Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе № 18.6**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. АТД. Контейнеры

Вариант 21

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

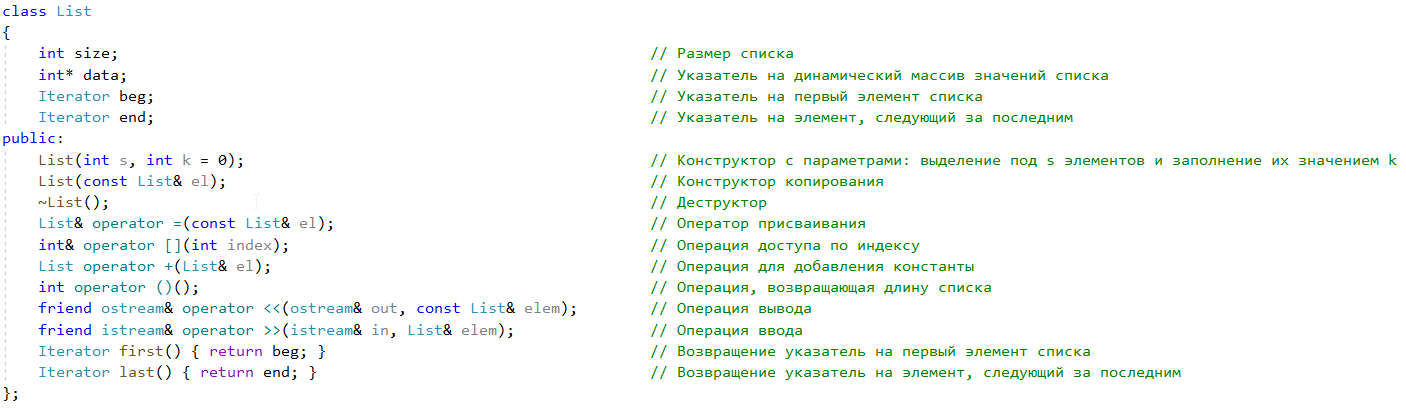
1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Реализация класса-контейнера.

**Постановка задачи**

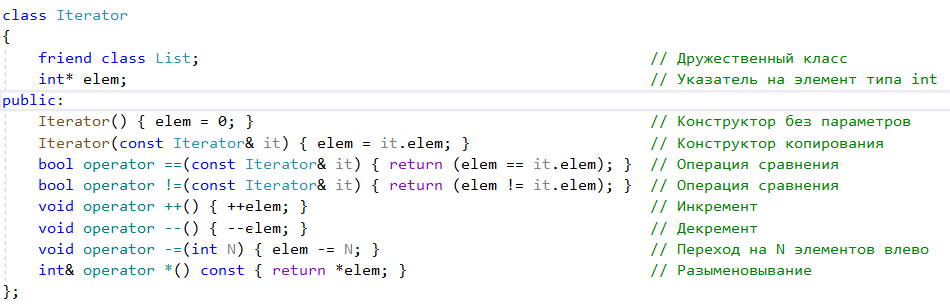
1. Определить класс-контейнер.
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.
5. Написать тестирующую программу, иллюстрирующую выполнение операций.

