Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: выполнение лабораторной работы 11

Выполнил студент ИВТ-22-2б:

Бобриков Михаил Александрович

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

1. Написать функцию для создания однонаправленного списка. Записи в линейном списке содержат ключевое поле типа int. Функция может создавать пустой список, а затем добавлять в него элементы.
2. Написать функцию для печати списка. Функция должна предусматривать вывод сообщения, если список пустой.
3. Написать функции для удаления из списка К элементов, начиная с заданного номера, и добавления элемента списка перед элементом с заданным ключом.
4. Выполнить изменения в списке и печать списка после каждого изменения.
5. Написать функцию для записи списка в файл.
6. Написать функцию для уничтожения списка.
7. Записать список в файл, уничтожить его и выполнить печать (при печати должно быть выдано сообщение "Список пустой").
8. Написать функцию для восстановления списка из файла.
9. Восстановить список и распечатать его.
10. Уничтожить список.

**Алгоритм решения (однонаправленный список)**

1. Создание списка. Пользователь вводит число узлов в списке, в цикле создаются узлы с шагом 1, узлы добавляются в конец списка.

2. Печать списка. Если передаваемый в функцию печати указатель на 1-ый элемент списка (head) пуст (= nullptr), то выводится сообщение «Список пуст». Иначе в цикле выводится на экран ключевое поле списка, цикл заканчивается, когда указатель на следующий элемент списка пуст (= nullptr).

3. Удаление из списка **К** элементов, начиная с заданного номера. Пользователь вводит номер первого удаляемого элемента и количество удаляемых элементов. В цикле указатель устанавливается на элемент с номером, предшествующим введенному пользователем. Затем во вложенном цикле (повторяющемся **К** раз) указатель **next** устанавливается на элемент, следующий за следующим (**next-> next**), а следующий элемент удаляется.

4. Добавление элемента списка перед элементом с заданным ключом. Пользователь вводит ключ элемента, перед которым надо добавить новый элемент. В цикле в списке ищется элемент с заданным ключом. Если он найден, то пользователь вводит значение нового элемента. У предыдущего элемента указатель устанавливается на новый элемент, а у нового элемента указатель устанавливается на следующий элемент.

5. Запись списка в файл. Открывается файл в режиме записи. В цикле до конца списка каждый элемент списка записывается в файл. Файл закрывается.

6. Уничтожение списка. В цикле до конца списка указатель головы списка устанавливается на следующий элемент, а текущий элемент удаляется.

7. Восстановление списка из файла. Файл открывается для чтения. Если файл не открылся, выдается сообщение «Файл не существует», работы функции завершается. Иначе, считаем количество структур (узлов) в файле. В цикле до конца файла считываем по одной структуре во временную переменную. Затем из этой переменной создаем узел списка.

**Алгоритм решения (двунаправленный список)**

1. Создание списка. Пользователь вводит число узлов в списке, в цикле создаются узлы с шагом 1, узлы добавляются в конец списка.

2. Печать списка. Если передаваемый в функцию печати указатель на 1-ый элемент списка (head) пуст (= nullptr), то выводится сообщение «Список пуст». Иначе в цикле выводится на экран ключевое поле списка, цикл заканчивается, когда указатель на следующий элемент списка пуст (= nullptr).

3. Удаление из списка **К** элементов, начиная с заданного номера. Пользователь вводит номер первого удаляемого элемента и количество удаляемых элементов. В цикле указатель устанавливается на элемент с заданным номером. Если предыдущий элемент существует, то в нем указатель **next** устанавливается на элемент, следующий за текущим, а в следующем элементе указатель **prev** устанавливается на элемент, предшествующий текущему.

4. Добавление элемента списка перед элементом с заданным ключом. Пользователь вводит ключ элемента, перед которым надо добавить новый элемент. В цикле в списке ищется элемент с заданным ключом. Если он найден, то пользователь вводит значение нового элемента. У предыдущего элемента указатель устанавливается на новый элемент, а у нового элемента указатель устанавливается на следующий элемент.

**Алгоритм решения (стек)**

1. Создание стека. Пользователь вводит число узлов в стеке, в цикле создаются узлы с шагом 1, узлы добавляются в вершину стека.

2. Печать стека. Если передаваемый в функцию печати указатель на верхний элемент стека (head) пуст (= nullptr), то выводится сообщение «Список пуст». Иначе в цикле выводится на экран ключевое поле списка, цикл заканчивается, когда указатель на следующий элемент списка пуст (= nullptr).

