

# Programming with C++

## COMP2011: Introduction

Cecia Chan  
Brian Mak  
Dimitris Papadopoulos  
Pedro Sander  
Charles Zhang

Department of Computer Science & Engineering  
The Hong Kong University of Science and Technology  
Hong Kong SAR, China

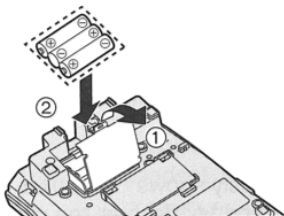


# Course Objectives

- To learn how to **solve problems** by writing **computer programs**.
- To learn how to **design** a computer program.
- To learn how to program in **C++**.
- To learn how to **debug** a computer program.
- To learn **object-oriented programming**.
- To prepare you for COMP2012 (OOP & Data Structures), etc.

Question: *computer science = programming?*

## Installing the Batteries

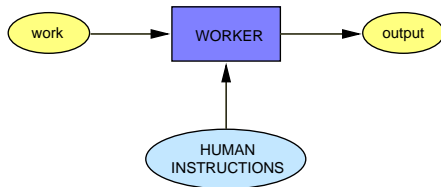


- 1 Press down in the direction of the arrow and open the cover (①).
- 2 Install the batteries in the proper order as shown (②), matching the correct polarity.
- 3 Close the battery cover.

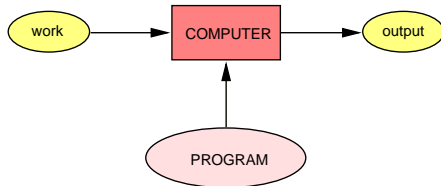
- Batteries are not included in the unit.
- Install three high quality “AA” size Alkaline (LR6) or Manganese (R6, UM-3) batteries. We recommend to use Alkaline batteries.  
Battery life is: —about six months in use of Alkaline batteries.  
                    —about three months in use of Manganese batteries.  
Battery life may depend on usage conditions and ambient temperature.

# What's a Computer Program? ..

Human work model

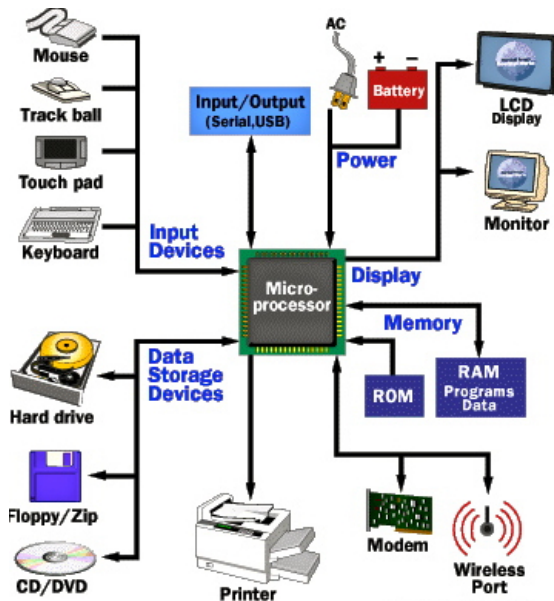


Computer work model

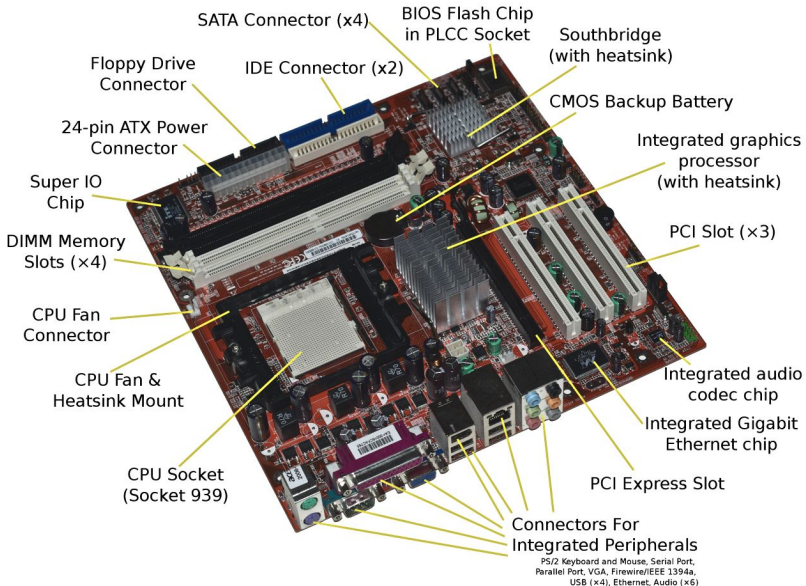


- A **computer program** is a set of **machine-readable instructions** that tells a computer how to perform a specific task. (During the execution of the program, it may interact with the users and its environment.)

