Лабораторная работа №1

«Работа с таблицей идентификаторов»

**Задание:**

Разработать функции для включения описания идентификатора в таблицу, поиска по имени и исключения описания идентификатора из таблицы. Описание идентификатора состоит из имени идентификатора и его атрибута.

Разработать программу, демонстрирующую использование указанных функций. Программа должна быть разработана как консольное приложение на языке C# в среде Visual Studio.

Способ представления таблицы – массив указателей на динамические элементы. Метод поиска – бинарный поиск.

*Класс, применяемый в программе:*

public unsafe class Iden // char unsafe

{

public int atr;

public string id;

}

public unsafe class Mass // unsafe

{

public int size;

public Iden[] num2 = new Iden[100];

public Mass()

{

size = 0;

Iden myIden = new Iden();

myIden.atr = 0;

myIden.id = " ";

num2[0] = myIden;

}

~Mass()

{

}

**Описание работы программы**

static void Main(string[] args) - главная функция, выводящее меню и позволяющая пользователю выбрать одно из действий для дальнейшей работы

public void Add(int el1, string el2)- функция вызываемая при вводе элемента(атрибута и идентификатора) и добавляющая их в массив

public int Inde(char nu1) - вспомогательная функция вызываемая из Add и позволяющая сортировать элементы в массиве при добавлении. Выдаёт номер буквы в алфавите

public void Vivod() - функция осуществляющая вывод содержимого массива и его размера

public void Poisk2(string key) - функция осуществляющая поиск по идентификатору в отсортированном массиве(сначала по первому символу, если он совпал то по следующему и так до конца)

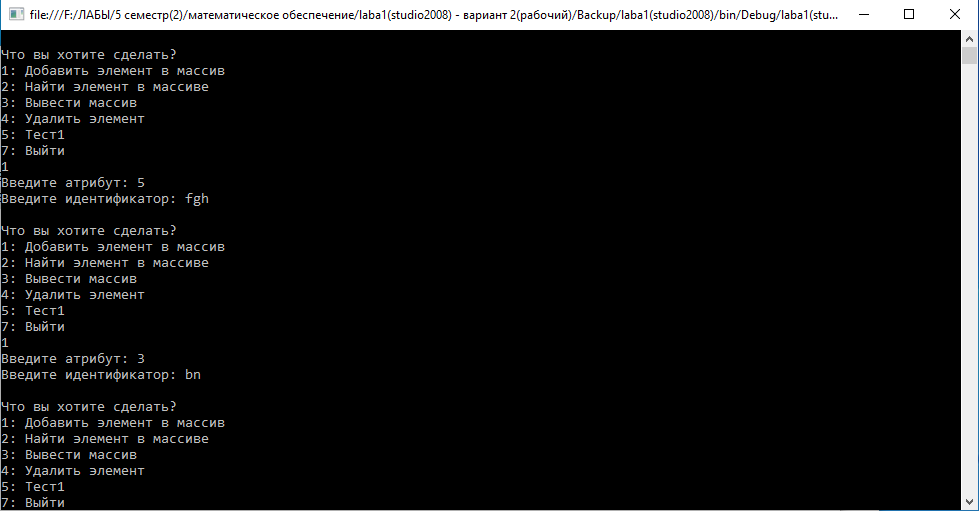
public void Del(string key) - функция удаляющая элемент из массива. После удаление элемента размер массива уменьшается, а сам он сдвигается влево с индекса удаленного элемента.