3. Удаление из стека **К** элементов, начиная с заданного номера. Пользователь вводит номер первого удаляемого элемента и количество удаляемых элементов. Создается второй вспомогательный стек. Из 1-го стека выталкиваются и во 2-ой стек помещаются элементы с номерами больше, чем последний удаляемый. Затем из 1-го стека выталкивается **К** удаляемых элементов. Затем все элементы из 2-го стека выталкиваются и помещаются обратно в 1-ый стек. 2-ой стек удаляется.

4. Добавление элемента в стек перед элементом с заданным ключом. Пользователь вводит ключ элемента, перед которым надо добавить новый элемент. Создается второй вспомогательный стек. В цикле значение вершины стека сравнивается с заданным ключом. Если не совпало, то вершина стека выталкивается из 1-го стека и помещается во 2-ой.

Если совпало, то вершина стека выталкивается из 1-го стека и помещается во 2-ой, новый элемент помещается в 1-ый стек, флагу присваивается значение 0, и происходит выход из цикла. Затем все элементы из 2-го стека выталкиваются и помещаются обратно в 1-ый стек. 2-ой стек удаляется.

После этого, если флаг остался со значением 1, то выводится сообщение, что элемент с заданным ключом не найден.

**Алгоритм решения (очередь)**

1. Создание очереди. Пользователь вводит число узлов в очереди, в цикле создаются узлы с шагом 1, узлы добавляются в конец очереди.

2. Печать очереди. Если передаваемый в функцию печати указатель на первый элемент очереди (head) пуст (= nullptr), то выводится сообщение «Список пуст». Иначе в цикле выводится на экран ключевое поле списка, цикл заканчивается, когда указатель на следующий элемент списка пуст (= nullptr).

3. Удаление из очереди **К** элементов, начиная с заданного номера. Пользователь вводит номер первого удаляемого элемента и количество удаляемых элементов. В цикле элементы с номерами меньше заданного удаляются из начала очереди и добавляются в конец очереди. Затем в цикле из начала очереди выталкивается **К** удаляемых элементов. Затем в цикле элементы после последнего удаленного до конца очереди удаляются из начала очереди и добавляются в конец очереди (чтобы голова очереди снова оказалась на 1-ом месте).

4. Добавление элемента в очередь перед элементом с заданным ключом. Пользователь вводит ключ элемента, перед которым надо добавить новый элемент. В цикле значение начала очереди сравнивается с заданным ключом. Если не совпало, то текущий элемент выталкивается из начала очереди и добавляется в конец очереди.

Если совпало, то новый элемент помещается в конец очереди, текущий элемент выталкивается из начала очереди и добавляется в конец очереди, флагу присваивается значение 0. Цикл продолжается, пока голова очереди не окажется снова на 1-ом месте.

После этого, если флаг остался со значением 1, то выводится сообщение, что элемент с заданным ключом не найден.

**Блок-схема (однонаправленный список)**

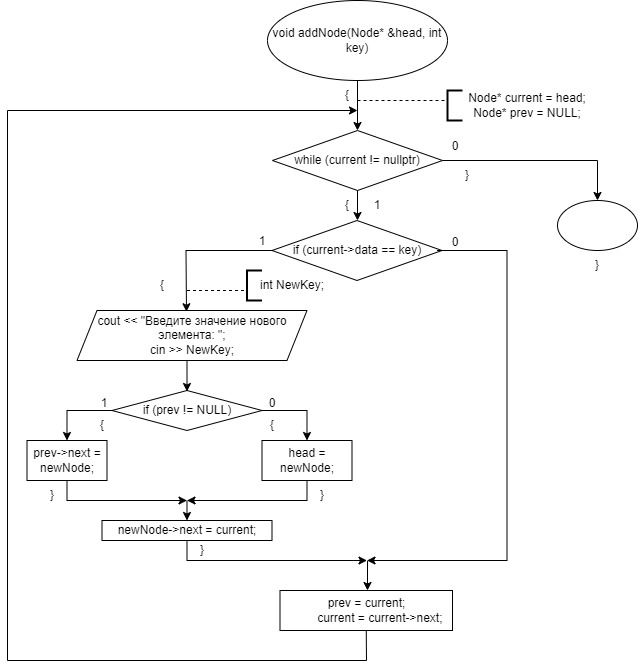
****

Рис. 1.Добавление элемента списка перед элементом с заданным ключом.

