Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №11

за 2 семестр

По дисциплине: «ОАиП»

Тема: «Бинарные и текстовые файлы»

Выполнила:

Студентка 1 курса

Группы ПО-6(1)

Мартынович Д.

Проверил:

Хацкевич М. В.

2021

Лабораторная работа №12

Бинарные и текстовые файлы

Цель работы: изучить принципы программирования с использованием бинарных файлов. Ознакомиться с основными функциями для работы с бинарными файлами.

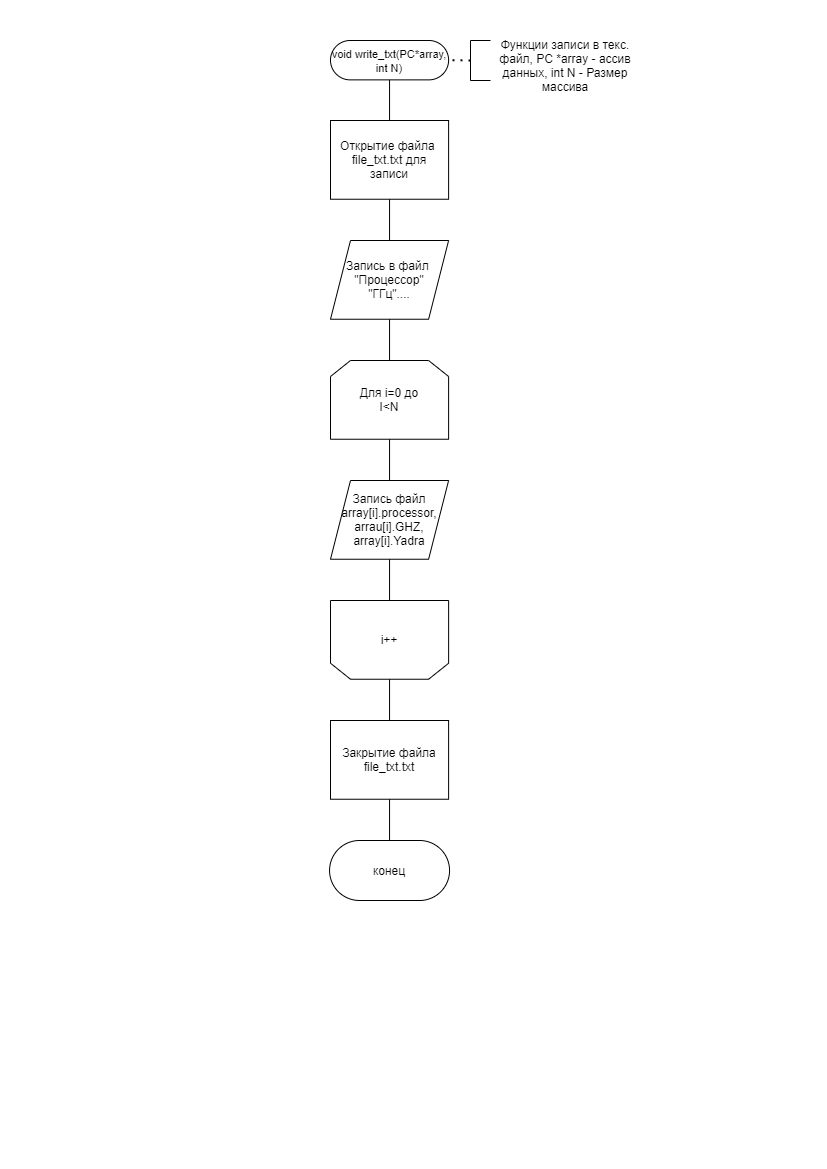
Вариант 18

**Задание:**

В программу, разработанную в лабораторной работе 10, добавить чтение и сохранение данных массива структур при помощи бинарных файлов следующим образом:

1. При первом запуске программы должен создаваться бинарный или текстовый файл на выбор пользователя для хранения данных из массива структур.
2. При добавлении новой записи в массив структур в файл должна дописываться новая запись, без изменения остальных записей.
3. При повторном запуске программы, если файл уже существует, то информация в массив структур должна читаться из этого файла. Если файл отсутствует, то он должен создаваться (см. Пункт 1).
4. Все изменения (сортировка, изменения полей записи, удаление записи) – сохраняются в файле при помощи полной перезаписи содержимого.
5. Сделать вывод о том, какие преимущества использования конкретного типа файлов (бинарные или текстовые) в решаемой вами задаче.

