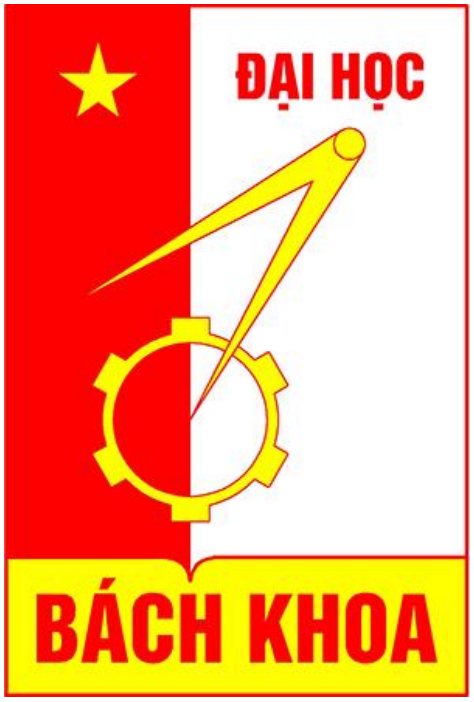
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO CUỐI KÌ

BÀI TẬP THỰC HÀNH ALCHEMI

Học phần: Tính toán song song

Giảng viên hướng dẫn: **TS. Đoàn Duy Trung**

Sinh viên thực hiên: Nguyễn Văn Triển

MSSV: 20195934

Lớp: Toán tin 02 – K64

Hà Nội, 01/2021

Mục lục

Phần 1. Bài tập chung

1. Số chính phương

Đề bài : Liệt kê các số chính phương từ 1 đến n, với n nhập từ bàn phím. Phân chia đoạn [1;n] thành các đoạn để chạy trên các executor trong Lưới

1.1 Nội dung

Kiểm tra 1 số có phải số chính phương không : Nếu căn bậc 2 của số đó là số nguyên thì đó là số chính phương

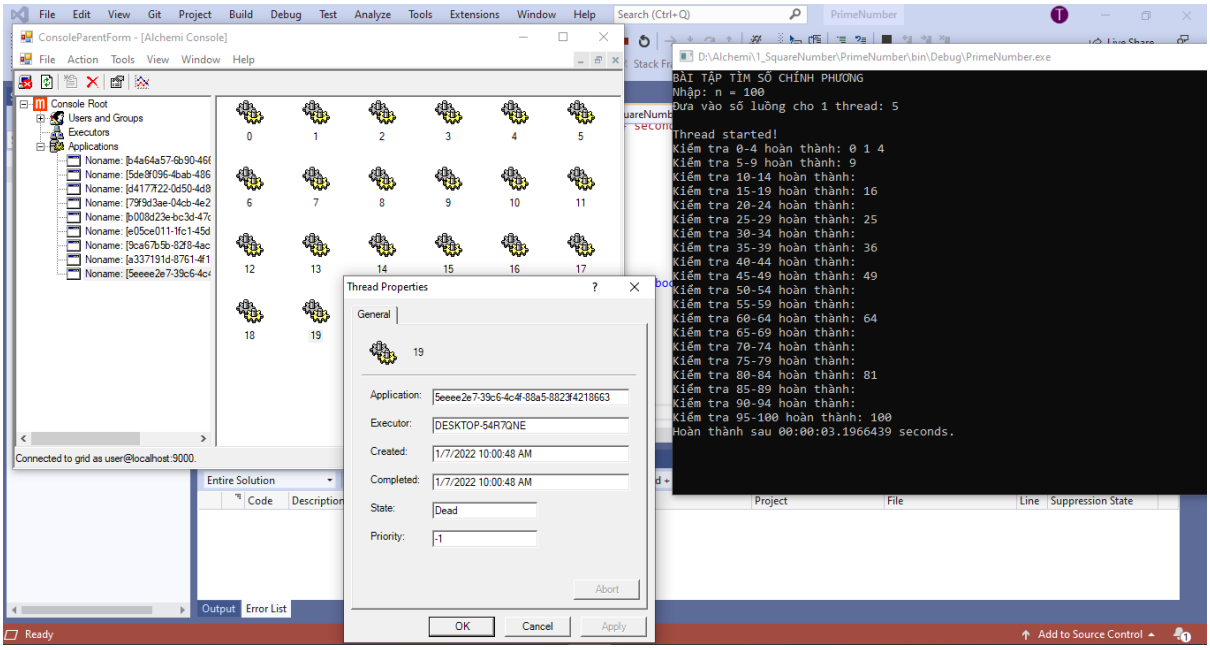
```
if (Math.sqrt(n) == (int)Math.sqrt(n))
{
        n is SquareNumber
}
```