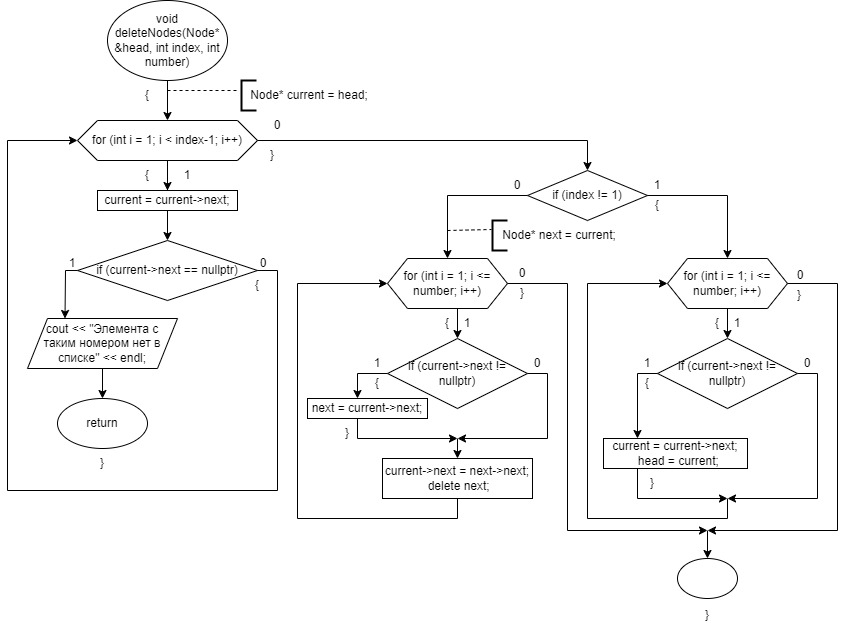
****

Рис. 2. Удаление из списка элементов, начиная с заданного номера.

**Блок-схема (двунаправленный список)**

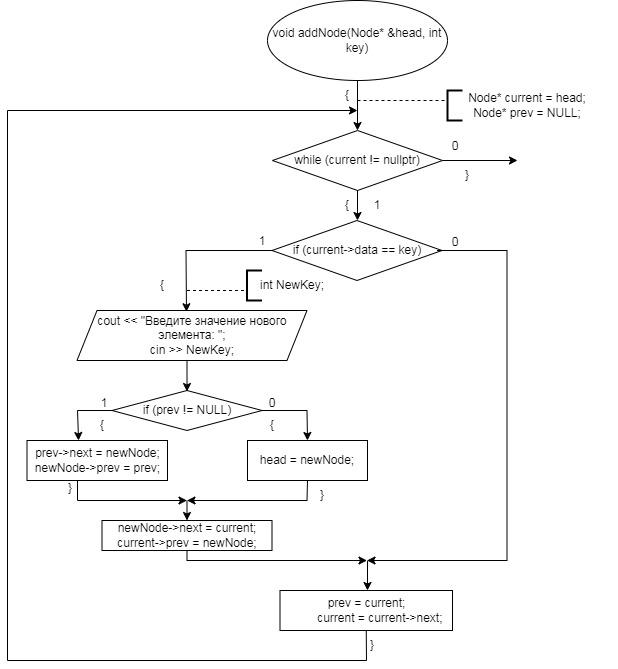


Рис. 3.Добавление элемента списка перед элементом с заданным ключом.

