#include<iostream>

using namespace std;

template<typename T>

class List {

public:

List();

~List();

void push\_back(T data);

int getsize() { return size; }

T& operator[](const int index);

void pop\_front();

void push\_front(T data);

void insert(T value,const int index);

void removeAt(const int index);

void pop\_back();

void clear();

private:

template<typename T>

class Node {

public:

T data;

Node\* nextptr;

Node(T data=T(),Node\*nextptr=nullptr) {

this->data = data;

this->nextptr = nextptr;

}

};

Node<T>\* head;

int size;

};

template<typename T>

List<T>::List() {

head = nullptr;

size = 0;

}

template<typename T>

List<T>::~List()

{

clear();

}

template<typename T>

void List<T>::push\_back(T data)

{

if (head == nullptr)

head = new Node<T>(data);

else {

Node<T>\* current = this->head;

while (current->nextptr != nullptr)

current = current->nextptr;

current->nextptr = new Node<T>(data);

}

size++;

}

template<typename T>

T& List<T>::operator[](const int index)

{

int counter = 0;

Node<T>\* current = this->head;

while (current != nullptr) {

if (counter == index)

return current->data;

current = current->nextptr;

counter++;

}

}

template<typename T>

void List<T>::pop\_front()

{

Node<T>\* temp = head;

head = head->nextptr;

delete temp;

size--;

}

template<typename T>

void List<T>::push\_front(T data)

{

head = new Node<T>(data, head);

size++;

}

template<typename T>

void List<T>::insert(T value, const int index)

{

if (index == 0)

push\_front(value);

else {

Node<T>\* previous =this-> head;

for (int i = 0; i < index - 1; i++)

previous = previous->nextptr;

previous->nextptr = new Node<T>(value, previous->nextptr);

size++;

}

}

template<typename T>

void List<T>::removeAt(const int index)

{

if (index == 0)

pop\_front();

else {

Node<T>\* previous = this->head;

for (int i = 0; i < index - 1; i++)

previous = previous->nextptr;

Node<T> \*ToDelete = previous->nextptr;

previous->nextptr= ToDelete->nextptr;

delete ToDelete;

size--;

}

}

template<typename T>

void List<T>::pop\_back()

{

removeAt(size - 1);

}

template<typename T>

void List<T>::clear()

{

while (size) {

pop\_front();

}

}

int main() {

setlocale(LC\_ALL, "RUSSIAN");

List<int> list;

/\*int N;

cin >> N;

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

list.push\_back(rand() % 21);

cout << endl << "N=" << list.getsize();

for(int i=0;i<list.getsize(); i++)

cout << list[i]<<endl;

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

//list.pop\_front();k

cout << endl << "N=" << list.getsize() << endl;

for (int i = 0; i < list.getsize(); i++)

cout << list[i] << endl;\*/

list.push\_front(5);

list.push\_front(32);

list.push\_front(311);

list.push\_front(211);

cout << "\nn=" << list.getsize() << endl;

for (int i = 0; i < list.getsize(); i++)

cout << list[i] << endl;

cout << endl;

list.removeAt(2);

cout << "\nn=" << list.getsize() << endl;

for (int i = 0; i < list.getsize(); i++)

cout << list[i] << endl;

system("pause");

return 0;

}