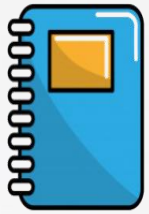
江西理工大学 Jiangxi University of Science and Technology
信息工程学院 School of information engineering



# 高级算法分析与设计

# Advanced Algorithm Analysis and Design

## Lecture 02:
## What is an algorithm?

Dr Ata Jahangir Moshayedi

EMAIL: ajm@jxust.edu.cn

Prof Associate ,
School of information engineering Jiangxi university of science and technology, China

LIVE Lecture series

Autumn _2022

江西理工大学 JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

# 高级算法分析与设计

## Advanced Algorithm Analysis and Design

LECTURE 01: What is an algorithm?

什么是算法?

# Agenda

# 目录

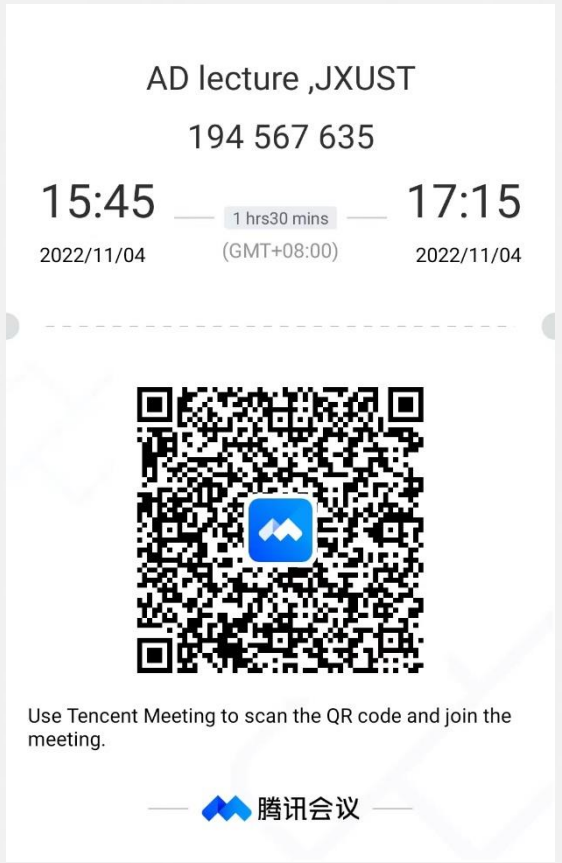
CONTENTS

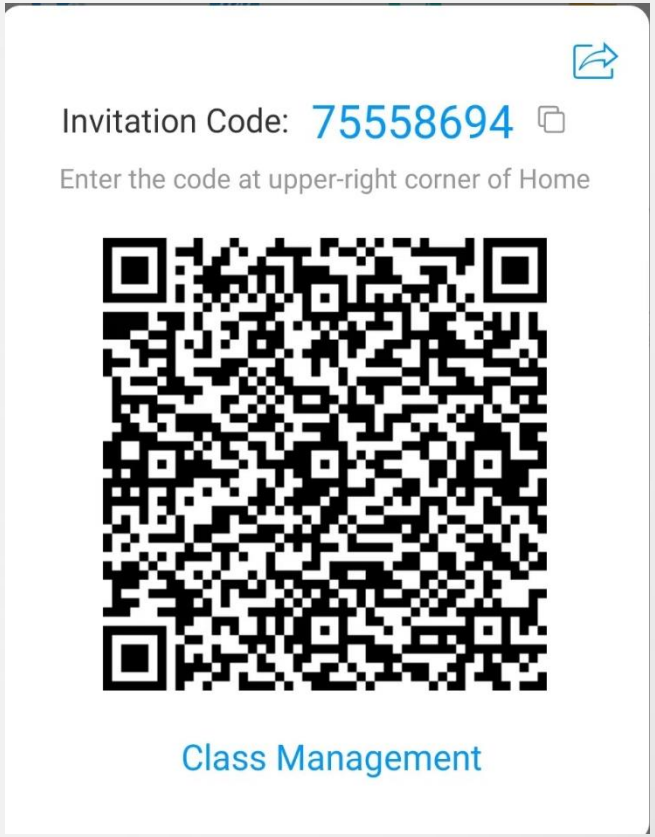


# Needed QR

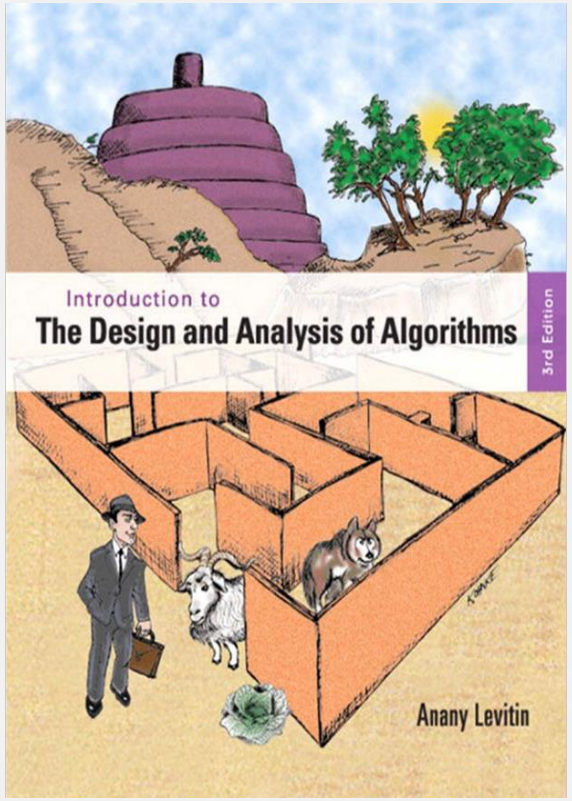

DR ajm lectures
Valid until 11/9 and will update upon joining group


Dr (阿 塔)
Jiangxi Ganzhou

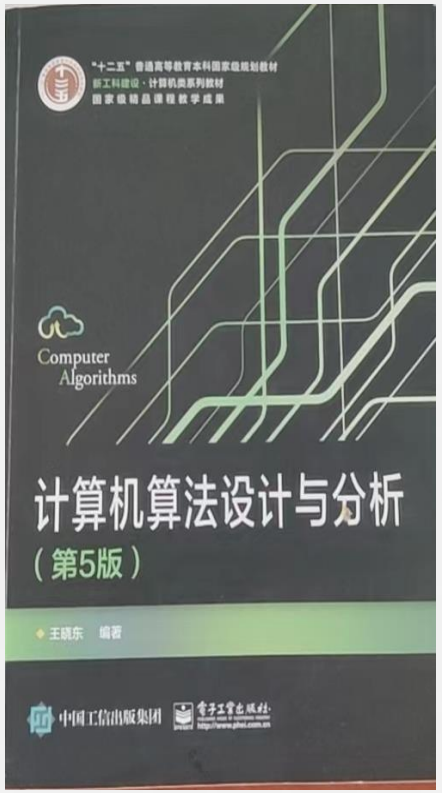

AD lecture ,JXUST
194 567 635
15:45
1 hrs30 mins
17:15
2022/11/04
(GMT+08:00)
2022/11/04
Use Tencent Meeting to scan the QR code and join the meeting.
腾讯会议


Invitation Code: 75558694
Enter the code at upper-right corner of Home
Class Management

# Reference book




1




2

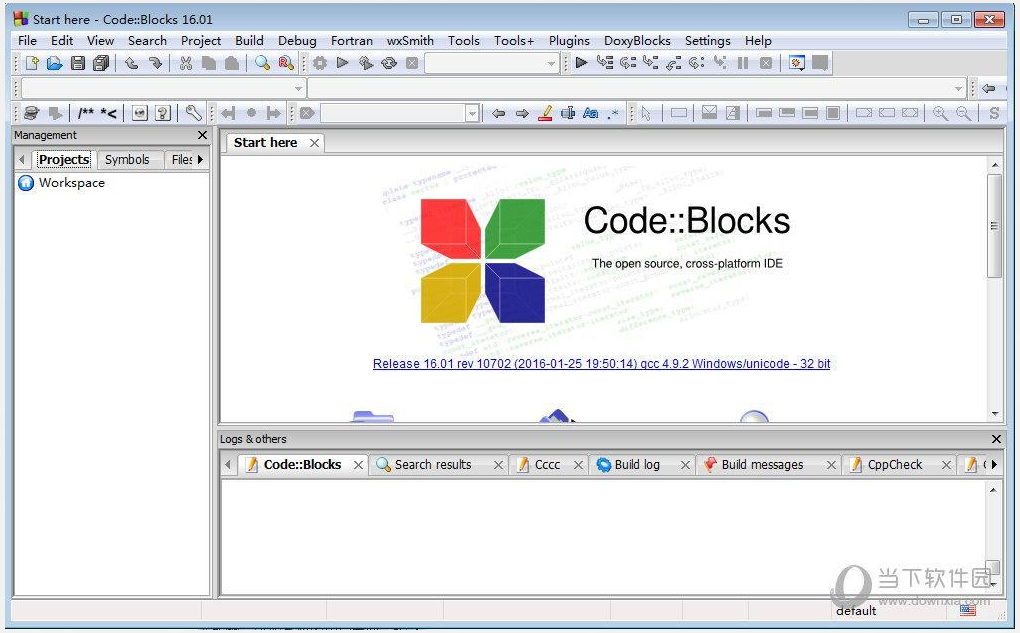
+

MY POWER POINT AND TASK

3

# Please remember .....

- for the doing the code in lecture you can use code block and for the task you can use PyCharm or any other IDE
- 对于在讲座中编写代码，您可以使用代码块，对于任务，您可以用PyCharm或任何其他IDE




# Which one is your goal

# 哪一个是你的目标

- you want to be a computer scientist?
- 你想成为一名计算机科学家吗?

- Is your goal to be a mundane programmer?
- 你的目标是做什么吗 一个平凡的程序员?