**Анализ задачи**

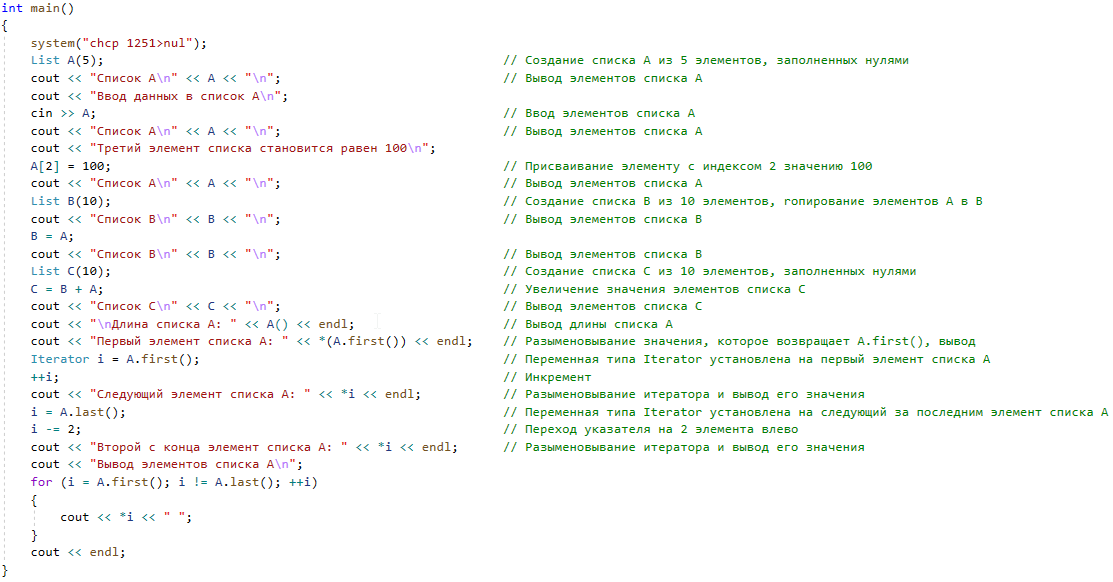
1. Для решения задачи необходимо:
   1. Реализовать определение шаблонного класса List и определение необходимых методов и перегруженных операторов данного класса.



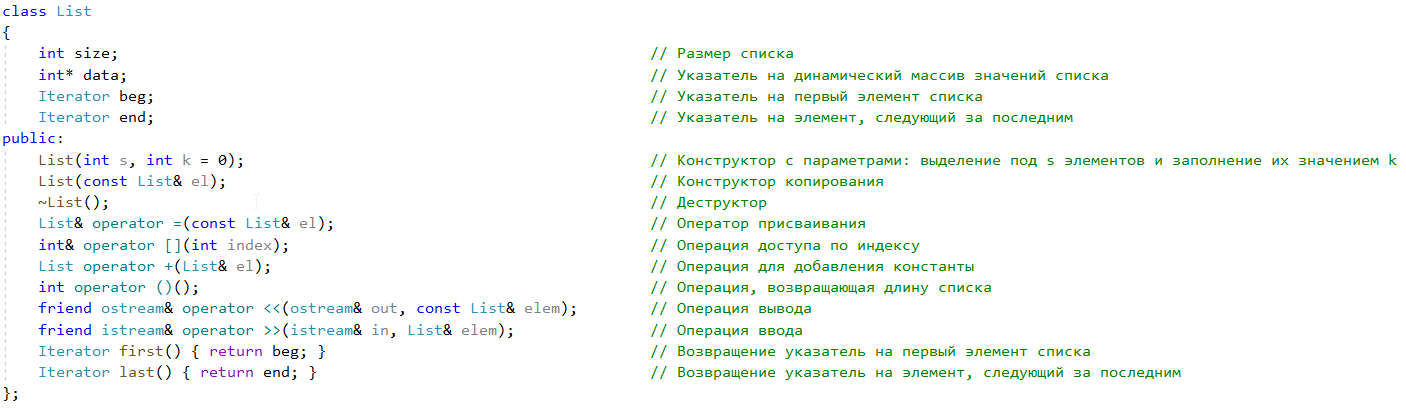
* 1. Реализовать определение класса Iterator и определение необходимых методов и операторов данного класса.

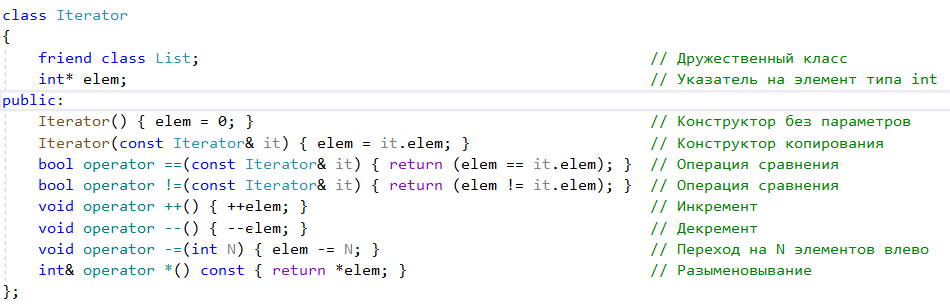


* 1. Реализовать применение этих функций в главной функции.



