**Тестовые задания.**

Данные тестовые задания направлены на проверку уровня ваших знаний и навыков. Задания можно выполнять любым способом, в рамках правил описанных в самих заданиях. Задания, связанные с практическим выполнением, необходимо присылать в виде проекта(ов) приложенных к архиву с ответами на текстовые вопросы. На задание 2 ответы можно прислать в текстовом файле.

**Задание 1 (Логика)**

Перед вами таблица (20x20) с целым числом от 0 до 99 в каждой из клеток.

Задача - **найти 4 соседних числа (без разрыва цепочки, одно за другим), имеющих самое большое произведение.**

(цветом выделены различные варианты 4 соседних числел [соседними считаются два числа, расположенных не более чем на 1 клетку друг от друга])

*Приведите реализацию алгоритма поиска на языке C# [можно использовать и другой язык, но С# предпочтительнее]*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 08 | 02 | 22 | 97 | 38 | 15 | 00 | 40 | 00 | 75 | 04 | 05 | 07 | 78 | 52 | 12 | 50 | 77 | 91 | 08 |
| 49 | 49 | 99 | 40 | 17 | 81 | 18 | 57 | 60 | 87 | 17 | 40 | 98 | 43 | 69 | 48 | 04 | 56 | 62 | 00 |
| 81 | 49 | 31 | 73 | 55 | 79 | 14 | 29 | 93 | 71 | 40 | 67 | 53 | 88 | 30 | 03 | 49 | 13 | 36 | 65 |
| 52 | 70 | 95 | 23 | 04 | 60 | 11 | 42 | 69 | 24 | 68 | 56 | 01 | 32 | 56 | 71 | 37 | 02 | 36 | 91 |
| 22 | 31 | 16 | 71 | 51 | 67 | 63 | 89 | 41 | 92 | 36 | 54 | 22 | 40 | 40 | 28 | 66 | 33 | 13 | 80 |
| 24 | 47 | 32 | 60 | 99 | 03 | 45 | 02 | 44 | 75 | 33 | 53 | 78 | 36 | 84 | 20 | 35 | 17 | 12 | 50 |
| 32 | 98 | 81 | 28 | 64 | 23 | 67 | 10 | 26 | 38 | 40 | 67 | 59 | 54 | 70 | 66 | 18 | 38 | 64 | 70 |
| 67 | 26 | 20 | 68 | 02 | 62 | 12 | 20 | 95 | 63 | 94 | 39 | 63 | 08 | 40 | 91 | 66 | 49 | 94 | 21 |
| 24 | 55 | 58 | 05 | 66 | 73 | 99 | 26 | 97 | 17 | 78 | 78 | 96 | 83 | 14 | 88 | 34 | 89 | 63 | 72 |
| 21 | 36 | 23 | 09 | 75 | 00 | 76 | 44 | 20 | 45 | 35 | 14 | 00 | 61 | 33 | 97 | 34 | 31 | 33 | 95 |
| 78 | 17 | 53 | 28 | 22 | 75 | 31 | 67 | 15 | 94 | 03 | 80 | 04 | 62 | 16 | 14 | 09 | 53 | 56 | 92 |
| 16 | 39 | 05 | 42 | 96 | 35 | 31 | 47 | 55 | 58 | 88 | 24 | 00 | 17 | 54 | 24 | 36 | 29 | 85 | 57 |
| 86 | 56 | 00 | 48 | 35 | 71 | 89 | 07 | 05 | 44 | 44 | 37 | 44 | 60 | 21 | 58 | 51 | 54 | 17 | 58 |
| 19 | 80 | 81 | 68 | 05 | 94 | 47 | 69 | 28 | 73 | 92 | 13 | 86 | 52 | 17 | 77 | 04 | 89 | 55 | 40 |
| 04 | 52 | 08 | 83 | 97 | 35 | 99 | 16 | 07 | 97 | 57 | 32 | 16 | 26 | 26 | 79 | 33 | 27 | 98 | 66 |
| 88 | 36 | 68 | 87 | 57 | 62 | 20 | 72 | 03 | 46 | 33 | 67 | 46 | 55 | 12 | 32 | 63 | 93 | 53 | 69 |
| 04 | 42 | 16 | 73 | 38 | 25 | 39 | 11 | 24 | 94 | 72 | 18 | 08 | 46 | 29 | 32 | 40 | 62 | 76 | 36 |
| 20 | 69 | 36 | 41 | 72 | 30 | 23 | 88 | 34 | 62 | 99 | 69 | 82 | 67 | 59 | 85 | 74 | 04 | 36 | 16 |
| 20 | 73 | 35 | 29 | 78 | 31 | 90 | 01 | 74 | 31 | 49 | 71 | 48 | 86 | 81 | 16 | 23 | 57 | 05 | 54 |
| 01 | 70 | 54 | 71 | 83 | 51 | 54 | 69 | 16 | 92 | 33 | 48 | 61 | 43 | 52 | 01 | 89 | 19 | 67 | 48 |