- Or a great leader and thinker?
- 还是一个伟大的领导者和思想家?

# What is meant by Algorithm Analysis?
# 算法分析是什么意思?

- Algorithm analysis is an important part of computational complexity theory, which provides theoretical estimation for the required resources of an algorithm to solve a specific computational problem.
- Analysis of algorithms is the determination of the amount of time and space resources required to execute it.
- 算法分析是计算复杂度理论的重要组成部分，它为解决特定计算问题所需的算法资源提供了理论估计。
- 算法的分析是确定执行该算法所需的时间和空间资源的量。

# Why Analysis of Algorithms is important?

- To predict the behavior of an algorithm without implementing it on a specific computer.
- It is much more convenient to have simple measures for the efficiency of an algorithm than to implement the algorithm and test the efficiency every time a certain parameter in the underlying computer system changes.
- It is impossible to predict the exact behavior of an algorithm. There are too many influencing factors.
- The analysis is thus only an approximation; it is not perfect.
- More importantly, by analyzing different algorithms, we can compare them to determine the best one for our purpose.

# 为什么算法的分析很重要?

- 预测算法而不在特定计算机上实现算法的行为。
- 与底层算法的效率发生变化相比，用简单的方法比实现算法和测试效率要方便得多。
- 要预测一个算法的确切行为是不可能的。影响因素过多。
- 因此，这个分析只是一个近似值；它并不是完美的。
- 更重要的是，通过分析不同的算法，我们可以比较它们，以确定适合我们的目的的最佳算法。

# 01

## What is an algorithm?

## 什么是算法?

# What is an algorithm?

What is an **algorithm**?

A simple, unambiguous, mechanical procedure to carry out some task.

Why algorithm instead of program?

1. Writing an algorithm is simpler (we don't need to worry about the detailed implementation, or the language syntax).
2. An algorithm is easier to read than a program written in, for instance, C.

- There are various definitions available for term Algorithm as:
- **"A set of rules for solving a problem in a finite number of steps"** - Dictionary
- **"A set of steps that are followed in order to solve a mathematical problem or to complete a computer process"** - Merriam-Webster
- **"An Algorithm is any well defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values as output"**- refer to References
- **"A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer."** - Google

# 什么是算法?

什么是算法?

一个简单、明确、机械的执行某些任务的机械程序。

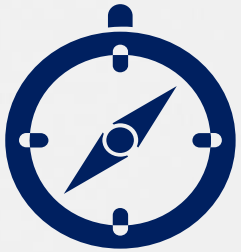
为什么要用算法来代替程序呢?

1. 编写算法更简单（我们不需要担心详细的实现或语言语法）。
2. 算法比用编写的程序更容易读取，例如C。

- 对于术语算法有多种定义，如:
- **“用有限数量的步骤解决问题的一套规则”-字典**
- **“为了解决数学问题或完成计算机过程所遵循的一组步骤”**
- **“算法是任何定义良好的计算过程，它以一些值，或一组值作为输入，并产生一些值，或一组值作为输出”-参考参考文献**
- **在计算或其他解决问题的操作中需要遵循的一套过程或规则，特别是由计算机。“-谷歌**

# Design and Analysis of Algorithms

## 算法的设计与分析



***Analysis: 分析:***

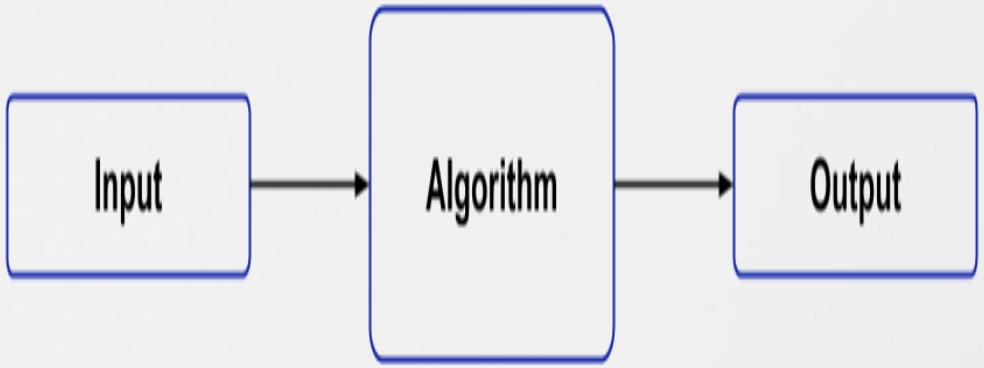
- predict the cost of an algorithm in terms of resources and performance
- 从资源和性能方面预测算法的成本

***Design: 设计***

- design algorithms which minimize the cost
- 设计出能使成本最小化的算法

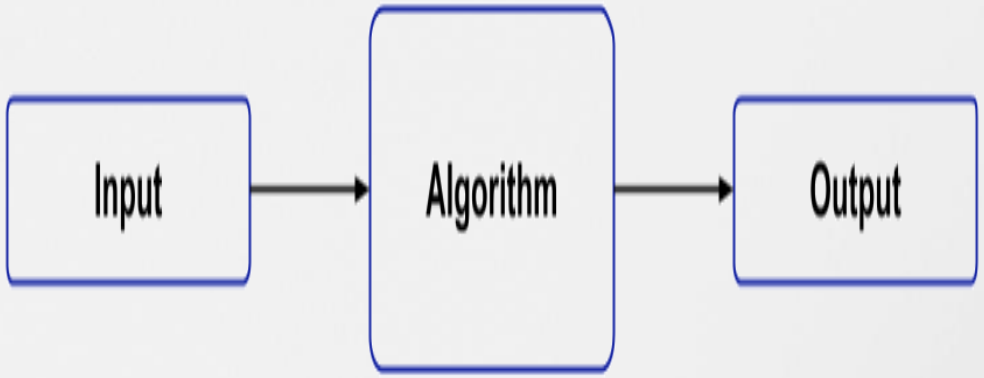
# Five characteristics of an Algorithm are:

- **Input**: Accept Zero or more inputs externally.
- **Output**: Must produce at least one output as result of procedure.
- **Definiteness**: Each instruction must be clear and unambiguous.
- **Finiteness**: If all the instructions are traced in algorithm, then for all cases the algorithm must terminate after a finite number of steps.
- **Effectiveness**: Every instruction must be very basic so that it can be carried out and must be feasible to perform on any machine.



# 一种算法的五个特征是：

- 输入：在外部接受零个或更多的输入。
- 输出：必须在操作过程中产生至少一个输出。
- 明确性：每条指令都必须明确和明确无误。
- 有限性：如果在算法中跟踪了所有的指令，那么对于所有的情况，算法都必须在有限的步骤数后终止。
- 有效性：每条指令都必须是非常基本的，以便它能够执行，并且必须能够在任何机器上执行。



# 如何表示一个算法?

## How to represent an algorithm?

1. Give a description in your own language, e.g., English, Spanish, …
    1. 用你自己的语言来描述一下，比如，英语，西班牙语，……。
2. Pseudo code
    2. 伪代码
3. Graphical
    3. 图形的

# Example :
# represent an algorithm add two numbers

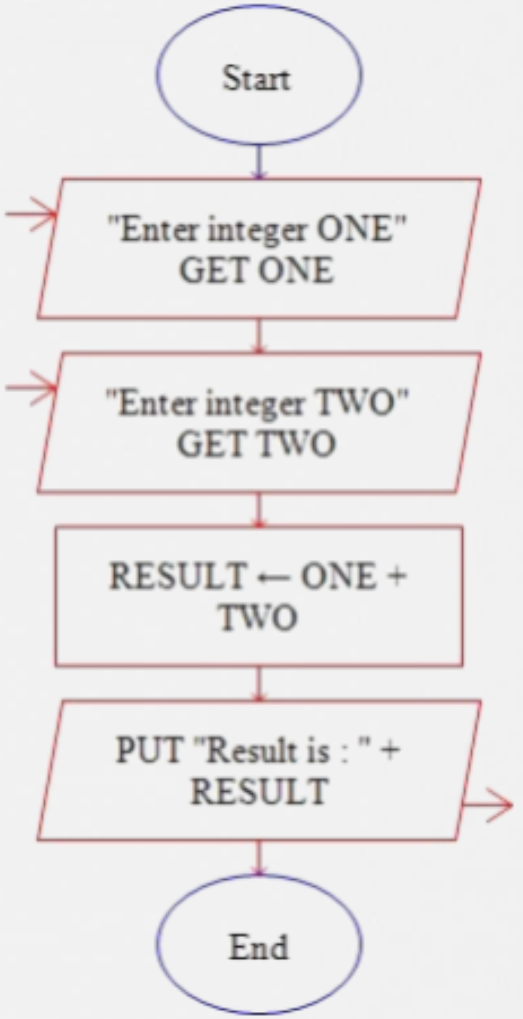
## • English language Type

1. Ask user to enter two integer numbers.
2. Add both the integer numbers.
3. Print the final sum value of two numbers

## • Pseudo Code Type

```
Algorithm SUM
    Input: Two Integer Numbers as ONE, TWO
    Output: sum of ONE & TWO as RESULT
       ONE <- User input required
       TWO <- User input required
    RESULT <- ONE + TWO
    return RESULT
```

