Họ tên SV: **LÝ HẢI NAM.**

MSSV: **20152557.**

Học phần: *Cấu trúc dữ liệu và giải thuật.*

Mã HP: *MI3060.*

Tên tiếng Anh: *Data Structures and Algorithms.*

Thời lượng: *60 tiết.*

Khối lượng: *3(3-1-0-6).*

## Week 02 (Monday, September 11th, 2017)

1. Thực hiện cài đặt môi trường lập trình C#.
   Đã cài đặt Microsoft Visual Studio 2010 Ultimate.

2. Viết một chương trình console đơn giản
   Chương trình Hello World!!

3. Viết một chương trình thực hiện việc vào/ra
   Nhập vào một số nguyên. In ra chuỗi nhị phân tương ứng với số đã nhập

4. Các kiểu dữ liệu cơ bản:
   4.1. Kiểu int: Viết chương trình tính n!, chuyển sang sử dụng kiểu dữ liệu BigInteger.

   4.2. Kiểu double: Viết chương trình tính số có kiểu double.

   4.3. Kiểu char: khởi tạo các ký tự và gán giá trị dưới dạng các bảng mã, sau đó in các ký tự ra màn hình.

   4.4. String: Viết chương trình:
      4.4.1. Khởi tạo hai chuỗi
      4.4.2. Nối hai chuỗi và In chuỗi mới ra màn hình
      4.4.3. So sánh chuỗi mới vừa thu được với một chuỗi để đưa ra T/F.
      4.4.4. In ra độ dài chuỗi
      4.4.5. Chuyển các ký tự của chuỗi sang ký tự in hoa.

4.5. Array: Viết chương trình nhập hai ma trận. Đưa hai ma trận ra màn hình và tính tổng hai ma trận, đưa kết quả ra màn hình

4.6. List: Viết chương trình:

4.6.1. Khởi tạo danh sách rỗng

4.6.2. Chèn thêm các phần tử vào danh sách

4.6.3. In ra số phần tử của danh sách vừa nhập.

4.6.4. Đảo ngược danh sách và in ra màn hình.

4.6.5. Thêm phần tử vào cuối danh sách và in ra danh sách mới.

4.6.6. Thêm / Xóa một phần tử tại một vị trí cụ thể trong danh sách.

Kiểu List trong C# là một kiểu dữ liệu rất linh động, chúng ta có thể thêm vào đó một phần tử với kiểu dữ liệu bất kỳ, điều đó là ko thể với các ngôn ngữ như C, Pascal.

5. Các cấu trúc điều khiển If, For, While, Do...While, Switch-Case, Goto, được đề cập trong các bài toán trên.

5.1. If: Giải phương trình bậc 2

5.2. For: Đã đề cập trong vấn đề nhập, in các ma trận và xử lý tổng hai ma trận

5.3. While

5.4. Do..While

5.5. Switch – Case: Nhập vào một số, nếu sai yêu cầu nhập lại, in ra màn hình thứ tương ứng trong tuần với số đã nhập vào.

