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

**ОТЧЁТ**

**по дисциплине «Информатика»**

Семестр: 2

На тему: «Информационные динамические структуры»

Выполнил студент ИВТ-22-2б:

Зайцев Павел Денисович

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

**Для каждого вариант разработать следующие функции:**

1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла

**Вариант №9**

Записи в линейном списке содержат ключевое поле типа int. Сформировать двунаправленный список. Удалить из него К элементов перед элементом с заданным номером, добавить К элементов в конец списка.

**Анализ задачи**

1. Двунаправленный список ₋динамическая структура, каждый элемент которой, называемой узлом, имеет ссылку на следующий элемент и на предыдущий.
2. Реализовать функцию меню для непрерывной работы программы.

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

using namespace std;

int kol = 0;//Счетчик количества узлов

struct spisok2 {

int key;

spisok2\* left;

spisok2\* right;

};

spisok2\* make\_spisok(int n) {//Функция создания списка

spisok2 \* p,\*l,\*m;

l = new(spisok2);//Объявление и заполнение первого узла

l->left = NULL;

l->right = NULL;

l->key = 0;

m = l;//Фиксация первого элемента

if (n > 1) {//Объявление последующих узлов

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

p = new(spisok2);

l->right = p;

p->left = l;

p->right = NULL;

p->key = 0;

l = p;

}

}

return m;//функция возвращает указатель на первый узел

}

void cout\_spisok(spisok2\* jj) {

spisok2\* mm = jj;//Инициализация указателя на первый узел списка

spisok2\* vv = jj;//Инициализация указателя на первый узел списка

int i = 1;//Инициализация счетчика для вывода номера узла списка

bool flag=true;

if (jj == NULL) {//Выход из функции, если передается указатель без списка

cout << "\nСписка не существует!!!\n";

}

else {

while ((vv->right) != NULL) {//Проверка списка на заполненность

if ((vv->key) != 0)flag = false;

vv = vv->right;

}

if (((vv->left) == NULL) &&((vv->right) == NULL) && ((vv->key) != 0)){//Проверка списка на заполненность, если список состоит из 1 узла

flag = false;

}

if (flag == false) {//Вывод списка

cout << "\nНаш список: NULL<-> ";

while ((mm->right) != NULL) {

cout <<"(" << i << ")" << mm->key << " <-> ";

mm = mm->right;

i++;

}

cout << "(" << i << ")" << mm->key << " <-> NULL\n";

}

else {

cout << "\nСписок пуст!!!\n";

}

}

}

void vvod\_spisok(spisok2\* jj) {

int i = 1;//Инициализация счетчика для вывода номера узла списка

spisok2\* hh = jj;//Инициализация указателя на первый узел списка

if (hh != NULL) {

if ((hh->right != NULL)) {

while ((hh->right) != NULL) {

cout << "Введите " << i << "-ый элемент списка:";

cin >> hh->key;//Ввод ключа списка

cout << endl;

i++;

hh = hh->right;//Переход на следующий узел списка

}

cout << "Введите " << i << "-ый элемент списка:";

cin >> hh->key;//Ввод ключа последнего узла списка

}

else {//Ввод ключа единственного узла списка

cout << "Введите ключ узла:";

cin >> hh->key;

cout << "\n";

}

}

else cout << "Списка несуществует\n";

}

spisok2\* udalit(spisok2\* jj) {

int hh,nn;

spisok2\* k = jj;//Инициализация указателя на первый узел списка

spisok2\* y,\*f,\*G;//Инициализация указателей для выполнения удаления

if (k == NULL) {//Проверка на наполненность передоваемого указателя

cout << "\n!!!ОШИБКА!!!\n";

return k;

}

if (((k->left) == NULL) && ((k->right)==NULL)) {//Проверка присутствия файла

cout << "\nОперация недоступна!!!!\n";

return k;

}

cout << endl;

cout << "Введите № узла от которого хотите удалять перед ним стоящие узлы (от 2-"<<kol<<"):";

cin >> hh;

cout << "Введите количество узлов которые хотите удалить (от 1-" << hh- 1 << "):";

cin >> nn;

kol -= nn;

for (int i = 1; i <hh; i++) {//перенос головы на нужный элемент K

k = k->right;

