Attendance record

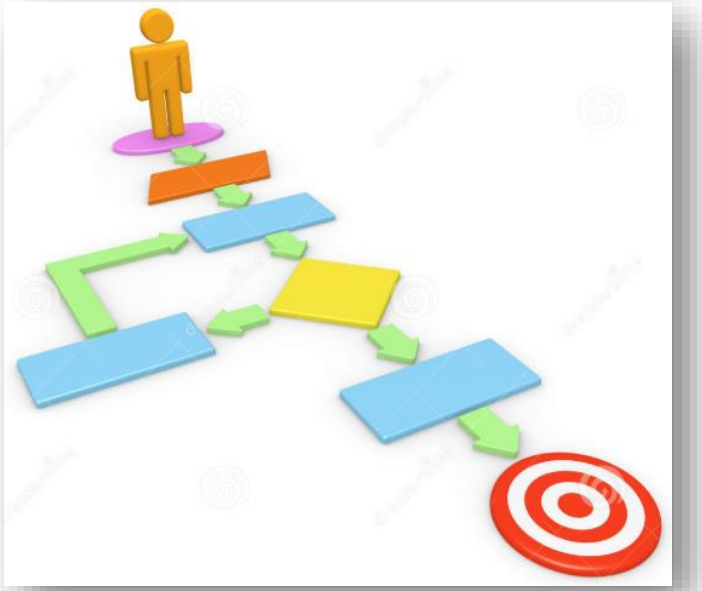
- To check what devices students are using to join class
- To check attendance

https://forms.gle/pipLP7fyJyZ92H2S7

Start class: **3:10pm**

DATA STRUCTURE & PROGRAMMING I

Chapter 1- Introduction



Prepared by: Dr. VALY Dona and Mr. NOU Sotheany
Updated and Lectured by: Mr. BOU Channa

OUTLINE

- ❑ Attendance and Survey students’ devices being used
- ❑ Introduction and Getting to know each other
- ❑ eLearning Moodle overview
- ❑ Introduction to Algorithms and Programming

Introduction

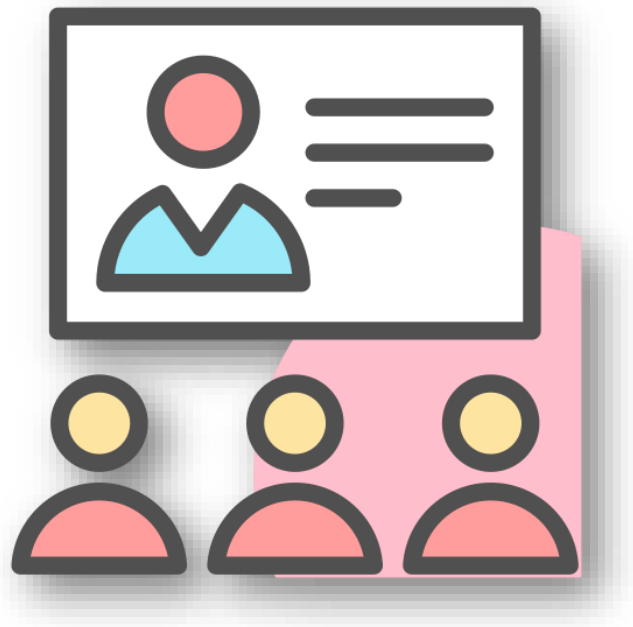
- Bachelor of Engineer, Computer Science, ITC (2011-2016)
- Master of Science, Software Engineering, SIIT Thailand (2016-2018)
- Experiences:
  - 2017-2018 : Teaching assistant in Python Lab, SIIT Thailand
  - 2018-2020 : Part-time lecturer at WU
  - 2018-Present : Part-time lecturer at CADT (NIPTICT)
  - 2018-Present : Full-time lecturer at GIC, ITC

*Additional materials will be shared at:
Channa BOU
https://www.youtube.com/channel/UC0iJz7fpHJwMAq73cIcYqQg

BOU Channa
bouchanna.itc@gmail.com
F307

Introduction

- YOUR TURN!



Course objectives

❑ Objectives of the course

- Upon completion of this course, students will be able to
  - Understand concepts relating to problem solving with the efficient use of **algorithms**
  - Understand a given problem
  - Analyze problem and provide a solution with step-by-step procedure
  - Write an algorithm for solving a given problem
  - Understand and use different types of data type in programming
  - Learn main concepts in programming
  - C programming for practical work in lab (TP)