## • Flowchart Representation



# 样例
# 表示一个算法，并添加两个数字

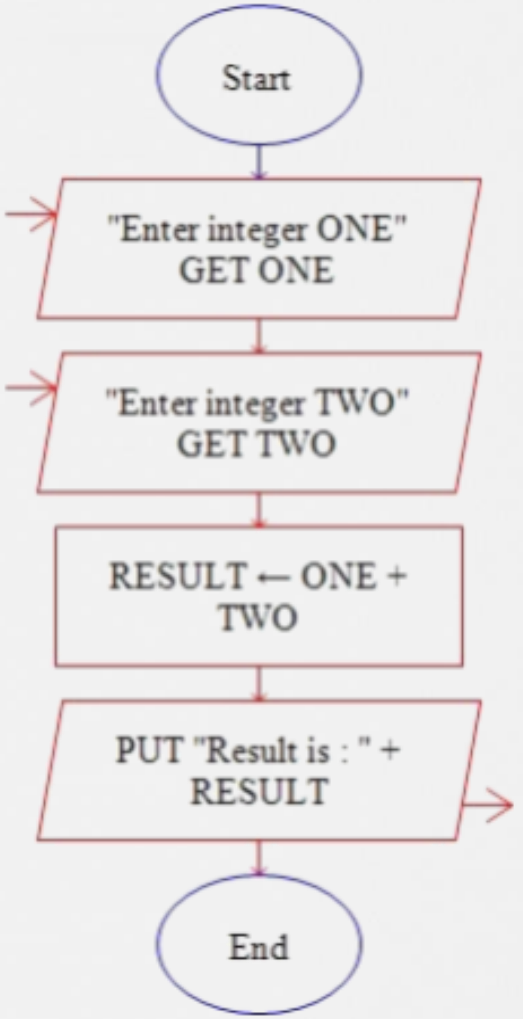
## • 英语类型

1. 要求用户输入两个整数。
2. 同时添加这两个整数。
3. 打印两个数字的最终和值

## • 伪代码类型

算法SUM
输入：两个整数为一，二
输出：一对和
一个<-需要用户输入
两个<-需要用户输入
结果<-ONE+2
返回结果

## • 流程图表示



# What is an algorithm? 什么是算法?

- Algorithms are the ideas behind computer programs.
- An algorithm is the thing that stays the same whether the program is in Pascal running on a Windows or is in JAVA running on a Macintosh!

- 算法是计算机程序背后的思想。
- 无论程序是在Windows上运行的Pascal中，还是在JAVA上运行的，算法都是保持不变的!

# What is an algorithm? (cont')

# 什么是算法？（续”）

- An algorithm is a precise and unambiguous specification of a sequence of steps that can be carried out to solve a given problem or to achieve a given condition.
- An algorithm accepts some value or set of values as input and produces a value or set of values as output.
- Algorithms are closely intertwined with the nature of the data structure of the input and output values

- 算法是对一系列可以用来解决给定问题或达到给定条件的步骤的精确和明确的规范。
- 算法接受一些值或一组值作为输入，并生成一个值或一组值作为输出。
- 算法与输入值和输出值的数据结构的本质紧密相连

# How to express algorithms?

# 如何表达算法?

提高精度 Increasing precision

English 英语

Pseudocode 伪码

Real programming languages

真正的编程语言

Ease of expression 表达方便

Describe the *ideas* of an algorithm in English.

Use pseudocode to clarify sufficiently tricky details of the algorithm.

用英语来描述一个算法的思想。

使用伪代码来澄清足够棘手的算法细节。

# How to express algorithms?

提高精度 Increasing precision



English 英语

Pseudocode 伪码

Real programming languages

真正的编程语言

Ease of expression 表达方便

To understand / describe an algorithm:

Get the big idea first.

Use pseudocode to clarify sufficiently tricky details

要理解/描述一种算法：

先出个大概想法。

使用伪代码来澄清足够棘手的细节

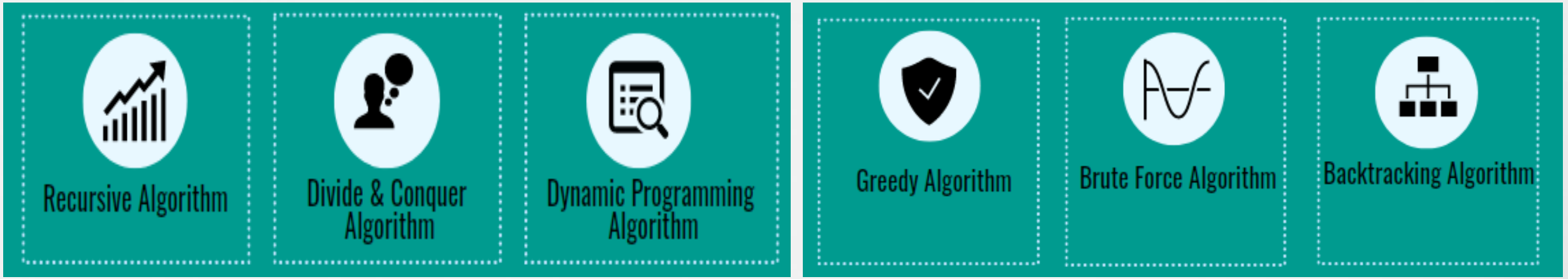
# Example: sorting

- Input: A sequence of N numbers $a_1 \ldots a_n$
- Output: the permutation (reordering) of the input sequence such that $a_1 \leq a_2 \ldots \leq a_n$.
- Possible algorithms you've learned so far
  - Insertion, selection, bubble, quick, merge, …
  - More in this course
- We seek algorithms that are both *correct* and *efficient*

- 输入：一个由N个数字a组成的序列$_1 \cdots a_n$
- 输出：输入序列的排列（重新排序），使$a_1 \leqslant a_2 \cdots \leqslant a_n$.
- 到目前为止，你已经学到的可能的算法
  - 插入，选择，气泡，快速，合并，……。
  - 本课程更多内容
- 我们寻找既正确又有效的算法

# Types of algorithms 算法类型

- We will talk later about them
- 我们稍后会讨论它们



# What is an Algorithm?

- Steps used to solve a problem
- Problem must be
  - Well defined
  - Fully understood by the programmer
- Steps must be
  - Ordered
  - Unambiguous
  - Complete

## *Program Development*

1. Understand the problem
2. Represent your solution (your algorithm)
   - Pseudocode
   - Flowchart
3. Implement the algorithm in a program
4. Test and debug your program

# 什么是算法?

- 用来解决问题的步骤
- 问题必须是
  - 明确的
  - 完全理解
    由程序员
- 步骤必须是
  - 订购
  - 明确的
  - 完成

***项目开发***

1. 了解问题
2. 表示您的解决方案（您的算法）
   - 伪码
   - 流程图
3. 在一个程序中实现该算法
4. 测试和调试您的程序

# Algorithm VS Pseudocode

- An **algorithm** is a procedure for solving a problem in terms of the actions to be executed and the order in which those actions are to be executed.
- An algorithm is merely the sequence of steps taken to solve a problem.
- The steps are normally "sequence," "selection, " "iteration," and a case-type statement.

- **Pseudocode** is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.
- The rules of Pseudocode are reasonably straightforward.
- All statements showing "dependency" are to be indented.
- These include while, do, for, if, switch.

# 算法VS伪码

- 算法是根据要执行的动作和执行这些动作的顺序来解决问题的一个过程。
- 一个算法仅仅是为解决一个问题所采取的一系列步骤。
- 这些步骤通常是“序列”、“选择”、“迭代”和一个大小写类型的语句。

- 伪代码是一种人工的非正式语言，可以帮助程序员开发算法。伪代码是一个“基于文本”的细节（算法）设计工具。
- 伪代码的规则相当简单。
- 所有显示“依赖性”的语句都要被缩进。
- 这些包括当，做，为，如果，切换。

# Pseudocode

- **Definition 1:** A way of expressing algorithms that uses a mixture of *English phrases* and *indention* to make the steps in the solution explicit
- There are no grammar rules in pseudocode
- Pseudocode is not case sensitive

- **Definition 2:** A pseudocode is an informal way of writing a program. However, it is not a computer program. It only represents the algorithm of the program in natural language and mathematical notations.
- Besides, there is no particular programming language to write a pseudocode.
- Unlike in regular programming languages, there is no syntax to follow when writing a pseudocode. Furthermore, it is possible to use pseudocodes using simple English language statements.

# 伪码

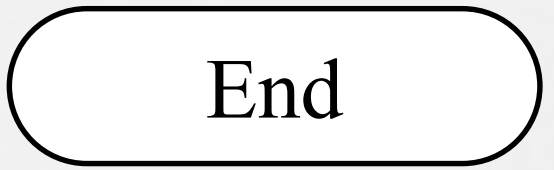
- 定义1：一种表达算法的方法，它混合使用英语短语和插入语来使解决方案中的步骤显式
- 在伪代码中没有语法规则
- 伪代码不区分大小写

- 定义2：伪代码是编写程序的一种非正式方式。然而，它并不是一个计算机程序。它只用自然语言和数学符号表示程序的算法。
- 此外，也没有特定的编程语言来编写伪代码。
- 与常规编程语言不同，在编写伪代码时不需要遵循任何语法。此外，还可以使用简单的英语语言语句来使用伪代码。

# Flowchart and Flowchart Symbols

- A flowchart is a type of diagram that represents an algorithm , showing the steps as boxes of various kinds
- [ex: rectangles, diamonds, ovals], and their order by connecting these with arrows
- 流程图是一种表示算法的图，以各种方框的形式显示步骤
- ［例如：矩形、菱形、椭圆形］，以及用箭头连接它们的顺序

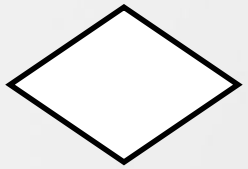

Start Symbol
开始符号




End Symbol
结束符号

Data Processing Symbol
数据处理符号

Input/Output
Read/Print
输入输出
读/打印

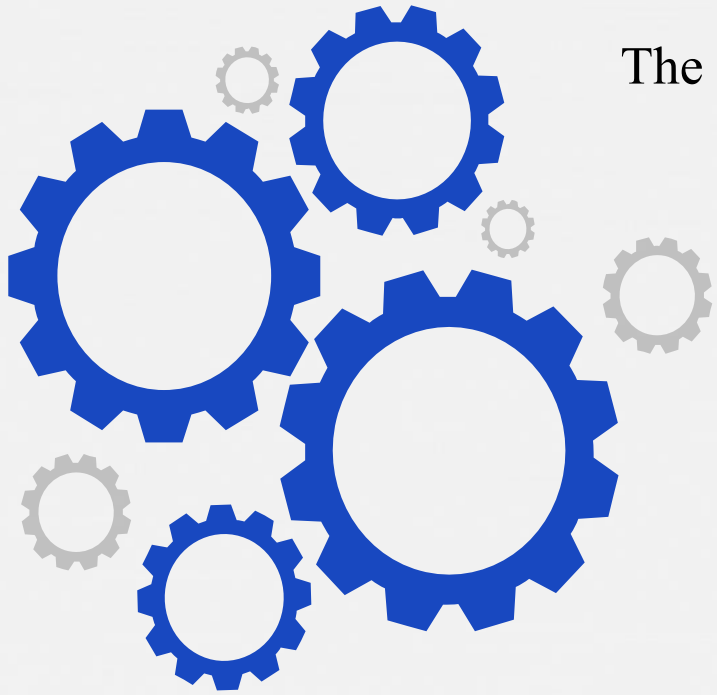


Decision Symbol
决策符号

Flow Control Arrows
流控制箭头

# Some Keywords That Should be Used

# 一些关键字这应该被使用



For looping and selection,

The keywords that are to be used include

- Do While...EndDo;
- Do Until...Enddo;
- Case...EndCase;
- If...Endif; Call ... with (parameters);
- Call; Return ....;
- Return;
- When;

Always use scope terminators for loops and iteration.