}

y = k->left;

f = k;

for (int r = 0; r <nn+1 ; r++) {// Cоединие списка

f = f->left;

}

k->left = f;

if (nn < hh - 1) {

f->right = k;

while (k->left != NULL) { //Перенос головы в начало списка

k = k->left;

}

}

for (int j = 1; j <= nn; j++) { //Удаление узлов перед элементом K

G = y;

for (int k = 0; k < nn - j;k++) {

G = G->left;

}

delete G;

}

return k;

}

void dobav\_spisok(spisok2\* jj) {

if (jj == NULL) {//Проверка на наполненность передоваемого указателя

cout << "\n!!!ОШИБКА!!!\n";

return;

}

cout << "\nВведите количество узлов,которые вы хотите добавить в конец списка:";

int vv;

int d = 1;

cin >> vv;

if (vv < 1) {

cout << "Неправильный ввод. Нужно повторить операцию\n";

dobav\_spisok(jj);//Рекурсивный вызов функции при неправильном вводе

}

kol += vv;

spisok2\* t;//Инициализация указателя

t= make\_spisok(vv);//Создание нового списка

vvod\_spisok(t);//Запонение нового списка

spisok2\*ff = jj;//Инициализация указателя на первый узел изначального списка

while (ff->right != NULL) {//Переход в конец изначального списка

ff = ff->right;

}

ff->right = t;//Соединение изначального списка с новым

t->left = ff;//Соединение нового списка с изначальным

}

spisok2\* unich(spisok2\* jj) {

spisok2\* kk = jj;//Инициализация указателя на первый узел списка

spisok2\* mm;//Инициализация указателя

if (kk == NULL) {//Проверка на наполненность передоваемого указателя

cout << "\n!!!ОШИБКА!!!\n";

return kk;

}

while (kk->right != NULL) {//Удаление узлов списка из памяти

mm = kk->right;

delete kk;

kk = mm;

}

delete kk;//Удаление последнего узла списка из памяти

return NULL;

}

void file1(spisok2\* jj) {

spisok2\* t = jj;//Инициализация указателя на первый узел списка

int f=0;//Буферная переменная

FILE\* kk;//Ссылка на файл

if (jj == NULL) {//Проверка на наполненность передоваемого указателя

cout << "\n!!!ОШИБКА!!!\n";

return;

}

if ((kk = fopen("123.txt", "w")) == NULL) {//Проверка наличия файла

cout << "Произошла ошибка\n";

exit(1);

}

while (t->right != NULL) {//Запись узлов списка в файл

f = t->key;

fwrite(&f, sizeof(int), 1, kk);

t = t->right;

}

f = t->key;//Запись последнегго узла списка в файл

fwrite(&f, sizeof(int), 1, kk);

fclose(kk);

}

spisok2\* file2() {

kol = 0;

spisok2\* p,\*gol,\*m;

int g;

FILE\* G;

if ((G = fopen("123.txt", "r")) == NULL) {

cout << "Произошла ошибка\n";

exit(1);

}

fread(&g, sizeof(int), 1, G);

p = new(spisok2);

p->left = NULL;

p->right = NULL;

p->key = g;

gol = p;

kol++;

while (fread(&g, sizeof(int), 1, G)) {

m = new(spisok2);

p->right = m;

m->left = p;

m->right = NULL;

m->key = g;

p = m;

kol++;

}

cout << "\nEсли до 8-ой команды вы работали с другим списком, то он будет утерян.\n";

return gol;

}

//

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n = 0;

int h;

spisok2\* j=NULL;

while (true) {

cout << "\nВведите номер операции.\n1)Cоздать список\n2)Заполнить список\n3)Вывести спиcок\n4)Удалить К элементов\n5)Вставить К элементов\n6)Удалить список\n7)Сохранить спиcок в файл\n8)Достать список из файла\n9)Закрыть программу\n №=";

cin >> n;

switch(n) {

case 1:cout << "Введите количество элементов в списке:"; cin >> h; kol = h; j = make\_spisok(h); break;

case 2: vvod\_spisok(j); break;

case 3: cout\_spisok(j); break;

case 4: j = udalit(j); ; break;

case 5: dobav\_spisok(j); break;

case 6: j = unich(j); break;

case 7: file1(j); cout << "7+\n"; break;

case 8: j = file2(); break;

case 9: return 0;

default: cout << "\n!!Неправильный ввод!!\n";