Lecture overview

❑ Overall lectures

Mid-term

1. Introduction to algorithm
2. Basic data types and statements
3. Control structures and Loop

Final exam

4. Array
5. Sub-programs
6. Data structure

Semester I

Mid-term

7. Recursive
8. Pointers
9. Linked Lists

Final exam

10. Stacks and Queues
11. Sorting algorithms
12. Trees

13. Intro to OOP

Semester II

Teaching and learning strategy

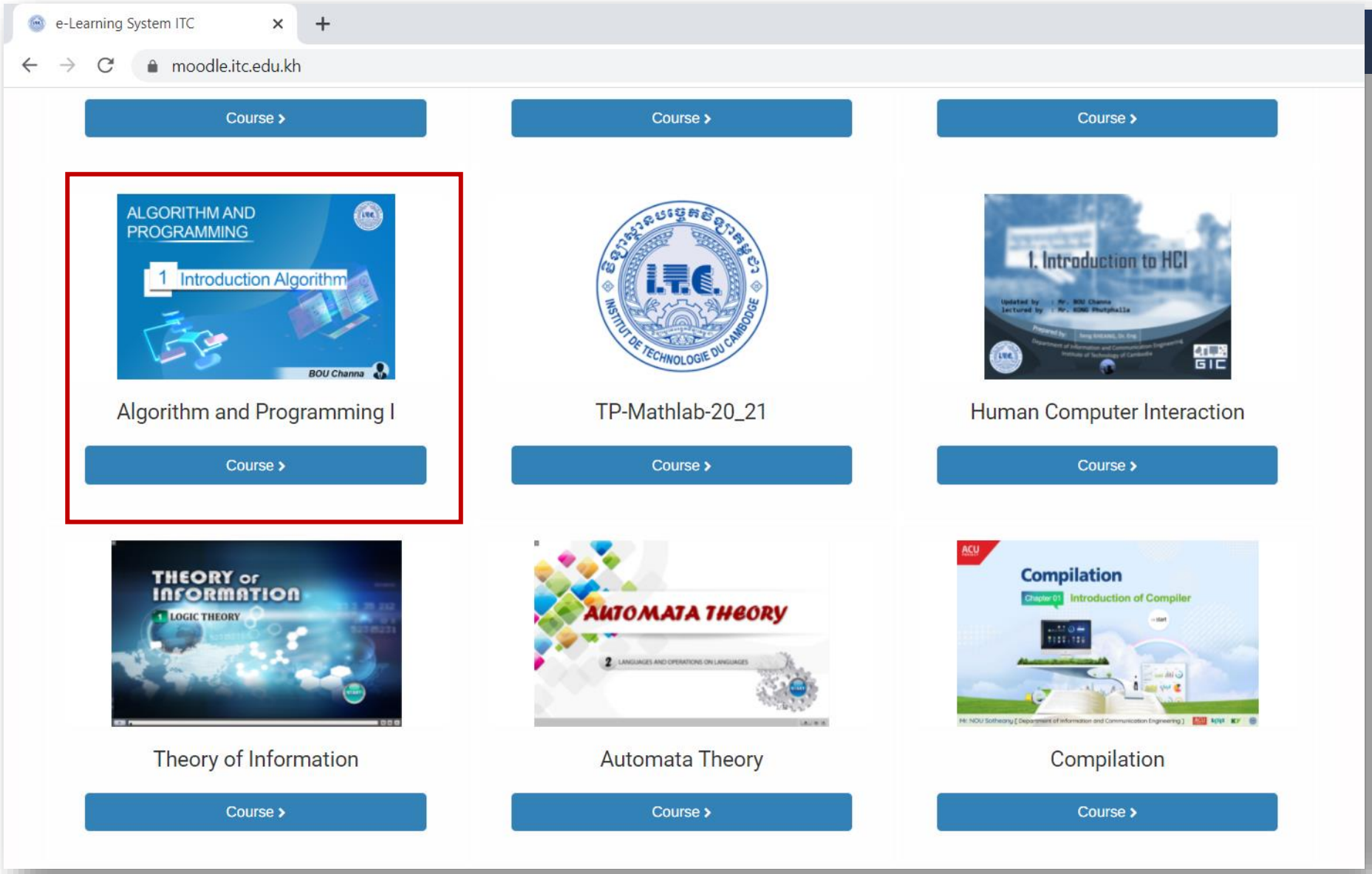
❑ Activity

- Lecture
- TD
- TP
- Quiz
- Mid-term exam
- Final exam

Class management

- Class on MS Team
  - Class code: bmwl6ko
- Join MS Team before class start
- Late management (Lab)
  - In 15mn: marked as late (L)
    - 2L = 1 absence
  - Late > 15mn
    - Allow to get in if suitable reasons

eLearning platform Link: moodle.itc.edu.kh



-Break: 10mn

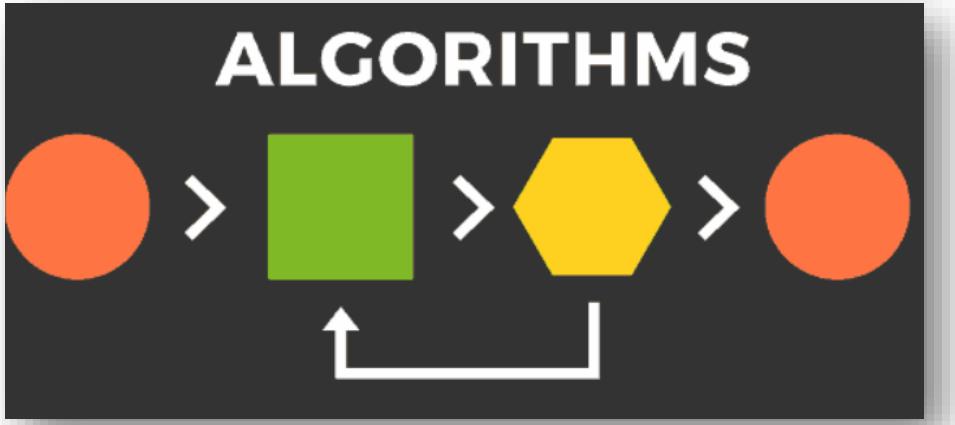
-Back: 2:45pm

Algorithm

❑ What is algorithm?

