FOPCS Tutoring Session

Nguyen Tri Tin

First Activity

1. Introduce yourself
   - Your name (the easiest way to call you ☺)
   - Your background
2. How are you studying so far?
   - How are you using the given solutions?

Tutoring session Objectives

- Be able to **make use of the given solutions**: from the question, how to reach the solution
- Understand and be able to apply a **simple technique** to **solve problems**
- Be able to **self-study** and make good use of the limited time

A sample problem

Print out all prime numbers from 5 to 1000

Method Definition

Consist of

1. Method name
2. Parameters
3. Return type
4. Method body
   5. Return value

```
static 3 int 1 min(
    2 int num1, int num2)
{
    if (num1 < num2)
     5 return num1;

 5 return num2;
}
```

4

A simple Problem Solving technique

1. What is/are the input?
2. What is/are the output?
3. From the input, how to produce the output?
4. Test

Input

What are given to your program?

What are their respective types?

Convert them into method's parameters

Example

Write a C# function that would sort a numeric array in descending order using the simplified selection sort method described.

1. What are given? (1 input) *A numeric array*
2. Input type: array of integer, or float, or double…
3. Convert to method parameters:

   public static void sort(**int[] arr**)

Practice 1

Write a function (static method) that would take in an integer and return the hexadecimal

1. What are given? (1 input) *An integer number*
2. Input type: *integer*
3. Convert to method parameters:

   public static string convert ***(int number)***

Practice 2

Given 2 strings, write a static method to find if the string s2 occurs in s1 and return an integer that would indicate the starting position of the word s2 in s1.

If the word does not occur, the return value should be –1.

1. What given? *(2 input) s1 and s2*
2. Input type: *String and String*
3. Convert to method parameters:

   *public static int contain****(string s1, string s2)***

Practice 3

Write a static method that prints all prime numbers less than 1000

1. What are given? *No*
2. Input type: *NA*
3. Convert to method parameters:

   *public static void print **()***

Output

What should be produced from the program?

Do we need to print anything?

Do we need to return anything?

Convert to the return type in method

Example

Write a C# function that would sort a numeric array in descending order using the simplified selection sort method described.

1. What should be produced? *The changed values of the given array*
2. Do we need to print anything? *No*
3. Do we need to return anything? *No*
4. Convert to return type

public static ***void*** *sort*(int[] arr)

Practice 1

Write a function (static method) that would take in an integer and return the hexadecimal.

1. What should be produced? *The respective hexadecimal*
2. Do we need to print anything? *No*
3. Do we need to return anything? *Yes, the hexadecimal*
4. Convert to return type

public static ***string***
convertToHexa(int number)

Practice 2

Given 2 strings, write a static method to find if the string s2 occurs in s1 and return an integer that would indicate the starting position of the word s2 in s1.

If the word does not occur, the return value should be –1.

1. What should be produced? *An integer indicating the starting position of word s2 in s2, or -1*
2. Do we need to print anything? *No*
3. Do we need to return anything? *Yes, the position*
4. Convert to return type

public static ***int*** *getContainPostion* (string s1, string s2)

Practice 3

Write a static method that prints all prime numbers less than 1000

1. What should be produced? *All prime numbers less than 1000*
2. Do we need to print anything? *Yes, the prime numbers*
3. Do we need to return anything? *No*
4. Convert to return type

public static ***void*** *printPrimeNumbers()*

From the input, how can we produce the output?

- We need to **think of steps**
- Each step may **become a smaller problem**
  - Solve **each** of them
  - Each of the **smaller problems** will follow the **same described technique**

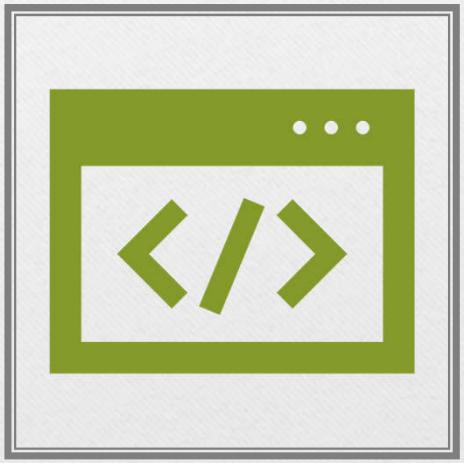
From input to output

I am new to Programming, from the input, it takes me **forever** to produce the output.

Is there any **tool** to **help** me?

From input to output

1. Pseudo-code
2. Break it into smaller problems
3. Learn from MANY problems/solutions



Pseudo code

- Let's forget about C#
- We solve problem **using English**
- Then, we **convert English to C#**

Advantages of Pseudo-code

- **Easy to read and write**
- Allow the programmer to **concentrate on the logic** of the problem
- Instead of the specific programming language‘s syntax

Pseudo-code conventions

- Steps are written in simple **English**
- Each step is written on a **separate line**
- Each set of steps is written **from top to bottom**, with only one entry and one exit

A Problem Solving technique summary

What is/are the input?

- What is/are given?
- What is/are the respective types?

What is the output?

- What to produce?
- Do we need to print anything?
- Do we need to return anything?

From the input, how can we produce the output

- **Thinks in steps**
- May **break it into smaller problems**
  - Each of the smaller problem will follow the same technique

Test

Now, let's solve this problem

Print out all prime numbers from 5 to 1000

Method Calling

```
int 3 min1 = 1 min(2 1, 2);
Console.WriteLine(min1);

Console.WriteLine(4 min(4, 3));
```

1. Call method **name**

2. Provide the **parameter values**

3. The **return result** can be assigned to a variable

4. Or, the **return result** can be used directly

Problem practice

Given a number, print all perfect numbers less than it

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself

You do this after this session

1. **Type, not read, not copy-and-paste**, the solution code
   - Open the solution in a Text Editor app on one window
   - Open Visual Studio, create an empty .cs file on another window
   - Type whatever in the solution
   - Fix bugs when necessary
   - Make sure the new .cs file can run

You do this after this session

2. After that, **try to understand the program** you have typed (following today's technique)
   - What is/are the input?
   - What is/are the output?
   - From the input, what are the steps to produce the output?