|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования |
| Московский Государственный Технический Университет имени Н.Э. Баумана |
|  |

Факультет              ФУНДАМЕНТАЛЬНЫЕ НАУКИ

Кафедра             МАТЕМАТИКА И МАТЕМАТИЧЕСКАЯ ФИЗИКА

**Отчет по домашнему заданию № 2.1**

**по курсу «Информатика»**

Студента           Борисов Иван Дмитриевич ­­­­­­­­­­­

(фамилия, имя, отчество)

Группа                                          ФН11-22Б

Преподаватель \_   доцент, к.т.н. Ничушкина Т. Н.

Должность, ФИО, подпись

2019

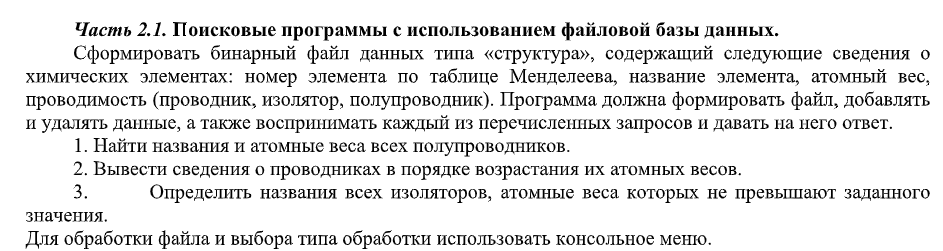
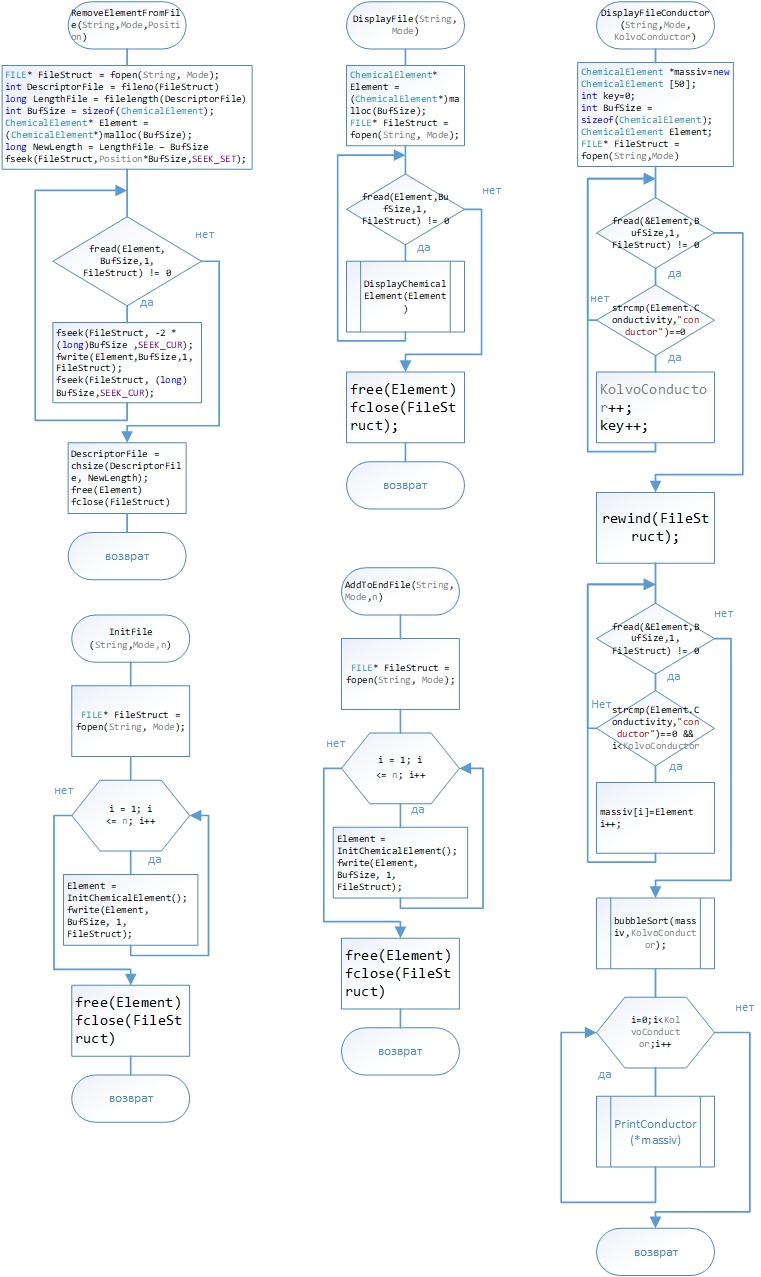
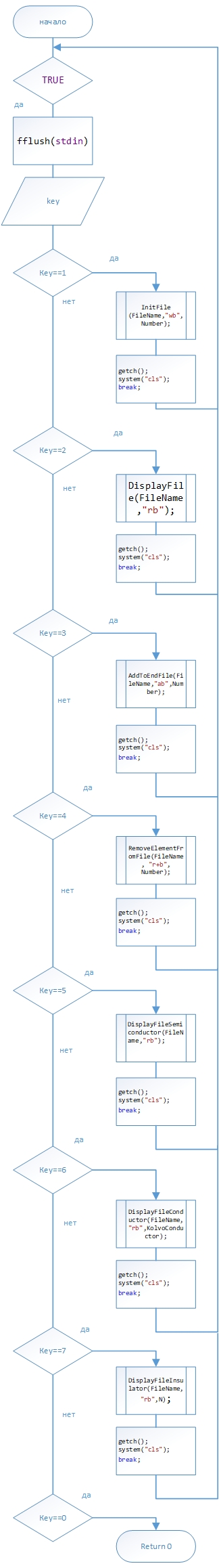