1.2 Code

```
using System;
using Alchemi.Core.Owner;

namespace SquareNumber
{
    class SquareNumber : GApplication
    {
        public static GApplication App = new GApplication();
        private static bool[] matrix;//Mang chua so can kiem tra
        private static int NumPerThread;//So luong so trong 1 thread
        private static DateTime start;

        [STAThread]
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            Console.WriteLine("BÀI TẬP TÌM SỐ CHÍNH PHƯƠNG");
            Console.Write("Nhập: n = ");
            int n = Int32.Parse(Console.ReadLine());
            Console.Write("Đưa vào số luồng cho 1 thread: ");
            NumPerThread = Int32.Parse(Console.ReadLine());
            matrix = new bool[n+1];
            for (int i = 0; i <= n; i++)
            {
                matrix[i] = false;
            }
            Console.WriteLine();
            int NumRemain = n;
            int NumCur = 0;
            int NumberOfThread, Begin, End;
            while (NumRemain > 0)
            {
                if (NumRemain > NumPerThread)
                {
                    NumberOfThread = NumPerThread;
                }
                else
                {
                    NumberOfThread = NumRemain;
                }
                Begin = NumCur;
                NumCur += NumberOfThread;
                if (NumCur >= n) { End = n; }
```

```
                else { End = NumCur - 1; }

                App.Threads.Add(new SquareNumberCheck(Begin, End, matrix));
                NumRemain -= NumberOfThread;
            }
            App.Connection = new GConnection("localhost",9000,"user","user");
            App.Manifest.Add(new
ModuleDependency(typeof(SquareNumberCheck).Module));
            App.ThreadFinish += new GThreadFinish(App_ThreadFinish);
            App.ApplicationFinish += new
GApplicationFinish(App_ApplicationFinish);
            start = DateTime.Now;
            Console.WriteLine("Thread started!");
            App.Start();
            Console.ReadLine();

        }

        private static void App_ThreadFinish(GThread thread)
        {
            SquareNumberCheck snc = (SquareNumberCheck)thread;
            Console.Write("Kiểm tra {0}-{1} hoàn thành: ", snc.Begin, snc.End);
            for (int i = snc.Begin; i <= snc.End; i++)
                if (snc.Snumbers[i] == true) Console.Write(i + " ");
            Console.WriteLine();
        }

        private static void App_ApplicationFinish()
        {
            Console.WriteLine("Hoàn thành sau {0} seconds.",DateTime.Now-
start);
        }
    }
    [Serializable]
    class SquareNumberCheck : GThread
    {
        public int Begin, End;
        public bool[] Snumbers;
        public SquareNumberCheck(int begin, int end, bool[] numbers)
        {
            this.Begin = begin;
            this.End = end;
            this.Snumbers = numbers;
        }
        public override void Start()
        {
            for (int i = Begin; i <= End; i++)
            {
                if (Math.Sqrt(i) == (int)(Math.Sqrt(i)))
                {
                    Snumbers[i] = true;
                }
            }
        }
    }
}
```

1.3 Kết quả



2. Tính gần đúng tích phân

Đề bài : Cho n nhập từ bàn phím. Tính gần đúng tích phân sau :

$$I = \int_0^n f(x)dx$$

Ở đó hàm f(x) tùy ý

Phân chia [0;n] vào các Executor để chạy trong Lưới.