对于循环和选择，

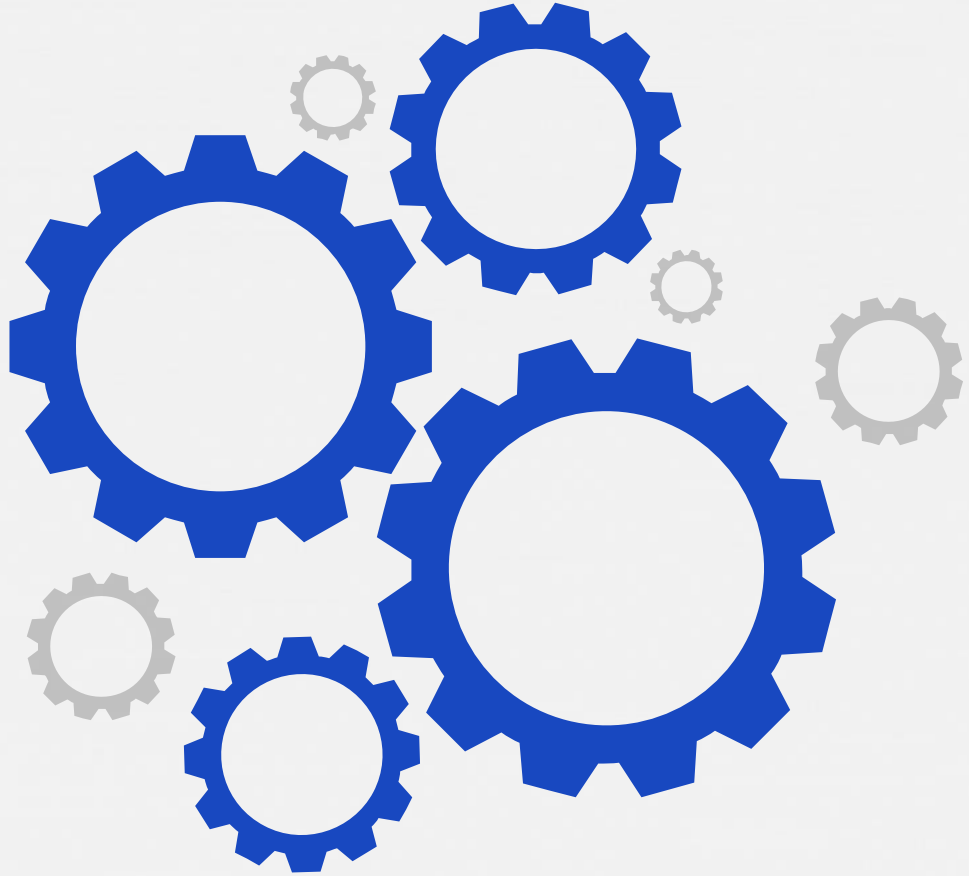
要使用的关键字包括

- 做...EndDo;
- 做直到...Enddo;
- 情况EndCase;
- 如果... Endif; 呼叫...，具有（参数）；
- 呼叫；返回....;
- 返回;
- 当

始终对循环和迭代使用作用域终止器。

# Some Keywords That Should be Used

# 一些关键字这应该被使用



As verbs, use the words

- Generate,
- Compute,
- Process, etc.

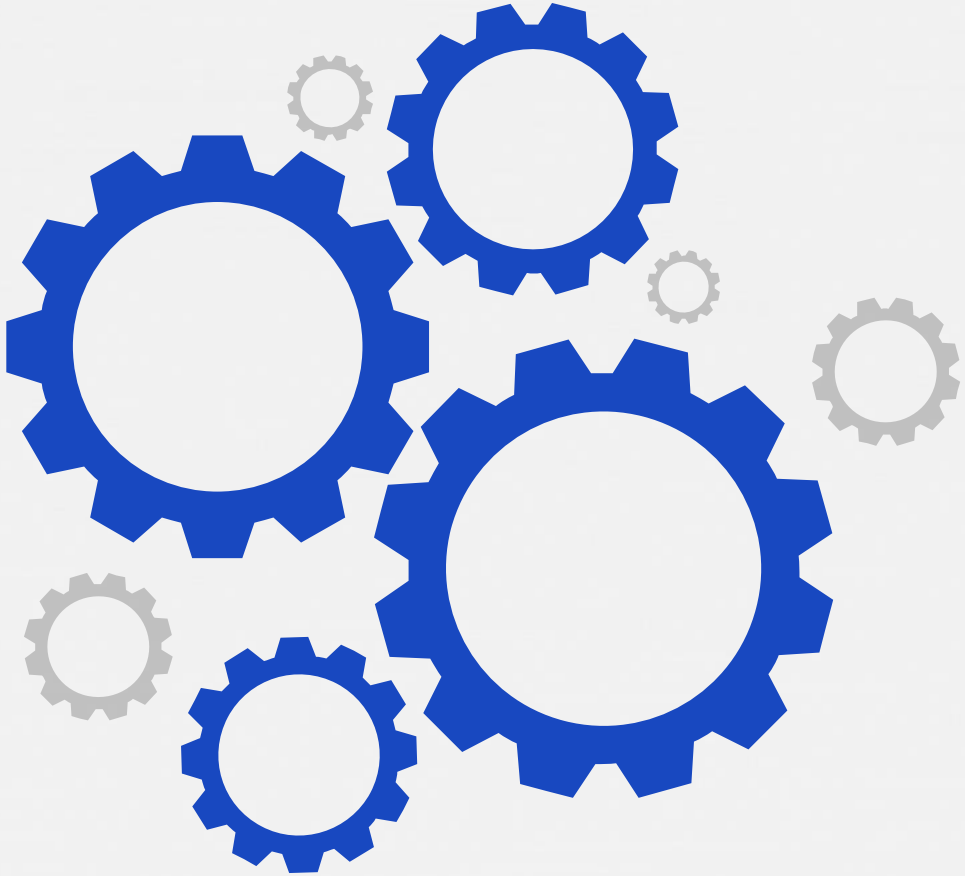
作为动词，请使用这些词

- 生成，
- 计算，
- 过程等。

- Do not include data declarations in your pseudocode.
- 不要在伪代码中包含数据声明。

# Some Keywords That Should be Used

# 一些关键字这应该被使用



Words such as

- set,
- reset,
- increment,
- compute,
- calculate,
- add,
- sum,
- multiply, ... print,
- display,
- input,
- output,
- edit,
- test , etc.

词如

- 放置
- 重置
- 增长
- 计算
- 计算
- 添加
- 总数
- 乘以，. . . 打印，
- 陈列
- 输入
- 产量
- 剪辑
- 测试等。

# 5 Pseudocode Example 5伪代码示例

- Write a Program to Print the Sum of two integer Numbers

- *Start the program*
- *Read the first number and save in the variable ( N1 )*
- *Read the second number and save in the variable ( N2 )*
- *Sum the both numbers and save the result in the variable ( Sum )*
- *Sum = N1 + N2*
- *Print the variable ( Sum )*
- *End the program*

- 编写一个程序来打印两个整数的和

- 启动程序
- 读取第一个数字并保存在变量（N1）中
- 读取第二个数字并保存在变量（N2）中
- 将这两个数字相加，并将结果保存在变量（Sum）中
- 和= N1 + N2
- 打印变量（和）
- 结束程序

# PSEUDOCODE VS FLOWCHART

## 伪代码与流程图

| PSEUDOCODE | FLOWCHART |
| --- | --- |
| An informal high-level description of the operating principle of an algorithm | A diagrammatic representation that illustrates a solution model to a given problem |
| Written in natural language and mathematical notations help to write pseudocode | Written using various symbols<br><br>Visit www.PEDIAA.com |


# 02

# Example on Pseudocode

## 伪代码示例

# Pseudocode Vs flowchart
# "Weekly Pay" Example

- Input
  - What information or data are you given?
- Process
  - What must you do with the information/data?
  - **This is your algorithm!**
- Output
  - What are your deliverables?

## "Weekly Pay" Example

- Create a program to calculate the weekly pay of an hourly employee
  - What is the input, process, and output?
- ***Input:*** *pay rate and number of hours*
- ***Process:*** *multiply pay rate by number of hours*
- ***Output:*** *weekly pay*

# 伪代码vs流程图 “周薪”示例

- 输入
  - 你给你提供了哪些信息或数据?
- 过程
  - 你必须如何处理这些信息/数据?
  - **这是你的算法!**
- 输出
  - 你的可交付成果是什么?

