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 18.6**

Тема: Классы и объекты. АТД. Контейнеры

Выполнил работу

студент группы РИС-20-1б

Томилов В.А.

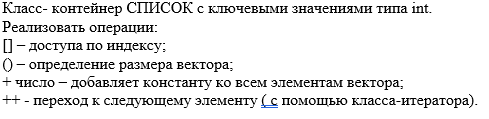
Проверила

Доцент кафедры ИТАС

к.т.н ПоляковаО.А.

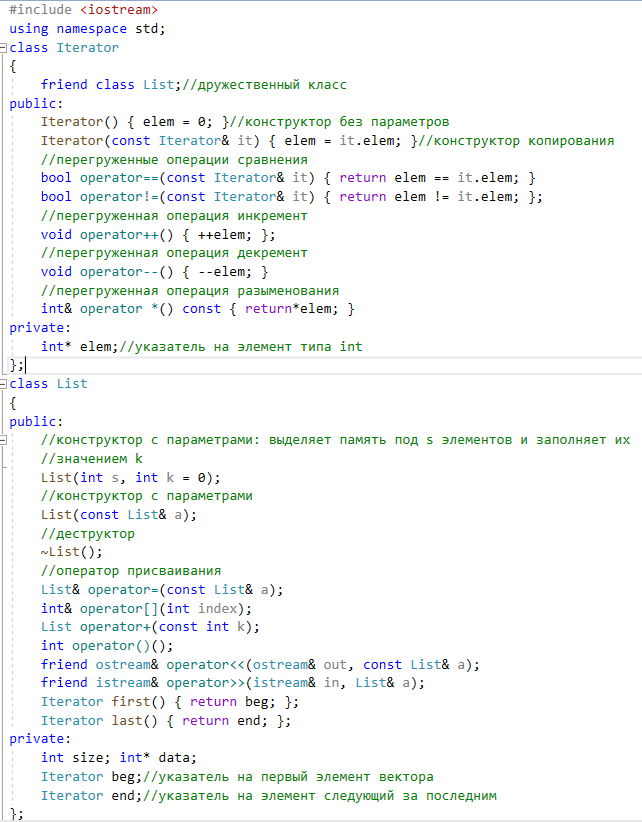
Пермь, 2021

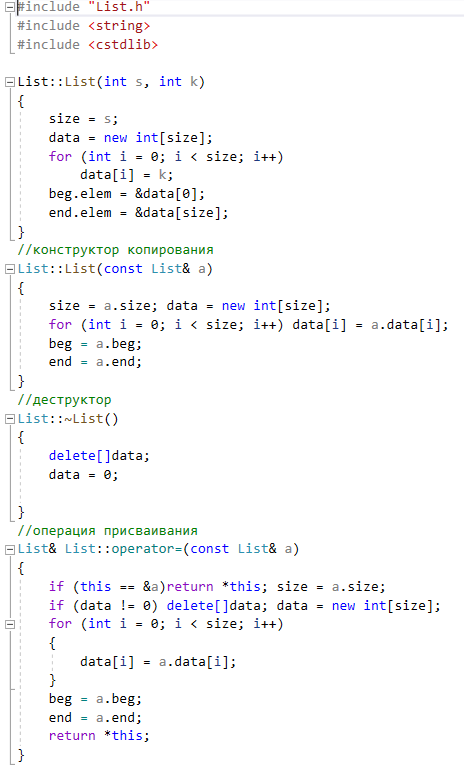
Постановка задачи

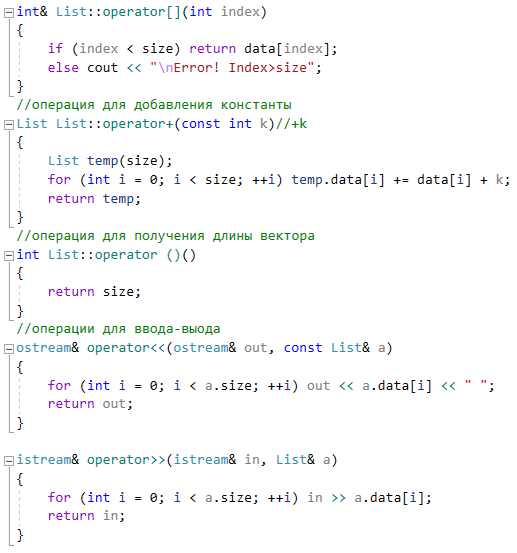


Анализ задачи

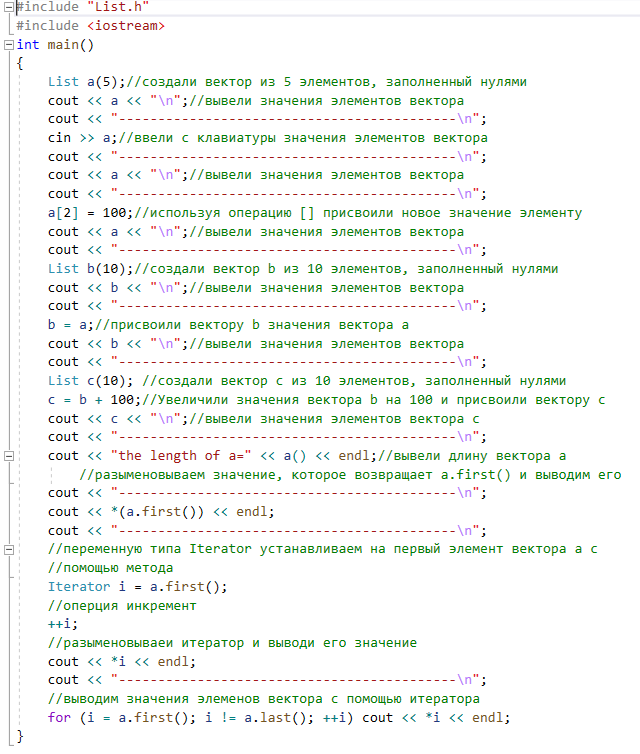
* 1. Для решения задачи необходимо
