*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(национальный исследовательский университет)*** |

ФАКУЛЬТЕТ \_\_\_\_ ФУНДАМЕНТАЛЬНЫЕ НАУКИ\_\_\_\_\_\_\_\_

КАФЕДРА\_\_\_ВЫЧИСЛИТЕЛЬНАЯ\_МАТЕМАТИКА\_И\_МАТЕМАТИЧЕСКАЯФИЗИКА\_(ФН11)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по домашней работе №\_\_\_2\_\_\_**

**Название домашней работы:**

Часть 1. Поисковые программы с использованием файловой базы данных.

Часть 2 Древовидная рекурсия.

**Дисциплина:** Информатика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент гр. ФН11-22б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Паскал**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. Н. Ничушкина**

(Подпись, дата) (И.О. Фамилия)

Москва, 2019

*Часть 1: Домашняя работа №2*

**Цель:** Создать поисковую программу с использованием файловой базы данных.

**Задача:** Сведения о домах на участках представлены для каждого их них этажностью, площадью, годом застройки, материалом (кирпичный, деревянный,…), стоимостью, фамилией владельца. Программа должна формировать бинарный файл типа «структура», добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Определить фамилии владельцев, имеющих дома указанной этажности.

2. Вывести данные о домах, изготовленных из указанного материала.

3. Определить фамилии владельцев, дома которых построены из дерева после указанного срока.

Для обработки файла и выбора типа обработки использовать консольное меню.

***Задача 1. Создание алгоритма .***

***C:\Users\Pascal\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Алгоритм_шаблон1.emf***

***Задача 2. Создание программы.***

Основная программа **DZ\_2\_Z1.cpp:**

#include "stdafx.h"

#include <locale.h>

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

struct house{

int floors;

double square;

int year;

char material[15];

double cost;

char owner[30];

};

void inputhouse(house \*cur);

void houseinfo(house cur);

void allinfo(FILE \*f);

bool dbtryopen(FILE \*\*f);

int main(void)

{

setlocale(LC\_ALL,"Rus");

int cmd;

FILE \*f;

house r;

printf("БАЗА ДАННЫХ СВЕДЕНИЙ О ДОМАХ\n\n");

printf("Введите:\n 1. Ввод данных о новом участке\n 2. Удаление данных об участке\n 3. Поиск по базе\n 4. Вывод на экран всей базы данных\n 5. Выход из программы");

do

{

printf("\n<");

scanf("%d", &cmd);

switch (cmd)

{

case 1:

{

/\* открытие на запись \*/

if((f=fopen("database.dat", "a+b"))==NULL)

{

printf("\nНевозможно открыть файл\n");

return 1;

}

/\* сохранение в файл f \*/

inputhouse(&r);

fwrite(&r, sizeof r, 1, f);

fclose(f);

break;

}

case 2:

{

int num;

printf("\nВведите номер дома в базе, который надо удалить: ");

do{

scanf("%d", &num);

if (num<1) printf("Нумерация начинается с \"1\". Посторите ввод: ");

} while (num<1);

FILE \*g;

/\* открытие на запись \*/

if (dbtryopen(&f)) break;

if((g=fopen("database2.dat", "wb"))==NULL)

{

printf("\nНевозможно открыть файл\n");

break;

}

int n=1;

while ((fread(&r, sizeof r, 1, f))==1)

{

if (n!=num) fwrite(&r, sizeof r, 1, g);

n++;

}

fclose(f);

fclose(g);

if (remove("database.dat")) printf("\nОшибка удаления файла.\n");

rename("database2.dat", "database.dat");

if (num>=n) printf("\nВ базе нет дома под этим номером.\n");

else printf("\nДанные о доме успешно удалены\n");

break;

}

case 3:

{

printf(" 1. Определить фамилии владельцев, имеющих дома указанной этажности.\n 2. Вывести данные о домах, изготовленных из указанного материала.\n 3. Определить фамилии владельцев, дома которых построены из дерева после указанного срока.\n <0> для возврата в главное меню.\n");

int key2;

do

{

printf("\nВведите номер типа поиска: ");

scanf("%d", &key2);

switch (key2)

{

case 1:

{

if (dbtryopen(&f)) break;

int fl;

printf("\nВведите колличество этажей: ");

scanf("%d", &fl);

int n=1;

while ((fread(&r, sizeof r, 1, f))==1)