2.1 Nội dung

Sử dụng công thức hình thang tính gần đúng tích phân

$$I = \frac{h}{2}((y_0 + y_n) + 2\sum_{i=1}^{n-1} y_i)$$

Với:

- n là số bước nhập từ bàn phím
- $h = \frac{b-a}{n}$
- Cho $i = \overline{0,n}$ thì $x_i = a + ih$ và $y_i = f(x_i)$

2.2 Code

```
using System;
using Alchemi.Core.Owner;

namespace Integral
{
    class Intergral : GApplication
    {
        public static GApplication App = new GApplication();
        private static int NumPerThread;//So luong so trong 1 thread
        private static DateTime start;
        private static double h;
        private static double I; //file ghi
        static double f(double x)
```

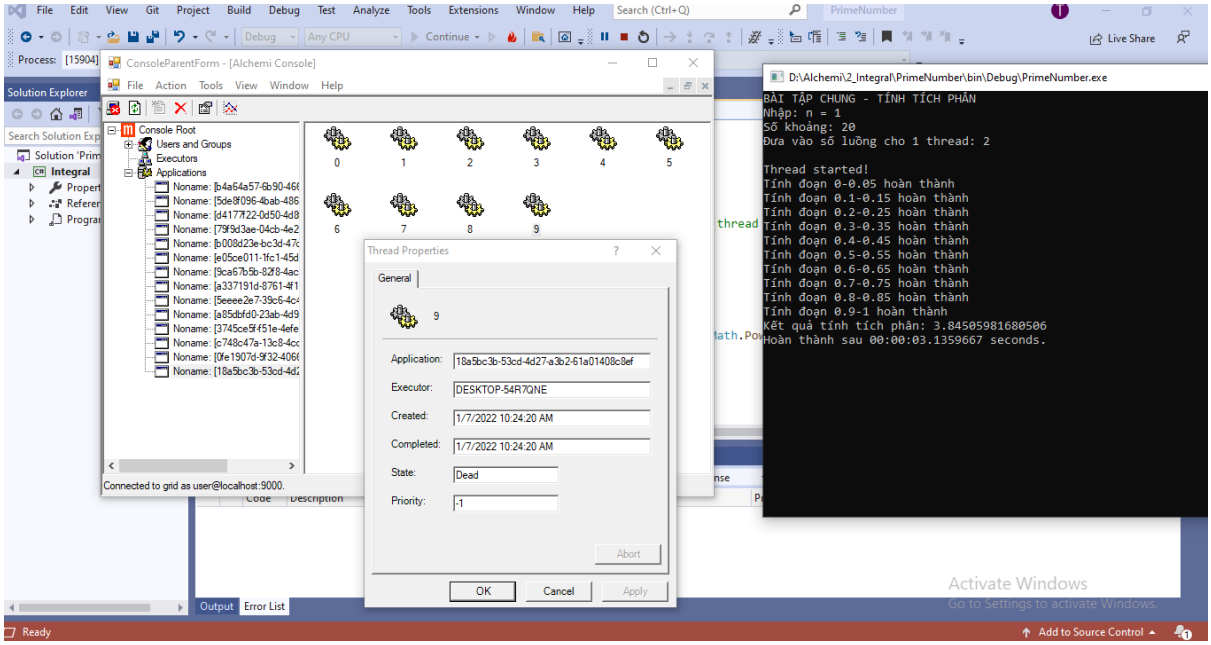
```
        {
            return
(double)(Math.Sin(Math.PI+Math.Pow(x,2))+(Math.Pow(x,2)+1)/2+Math.PI);
        }
        [STAThread]
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            Console.WriteLine("BÀI TẬP CHUNG - TÍNH TÍCH PHÂN");

            Console.Write("Nhập: n = ");
            int n = Convert.ToInt32(Console.ReadLine());
            double a = 0; // cận dưới
            double b = n; // cận trên
            I = (f(a) + f(b)) / 2;
            Console.Write("Số khoảng: ");
            int kc = Int32.Parse(Console.ReadLine());
            h = (b - a) / kc;
            Console.Write("Đưa vào số luồng cho 1 thread: ");
            NumPerThread = Int32.Parse(Console.ReadLine());

            Console.WriteLine();
            int NumRemain = kc;
            int NumCur = 0;
            int NumberOfThread, Begin, End;
            while (NumRemain > 0)
            {
                if (NumRemain > NumPerThread)
                {
                    NumberOfThread = NumPerThread;
                }
                else
                {
                    NumberOfThread = NumRemain;
                }
                Begin = NumCur;
                NumCur += NumberOfThread;
                if (NumCur >= kc) { End = kc; }
                else { End = NumCur - 1; }

                App.Threads.Add(new CalcIntegral(Begin, End, h));
                NumRemain -= NumberOfThread;
            }
            App.Connection = new GConnection("localhost", 9000, "user",
"user");
            App.Manifest.Add(new
ModuleDependency(typeof(CalcIntegral).Module));
            App.ThreadFinish += new GThreadFinish(App_ThreadFinish);
            App.ApplicationFinish += new
GApplicationFinish(App_ApplicationFinish);
            start = DateTime.Now;
            Console.WriteLine("Thread started!");
            App.Start();
            Console.ReadLine();
        }
        private static void App_ThreadFinish(GThread thread)
        {
            CalcIntegral ci = (CalcIntegral)thread;
            Console.WriteLine("Tính đoạn {0}-{1} hoàn thành", ci.x[ci.Begin],
ci.x[ci.End]);
            for (int i = ci.Begin; i <= ci.End; i++)
            {
```

```
                I += ci.Nums[i];
            }
        }
        private static void App_ApplicationFinish()
        {

            Console.WriteLine("Kết quả tính tích phân: {0}", I*h);
            Console.WriteLine("Hoàn thành sau {0} seconds.", DateTime.Now -
start);
        }
    }
    [Serializable]
    class CalcIntegral : GThread
    {
        public int Begin, End;
        public double[] Nums = new double[100];
        public double[] x = new double[100];
        private double h;
        public CalcIntegral(int begin, int end, double kc)
        {
            this.Begin = begin;
            this.End = end;
            this.h = kc;
        }
        private double f(double x)
        {
            return (double)(Math.Sin(Math.PI + Math.Pow(x, 2)) + (Math.Pow(x,
2) + 1) / 2 + Math.PI);
        }
        public override void Start()
        {
            for (int i = Begin; i <= End; i++)
            {
                x[i] = i * h;
                Nums[i] = f(x[i]);
            }
        }
    }
}
```