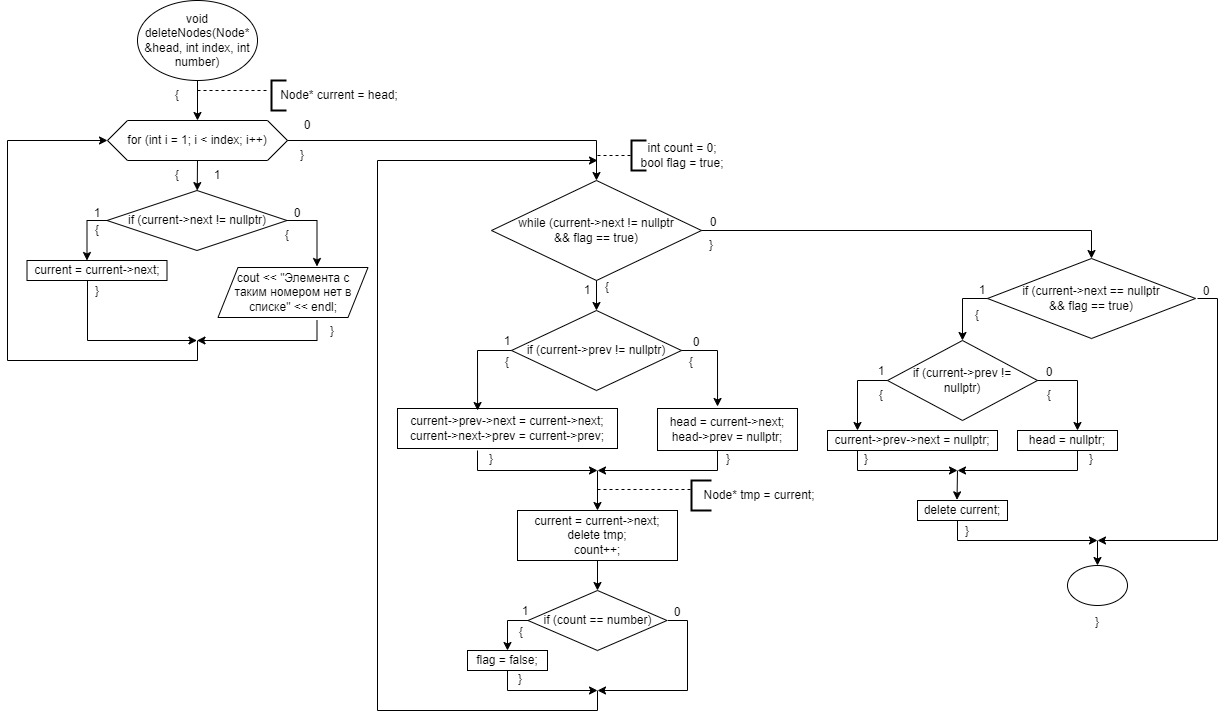


Рис. 4. Удаление из списка элементов, начиная с заданного номера.

**Блок-схема (стек)**

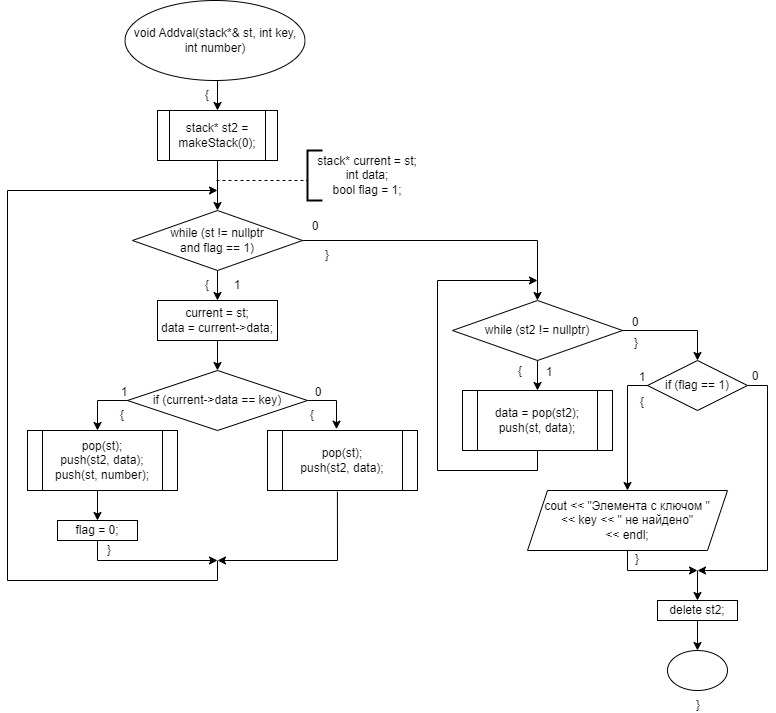
****

Рис. 5.Добавление элемента в стек перед элементом с заданным ключом.

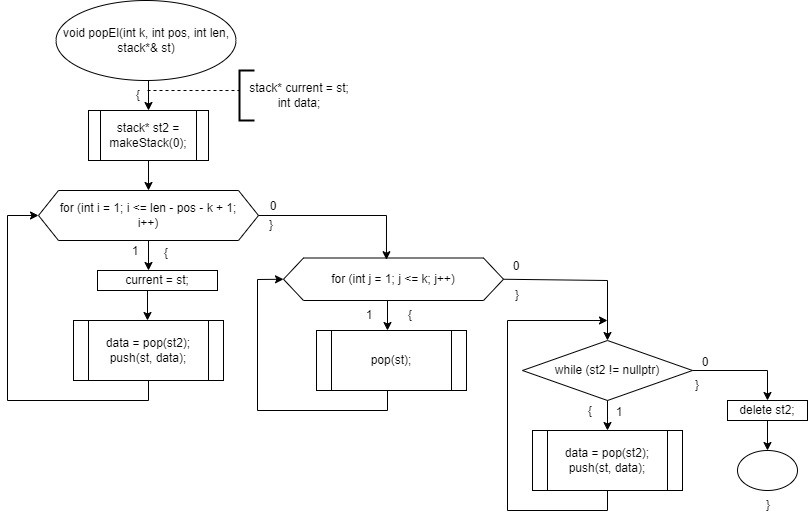
****

Рис. 6. Удаление из стека элементов, начиная с заданного номера.

