華南理工大學

South China University of Technology

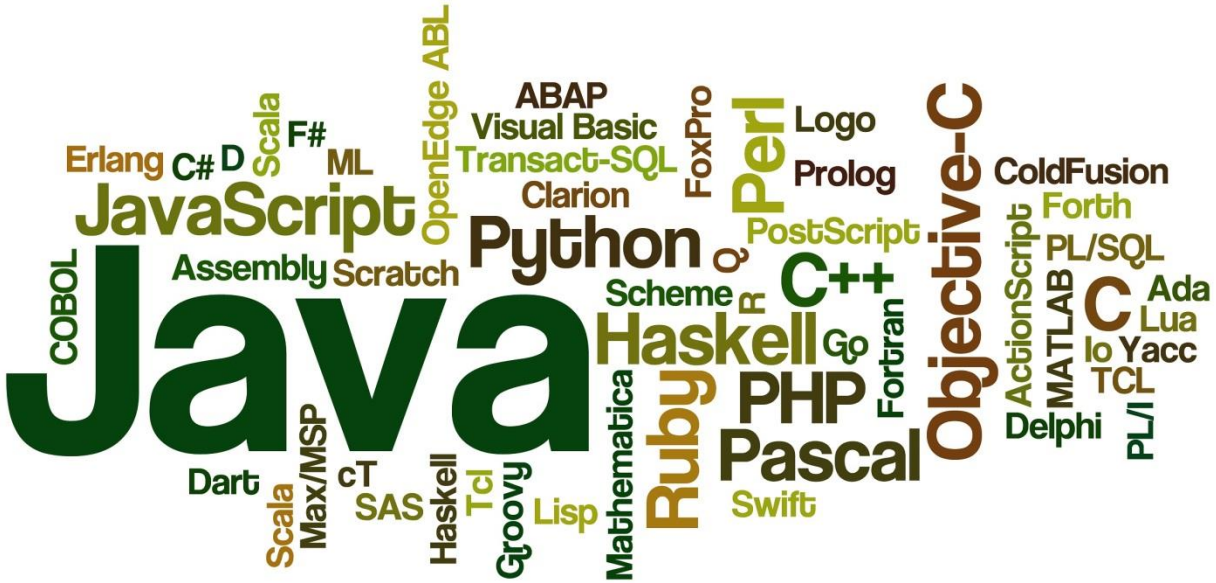
Intro to Computer Science and Software Engineering

Programming Languages

Dr Yubei Lin
yupilin@scut.edu.cn
School of Software Engineering

Computer Language

- A computer language is a set of predefined words that are combined into a program according to predefined rules (syntax).