## “周薪”示例

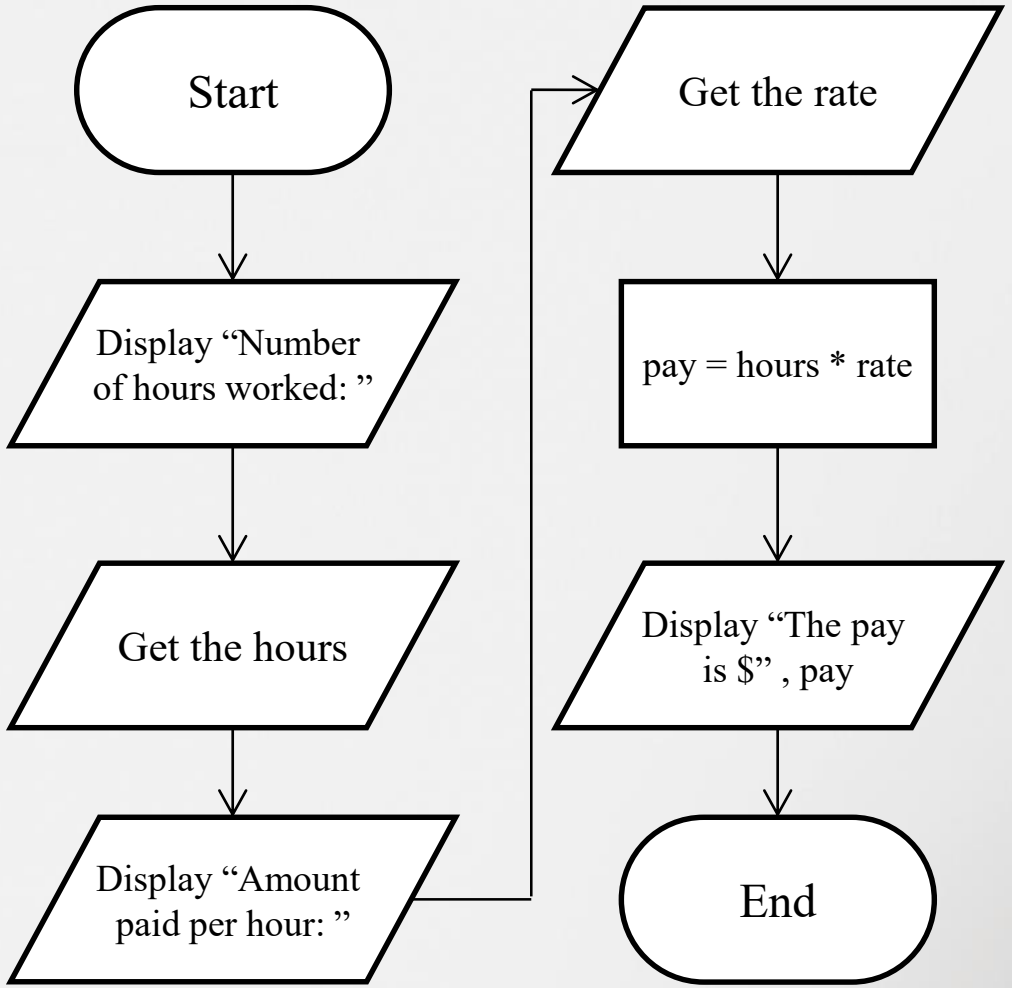
- 创建一个程序来计算一个小时工的周薪
  - 输入是什么、过程和输出?
- *输入：工资率和小时数*
- *流程：将工资率乘以小时数*
- *输出：周薪*

# Pseudocode Vs flowchart "Weekly Pay" Example

- Start with a plain English description, then…

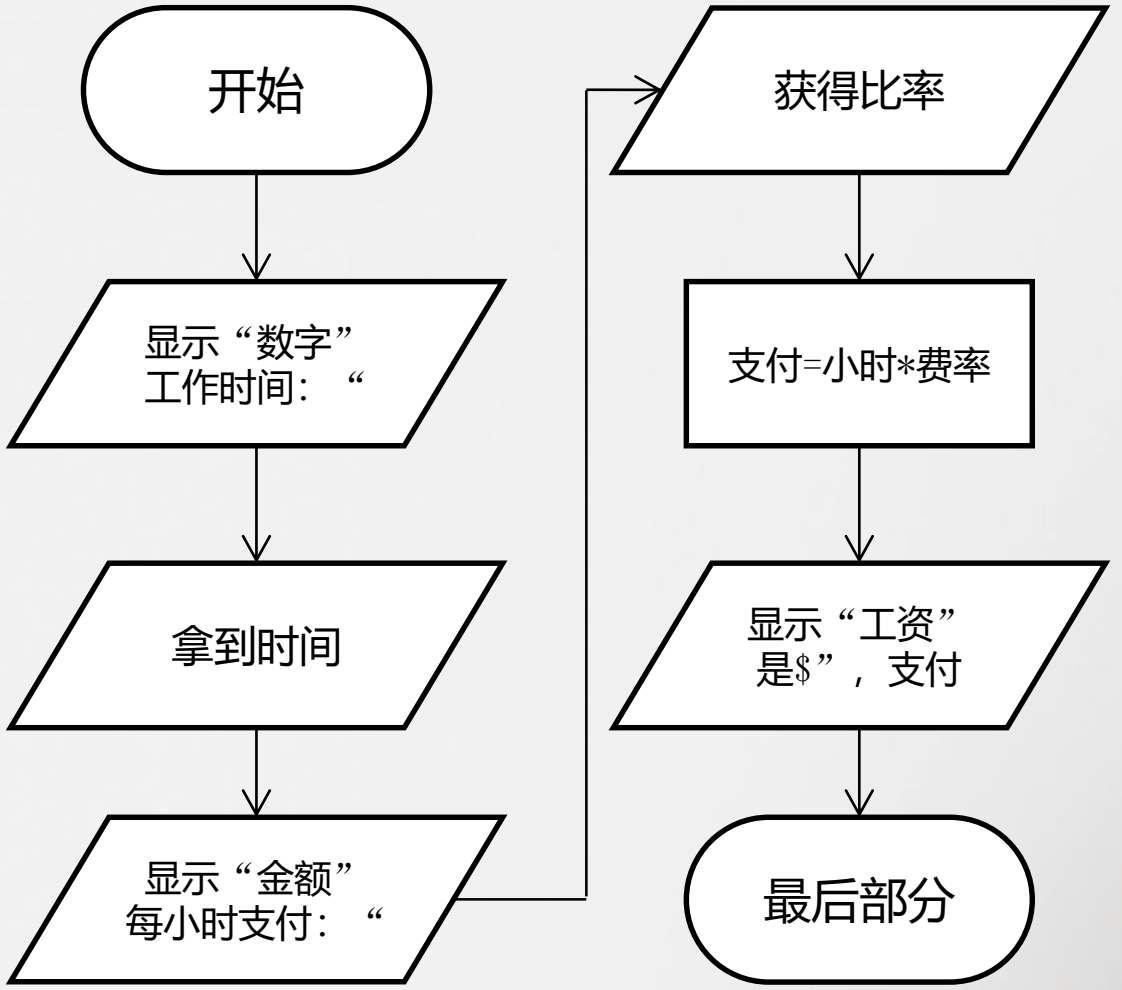
```
1. Display "Number of hours worked: "
2. Get the hours
3. Display "Amount paid per hour: "
4. Get the rate
5. Compute pay = hours * rate
6. Display "The pay is $" , pay
```

- 从一个简单的英语描述开始，然后。。
  1. 显示“工作小时数：”
  2. 拿到时间
  3. 显示“每小时支付的金额：”
  4. 获得比率
  5. 计算支付=小时*费率
  6. 显示“支付是$”，支付



# 伪代码vs流程图
# “周薪”示例

- 从一个简单的英语描述开始，然后。。
  1. 显示“工作小时数：”
  2. 拿到时间
  3. 显示“每小时支付的金额：”
  4. 获得比率
  5. 计算支付=小时*费率
  6. 显示“支付是$”，支付



# Pseudocode Vs flowchart

## How to take An Average of number

## 伪代码vs流程图

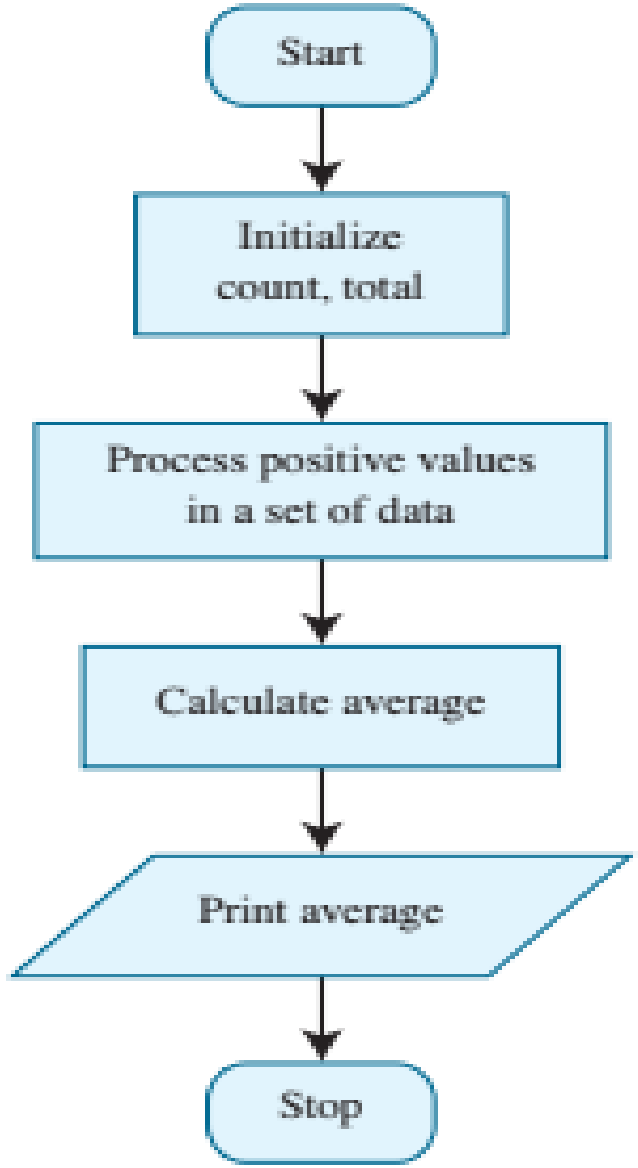
## 如何取一个数字的平均值

### Algorithm Example 算法示例

1. Input, validate, and add the data values one by one, counting the number of data values
   1. 输入、验证并逐个添加数据值，计算数据值的数量
2. Divide the sum of the data values by the number of data values
   2. 将数据值之和除以数据值的数量
3. Print the average depth
   3. 打印平均深度

