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 очередь. Удалить из неё К элементов перед элементом с заданным номером, добавить К элементов в конец очереди.

**Функции**

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

**Анализ задачи**

1. Очередь список это структура, каждый элемент которой,называемой узлом, связан с последующим элементом и предыдущим.
2. Объявить глобальную переменную счётчик элементов в списке.
3. Реализовать функцию меню для непрерывной работы программы.
4. Объявить глобальные переменные указатели на начало и конец очереди.

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

using namespace std;

int kol = 0;//Счетчик количества узлов

struct och{

int key;

och\* left;

och\* right;

};

och\* nach = NULL;//Глобальные указатели начала и конца очереди

och \* kon = NULL;

void make\_och(int n) {

nach = new(och);//Создание первого элемента очереди

nach->left = NULL;

nach->right = NULL;

nach->key = 0;

kon = nach;

for (int i = n; i > 1; i--) {//Создание последующих элементов очереди

och\*p = new(och);

p->key = 0;

p->right = kon;

p->left = NULL;

kon->left=p;

kon = p;

}

}

void vvod\_och() {

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

och\* hh = nach;

if (kon != NULL) {

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

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

cout << "Введите " << i << "-ый узел очереди:";

cin >> hh->key;//Ввод ключа i-го списка

cout << endl;

i++;

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

}

cout << "Введите " << i << "-ый узел 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";

}

void cout\_och() {

och\* mm = nach;//Инициализация указателя на первый узел списка

och\* jj = kon;

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

bool flag = true;

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

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

return;

}

else {

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

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

mm = mm->left;

}

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

flag = false;

}

if (flag == false) {//Вывод списка

cout << "\nНаша очередь: ";

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

cout << jj->key << " -> ";

jj= jj->right;

i++;

}

cout << jj->key << " -> NULL\n";

}

else {

cout << "\nСписок пуст!!!\n";

}

}

}

void shag() {

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

cout << "\n!!!ОШИБКА!!!\n";

return;

}

if (nach->left != NULL) {//Перенос первого элемента очереди в его конец

och\* bb = nach->left;//Указатель на следующий элемент очереди

nach->left = NULL;

nach->right = kon;

kon->left = nach;

bb->right = NULL;

kon = nach;

nach = bb;

}

}

void izoch() {

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

cout << "\n!!!ОШИБКА!!!\n";

return;

}

if (nach->left != NULL) {//Удаление первого элемента очереди

och\* nn = nach->left;//Указатель на следующий элемент очереди

nn->right = NULL;

nach->left = NULL;

nach->right = NULL;

delete nach;//удаление узла

nach = nn;//Перенос указателя нчала очереди

}

}

void udalit() {

int hh, nn;

och\* kk = kon;//Инициализация указателя на первый узел списка

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

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

cout << "\n!!!ОШИБКА!!!\n";

}

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

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

return;

}

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;

int x = 0;

if (nn == hh - 1) {

for (int i = 1; i < hh; i++) {//удаление элементов из начала очереди

izoch();//Вызов функции удаления элемента очереди

}

}

else {

for (int h = 1; h <=(hh - 1 - nn); h++) {//Перенос элементов в конец очереди

shag();//Вызов функции переноса первого элемента очереди в ее конец

}

for (int x = 0; x < nn; x++) {//Удалене элементов из очереди

izoch();//Вызов функции удаления элемента очереди

}

}

kol -= nn;

}

void dobav\_och() {

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

cout << "\n!!!ОШИБКА!!!\n";

return;

}

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

int vv;

int d = 1;//Счетчик узлов списка

cin >> vv;

if (vv < 1) {

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

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

}

kol += vv;

for (int r = 0; r < vv; r++) {

och\* hh = new(och);

cout << "\nВведите " << d << "-ый добавляемый элемент очереди:";

cin >> hh->key;//Ввод ключа i-го списка

hh->right = kon;

hh->left = NULL;

kon->left = hh;

kon = hh;

d++;

}

}

void unich() {

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

cout << "\n!!!ОШИБКА!!!\n";

return;