**Блок-схема (очередь)**

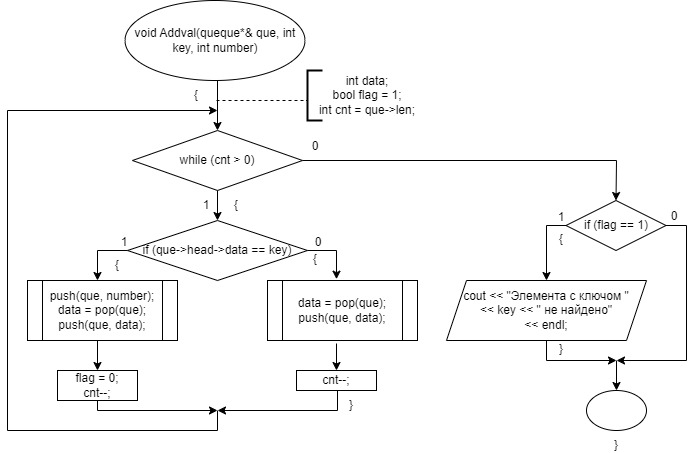
****

Рис. 7.Добавление элемента в очередь перед элементом с заданным ключом.

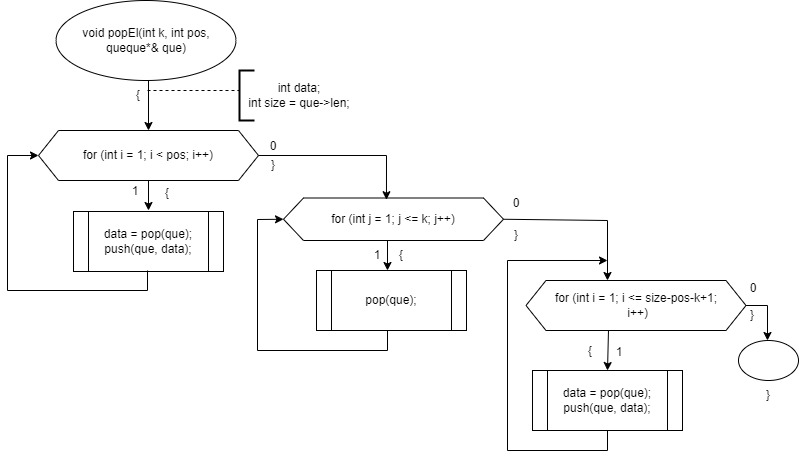
****

Рис. 8. Удаление из очереди элементов, начиная с заданного номера.

**Код программы (однонаправленный список)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <cstring>

#include <windows.h>

using namespace std;

// Структура узла списка

struct Node {

int data; // Данные узла - строка символов

Node\* next; // Указатель на следующий элемент

};

// Функция для создания узла списка

Node\* createNode(int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = nullptr;

return newNode;

}

// Функция для создания списка

Node\* createList() {

Node\* head = nullptr;

Node\* tail = nullptr;

int n;

cout << "Введите число узлов: ";

cin >> n;

for (int i = 1; i <= n; i++) {

//Вариант 1: ручное заполнение списка

//int data;

//cout << "Введите значение узла " << i << ": ";

//cin >> data;

//Node\* newNode = createNode(data); // значение для элемента вводится с клав-ры

//Вариант 2: автоматическое заполнение списка

Node\* newNode = createNode(i); // значение для элемента берется из i

if (tail != nullptr) {

tail->next = newNode;

}

else {

head = newNode;

}

tail = newNode;

}

return head;

}

// Функция для печати списка

void printList(Node\* head) {

if (head == nullptr) {

cout << "Список пуст" << endl;

return;

}

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функция для удаления заданного количества элементов, начиная с заданного номера

void deleteNodes(Node\* &head, int index, int number) {

Node\* current = head;

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

current = current->next;

if (current->next == nullptr) {

cout << "Элемента с таким номером нет в списке" << endl;

return;

}

}

if (index != 1) { // Если заданный номер не равен 1

Node\* next = current;

for (int i = 1; i <= number; i++) {

if (current->next != nullptr) {

next = current->next;

}

current->next = next->next;

delete next;

}

}

else {

for (int i = 1; i <= number; i++) {

if (current != nullptr) {

current = current->next;

head = current;

}

}

}

}

// Функция для добавления элемента перед элементом с заданным ключом

void addNode(Node\*& head, int key) {

Node\* current = head;

Node\* prev = NULL;

while (current != nullptr) {

if (current->data == key) {

int NewKey;

cout << "Введите значение нового элемента: ";

cin >> NewKey;

Node\* newNode = createNode(NewKey);

if (prev != NULL) {

prev->next = newNode;

}

else {

head = newNode;

}

newNode->next = current;

}

prev = current;

current = current->next;