5.6. Goto: Kết hợp với Switch – Case.

## Week 03 (Monday, September 18th, 2017)

1. Giải phương trình bậc 2 trên trường số phức.
   1.1. Version 1: Sử dụng lớp Số phức được xây dựng sẵn trong C#
      1.1.1. Khai báo thư viện System.Numerics và sử dụng lớp Complex cùng các phép toán đã được xây dựng sẵn.
      1.1.2. Add Referrences → .NET → System.Numerics để sử dụng.
      1.1.3. Giải phương trình bậc 2 trên trường số phức.

```
        static void Main()
        {
            Console.Write("Enter the real part of the complex: ");
            double a1 = double.Parse(Console.ReadLine());
            Console.Write("Enter the imaginary part of the complex: ");
            double a2 = double.Parse(Console.ReadLine());
            Complex a = new Complex(a1, a2);
            Console.WriteLine("Constant a: {0} + {1}i", a1, a2);
            Console.Write("Enter the real part of the complex: ");
            double b1 = double.Parse(Console.ReadLine());
            Console.Write("Enter the imaginary part of the complex: ");
            double b2 = double.Parse(Console.ReadLine());
            Complex b = new Complex(b1, b2);
            Console.WriteLine("Constant a: {0} + {1}i", b1, b2);
            Console.Write("Enter the real part of the complex: ");
            double c1 = double.Parse(Console.ReadLine());
            Console.Write("Enter the imaginary part of the complex: ");
            double c2 = double.Parse(Console.ReadLine());
            Complex c = new Complex(c1, c2);
            Console.WriteLine("Constant a: {0} + {1}i", c1, c2);
            Complex Delta = b * b - 4 * a * c;
            if (Delta == 0)
            {
                Console.WriteLine("\nThe quadratic has a repeated root
is: z = {0} + {1}i", (-b/a).Real, (-b/a).Imaginary);
            }
            // Taking the Square root of the Delta formula
            Complex Omega = Complex.Sqrt(Delta);

            // Solving both of the formula
            Complex z1 = (-b + Omega) / 2 / a;
            Complex z2 = (-b - Omega) / 2 / a;

            Console.WriteLine("\nThe quadratic equation need to solve:
({0} + {1}i)z*z + ({2} + {3}i)z + ({4} + {5}i) = 0\n", a1, a2, b1, b2,
c1, c2);
            Console.WriteLine("The quadratic formulas of this equation
are:\nz1 = {0} + {1}i", z1.Real, z1.Imaginary);
            Console.WriteLine("z2 = {0} + {1}i", z2.Real,
z2.Imaginary);
            Console.ReadLine();
        }
```

### 1.2. Version 2: Xây dựng lớp Số phức, sử dụng toán tử nạp chồng (Overloading)

```
public class Complex
    {
        public double re, im;
        public Complex(double re, double im)
        {
            this.re = re;
            this.im = im;
        }
        public static Complex Enter()
        {
            Console.Write("Enter the real part of the complex: ");
            double a1 = double.Parse(Console.ReadLine());
            Console.Write("Enter the imaginary part of the complex: ");
            double a2 = double.Parse(Console.ReadLine());
            return new Complex(a1, a2);
        }
        // Declare which operator to overload (+)
        public static Complex operator +(Complex c1, Complex c2)
        {
            return new Complex(c1.re + c2.re, c1.im + c2.im);
        }
        // Declare which operator to overload (-)
        public static Complex operator -(Complex c1, Complex c2)
        {
            return new Complex(c1.re - c2.re, c1.im - c2.im);
        }
        // Declare which operator to overload (*)
        public static Complex operator *(Complex c1, Complex c2)
        {
            return new Complex(c1.re * c2.re - c1.im * c2.im, c1.re *
c2.im + c1.im * c2.re);
        }
        // Declare which operator to overload (/)
        public static Complex operator /(Complex c1, Complex c2)
        {
            return new Complex((c1.re * c2.re + c1.im * c2.im) / (c2.re
* c2.re + c2.im * c2.im), (c1.im * c2.re - c1.re * c2.im) / (c2.re *
c2.re + c2.im * c2.im));
        }
        // Declare which method to take the square root of a complex
number
        public static Complex Sqrt(Complex c1)
        {
            double x, y;
            x = Math.Sqrt((c1.re + Math.Sqrt(c1.re * c1.re + c1.im *
c1.im)) / 2);
            y = Math.Sqrt((-c1.re + Math.Sqrt(c1.re * c1.re + c1.im *
c1.im)) / 2);
            return new Complex(x, y);
```

```
        }
        // Override the ToString method to display an complex number in
the suitable format:
        public override string ToString()
        {
            return (String.Format("{0} + {1}i", re, im));
        }
    }
```

2. Tính xấp xỉ e mũ x bằng công thức:

$$e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots, \qquad x \in R$$

```
static void Main()
        {
            int x, i = 1;
            double epsi, t1 = 0.0, t2 = 1.0, p = 1.0;

            Console.Write("Nhap x: ");
            x = int.Parse(Console.ReadLine());
            do
            {
                Console.Write("Nhap epsilon: ");
                epsi = double.Parse(Console.ReadLine());
            } while (epsi <= 0);

            while (Math.Abs(t2 - t1) >= epsi)
            {
                t1 = t2;
                p = p * ((double)x / i);
                t2 += p;
                i++;
            }
            Console.WriteLine("\nGia tri xap xi cua e mu {0} = {1}.", x,
t2);
            Console.ReadLine();
        }
```