Схема алгоритма





Текст программы

#include "stdafx.h"

#include <io.h>

#include <locale.h>

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <process.h>

#include <iostream>

using namespace std;

#define TRUE 1

typedef struct

{

// Имя и Проводимость

char Name[200],Conductivity [200];// conductor semiconductor insulator

unsigned Number,AtomicWeight;// номер в таблице Менделеева,атомная масса

} ChemicalElement;

void InitFile (char\* , char\* , long );

ChemicalElement\* InitChemicalElement ();

void DisplayFile (char\* , char\* );

void DisplayChemicalElement (ChemicalElement\* );

void AddToEndFile (char\* , char\* , long );

void RemoveElementFromFile (char\* , char\* , long );

void DisplaySemiconductor (ChemicalElement\*);

void DisplayFileSemiconductor(char\*, char\*);

void DisplayFileInsulator(char\*, char\*,int);

void DisplayInsulator (ChemicalElement\*);

void DisplayConductor (ChemicalElement\*);

void DisplayFileСonductor(char\* , char\* ,int);

void bubbleSort(ChemicalElement\* , int ); // сортировка пузырьком

int main(void)

{

setlocale (0,"russian");

char FileName[20];

int KolvoConductor=0;

char Key;

long Number;

puts("Enter the name of file: ");

gets(FileName);

while(TRUE)

{

printf("\n Enter the number - the mode of operations with file: "

"\n 1 - ФОРМИРОВАНИЕ ФАЙЛА"

"\n 2 - ПОСМОТРЕТЬ СОДЕРЖИМОЕ ФАЙЛА"

"\n 3 - ДОБАВИТЬ ЭЛЕМЕНТ В КОНЕЦ ФАЙЛА"

"\n 4 - УДАЛИТЬ ЭЛЕМЕНТ ИЗ ФАЙЛА"

"\n 5 - ВЫВЕСТИ ВСЕ ПОЛУПРОВОДНИКИ И ИХ АТОМНЫЕ ВЕСА"

"\n 6 - ВЫВЕСТИ ВСЕ ПРОВОДНИКИ В ПОРЯДКЕ ВОЗРАСТАНИЯ ИХ АТОМНОГО ВЕСА"

"\n 7 - ВЫВЕСТИ НАЗВАНИЯ ВСЕХ ИЗОЛЯТОРОВ,АТОМНЫЕ ВЕСА КОТОРЫХ НЕ ПРЕВЫШАЮТ N"