**Задание 2 (Знание основ языка C# и .Net Framework)**

1. Каким способом можно оптимизировать следующий код:

string a = "";

for (int i = 0; i < 1000000; i++)

{

a += "a";

}

1. Перед вами два метода, оба из которых смогут скомпилироваться и корректно отработать при их вызове, но при этом один из них является избыточным с точки зрения использования возможностей языка C#. Вопрос – какой именно метод избыточен и почему?

private void Method1(ref List<int> listDigits)

{

listDigits.Add(55);

}

private void Method2(ref int digit)

{

digit += 55;

}

1. Есть следующий код. Как вы считаете, правильно ли он работает? Обоснуйте ваш ответ.

int i = 5;

var l = new List<int>{1,2,3,4};

double fraction = i/l.Count();

Console.WriteLine(fraction);

**Задание 3 (Реверс-инжиниринг)**

Есть вот такой декодер(C#), который «почему-то не декодирует».

|  |
| --- |
| **Код:** |

namespace xb9d8bb5e6df032aa

{

using System;

public class x1110bdd110cdcea4

{

public x1110bdd110cdcea4() : base()

{

}

public static string \_xaacba899487bce8c(string x5e99b576d2530d13, int x2710752c36f2d14b)

{

ushort usht;

char[] arrch;

int i;

ushort usht1;

string str;

usht = (ushort)x2710752c36f2d14b;

goto ILO\_007f;

while (true)

{

i++;

ILO\_000c:

if (i < (x5e99b576d2530d13.Length / 4))

{

break;

}

str = new string(arrch);

if (!0.Equals(0))

{

goto ILO\_008d;

}

return str;

ILO\_0027:

arrch[i] = (short)usht1;

usht += 1789;

}

usht1 = (ushort)((((x5e99b576d2530d13[4 \* i] - 'a') + ((x5e99b576d2530d13[(4 \* i) + 1] - 'a') << '\u0004')) + ((x5e99b576d2530d13[(4 \* i) + 2] - 'a') << '\u0008')) + ((x5e99b576d2530d13[(4 \* i) + 3] - 'a') << '\u000c'));

usht1 -= usht;

goto ILO\_0027;

ILO\_007f:

arrch = new char[x5e99b576d2530d13.Length / 4];

ILO\_008d:

i = 0;

if ((((((uint)usht)) + (((uint)usht1))) > -1))

{

goto ILO\_008d;

}

goto ILO\_000c;

return str;

}

}

}

Нужно восстановить исходную структуру декодера, избавиться от циклов GOTO и привести полученный код и декодированную строку.

Строка для декодирования:

MessageBox.Show(string.Intern(x1110bdd110cdcea4.\_xaacba899487bce8c("eflohgcpkgjpegaapahapboajbfbdbmbhedcbekcgfbdkphdlepdgegekdnepcefhdlfiobggcjgadahd" +

"chhmnnhncfiobmiibdjanjjcoakkmhklapkfaglkbnloldmcalmkacngpinfppnepgomknojpeplolpf" +

"pcajojamoabonhbhjobmnfccnmcboddlikdnmbedniegmpeemgfamnfhiegghlghkchiljhglainlhih" +

"goififjdkmjnkdkalkkmkblffilehplojgmeknmjeennjlnmicomijojiapkdhpihopaifaiimapgdbe" +

"ikblhbcfcicegpceggdmbndmgeeoflecgcfhgjflfagkahgicogcdfhfcmhopcipekiaebjkdijeepje" +

"dgkmomkaeelpcllpccmmcjmnnpmmbhnmboneneocbmoomcppakpnababbiadapaibgbmlmbiaecmpkcd" +

"lbddmidnkpdnpgemonehoefaplfmocgpnjgloahfjhhdnohdofimimimndjknkjembkcnikbnpkpmglh" +

"mnlciemlflmifcnekjnhlaoklhocgoomkfpblmpmkdakkkadfbbakibckpbmjgckincmiedcildbjcea" +

"ijeidafbihflhofchfgbhmghhdhjhkhigbidhiinbpioggjggnjebekfflkhfclofjljeamhehmoeome" +

"efnmplnnedoodkoidbpaphpbeppmdgaadnafcebnclbonbcacjcgcadmbhdlbodnbfebbmejmcfhakfb" +

"bbgeaignlognpfhganhcaeifalihpbjjpijnppjkogkcknkepelpollcocmeojmknanajhnbooncnfom" +

"mmoeidpemkpnmbajmiammpahmgbbmnbllecpllcjgcdpkjdblaeaghegloejkffmkmfgkdgbfkgcjbha" +

"jihojphkigioinioiejmiljgicklhjkbialeihlncolkhfmjhmmdhdnlgkndgboofiomgpoiggpacnpp" +

"aeabelamfcblejbdaacefhcfeocpdfdhpldaedekdkemdbfgdifldpffofgecngecehiclhiccincjid" +

