Лаба 29 UML – диаграмма

**Код программы :**

/\*Класс- контейнер МНОЖЕСТВО с элементами типа int.

Реализовать операции: [] – доступа по индексу;

!= - проверка на неравенство;

< число – принадлежность числа множеству;

+ n – переход вправо к элементу с номером n ( с помощью класса-итератора).\*/

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* pNext;

};

class List // Определяем односвязный список

{

private:

Node\* pFirst;

public:

List() : pFirst(NULL) {}

// метод добавления записи

List& push\_front(int);

// Перегрузка операции обращения по индексу

int& operator [] (int);

// Объявление внутреннего класса-итератора

class Iterator

{

protected:

Node\* iter;

public:

Iterator(Node\* node) : iter(node) {}

// перегрузка инкрементов

Iterator& operator ++ ();

Iterator operator ++ (int);

// перегрузка сравнения итераторов

bool operator != (Iterator right);

bool operator < (Iterator right);

// получение ссылки на запись

int& get();

};

// Получение итераторов

Iterator start();

Iterator end();

Iterator n(int n1);

};

List& List::push\_front(int data)

{

Node\* temp = new Node;

temp->data = data;

temp->pNext = pFirst;

pFirst = temp;

return \*this;

}

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

{

Node\* temp = pFirst;

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

{

temp = temp->pNext;

}

return temp->data;

}

List::Iterator& List::Iterator::operator ++ ()

{

iter = iter->pNext;

return \*this;

}

List::Iterator List::Iterator::operator ++ (int)

{

List::Iterator temp(iter);

iter = iter->pNext;

return temp;

}

int& List::Iterator::get()

{

return iter->data;

}

bool List::Iterator::operator != (Iterator right)

{

return bool(iter != right.iter);

}

List::Iterator List::start()

{

return List::Iterator(pFirst);

}

List::Iterator List::end()

{

return List::Iterator(NULL);

}

List::Iterator List::n(int n1)

{

Node\*tmp = pFirst;

for (int i = 0; i < n1; i++)

{

tmp = tmp->pNext;

}

cout << tmp->data;

return 0;

}

bool List::Iterator::operator < (Iterator right)

{

return bool(iter < right.iter);

}

int main()

{

setlocale(LC\_ALL,"Russian");

List lost\_of;

int n1;

lost\_of.push\_front(87).push\_front(10).push\_front(25).push\_front(46).push\_front(101);

cout << "Односвязный список имеет элементы:" << endl;

for (List::Iterator iter = lost\_of.start(); iter != lost\_of.end(); iter++)

{

cout << iter.get() << "\t";

}

cout << endl << endl;

cout << "Операция доступа по индексу 2" << endl;

cout << lost\_of[2];

cout << endl << endl;

cout << "Операция принадлежность числа множеству сотни" << endl;

cout << "Взяли элемент под номером 0" << endl;

cout << lost\_of[0] << endl << endl;

if (lost\_of[0] < 100)

{

cout << "Число входит в множество меньше сотни" << endl;

}

else

{

cout << "Число не входит в множество меньше сотни" << endl;

}

cout << endl << endl;

cout << "Введите переход к какому элементу сделать" << endl;

cin >> n1;

cout << endl;

while (n1 < 0 || n1>4)

{

cout << "Неправильный ввод, введите заного" << endl;

cin >> n1;

cout << endl;

}

cout << "Мы перешли к элементу " << n1 << endl;

List::Iterator iter = lost\_of.n(n1);

cout << endl << endl;

}

**UML – диаграмма этого кода:**