{

if (fl==r.floors)

{

if (n==1) printf("\nФамилии владельцев:\n");

printf(" %d) %s\n", n, r.owner);

n++;

}

}

if (n==1) printf("Совпадений не найдено\n");

fclose(f);

break;

}

case 2:

{

if (dbtryopen(&f)) break;

char mat[15];

printf("\nВведите название материала (brick, wooden,...): ");

scanf("%s", &mat);

int n=1;

while ((fread(&r, sizeof r, 1, f))==1)

{

if (strcmp(r.material, mat)==0)

{

printf("\nДанные о %d-м доме:\n", n);

houseinfo(r);

n++;

}

}

if (n==1) printf("Совпадений не найдено\n");

fclose(f);

break;

}

case 3:

{

if (dbtryopen(&f)) break;

int y;

printf("\nВведите год: ");

scanf("%d", &y);

int n=1;

while ((fread(&r, sizeof r, 1, f))==1)

{

if ((r.year > y) && (strcmp(r.material, "wooden")==0))

{

if (n==1) printf("\nФамилии владельцев:\n");

printf(" %d) %s\n", n, r.owner);

n++;

}

}

if (n==1) printf("Совпадений не найдено\n");

fclose(f);

break;

}

case 0:

{

break;

}

default:

{

printf("\nНеверная команда\n");

}

}

} while (key2!=0);

break;

}

case 4:

{

if (dbtryopen(&f)) break;

allinfo(f);

fclose(f);

break;

}

case 5:

{

break;

}

default:

{

printf("Неверная команда");

break;

}

}

} while (cmd !=5);

printf("\n\nнажмите любую клавишу для завершения...");

//remove("database.dat");

getch();

return 0;

}

void inputhouse(house \*cur)

{

printf("\nВведите данные о доме:\n");

printf("Колличество этажей: ");

scanf("%d", &cur->floors);

printf("Площадь: ");

scanf("%lf", &cur->square);

printf("Год: ");

scanf("%d", &cur->year);

printf("Материал (brick, wooden,...): ");

scanf("%s", &cur->material);

printf("Цена: ");

scanf("%lf", &cur->cost);

printf("Фамилия владельца: ");

scanf("%s", &cur->owner);

}

void houseinfo(house cur)

{

printf(" Колличество этажей: %d\n", cur.floors);

printf(" Площадь: %lf\n", cur.square);

printf(" Год: %d\n", cur.year);

printf(" Материал (brick, wooden,...): %s\n", cur.material);

printf(" Цена: %lf\n", cur.cost);

printf(" Фамилия владельца: %s\n", cur.owner);

}

bool dbtryopen(FILE \*\*f)

{

if((\*f=fopen("database.dat", "rb"))==NULL)

{

printf("\nНевозможно открыть файл\n");

return 1;

}

else return 0;

}

void allinfo(FILE \*f)

{

int n=1;

house r;

while ((fread(&r, sizeof r, 1, f))==1)

{

printf("\nДанные о %d-м доме:\n", n);

houseinfo(r);

n++;

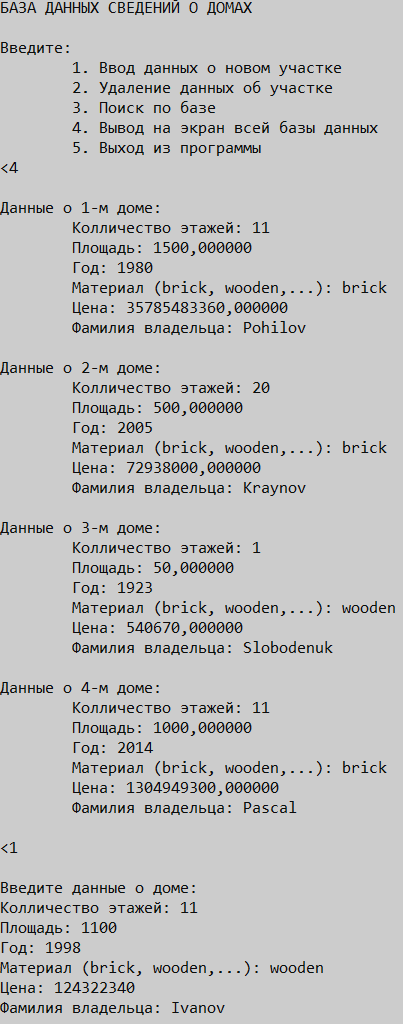
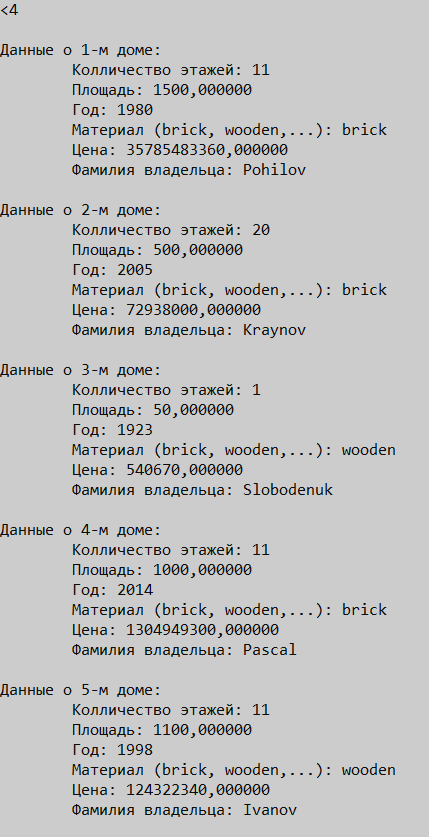
}