**Скриншоты работы программы**

Основное меню программы



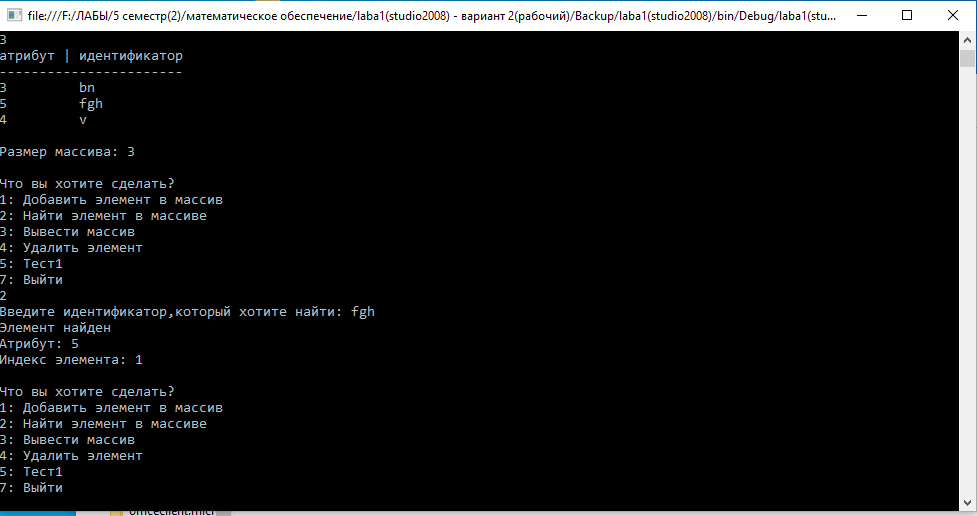
Добавление элемента



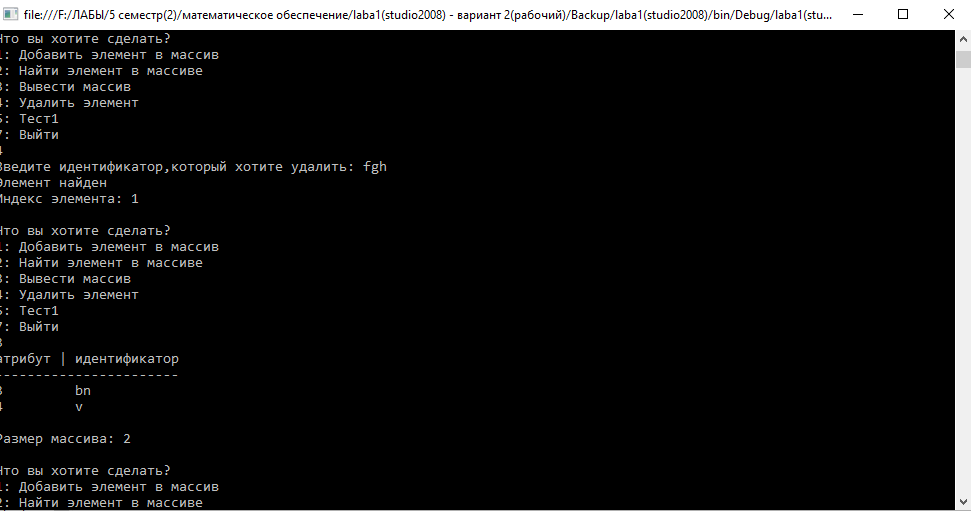
Вывод массива



Поиск элемента по идентификатору



Удаление элемента



**Тест**



**Листинг**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

// на данный момент идентификаторы добавляются с одним пробелом в конце

namespace laba1\_studio2008\_

{

public unsafe class Iden // char unsafe

{

public int atr;

public string id;

}

public unsafe class Mass // unsafe

{

public int size;

public Iden[] num2 = new Iden[100];

public Mass()

{

size = 0;

Iden myIden = new Iden();

myIden.atr = 0;

myIden.id = " ";

num2[0] = myIden;

}

~Mass()

{

}

public int Inde(char nu1)

{

int fin1 = 0;

int g = size;

string alph = "AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz ";

bool flag3 = true;

int k = 0;

while (flag3 == true)

{

if (nu1 == alph[k])

{

fin1 = k;

flag3 = false;

break;

}

else

k += 1;

if (alph[k] == ' ') // исключение

{

fin1 = -1;

flag3 = false;

break;

}

}

return fin1;

}

// проверить сортировку

public void Add(int el1, string el2)

{

char pred1;

char pred2;

int pr1 = 0;

int pr2 = 0;

int k = 0;

int index = 0; // номер буквы идентификатора

int index2 = 0;

el2 = el2.Trim(); // удаляем пробелы в начале и конце строки

el2 = el2 + ' '; //

int len\_el2 = el2.Length;

int k2 = 0;

bool flag = true;

string Ch = "";

// проверка на повторное вхождение идентификатора

while (k2 < size)

{

if (el2 == num2[k2].id)

{

flag = false;

}

k2 += 1;

}

if (flag == false)

{

return;

}

foreach (int i in el2)

{

k += 1;

}

if (size == 0) // можно убрать условие

{

Iden elem = new Iden();

elem.atr = el1;

elem.id = el2;

num2[size] = elem;

size += 1;

}

else

{

Iden elem = new Iden();

elem.atr = el1;

elem.id = el2;

num2[size] = elem;

int g = size;

char nu1 = num2[g - 1].id[0];

char nu2 = num2[g].id[0];

int fin1 = 0; // номер буквы в алфавите

int fin2 = 0;

bool flag2 = true;

bool sort = true;

while (flag2 == true)

{

// предварительная проверка

for (index2 = 0; index2 <= index; index2++)

{

// можно еще добавить проверку nu2

nu1 = num2[g - 1].id[index2];

if (nu1 == ' ')

{

flag2 = false;

break;

}

}

if (flag2 == false)

break;

nu1 = num2[g - 1].id[index];

nu2 = num2[g].id[index];

fin1 = Inde(nu1);

fin2 = Inde(nu2);

if (fin2 < fin1) // if (fin2 <= fin1)

{

// просматриваем предыдущую букву для правильной сортировки

if ((index > 0) & (g > 0))

{

pred1 = num2[g - 1].id[index - 1];

pred2 = num2[g].id[index - 1];

pr1 = Inde(pred1);

pr2 = Inde(pred2);

if (pr1 != pr2)

{

sort = false;

}

}

if (sort == true)

{

Iden elem2 = new Iden();

elem2 = num2[g - 1];

num2[g - 1] = elem;

num2[g] = elem2;

}

g -= 1;

if (g == 0)

{

flag2 = false;

}

}

else if (fin2 == fin1)

{

if (index >= k) // if (index >= k)

