Індивідуальне завдання № 1

Виконав: Грубий Назар, ПМІ-44.

На підставі наведеного нижче алгоритму для виявлення ярусів ЯПФ написати програму та реалізувати її на ПК.

Крок 1: номери вхідних вершин занести в масив *S*1.
Крок 2: знайти вершини, залежні (за ребрами) від вершин, які входять лише в масив *S*1; занести знайдені вершини в деякий масив *S*2.
Крок 3: якщо масив *S*2 містить вершини, то додаємо ці вершини в масив *S*1 і переходимо на крок 2, інакше – закінчуємо роботу.

Програма написана за допомогою мови програмування C#.

Код програми:

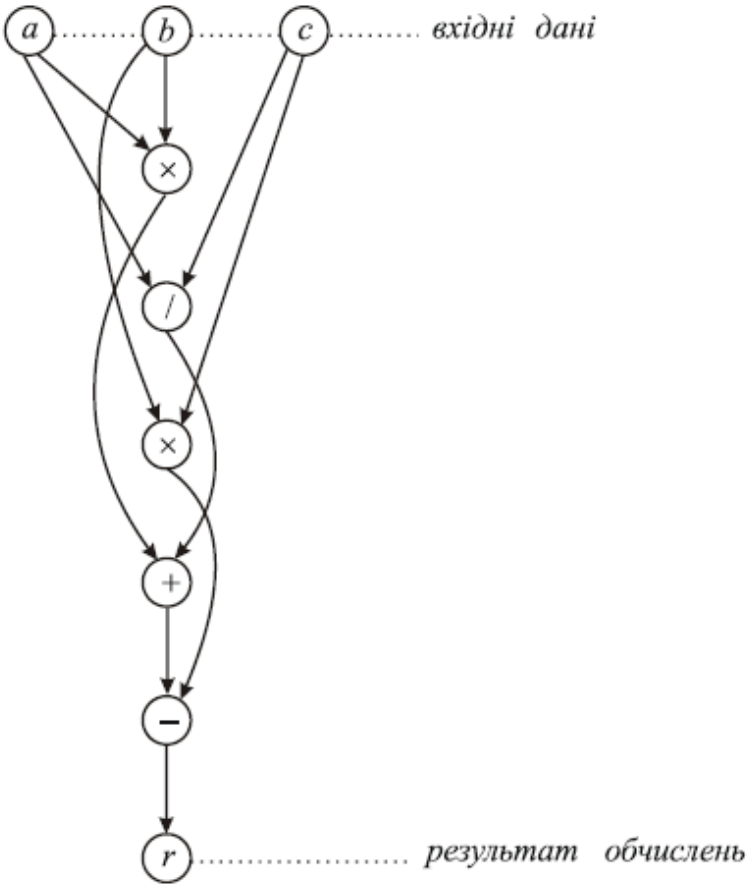
```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Individual_task__1
{
    class Program
    {
        static void Main(string[] args)
        {
            // приклад  1
            //int[,] adjacencyMatrix =
            //{
            //    { 0,0,0,1,0 },
            //    { 0,0,0,1,0 },
            //    { 0,0,0,0,1 },
            //    { 0,0,0,0,1 },
            //    { 0,0,0,0,0 },
            //};

            // приклад  2
            int[,] adjacencyMatrix =
            {
                { 0,1,0,0,0,0 },
                { 0,0,1,0,0,0 },
                { 0,0,0,1,0,0 },
                { 0,0,0,0,0,1 },
                { 0,0,0,0,0,1 },
                { 0,0,0,0,0,0 },
            };

            int n = adjacencyMatrix.GetLength(0);

            List<int> S1 = new List<int>();
            List<int> S2 = new List<int>();
            List<List<int>> layers = new List<List<int>>();
```

```
// Крок 1
for (int i = 0; i < n; i++)
{
    bool independent = true;
    for (int j = 0; j < n; j++)
    {
        if (adjacencyMatrix[j, i] == 1)
        {
            independent = false;
            break;
        }
    }

    if (independent == true)
        S1.Add(i);
}

layers.Add(S1.ToList());

// Кроки 2 та 3
do
{
    S2.Clear();

    for (int i = 0; i < n; i++)
    {
        if (S1.Contains(i))
            continue;

        bool dependsOnS1Only = true;
        for (int j = 0; j < n; j++)
        {
            if (adjacencyMatrix[j, i] == 1 && !S1.Contains(j))
            {
                dependsOnS1Only = false;
                break;
            }
        }

        if (dependsOnS1Only == true)
            S2.Add(i);
    }

    if (S2.Count > 0) layers.Add(S2.ToList());
    else break;

    foreach (int vertex in S2)
    {
        S1.Add(vertex);
    }

} while (true);

Console.WriteLine("Результати:");
for (int i = 0; i < layers.Count; i++)
```