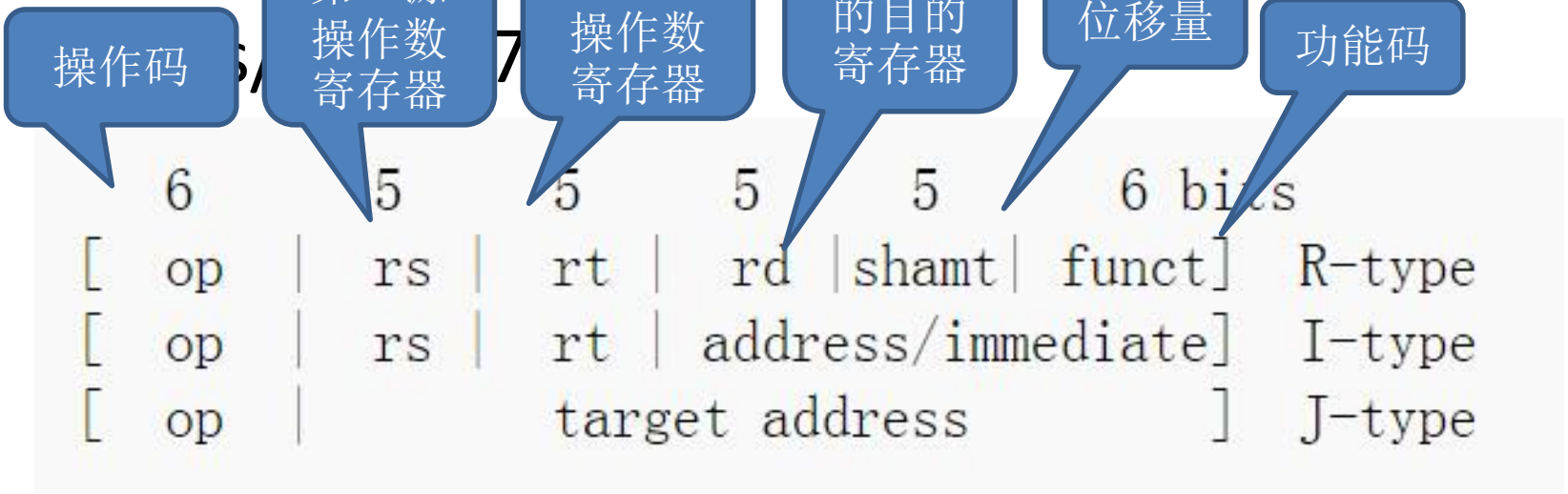


Machine Languages

- A set of instructions executed directly by a computer's CPU
  - lowest-level
  - only language understood by a computer directly
- Each kind of computer has its own machine language
  - Specific to CPU
- The instructions must be in streams of 0s and 1s, in case of electronic computers

Machine Languages

- The MIPS instruction set example @https://blog.csdn.net/qq_42146369/article/




```
[  op  |  rs  |  rt  |  rd  |shamt| funct]
   0      1      2      6      0     32      decimal
 000000 00001 00010 00110 00000 100000      binary
```

Symbolic Languages

- Mirrored the machine languages using symbols or mnemonics
  - By Grace Hopper
  - e.g. ADD for addition of two nums
- **Assembler**
  - A special program translate symbolic code into machine language.
- Also known as assembly languages

Assembly languages

- A low-level programming language for a computer, or other programmable device, in which there is a very strong (generally **one-to-one**) correspondence between the language and the architecture's machine code instructions.

Assembly languages

```
; Example of IBM PC assembly language
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32   PROC            ; procedure begins here
        CMP   AX,97     ; compare AX to 97
        JL    DONE      ; if less, jump to DONE
        CMP   AX,122    ; compare AX to 122
        JG    DONE      ; if greater, jump to DONE
        SUB   AX,32     ; subtract 32 from AX
DONE:   RET             ; return to main program
SUB32   ENDP            ; procedure ends here
```

