# SWT – Lab02: Tower Of Hanoi

## Specification

- Input: positive integer number

- Output: non-negative integer number

- When user enters invalid integer number format, the program displays a message “Must be an integer” and returns -1 for steps.

- When user enters an integer number whose value is less than or equal to 0, the program displays a message “Disk number must be more than 0!” and return -1 for steps.

- When user enters a positive integer number, the program displays detailed steps for solving the problem.

## Rewrite C code to C# code

public int Steps { get; set; } = 0;

        public List<string> StepList { get; set; } = new List<string>();

        static void Main(string[] args)

        {

            int sodia;

            Write("Cho biet so dia can chuyen: ");

            string sNumber = ReadLine();

            int.TryParse(sNumber, out sodia);

            WriteLine("\nCac buoc chuyen nhu sau:\n\n");

            TowerOfHanoi towerOfHanoi = new TowerOfHanoi();

            towerOfHanoi.thaphanoi(sodia, 'a', 'c', 'b');

            ReadLine();

        }

        public void chuyen(int n, char a, char c)

        {

            Steps++;

            StepList.Add("H");

            WriteLine($"Chuyen dia thu {n} tu coc {a} sang coc {c} \n");

            StepList.Add("I");

            return;

        }

        public void thaphanoi(int n, char a, char c, char b)

        {

            if (n <= 0)

            {

                WriteLine("So dia phai lon hon 0!");

                string sSodia = "";

                do

                {

                    WriteLine("Nhap lai so dia: ");

                    sSodia = ReadLine();

                    if (!int.TryParse(sSodia, out n))

                    {

                        WriteLine("So dia phai la mot so nguyen!");

                    }

                    else if (n <= 0)

                    {

                        WriteLine("So dia phai lon hon 0!");

                    }

                    else

                    {

                        break;

                    }

                } while (true);

                thaphanoi(n, a, c, b);

            }

            if (n == 1) chuyen(1, a, c);

            else

            {

                thaphanoi(n - 1, a, b, c);

                chuyen(n, a, c);

