# 2D MASIVI ILGVARS LICKRASTINS 110 GRUPA:

# 1Uzdevums: Tabulas aizpilde izmantojot 2D masīvu

* Izveido mainīgo vards, kas PRASA ievadīt Vārdu
* Izveido mainīgo uzvards, kas PRASA ievadīt Uzvārdu
* Izvada no mainīgā vārdu un tā garumu (Katrs **otrais** vārda burts LIELAIS)
* Izvada tabulu, kur pirmā kolonā pirmā rindā ir “X”, bet pārejās ir uzvārds (Katrs **otrais** vārda burts LIELAIS)
* Pēc tam pirmo rindu papildina ar sum, min, max
* NĀKAMĀS rindās tabulas pirmās rindā, sākot ar 2 pozīciju ir Uzvārda burti no beigām(Katrs **otrais** vārda burts LIELAIS, net no sākuma),
* Tad aizpilda tabulu ar 2D masīva vērtībām
* tad sum, min, max
* Pārējās rindās zem vārda burtiem ir GADĪJUMA skaitlis robežās no Varda garuma līdz Varda+Uzvarda garumam (ieskaitot)
* Tālāk rindā ievada aprēķinus sum, min, max
* Zem kolonām sum, min, max raksta atbilstošo vērtību, bet pārējās zvaigznīti
* Tabulai iz malas, kas veidotas ar +,-,| simboliem datos, bet aprēķinos \*,=,|
* Izveido otru masīvu., kam samainītiem datiem KOLONU un RINDU vietām
* Pēc tam izvada tabulu no reversā masīva, kur vārdu un uzvārdu vietām

using System;

using System.Linq;

class Program

{

static void Main(string[] args)

{

Console.Write("Ievadi savu vārdu: ");

string vards = Console.ReadLine();

Console.Write("Ievadi savu uzvārdu: ");

string uzvards = Console.ReadLine();

string formatetsVards = FormatetKatruOtroBurtu(vards.ToLower());

string apgrieztsUzvards = new string(uzvards.ToLower().Reverse().ToArray());

string formatetsUzvards = FormatetKatruOtroBurtu(apgrieztsUzvards);

int rindas = formatetsUzvards.Length;

int kolonnas = formatetsVards.Length;

int[,] dati = AizpilditMasivu(rindas, kolonnas, vards.Length, vards.Length + uzvards.Length);

IzvadeTabula(formatetsVards, formatetsUzvards, dati);

int[,] transponets = TransponetMasivu(dati);

Console.WriteLine("\n--- TRANSPONĒTA TABULA ---\n");

IzvadeTabula(formatetsUzvards, formatetsVards, transponets);

}

static string FormatetKatruOtroBurtu(string teksts)

{

string rezultats = "";

for (int i = 0; i < teksts.Length; i++)

{

rezultats += (i % 2 == 0) ? Char.ToUpper(teksts[i]) : Char.ToLower(teksts[i]);

}

return rezultats;

}

static int[,] AizpilditMasivu(int rindas, int kolonnas, int min, int max)

{

Random rnd = new Random();

int[,] masivs = new int[rindas, kolonnas];

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

{

for (int j = 0; j < kolonnas; j++)

{

masivs[i, j] = rnd.Next(min, max + 1);

}

}

return masivs;

}

static void IzvadeTabula(string kolVirsraksts, string rindVirsraksts, int[,] dati)

{

int rindas = dati.GetLength(0);

int kolonnas = dati.GetLength(1);

string sep = "+";

for (int i = 0; i < kolonnas + 4; i++) sep += "-------+";

Console.WriteLine(sep);

Console.Write("| X ");

foreach (char burts in kolVirsraksts)

{

Console.Write($"| {burts} ");

}

Console.Write("| sum | min | max |");

Console.WriteLine();

Console.WriteLine(sep);

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

{

Console.Write($"| {rindVirsraksts[i]} ");

int sum = 0, min = int.MaxValue, max = int.MinValue;

for (int j = 0; j < kolonnas; j++)

{

int val = dati[i, j];

sum += val;

min = Math.Min(min, val);

max = Math.Max(max, val);

Console.Write($"| {val.ToString().PadLeft(2).PadRight(5)}");

}

Console.Write($"| {sum.ToString().PadLeft(3).PadRight(5)}");

Console.Write($"| {min.ToString().PadLeft(3).PadRight(5)}");

Console.Write($"| {max.ToString().PadLeft(3).PadRight(5)}|");

Console.WriteLine();

Console.WriteLine(sep);

}

// SUM KOLONNU

Console.Write("| sum ");

for (int j = 0; j < kolonnas; j++)

{

int sum = 0;

for (int i = 0; i < rindas; i++) sum += dati[i, j];

Console.Write($"| {sum.ToString().PadLeft(2).PadRight(5)}");

}

Console.Write("| \* | \* | \* |");

Console.WriteLine();

Console.WriteLine(sep.Replace('+', '\*').Replace('-', '='));

// MIN KOLONNU

Console.Write("| min ");

for (int j = 0; j < kolonnas; j++)

{

int min = int.MaxValue;

for (int i = 0; i < rindas; i++) min = Math.Min(min, dati[i, j]);

Console.Write($"| {min.ToString().PadLeft(2).PadRight(5)}");

}

Console.Write("| \* | \* | \* |");

Console.WriteLine();

Console.WriteLine(sep.Replace('+', '\*').Replace('-', '='));

// MAX KOLONNU

Console.Write("| max ");

for (int j = 0; j < kolonnas; j++)

