ЦЕЛЬ РАБОТЫ

Продемонстрировать работа с динамическими структурами данных, на примере списков, очередей, стеков или бинарных деревьев.

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Реализовать программу согласно варианту задания.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Вариант 10

В файловой системе каталог файлов организован в виде линейного списка. Для каждого файла в каталоге содержится имя файла, дата создания и количество обращений к файлу. Написать программу, которая обеспечивает:

* 1. начальное формирование каталога файлов;
  2. вывод каталога файлов;
  3. удаление файлов, дата создания которых меньше заданной;
  4. выборку файла с наибольшим количеством обращений.

Программа должна обеспечивать диалог с помощью меню и контроль ошибок ввода.

БЛОК-СХЕМА АЛГОРИТМА

Блок-схема алгоритма представлена на рис. 1-2.



*Рисунок 1 Блок-схема программы*



*Рисунок 2 Блок-схема программы*

ЛИСТИНГ

#include <iostream>

#include <string>

using namespace std;

struct file

{

int day; //день создания файла

int month; //месяц создания файла

int year; //год создания файла

string name; //имя файла

int call; //кол-во вызовов файла

file \*next; //указатель на след. файл

};

file \*head;

file \*current;

file \*ptr;

file \*call1;

//функция создания определенного количества записей о файлах

void Make\_File(int n, file \*\*head)

{

if (n > 0) {

(\*head) = new file();//выделяем память под новый элемент

cout << "Enter file name: ";

cin >> (\*head)->name; //ввод имени файла и других данных

cout << "Enter creation date (dd/mm/yy): ";

cin >> (\*head)->day >> (\*head)->month >> (\*head)->year;

cout << "Enter number of file calls: ";

cin >> (\*head)->call;

cout << endl;

(\*head)->next = NULL;//обнуление адресного поля

Make\_File(n - 1, &((\*head)->next)); //рекурсия со ссылкой на следующий адрес

}

}

//функция печати списка на экран

void Print\_File(file \*head) {

if (head != NULL) {

cout << head->name << " " << head->day << "/" << head->month << "/" << head->year << " " << head->call << endl;

Print\_File(head->next);

}

else cout << "\n";

}

//функция удаления записей файлов, дата создания которых меньше заданной

void Delete\_File(file \*head, int n, int dayy, int monthh, int yearr)

{

if (n>0) {

file \*ptr = head->next;

while (head->next != NULL)

{

if (ptr->day < dayy || ptr->month < monthh || ptr->year < yearr) {

head->next = ptr->next; //происходит передача адреса элемента следующего за удаляемым

n--;

cout << "File was deleted" << endl;

Delete\_File(head, n, dayy, monthh, yearr);

}

else

{

n--;

Delete\_File(head, n, dayy, monthh, yearr);

}

}

}

}

//функция вывода на экран информации о том, вызов какого файла производился чаще всего

void Bigg\_Calls(file \*head, int n)

{

if (n > 0)

{

file \*maxCalls = head;

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

{

file \*prev = head->next;

if (maxCalls->call < prev->call)

maxCalls = prev;

prev = prev->next;

}

cout << "Highest number of calls: "<< maxCalls->name << " ("<< maxCalls->call << ")" << endl;

}

return;

}

int main()

{

int num, d, m, y, maximum;

string num1;

num1 = " " ;

cout << "Enter number of files: ";

cin >> num;

Make\_File(num, &head);

maximum = -172715;

Bigg\_Calls(head, num);

cout << "Enter delete date: ";

cin >> d >> m >> y;

Delete\_File(head, num, d, m, y);

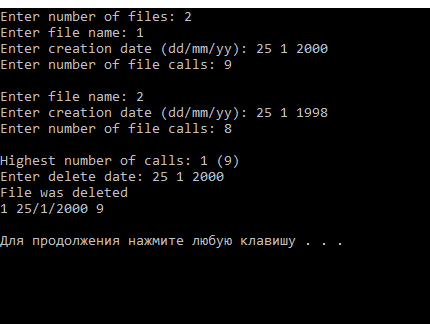
Print\_File(head);

system("pause");

}

ПРОТОКОЛ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Протокол выполнения программы представлен на рис. 3.



*Рисунок 3 Протокол программы*