}

}

return 0;

}

**Блок схема программы**

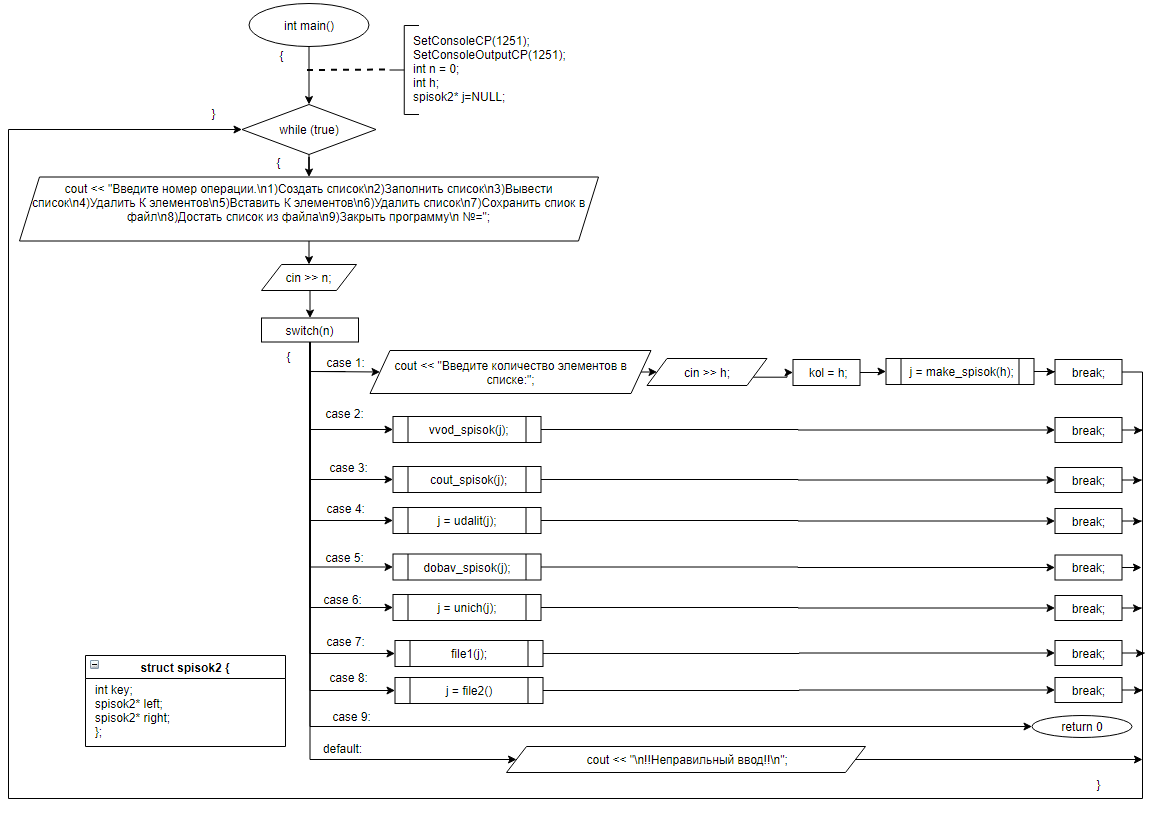


Рисунок 1 -Блок схема основной функции

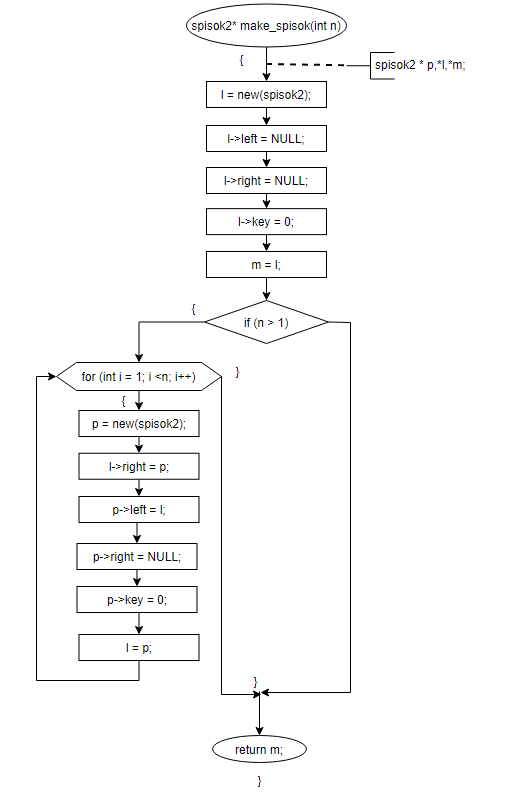


