### 1.1

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace first

{

internal class first

{

static void Main(string[] args)

{

string name = Console.ReadLine();

string surname = Console.ReadLine();

Console.WriteLine("hello, " + name + " " + surname);

}

}

}

### 1.2

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace first

{

internal class first

{

static void Main(string[] args)

{

int year = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("your age: " + (2023 - year));

}

}

}

### 1.3

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace first

{

internal class first

{

static void Main(string[] args)

{

int one = Convert.ToInt32(Console.ReadLine());

int two = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("the sum is: " + (one + two));

}

}

}

### 1.4

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

int one = Convert.ToInt32(Console.ReadLine());

Console.WriteLine($"the numbers are: {one - 1}, {one}, {one + 1}");

}

}

}

### 1.5

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

int one = Convert.ToInt32(Console.ReadLine());

int two = Convert.ToInt32(Console.ReadLine());

Console.WriteLine($"the sum is: {one + two}");

Console.WriteLine($"the sub is: {one - two}");

}

}

}

### 2.1

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

int one = Convert.ToInt32(Console.ReadLine());

if (one % 3 == 0)

{

Console.WriteLine($"{one} divides by 3");

}

else

{

Console.WriteLine($"{one} does NOT divide by 3");

}

}

}

}

### 2.2

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

int one = Convert.ToInt32(Console.ReadLine());

if ((one % 5 == 2) & (one % 7 == 1))

{

Console.WriteLine($"{one} is correct");

}

else

{

Console.WriteLine($"{one} is NOT correct");

}

}

}

}

### 2.3

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

int one = Convert.ToInt32(Console.ReadLine());

if (one < 0)

{

one = -one;

}

Console.WriteLine($"{one / 1000} thousands");

}

}

}

### 2.4

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

int one = Convert.ToInt32(Console.ReadLine());

if (one <= 10 & one >= 5)

{

Console.WriteLine($"{one} is correct");

}

else

{

Console.WriteLine($"{one} is NOT correct");

}

}

}

}

### 3.1

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

try

{

string one = Console.ReadLine();

bool isNumber1 = int.TryParse(one, out int numericValue1);

if (isNumber1 == true)

{

int one\_n = Convert.ToInt32(one);

if (one\_n % 3 == 0 & one\_n % 7 == 0)

{

Console.WriteLine($"{one\_n} is correct");

}

else

{

Console.WriteLine($"{one\_n} is NOT correct");

}

}

else

{

throw new ArgumentException(nameof(isNumber1), "Argument1IsNotAnInteger");

}

}

catch (ArgumentException)

{

Console.WriteLine("Incorrect type of value");

}

}

}

}

### 3.2

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

try

{

string one = Console.ReadLine();

string two = Console.ReadLine();

bool isNumber1 = int.TryParse(one, out int numericValue1);

bool isNumber2 = int.TryParse(two, out int numericValue2);

if (isNumber1 == true & isNumber2 == true)

{

int one\_n = Convert.ToInt32(one);

int two\_n = Convert.ToInt32(two);

if (one\_n > two\_n)

{

Console.WriteLine($"{one\_n} is more");

}

else if (two\_n > one\_n)

{

Console.WriteLine($"{two\_n} is more");

}

else

{

Console.WriteLine("nums are equal");

}

}

else if (isNumber1 == false)

{

throw new ArgumentException(nameof(isNumber1), "Argument1IsNotAnInteger");

}

else if (isNumber2 == false)

{

throw new ArgumentException(nameof(isNumber2), "Argument2IsNotAnInteger");

}

}

catch (ArgumentException)

{

Console.WriteLine("Incorrect type of value");

}

}

}

}

### 3.3

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

try

{

string one = Console.ReadLine();

bool isNumber = int.TryParse(one, out int numericValue);

if (isNumber == true)

{

int num = Convert.ToInt32(one);

List<string> days = new List<string>() { "monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday" };

if (num > 7 | num < 1)

{

Console.WriteLine("Incorrect number");

}

else

{

Console.WriteLine(days[num - 1]);

}

}

else

{

throw new ArgumentException(nameof(isNumber), "ArgumentIsNotAnInteger");

}

}

catch (ArgumentException)

{

Console.WriteLine("Incorrect type of value");

}

}

}

}

### 3.4

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

try

{

Console.WriteLine("enter the number:");

string one = Console.ReadLine();

bool isNumber1 = int.TryParse(one, out int numericValue1);

if (isNumber1 == true)

{

int num = Convert.ToInt32(one);

int sum = 0;

int i = 1;

while (i <= num)

{

sum += i;

i += 2;

}

Console.WriteLine($"summ is {sum}");

}

else

{

throw new ArgumentException(nameof(isNumber1), "Argument1IsNotAnInteger");

}

}

catch (ArgumentException)

{

Console.WriteLine("Incorrect type of value");