- An algorithm is a **step-by-step procedure** describing how to solve a problem / reach a goal
- It may take some *input*, *process the input* to achieve the goal, and finally display the *output/result* if needed

- Example:
  - How to make a fruit salad
  - How to validate an email address
  - How to add two numbers from a user's inputs



Algorithm Examples (1/3)

❑ Example 1: How to make a "Fruit Salad"

Ingredient (input)

Procedures

Product (output)

```
You will need:
2 bananas
2 strawberries
2 oranges
2 apple

What to do:
1. Wash your hands and clean your cooking area
2. Wash bananas, strawberries, oranges, apples
3. Peel bananas and oranges
4. Cut all the fruits according to recipe
5. Place fruit in a large bowl and mix

Output:
Fruit salad
```

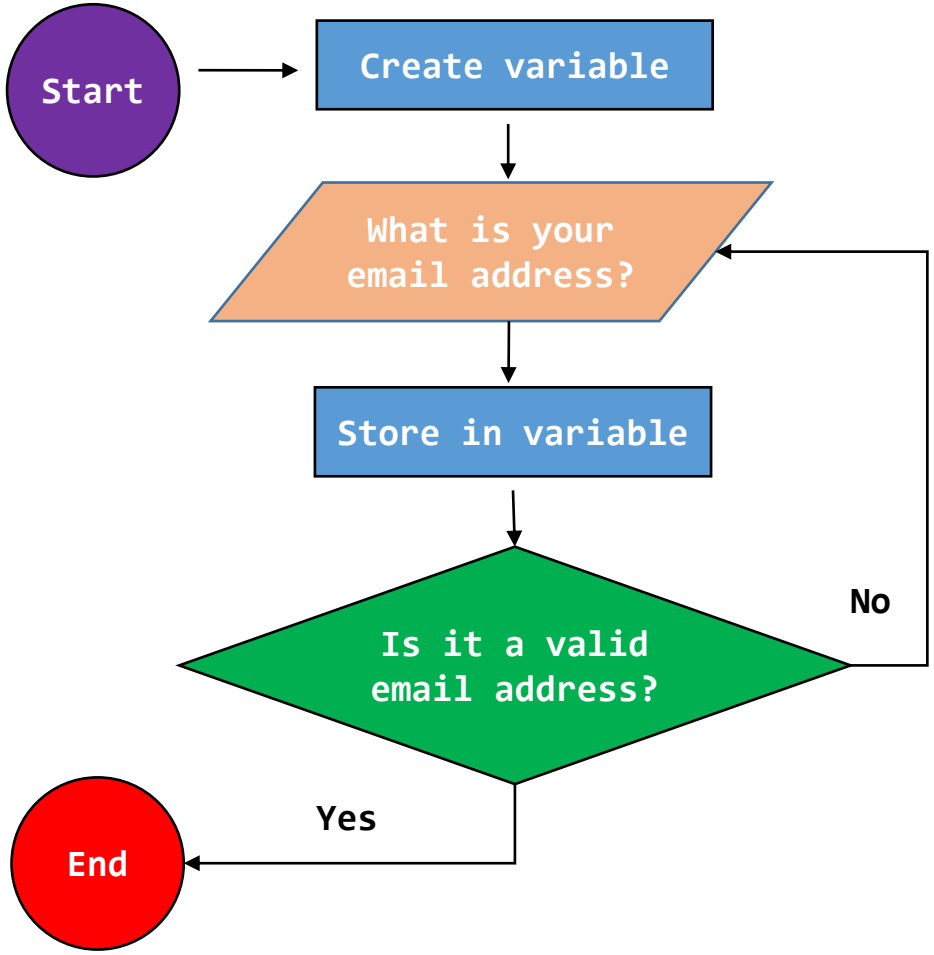
Algorithm Examples (2/3)

❑ Example 2: How to validate an input email address

START

- Declare variable *address* (text)
- Ask user to input address
- Do operation: check address if valid
  - If not valid, user input again
  - If valid, done.

STOP



Algorithm Examples (3/3)

❑ Example 3: How to add two numbers (more specific to coding)

START

- Declare variable a, b, c (Number)
- Get values a and b from user
- Do operation c = a +b
- Display c

STOP

Remarks

❑ Algorithm and Computer Programming Language

- An algorithm is not computer code
  - It's just written in simple English or whatever the programmer speaks
  - But it has a *start*, a *middle* and an *end*

- Converting algorithm to a specific **computer programming language** is needed to make computer understand and solve a given problem
  - **C**, C++, C#, Java, PHP, JavaScript etc.

Algorithm

❑ Why algorithm?

- ✓ It gives us idea how to deal with a problem
- ✓ It helps us to analyze a problem
- ✓ It also tells us how to solve a problem
- ✓ Generally, a problem can be solved by more than one algorithm

- Which algorithm is the best to solve a given problem?
  - Time consuming
  - Memory usage
  - Code Length of applicable programming language (how long is the applicable code)
  - Complexity

Algorithm

❑ Fundamental idea on making an algorithm?

