Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

C#.

Виконав:

студент групи ІП-32

Ковальчук О. М.

Перевірив:

Корочкін О. В.

Київ - 2015

Лабораторна робота №5. C#

Мета роботи: вивчення засобів мови C# для роботи з потоками

Мова програмування: C#

Завдання: Розробити програму, що містить паралельні потоки, кожен з яких реалізує функції F1, F2, F3 з лабораторної роботи №1. Вимоги до потоків такі ж, як в лабораторній роботі № 2.

Функції:
F1: C = A - B * (MA * MD)
F2: o = Min(MK * MM)
F3: T = (MS * MZ) * (W + X)

Лістинг програми

05-csharp/05-csharp/Program.cs

```
using System;
using System.Collections.Generic;
using System.Threading;

/**
 * Parallel programming
 * Lab 5
 *
 * Functions:
 * F1: C = A - B * (MA * MD)
 * F2: o = Min(MK * MM)
 * F3: T = (MS * MZ) * (W + X)
 *
 * @since 2015-10-29
 * @author Olexandr Kovalchuk
 * @group IP-32
 */

namespace _05_csharp {
  class Program {
    static void Main(string[] args) {

      Console.WriteLine("Lab 5 started");

      int threadNum;
      int size;

      {
        var arguments = new Dictionary<string, string>();
        foreach (string argument in args) {
          string[] splitted = argument.Split('=');
          if (splitted.Length == 2) {
            arguments[splitted[0]] = splitted[1];
          }
        }

        try {
          threadNum = int.Parse(arguments["threads"]);
        } catch (Exception e) { threadNum = 3; }

        try {
          size = int.Parse(arguments["size"]);
        } catch (Exception e) { size = 4; }
      }

      List<Thread> threads = new List<Thread>();
```

```
47        for (int i = 0; i < threadNum; ++i) {
48          Thread thrd;
49          switch (i % 3) {
50            case 0:
51              thrd = new Thread(Tasks.Task1);
52              break;
53            case 1:
54              thrd = new Thread(Tasks.Task2);
55              break;
56            default: case 2:
57              thrd = new Thread(Tasks.Task3);
58              break;
59          }
60          threads.Add(thrd);
61        }
62
63        foreach (Thread t in threads) {
64          t.Start(size);
65        }
66
67        foreach(Thread t in threads) {
68          t.Join();
69        }
70
71        Console.WriteLine("Lab 5 finished");
72        Console.ReadKey();
73      }
74    }
75  }
```

05-csharp/05-csharp/Tasks.cs

```
1   using System;
2   using System.Text;
3   using System.Diagnostics;
4   using System.Threading;
5
6   /**
7    * Parallel programming
8    * Lab 5
9    *
10   * Functions:
11   * F1: C = A - B * (MA * MD)
12   * F2: o = Min(MK * MM)
13   * F3: T = (MS * MZ) * (W + X)
14   *
15   * @since 2015-10-29
16   * @author Olexandr Kovalchuk
17   * @group IP-32
18   */
19
20  namespace _05_csharp {
21    class Tasks {
22      public static void Task1(Object sz) {
23        Debug.Assert(sz is int);
24        int size = (int)sz;
25
26        Console.WriteLine(
27          "task 1 started on the thread {0}",
28          Thread.CurrentThread.ManagedThreadId
29        );
30
31        Thread.Sleep(200);
32
33        int[] a = Vector.GenerateVector(size);
34        int[] b = Vector.GenerateVector(size);
35        int[,] ma = Matrix.GenerateMatrix(size);
36        int[,] md = Matrix.GenerateMatrix(size);
```

```
37
38        int[] result = Functions.Func1(a, b, ma, md);
39        if (size < 8) {
40          StringBuilder sb = new StringBuilder();
41          sb.Append("task 1: [");
42          for (int i = 0; i < result.Length; ++i) {
43            sb.Append(result[i]).Append(",");
44          }
45          sb.Append("];");
46          Console.WriteLine(sb.ToString());
47        }
48
49        Console.WriteLine("task 1 finished");
50
51      }
52
53      public static void Task2(Object sz) {
54        Debug.Assert(sz is int);
55        int size = (int)sz;
56
57        Console.WriteLine(
58          "task 2 started on the thread {0}",
59          Thread.CurrentThread.ManagedThreadId
60        );
61
62        Thread.Sleep(200);
63
64        int[,] mk = Matrix.GenerateMatrix(size);
65        int[,] mn = Matrix.GenerateMatrix(size);
66
67        int result = Functions.Func2(mk, mn);
68        if (size < 8) {
69          Console.WriteLine("task 2: {0}", result);
70        }
71
72        Console.WriteLine("task 2 finished");
73      }
74
75      public static void Task3(Object sz) {
76        Debug.Assert(sz is int);
77        int size = (int)sz;
78
79        Console.WriteLine(
80          "task 3 started on the thread {0}",
81          Thread.CurrentThread.ManagedThreadId
82        );
83
84        Thread.Sleep(200);
85
86        int[] w = Vector.GenerateVector(size);
87        int[] x = Vector.GenerateVector(size);
88        int[,] ms = Matrix.GenerateMatrix(size);
89        int[,] mz = Matrix.GenerateMatrix(size);
90
91        int[] result = Functions.Func3(ms, mz, w, x);
92        if (size < 8) {
93          StringBuilder sb = new StringBuilder();
94          sb.Append("task 3: [");
95          for (int i = 0; i < result.Length; ++i)
96          {
97            sb.Append(result[i]).Append(",");
98          }
99          sb.Append("];");
100         Console.WriteLine(sb.ToString());
101       }
102
```

```
      Console.WriteLine("task 3 finished");
    }
  }
}
```