}

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

och\* mm = kon->right;//Буферный езел следующиго элемента очереди

delete kon;//Удаление узла

kon = mm;//Перевод узла на следующий элемент очереди

}

delete kon;//Удаление последнего узла списка из памяти

kon = NULL;

nach = NULL;

return;

}

void file1() {

int f = 0;//Буферная переменная

FILE\* kk;//Ссылка на файл

och\* point = nach;

if (kon == 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 (point->left!= NULL) {//Запись узлов очереди в файл

f = point->key;

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

point = point->left;

}

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

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

fclose(kk);

}

void file2() {

kol = 0;

int g;//Буферная переменная для чтения узлов с файла

FILE\* G;//Объявление ссылки на файл

if ((G = fopen("123.txt", "r")) == NULL){ //Проверка наличие файла и открытие его для чтения

cout << "Произошла ошибка\n";

exit(1);

}

fread(&g, sizeof(int), 1, G);//Считывание узла с файла

och\* p = new(och);//Буферный указатель на первый узел очереди

p->right = NULL;

p->left = NULL;

p->key = g;

nach= p;

kon = p;

kol ++;

while (fread(&g, sizeof(int), 1, G)) {//Пока происходит чтение следующих узлов, создаются новые узлы

och \*m = new(och);

m->right = kon;

m->left = NULL;

m->key = g;

kon->left = m;

kon = m;

kol++;

}

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

return;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n = 0;

int h;

while (true) {

cout << "\nВведите номер операции.\n1)Cоздать очередь\n2)Заполнить очередь\n3)Вывести очередь\n4)Удалить К элементов\n5)Вставить К элементов\n6)Удалить очередь\n7)Сохранить очередь в файл\n8)Достать очередь из файла\n9)Переместить очередь \n10)Закрыть программу\n №=";

cin >> n;

switch (n) {

case 1:cout << "\nВведите количество элементов в очереди:"; cin >> h; kol = h; make\_och(h); break;

case 2: vvod\_och(); break;

case 3: cout\_och(); break;

case 4: udalit(); ; break;

case 5: dobav\_och(); break;

case 6: unich(); break;

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

case 8: file2(); break;

case 9: cout << "\nВведите количество перемещений по кругу:"; cin >> h; for (int y = 0; y < h; y++)shag(); break;

case 10: return 0;