- When a problem is given,
  - Check it whether there is an algorithm can solve it
  - Then how fast or complicated is your algorithm can solve the problem
  - Check the result whether it is what you need

Algorithm

❑ Main components of algorithm

Ingredients	=	Input	Input
Recipe	=	Procedure	
Fruit salad	=	Output	Output

Referred to 1st example on how to make a fruit salad

Algorithm

❑ How to write an algorithm?

1. Divide a given problem into sub-problems
2. Arrange sub-problem in order
3. Determine the solution for each sub-problem
4. Solve each sub-problem
5. Then, the whole problem is solved

Algorithm

❑ Important guidelines

- Clarity
  - Semantic (Meaning)
  - Syntactic (Grammar)

- Readability
  - Separation of different information
    - Declaration
    - Calculation or process
    - Displaying information
  - Indentation
    - Use tab when needed

Algorithm

❑ Structure of writing an algorithm

Break
Back 4:15pm

- Consist of two main parts
  1. Variable declaration (reserve memory space)
  2. Algorithm
     - **Get Input,**
     - **Process input,**
     - **Output result**

```
Var name1, name2 : Type of variable
Begin
     // your algorithms start here
      statement 1
      statement 2
      statement 3
End
```

An example of structure of an algorithm

Break

Algorithm

❑ Example #1 (Simple)

```
Var n1, n2, result : Number
Begin
    Get n1, n2 from user
    Do operation result = n1+n2
    Display result

End
```

Sum two input numbers and display the result

Algorithm

❑ Example #2

```
Var n1, n2, result : Number
Begin
    read(n1, n2)
    result ← (n1+n2)*2
    write(result)

End
```

Multiply the sum of two input numbers with 2 and display the result

Algorithm

❑ Example #3

```
Var n1, n2 : Number
Begin
    read(n1, n2)
    n1 ← (n1+n2)*2
    write(n1)

End
```

Multiply the sum of two input numbers with 2 and store in the 1st number variable

TD exercises

1. Write an algorithm to greet message. Ask a user for a name then display welcome message

   Input name: John

   Hi, John! Welcome to our department!

2. Write an algorithm to ask year of birth of a user. Tell age of the user based on his/her given year of birth.

   Input year of birth: 2000

   You were born in 2000 and you are 21 year old.

3. Write an algorithm to do basic math operations + - * /. Get two input numbers from user then do the operation above.

#1

```
Var name : text | sequence of characters
Begin
    write("Enter name: ")
    read(name)
    write("Hi ", name, "! welcome
        to class")
End
```

#2

```
Var y, age : Number

Begin
  Write("Input year birth: ")
  read(y)
  age ← 2021 - y
  Write("You were born in", y, ". You
        are", age, "year old")
End
```

Variable/Identifier

Variable/Identifier

❑ What and Why?

- A **variable** (also called **identifier**) is a holder of value which can be used and changed throughout the program
- It is used to store value and its value can be changed during the program run

- **Why need variable?**
  - Understandability and Readability
  - Maintainability
  - Ex: Exchange rate computation

```
t1 = a1 * 4100
t2 = a2 * 4100
t3 =  a3 * 4100
```

```
rate = 4100
t1 = a1 * rate
t2 = a2 * rate
t3 =  a3 * rate
```

What if the multiplier is changed to 4050?
=>You will need to change 3 times.
What if you have 100 lines using the multiplier (4100)?
=>Change 100 times ???

What if the multiplier is changed to 4050?
=>You just need to update value of **rate** to 4050

Variable/Identifier

❑ How to create a variable?

- A variable is created by variable declaration.
- We **declare a variable** in order to reserve memory space for it

- Syntax to declare a variable (for writing algorithm)

```
Var list_of_variables : Type of variable
```

keyword

Variable name
Use comma (,) if u have more than one variable to be created

Colon

Its type

Variable/Identifier

❑ Basic types of variables

- **Number**
  - Integer : A non-fractional number, e.g: 5
  - Float : A non-fractional and fractional number (real number), e.g: 4 or 5.8
- **Character** : A single letter. It is placed inside a single quote, e.g: 'M'
- **Sequence of character** : A string (a set of letters). It is placed inside a double quote, e.g: "Sok"
- NOTE: Different types of variables consume different memory space

- Examples of variable declaration:

```
Var gender: Character
Var age, number: Integer
Var name, title: Sequence of character
Var person_weight: Float
```

Variable or Identifier

❑ Naming a variable

- Rule for naming a variable
  - ✓ Start with a letter (small or capital is okay) or underscore
  - ✓ Not start with number or any symbol (+, -, @, #, !, &, *, /, ., ... etc)
  - ✓ No space is allowed.
    - If the name is long, use underscore (_) or camel case (my**V**ar)
- Naming convention (good practice)
  - Name of a variable should reflect the value it will store
  - The name should be readable and not too short
- Examples:

```
Var age: Integer
Var price: Float
Var name: Sequence of character
Var _tel: Integer
Var studentName: Sequence of character
Var work_position: Sequence of character
```

```
Var 7age: Integer
Var @price: Float
Var +name: Sequence of character
Var *tel: Integer
Var &student Name: Sequence of character
Var work Position: Sequence of character
```

Variable or Identifier

❑ NOTE: Naming a variable

- Name of a variable is **case sensitive** (Ex: age, Age, AGE are different names)

```
Var age: Integer
Var Age: Integer
Var aGe: Integer
Var AGE: Integer
```

