МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа № 6:**

"АТД. Контейнеры."

Выполнила: студентка гр.РИС-23-3Б

Мокроусова Ангелина Андреевна

Проверила: доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2024

**Постановка задачи:**

Общая часть:

1. Реализовать класс – контейнер.
2. Реализовать конструкторы, деструктор, операции ввода – вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Написать тестирующую программу, иллюстрирующую выполнение операций.

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

**Анализ задачи:**

Класс-контейнер List:

* Реализация динамического массива для хранения элементов.
* Предоставление конструкторов, деструктора, методов добавления и удаления элементов.
* Перегрузка операторов [] для доступа по индексу, + для сложения списков и сложения числа со списком.
* Перегрузка оператора -- для получения текущего элемента и перехода к следующему элементу.

Тестирующая программа:

* Создание экземпляров класса List.
* Демонстрация операций доступа по индексу, сложения и итерации.
* Вывод результатов операций на экран.

**Код на С++ с комментариями:**

#include <iostream>

#include <vector>

using namespace std;

// класс-контейнер list

template <typename t>

class list

{

public:

// конструктор

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

// конструктор копирования

list(const list& other) : elements(new 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 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; }

// доступ по индексу

t& operator[](int index) { return elements[index]; }

// сложение списков

list operator+(const list& other)

{

list result;

result.num\_elements = num\_elements;

result.elements = new t[result.num\_elements];

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

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

return result;

}

// сложение числа со списком

list operator+(t value)

{

list result(\*this);

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

result.elements[i] += value;

return result;

}

list& operator--()

{

num\_elements--;

return \*this;

}

// ввод списка

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

{

int num\_elements;

in >> num\_elements;

list.elements = new t[num\_elements];

list.num\_elements = num\_elements;

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

in >> list.elements[i];

return in;

}

// вывод списка

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

{

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

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

return out;

}

private:

t\* elements;

int num\_elements;

};

// тестирующая программа

int main()

{

setlocale(LC\_ALL, "ru");

// создание списков

list<int> list1;

list<int> list2;

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

// ввод списков

cout << "Введите количество элементов первого списока: ";

cin >> list1;

cout << "Введите количество элементов второго списока (должно быть одинаково в обоих списках): ";

cin >> list2;

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

// сложение списков

list<int> result = list1 + list2;

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

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

// сложение списка с числом

int a;

cout << "Введите число для суммирования с элементами списка: ";

cin >> a;

result = result + a;

cout << "Сумма с числом: " << result << endl;

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

// переход к предыдущему элементу списка

--result;

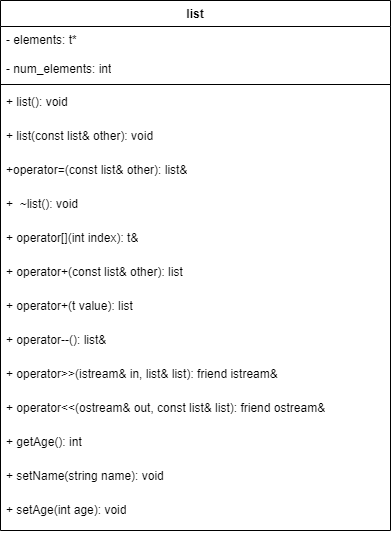
cout << "После оператора '--': " << result << endl;