"\n 0 - EXIT\n");

fflush(stdin);

scanf("%c",&Key);

switch (Key)

{

case '1':

while(TRUE)

{

printf("\nВведите количество элементов,которые хотите добавить:");

scanf("%ld",&Number);

if (Number > 0)

break;

printf("\n Number is incorrect! Try again!!!\n");

}

InitFile (FileName,"wb",Number);

printf("\n\nPress any key to return in the menu...");

getch();

system("cls");

break;

case '2':

DisplayFile(FileName,"rb");

printf("\n\nPress any key to return in the menu...");

getch();

system("cls");

break;

case '3':

while(TRUE)

{

printf("\nВведите количество элементов,которые хотите добавить:");

scanf("%ld",&Number);

if (Number > 0)

break;

printf("\n Number is incorrect! Try again!!!\n");

}

AddToEndFile(FileName,"ab",Number);

printf("n\nPress any key to return in the menu...");

getch();

system("cls");

break;

case '4':

printf("\nВведите номер элемента в файле,который хотите удалить :");

DisplayFile(FileName,"rb");

scanf("%ld",&Number);

RemoveElementFromFile(FileName, "r+b", Number);

printf("\n\nPress any key to return in the menu...");

getch();

system("cls");

break;

case '5':

DisplayFileSemiconductor(FileName,"rb");

printf("\n\nPress any key to return in the menu...");

getch();

system("cls");

break;

case '6':

//ВЫВЕСТИ ВСЕ ПРОВОДНИКИ В ПОРЯДКЕ ВОЗРАСТАНИЯ ИХ АТОМНОГО ВЕСА

DisplayFileСonductor(FileName,"rb",KolvoConductor);

printf("\n\nPress any key to return in the menu...");

getch();

system("cls");

break;

case '7':

int N;

puts("Введите N");

scanf("%d",&N);

DisplayFileInsulator(FileName,"rb",N);

printf("\n\nPress any key to return in the menu...");

getch();

system("cls");

break;

case '0':

return 0;

default:

printf("\nIncorrect input! Try again!!!");

printf("\n\nPress any key to return in the menu...");

getch();

system("cls");

break;

}

}

}

// запись структур в бинарный файл

void InitFile(char\* String, char\* Mode, long n)

{

long i;

int BufSize = sizeof(ChemicalElement);

ChemicalElement\* Element;

FILE\* FileStruct = fopen(String, Mode);

if (FileStruct == NULL)

{

printf("Can't open file to write.");

getch();

abort();

}

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

{

printf("\nEnter information for the ChemicalElement number %ld \n", i);

Element = InitChemicalElement();

fwrite(Element, BufSize, 1, FileStruct);

}

free(Element);

fclose(FileStruct);

}

// добавление структур в бинарный файл

void AddToEndFile(char\* String, char\* Mode, long n)

{

long i;

int BufSize = sizeof(ChemicalElement);

ChemicalElement\* Element;

FILE\* FileStruct = fopen(String, Mode);

if (FileStruct == NULL)

{

printf("Can't open file to write.");

getch();

abort();

}

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

{

printf("\nEnter information for the ChemicalElement number %ld \n", i);

Element = InitChemicalElement();

fwrite(Element, BufSize,1, FileStruct);

}

free(Element);

fclose(FileStruct);

}

// удаление структур из бинарного файла

void RemoveElementFromFile(char\* String, char\* Mode, long Position)

{

FILE\* FileStruct = fopen(String, Mode);

int DescriptorFile = fileno(FileStruct);

long LengthFile = filelength(DescriptorFile);

int BufSize = sizeof(ChemicalElement);

ChemicalElement\* Element = (ChemicalElement\*)malloc(BufSize);

long NewLength = LengthFile - BufSize;

if (FileStruct == NULL)

{

printf("Can't open file to write.");

getch();

abort();

}

fseek(FileStruct,Position \* BufSize,SEEK\_SET);

while(fread(Element, BufSize,1, FileStruct) != 0)

