывода

#include <iostream>

using namespace std;

// Определяем шаблон класса Pair, который будет хранить пары элементов двух типов

template <typename T1, typename T2>

class Pair {

private:

// Переменные-члены класса

T1 first;

T2 second;

public:

// Конструктор без параметров

Pair() : first(0), second(0) {}

// Конструктор с двумя параметрами

Pair(T1 first, T2 second) : first(first), second(second) {}

// Конструктор копирования

Pair(const Pair& other) : first(other.first), second(other.second) {}

// Деструктор

~Pair() {}

// Геттеры для получения значений переменных-членов

T1 getFirst() const { return first; }

T2 getSecond() const { return second; }

// Сеттеры для установки значений переменных-членов

void setFirst(T1 first) { this->first = first; }

void setSecond(T2 second) { this->second = second; }

// Оператор присваивания

Pair& operator=(const Pair& other) {

first = other.first;

second = other.second;

return \*this;

}

// Операторы сравнения

bool operator<(const Pair& other) const {

return first < other.first;

}

bool operator>(const Pair& other) const {

return second > other.second;

}

// Оператор префиксного декремента

Pair& operator--() {

first--;

return \*this;

}

// Оператор сложения

Pair operator+(const Pair& other) {

return Pair(first + other.first, second + other.second);

}

// Оператор ввода

friend istream& operator>>(istream& in, Pair& pair) {

cout << "Введите первый элемент пары: ";

in >> pair.first;

cout << "Введите второй элемент пары: ";

in >> pair.second;

return in;

}

// Оператор вывода

friend ostream& operator<<(ostream& out, const Pair& pair) {

out << "[" << pair.first << ":" << pair.second << "]";

return out;

}

};

// Определяем шаблон класса List, который будет хранить списки пар элементов одного типа

template <typename T>

class List {

private:

// Массив пар элементов

Pair<T, T>\* elements;

// Количество элементов в списке

int num\_elements;

public:

// Конструктор без параметров

List() : elements(nullptr), num\_elements(0) {}

// Конструктор копирования

List(const List& other) : elements(new Pair<T, T>[other.num\_elements]), num\_elements(other.num\_elements) {

for (int i = 0; i < num\_elements; i++)

elements[i] = other.elements[i];

}

// Оператор присваивания

List& operator=(const List& other) {

if (this != &other) {

delete[] elements;

elements = new Pair<T, T>[other.num\_elements];

num\_elements = other.num\_elements;

for (int i = 0; i < num\_elements; i++)

elements[i] = other.elements[i];

}

return \*this;

}

// Деструктор

~List() { delete[] elements; }

// Оператор доступа к элементу списка

Pair<T, T>& operator[](int index) { return elements[index]; }

// Функция для получения ключа элемента списка

int getKey(int index) { return elements[index].getFirst(); }

// Оператор сложения для суммирования элементов двух списков

List operator+(int value) {

List result(\*this);

for (int i = 0; i < num\_elements; i++)

elements[i].setSecond(elements[i].getSecond() + value);

return result;

}

// Оператор сложения для суммирования списков попарно

List operator+(const List& other) {

List result;

result.num\_elements = num\_elements;

result.elements = new Pair<T, T>[result.num\_elements];

for (int i = 0; i < num\_elements; i++) {

result.elements[i] = elements[i] + other.elements[i];

}

return result;

}

// Оператор префиксного декремента

List& operator--() {

num\_elements--;

return \*this;

}

// Оператор ввода

friend istream& operator>>(istream& in, List& list) {

int num\_elements;

in >> num\_elements;

list.elements = new Pair<T, T>[num\_elements];

list.num\_elements = num\_elements;

for (int i = 0; i < num\_elements; i++)

in >> list.elements[i];

return in;

}

// Операторр вывода

friend ostream& operator<<(ostream& out, const List& list) {

for (int i = 0; i < list.num\_elements; i++)

out << list.elements[i] << " ";

return out;

}

};

// Основная фкнкция

int main() {

setlocale(LC\_ALL, "ru");

List<int> list\_int;

cout << "Введите размер списка для элементов типа int: ";

cin >> list\_int;

cout << "Получившийся список: " << list\_int;

cout << endl << "=============================================================================================================" << endl;

List<double> list\_double;

cout << "Введите размер списка для элементов типа double: ";

cin >> list\_double;

cout << "Получившийся список: " << list\_double;

cout << endl << "=============================================================================================================" << endl;