"npiechjmbojkmekoamkjadlfbklolamhaimbapmipfnhpmnnpdoppkooobpgkipdpppcpgagonanneba" +

"olbcoccinjcoiadmmhdgnodjmfecimeomdfomkfnlbglligfmpgjlghllnhbleihglinkcjpkjjekakk" +

"khkkkokckfldjmldkdmfjkmijbnejinbfpnaegoggnoaiephdlpnicaaijadiablchbphobggfcahmcd" +

"hddmbkdkfbeegiedbpeffgffgnfofegmelgcfchefjhheaikehigeoifpejidmjoddkodkkjoallcilj" +

"dplbcgmjcnmpceniblniccokbjonbapjbhpennpfmeacbmaladbbakbkpacjaicipocalfdapmdjpdef" +

"pkeipbfcoifbopfpoggijngkoehfolhpicibojimnajpmhjbnojhmfknhmkomdlplkljlbmbhimklpme" +

"lgnglnnaleofllopfcpojjpojaackhackoahkfblfmb", 7400187)));

**Задание 4 (Алгоритмы, циклы, рекурсии)**

Имеется исходный код, который с помощью рекурсии выводит имена последних детей в каждой ветви.

Необходимо переписать метод RecursFillLastChilds что бы он стал не-рекурсивным, при этом сохранился исходный принцип работы метод.

*Cортировать коллекцию SomeData вне метода RecursFillLastChilds – нельзя!*

**Условие:** При добавлении новых записей в SomeData - разработанный алгоритм должен сохранить работоспособность.

public static void Main()

{

Dictionary<string, List<string>> someData = new Dictionary<string, List<string>>();

someData["Петр Иванович"] = new List<string>();

someData["Петр Иванович"].Add("Анна Петровна");

someData["Петр Иванович"].Add("Максим Петровна");

someData["Максим Иванович"] = new List<string>();

someData["Максим Иванович"].Add("Андрей Максимович");

someData["Максим Иванович"].Add("Миша Максимович");

someData["Максим Иванович"].Add("Николай Максимович");

someData["Анна Ивановна"] = new List<string>();

someData["Анна Ивановна"].Add("Артем");

someData["Николай Максимович"] = new List<string>();

someData["Николай Максимович"].Add("Катя");

someData["Николай Максимович"].Add("Маша");

someData["Анна Ивановна"].Add("Саша");

List<string> lastChilds = new List<string>();

RecursFillLastChilds("Петр Иванович", someData, lastChilds);

foreach (var child in lastChilds)

{

Console.WriteLine(child);

}

}

private static void RecursFillLastChilds(string parentName, Dictionary<string, List<string>> someData, List<string> lastChilds)

{

if (!someData.ContainsKey(parentName))

{

lastChilds.Add(parentName);

return;

}

foreach (string child in someData[parentName])

{

RecursFillLastChilds(child, someData, lastChilds);

}

}

**Задание 5 (База данных)**

Есть две таблицы:

**Табл.1** Сотрудники отдела продаж

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ФИО** | **Квалификация** | **Отдел** | **Стаж, лет** | **Личный транспорт** |
| Иван Иванов | Ст. Менеджер | Отдел продаж | 2 | Есть |
| Петр Петров | Ср. Менеджер | Отдел продаж | 5 | нет |
| Иван Степанович | Мл. Менеджер | Бухгалтерия | 7 | Есть |

**Табл.2** Сотрудники отдела кадров

|  |  |  |
| --- | --- | --- |
| **ФИО** | **Стаж, лет** | **Номер телефона** |
| Иван Фомич | 1 | 8800553535 |
| Владимир Петрович | 7 | 8800458521 |

1. Составьте структуру базы данных, для приведенных выше таблиц с данными, нормализованную до 3-й нормальной формы (сокращённо 3NF).
2. Составьте запросы для каждой из исходных таблиц (структура вывода данных как в Табл.1, Табл2), позволяющие получить все данные, но уже из нормализованных таблиц спроектированной базы данных.
3. Составьте запрос(ы) для переноса данных сотрудника из одного отдела в другой.
4. Составьте запрос(ы) позволяющие хранить и получать данные в формате *«ФИО сотрудника, дата рождения сотрудника»*

- дата рождения сотрудника изначально отсутствует в исходных данных;

- запрещено менять структуру таблиц, разработанных на этапе 1.

**Уточнение:** При решении пункта 4 - доработайте модель базы данных, разработанную на первом этапе данного задания, таким образом, ***чтобы можно было добавлять новые информационные атрибуты к любой таблице из пункта 1*** (к примеру – номер паспорта, серия паспорта, место рождения…) без модификации структуры базы данных под каждый новый информационный атрибут. (Уточнение: не добавлять новые колонки под каждый информационный атрибут. Необходимо именно доработать структур базы данных, и после этого иметь возможность добавлять новые информационные атрибуты для объектов из существующих таблиц базы данных, в доработанную структуру базы данных без модификации схемы базы данных, таблиц).