{

fseek(FileStruct, -2 \* (long)BufSize ,SEEK\_CUR);

fwrite(Element,BufSize,1, FileStruct);

fseek(FileStruct, (long) BufSize,SEEK\_CUR);

}

DescriptorFile = chsize(DescriptorFile, NewLength);

free(Element);

fclose(FileStruct);

}

// заполнение полей структуры ChemicalElement

ChemicalElement\* InitChemicalElement()

{

ChemicalElement\* Element = (ChemicalElement\*)malloc(sizeof(ChemicalElement));

printf("\nEnter name:");

scanf("%s", Element->Name);

printf("\nEnter Number:");

scanf("%d",&Element ->Number);

printf("\nEnter Atomic Weight:");

scanf("%d",&Element->AtomicWeight);

printf("\nEnter Conductivity:");

scanf("%s", Element ->Conductivity);

return Element;

}

//выводит на экран содержимое бинарного файла

void DisplayFile(char\* String, char\* Mode)

{

int BufSize = sizeof(ChemicalElement);

ChemicalElement\* Element = (ChemicalElement\*)malloc(BufSize);

FILE\* FileStruct = fopen(String, Mode);

if ( FileStruct == NULL)

{

printf("Can't open file to read.");

getch();

abort();

}

while(fread(Element,BufSize,1, FileStruct) != 0)

{

DisplayChemicalElement(Element);

}

free(Element);

fclose(FileStruct);

}

//выводит на экран структуру типа ChemicalElement из бинарного файла

void DisplayChemicalElement (ChemicalElement\* Element)

{

printf("\nName of Element: %s; Number of Element: %d; Conductivity of Element: %s; Atomic Weight of Element: %d;", Element->Name,Element->Number, Element->Conductivity,Element->AtomicWeight);

}

void DisplaySemiconductor (ChemicalElement\* Element)

{

printf("\nName of Element: %s;Conductivity of Element: %s; Atomic Weight of Element: %d;", Element->Name,Element->Conductivity,Element->AtomicWeight);

}

void DisplayFileSemiconductor(char\* String, char\* Mode)

{

int key=0;

int BufSize = sizeof(ChemicalElement);

ChemicalElement\* Element = (ChemicalElement\*)malloc(BufSize);

FILE\* FileStruct = fopen(String,Mode);

if ( FileStruct == NULL)

{

printf("Can't open file to read.");

getch();

abort();

}

while(fread(Element,BufSize,1, FileStruct) != 0)

{

if (strcmp(Element->Conductivity,"semiconductor")==0)

{

DisplaySemiconductor(Element);

key++;

}

}

if (key==0)

puts("В файле нет нужных элементов!!!");

free(Element);

fclose(FileStruct);

}

void DisplayInsulator (ChemicalElement\* Element)

{

printf("\nName of Element: %s;Conductivity of Element: %s; Atomic Weight of Element: %d;", Element->Name,Element->Conductivity,Element->AtomicWeight);

}

void DisplayFileInsulator(char\* String, char\* Mode,int N)

{

int key=0;

int BufSize = sizeof(ChemicalElement);

ChemicalElement\* Element = (ChemicalElement\*)malloc(BufSize);

FILE\* FileStruct = fopen(String,Mode);

if ( FileStruct == NULL)

{

printf("Can't open file to read.");

getch();

abort();

}

while(fread(Element,BufSize,1, FileStruct) != 0)

{

if (strcmp(Element->Conductivity,"insulator")==0 && Element->AtomicWeight<=N)

{

DisplayInsulator(Element);

key++;

}

}

if (key==0)

puts("В файле нет нужных элементов!!!");

free(Element);

fclose(FileStruct);

}

void DisplayFileСonductor(char\* String, char\* Mode,int KolvoConductor)

{

ChemicalElement \*massiv=new ChemicalElement [50];

int key=0;

int BufSize = sizeof(ChemicalElement);

ChemicalElement Element;

FILE\* FileStruct = fopen(String,Mode);

if ( FileStruct == NULL)

{

printf("Can't open file to read.");

getch();

abort();

}

while(fread(&Element,BufSize,1, FileStruct) != 0)

{

if (strcmp(Element.Conductivity,"conductor")==0)