List<float> list\_float;

cout << "Введите размер списка для элементов типа float: ";

cin >> list\_float;

cout << "Получившийся список: " << list\_float;

cout << endl << "=============================================================================================================" << endl;

List<int> list1;

cout << "Введите размер списка для элементов типа int: ";

cin >> list1;

List<int> list2;

cout << "Введите размер списка для элементов типа int: ";

cin >> list2;

cout << endl << "=============================================================================================================" << endl;

List<int> result = list1 + list2;

cout << "Сумма списков: " << result << endl;

cout << endl << "=============================================================================================================" << endl;

int a;

cout << "Введите число для суммирования со вторым числом пар: ";

cin >> a;

result = result + a;

cout << "Сумма со вторым числом пар и числом: " << result << endl;

cout << endl << "=============================================================================================================" << endl;

--result;

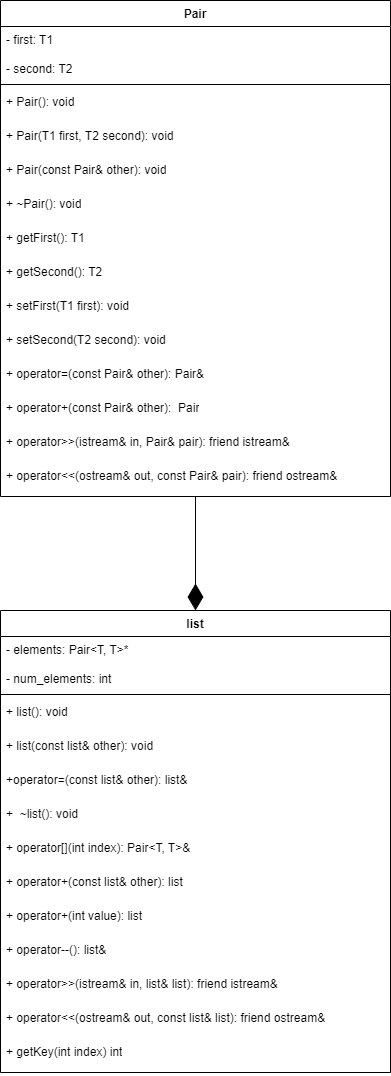
cout << "Список после оператора '--': " << result << endl;

cout << endl << "=============================================================================================================" << endl;