2.3 Kết quả

Kiểm tra kết quả hàm số :

$$f(x) = \sin(x^2 + \pi) + \frac{x^2 + 1}{2} + \pi$$



Phần 2. Bài tập riêng

Bài 1 : đọc file văn bản, xử lí mã ASCII

Đề bài : Đọc 1 văn bản dưới dạng file input.txt, thực hiện hàm số f(x) = 2x + 1, ở đó x là ký tự mã ASCII của từng ký tự trong file input.txt và in ký tự mã ASCII tương ứng f(x) ra file output.txt. Yêu cầu, phân chia file input.txt thành k luồng (k nhập từ bàn phím), theo kích thước của tệp tin input.txt để thực hiện trên các nút trong lưới tính toán.

1.1 Nội dung

Đọc file input.txt và lưu vào mảng kí tự *rdoc*

Chuyển mảng kí tự *rdoc* thành mảng số (mã ascii) tương ứng *asciiBytes*

Thực hiện tính toán mảng *asciiBytes*

Chuyển mảng số *asciiBytes* về mảng kí tự ascii tương ứng

Lưu mảng kí tự sau khi chuyển đổi vào file output.txt

(Bảng mã có 256 kí tự nên nếu vượt quá sẽ bị out)

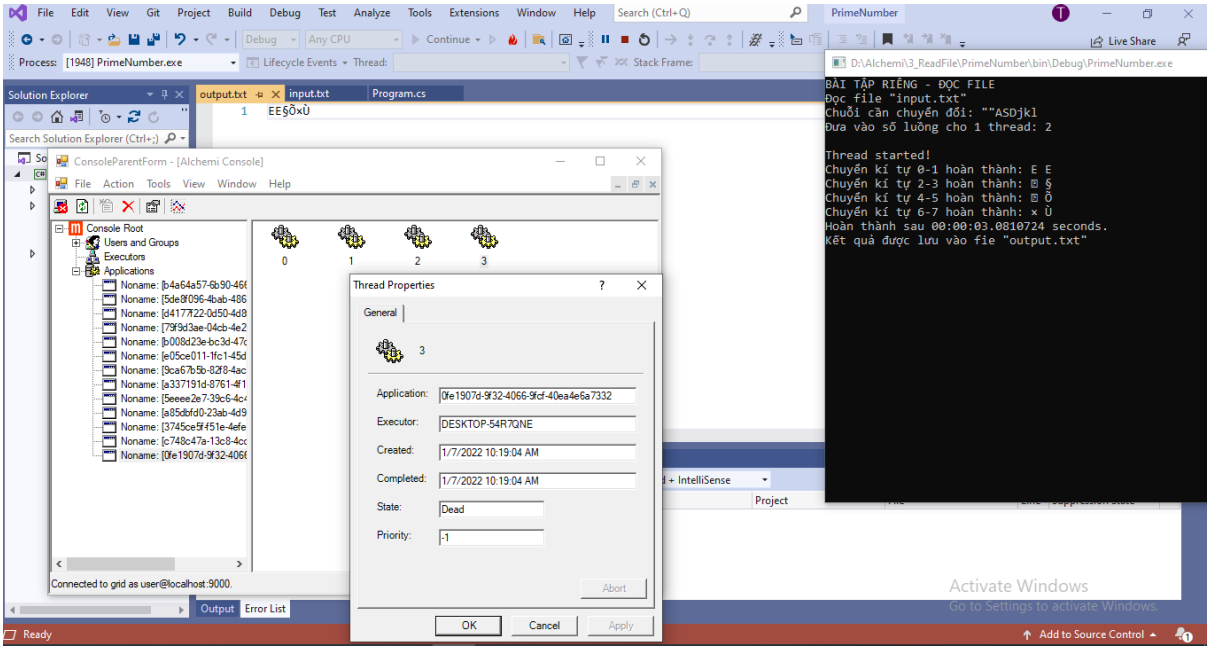
1.2 Code

```
using System;
using System.IO; //read file
using System.Text;
using Alchemi.Core.Owner;

namespace SquareNumber
{
    class SquareNumber : GApplication
    {
        public static GApplication App = new GApplication();
        private static int NumPerThread;//So luong so trong 1 thread
        private static DateTime start;
        private static string wdoc; //file ghi
```

```
        [STAThread]
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            Console.WriteLine("BÀI TẬP RIÊNG - ĐỌC FILE");
            Console.WriteLine("Đọc file \"input.txt\"");
            string rdoc = File.ReadAllText("../../input.txt"); // file đọc
            Console.WriteLine("Chuỗi cần chuyển đổi: " + rdoc);

            Console.Write("Đưa vào số luồng cho 1 thread: ");
            NumPerThread = Int32.Parse(Console.ReadLine());

            Console.WriteLine();
            byte[] asciiBytes = Encoding.ASCII.GetBytes(rdoc); // convert ascii
-> num
            int n = asciiBytes.Length;
            int NumRemain = n;
            int NumCur = 0;
            int NumberOfThread, Begin, End;
            while (NumRemain > 0)
            {
                if (NumRemain > NumPerThread)
                {
                    NumberOfThread = NumPerThread;
                }
                else
                {
                    NumberOfThread = NumRemain;