{

KolvoConductor++;

key++;

}

}

rewind(FileStruct);

int i=0;

while(fread(&Element,BufSize,1, FileStruct) != 0)

{

if (strcmp(Element.Conductivity,"conductor")==0 && i<KolvoConductor)

{

massiv[i]=Element;

i++;

}

}

bubbleSort(massiv,KolvoConductor);

if (key==0)

puts("В файле нет нужных элементов!!!");

for (i=0;i<KolvoConductor;i++)

printf("\nName of Element: %s;Conductivity of Element: %s; Atomic Weight of Element: %d;", massiv[i].Name,massiv[i].Conductivity,massiv[i].AtomicWeight);

fclose(FileStruct);

}

void bubbleSort(ChemicalElement\* arrayPtr,int length\_array) // сортировка пузырьком

{

ChemicalElement temp; // временная переменная для хранения элемента массива

bool exit = false; // болевая переменная для выхода из цикла, если массив отсортирован

while (!exit) // пока массив не отсортирован

{

exit = true;

for (int int\_counter = 0; int\_counter < (length\_array - 1); int\_counter++) // внутренний цикл

if (arrayPtr[int\_counter].AtomicWeight > arrayPtr[int\_counter + 1].AtomicWeight) // сравниваем два соседних элемента

{

// выполняем перестановку элементов массива

temp = arrayPtr[int\_counter];

arrayPtr[int\_counter] = arrayPtr[int\_counter + 1];

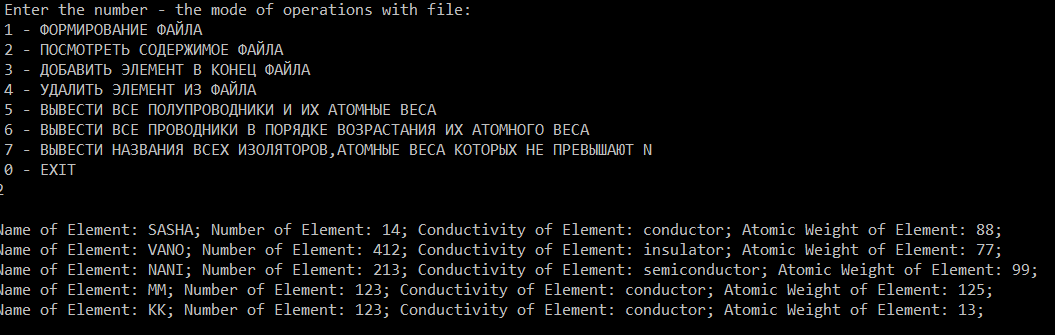
arrayPtr[int\_counter + 1] = temp;

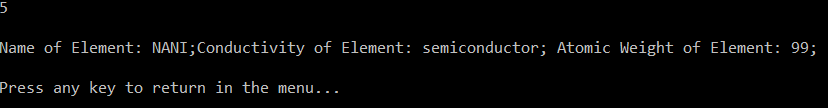
exit = false; // на очередной итерации была произведена перестановка элементов

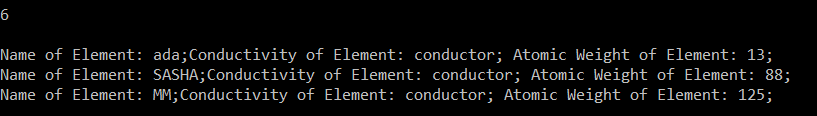
}

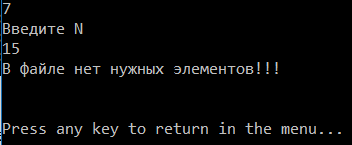
}

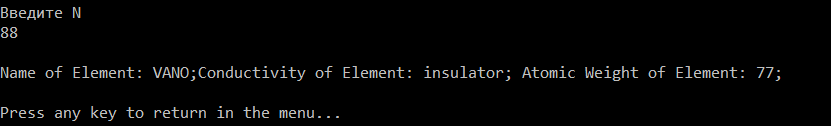
}

Тесты  










Вывод

Я закрепил материал по работе с файловыми структурами данных и применение этих структур в практике программирования.