Блок-схема:



Текст программы:

#include <iostream>

#include <windows.h>

#include <fstream>

#include <iomanip>

using namespace std;

struct PC

{

char processor[100];

float GHZ;

int Yadra;

int cache\_memory;

union

{

int map;

int crash;

};

};

int menu(); //организация меню

void input(PC\*& array, int& N); //ввод данных массива структуры

void output(PC\* array, int N); //вывод данных массива структур

void sort(PC\* array, int N, int key); //сортировка во алфавиту по фамилии

int search(PC\* array, int N); //поиск по заданному параметру

void deleted(PC\*& array, int& N, int key); //удаление структуры из массива структур

void write\_txt(PC\* array, int N);

void read\_txt(PC\*& array, int& N);

void add\_txt(PC\*& array, int& N);

void write\_bin(PC\* array, int N);

void read\_bin(PC\*& array, int& N);

void add\_bin(PC\*& array, int& N);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

menu();

return 0;

}

int menu() {

int choise, N = 0;

PC\* array = new PC[N];

int key;

cout << "Работать с текстовым(1) или бинарным файлом(0)?:\n";

cin >> key;

system("cls");

do {

cout << "Выберите действие:\n";

cout << "1) Ввод массива структур\n";

cout << "2) Вывод массива структур\n";

cout << "3) Сортировка массива структур по алфавит\n";

cout << "4) Удаление заданной структуры\n";

cout << "5) Добавление новых элементов\n";

cout << "6) Выход\n";

cin >> choise;

system("cls");

switch (choise)

{

case 1:

if (key == 1) {

read\_txt(array, N);

}

else if (key == 0) {

read\_bin(array, N);

}

break;

case 2:

cin.ignore();

output(array, N);

break;

case 3:

cin.ignore();

sort(array, N, key);

break;

case 4:

cin.ignore();

deleted(array, N, key);

break;

case 5:

cin.ignore();

if (key == 1) {

add\_txt(array, N);

}

else if (key == 0) {

add\_bin(array, N);

}

break;

case 6: return 0;

}

cin.ignore();

system("cls");

} while (true);

delete[] array;

}

void input(PC\*& array, int& N) {

int n;

cout << "Введите количество структур: \n";

cin >> n;

N = N + n;

PC\* tempArr = new PC[N];

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

tempArr[i] = array[i];

}

array = new PC[N];

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

array[i] = tempArr[i];

}

delete tempArr;

system("cls");

cin.ignore();

for (int i = N - n; i < N; i++) {

cout << "1) Название процессора: ";

cin>>(array[i].processor);

cout << "2) ГГц: ";

cin>>(array[i].GHZ);

cout << "3) Кол-во ядер: ";

cin>>(array[i].Yadra);

cout << "4) Кэш-память: ";

cin>>(array[i].cache\_memory);

cin.ignore();

system("cls");

}

}

void output(PC \* array, int N)

{

cout << setw(15) << left << "№";

cout << setw(15) << left << "Процессор";

cout << setw(15) << left << "Ггц";

cout << setw(15) << left << "Ядра";

cout<< setw(15) << left << "Память";

for (int i = 0; i < N; i++)

{

cout << endl;

cout << setw(15) << left << i + 1 ;

cout << setw(15) << left << array[i].processor;

cout << setw(15) << left << array[i].GHZ;

cout << setw(15) << left << array[i].Yadra;

cout << setw(15) << left << array[i].cache\_memory;

}

}

void sort(PC \* array, int N, int key) {

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

for (int j = i; j < N; j++) {

if (strcmp(array[j].processor, array[i].processor) < 0)

swap(array[i], array[j]);

}

}

if (key == 1) {

write\_txt(array, N);

}

else if (key == 0) {

write\_bin(array, N);

}

output(array, N);

}

int search(PC \* array, int N) {

int max = array[0].cache\_memory, max\_ind = 0;

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

if (array[i].cache\_memory > max)

max = array[i].cache\_memory;

}

return max;

}

void deleted(PC \* &array, int& N, int key) {

char Name[20];

cout << "Введите название структуры, которую хотите удалить.\n";

for (int i = 0; i < N; i++)

