**Лабораторна робота № 3.**

**ГЕНЕРАЦІЯ КОМБІНАТОРНИХ КОНФІГУРАЦІЙ**

**Мета:** на практицінавчитися генерації комбінаторних конфігурацій.

**Варіант № 24**

1. Скільки різних «слів» можна скласти з слова: а) «таблиця» ; б)

«тактика»

Таблиця

А) 7!= 5040 слів

Тактика – т(2), а(2), к(2), и(1)

Б) = 630 слів

2. На міжвузівську конференцію необхідно представити п’ять

студентів з доповідями з п’яти даних тем. У ЗНТУ після проведення

відбору між студентами усіх факультетів було відібрано 12

кандидатів. Скількома способами серед них можна вибрати п’ятьох

студентів для підготовки необхідних докладів з врахуванням

розподілу тем?

способи

3. Скільки можна побудувати різних прямокутників, довжини

сторін яких виражаються натуральними числами від 1 до 7?

*комбінації*

4. У вазі стоїть пронумеровані 12 червоних і 7 рожевих гвоздик.

Скількома способами можна вибрати з вази п’ять квітів так, щоб усі

вони були одного кольору?

*=813*

5. Скількома способами можна переставити букви в слові

«оборонятися», щоб однакові букви не стояли поряд?

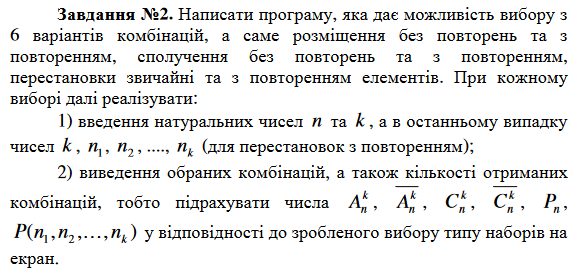
Оборонятися

О(3), я(2)

P(3,2,1,1,1,1,1,1)= (3+2+1+1+1+1+1+1)!/3!2!=3326400 способи.

6. Скількома способами можна розділити 10 різних чашок, та 10 різних тарілок на п’ять наборів по дві чашки та дві тарілки?

(()/5!)*²=((45\*28\*15\*6\*1)/120)²=893025 способами*  
7. У гардероб здають пальта 5 чоловік. Назад їм вертають пальта навмання. Скільки способів видачі пальт так, щоб не одно з них не досталося хазяїну?



Лістинг програми:

using Lab3.Properties;

using SetLibrary;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

enum DistributionsEnum

{

orderedDistribution,

orderedDistributionRep,

unorderedDistribution,

unorderedDistributionRep,

permutation,

permutationRep

}

namespace Lab3

{

public partial class Form1 : Form

{

private DistributionsEnum \_selectedAction = DistributionsEnum.orderedDistribution;

private Point labelEqualStartPos;

private Point textboxResultStartPos;

private Size textBoxNStartSize;

public Form1()

{

InitializeComponent();

labelEqualStartPos = labelEquals.Location;

textboxResultStartPos = textBoxResult.Location;

textBoxNStartSize = textBoxM.Size;

}

private void FormLab3\_Load(object sender, EventArgs e)

{

textBoxM.LostFocus += OnLostFocus;

textBoxM.GotFocus += OnGotFocus;

textBoxN.LostFocus += OnLostFocus;

textBoxN.GotFocus += OnGotFocus;

textBoxM.Text = "3";

textBoxN.Text = "4";

var actions = new string[] {DistributionsEnum.orderedDistribution.ToString(), DistributionsEnum.orderedDistributionRep.ToString(),

DistributionsEnum.unorderedDistribution.ToString(), DistributionsEnum.unorderedDistributionRep.ToString(),

DistributionsEnum.permutation.ToString(), DistributionsEnum.permutationRep.ToString() };

comboBoxAction.Items.AddRange(actions);

comboBoxAction.SelectedIndex = 0;

}

private void FormLab3\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void OnLostFocus(object sender, EventArgs e)