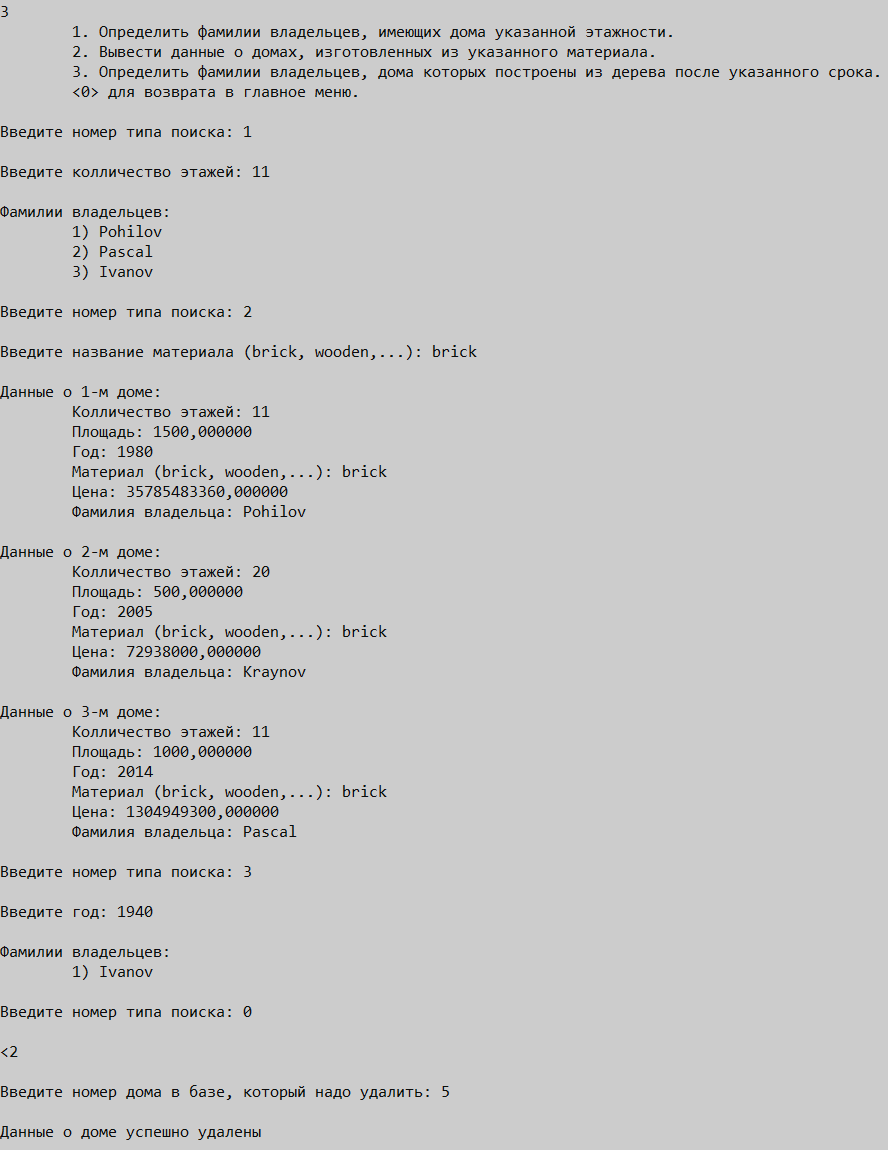
if (n==1) printf("\nБаза пуста.\n", n);