                }
                Begin = NumCur;
                NumCur += NumberOfThread;
                if (NumCur >= n) { End = n - 1; }
                else { End = NumCur - 1; }

                App.Threads.Add(new FileChange(Begin, End, asciiBytes));
                NumRemain -= NumberOfThread;
            }
            App.Connection = new GConnection("localhost", 9000, "user",
"user");
            App.Manifest.Add(new ModuleDependency(typeof(FileChange).Module));
            App.ThreadFinish += new GThreadFinish(App_ThreadFinish);
            App.ApplicationFinish += new
GApplicationFinish(App_ApplicationFinish);
            start = DateTime.Now;
            Console.WriteLine("Thread started!");
            App.Start();
            Console.ReadLine();

        }

        private static void App_ThreadFinish(GThread thread)
        {
            FileChange fc = (FileChange)thread;
            Console.Write("Chuyển kí tự {0}-{1} hoàn thành: ", fc.Begin,
fc.End);
            wdoc = "";
            for (int i = fc.Begin; i <= fc.End; i++)
            {
                Console.Write(fc.Texts[i] + " ");
                wdoc += fc.Texts[i];
            }
```

```
            File.AppendAllText("../../output.txt", wdoc);
            Console.WriteLine();
        }
        private static void App_ApplicationFinish()
        {
            Console.WriteLine("Hoàn thành sau {0} seconds.", DateTime.Now -
start);
            Console.WriteLine("Kết quả được lưu vào fie \"output.txt\"");
        }
    }
    [Serializable]
    class FileChange : GThread
    {
        public int Begin, End;
        public byte[] Nums = new byte[100];
        public char[] Texts = new char[100];
        public FileChange(int begin, int end, byte[] nums)
        {
            this.Begin = begin;
            this.End = end;
            this.Nums = nums;
        }
        private byte f(byte x)
        {
            return (byte)(2 * x + 1);
        }
        public override void Start()
        {
            for (int i = Begin; i <= End; i++)
            {
                Nums[i] = f(Nums[i]);
                if (Nums[i] >= 255) { Nums[i] -= 255; }
                Texts[i] = Convert.ToChar(Nums[i]); // convert num -> ascii
            }
        }
    }
}
```