- Each variable can't have the same name
  - Once a name is given to a variable, that name can't be used for naming another variable

```
Var sex: Character
Var sex: Sequence of character
Var price: Integer
Var price: Float
```

Variable or Identifier

❑ Assign value to a variable

- Assign value to a variable **means that** you give a value to that variable
- Syntax for assigning a value to a variable:
  - *<variable>* ← *<value* **or** *other variable (same type)* **or** *expression>*
- The type of value assigning to the variable must be the same as variable's type
- A variable can be used once a value was assigned to that variable
- Examples:

```
Var name, name2 : Sequence of character
Var val : Integer
name  ← "Sok"
name2 ← "Sao"
name  ← name2
val   ← 5
val   ← val*2
```

```
Var name, name2 : Sequence of character
Var val : Integer
name  ← "Sok"
name  ← "name2"
name  ← name2
val   ← name
val   ← "Dara"
```

Algorithm

❑ Constant

- It is a non-changeable value (its value is fixed for all algorithm)
- Syntax for declaring a constant:

```
const(identifier: Type) ← value or expression
```

keyword — name — Its type — Fixed value or result of an operation

- Examples:

```
const(YEAR: Integer) ← 2018
const(NEXT3YEAR: Integer) ← YEAR+3
const(EXCHANGE_RATE: Integer) ← 4100
```

- Rule for naming a constant and naming convention)
  - Same as variables' rules, **but**
  - All letters should be capitalized

Algorithm

❑ Entering data / Get user's inputs

- Syntax for entering a data:

```
read(<list of variables>)
```

keyword

Variable name
If more than one, use comma (,)

- It is an instruction to get input from user and assign to a variable
- **Examples:**
  - `read(number)` : ask a user for an input and store it in a variable named `number`
  - `read(name, surname)` : ask a user for two inputs and store it in the variables namely `name, surname`
  - `read(age)` : ask a user for an input and store it in the variable named `age`

Algorithm

❑ Displaying the information or data

- It is an instruction to display information
- Syntax for displaying a data:

```
write("static text", <variables, constants, or expression>)
```

keyword

Display what you write
(placed inside double quotes)

Placeholder/value
(just put a variable name or constant name or an expression/operation)

- Examples:
  - `write("Enter a number: ")`
  - `read(number)` `//Suppose user inputs number 7 here`
  - `write("This is the input from user: ", number)`
    - This is the input from user: 7
  - `write("The sum of ", number, " and 3 ", "is ", number+3)`
    - The sum of 7 and 3 is 10

Examples of Algorithms

❑ How to add two numbers

- Writing an algorithm for adding input two numbers. Then display the result.

```
Var num1, num2: integer
Var result: integer
Begin
    write("Enter the first number: ")
    read(num1)
    write("Enter the second number: ")
    read(num2)

    result ←  num1 + num2
    write("The sum is: ", result)
    write("The sum of ", num1, " and ", num2, "is: ", result)
End
```

Output: ...?

More Examples of Writing Algorithms

❑ Ex1: Calculate a given price with tax

- Suppose that the tax is 3% and a user is required to input a price. As a result we calculate and display the final price including the tax.

```
Var ..........: …Type…
Begin


        ....



End
```

More Examples of Writing Algorithms

❑ Ex1: Calculate a given price with tax

- Suppose that the tax is 3% and a user is required to input a price. As a result we calculate and display the final price including the tax.

```
const(TAX: int) ← 3
const(TITLE: string) ← "Result"
Var price, priceWithTax: float
Begin
    write("Give me the price exclude tax:")
    read(price)
    priceWithTax ← price + (price*TAX)/100
    write(TITLE)
    write(price, "dollars exclude tax.", priceWithTax, "dollars include tax")
End
```

Output: ...?

Examples of Algorithms

❑ Ex2: Student information

- Ask a student's information (name, department, birth's year). Then greet him/her with the following information:

```
Enter your name:
Which department are you from?:
Which year were you born?:

  Hello <name>! Welcome to <department>.
  You are <age> years old.
```

```
Var name, department: Sequence of characters
Var year_birth: Integer
Var age: Integer
Begin
    write("Enter your name: ")
    read(name)
    write("Which department are you from?: ")
    read(department)
    write("Which year were you born?": )
    read(year_birth)

    age ← 2021 – year_birth
    write("Hello", name, "! Welcome to ", department, ".")
    write("You are ", age, "years old.")
End
```

Examples of Algorithms

❑ Ex3: Currency exchange program

- Write an algorithm for computing an exchange rate USD-Riel. Suppose that 1USD= 4100 R. A user is required to input an amount of money in USD then a program will convert it into riel currency and display the following information

```
How much in US dollars do you want
to exchange?:
   Your exchange amount of
   <amountUSD> dollars are equal to:
   <amountRiel> when the exchange
   rate is 1 USD = 4100 riels
```

```
CONST(RATE: Float) ← 4100
Var amountUSD: Float
Var amountRiel: Float
Begin
    write("How much in US dollars do you want to exchange?: ")
    read(amountUSD)
    amountRiel ← amountUSD * RATE
    write("Your exchange amount of ", amountUSD, " dollars are
         equal to: ", amountRiel, "when the exchang rate is
         1 USD = ", RATE, "riels")
End
```

TD 1

❑ Write an algorithm for each of the question below:

1. Read a last name and first name from a user. Then display a phrase as follows:

```
What is your last name?
What is your first name?
      Welcome <lastname> <firstname>!
```

2. Read a number from a user and calculate square of that number then display its result.

```
Enter a number:
      The square of <number> is ….
```

3. Read two numbers (say n1 and n2) from a user then display their summation, subtract, and multiplication.

```
Enter the first number:
Enter the second number:
      The summation of <n1> and <n2> is: …
      The subtraction of <n1> and <n2> is: …
      The multiplication of <n1> and <n2> is: …
      The division of <n1> and <n2> is: …
```

TD 1

- Write an algorithm for each of the question below:

4. A program to ask user for firstname, lastname and department. Then display this message:

   Welcome to *department*, *lastname firstname*!