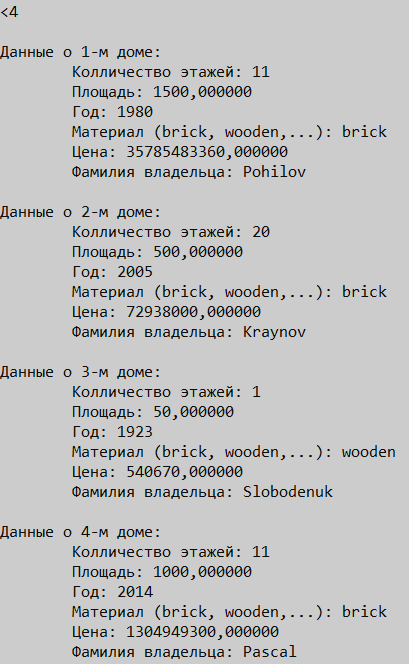
}

***Задача 3. Проверка работоспособности программы.***

Результат выполнения программы:







**Вывод:** В процессе выполнения второй части ДЗ№3 поставленную цель удалось достичь.

*Часть 2: Домашняя работа №2*

**Цель**: Составить программу, используя рекурсивную процедуру (функцию).

**Задача:** Написать программу, реализующую рекурсивный алгоритм нахождения указанного числа в некотором упорядоченном массиве из n чисел при помощи разделения массива пополам (бинарный поиск).

Оттестировать программу на массивах разной размерности. Определить, как влияет размер массива на количество вызовов рекурсивной подпрограммы. Вывести на экран всю необходимую информа-цию. В отчете привести дерево рекурсии для небольшого массива.

Дерево рекурсии для массива { 0, 1, 4, 9, 16, 25, 36 }. Поиск индекса элемента 9.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 4 | 9 | 16 | 25 | 36 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 4 | 9 |

|  |  |  |
| --- | --- | --- |
| 16 | 25 | 36 |

|  |  |
| --- | --- |
| 4 | 9 |

|  |  |
| --- | --- |
| 16 | 25 |

|  |
| --- |
| 36 |

|  |  |
| --- | --- |
| 0 | 1 |

|  |
| --- |
| 4 |

C:\Users\Pascal\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Алгоритм_шаблон.emf***Задача 1. Создание алгоритма программы .***

***C:\Users\Pascal\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Алгоритм_шаблон.emf***

***Задача 2. Создание программы.***

Основная программа **DZ\_2\_Z2.cpp:**

include "stdafx.h"

#include <locale.h>

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

int binarysearch (int \*arr, int b, int num, int &counter, int a=0)

{

counter++;

int m = a+(b-a)/2;

if (a-b != 0)

{

if (arr[m] == num) return m;

if (arr[m] < num) binarysearch(arr, b, num, counter, m+1);

else if (arr[m] > num) binarysearch(arr, m, num, counter, a);

}

else if (arr[m] != num) return -1;

else return m;

}

int main(void)

{

setlocale(LC\_ALL,"Rus");

int \*arr, n;

printf("Введите кол-во элементов в массиве: ");

scanf("%d", &n);

arr = new int [n];

printf("Сформированный массив:");

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

{

arr[i]=i\*i;

printf(" %d", arr[i]);

}

int k;

printf("\nЭлемент, индекс которого ищем: ");

int num, counter=0;

scanf("%d", &num);

k = binarysearch(arr,n-1,num, counter);

if (k != -1)

{

printf("\nИндекс элемента: %d", k);

printf("\nЭлемент: %d", arr[k]);

printf("\nкол-во вызовов рекурсии: %d", counter);

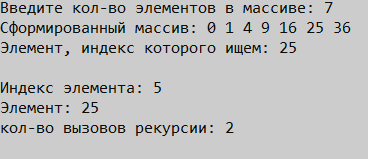
}

else printf("Элемент не найден.");

getch();

}

***Задача 3. Проверка работоспособности программы.***



**Вывод:** В процессе выполнения второй части ДЗ№3 поставленную цель удалось достичь.