  2. Добавить в проект файл list.h, содержащий описание класса list и iterator.



* 1.  Добавим в проект файл list.cpp, содержащий описание методов класса list.



* 1. Добавить в проект файл Lab6\_main.cpp, содержащий основную программу.

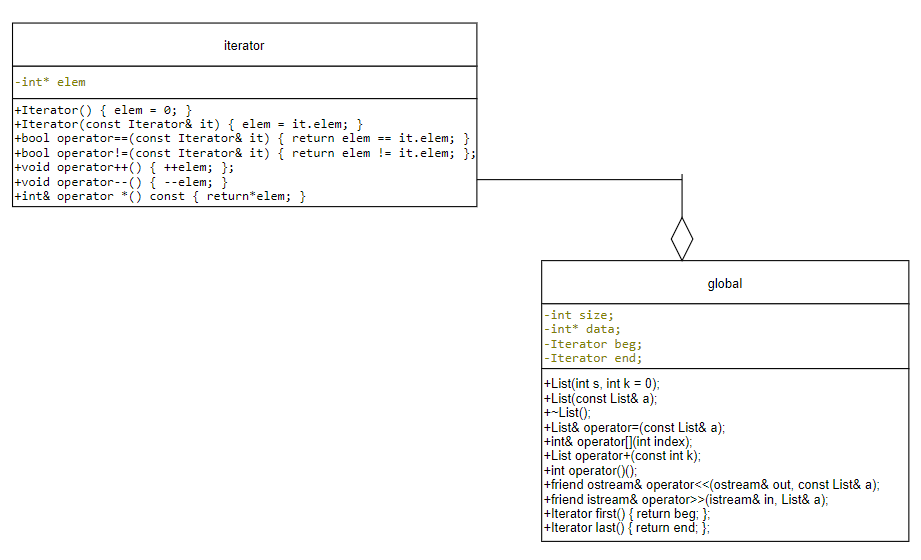


* 1. В ходе работы были использованы следующие типы данных:

2.1 (int\*) elem - указатель на элемент типа int

2.2 (int) size, data -размер списка, значения элемента списка

2.3 (Iterator) beg, end - указатель на первый элемент вектора и указатель на элемент следующий за последним.