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

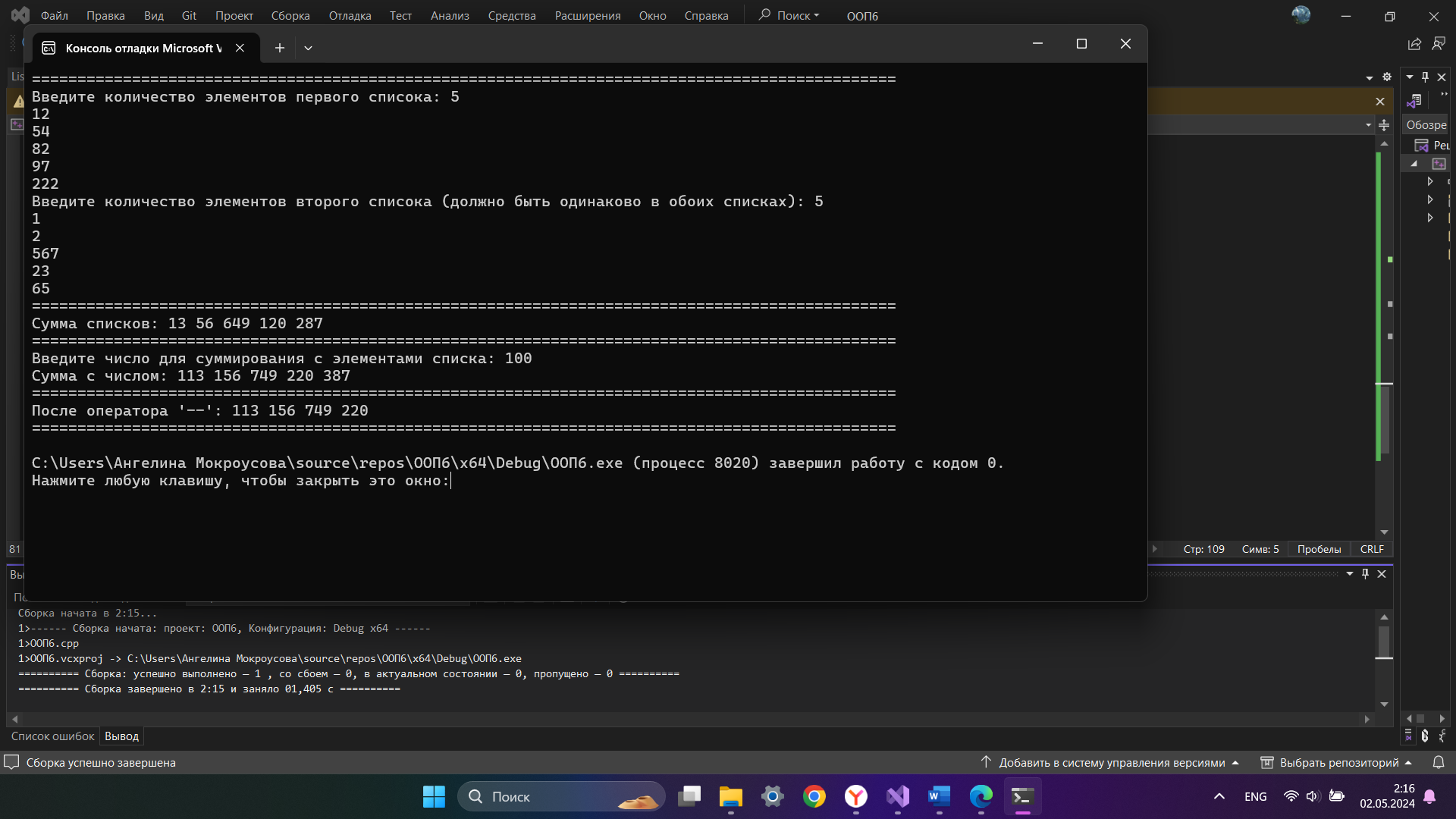
return 0;

}

**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. Что такое абстрактный тип данных? Привести примеры АТД. Абстрактный тип данных — это тип данных, доступ к которым осуществляется только через некоторый набор действий (операций, команд). Этот набор действий называется интерфейсом абстрактного типа данных. Примеры: классы и структуры c++;
2. Привести примеры абстракции через параметризацию.

Параметры функций. Есть некие переменные, на место которых можно подставить реальные объекты и по отношению в ним сделать действия, описанные в функции;

1. Привести примеры абстракции через спецификацию.

Значения, возвращаемые функциями. Мы не знаем, как именно это значение было получено, но мы записываем его в реальный объект нужного типа;

1. Что такое контейнер?

Привести примеры. Контейнер – набор данных одного типа. Примеры – массивы, стэки, списки и т.д.;

1. Какие группы операций выделяют в контейнерах?

Операции добавления, удаления, поиска, доступа и редактирования, объединения, специфические для конкретного вида;

1. Какие виды доступа к элементам контейнера существуют? Привести примеры.

Последовательный (<название\_контейнера>++ - вернет следующий элемент списка после текущего), прямой (<название\_контейнера>[<индекс элемента>]) и ассоциативный (<название\_контейнера>[<элемент – ключ, по которому хранится элемент>]);

1. Что такое итератор?

Итератор – объект, который позволяет удобно релизовать последовательный доступ к элемента контейнера;

1. Каким образом может быть реализован итератор?

Итератор может быть реализован как отдельный класс, объект которого является полем контейнера, либо же как набор методов самого контейнера;

1. Каким образом можно организовать объединение контейнеров?

А) В новый контейнер попадают элементы обоих контейнеров, Б) В новый контейнер элементы составляющих контейнеров попадают отсортироваными, В) В новый контейнер попадают те объекты, которые есть хотя бы в одном, Г) В новый контейнер попадают только те элементы, которые есть в обоих контейнерах, Д) В новый контейнер попадают только те элементы первого контейнера, которых нет во втором, Е) Излечение части элементов из контейнера в новый контейнер;

1. Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?

Ассоциативный;

1. Как называется контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера?

Стек;

1. Какой из объектов (a,b,c,d) является контейнером?

a) int mas=10;

b) int mas;

c) struct {char name[30]; int age;} mas;

d) int mas[100];

Ответ – d;

1. Какой из объектов (a,b,c,d) не является контейнером?

a) int a[]={1,2,3,4,5};

b) int mas[30];

c) struct {char name[30]; int age;} mas[30];

d) int mas;

Ответ – d;

1. Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера? Прямым;
2. Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера? Последовательным;

**Скриншот с GitHab:**