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.

**Вариант**

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

**Функции**

1. Функция make\_point() создаёт список.
2. Функция cout\_spisok() выводит в консоль список.
3. Функция vvod\_spisok() вводит в список значения список.
4. Функция udalit() удаляет К элементов от К-ого элемента список.
5. Функция dobav\_spisok() добавляет элементы в конец списка список.
6. Функция unich() удаляет список список.
7. Функция file1() записывает список в файл.
8. Функция file2() считывает список с файла.

**Анализ задачи**

1. Однонапрввленный список это динамическая структура, каждый элемент которой,называемой узлом, связан с последующим элементом.
2. Обявить глобальную переменную счетчик элементов в списке.
3. Реализовать функцию меню для неперывной работы программы.

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

using namespace std;

int kol = 0;

struct spisok{

int key;

spisok\* point;

};

spisok\* make\_point(int n) {

spisok\* p, \* l, \* m;

l = new (spisok);

l->point = NULL;

l->key = 0;

m = l;

if (n > 1) {

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

p = new(spisok);

l->point = p;

p->point = NULL;

p->key = 0;

l = p;

}

}

return m;

}

void cout\_spisok(spisok\* jj) {

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

spisok\* cc = jj;

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 ((mm->point) != NULL) {//Проверка списка на заполненность

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

mm = mm->point;

}

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

flag = false;

}

if (flag == false) {//Вывод списка

cout << "\nНаш список: ";

while ((cc->point) != NULL) {

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

cc = cc->point;

i++;

}

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

}

else {

cout << "\nСписок пуст!!!\n";

}

}

}

void vvod\_spisok(spisok\* jj) {

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

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

if (hh != NULL) {

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

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

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

cin >> hh->key;//Ввод ключа списка

cout << endl;

i++;

hh = hh->point;//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";

}

spisok\* udalit(spisok\* jj) {

int hh, nn;

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

spisok\* y = jj;

spisok\* f, \* G,\*D;//Инициализация указателей для выполнения удаления

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

cout << "\n!!!ОШИБКА!!!\n";

return kk;

}

if (((kk->point) == NULL)) {//Проверка присутствия файла

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

return kk;

}

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;

int x = 0;

if (nn == hh - 1) {

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

y = y->point;

}

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

G = kk;

kk = kk->point;

delete G;

}

return y;

}

else {

f = kk;

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

y = y->point;

}

for (x = 0; x < hh-2-nn; x++) {//перенос c головы на нужный элемент от которого будут сшивать

kk = kk->point;

}

G = kk->point;

kk->point = y;

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

D = G;

G = G->point;

delete D;

}

return f;

}

}

void dobav\_spisok(spisok\* 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;

spisok\* t;//Инициализация указателя

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

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

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

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

ff = ff->point;

}

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

}

spisok\* unich(spisok\* jj) {

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

spisok\* mm;//Инициализация указателя

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

cout << "\n!!!ОШИБКА!!!\n";

return kk;

}

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

mm = kk->point;

delete kk;

kk = mm;

}

delete kk;//Удаление последнего узла списка из памяти

return NULL;

}

void file1(spisok\* jj) {

spisok\* 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->point != NULL) {//Запись узлов списка в файл

f = t->key;

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

t = t->point;

}

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

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

fclose(kk);

}

spisok\* file2() {

kol = 0;

spisok\* 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(spisok);

p->point = NULL;

p->key = g;

gol = p;

kol++;

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

m = new(spisok);

p->point = m;

m->point= 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);

spisok\* j = NULL;

int n = 0;

int h;

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\_point(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**

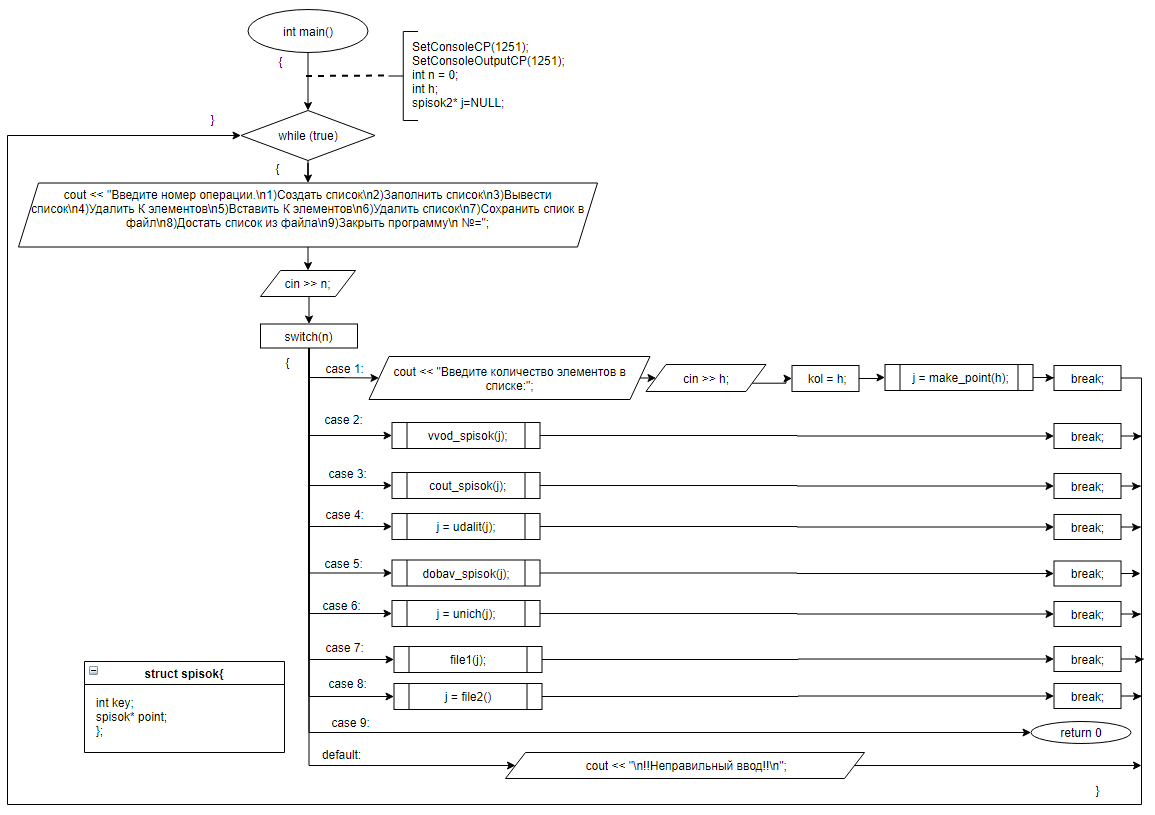


Рисунок 1-Блок схема функции main()