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

}

}

}

**Блок схема**

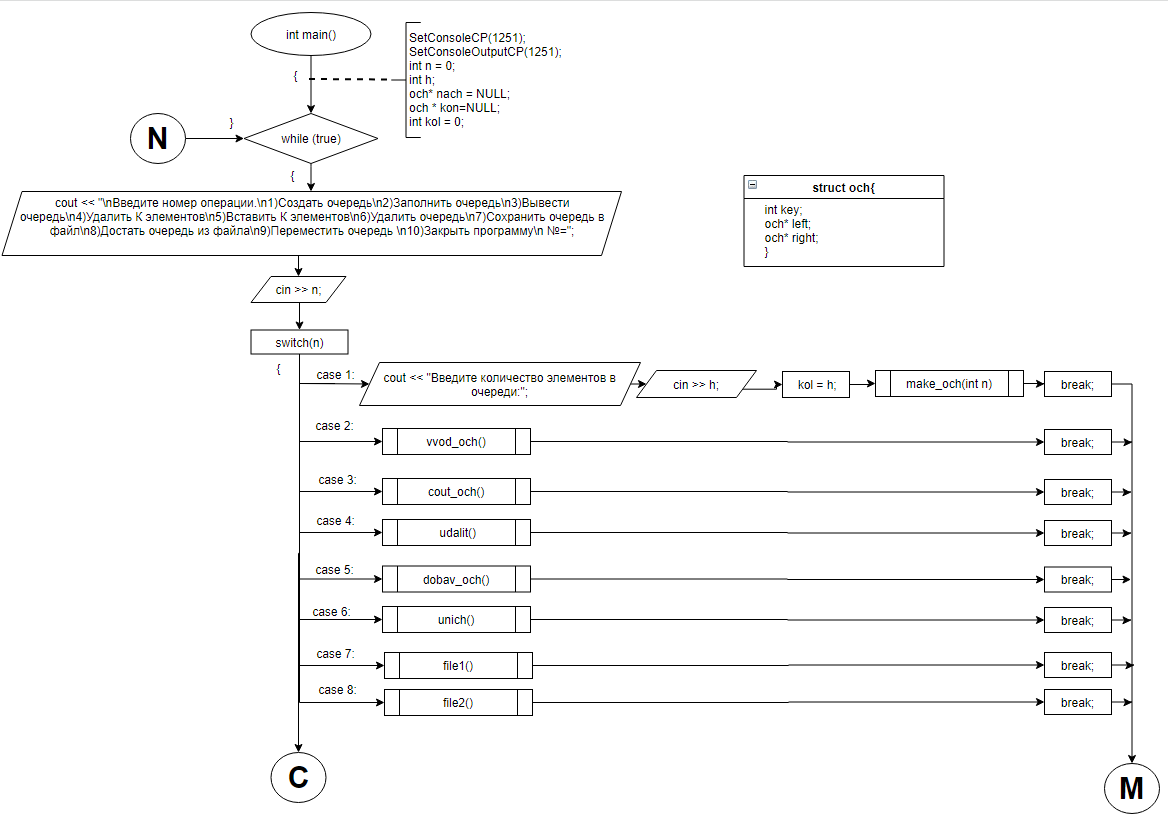


Рисунок 1-Блок схема функции main()

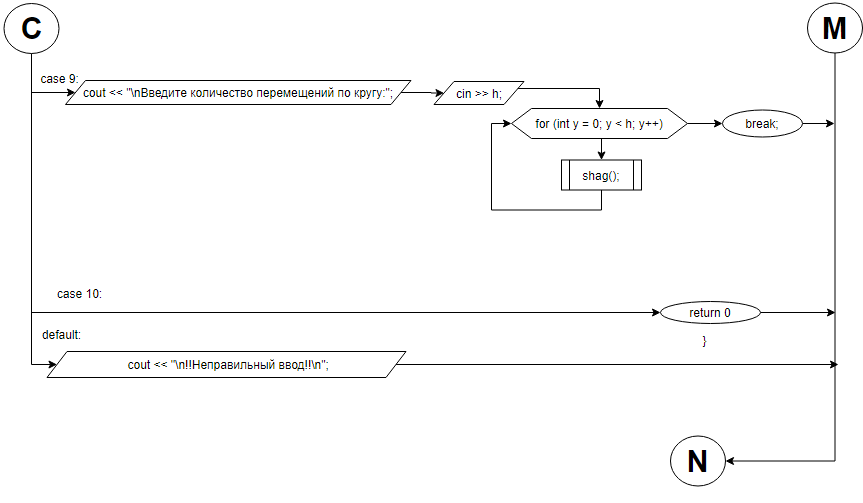


Рисунок 2-Блок схема функции main()

**Вывод программы**

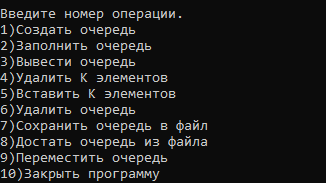


Рисунок 3-Вывод меню программы



Рисунок 4-Вывод команды №1

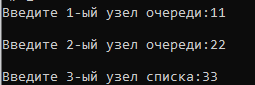


Рисунок 5-Вывод команды №2



Рисунок 6-Вывод команды №3



Рисунок 7-Вывод команды №4

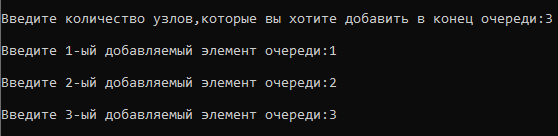


Рисунок 8-Вывод команды №5



Рисунок 9-Вывод команды №9