## Week 04 (Monday, September 25th, 2017)

1. Đếm số từ trong xâu ký tự
   Sử dụng hàm Split có sẵn trong C#

```
        static void Main()
        {
            string[] s;
            char[] c = { '\t', ' ', '`', '~', '!', '@', '#',
                         '$', '%', '^', '&', '*', '(', ')',
                         '-', '_', '=', '+', '[', ']', '{',
                         '}', ';', '\', '/', '\\', ':', '\"',
                         '|', ',', '.', '/', '<', '>', '?' };
            Console.Write("Enter a string: ");
            s = Console.ReadLine().Split(c,
StringSplitOptions.RemoveEmptyEntries);

            Console.WriteLine("The string you have entered has {0}
word(s)", s.Length);
        }
```

2. Tính xấp xỉ số PI

```
        static void Main()
        {
            double epsilon = 0, PI, PI_1 = 0.0, PI_2 = 1.0;
            int i = 1, sign = -1;
            Console.Write("Nhap sai so epsilon: ");
            epsilon = double.Parse(Console.ReadLine());

            while (Math.Abs(PI_1 - PI_2) > epsilon)
            {
                PI_1 = PI_2;
                PI_2 += 1 / (double)(sign * (2 * i + 1));
                sign = -sign;
                i++;
            }
            PI = 4 * PI_2;
            Console.WriteLine("\nGia tri xap xi cua PI la: {0}", PI);
            Console.ReadLine();
        }
```

3. Tính trung bình cộng của một dãy các số nguyên dương (sử dụng các cấu trúc trong lớp đối tượng Collections)

```
        static void Main(string[] args)
        {
            List<int> list = new List<int>(new int[]{-1, 2, 5, 6, -2, -4,
6, 3, 8, 7, 6, 9, -2});
            int tong = 0;
            int count = 0;
```

```
            Console.Write("Give a sequence: ");
            foreach (int i in list)
            {
                Console.Write("{0}, ", i);
            }
            for (int i = 0; i < list.Count; i++)
            {
                if (list[i] > 0 && list[i] % 2 == 0)
                {
                    tong += list[i];
                    count++;
                }
            }
            double tb = (1.0 * tong) / count;
            Console.WriteLine("\nTrung binh cong cac so chan trong day so
tren la: {0}", tb);
            Console.Read();
        }
```

## Week 05 (Monday, October 2nd, 2017)

1. Tính căn bậc n của một số dương m.
   Ideaaa: Đưa về bài toán giải pt $f(x) = 0$ với hàm $f(x) = x^n - m,\quad n,\ m$ nhập từ bàn phím. Sử dụng "***phương pháp chia đôi***" để giải bài toán trên với một sai số epsilon.
   **Chương trình:**

```
using System;
namespace Test2
{
    class Program
    {
        public static double luythua(double x, int a, int b)
        {
            return Math.Pow(x, a) - b;
        }

        static void Main()
        {
            double c, z, epsilon, a0, b0;
            Console.WriteLine("Chuong trinh tinh can bac n cua m (m >
0)");
            Console.Write("Nhap n: ");
            int n = int.Parse(Console.ReadLine());
            Console.Write("Nhap m: ");
            int m = int.Parse(Console.ReadLine());
            Console.Write("Nhap epsilon: ");
            epsilon = double.Parse(Console.ReadLine());

            a0 = 0; b0 = m;
            c = (1.0 * m) / 2;
            z = luythua(c, n, m);
            while (Math.Abs(a0 - b0) >= epsilon)
            {
                if (z * luythua(a0, n, m) < 0)
                {
                    b0 = c;
                    c = (a0 + b0) / 2;
                    z = luythua(c, n, m);
                }
                else
                {
                    a0 = c;
                    c = (a0 + b0) / 2;
                    z = luythua(c, n, m);
                }
            }
            Console.WriteLine("\nCan bac {0} cua {1} la: {2}", n, m, c);
        }
    }
}
```

**Nhắc lại về cơ sở của phương pháp chia đôi:**

Thu hẹp dần khoảng phân li nghiệm bằng cách chia đôi.