}

}

// Функция для уничтожения списка

void deleteList(Node\*& head) {

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

// Функция для сохранения списка в файл

void SaveList(Node\* head) {

Node\* current = head;

char FName[] = "list.txt"; //Путь к файлу

ofstream f1(FName, ios\_base::out|ios\_base::binary); // открываем файл для записи

while (current != nullptr) {

f1.write((char\*)current, sizeof(Node)); /\*записываем структуру в файл\*/

current = current->next;

}

f1.close(); // закрываем файл

}

// Функция для чтения списка из файла

Node\* LoadList() {

Node\* head = nullptr;

Node\* tail = nullptr;

Node current;

char FName[] = "list.txt"; //Путь к файлу

ifstream f2(FName); // открываю файл для чтения

if (!f2) {

cout << "Файл не существует\n\n";

f2.close();

return nullptr;

}

f2.seekg(0, ios::end); //функция перехода в конец файла

long NumBytes = (long)f2.tellg(); //выяснение кол-ва байтов в файле

float NumStruct = NumBytes / sizeof(Node); //выяснение кол-ва структур в файле

f2.seekg(0, ios::beg); //переход в начало файла

for (int i = 1; i <= NumStruct; i++) {

f2.read((char\*)&current, sizeof(Node)); // запись структуры из файла в переменную current

Node\* newNode = createNode(current.data);

if (tail != nullptr) {

tail->next = newNode;

}

else {

head = newNode;

}

tail = newNode;

}

f2.close();

return head;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(0, "ru");

cout << "Создание списка" << endl;

Node\* head = createList();

printList(head);

cout << "Добавление элемента перед элементом с заданным ключом" << endl;

int key;

cout << "Введите ключ, перед которым необходимо добавить элемент: ";

cin >> key;

addNode(head, key);

printList(head);

cout << "Удаление заданного количества элементов, начиная с заданного номера" << endl;

int index;

cout << "Введите номер первого удаляемого элемента: ";

cin >> index;

int number;

cout << "Введите количество удаляемых элементов: ";

cin >> number;

deleteNodes(head, index, number);

printList(head);

cout << "Сохранение списка в файл" << endl;

SaveList(head);

printList(head);

cout << "Уничтожение списка" << endl;

deleteList(head);

printList(head);

cout << "Восстановление списка из файла" << endl;

head = LoadList();

printList(head);

cout << "Уничтожение списка" << endl;

deleteList(head);

printList(head);

return 0;

}

**Код программы (двунаправленный список)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <windows.h>

using namespace std;

// Структура узла списка

struct Node {

int data; // Данные узла - строка символов

Node\* next; // Указатель на следующий элемент

Node\* prev; // указатель на предыдущий элемент

};

// Функция для создания узла списка

Node\* createNode(int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = nullptr;

newNode->prev = nullptr;

return newNode;

}

// Функция для создания списка

Node\* createList() {

Node\* head = nullptr;

Node\* tail = nullptr;

int n;

cout << "Введите число узлов: ";

cin >> n;

for (int i = 1; i <= n; i++) {

//Вариант 1: ручное заполнение списка

//int data;

//cout << "Введите значение узла " << i << ": ";

//cin >> data;

//Node\* newNode = createNode(data); // значение для элемента вводится с клав-ры

//Вариант 2: автоматическое заполнение списка

Node\* newNode = createNode(i); // значение для элемента берется из i

if (tail != nullptr) {

tail->next = newNode;

newNode->prev = tail;

}

else {

head = newNode;

}

tail = newNode;

}

return head;

}

// Функция для печати списка

void printList(Node\* head) {

if (head == nullptr) {

cout << "Список пуст" << endl;

return;

}

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функция для удаления заданного количества элементов, начиная с заданного номера

void deleteNodes(Node\*& head, int index, int number) {

Node\* current = head;

for (int i = 1; i < index; i++) {

if (current->next != nullptr) {

current = current->next;

}

else {

cout << "Элемента с таким номером нет в списке" << endl;

return;

}

}

int count = 0;

bool flag = true;

while (current->next != nullptr && flag == true) {

if (current->prev != nullptr) { //предыдущий элемент существует

current->prev->next = current->next;

current->next->prev = current->prev;

}

else { //предыдущий элемент не существует, т.е. мы удаляем первый элемент в списке

head = current->next;

head->prev = nullptr;

}

Node\* tmp = current;

current = current->next;

delete tmp;

count++;

if (count == number) { //удалили заданное количество элементов

flag = false;

}

}

if (current->next == nullptr && flag == true) { //мы находимся на последнем элементе списка, но нам надо его удалить

if (current->prev != nullptr) { //предыдущий элемент существует

current->prev->next = nullptr;

}

else { //предыдущий элемент не существует, т.е. мы удаляем единственный элемент в списке

head = nullptr;