Рисунок 2 -Блок схема функции make\_spisok()

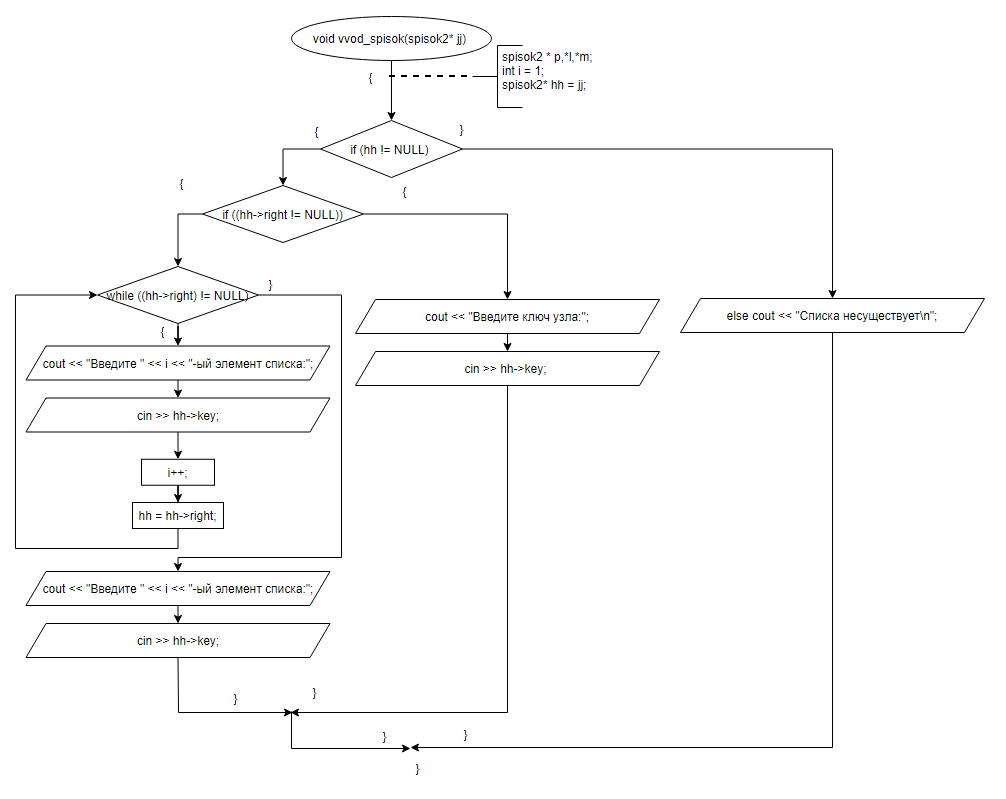


Рисунок 3-Блок схема функции vvod\_spisok()