}

}

}

}

### 3.5

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

try

{

Console.WriteLine("enter the number of Fibonacci nums:");

string one = Console.ReadLine();

bool isNumber1 = int.TryParse(one, out int numericValue1);

if (isNumber1 == true)

{

int num = Convert.ToInt32(one);

int first = 0;

int second = 1;

Console.WriteLine("Fibonacci row: ");

int i = 0;

while (i < num)

{

Console.WriteLine($"{first} ");

int temp = first + second;

first = second;

second = temp;

i++;

}

}

else

{

throw new ArgumentException(nameof(isNumber1), "Argument1IsNotAnInteger");

}

}

catch (ArgumentException)

{

Console.WriteLine("Incorrect type of value");

}

}

}

}

### 4.1

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

try

{

Console.WriteLine("enter the length:");

string one = Console.ReadLine();

bool isNumber1 = int.TryParse(one, out int numericValue1);

if (isNumber1 == true)

{

int num = Convert.ToInt32(one);

if (num < 1)

{

throw new ArgumentException(nameof(num), "ArgumentIsNotCorrect");

}

int[] numbers = new int[] {};

for (int i = 2; i < num \* 5; i+=5)

{

Array.Resize(ref numbers, numbers.Length + 1);

numbers[numbers.Length - 1] = i;

}

Console.WriteLine("[{0}]", string.Join(", ", numbers));

}

else

{

throw new ArgumentException(nameof(isNumber1), "ArgumentIsNotAnInteger");

}

}

catch (ArgumentException)

{

Console.WriteLine("Incorrect type of value");

}

}

}

}

### 4.2

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

double[] numbers = new double[] { };

for (int i = 0; i < 11; i++)

{

Array.Resize(ref numbers, numbers.Length + 1);

numbers[numbers.Length - 1] = Math.Pow(2, i);

}

Console.WriteLine("[{0}]", string.Join(", ", numbers));

Array.Reverse(numbers);

Console.WriteLine("[{0}]", string.Join(", ", numbers));

}

}

}

### 4.3

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

char[] numbers = new char[] {'a', 'b', 'c', 'd', 'e', 'f'};

Console.WriteLine("[{0}]", string.Join(", ", numbers));

Array.Reverse(numbers);

Console.WriteLine("[{0}]", string.Join(", ", numbers));

}

}

}

### 4.4

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Main(string[] args)

{

try

{

Console.WriteLine("enter the length:");

string one = Console.ReadLine();

bool isNumber1 = int.TryParse(one, out int numericValue1);

if (isNumber1 == true)

{

int num = Convert.ToInt32(one);

if (num < 1)

{

throw new ArgumentException(nameof(num), "ArgumentIsNotCorrect");

}

int[] numbers = new int[] { };

for (int i = 0; i < num; i++)

{

Console.WriteLine($"enter the {i + 1} element:");

string elem = Console.ReadLine();

bool isNumber = int.TryParse(elem, out int numericValue);

if (isNumber == false)

{

throw new ArgumentException(nameof(isNumber), "ArgumentIsNotCorrect");

}

int el = Convert.ToInt32(elem);

Array.Resize(ref numbers, numbers.Length + 1);

numbers[numbers.Length - 1] = el;

}

int cp = 0;

int co = 0;

foreach (int j in numbers)

{

if (j > 0)

{

cp++;

}

else if (j < 0)

{

co++;

}

}

Console.WriteLine($"{cp} positive nums and {co} negative");

}

else

{

throw new ArgumentException(nameof(isNumber1), "ArgumentIsNotAnInteger");

}

}

catch (ArgumentException)

{

Console.WriteLine("Incorrect type of value");

}

}

}

}

### 5.1

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

namespace First

{

internal class First

{

static void Text(string res, params char[] symb)

{

for (int i = 0; i < symb.Length; i++)

{

res += symb[i].ToString();

}

Console.WriteLine($"result text: {res}");

}

static void Main(string[] args)

{

Console.WriteLine("enter text: ");

string text = Console.ReadLine().ToString();

Console.WriteLine("enter symbols:");

string symbs = Console.ReadLine().ToString();

char[] a = symbs.ToCharArray();

Text(text, a);

Console.ReadKey();

}

}

}

### 5.2

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

using System.Text.RegularExpressions;

class Program

{

static int[] MinMax(params int[] nums)

{

int minim = int.MaxValue;

int maxim = int.MinValue;

foreach (int num in nums)

{

if (num < minim)

{

minim = num;

}

if (num > maxim)

{

maxim = num;

}

}

return new int[] { maxim, minim };

}

static void Main()

{

Console.WriteLine("enter the numbers: ");

string n = Console.ReadLine().ToString();

int[] numbers = new int[] { };

foreach (int elem in Array.ConvertAll(Regex.Split(n, @"\s+"), int.Parse))

{

Array.Resize(ref numbers, numbers.Length + 1);

numbers[numbers.Length - 1] = elem;

}

int[] res = MinMax(numbers);