1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы с данными используются атрибуты класса.





1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных реализован с помощью оператора cin, используемых при реализации в главной функции и функций перегрузки оператора.

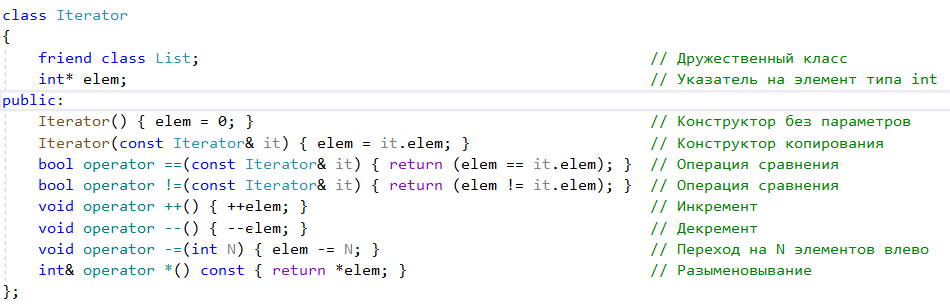
C:\Users\Асус\Documents\ShareX\Screenshots\2021-06\devenv_BZymUUkKKX.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-06\devenv_26Ri2oc5A8.png

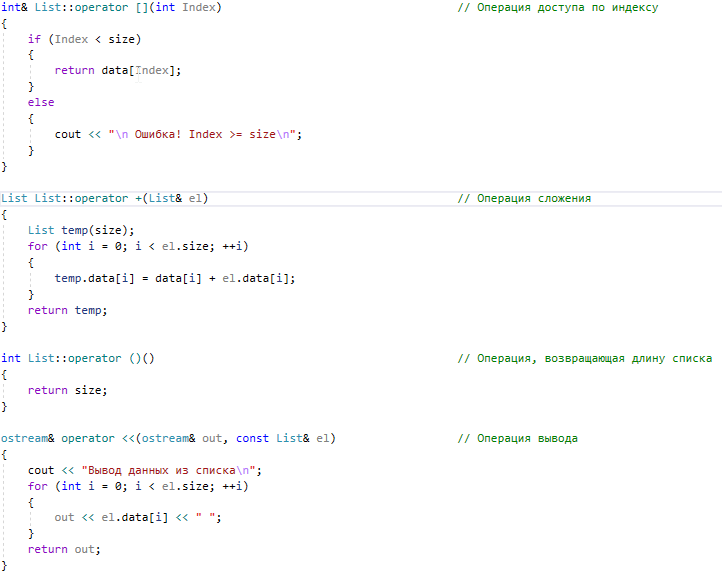
* 1. Вывод данных реализован с помощью оператора cout, используемого при реализации в главной функции и функций перегрузки операторов ввода-вывода.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-06\devenv_UUM8ED55Wv.png

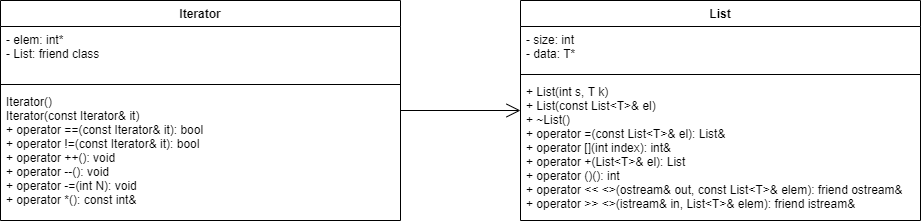
1. Поставленные задачи решены следующими действиями:
   1. В классе Iterator находится реализация итератора списка, операторов сравнения, инкремента и декремента и разыменовывания итератора.