1.3 Kết quả



Phần 3. Bài tập thêm

Bài 5 : Tính số Fibonacci

Đề bài : Liệt kê dãy Fibonacci đến n số cho trước (n – nhập từ bàn phím)

1.1 Nội dung

Sử dụng công thức tính số Fibonacci thứ n :

$$f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}^{n} + \frac{1-\sqrt{5}}{2}^{n}\right)$$

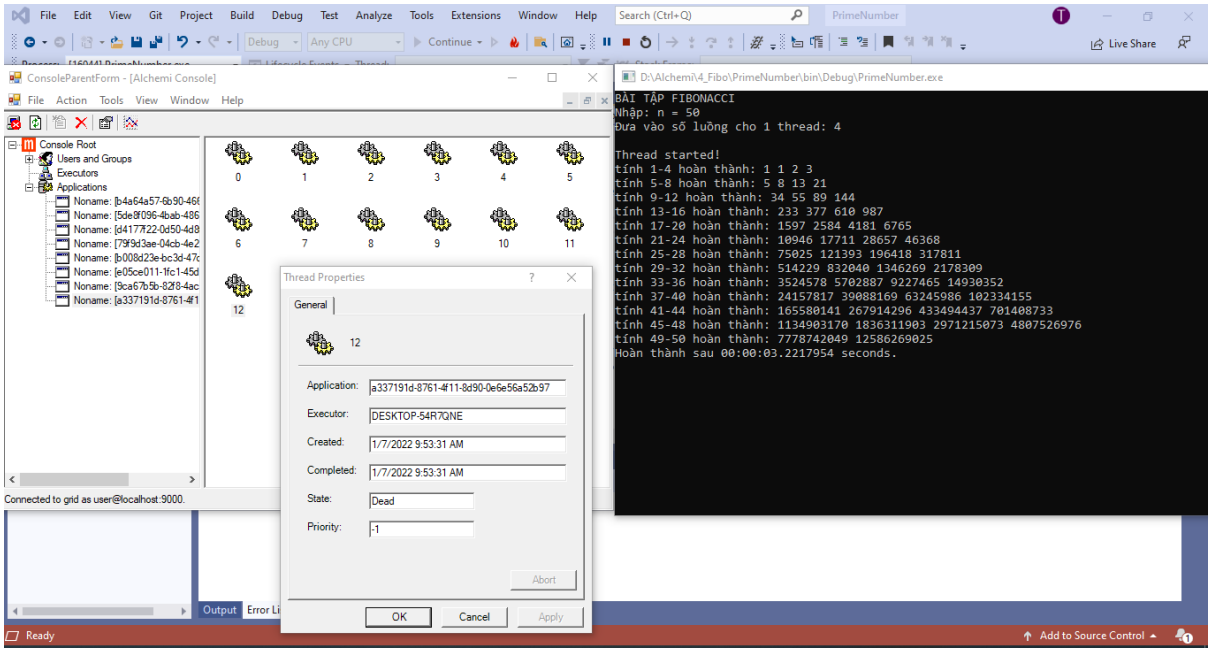
(Làm tròn số)