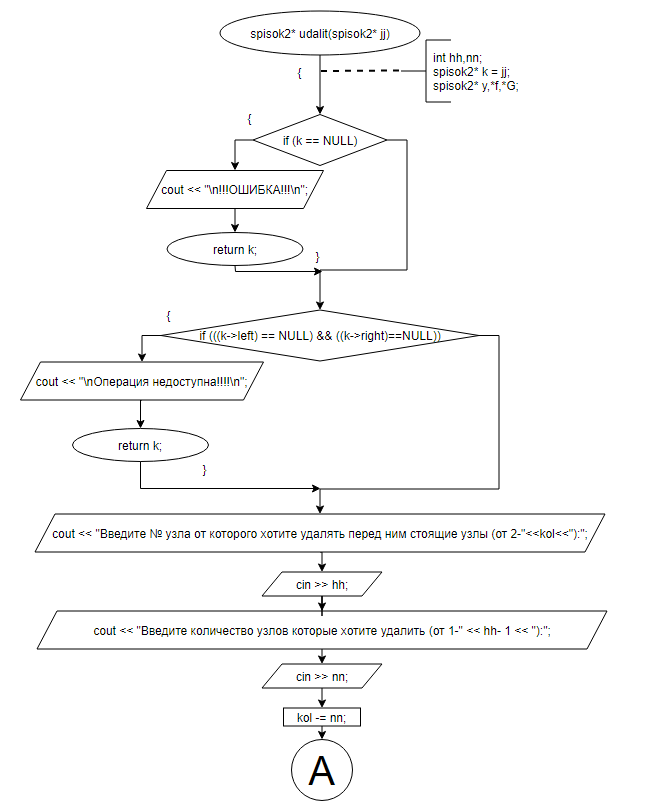


Рисунок 4 -Блок схема функции udalit()

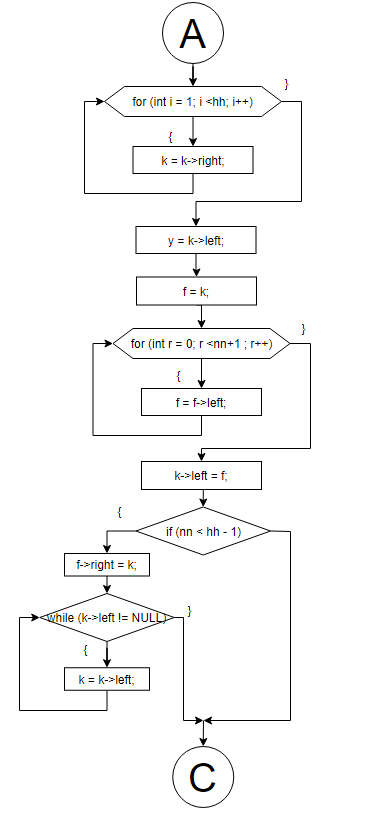


Рисунок 5 -Блок схема функции udalit()

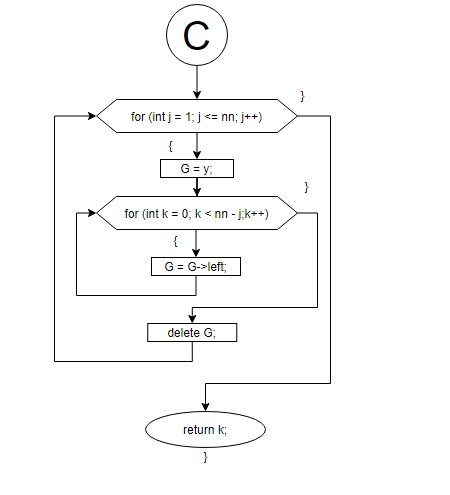


Рисунок 6 -Блок схема функции udalit()

