Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа по ООП**

**«№6»**

Выполнил:

студент группы РИС-23-2б

Ившин Максим Сергеевич

Проверила:

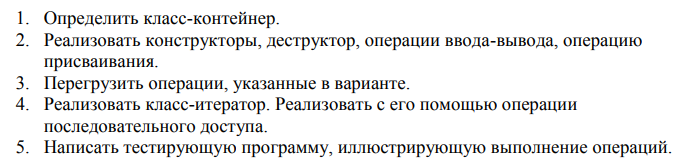
доцент кафедры ИТАС

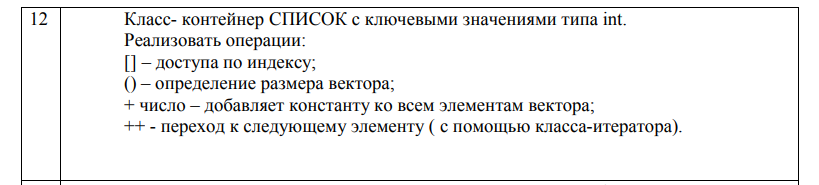
О.А. Полякова

2024 г.

**Разработка алгоритма**

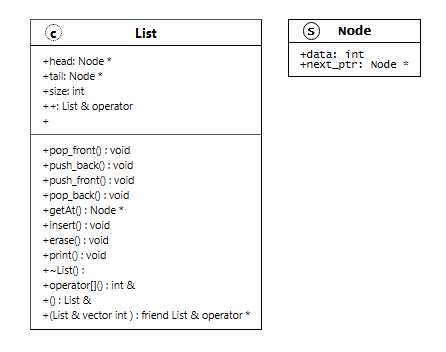
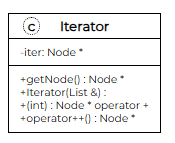
**Постановка задачи:**





**Анализ задачи:**

1. Создадим несколько файлов заголовков и файлов с кодом для удобной работы.
2. Создадим класс-контейнер список List, который будет хранить указатели на головной элемент и конечный элемент списка.
3. Список будет использовать дополнительную структуру Node для реализации узлов списка.
4. Для удобного прохода по списку реализуем итератор Iterator, оперирующий указателями на узлы списка.
5. В главном файле программы создадим список и продемонстрируем все реализованные операции.



Код на C++:

main.cpp:

#include <iostream>

#include <vector>

#include "list.h"

#include "list.cpp"

#include "iterator.h"

#include "iterator.cpp"

using namespace std;

int main()

{

     List list;

     for (int i = 0; i < 5; ++i) // заполнение списка

          list.push\_back(rand() % 100);

     list.print();

     cout << list[2] << endl;

     list(10);

     list += 10;

     list.print();

}

iterator.h:

#pragma once

using namespace std;

#include "List.h"

class Iterator

{

private:

    Node \*iter;

public:

    Node \*getNode();

    Iterator(List &a);

    Node \*operator+(int a);

    Node \*operator++();

    Node \*operator++(int);

};

list.h:

#pragma once

#include <iostream>

#include <vector>

using namespace std;

struct Node

{

    int data;

    Node \*next\_ptr = nullptr;

};

class List

{

public:

    Node \*head = nullptr;

    Node \*tail = nullptr;

    int size = 0;

    void pop\_front();

    void push\_back(int);

    void push\_front(int);

    void pop\_back();

    Node \*getAt(int);

    void insert(int, int);

    void erase(int);

    void print();

    ~List();

    int &operator[](int);

    List &operator()(int);

    List &operator+=(int);

    friend List &operator\*(List &a, vector<int> b);

};

list.cpp:

#pragma once

#include <iostream>

#include <vector>

#include "list.h"

#include "iterator.h"

using namespace std;

void List::pop\_front()

{

    if (head == nullptr)

        return;

    if (head == tail)

    {

        delete tail;

        head = tail = nullptr;

        return;

    }

    Node \*node = head;

    head = node->next\_ptr;

    delete node;

    --size;

}

void List::push\_back(int data)

{

    Node \*node = new Node;

    node->data = data;

    if (head == nullptr)

        head = node;

    if (tail != nullptr)

        tail->next\_ptr = node;

    tail = node;

    ++size;

}

void List::push\_front(int data)

{

    Node \*node = new Node;

    node->data = data;

    node->next\_ptr = head;

    head = node;

    if (tail == nullptr)

        tail = node;

    ++size;

}

void List::pop\_back()

{

    if (tail == nullptr)

        return;

    if (head == tail)

    {

        delete tail;

        head = tail = nullptr;

        return;

    }

    Node \*node = head;

    for (; node->next\_ptr != tail; node = node->next\_ptr)

        ;

    node->next\_ptr = nullptr;

    delete tail;

    tail = node;

    --size;

}

Node \*List::getAt(int k)

{

    if (k < 0)

        return 0;

    Node \*node = head;

    int n = 0;

    while (node && n != k && node->next\_ptr)

    {

        node = node->next\_ptr;

        n++;

    }

    if (n == k)

        return node;

    else

        return nullptr;

}

void List::insert(int k, int data)

{

    Node \*left = getAt(k);

    if (left == nullptr)

        return;

    Node \*right = left->next\_ptr;

    Node \*node = new Node;

    node->data = data;

    left->next\_ptr = node;

    node->next\_ptr = right;

    if (right == nullptr)

        tail = node;

    size++;

}

void List::erase(int k)

{

    if (k < 0)

        return;

    if (k == 0)

    {

        pop\_front();

        return;

    }

    Node \*left = getAt(k - 1);

    Node \*node = left->next\_ptr;

    if (node == nullptr)

        return;

    Node \*right = node->next\_ptr;

    left->next\_ptr = right;

    if (node == tail)

        tail = left;

    delete node;

    --size;

}

void List::print()

{

    for (Iterator iterator(\*this); iterator.getNode() != nullptr; iterator++) // вывод

        cout << iterator.getNode()->data << ' ';

    cout << endl

         << endl;

}

List &List::operator+=(int k)

{

    for (Iterator iterator(\*this); iterator.getNode() != nullptr; iterator++) // вывод

        iterator.getNode()->data += k;

    return \*this;

}

List::~List()

{

    while (head != nullptr)

        pop\_front();

}

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

{

    return getAt(a)->data;

}

List &List::operator()(int k)

{

    while (size > k)

    {

        this->pop\_back();

        size--;

    }

    while (size < k)

    {

        this->push\_back(0);

    }

    return \*this;

}

iterator.cpp:

#pragma once

#include <iostream>

#include <string>

#include "iterator.h"

#include "list.h"

using namespace std;

Node \*Iterator::getNode()

{

    return this->iter;

}

Iterator::Iterator(List &a)

{

    this->iter = a.head;

}

Node \*Iterator::operator+(int a)

{

    Node \*iter = this->iter;

    for (int i = 0; i < a; ++i)

        iter = iter->next\_ptr;

    return iter;

}

Node \*Iterator::operator++()

{

    iter = iter->next\_ptr;

    return iter;

}

Node \*Iterator::operator++(int)

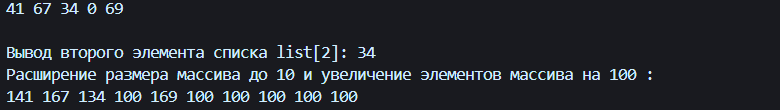
{

    iter = iter->next\_ptr;

    return iter;

}

Пример работы программы:



**GitHub:** *https://github.com/geroineee/PNRPU*