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**

Дисциплина: Основы теории алгоритмов и структуры данных.

Тема: “ Объектно-ориентированное программирование.

АДФ. Контейнеры.”

**Вар.15**

Выполнила:

студент группы ИВТ-20-1б

Лыкова Анна Викторовна

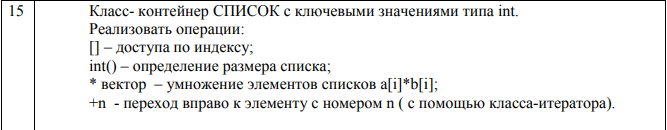
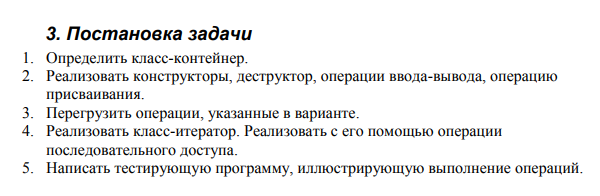
Проверил:

Ст. пр. кафедры ИТАС

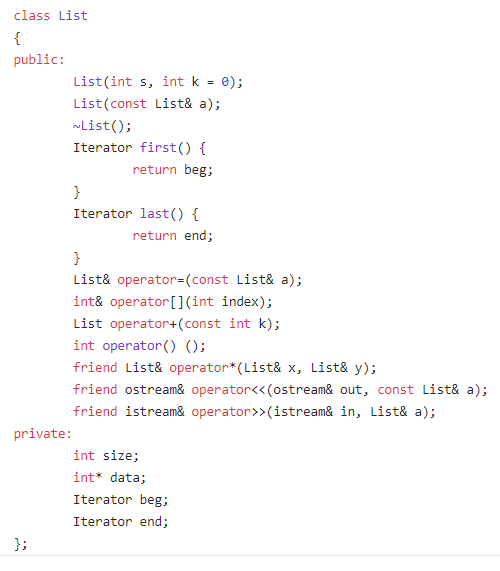
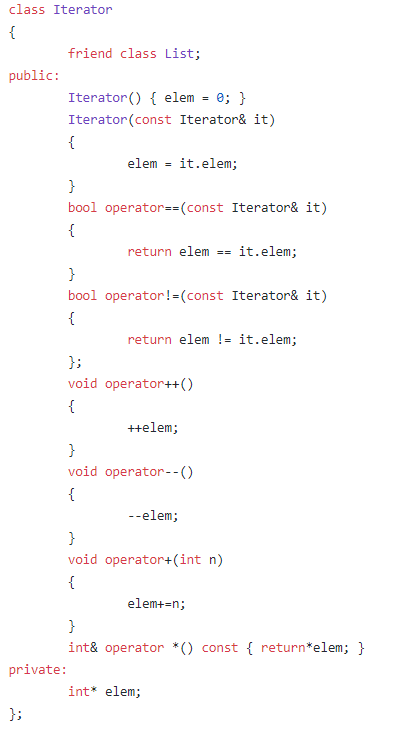
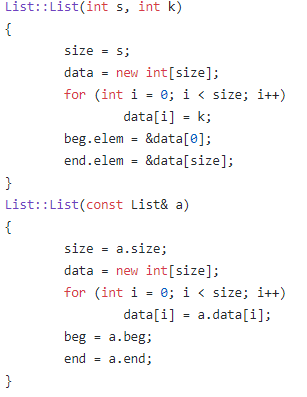
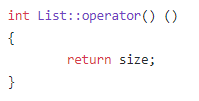
к.т.н Яруллин Д. В.

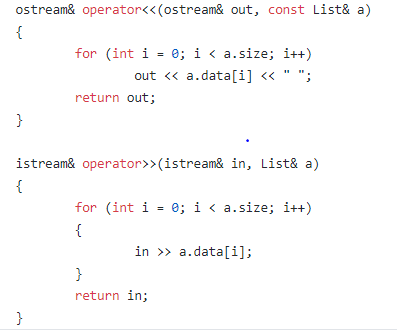
Пермь, 2021

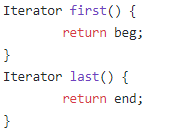
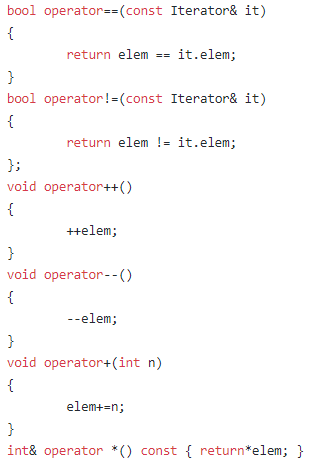
**Постановка задачи**