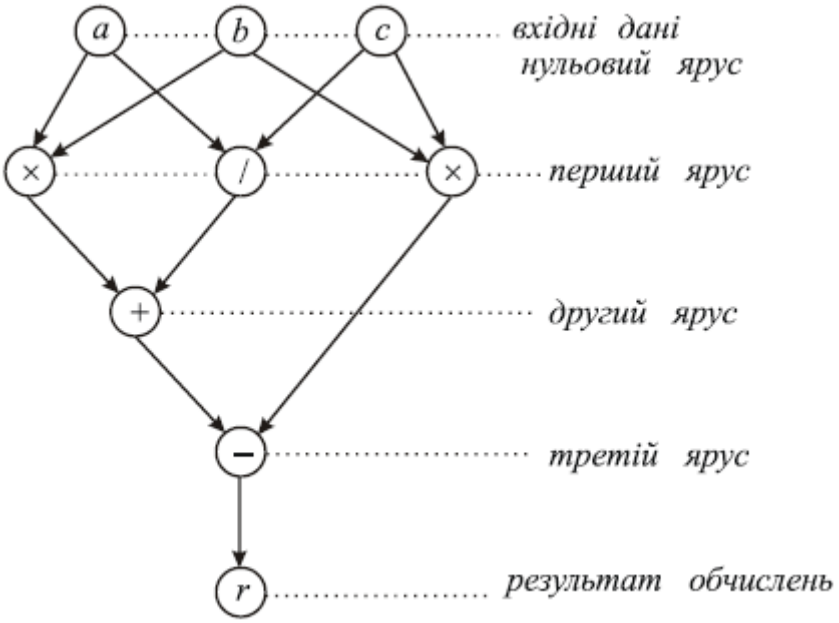
```
            {
                Console.Write($"Ярус {i}: ");
                for (int j = 0; j < layers[i].Count; j++)
                {
                    Console.Write($"{layers[i][j]} ");
                }
                Console.WriteLine();
            }

            Console.WriteLine("Натисніть будь-яку клавішу, щоб завершити
програму...");
            Console.ReadKey();
        }
    }
}
```

Приклад 1.

Розглянемо алгорим обчислень за формулою $r = a \times b + a / c - b \times c$.
Граф цього алгоритму має наступний вигляд:



Відповідна ЯПФ:



Матриця суміжності ГА виглядатиме таким чином:

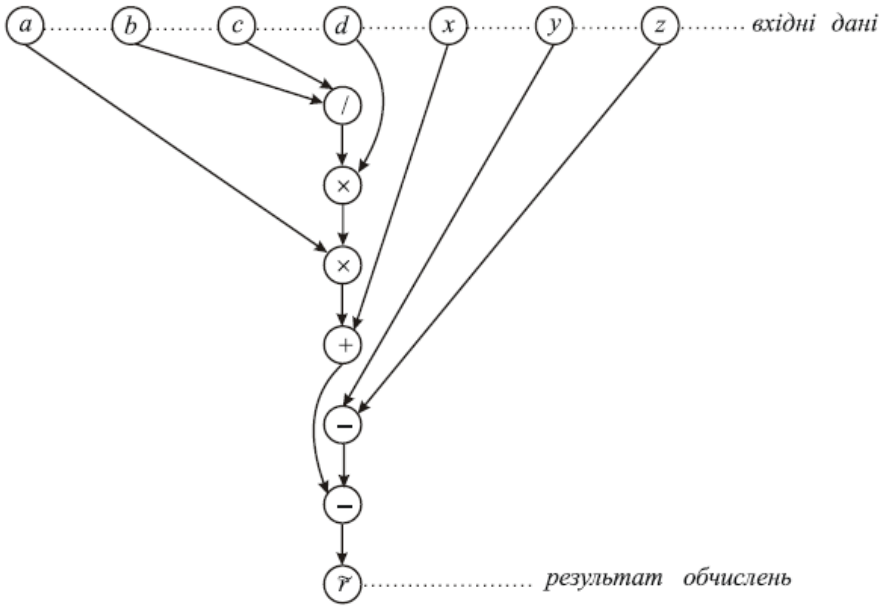
	*	/	*	+	-
*	0	0	0	1	0
/	0	0	0	1	0
*	0	0	0	0	1
+	0	0	0	0	1
-	0	0	0	0	0

Результати запуску програми:

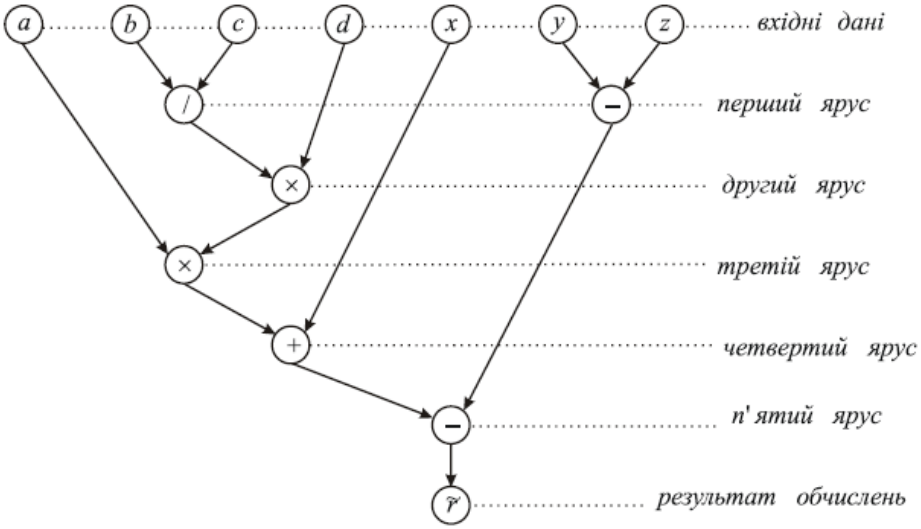
```
E:\C# projects\Paralelni\Individual task №1\bin\Debug\Individual task №1.exe
Результати:
Ярус 0: 0 1 2
Ярус 1: 3
Ярус 2: 4
Натисніть будь-яку клавішу, щоб завершити програму...
```

Приклад 2.

Розглянемо алгорим обчислень за формулою $\tilde{r} = (x + (a \times ((b / c) \times d))) - (y - z)$.
Граф цього алгоритму має наступний вигляд:



Відповідна ЯПФ:



Матриця суміжності ГА виглядатиме таким чином:

	/	*	*	+	-	-
/	0	1	0	0	0	0
*	0	0	1	0	0	0
*	0	0	0	1	0	0
+	0	0	0	0	0	1
-	0	0	0	0	0	1
-	0	0	0	0	0	0

Результати запуску програми:

```
E:\C# projects\Paralelni\Individual task №1\bin\Debug\Individual task №1.exe
Результати:
Ярус 0: 0 4
Ярус 1: 1
Ярус 2: 2
Ярус 3: 3
Ярус 4: 5
Натисніть будь-яку клавішу, щоб завершити програму...
```

Як ми бачимо, в обох прикладах програмний алгоритм працює правильно та отримані результати співпадають з очікуваними.