{

cout << setw(15) << left << i + 1;

cout << setw(15) << left << array[i].processor;

cout << setw(15) << left << array[i].GHZ;

cout << setw(15) << left << array[i].Yadra;

cout << setw(15) << left << array[i].cache\_memory;

cout << endl;

}

gets\_s(Name);

for (int i = 0; i < N; i++) {

if (strcmp(array[i].processor, Name) == 0) {

N--;

for (int j = i; j < N; j++)

array[j] = array[j + 1];

i--;

}

}

PC\* tempArr = new PC[N];

for (int i = 0; i < N; i++)

tempArr[i] = array[i];

array = new PC[N];

for (int i = 0; i < N; i++)

array[i] = tempArr[i];

delete tempArr;

system("cls");

output(array, N);

if (key == 1) {

write\_txt(array, N);

}

else if (key == 0) {

write\_bin(array, N);

}

}

void write\_txt(PC \* array, int N) {

ofstream file\_txt;

file\_txt.open("file\_txt.txt");

if (!file\_txt) {

std::cout << "Ошибка при открытии файла.\n";

}

file\_txt << setw(15) << left << "Процессор" << setw(15) << left << "ГГц";

file\_txt << setw(15) << left << "Ядра" << setw(20) << left << "кэш-память";

for (int i = 0; i < N; i++) {

file\_txt << endl;

file\_txt << setw(15) << left << array[i].processor;

file\_txt << setw(15) << left << array[i].GHZ;

file\_txt << setw(15) << left << array[i].Yadra;

file\_txt << setw(15) << left << array[i].cache\_memory;

}

file\_txt.close();

}

void read\_txt(PC \* &array, int& N) {

ifstream file\_txt;

file\_txt.open("file\_txt.txt");

N = 0;

if (!file\_txt.is\_open()) {

input(array, N);

write\_txt(array, N);

}

else {

array = new PC[N];

char temp[1024];

file\_txt.getline(temp, 1024, '\n');

for (int i = 0; !file\_txt.eof(); i++) {

N++;

PC\* tempArr = new PC[N - 1];

for (int j = 0; j < N - 1; j++) {

tempArr[j] = array[j];

}

array = new PC[N];

for (int j = 0; j < N - 1; j++) {

array[j] = tempArr[j];

}

delete tempArr;

file\_txt >> array[i].processor >> array[i].GHZ >> array[i].Yadra>>array[i].cache\_memory;

}

cout << "Данные считаны из файла.\n";

cin.ignore();

file\_txt.close();

}

}

void add\_txt(PC \* &array, int& N) {

ofstream file\_txt;

file\_txt.open("file\_txt.txt", ios\_base::app);

input(array, N);

write\_txt(array, N);

file\_txt.close();

}

void write\_bin(PC \* array, int N) {

ofstream file\_bin;

file\_bin.open("file\_bin.bin", ios\_base::binary);

if (!file\_bin) {

std::cout << "Ошибка при открытии файла.\n";

}

for (int i = 0; i < N; i++) {

file\_bin.write((char\*) & array[i], sizeof(array[i]));

}

file\_bin.close();

}

void read\_bin(PC \* &array, int& N) {

ifstream file\_bin;

file\_bin.open("file\_bin.bin", ios\_base::binary);

N = 0;

if (!file\_bin.is\_open()) {

input(array, N);

write\_bin(array, N);

}

else {

for (int i = 0; !file\_bin.eof() && (file\_bin.peek() != std::ifstream::traits\_type::eof()); i++) {

N++;

PC\* tempArr = new PC[N - 1];

for (int j = 0; j < N - 1; j++) {

tempArr[j] = array[j];

}

array = new PC[N];

for (int j = 0; j < N - 1; j++) {

array[j] = tempArr[j];

}

delete tempArr;

file\_bin.read((char\*) & array[i], sizeof(array[i]));

}

cout << "Данные считаны из файла.\n";

cin.ignore();

file\_bin.close();

}

}

void add\_bin(PC \* &array, int& N) {

ofstream file\_bin;

file\_bin.open("file\_bin.bin", ios\_base::binary | ios\_base::app);

input(array, N);

write\_bin(array, N);

file\_bin.close();

}