{

int max = int.MinValue;

for (int i = 0; i < rindas; i++) max = Math.Max(max, dati[i, j]);

Console.Write($"| {max.ToString().PadLeft(2).PadRight(5)}");

}

Console.Write("| \* | \* | \* |");

Console.WriteLine();

Console.WriteLine(sep.Replace('+', '\*').Replace('-', '='));

}

static int[,] TransponetMasivu(int[,] masivs)

{

int rindas = masivs.GetLength(0);

int kolonnas = masivs.GetLength(1);

int[,] jauns = new int[kolonnas, rindas];

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

{

for (int j = 0; j < kolonnas; j++)

{

jauns[j, i] = masivs[i, j];

}

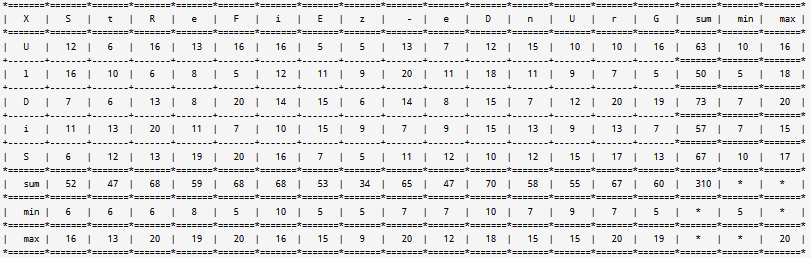
}

return jauns;

}

}

# 



# Uzdevums- robainie masīvi.

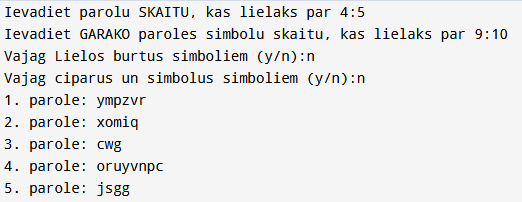
Nodefinēt masīvu ar garumu ne mazāku par 5 vērtībām (lietotājs ievada), kurā glabāsies ģenerētās paroles

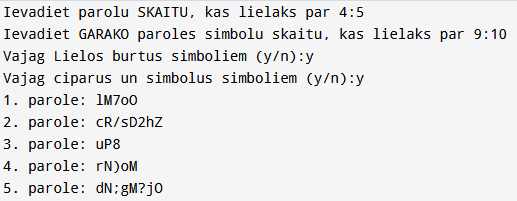
Katra masīva vērtība ir gadījuma virkne, kas sastāv PAMATĀ no MAZIEM burtiem

Garāko masīva vērtību ierobežo lietotājs, bet ne mazāk 9 vērtībām

Pēc lietotāja apstiprinājuma tiek pievienoti gan LIELIE gan SIMBOLI

Izmantojot *foreach* ciklu, izvadīt masīva vērtības uz ekrāna.





using System;

using System.Linq;

class Program

{

static void Main()

{

string burtiMazie = "abcdefghijklmnopqrstuvwxyz";

string burtiLielie = burtiMazie.ToUpper();

string simboliCipari = "1234567890!@#$%^&\*()-\_=+{}[]|\\:;\"'<>,./?~`";

string visiSimboli = burtiMazie;

bool lietotLielos = false;

bool lietotSpec = false;

char ievadeLielie = ' ';

char ievadeSpec = ' ';

int skaitsParoles = 0;

int simboluGarums = 0;

Random nejauši = new Random();

// Prasa paroles skaitu

while (true)

{

Console.Clear();

Console.Write("Cik daudz paroles (vairāk par 4): ");

skaitsParoles = Convert.ToInt32(Console.ReadLine());

if (skaitsParoles > 4)

break;

}

// Prasa paroles simbolu skaitu

while (true)

{

Console.Clear();

Console.Write("Paroles garums (vairāk par 9): ");

simboluGarums = Convert.ToInt32(Console.ReadLine());

if (simboluGarums > 9)

break;

}

// Prasa par lielajiem burtiem

while (true)

{

Console.Clear();

Console.Write("Vai izmantot lielos burtus? (y/n): ");

ievadeLielie = char.ToLower(Console.ReadKey().KeyChar);

Console.WriteLine();

if (ievadeLielie == 'y')

lietotLielos = true;

if (ievadeLielie == 'y' || ievadeLielie == 'n')

break;

}

// Prasa par simboliem/cipariem

while (true)

{

Console.Clear();

Console.Write("Vai iekļaut simbolus un ciparus? (y/n): ");

ievadeSpec = char.ToLower(Console.ReadKey().KeyChar);

Console.WriteLine();

if (ievadeSpec == 'y')

lietotSpec = true;

if (ievadeSpec == 'y' || ievadeSpec == 'n')

break;

}

// Pievieno izvēlētos simbolus

if (lietotLielos)

visiSimboli += burtiLielie;

if (lietotSpec)

visiSimboli += simboliCipari;

// Ģenerē un izvada paroles

string[] izveidotParoles = new string[skaitsParoles];

for (int k = 0; k < skaitsParoles; k++)

{

izveidotParoles[k] = "";

for (int s = 0; s < nejauši.Next(5, simboluGarums); s++)

{

int index = nejauši.Next(visiSimboli.Length);

string viensSimbols = visiSimboli.Substring(index, 1);

izveidotParoles[k] += viensSimbols;

}

}

for (int u = 0; u < skaitsParoles; u++)

{

Console.WriteLine($"{u + 1}. Parole: {izveidotParoles[u]}");

}

}

}