* 1. В классе List находится реализация списка для данных типа int, определение размера списка, доступ элемента списка по индексу, сложение векторов.



**UML-Диаграмма**



**Реализация задачи на языке С++**

**Заголовочный файл List.h**

#pragma once

#include <iostream>

using namespace std;

class Iterator

{

friend class List;

int\* elem;

public:

Iterator() { elem = 0; }

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

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

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

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

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

void operator -=(int N) { elem -= N; }

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

};

class List

{

int size;

int\* data;

Iterator beg;

Iterator end;

public:

List(int s, int k = 0);

List(const List& el);

~List();

List& operator =(const List& el);

int& operator [](int index);

List operator +(List& el);

int operator ()();

friend ostream& operator <<(ostream& out, const List& elem);

friend istream& operator >>(istream& in, List& elem);

Iterator first() { return beg; }

Iterator last() { return end; }

};

**Файл с описанием методов класса List.cpp**

#include "List.h"

#include <iostream>

using namespace std;

List::List(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];

}

List::List(const List& el)

{

if (size < el.size)

{

size = el.size;

}

data = new int[size];

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

{

data[i] = el.data[i];

}

beg = el.beg;

end = el.end;

}

List::~List()

{

delete[] data;

data = 0;

}

List& List::operator =(const List& el)

{

if (this != &el)

{

if (size < el.size)

{

size = el.size;

}

if (data != 0)

{

delete[] data;

}

data = new int[size];

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

{

data[i] = 0;

}

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

{

data[i] = el.data[i];

}

}

beg = el.beg;

end = el.end;

return \*this;

}

int& List::operator [](int Index)

{

if (Index < size)

{

return data[Index];

}

else

{

cout << "\n Ошибка! Index >= size\n";

}

}

List List::operator +(List& el)

{

List temp(size);

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

{

temp.data[i] = data[i] + el.data[i];

}

return temp;

}

int List::operator ()()

{

return size;

}

ostream& operator <<(ostream& out, const List& el)

{

cout << "Вывод данных из списка\n";

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

{

out << el.data[i] << " ";

}

return out;

}

istream& operator >>(istream& in, List& el)

{

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

{

cout << "Введите данные " << i + 1 << " объекта в список\n";

in >> el.data[i];

}

return in;

}

**Файл с главной программой Лабораторная работа № 18.6.cpp**

#include "List.h"

#include <iostream>

using namespace std;

int main()

{

system("chcp 1251>nul");

List A(5);

cout << "Список А\n" << A << "\n";

cout << "Ввод данных в список А\n";

cin >> A;

cout << "Список А\n" << A << "\n";

cout << "Третий элемент списка становится равен 100\n";

A[2] = 100;

cout << "Список А\n" << A << "\n";

List B(10);

cout << "Список B\n" << B << "\n";

B = A;

cout << "Список B\n" << B << "\n";

List C(10);

C = B + A;

cout << "Список C\n" << C << "\n";

cout << "\nДлина списка А: " << A() << endl;

cout << "Первый элемент списка А: " << \*(A.first()) << endl;

Iterator i = A.first();

++i;

cout << "Следующий элемент списка A: " << \*i << endl;

i = A.last();

i -= 2;

cout << "Второй с конца элемент списка A: " << \*i << endl;

cout << "Вывод элементов списка A\n";

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

{

cout << \*i << " ";

}

cout << endl;

}

**Ответы на вопросы**

1. Множество элементов, тип элементов не важен
2. Сортировка массива, одной функцией можно сортировать разные массивы и один массив можно сортировать разными функциями
3. Добавление элемента в конец списка
4. Набор однотипных элементов
5. Доступ, удаление, добавление, изменение
6. Последовательный, прямой, ассоциативный
7. Указатель на элемент контейнера
8. Через класс с перегрузкой разыменования
9. Слияние, склеивание
10. Ассоциативный
11. Стек
12. D
13. D
14. Прямой
15. Последовательный