5. Ask a user to input height and base of a triangle. Calculate the surface of this triangle and display.

6. Ask a user for a, b and c length of a triangle. Calculate its surface using heron formula.

TP

- Coding in specific programming language: C Programming

C Programming

❑ Brief information

- First basic programming language
- Invented by Dennis Ritchie…….
- Mostly used to interact with machine (fast, secure, …)
- Short history
  - C was invented to write an operating system called UNIX.
  - C is a successor of B language which was introduced around the early 1970s.
  - The language was formalized in 1988 by the American National Standard Institute (ANSI).
  - The UNIX OS was totally written in C.
  - Most of the state-of-the-art software have been implemented using C.

C Programming

❑ Code syntax

- An extension of C programming is: **.c**
- Before starting out in C programming, make sure that you have a compiler
- A **compiler** turns code that you write into an **executable file** that computer can understand and run
- **Integrated Development Environment (IDE)** is a software application that provides comprehensive facilities (code color, code completion, …) to computer programmers for software development
  - **Code::Blocks** will be used

Download Codeblocks tool

codeblocks.org/downloads/binaries/

Code::Blocks

The IDE with all the features you need, having a consistent look, feel and operation across platforms.

- News
- Features
- Downloads
- User manual
- Forums
- Wiki
- License
- Donations

GPL v3 W3C CSS GetFirefox SOURCEFORGE

Sourceforge.net page.

NOTE: There are also more recent nightly builds available in the forums or (for Ubu
PPA repository. Please note that we consider nightly builds to be stable, usually.

NOTE: We have a Changelog for 20.03, that gives you an overview over the enhance
put in the new release.

NOTE: The default builds are 64 bit (starting with release 20.03). We also p
convenience.