Cho phương trình f(x) = 0, f(x) liên tục và trái dấu tại 2 đầu [a,b]. Trên đoạn [a,b] phương trình có ít nhất 1 nghiệm μ.

Để tìm nghiệm gần đúng c, ta thực hiện một số hữu hạn lần quá trình lặp các bước sau đây:

- Bc 1: c = (a + b) / 2.
- Bc 2: If f(c) = 0 → c là nghiệm then go to Bc 4

  Else go to Bc 3.
- Bc 3: If f(a).f(c) < 0 then b = c and go to loop Bc 1

  Else If f(a).f(c) > 0 then a = c and go to loop Bc 1.
- Bc 4: If |b - a| < epsilon (sai số cho trước) then Dừng thuật toán → Output.

2. Đếm số từ trong xâu (Tự viết)

```
        static void Main()
        {
            string s;
            int i = 0, count = 1;
            Console.Write("Enter a string: ");
            s = Console.ReadLine();

            while (i <= s.Length - 1)
            {
                if (s[i] == ' ' || s[i] == '\n' || s[i] == '\t')
                {
                    count++;
                }
                i++;
            }
            Console.WriteLine("The string you have entered has {0}
word(s)", count);
        }
```

## Week 06 (Monday, October 9th, 2017)

1. Khai báo kiểu dữ liệu Stack, Queue trong C#
    - Cấu trúc dữ liệu Stack và Queue là cấu trúc dữ liệu được xây dựng sẵn trong C#.
    - Stack và Queue thuộc lớp đối tượng Collections trong C#, là những cấu trúc dữ liệu động, sử dụng linh hoạt.
    - Phương thức của Stack PUSH và POP, chỉ có thể truy cập được lần lượt các phần tử PUSH vào Stack sau cùng, theo thứ tự LIFO (Last In, First Out).
    - Phương thức của Queue là ENQUEUE, chỉ truy xuất được lần lượt phần tử ở cuối Queue, theo thứ tự FIFO (First In, First Out).

    ***Tips:*** *Để sử dụng cấu trúc Stack, Queue, cần phải khai báo thư viện System.Collections.Generics.*

2. Sử dụng cấu trúc dữ liệu Stack để chuyển một số nguyên dương từ hệ cơ số 10 sang hệ nhị phân.
   **Chương trình:**

```
using System;
using System.Collections.Generic;
namespace Bin
{
    class Program
    {
        static void Main()
        {
            Stack<int> stack = new Stack<int>();
            Console.WriteLine("The following program converts a decimal
number to the binary string");
            Console.Write("\nEnter n: ");
            int n = int.Parse(Console.ReadLine());
            int m = n;
            do
            {
                stack.Push(m % 2);
                m = m / 2;
            } while (m > 0);
            Console.Write("\nThe Binary string of {0} is: ", n);
            foreach (int i in stack)
            {
                Console.Write(i);
            }
        }
    }
}
```

## Week 08 (Monday, October 30th, 2017)

1. Khai báo và sử dụng kiểu dữ liệu LinkedList (Danh sách liên kết trong C#). Linked List là kiểu dữ liệu thuộc lớp ICollection trong C# nên cũng như các kiểu dữ liệu khác cùng lớp ICollection, khi sử dụng ta cần khai báo thư viện như sau:

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

2. Chương trình sử dụng kiểu dữ liệu LinkedList

```
using System;
using System.Collections.Generic;
namespace LinkedList
{
    class Program
    {
        static void Main()
        {
            //
            // Create a new linked list.
            //
            LinkedList<string> linked = new LinkedList<string>();
            //
            // First add three elements to the linked list.
            //
            linked.AddLast("one");
            linked.AddLast("two");
            linked.AddLast("three");
            linked.AddFirst("first");
            int count = 0, i = 0;
            for (i = 0; i < linked.Count; i++)
            {
                count++;
            }
            // Insert a node before the second node (after the
first node)
            LinkedListNode<string> node = linked.Find("one");
            linked.AddAfter(node, "inserted");
            //
            // Loop through the linked list.
            //
            foreach (var value in linked)
            {
                Console.WriteLine(value);
            }
            Console.WriteLine(count);
```

```
            for (i = count; i < linked.Count; i++)
            {
                count++;
            }
            Console.WriteLine(count);
        }
    }
}
```