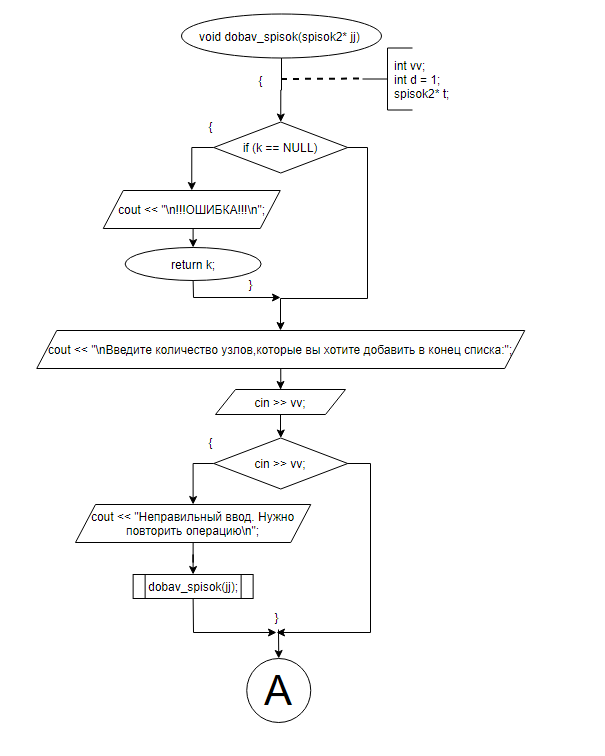


Рисунок 7-Блок схема функции dobav\_spisok()

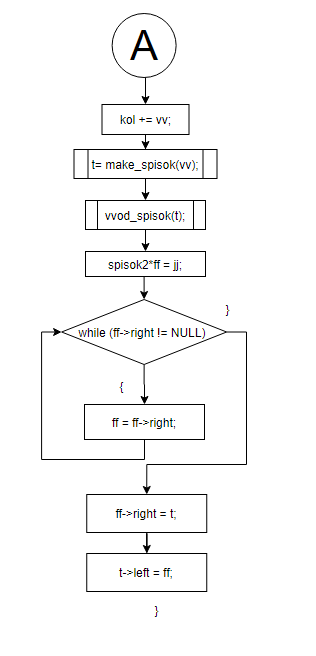


Рисунок 8-Блок схема функции dobav\_spisok()

**Вывод программы**

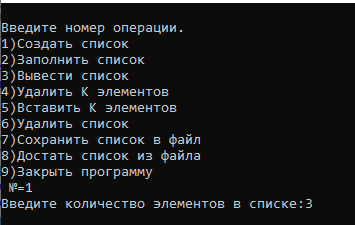


Рисунок 9-Вывод консоли первой команды

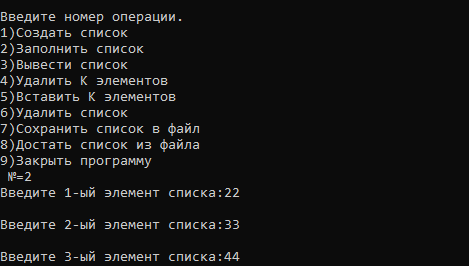


Рисунок 10-Вывод консоли второй команды

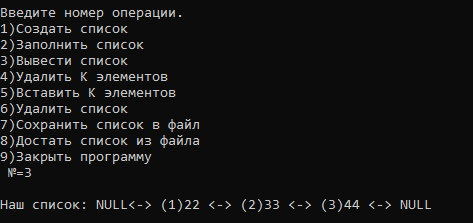


Рисунок 11-Вывод консоли третьей команды

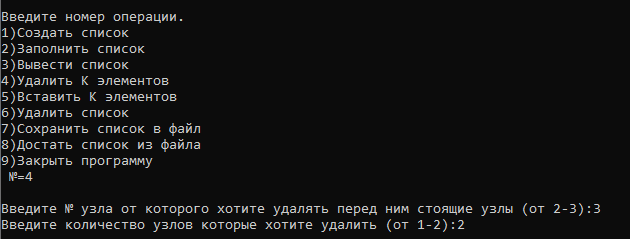


Рисунок 12-Вывод консоли четвёртой команды

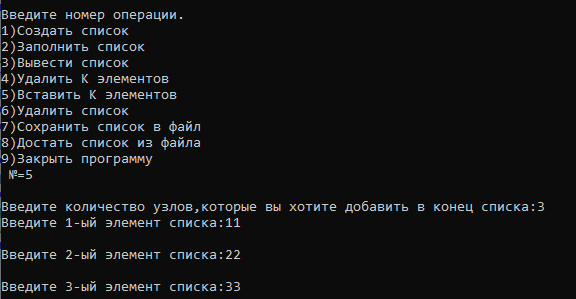


Рисунок 13-Вывод консоли пятой команды



Рисунок 14-Вывод консоли ошибки