**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать класс List.
   2. Организовать класс Iterator, дружественный классу List.
   3. Организовать перегрузку оператора [] для класса List.
   4. Организовать перегрузку оператора <<, дружественной классу List.
   5. Организовать перегрузку оператора ++ для класса Iterator.
   6. Организовать перегрузку оператора == для класса Iterator.
   7. Организовать перегрузку оператора != для класса Iterator.
   8. Организовать перегрузку оператора \* для класса Iterator.
2. В ходе работы были использованы следующие типы данных:
   1. Для класса list описаны следующие поля:  
      
   2. Для класса Iterator:  
      
3. Для решения задачи данные были представлены в следующем виде:
   1. Для инициализации объектов классов Iterator и List используются конструкторы с параметрами и по умолчанию: 
4. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Для заполнения и вывода списка перегружены соответствующие операторы:



1. Поставленные задачи будут решены следующими действиями:
   1. В классе List реализованы геттеры, которые возвращают размер списка, указатели на первый и последний элемент списка.  
      
   2. В классе Iterator реализованы требуемые перегрузки операторов:  
      

**Код программы**

**Файл Main.cpp:**

#include <iostream>

#include "list.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

List a(5);

cout << "Создание списка а" << endl;

cout << a << "\n";

cin >> a;

cout << " Заполненный список а" << endl;

cout << a << "\n";

a[2] = 100;

cout << "После обращению к эл по индеку и переприсваивания в списке а" << endl;

cout << a << "\n";

cout << "Разыменовывание указателя на 1 эл" << endl;

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

Iterator i = a.first();

int n;

cout << "Номер элемента списка: ";

cin >> n;

i+n;

cout << "Переход к n-ому элементу списка а" << endl;

cout << \*i << endl;

cout << "Вывод значений списка а с помощью итератора и разыменовывания указателя" << endl;

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

cout << \*i << " ";

cout << endl;

List b(10);

cout << "Создание списка b" << endl;

cout << b << "\n";

b = a;

cout << "Присваивание списку b список а" << endl;

cout << b << "\n";

cout << "Определение размера списка b: " << b() << endl;

List c(10);

c = a \* b;

cout << "Произведение a и b" << endl;

cout << c << "\n";

}

**Файл List.h:**

#pragma once

#include <iostream>

using namespace std;

class Iterator

{

friend class List;

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; }

private:

int\* elem;

};

class List

{

public:

List(int s, int k = 0);

List(const List& a);

~List();

Iterator first() {

return beg;

}

Iterator last() {

return end;

}

List& operator=(const List& a);

int& operator[](int index);

List operator+(const int k);

int operator() ();

friend List& operator\*(List& x, List& y);

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

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

private:

int size;

int\* data;

Iterator beg;

Iterator end;

};

**Файл List.cpp:**

#include "list.h"

#include <iostream>

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& 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;

}

List::~List()

{

delete[]data;

data = 0;

}

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;

}

int List::operator() ()

{

return size;

}

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

{

if (index < size)

return data[index];

else

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

}

List& operator\*(List& x, List& y) {

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

y.data[i] = x.data[i] \* y.data[i];

return y;

}

List List::operator+(const int k)

{

List temp(size);

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

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

return temp;

}

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++)

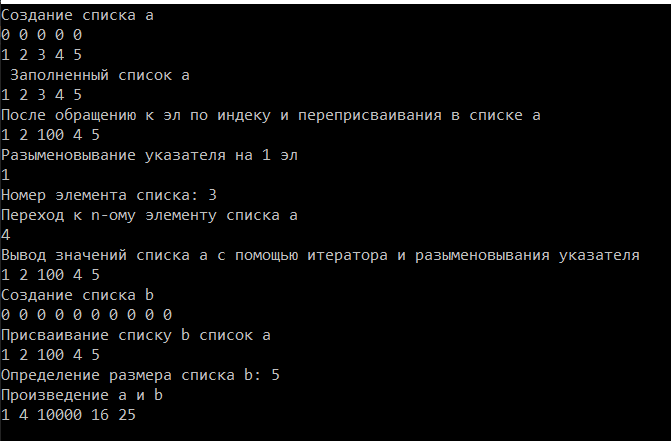
{

in >> a.data[i];

}

return in;

}

**Работа кода**

**Ответы на вопросы**