{

var textBox = sender as TextBox;

if (string.IsNullOrEmpty(textBox.Text))

textBox.Text = GetPlaceholder(textBox.Name);

}

private void OnGotFocus(object sender, EventArgs e)

{

var textBox = sender as TextBox;

if(textBox.Text == GetPlaceholder(textBox.Name))

textBox.Text = "";

}

private string GetPlaceholder(string elemName)

{

switch (elemName)

{

case nameof(textBoxM): return "m";

case nameof(textBoxN): return "n";

default: return "";

}

}

private void comboBoxAction\_SelectedIndexChanged(object sender, EventArgs e)

{

var comboBox = sender as ComboBox;

\_selectedAction = (DistributionsEnum)comboBox.SelectedIndex;

ChangePicture(\_selectedAction);

AdjustUI(\_selectedAction);

Culculate();

}

private void ChangePicture(DistributionsEnum action)

{

pictureBoxAction.Image = action switch

{

DistributionsEnum.orderedDistribution => Resources.orderedDistribution,

DistributionsEnum.orderedDistributionRep => Resources.orderedDistributionRep,

DistributionsEnum.unorderedDistribution => Resources.unorderedDistribution,

DistributionsEnum.unorderedDistributionRep => Resources.unorderedDistributionRep,

DistributionsEnum.permutation => Resources.permutation,

DistributionsEnum.permutationRep => Resources.permutation,

\_ => Resources.orderedDistribution,

};

}

private void AdjustUI(DistributionsEnum action)

{

if(action != DistributionsEnum.permutationRep)

{

labelEquals.Location = labelEqualStartPos;

textBoxResult.Location = textboxResultStartPos;

textBoxN.Size = textBoxNStartSize;

textBoxM.Visible = true;

textBoxN.TextAlign = HorizontalAlignment.Center;

textBoxM.TextAlign = HorizontalAlignment.Center;

}

if(action == DistributionsEnum.permutation)

{

textBoxM.Visible = false;

}

else if(action == DistributionsEnum.permutationRep)

{

textBoxM.Visible = false;

textBoxN.Size = new Size(textBoxN.Size.Width + 100, textBoxN.Size.Height);

labelEquals.Location = new Point(labelEquals.Location.X + 100, labelEquals.Location.Y);

textBoxResult.Location = new Point(textBoxResult.Location.X + 100, textBoxResult.Location.Y);

textBoxN.TextAlign = HorizontalAlignment.Left;

textBoxM.TextAlign = HorizontalAlignment.Left;

}

}

private void Culculate()

{

try

{

if (\_selectedAction != DistributionsEnum.permutation &&

\_selectedAction != DistributionsEnum.permutationRep)

{

float.TryParse(textBoxM.Text, out float m);

float.TryParse(textBoxN.Text, out float n);

if (m > n || m < 0 || n < 0)

throw new Exception();

switch (\_selectedAction)

{

case DistributionsEnum.orderedDistribution:

textBoxResult.Text = Distributions.OrderedDistribution(m, n).ToString();

break;

case DistributionsEnum.orderedDistributionRep:

textBoxResult.Text = Distributions.OrderedDistributionRep(m, n).ToString();

break;

case DistributionsEnum.unorderedDistribution:

textBoxResult.Text = Distributions.UnorderedDistribution(m, n).ToString();

break;

case DistributionsEnum.unorderedDistributionRep:

textBoxResult.Text = Distributions.UnorderedDistributionRep(m, n).ToString();

break;

}

}

else

{

switch (\_selectedAction)

{

case DistributionsEnum.permutation:

if (!float.TryParse(textBoxN.Text, out float n))

throw new Exception();

textBoxResult.Text = Distributions.Permutation(n).ToString();

break;

case DistributionsEnum.permutationRep:

var numStringList = textBoxN.Text.Split(',');

var numList = new List<float>();

foreach (var numSt in numStringList)

numList.Add(float.Parse(numSt));

textBoxResult.Text = Distributions.PermutationRep(numList).ToString();

break;

}

}

}

catch

{

textBoxResult.Text = "ERROR";