FIGURE 17. Assembly language

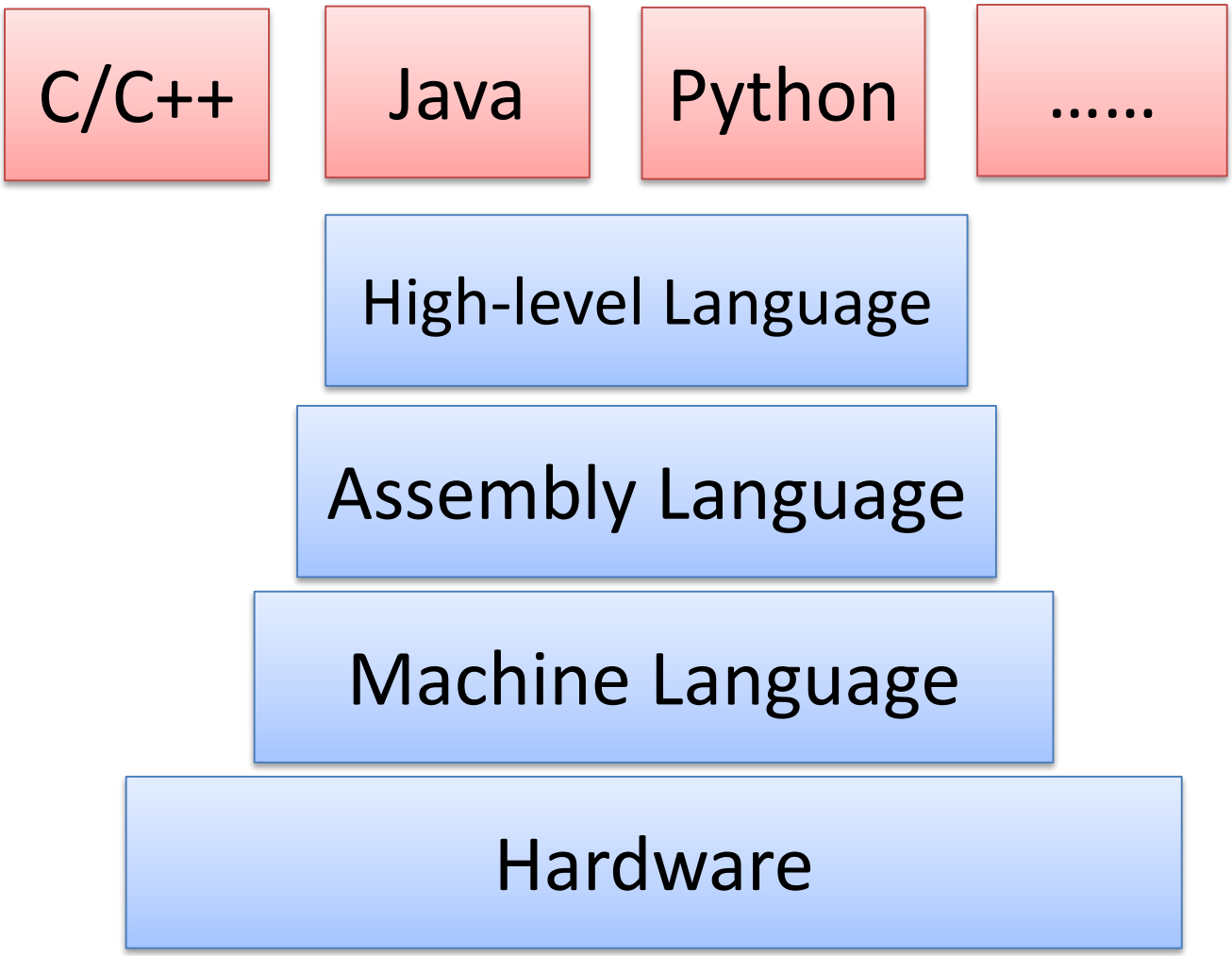
High-level languages

- Strong abstraction from the details of computers, hence portable to many different computers
- Allow the programmer to concentrate on the application (i.e. the problem) rather than the intricacies of the computer.
- Mostly English-like, but exists Chinese-like
- Compilation (编译)
  - The process that converting high-level codes to machine codes

High-level languages

- Abstraction penalty [Wiki]
  - While high-level languages are intended to make complex programming simpler, low-level languages often produce more efficient code.
  - High-level programming exhibits features like more generic data structures, run-time interpretation, and intermediate code files; which often result in slower execution speed, higher memory consumption, and larger binary program size.

The layer of languages



Natural Languages

- Just use your natural languages (e.g. English or Chinese) to write the codes, and the computer would understand it and execute your request immediately.