### Pseudocode Example 伪代码示例

1. initialize total and count.
   1. 初始化总数和计数。
2. process positive data values in a set of data values.
   2. 处理一组数据值中的正数据值。
3. calculate average
   3. 计算平均值
4. print average
   4. 打印平均值

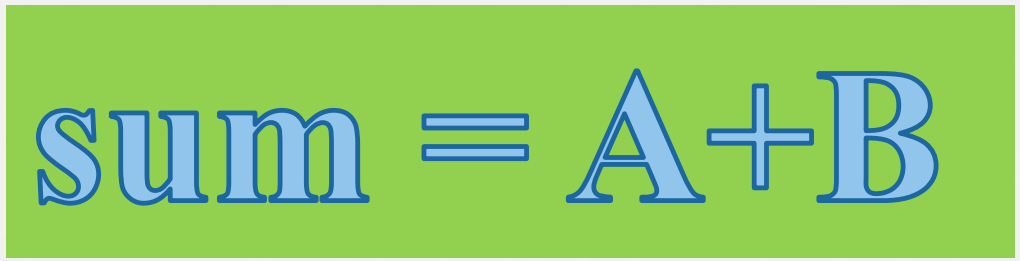


# Pseudocode Vs flowchart
# Sum Of Two Numbers: Example
# 伪代码vs流程图
# 两个数字的和：例子

- A pseudocode to find the total of two numbers is as follows.
- 查找两个数字总数的伪代码如下所示。

- SumOfTwoNumbers()
  1. Begin
  2. Set sum =0;
  3. Read: number 1, number 2;
  4. Set sum = number1 + number 2;
  5. Print sum;
- End

- TwoNumber ()
  1. 开始
  2. 设置和=0；
  3. 阅读：第1条，第2号；
  4. 设置和，=编号1，+编号2；
  5. 打印金额；
- 最后部分



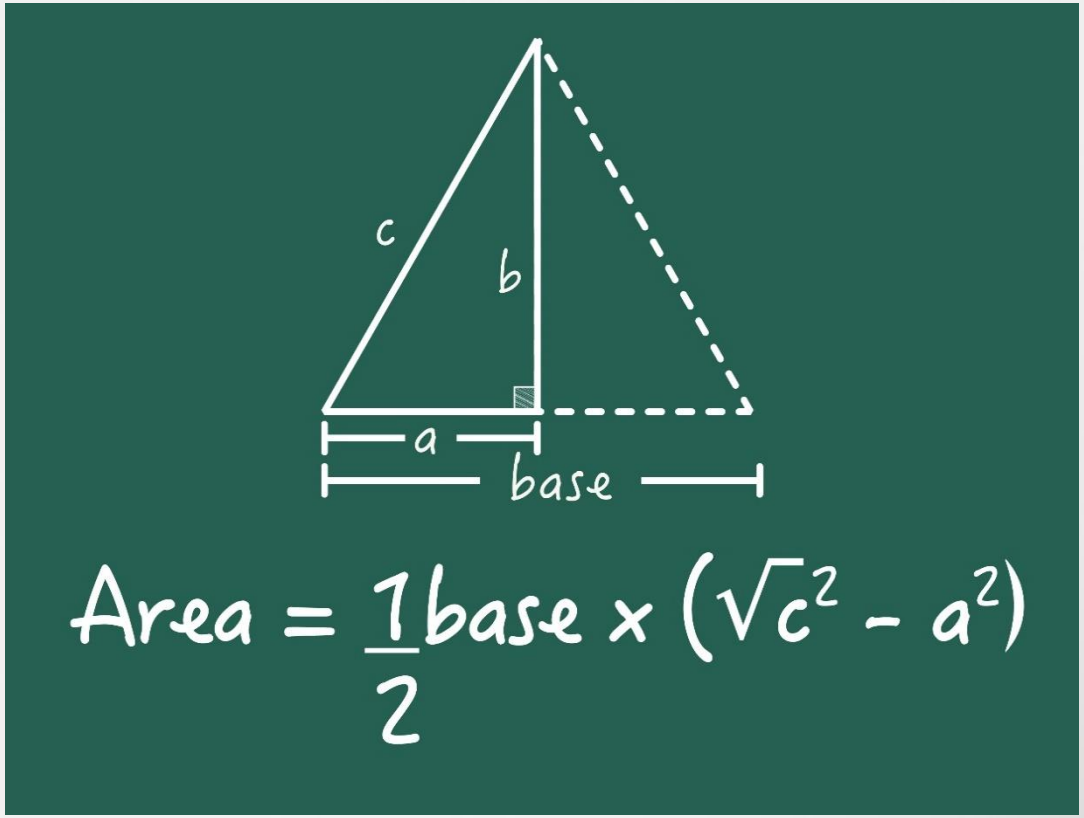
# Pseudocode Vs flowchart
# Area of Triangle : Example

# 伪代码vs流程图
# 三角形的面积：示例

- A pseudocode to find the area of a triangle is as follows.
- 寻找三角形面积的伪代码如下所示。

- AreaofTrinagle()
  1. Begin
  2. Read: base, height;
  3. Set area = 0.5 * base * height;
  4. Print area;
- End

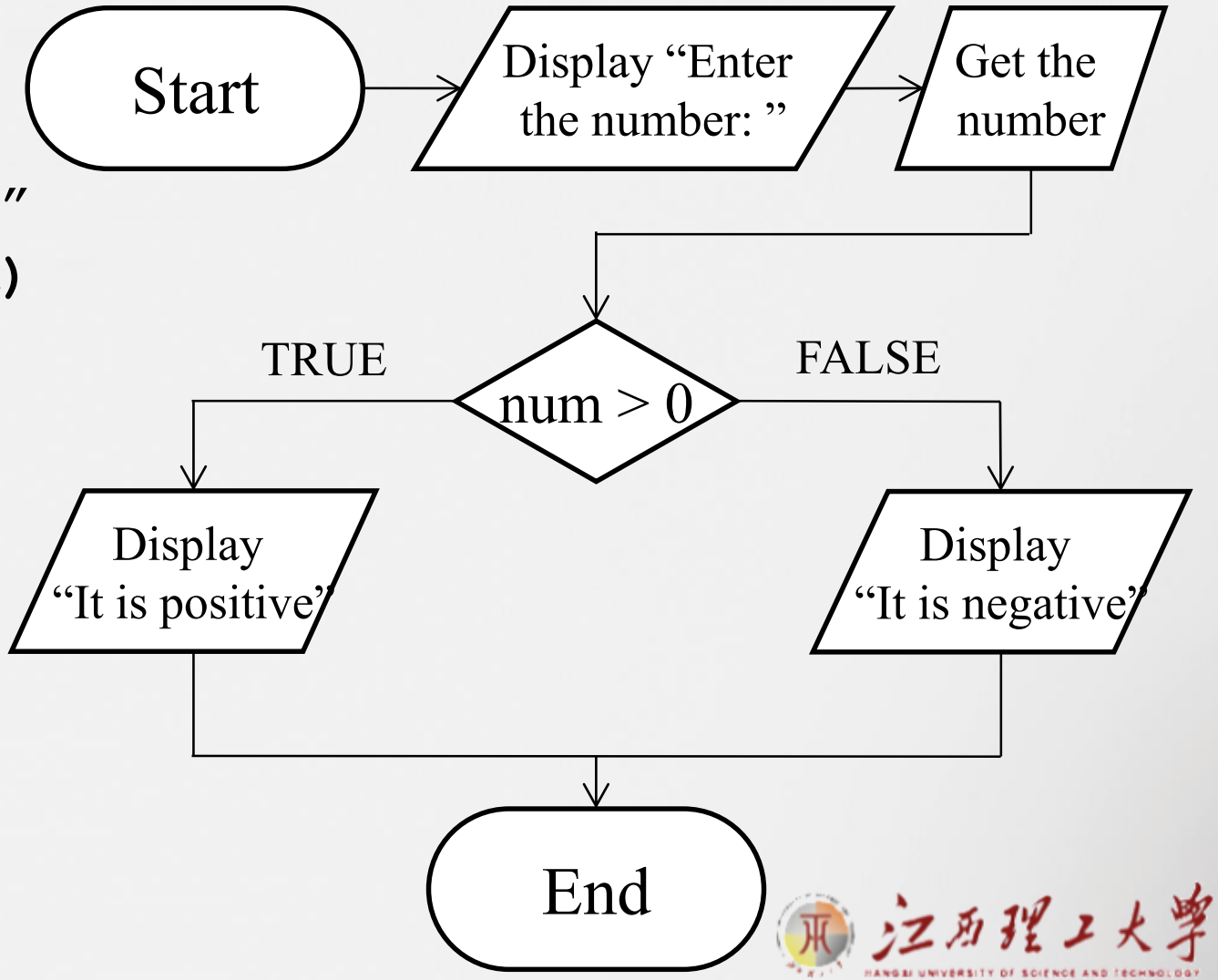
- 特里纳格尔区（）
  1. 开始
  2. 读取：底座、高度；
  3. 设置面积= 0.5 *基础*高度；
  4. 打印区域；
- 最后部分



# Pseudocode Vs flowchart
## Example: “Is a number positive?”

“Is a number positive?”