}

}

private void textBoxM\_TextChanged(object sender, EventArgs e)

{

Culculate();

}

}

}

Результат виконання програми:

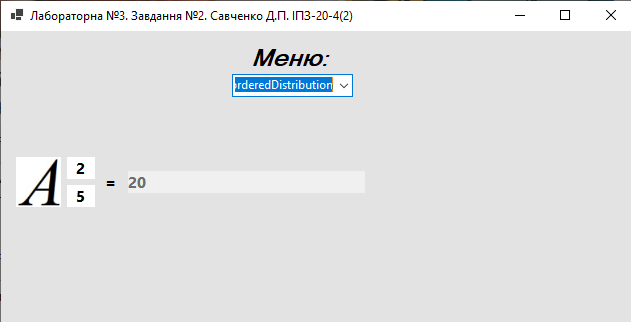


Рис.1. Результат виконання програми

Результат виконання програми:

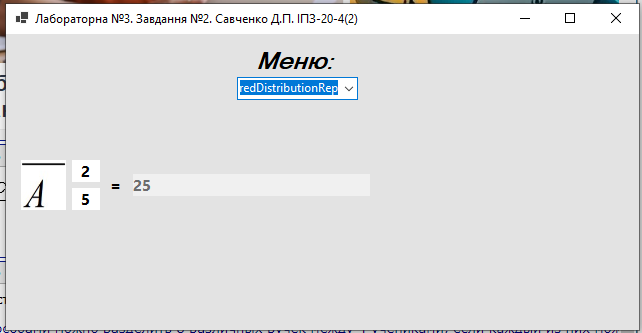


Рис.2. Результат виконання програми

Результат виконання програми:

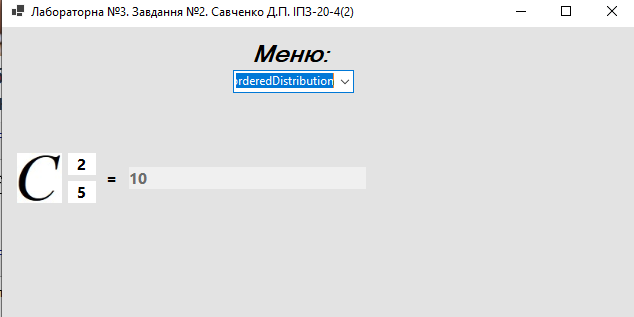


Рис.3. Результат виконання програми

Результат виконання програми:

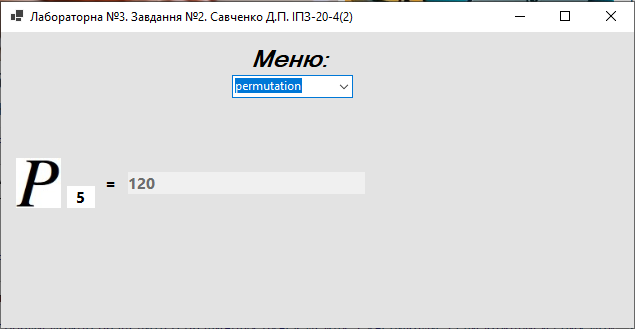


Рис.4. Результат виконання програми

Результат виконання програми:

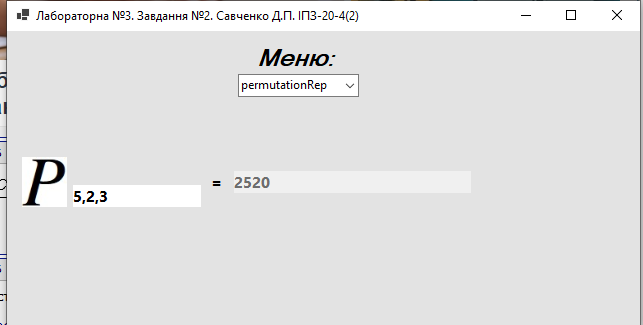


Рис.5. Результат виконання програми

**Висновок**: під час виконання лабораторної роботи я на практиці навчився будувати матриці відношень та визначати тип відношень.