Министерство образования РФ

Пермский национальный исследовательский политехнический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 6

ПО ИНФОРМАТИКЕ ЗА II СЕМЕСТР

Вариант 15

Выполнил студент:

Сташков Арсений Дмитриевич

Группа РИС-20-1бз

Шифр 20-ЭТФ-644

Проверила:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

ПЕРМЬ 2022

# Лабораторная работа №6 "АТД. Контейнеры"

**Вариант №15**

**Цель:** 1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Реализация класса-контейнера.

**Задача:** 1. Определить класс-контейнер.

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

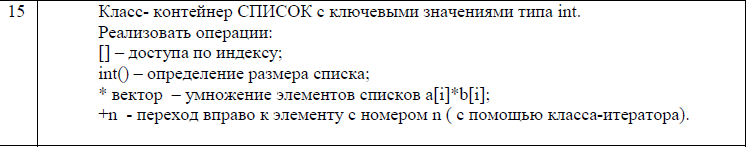
присваивания.

3. Перегрузить операции, указанные в варианте.

4. Реализовать класс-итератор. Реализовать с его помощью операции

последовательного доступа.

5. Написать тестирующую программу, иллюстрирующую выполнение операций.



**Решение задания:**

Файл List.h:

#pragma once

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

class Iterator {

friend class Listing;

int\* elem;

public:

Iterator() { elem = 0; }

Iterator(const Iterator& it) { elem = it.elem; }

// Перегруженные операции сравнения

bool operator == (const Iterator& it) { return elem == it.elem; }

bool operator != (const Iterator& it) { return elem != it.elem; };

// Перегруженные операции инкремента и декремента

void operator++ () { ++elem; };

void operator-- () { --elem; }

// Перегруженная операция разыменования

int& operator \*() const { return\*elem; }

};

struct Listing {

// Конструктор с параметрами: выделяет память под S элементов и заполняет их значением k

Listing(int s, int k = 0);

Listing(const Listing& a);

~Listing();

Listing& operator = (const Listing& a);

// Доступ по индексу

int& operator[] (int index);

// Умножение элементов списков

Listing operator\*(int a);

// Получение длины списка

int operator () ();

// Перегруженные операции ввода - вывода

friend ostream& operator << (ostream& out, const Listing& a);

friend istream& operator >> (istream& in, Listing& a);

Iterator first() { return beg; }

Iterator last() { return end; }

friend istream& operator >> (istream& in, Listing& a);

private:

// Размер списка

int size;

// Указатель на динамический массив значений списка

int\* data;

// Указатель на первый элемент списка

Iterator beg;

// Указатель на элемент, следующий за последним

Iterator end;

};

Файл List.cpp:

#include "List.h"

// Конструктор с параметрами

Listing::Listing(int s, int k) {

size = s;

data = new int[size];

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

{ data[i] = k; }

beg.elem = &data[0];

end.elem = &data[size];

}

// Конструктор копирования

Listing::Listing(const Listing& a) {

size = a.size;

data = new int[size];

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

{ data[i] = a.data[i]; }

beg = a.beg;

end = a.end;

}

// Деструктор

Listing::~Listing() { delete[]data; data = 0; }

// Присваивание

Listing& Listing::operator = (const Listing& a) {

if (this == &a) { return \*this; }

size = a.size;

if (data != 0) { delete[] data; }

data = new int[size];

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

{ data[i] = a.data[i]; }

beg = a.beg;

end = a.end;

return \*this;

}

// Доступ по индексу

int& Listing::operator[] (int index) {

if (index < size) { return data[index]; }

cout << "\nОшибка! Index больше, чем длина списка";

return data[0];

}

// Перемножение значений

Listing Listing::operator\*(int b) {

Listing temp(size);

temp.data[0] = data[0] \* data[b];

return temp;

}

// Переопределение оператора вызова функции

int Listing::operator () () { return size; }

// Ввод-вывод

ostream& operator << (ostream& out, const Listing& a) {

for (int i = 0; i < a.size; ++i)

{ out << a.data[i] << " "; }

return out;

}

istream& operator >> (istream& in, Listing& a) {

for (int i = 0; i < a.size; ++i)

{ in >> a.data[i]; }

return in;

}

Файл Контейнеры.cpp

#include "List.h"

#include <iostream>

#include <string>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

cout << "Создаем список из 5 элементов, заполненный нулями:\n";

Listing a(5);

cout << a << "\n";

cout << "\nТеперь введите 5 любых целых чисел:\n";

(cin >> a).get();

cout << "\nВы ввели числа:\n" << a << "\n";

cout << "\nПрисваиваем новое значение (100) элементу [2]:\n";

a[2] = 100;

cout << a << "\n";

cout << "\nСоздаем второй список:\n";

Listing b(5);

cout << b << "\n";

cout << "\nПрисваиваем списку b значения списка a:\n";

b = a;

cout << b << "\n";

Listing c(5);

cout << "\nПеремножаем значения [0] и [1] списка b, заносим в первый элемент и присваиваем списку с:\n";

c = b \* 1;

cout << c;

cout << "\n\nДлина списка a равна " << a() << endl;

cout << "\nРазыменование значения, которое возвращает a.first():\n";

cout << \*(a.first()) << endl;

cout << "\nУстановка переменной типа Iterator на второй элемент списка a:\n";

Iterator i = a.first();

++i;

cout << \*i << endl;

cout << "\nВыводим значения элементов списка с помощью итератора:\n";

for (i = a.first(); i != a.last(); ++i)

{ cout << \*i << endl; }

cin.get();

return 0;

}

**Результат:**

Создаем список из 5 элементов, заполненный нулями:

0 0 0 0 0

Теперь введите 5 любых целых чисел:

1

2

3

4

5

Вы ввели числа:

1 2 3 4 5

Присваиваем новое значение (100) элементу [2]:

1 2 100 4 5

Создаем второй список:

0 0 0 0 0

Присваиваем списку b значения списка a:

1 2 100 4 5

Перемножаем значения [0] и [1] списка b, заносим в первый элемент и присваиваем списку с:

2 0 0 0 0

Длина списка a равна 5

Разыменование значения, которое возвращает a.first():

1

Установка переменной типа Iterator на второй элемент списка a:

2

Выводим значения элементов списка с помощью итератора:

1

2

100

4

5