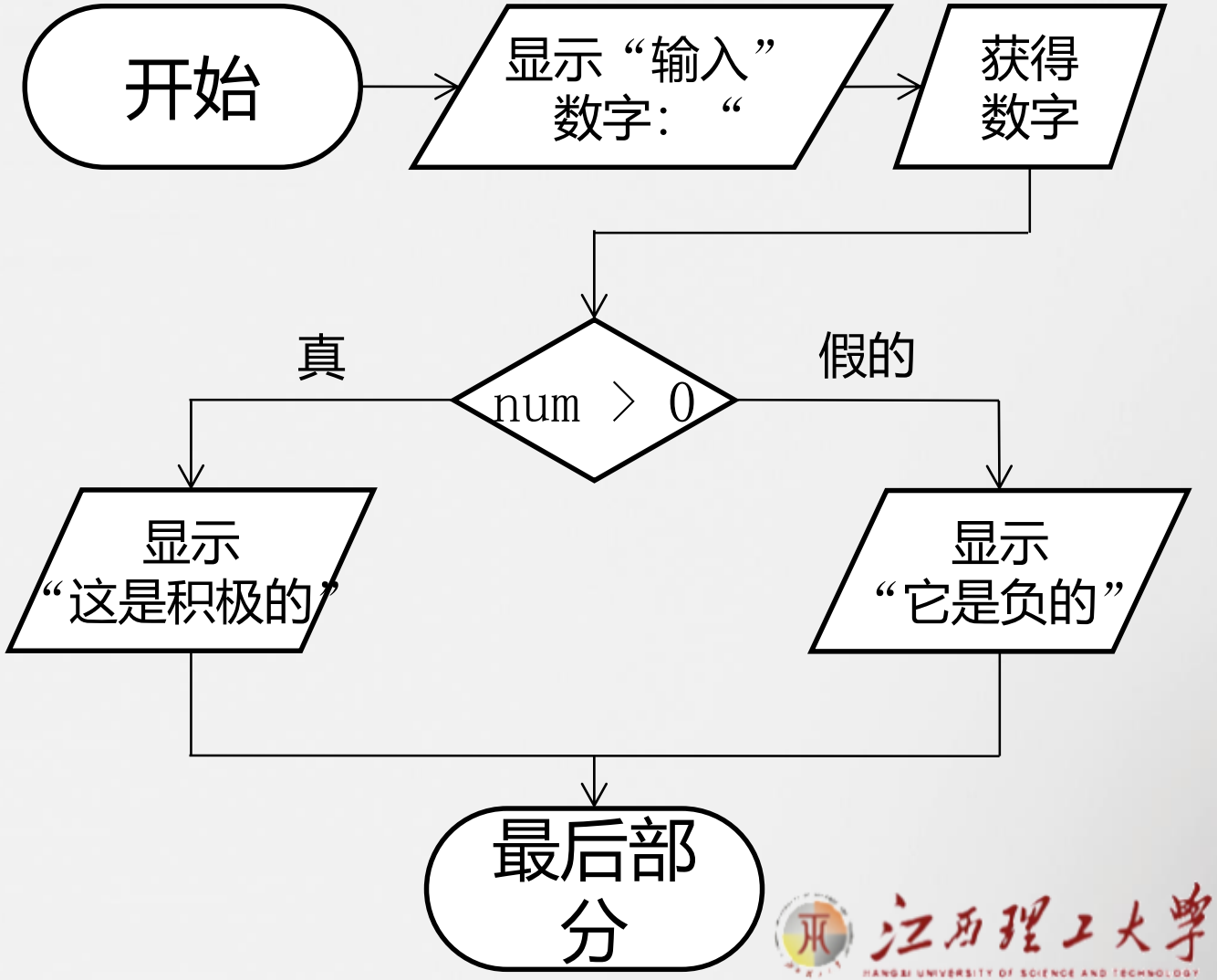
```
1. Display “Enter the number: ”
2. Get the number (call it num)
3. If num > 0
4. Display “It is positive”
5. Else
6. Display “It is negative”
```



# 伪代码vs流程图
# 例如：“一个数字是正的吗？”

“一个数字是正的吗？”

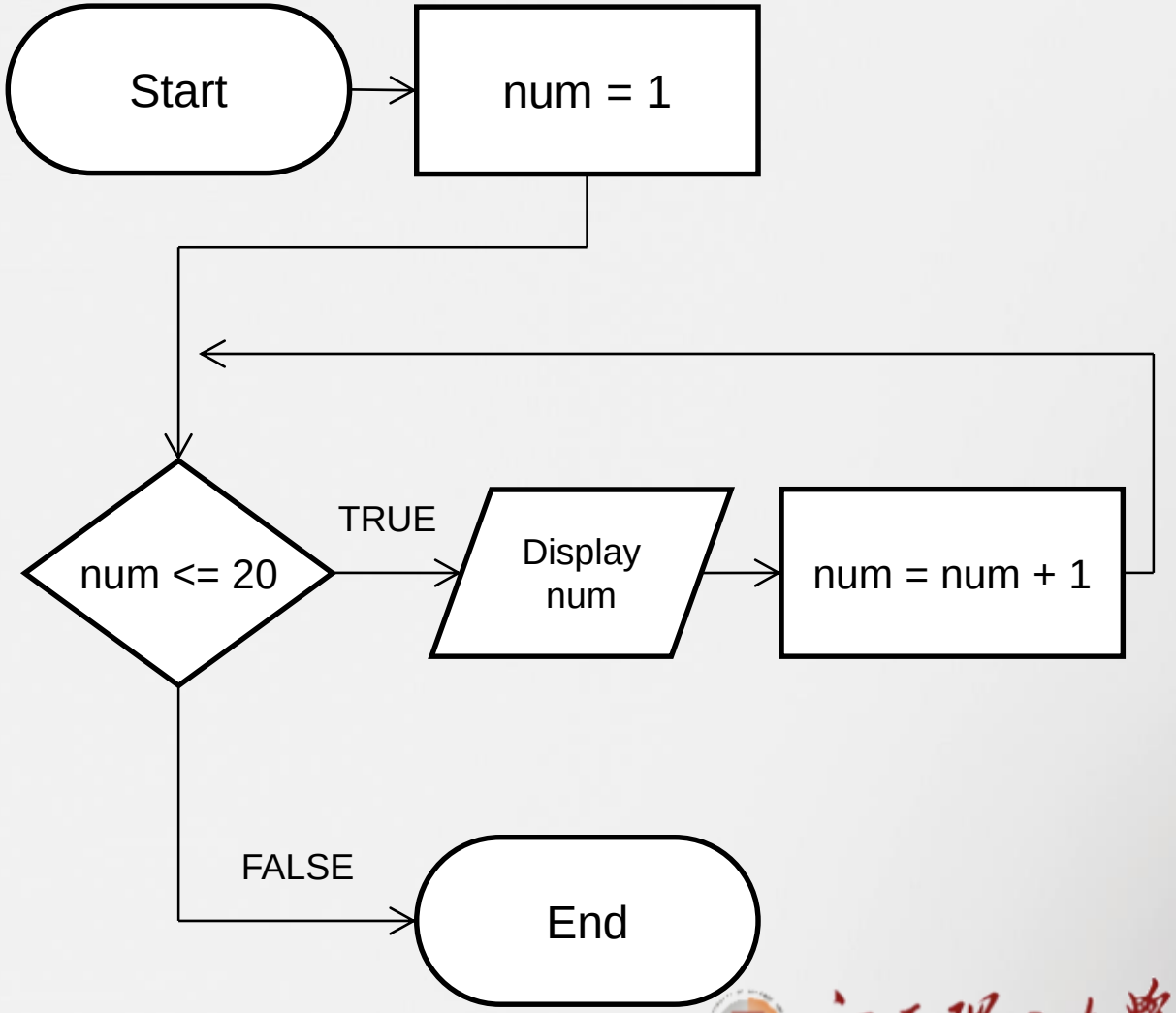
1. **显示“输入数字：”**
2. **获取该号码（称为num）**
3. **如果num > 0**
4. **显示“正”**
5. **其他的**
6. **显示“负”**



# Pseudocode Vs flowchart
# Example: “counts from 1-20”

- Write an algorithm that counts from 1-20
  - Start with a plain English description