Console.WriteLine($"max value is: {res[0]}; min value is: {res[1]}");

}

}

### 5.3

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

using System.Text.RegularExpressions;

class Program

{

static int factorialRec(int n)

{

if (n <= 0)

{

return 1;

}

return n \* factorialRec(n - 2);

}

static int factorialNonrec(int n)

{

int res = 1;

while (n > 0)

{

res \*= n;

n -= 2;

}

return res;

}

static void Main()

{

Console.WriteLine("enter the length: ");

int num = Convert.ToInt32(Console.ReadLine());

int fr = factorialRec(num);

int fn = factorialNonrec(num);

Console.WriteLine($"recursive res: {fr}");

Console.WriteLine($"non-recursive res: {fn}");

}

}

### 5.4

using System;

using System.Collections.Generic;

using System.Formats.Asn1;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading.Tasks.Dataflow;

using System.Text.RegularExpressions;

class Program

{

static void Main()

{

//добавление, удаление, вывод эдля коллекции чисел

List<int> nums = new List<int>() { 1, 2, 3, 4, 5, 6 };

nums.Add(7);

nums.Remove(1);

foreach (int number in nums)

{

Console.WriteLine(number);

}

Console.WriteLine("{0}", String.Join(" ", nums));

//добавление, удаление, вывод для коллекции символов

List<char> symbols = new List<char>() { 'a', 'b', 'c', 'd', 'e', 'f' };

symbols.Add('j');

symbols.Remove('a');

foreach (char s in symbols)

{

Console.WriteLine(s);

}

Console.WriteLine("{0}", String.Join(" ", symbols));

//добавление, удаление, вывод для словаря

Dictionary<string, int> letters = new Dictionary<string, int>() {

{"A", 1 },

{"B", 2 },

{"C", 3 }

};

letters.Add("D", 4);

letters.Remove("A");

foreach (KeyValuePair<string, int> elem in letters)

{

Console.WriteLine($"{elem.Key}: {elem.Value}");

}

Console.WriteLine("{0}", String.Join(" ", letters));

}

}

### 

### **ООП**

using System;

using System.Text.RegularExpressions;

namespace First

{

internal class Program

{

static void Main(string[] args)

{

Realty room = new Room(20, 275000);

room.Show();

Realty flat = new Flat(3, 46, 233000);

flat.Show();

Realty house = new House(3, 8, 222, 530000);

house.Show();

Console.ReadKey();

}

}

abstract class Realty

{

public abstract string Name();

public abstract string Cost();

public abstract string Type();

public void Show()

{

Console.WriteLine($"Realty name: {Name()}\n" +

$"Cost value: {Cost()}\n" +

$"Type of realty: {Type()}\n");

}

}

class Room : Realty

{

double s;

double i;

public Room(double roomArea, double iCost)

{

s = roomArea;

i = iCost;

}

public double roomArea

{

get { return s; }

set { s = value < 0 ? 0 : value; }

}

public double iCost

{

get { return i; }

set { i = value < 0 ? 0 : value; }

}

public override string Name()

{

return "Room";

}

public override string Cost()

{

return (s\*i).ToString();

}

public override string Type()

{

if (i < 250000)

{

return "low class";

}

else if (i > 500000)

{

return "high class";

}

else

{

return "mid class";

}

}

}

class Flat : Realty

{

double n;

double a;

double u;

public Flat(double nRooms, double aArea, double uCost)

{

n = nRooms;

a = aArea;

u = uCost;

}

public double nRooms

{

get { return n; }

set { n = value < 0 ? 0 : value; }

}

public double aArea

{

get { return a; }

set { a = value < 0 ? 0 : value; }

}

public double uCost

{

get { return u; }

set { u = value < 0 ? 0 : value; }

}

public override string Name()

{

return "Flat";

}

public override string Cost()

{

return (a\*u + 10000\*n).ToString();

}

public override string Type()

{

if (u < 250000)

{

return "low class";

}

else if (u > 500000)

{

return "high class";

}

else

{

return "mid class";

}

}

}

class House : Realty

{

double f;

double h;

double r;

double k;

public House(double fFloors, double hRooms, double rArea, double kCost)

{

f = fFloors;

h = hRooms;

r = rArea;

k = kCost;

}

public double fFloors

{

get { return f; }

set { f = value < 0 ? 0 : value; }

}

public double hRooms

{

get { return h; }

set { h = value < 0 ? 0 : value; }

}

public double rArea

{

get { return r; }

set { r = value < 0 ? 0 : value; }

}

public double kCost

{

get { return k; }

set { k = value < 0 ? 0 : value; }

}

public override string Name()

{

return "House";

}

public override string Cost()

{

return (r\*k + 15000\*h + 30000\*f).ToString();

}

public override string Type()

{

if (k < 250000)

{

return "low class";

}

else if (k > 500000)

{

return "high class";

}

else

{

return "mid class";

}

}

}

}