}

delete current;

}

}

// Функция для добавления элемента перед элементом с заданным ключом

void addNode(Node\*& head, int key) {

Node\* current = head;

Node\* prev = NULL;

while (current != nullptr) {

if (current->data == key) {

int NewKey;

cout << "Введите значение нового элемента: ";

cin >> NewKey;

Node\* newNode = createNode(NewKey);

if (prev != NULL) {

prev->next = newNode;

}

else {

head = newNode;

}

newNode->next = current;

}

prev = current;

current = current->next;

}

}

// Функция для уничтожения списка

void deleteList(Node\*& head) {

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

// Функция для сохранения списка в файл

void SaveList(Node\* head) {

Node\* current = head;

char FName[] = "list.txt"; //Путь к файлу

ofstream f1(FName, ios\_base::out | ios\_base::binary); // открываем файл для записи

while (current != nullptr) {

f1.write((char\*)current, sizeof(Node)); /\*записываем структуру в файл\*/

current = current->next;

}

f1.close(); // закрываем файл

}

// Функция для чтения списка из файла

Node\* LoadList() {

Node\* head = nullptr;

Node\* tail = nullptr;

Node current;

char FName[] = "list.txt"; //Путь к файлу

ifstream f2(FName); // открываю файл для чтения

if (!f2) {

cout << "Файл не существует\n\n";

f2.close();

return nullptr;

}

f2.seekg(0, ios::end); //функция перехода в конец файла

long NumBytes = (long)f2.tellg(); //выяснение кол-ва байтов в файле

float NumStruct = NumBytes / sizeof(Node); //выяснение кол-ва структур в файле

f2.seekg(0, ios::beg); //переход в начало файла

for (int i = 1; i <= NumStruct; i++) {

f2.read((char\*)&current, sizeof(Node)); // запись структуры из файла в переменную current

Node\* newNode = createNode(current.data);

if (tail != nullptr) {

tail->next = newNode;

}

else {

head = newNode;

}

tail = newNode;

}

f2.close();

return head;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(0, "ru");

cout << "Создание списка" << endl;

Node\* head = createList();

printList(head);

cout << "Добавление элемента перед элементом с заданным ключом" << endl;

int key;

cout << "Введите ключ, перед которым необходимо добавить элемент: ";

cin >> key;

addNode(head, key);

printList(head);

cout << "Удаление заданного количества элементов, начиная с заданного номера" << endl;

int index;

cout << "Введите номер первого удаляемого элемента: ";

cin >> index;

int number;

cout << "Введите количество удаляемых элементов: ";

cin >> number;

deleteNodes(head, index, number);

printList(head);

cout << "Сохранение списка в файл" << endl;

SaveList(head);

printList(head);

cout << "Уничтожение списка" << endl;

deleteList(head);

printList(head);

cout << "Восстановление списка из файла" << endl;

head = LoadList();

printList(head);

cout << "Уничтожение списка" << endl;

deleteList(head);

printList(head);

return 0;

}

**Код программы (стек)**

#include <iostream>

using namespace std;

struct stack {

int data;

stack\* prev;

};

stack\* push(stack\*& head, int val) {

stack\* p = new stack;

p->data = val;

p->prev = head;

head = p;

return head;

}

stack\* makeStack(int n) {

stack\* head = nullptr;

for (int i = 1; i <= n; i++) {

head = push(head, i);

}return head;

}

int printStack(stack\* head) {

int count = 0;

stack\* a = head;

if (a == nullptr) {

cout << "список пуст";

return 0;

}

while (a != nullptr) {

cout << a->data << " ";

a = a->prev;

count++;

}

return count;

}

int pop(stack\*& head) {

stack\* p = head;

stack\* t = p->prev;

int temp = p->data;

head = t;

delete p;

return temp;

}

int printStack2(stack\*& st) { // печать стека в правильном порядке

int count = 0;

int data;

stack\* current = st;

stack\* st2 = makeStack(0);

if (current == nullptr) {

cout << "список пуст";

return 0;

}

while (current != nullptr) { //переписываем весь стек st в стек st2 (в обратном порядке)

data = pop(st);

push(st2, data);

current = st;

}

current = st2;

while (current != nullptr) {

data = pop(st2);

cout << data << " ";

push(st, data);

count++;

current = st2;

}

return count;

}

void popEl(int k, int pos, int len, stack\*& st) {

stack\* current = st;

int data;

stack\* st2 = makeStack(0);

for (int i = 1; i <= len - pos - k + 1; i++) {

current = st;

data = pop(st);

push(st2, data);

}

for (int j = 1; j <= k; j++) {

pop(st);

}

while (st2 != nullptr) {

data = pop(st2);

push(st, data);