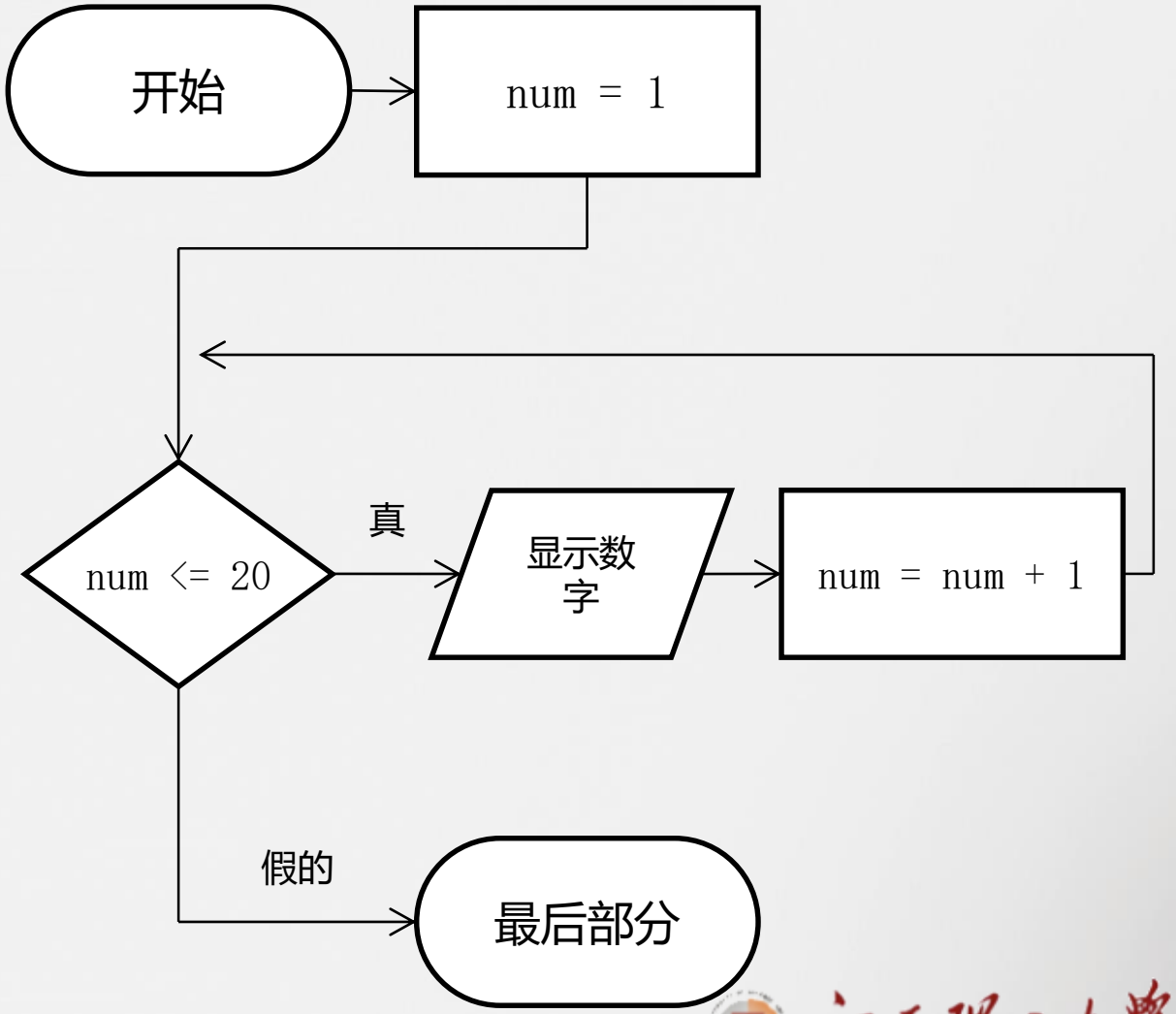
```
1. Set num = 1
2. While num <= 20
3. Display num
4. num = num + 1
5. (End loop)
```



# 伪代码vs流程图
# 例如：“1-20计数”

- 编写一个从1到20开始计算的算法
  - 从一个简单的英语描述开始

1. **设置**num = 1
2. **而**num <= 20
3. **显示数字**
4. num = num + 1
5. **(端环)**



# Student Task_1: AD

## 学生Task_1：广告

- Find one of Optimization Algorithm and write the pseudocode for them
  - 找到一个优化算法，并为其编写伪代码

| Nr. | Name | Short name | Type |
|---|---|---|---|
| 1 | Active-Set Optimization [24] | active-set | derivative-based |
| 2 | Biogeography-Based Optimization [25] | BBO | evolutionary |
| 3 | Bees Algorithm [26] | BeA | swarm-based |
| 4 | Cultural Algorithm [27] | CA | evolutionary |
| 5 | Evolution Strategy with Covariance Matrix Adaptation [28] | CMA-ES | evolutionary |
| 6 | Controlled Random Search with local mutation [29] | CRS2 | evolutionary |
| 7 | SCH Evolutionary Algorithm [30] | ESCH | evolutionary |
| 8 | Enhanced Scatter Search [31] | eSS | scatter search |
| 9 | Firefly Algorithm [32] | FA | swarm-based |
| 10 | Genetic Algorithm [33] | GA | evolutionary |
| 11 | Harmony Search [34] | HS | evolutionary |
| 12 | Interior Point Algorithm [35] | interior-point | derivative-based |
| 13 | Improved Stochastic Ranking Evolution Strategy [36] | ISRES | evolutionary |
| 14 | Levenberg-Marquardt [37] | LM | derivative-based |
| 15 | Nonlinear Optimization with Mesh Adaptive Direct Search [38] | NOMAD | direct search |
| 16 | MATLAB Particle Swarm Optimization [39] | particleswarm | swarm-based |
| 17 | Pattern Search [40] | patternsearch | direct search |
| 18 | Constrained Particle Swarm Optimization [41] | psopt | swarm-based |
| 19 | Particle Swarm Pattern Search [42] | PSwarm | swarm-based |
| 20 | Shuffled Complex Evolution [43] | SCE-UA | evolutionary |
| 21 | Constrained Simplex Method [44] | simplex | direct search |
| 22 | Simulated Annealing [45] | simulanneal | direct search |
| 23 | Sequential Quadratic Programming [46] | sqp | derivative-based |

**Send for Next lecture**

**送下一堂课**

- **Send On University MOOC system**
- **发送到大学的MOOC系统上**

江西理工大学
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Student Task_1: AD

# 学生Task_1：广告

| Swarm intelligence based algorithms | | | Bio-inspired (not SI-based) algorithms | | |
|---|---|---|---|---|---|
| Algorithm | Author | Reference | Algorithm | Author | Reference |
| Accelerated PSO | Yang et al. | [69], [71] | Atmosphere clouds model | Yan and Hao | [67] |
| Ant colony optimization | Dorigo | [15] | Biogeography-based optimization | Simon | [56] |
| Artificial bee colony | Karaboga and Basturk | [31] | Brain Storm Optimization | Shi | [55] |
| Bacterial foraging | Passino | [46] | Differential evolution | Storn and Price | [57] |
| Bacterial-GA Foraging | Chen et al. | [6] | Dolphin echolocation | Kaveh and Farhoudi | [33] |
| Bat algorithm | Yang | [78] | Japanese tree frogs calling | Hernández and Blum | [28] |
| Bee colony optimization | Teodorović and Dell'Orco | [62] | Eco-inspired evolutionary algorithm | Parpinelli and Lopes | [45] |
| Bee system | Lucic and Teodorovic | [40] | Egyptian Vulture | Sur et al. | [59] |
| BeeHive | Wedde et al. | [65] | Fish-school Search | Lima et al. | [14], [3] |
| Wolf search | Tang et al. | [61] | Flower pollination algorithm | Yang | [72], [76] |
| Bees algorithms | Pham et al. | [47] | Gene expression | Ferreira | [19] |
| Bees swarm optimization | Drias et al. | [16] | Great salmon run | Mozaffari | [43] |
| Bumblebees | Comellas and Martinez | [12] | Group search optimizer | He et al. | [26] |
| Cat swarm | Chu et al. | [7] | Human-Inspired Algorithm | Zhang et al. | [80] |
| Consultant-guided search | Iordache | [29] | Invasive weed optimization | Mehrabian and Lucas | [42] |
| Cuckoo search | Yang and Deb | [74] | Marriage in honey bees | Abbass | [1] |
| Eagle strategy | Yang and Deb | [75] | OptBees | Maia et al. | [41] |
| Fast bacterial swarming algorithm | Chu et al. | [8] | Paddy Field Algorithm | Premaratne et al. | [48] |
| Firefly algorithm | Yang | [70] | Roach infestation algorithm | Havens | [25] |
| Fish swarm/school | Li et al. | [39] | Queen-bee evolution | Jung | [30] |
| Good lattice swarm optimization | Su et al. | [58] | Shuffled frog leaping algorithm | Eusuff and Lansey | [18] |
| Glowworm swarm optimization | Krishnanand and Ghose | [37], [38] | Termite colony optimization | Hedayatzadeh et al. | [27] |
| Hierarchical swarm model | Chen et al. | [5] | Physics and Chemistry based algorithms | | |
| Krill Herd | Gandomi and Alavi | [22] | Big bang-big Crunch | Zandi et al. | [79] |
| Monkey search | Mucherino and Seref | [44] | Black hole | Hatamlou | [24] |
| Particle swarm algorithm | Kennedy and Eberhart | [35] | Central force optimization | Formato | [21] |
| Virtual ant algorithm | Yang | [77] | Charged system search | Kaveh and Talatahari | [34] |
| Virtual bees | Yang | [68] | Electro-magnetism optimization | Cuevas et al. | [13] |
| Weightless Swarm Algorithm | Ting et al. | [63] | Galaxy-based search algorithm | Shah-Hosseini | [53] |
| Other algorithms | | | Gravitational search | Rashedi et al. | [50] |
| Anarchic society optimization | Shayeghi and Dadashpour | [54] | Harmony search | Geem et al. | [23] |
| Artificial cooperative search | Civicioglu | [9] | Intelligent water drop | Shah-Hosseini | [52] |
| Backtracking optimization search | Civicioglu | [11] | River formation dynamics | Rabanal et al. | [49] |
| Differential search algorithm | Civicioglu | [10] | Self-propelled particles | Vicsek | [64] |
| Grammatical evolution | Ryan et al. | [51] | Simulated annealing | Kirkpatrick et al. | [36] |
| Imperialist competitive algorithm | Atashpaz-Gargari and Lucas | [2] | Stochastic difusion search | Bishop | [4] |
| League championship algorithm | Kashan | [32] | Spiral optimization | Tamura and Yasuda | [60] |
| Social emotional optimization | Xu et al. | [66] | Water cycle algorithm | Eskandar et al. | [17] |

Table 1. A list of algorithms

# Reference

- https://afteracademy.com/blog/time-and-space-complexity-analysis-of-algorithm

江西理工大学
Jiangxi University of Science and Technology
信息工程学院
School of information engineering

# 高级算法分析与设计

# Advanced Algorithm Analysis and Design


THANK YOU


"The beauty of research is that you never know where it's going to lead."

RICHARD ROBERTS
Nobel Prize in Physiology or Medicine 1993

"BE HUMBLE. BE HUNGRY.
AND ALWAYS BE THE
HARDEST WORKER
IN THE ROOM."