1.2 Code

```
using System;
using Alchemi.Core.Owner;

namespace FiboNumber
{
    class FiboNumber : GApplication
    {
        public static GApplication App = new GApplication();
        private static int NumPerThread;//So luong so trong 1 thread
        private static DateTime start;

        [STAThread]
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            Console.WriteLine("BÀI TẬP FIBONACCI");
            Console.Write("Nhập: n = ");
            int n = Int32.Parse(Console.ReadLine());
            Console.Write("Đưa vào số luồng cho 1 thread: ");
            NumPerThread = Int32.Parse(Console.ReadLine());

            Console.WriteLine();
            int NumRemain = n;
            int NumCur = 1;
            int NumberOfThread, Begin, End;
            while (NumRemain > 0)
            {
                if (NumRemain > NumPerThread)
                {
                    NumberOfThread = NumPerThread;
                }
                else
                {
                    NumberOfThread = NumRemain;
                }
                Begin = NumCur;
                NumCur += NumberOfThread;
                if (NumCur >= n) { End = n; }
                else { End = NumCur - 1; }

                App.Threads.Add(new FiboNumberCheck(Begin, End));
                NumRemain -= NumberOfThread;
            }
            App.Connection = new GConnection("localhost", 9000, "user",
"user");
```

```
            App.Manifest.Add(new
ModuleDependency(typeof(FiboNumberCheck).Module));
            App.ThreadFinish += new GThreadFinish(App_ThreadFinish);
            App.ApplicationFinish += new
GApplicationFinish(App_ApplicationFinish);
            start = DateTime.Now;
            Console.WriteLine("Thread started!");
            App.Start();
            Console.ReadLine();

        }

        private static void App_ThreadFinish(GThread thread)
        {
            FiboNumberCheck fnc = (FiboNumberCheck)thread;
            Console.Write("tính {0}-{1} hoàn thành: ", fnc.Begin, fnc.End);
            for (int i = fnc.Begin; i <= fnc.End; i++)
            {
                Console.Write( fnc.Fibos[i] + " ");
            }
            Console.WriteLine();
        }
        private static void App_ApplicationFinish()
        {
            Console.WriteLine("Hoàn thành sau {0} seconds.", DateTime.Now -
start);
        }
    }
    [Serializable]
    class FiboNumberCheck : GThread
    {
        public int Begin, End;
        public long[] Fibos = new long[100];
        public FiboNumberCheck(int begin, int end)
        {
            this.Begin = begin;
            this.End = end;
        }
        private long Fibo(int m) // Direct Calculation, correct for abs(m) <=
92
        {
            double s = 1.0 / Math.Sqrt(5.0);
            double g1 = (1.0 + Math.Sqrt(5.0)) / 2.0;
            double g2 = (1.0 - Math.Sqrt(5.0) / 2);
            return (long)Math.Round(s * (Math.Pow(g1, m) - Math.Pow(g2, m)));
        }
        public override void Start()
        {
            for (int i = Begin; i <= End; i++)
            {
                Fibos[i] = Fibo(i);
            }
        }
    }
}
```

1.3 Kết quả


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
PrimeNumber
Live Share
ConsoleParentForm - [Alchemi Console]
File Action Tools View Window Help
Console Root
Users and Groups
Executors
Applications
Connected to grid as user@localhost:9000.
D:\Alchemi\4_Fibo\PrimeNumber\bin\Debug\PrimeNumber.exe
BÀI TẬP FIBONACCI
Nhập: n = 50
Đưa vào số luồng cho 1 thread: 4
Thread started!
tính 1-4 hoàn thành: 1 1 2 3
tính 5-8 hoàn thành: 5 8 13 21
tính 9-12 hoàn thành: 34 55 89 144
tính 13-16 hoàn thành: 233 377 610 987
tính 17-20 hoàn thành: 1597 2584 4181 6765
tính 21-24 hoàn thành: 10946 17711 28657 46368
tính 25-28 hoàn thành: 75025 121393 196418 317811
tính 29-32 hoàn thành: 514229 832040 1346269 2178309
tính 33-36 hoàn thành: 3524578 5702887 9227465 14930352
tính 37-40 hoàn thành: 24157817 39088169 63245986 102334155
tính 41-44 hoàn thành: 165580141 267914296 433494437 701408733
tính 45-48 hoàn thành: 1134903170 1836311903 2971215073 4807526976
tính 49-50 hoàn thành: 7778742049 12586269025
Hoàn thành sau 00:00:03.2217954 seconds.
Thread Properties
General
12
Application: a337191d-8761-4f11-8d90-0e6e56a52b97
Executor: DESKTOP-54R7QNE
Created: 1/7/2022 9:53:31 AM
Completed: 1/7/2022 9:53:31 AM
State: Dead
Priority: -1
Abort
OK
Cancel
Apply
Output
Ready
Add to Source Control