- Some references:
  - @http://www.cs.cmu.edu/~NatProg/
  - Mathematica @http://www.wolfram.com/mathematica/ @http://blog.wolfram.com/2010/11/16/programming-with-natural-language-is-actually-going-to-work/

Building a Program

- To have an executable (machine language) file
  - Writing and editing the program
    - Editor/IDE: Source files
  - Compiling the program
    - Compiler: preprocessor (预处理~) and translator (转换~)
    - Result: Object module
  - Linking the program with the required library modules
    - Common libraries, e.g. I/O
    - Result: executable file

Building a Program

- To have an executable (machine language) file
  - Writing and editing the program
    - Editor/IDE: Source files

```
1     #include<stdio.h>
2
3
4   int main (void)
5
6   {
7
8         printf ("hello\n") ;
9
10        return 0;
11
12  }
```

Building a Program

- To have an executable (machine language) file
  - Compiling the program
    - Compiler: preprocessor (预处理~) and translator (转换~)
    - Result: Object module

- **预编译过程**

这个过程处理宏定义和include，去除注释，不会对语法进行检查。

可以看到预编译后，代码从6行扩展到了910行。

```
gcc -E a.c -o a.i
cat a.c|wc -l
5
cat a.i|wc -l
910
```

- **编译过程**

这个阶段，检查语法，生成汇编代码。

```
gcc -S a.i -o a.s
cat a.s|wc-l
59
```

- **汇编过程**

这个阶段，生成目标代码。

此过程生成ELF格式的目标代码。

```
gcc -c a.s -o a.o
file a.o
a.o:ELF64-bitLSBrelocatable,AMDx86-64,version1 (SYSV) ,notstripped
```

Building a Program

- To have an executable (machine language) file
  - Linking the program with the required library modules
    - Common libraries, e.g. I/O
    - Result: executable file
    - 静态链接：依赖的动态链接库较少，对动态链接库的版本不会很敏感，具有较好的兼容性
    - 动态链接：生成的程序比较小，占用较少的内存。

```
1 | gcc a.o -o a
```

程序运行：

```
1 | ./a
2 | hello
```

所有图片均来自网络

Building a Program

- The Compiling, Linking and Building C/C++ Applications, the GCC (GNU Compiler Collection) example
- @https://www.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html

- Please have a try!

Program execution

- Selecting a executable file by the ‘run’ command of the operating system upon the file
  - Become a job
- A loader program of the OS would locate the executable file and load it into memory when everything is ready.
  - Become a process in the ready queue

Categories of languages

- Categorized according to the approach they use in solving a problem and the category of problem they solve.
  - Procedural (imperative)
  - Object-Oriented
  - Functional
  - Logic (declarative)
  - Special
- Imperative(命令式) vs Declarative(声明式)

Imperative languages (命令式)

- A programming paradigm that uses statements that change a program's state.
- An imperative program consists of **commands** for the computer to perform.
- Imperative programming focuses on describing **how** a program operates.
  - The algorithm

Declarative Languages (声明式)

- A programming paradigm of building the structure and elements of computer programs, that expresses the logic of a computation without describing its control flow.
- Declarative programming focuses on **what** the program should accomplish without specifying **how** the program should achieve the result.

Imperative vs Declarative

```
var numbers = [1,2,3,4,5]
var doubled = []

for(var i = 0; i < numbers.length; i++) {
  var newNumber = numbers[i] * 2
  doubled.push(newNumber)
}
console.log(doubled) //=> [2,4,6,8,10]
```

```
var numbers = [1,2,3,4,5]

var doubled = numbers.map(function(n) {
  return n * 2
})
console.log(doubled) //=> [2,4,6,8,10]
```

@http://latentflip.com/imperative-vs-declarative/

Procedural Languages

- Imperative: must specify the set of instructions (of an algorithm)
- The concept of **Procedure call:** contain a series of computational steps to be carried out.
- Any given procedure might be called at any point during a program's execution, including by other procedures or itself.
- Example: Fortran, Cobol, Pascal, C, … …