{

if (index > 0)

{

flag2 = false;

}

}

if (g == 0)

{

g += 1;

}

index += 1;

}

else if (fin2 > fin1)

{

k = 0;

g -= 1;

if (g == 0)

{

flag2 = false;

}

}

}

size += 1;

}

}

public void Vivod() // public void Vivod()

{

int k = 0;

Console.WriteLine("атрибут | идентификатор");

Console.WriteLine("-----------------------");

while (k < size)

{

Console.Write(num2[k].atr);

Console.Write(" ");

Console.WriteLine(num2[k].id);

k += 1;

}

Console.WriteLine("");

Console.WriteLine("Размер массива: {0} ", size);

}

public void Poisk2(string key)

{

bool flag = true;

int index2;

int fin1 = 0;

int fin\_key = 0;

int fin3 = 0;

int left = 0;

int right = size - 1; // int right = size + 1;

int mid = 0;

int mid2 = 0;

int pov = 0;

int search = -1;

int index = 0;

int g = size;

int key\_size = 0;

char st;

int st2;

char nu3 = num2[mid].id[index];

key = key + ' '; //

key\_size = key.Length;

//int right = key\_size - 1;

while (left < right)

{

mid = left + (right - left) / 2; //

nu3 = num2[mid].id[index];

fin1 = Inde(nu3);

fin\_key = Inde(key[index]);

if (key == num2[mid].id) // if (key == num[mid].id)

{

search = 1;

break;

}

if (index < key\_size) { st = key[index]; } //

else

{

search = -1;

break;

}

if (fin1 == fin\_key)

{

if (num2[mid].id[index] == ' ')

//Console.WriteLine("Подошли к границе");

index += 1;

//continue;

}

else if (fin\_key > fin1)

{

left = mid;

}

else if (fin\_key < fin1)

{

right = mid;

}

// когда результат работы свелкя к двум соседним индексам

if (right - left == 1)

{

if (key == num2[right].id)

{

search = 1;

mid = right;

break;

}

if (key == num2[left].id)

{

Console.WriteLine("123");

search = 1;

mid = left;

break;

}

search = -1;

break;

}

}

if (search == -1)

Console.WriteLine("Элемент не найден");

else

{

Console.WriteLine("Элемент найден");

Console.Write("Атрибут: ");

Console.WriteLine(num2[mid].atr);

Console.Write("Индекс элемента: ");

Console.WriteLine(mid);

}

}

public void Del(string key)

{

key = key.Trim();

int search = -1; // если search = 1 то идентификатор найден, если search = -1 то не найден

int inde = 0; // номер найденного идентификатора элемента

string Ch;

int k = 0;

int le;

key = key + ' '; //

int len\_key = key.Length;

string v1;

int v2;

while (k < size)

{

Ch = num2[k].id;

// //Ch = Ch + ' ';

if (key == Ch)

{

search = 1;

inde = k;

break;

}

k += 1;

}

if (search == 1)

{

Console.WriteLine("Элемент найден");

Console.WriteLine("Индекс элемента: {0}", inde);

for (le = inde; le < size - 1; le++)

{

v1 = num2[le + 1].id;

v2 = num2[le + 1].atr;

num2[le].id = v1;

num2[le].atr = v2;

num2[le + 1].id = "";

num2[le + 1].atr = 0;

}

size -= 1;

}

if (search == -1)

{

Console.WriteLine("Нельзя удалить несуществующий элемент");

}

}

}

class Program

{

static void Main(string[] args)

{

Mass L = new Mass();

bool flag = true;

while (flag == true)

{

Console.WriteLine("");

Console.WriteLine("Что вы хотите сделать?");

Console.WriteLine("1: Добавить элемент в массив");

Console.WriteLine("2: Найти элемент в массиве");

Console.WriteLine("3: Вывести массив");

Console.WriteLine("4: Удалить элемент");

Console.WriteLine("5: Тест1");

Console.WriteLine("7: Выйти");

string w = Console.ReadLine();

int w2 = Convert.ToInt32(w);

switch (w2)

{

case 1:

{

Console.Write("Введите атрибут: ");

string el = Console.ReadLine();

int el1 = Convert.ToInt32(el);

Console.Write("Введите идентификатор: ");

string el2 = Console.ReadLine();

L.Add(el1, el2);

break;

}

case 2:

{

Console.Write("Введите идентификатор,который хотите найти: ");

string el1 = Console.ReadLine();

L.Poisk2(el1);

break;

}

case 3:

{

L.Vivod();

Console.Write("");

break;

}

case 4:

{

Console.Write("Введите идентификатор,который хотите удалить: ");

string el1 = Console.ReadLine();

L.Del(el1);

break;

}

case 5:

{

L.Add(5, "gh");

L.Add(3, "ghnb");

L.Add(65, "v");

L.Add(3, "bnm");

L.Add(3, "bn");

L.Add(3, "bnmm");

L.Add(3, "ghn");

L.Add(3, "gha");

L.Add(3, "bnb");

L.Add(3, "ghhh");

L.Vivod();

break;

}

case 7:

{

flag = false;

break;

}

}

}

}

}

}