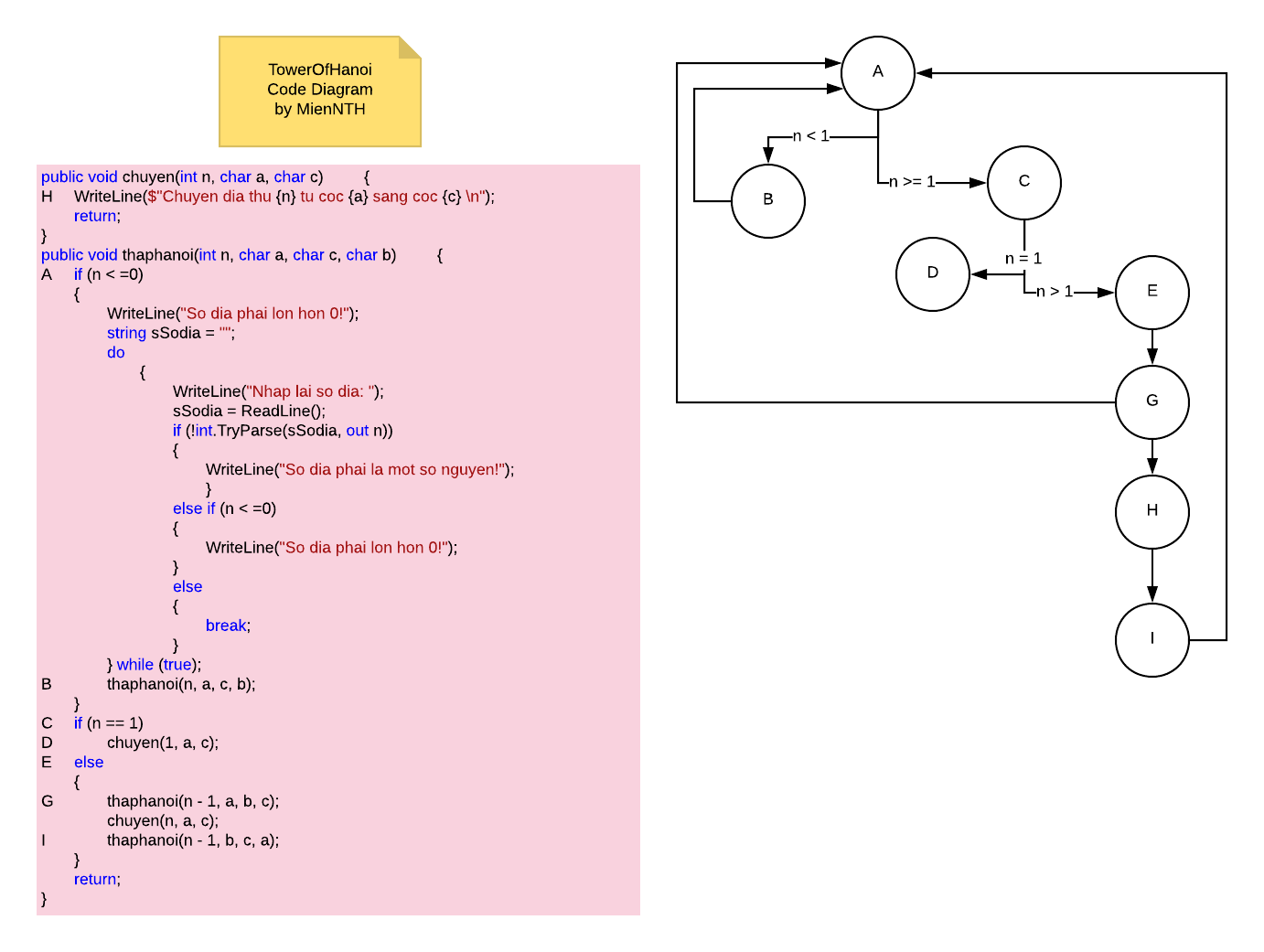
                thaphanoi(n - 1, b, c, a);

            }

            return;

        }

## Code diagram



## Testing types and techniques applied

### Black-box testing

#### Specification-based

Test that it has 3 branches with 2 if-else checks

#### Equivalence partitioning

|  |  |  |
| --- | --- | --- |
| **Partition 1** | **Partition 2** | **Partition 3** |
| N <= 0 | N = 1 | N > 1 |

#### Boundary value analysis

|  |  |  |
| --- | --- | --- |
| **Boundary 1** | **Boundary 2** | **Boundary 3** |
| N = 0  N = Integer.MinValue | N = 1 | N = 2  N = Integer.MaxValue |

### White-box testing

#### Statement coverage

public void chuyen(int n, char a, char c)

         {

**1**  WriteLine($"Chuyen dia thu {n} tu coc {a} sang coc {c} \n");

            return;

        }

        public void thaphanoi(int n, char a, char c, char b)

        {

**2** if (n < 1)

            {

**3**  WriteLine("So dia phai lon hon 0!");

                string sSodia = "";

                do

                {

                    WriteLine("Nhap lai so dia: ");

                    sSodia = ReadLine();

                    if (!int.TryParse(sSodia, out n))

                    {

                        WriteLine("So dia phai la mot so nguyen!");

                    }

                    else if (n < 1)

                    {

                        WriteLine("So dia phai lon hon 0!");

                    }

                    else

                    {

                        break;

                    }

                } while (true);

**4** thaphanoi(n, a, c, b);

            }

**5** if (n == 1)

**6** chuyen(1, a, c);

**7**  else

            {

**8**   thaphanoi(n - 1, a, b, c);

**9**  chuyen(n, a, c);

**10**  thaphanoi(n - 1, b, c, a);

            }

            return;

        }

Total statements in code: 10

#### Decision coverage

Total decisions in code: 2

If (n <= 0) and if (n == 1)

#### Path coverage

Nodes: A-I