Object-oriented programming

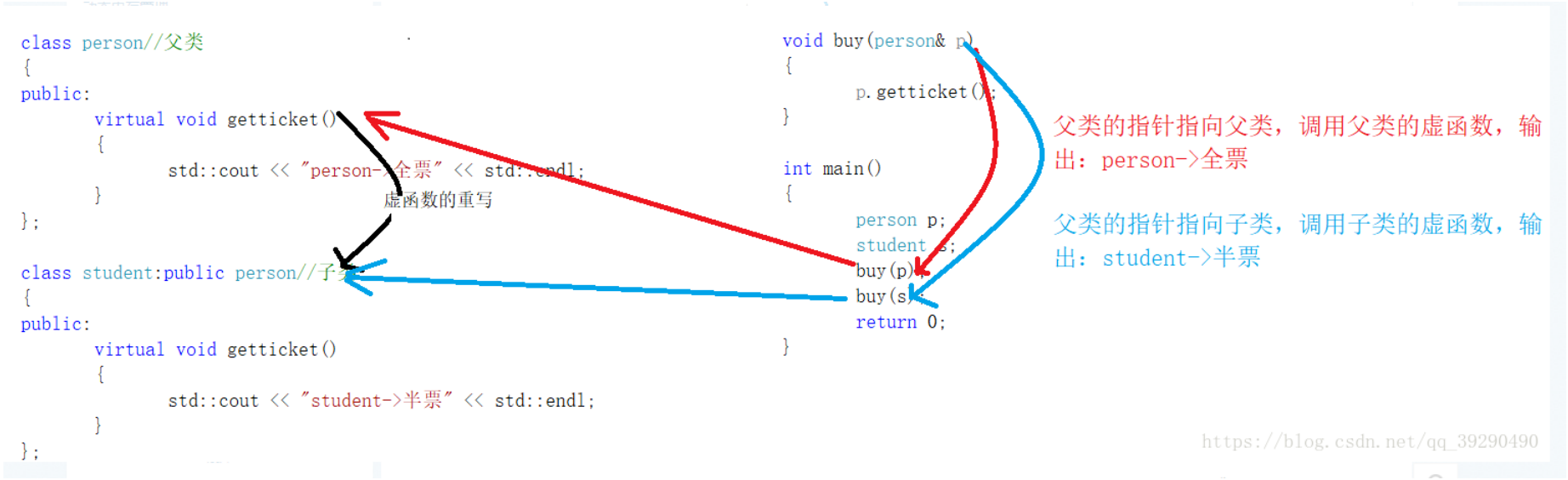
- Imperative
- The concept of **objects**:
  - data structures that contain **data**, in the form of **fields**, often known as ***attributes;***
  - and code, in the form of **procedures**, often known as ***methods***.
- A method call is also known as **message passing**. It is conceptualized as a **message** (the name of the method and its input parameters) being passed to the object for dispatch.

Object-oriented programming

- Encapsulation (封装)
  - Hiding the data, and accessing the data only through interfaces/methods
- Composition (组合)
  - Objects can contain other objects in their instance variables
- Inheritance (继承)
  - allow classes to be arranged in a hierarchy that represents "is-a-type-of" relationships.

Object-oriented programming

- Polymorphism(多态)
  - is when calling code can be agnostic as to whether an object belongs to a parent class or one of its descendants.



Procedural vs Object-oriented

Procedural	Object-oriented
Procedure	method
record	object
module	class
Procedure call	message

Functional languages

- Declarative
- A programming paradigm that treats computation as the **evaluation of mathematical functions** and avoids changing-state and mutable data.
  - Predefines a set of primitive (atomic) functions
  - Combine primitive functions to create new functions
  - 强调函数的计算比指令的执行重要，函数的计算可随时调用

Functional languages

- Primitive functions
  - FIST: extract the first element of a list
  - REST: extract all the elements except the first
- Program: extract the third element of a list
  - FIST(REST(REST(1,2,3,4,5,6)))
- Example: LISP, Scheme

Logic languages

- Declarative
- 基于一组已知规则的形式逻辑系统
- Example:Prolog

Special languages

- HTML
- XML
- SQL
- … …

所有图片均来自网络