05-csharp/05-csharp/Functions.cs

```
/**
 * Parallel programming
 * Lab 5
 *
 * Functions:
 * F1: C = A - B * (MA * MD)
 * F2: o = Min(MK * MM)
 * F3: T = (MS * MZ) * (W + X)
 *
 * @since 2015-10-29
 * @author Olexandr Kovalchuk
 * @group IP-32
 */

namespace _05_csharp {
  class Functions {
    public static int[] Func1(int[] a, int[] b, int[,] ma, int[,] md) {
      return (
        Vector.Substract(
          a,
          Vector.Multiply(
            b,
            Matrix.Multiply(ma, md)
          )
        )
      );
    }

    public static int Func2(int[,] mk, int[,] mn) {
      return (
        Matrix.Min(
          Matrix.Multiply(mk, mn)
        )
      );
    }

    public static int[] Func3(int[,] ms, int[,] mz, int[] w, int[] x) {
      return (
        Vector.Multiply(
          Matrix.Multiply(ms, mz),
          Vector.Add(w, x)
        )
      );
    }
  }
}
```

05-csharp/05-csharp/Matrix.cs

```
using System.Diagnostics;

/**
 * Parallel programming
 * Lab 5
 *
 * Functions:
 * F1: C = A - B * (MA * MD)
 * F2: o = Min(MK * MM)
 * F3: T = (MS * MZ) * (W + X)
 *
 * @since 2015-10-29
 * @author Olexandr Kovalchuk
 * @group IP-32
```

```
 */

namespace _05_csharp {
  class Matrix {

    public static int[,] GenerateMatrix(int size, int filler = 1) {
      return GenerateMatrix(size, size, filler);
    }

    public static int[,] GenerateMatrix(int rows, int columns, int filler = 1) {
      int[,] result = new int[rows,columns];
      for (int r = 0; r < rows; ++r) {
        for (int c = 0; c < columns; ++c) {
          result[r, c] = filler;
        }
      }
      return result;
    }

    public static int[,] Multiply(int[,] left, int[,] right) {
      Debug.Assert(left.GetLength(0) > 0);
      Debug.Assert(right.GetLength(0) > 0);
      Debug.Assert(left.GetLength(1) == right.GetLength(0));

      int[,] result = GenerateMatrix(left.GetLength(0), right.GetLength(1), 0);
      for (int i = 0; i < left.GetLength(0); ++i) {
        for (int j = 0; j < right.GetLength(1); ++j) {
          for (int k = 0; k < left.GetLength(1); ++k) {
            result[i,j] += left[i,k] * right[k,j];
          }
        }
      }
      return result;
    }

    public static int Min(int[,] mtrx) {
      int result = mtrx[0,0];
      for (int r = 0; r < mtrx.GetLength(0); ++r) {
        for (int c = 0; c < mtrx.GetLength(1); ++c) {
          if (mtrx[r,c] < result) {
            result = mtrx[r,c];
          }
        }
      }
      return result;
    }
  }
}
```

05-csharp/05-csharp/Vector.cs

```
using System.Diagnostics;

/**
 * Parallel programming
 * Lab 5
 *
 * Functions:
 * F1: C = A - B * (MA * MD)
 * F2: o = Min(MK * MM)
 * F3: T = (MS * MZ) * (W + X)
 *
 * @since 2015-10-29
 * @author Olexandr Kovalchuk
 * @group IP-32
 */

namespace _05_csharp {
```

```
18    class Vector {
19
20      public static int[] GenerateVector(int size, int filler = 1) {
21        int[] result = new int[size];
22        for (int i = 0; i < size; ++i) {
23          result[i] = filler;
24        }
25        return result;
26      }
27
28      public static int[] Multiply(int[,] left, int[] right) {
29        Debug.Assert(left.GetLength(0) > 0);
30        Debug.Assert(right.Length > 0);
31        Debug.Assert(right.Length == left.GetLength(1));
32
33        int[] result = new int[left.GetLength(0)];
34        for (int i = 0; i < left.GetLength(0); ++i) {
35          for (int j = 0; j < right.Length; ++j) {
36            result[i] += left[i,j] * right[j];
37          }
38        }
39        return result;
40      }
41
42      public static int[] Multiply(int[] left, int[,] right) {
43        Debug.Assert(left.Length > 0);
44        Debug.Assert(right.GetLength(0) > 0);
45        Debug.Assert(left.Length == right.GetLength(1));
46
47        int[] result = new int[left.Length];
48        for (int i = 0; i < right.GetLength(1); ++i) {
49          for (int j = 0; j < right.GetLength(0); ++j) {
50            result[i] += right[i,j] * left[j];
51          }
52        }
53        return result;
54      }
55
56      public static int[] Add(int[] left, int[] right) {
57        Debug.Assert(left.Length > 0);
58        Debug.Assert(left.Length == right.Length);
59
60        int[] result = new int[left.Length];
61        for (int i = 0; i < result.Length; ++i) {
62          result[i] = left[i] + right[i];
63        }
64        return result;
65      }
66
67      public static int[] Substract(int[] left, int[] right) {
68        Debug.Assert(left.Length > 0);
69        Debug.Assert(left.Length == right.Length);
70
71        int[] result = new int[left.Length];
72        for (int i = 0; i < result.Length; ++i) {
73          result[i] = left[i] - right[i];
74        }
75
76        return result;
77      }
78
79    }
80  }
```