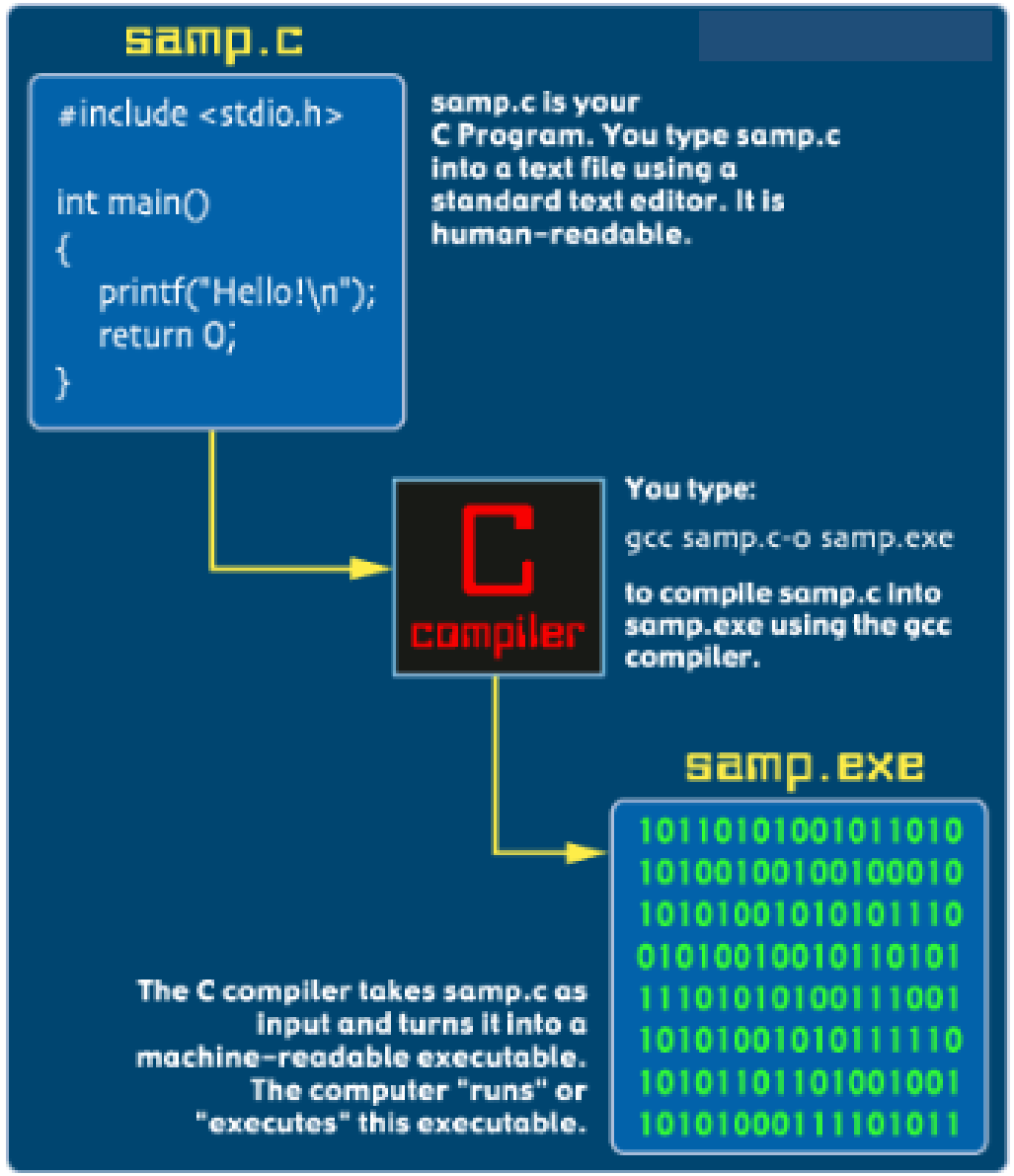
Microsoft Windows

File	Download from
codeblocks-20.03-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	FossHUB or Sourceforge.net

C Programming

❑ Compiling C language

- Source code
  - `filename.c`
- How to compile code with command prompt **cmd**
  - `gcc filename.c –o filename.exe`
- Executable file
  - `filename.exe`



Structure of C programming

❑ Structure

- Code

```
#include <stdio.h>

main(){

        write code here
        …………………………………………………………
}
```

Header (library)
stdio: standard input output

main function
-It runs its codes inside when the program executes
-Use { } and put code inside it

Code
write your code here

C Programming

❑ Code syntax

- All C program statement begins inside a function called main()
- The main function is always called when the program first executes
- To access the standard functions that comes with compiler, a header with the
  - #include <stdio.h> need to be included on top

- Structure of code

```
#include<stdio.h>
main(){
    … Write your code here …
}
```

```
#include<stdio.h>
main(){
        printf("Hello World!");
        printf("Welcome to C Programming.");
}
```

Example of our 1st Program in C programming

- Each line of code ends with a semicolon ;

❑ Commands in C

- Creating variable
  - ✓ int a, b, c; : Integer
  - ✓ char ch; : Character
  - ✓ float price; : Floating number
  - ✓ char name[10]; : String (sequence of character)

printf(“ asdfsadfasd ”);

printf(“asdfasdf %d”, variableName)

- Display variable value
  - It requires placeholder
    - %d: for int
    - %f: for float
    - %c: for char
    - %s: for string (sequence of character)
- **Note**: To insert special character, use the following inside double quote “ ”
  - \n: for newline
  - \t: for tab

test1.c

```c
#include <stdio.h>
int main(){
    int age=22;
    char sex='M';
    char name[]="Dara";
    printf("Welcome new student: \n");
    printf("Name: %s\n", name);
    printf("Sex: %c\n", sex);
    printf("Age: %d\n", age);
}
```

```
Welcome new student:
Name: Dara
Sex: M
Age: 22
```

❑ Example 1

```
#include <stdio.h>
main(){
        int age;
        char sex;
        char name[15];
        age=22;
        sex=‘M’;
        name=“Dara”; //error
        printf(“Welcome to new student: \n”)
        printf(“Name: %s”, name);
        printf(“Sex: %c”, sex);
        printf(“Age: %d”, age);
}
```

Example of displaying information

```
#include <stdio.h>
main(){
        int age=22;
        char sex=‘M’;
        char name[]=“Dara”;
        printf(“Welcome a new student: \n”)
        printf(“Name: %s”, name);
        printf(“Sex: %c”, sex);
        printf(“Age: %d”, age);
}
```

Example of displaying information
(assign value immediately when creating variable)

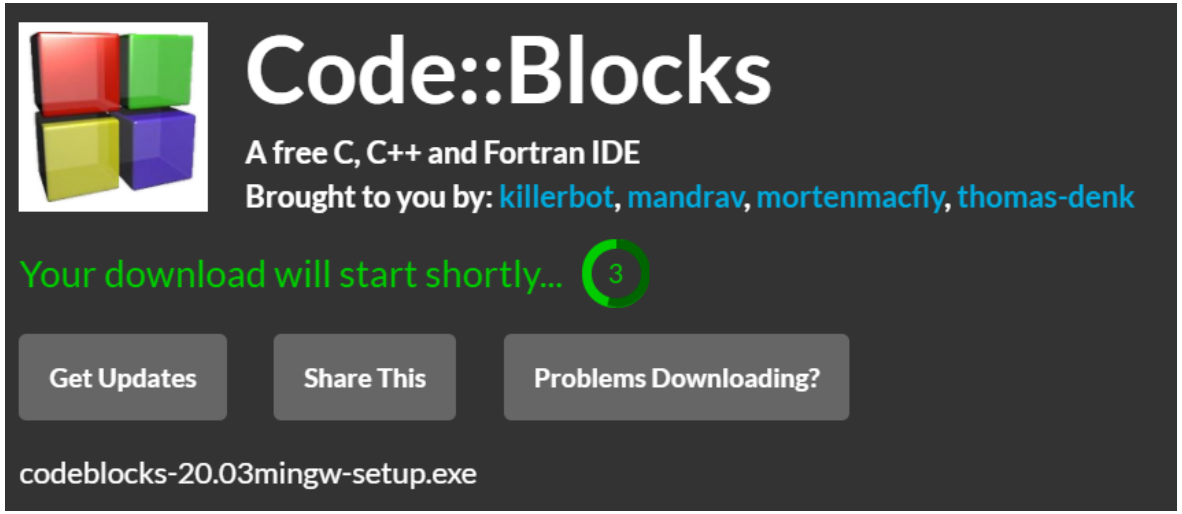
❑ Template structure

```
#include <stdio.h>
main(){
        //Declare variable here
        ……………….code here
        ……………….
        //Get input
        ……………….code here …
        ……………….
        //Process input
        ……………….code here …
        ……………….
        //Display info or show the output
        ……………….code here …
        ……………….
}
```