}

delete st2;

}

void Addval(stack\*& st, int key, int number) {

stack\* st2 = makeStack(0);

stack\* current = st;

int data;

bool flag = 1;

while (st != nullptr and flag == 1) {

current = st;

data = current->data;

if (current->data == key) {

pop(st);

push(st2, data);

push(st, number);

flag = 0;

}

else {

pop(st);

push(st2, data);

}

}

while (st2 != nullptr) {

data = pop(st2);

push(st, data);

}

if (flag == 1) {

cout << "Элемента с ключом " << key << " не найдено" << endl;

}

delete st2;

}

int main() {

setlocale(0, "ru");

cout << "введите кол-во элементов: ";

int n, p, k, pos, len, key, newkey;

cin >> n;

cout << "исходный стек: ";

stack\* st = makeStack(n);

len = printStack2(st);

cout << endl;

cout << "Введите номер, с которого удалять: "; cin >> pos;

cout << "введите кол - во удаляемых элементов : "; cin >> k;

popEl(k, pos, len, st);

cout << "Стек после удаления элементов: ";

len = printStack2(st);

cout << endl;

cout << "Введите ключ, перед которым необходимо добавить элемент: "; cin >> key;

cout << "Введите значение нового элемента: "; cin >> newkey;

Addval(st, key, newkey);

cout << "Стек после добавления элемента: ";

len = printStack2(st);

cout << endl;

return 0;

}

**Код программы (очередь)**

// очередь.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

using namespace std;

struct node {

int data;

node\* next;

};

struct queque {

int len;

node\* head;

node\* tail;

};

void push(queque\*& que, int val) {

node\* p = new node;

p->data = val;

p->next = nullptr;

que->tail->next = p;

que->tail = p;

que->len++;

}

queque\* makeQue(int n) {

queque\* que = new queque;

node\* first = new node;

first->data = 1;

first->next = nullptr;

que->head = first;

que->tail = first;

que->len = 1;

for (int i = 2; i <= n; i++) {

push(que, i);

}

return que;

}

void printQue(queque \* que) {

node\* a = que->head;

if (a == nullptr) {

cout << "список пуст";

return;

}

while (a != nullptr) {

cout << a->data << " ";

a = a->next;

}

}

int pop(queque\* que) {

node\* p = que->head;

int temp = p->data;

que->head = que->head->next;

que->len--;

delete p;

return temp;

}

void popEl(int k, int pos, queque\*& que) {

int data;

int size = que->len;

for (int i = 1; i < pos; i++) {

data = pop(que);

push(que, data);

}

for (int j = 1; j <= k; j++) {

pop(que);

}

for (int i = 1; i <= size-pos-k+1; i++) {

data = pop(que);

push(que, data);

}

}

void Addval(queque\*& que, int key, int number) {

int data;

bool flag = 1;

int cnt = que->len;

while (cnt > 0) {

if (que->head->data == key) {

push(que, number);

data = pop(que);

push(que, data);

flag = 0;

cnt--;

}

else {

data = pop(que);

push(que, data);

cnt--;

}

}

if (flag == 1) {

cout << "Элемента с ключом " << key << " не найдено" << endl;

}

}

int main() {

setlocale(0, "ru");

cout << "введите кол-во элементов: ";

int n, k, pos, key, newkey;

cin >> n;

cout << "исходная очередь: ";

queque\* que = makeQue(n);

printQue(que);

cout << endl << endl;

cout << "Введите номер, с которого удалять: "; cin >> pos;

cout << "введите кол - во удаляемых элементов : "; cin >> k;

popEl(k, pos, que);

cout << "Очередь после удаления элементов: ";

printQue(que);

cout << endl << endl;

cout << "Введите ключ, перед которым необходимо добавить элемент: "; cin >> key;

cout << "Введите значение нового элемента: "; cin >> newkey;

Addval(que, key, newkey);

cout << "Очередь после добавления элемента: ";

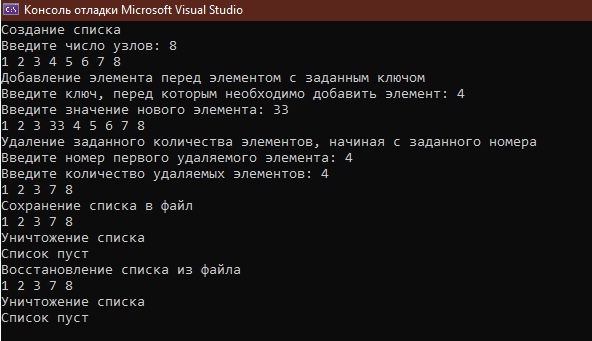
printQue(que);

cout << endl;

return 0;

}

**Вывод программы**

****