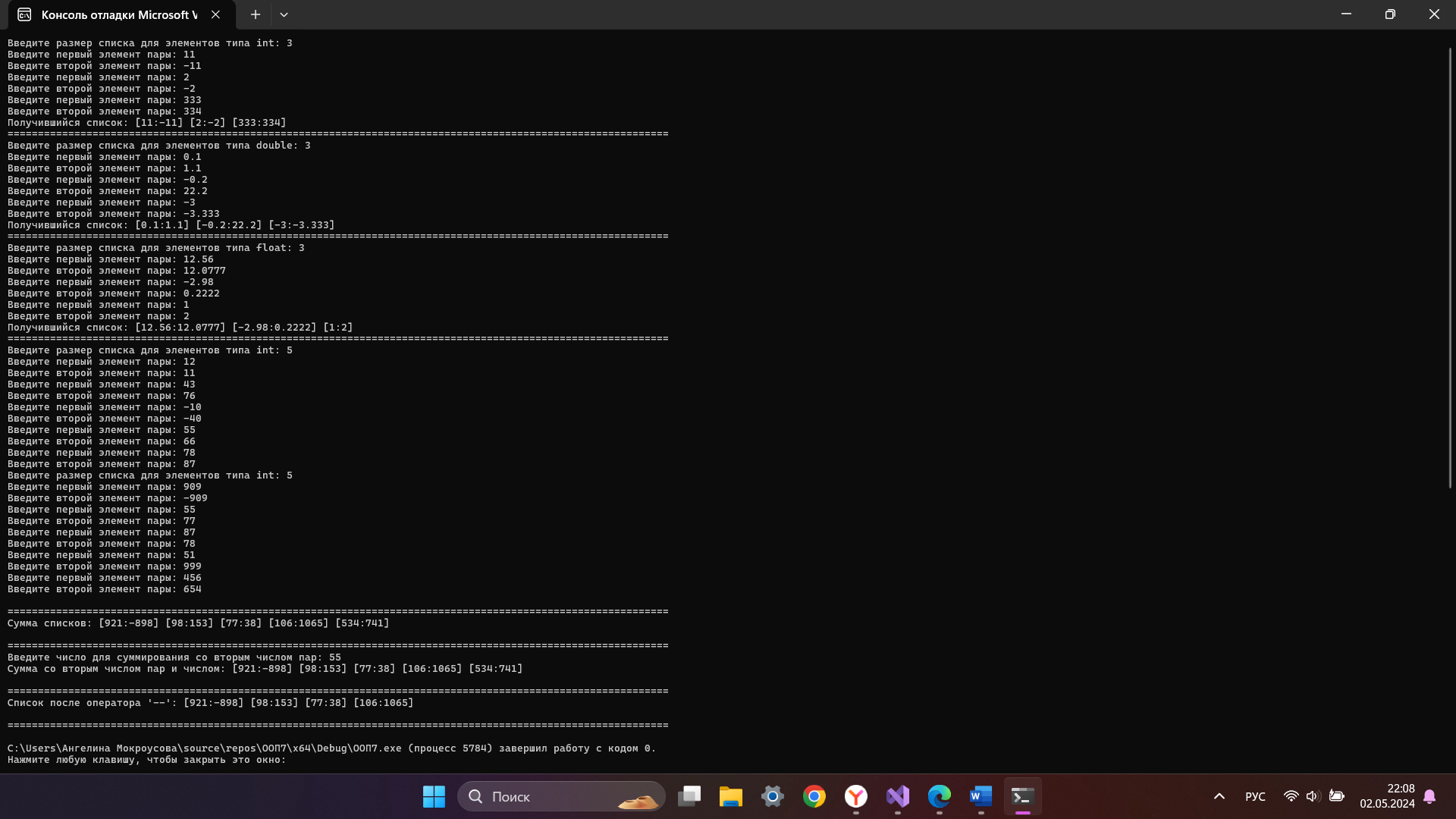
return 0;

}

**UML диаграмма:**

****

**Скриншот работы программы:**



**Вывод:**

Мною была аписана программа, которая успешно выполняет поставленную перед ней задачу. В процессе написания был повторен и закреплен на практике пройденный ранее материал.

**Ответы на вопросы:**

1. В чем смысл использования шаблонов?

Шаблоны позволяют использовать один и тот же код для множества типов данных, при этом функцию/класс не приходится переделывать вручную под каждый тип данных;

1. Каковы синтаксис/семантика шаблонов функций?

template < параметры\_шаблона ><тип\_функции> <название> <параметры> (<аргументы>) {<тело>}

1. Каковы синтаксис/семантика шаблонов классов?

template <параметры\_шаблона >

class <название\_класса> {<тело\_класса>};

1. Что такое параметры шаблона функции?

Параметры щшаблона функции – некие “заменители”, на место которых будут подставляться конкретные типы данных при создании функции по шаблону компилятором;

1. Перечислите основные свойства параметров шаблона функции.

А) функция не может иметь 2 параметра с одним имененм, Б) функция может иметь неограниченное число параметров, Г) один параметр может замещать собой только один тип данных, Д) имя параметра шаблона видно во всем определении;

1. Как записывать параметр шаблона?

template <typename (имя\_параметра)>;

1. Можно ли перегружать параметризованные функции? Да;
2. Перечислите основные свойства параметризованных классов.

А) при объявлении параметризованного класса никакого кода на самом деле не генерируется, Б) параметризованные классы могут иметь дружеские функции и классы, также могут наследовать и быть наследовательными;

1. Все ли компонентные функции параметризованного класса являются параметризованными? Да;
2. Являются ли дружественные функции, описанные в параметризованном классе, параметризованными?

Только если они сами являются параметризованными, иначе будет существовать одна функция, дружественая ко всем классам, созданым по шаблону;

1. Могут ли шаблоны классов содержать виртуальные компонентные функции? Да;
2. Как определяются компонентные функции параметризованных классов вне определения шаблона класса?

Необходимо отдельно прописать параметр для данной функции;

1. Что такое инстанцирование шаблона?

Это процесс создание конкретных функций/классов из шаблона;

1. На каком этапе происходит генерирование определения класса по шаблону?

На этапе когда компилятор впервые встретил вызов шаблонной функции определенного типа.

**Скриншот с GitHab:**