Homework

❑ Task 1

- Download and install a program
  - **CodeBlock IDE**
  - Go to the following link and download
    - https://sourceforge.net/projects/codeblocks/files/Binaries/20.03/Windows/codeblocks-20.03mingw-setup.exe/download



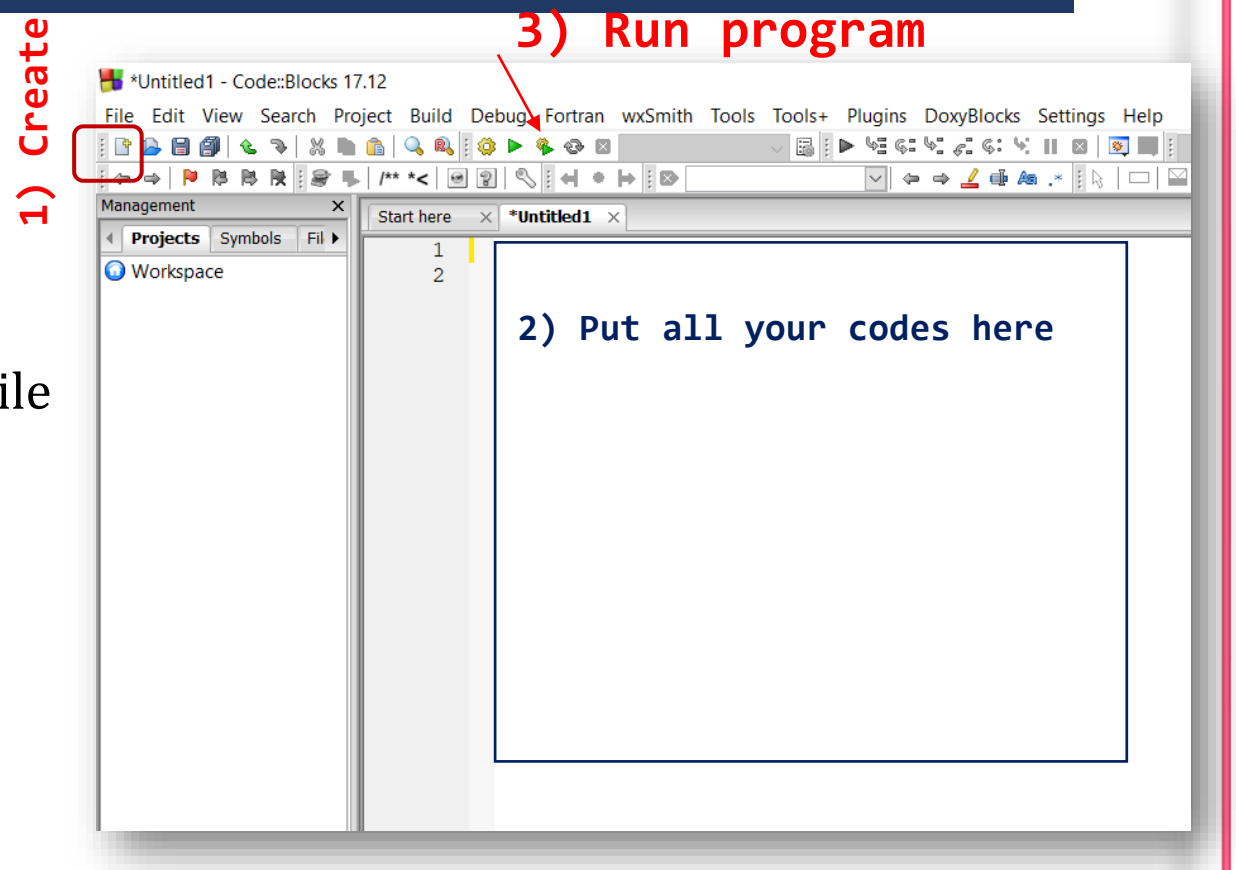
Homework

❑ Task 2

- Create your first program
  - Open the installed program
  - **File -> new -> empty file**
    - Or just click on new file icon
  - Copy code below and paste to the created file
  - Save file: ctrl+s
  - Type name of file follow by **.c**
    - Ex: Test.c

```
#include <stdio.h>
main(){
        printf(“"Hello World!")
        printf("Welcome to C Programming.")
}
```

Example of code in C programming



Interface of CodeBlock IDE

❑ Task 3

- Read book to getting started
  - https://www.tutorialspoint.com/cprogramming/cprogramming_tutorial.pdf