UML – диаграмма

Код программы

Main.cpp

#include "List.h"

#include <iostream>

int main()

{

List a(5);//создали вектор из 5 элементов, заполненный нулями

cout << a << "\n";//вывели значения элементов вектора

cout << "-------------------------------------------\n";

cin >> a;//ввели с клавиатуры значения элементов вектора

cout << "-------------------------------------------\n";

cout << a << "\n";//вывели значения элементов вектора

cout << "-------------------------------------------\n";

a[2] = 100;//используя операцию [] присвоили новое значение элементу

cout << a << "\n";//вывели значения элементов вектора

cout << "-------------------------------------------\n";

List b(10);//создали вектор b из 10 элементов, заполненный нулями

cout << b << "\n";//вывели значения элементов вектора

cout << "-------------------------------------------\n";

b = a;//присвоили вектору b значения вектора a

cout << b << "\n";//вывели значения элементов вектора

cout << "-------------------------------------------\n";

List c(10); //создали вектор c из 10 элементов, заполненный нулями

c = b + 100;//Увеличили значения вектора b на 100 и присвоили вектору c

cout << c << "\n";//вывели значения элементов вектора c

cout << "-------------------------------------------\n";

cout << "the length of a=" << a() << endl;//вывели длину вектора a

//разыменовываем значение, которое возвращает a.first() и выводим его

cout << "-------------------------------------------\n";

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

cout << "-------------------------------------------\n";

//переменную типа Iterator устанавливаем на первый элемент вектора а с

//помощью метода

Iterator i = a.first();

//оперция инкремент

++i;

//разыменовываеи итератор и выводи его значение

cout << \*i << endl;

cout << "-------------------------------------------\n";

//выводим значения элеменов вектора с помощью итератора

for (i = a.first(); i != a.last(); ++i) cout << \*i << endl;

}

List.h

#include <iostream>

using namespace std;

class Iterator

{

friend class List;//дружественный класс

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; }

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

private:

int\* elem;//указатель на элемент типа int

};

class List

{

public:

//конструктор с параметрами: выделяет память под s элементов и заполняет их

//значением k

List(int s, int k = 0);

List(const List& a);

~List();

List& operator=(const List& a);

int& operator[](int index);

List operator+(const int k);

int operator()();

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

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

Iterator first() { return beg; };

Iterator last() { return end; };

private:

int size; int\* data;

Iterator beg;//указатель на первый элемент вектора

Iterator end;//указатель на элемент следующий за последним

};

List.cpp

#include "List.h"

#include <string>

#include <cstdlib>

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];

}

//конструктор копирования

List::List(const List& 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

List::~List()

{

delete[]data;

data = 0;

}

//операция присваивания

List& List::operator=(const List& 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;

}

//операция доступа по индексу

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

{

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

else cout << "\nError! Index>size";

}

//операция для добавления константы

List List::operator+(const int k)//+k

{

List temp(size);

for (int i = 0; i < size; ++i) temp.data[i] += data[i] + k;

return temp;

}

//операция для получения длины вектора

int List::operator ()()

{

return size;

}

//операции для ввода-выода

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

{

for (int i = 0; i < a.size; ++i) out << a.data[i] << " ";

return out;

}

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

{

for (int i = 0; i < a.size; ++i) in >> a.data[i];

return in;

}

Ответы на вопросы

1. АТД - тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов.
2. –
3. –
4. Контейнер – набор однотипных элементов.
5. Встроенные массивы в С++ - частный случай контейнера.
6. Доступ к элементам контейнера бывает: последовательный, прямой и ассоциативный (перейти к первому элементу, доступ по индексу, доступ по значению).
7. Итератор – это объект, который обеспечивает последовательный доступ к элементам контейнера
8. Итератор можно реализовать как класс, представляющий такой же набор операций

* Простое сцепление двух контейнеров: в новый контейнер попадают сначала элементы первого контейнера, потом второго, операция не коммутативна.
* Объединение упорядоченных контейнеров, новый контейнер тоже будет упорядочен, операция коммутативна.
* Объединение контейнеров как объединение множеств, в новый контейнер попадают только те элементы, которые есть хотя бы в одном контейнере, операция коммутативна.
* Объединение контейнеров как пересечение множеств, в новый контейнер попадают только те элементы, которые есть в обоих контейнерах, операция коммутативна.
* Для контейнеров-множеств может быть еще реализована операция вычитания, в контейнер попадают только те элементы первого контейнера, которых нет во втором, операция не коммутативна.
* Извлечение части элементов из контейнера и создание нового контейнера. Эта операция может быть выполнена с помощью конструктора, а часть контейнера задается двумя итераторами.

1. Ассоциативный
2. Стек
3. C.
4. D.
5. Прямой
6. Ассоциативный