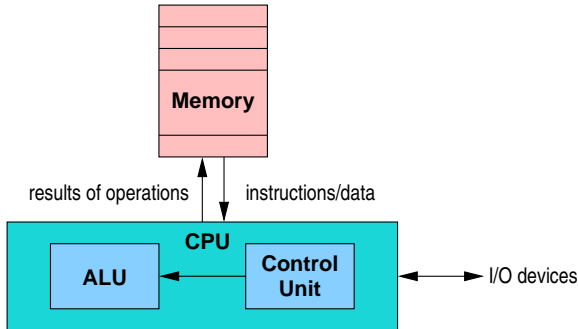
# Schematic Diagram of a Personal Computer



# A Typical Motherboard



# von Neumann Computer Architecture



- Designed by **John von Neumann**, a mathematician, in 1945.
- It is still today's dominant computer architecture.
- **CPU** = Central Processing Unit
- **ALU** = Arithmetic Logic Unit.
- For **efficiency**, many programming languages, including C++, are designed to take advantage of the architecture.
- More on this in COMP2611 (Computer Organization).

# Can You Understand This?

0000100100101110011001100110100101101100011001010000100100100010011011000110010101100011011101000111  
010101110010011001010011000100101110011000110010001000001010011001110110001101100011001100100101111  
0110001101101111011011010111000001101001011011000110010101100100001011100011101000001010001011100111  
0011011001010110001101110100011010010110111101101110000010010010001000101110011101000110010101111000  
01110100001000100000101000001001001011100110000101101000110100101100111010111000100000001101000000  
1010000010010010111001101101101000110111011000100110000010110100001000000110110101100000101101001  
011011100000101000001001001011100111010001110010111000001100101000010010010000001101101011000010110  
1001011011100010110000100011011001100111010101101110011000110111010001101001011011110110111000001010  
0000100100101110011100000111001001101111011000110000100100110000001101000000101001101101011000010110  
100101101110001110100000101000001001001000010010001101010000101001001001111010011000100111101000111  
0101010101000101001000110010000000110000000010100000100101110011011000010111011001100101001000000010  
0101011100110111000000101100001011010011000100110010001110000010110000100101011100110111000000001010  
00001010000010010010000100100011010100000101001001001111010011000100111101000111010101010001010010  
00110010000000110001000010100000100101101101011011101110110001000000011000100101100001001010110111  
001100000000101000001001011100110111010000100000001001010110111001100000010110001011011001001010110  
0110011100000010110011001000110000010111010000101000001001011010110111011101100010000000110010  
00101100001001010110111001100000000101000010010110011011010000100000001001010110111001100000010  
110001011011001001010110011001110000001011010011001000110100001010000010010110110001100100  
001000000101101100100101011001100111000000101101001100100011000001011101001011000010010101110011  
00000000101000001001011011000110010000100000010110110010010101100110000001011010011001000110100  
010111010010110000100101101110011000100001010000010010110000110010001100100001000000001001010110  
111100110000001011000010010101110011000100010110000100101011011100110000000010100000100101110011  
011101000001000000010010101101110011000000101100010110110010010101100110011100000010110100110010001  
100001011101000010100000100101101101011011110111011000100000011000000101100001001010110100100110000  
000010100000100101100010001000000010111001001100010011000011000100001010000010010110111001101110111  
0000000010100010111001001100010011000011000100111010000010100000100101110010011001010111010000001010  
00001001011100100110010111000101110100011011101110010011001010000101000010110010011000100011000110  
01100110010100110001001110100000101000001001001100011001100101001011101001011010010100001000000  
011011010110000101101001011011100010110000101110010011000100110001100110010100110001001011010110  
1101011000010110100101101110000010100000100100101110011010010110010001100101011011100111010000001001  
0010001001000111010000110100001100111010001000000010100001000111010011100101010100101001001000000011  
0010001011100011100000101110001100010010001000001010



# How About This?

main:

```
!#PROLOGUE# 0  
save %sp,-128,%sp
```

```
!#PROLOGUE# 1  
mov 1,%o0  
st %o0,[%fp-20]  
mov 2,%o0  
st %o0,[%fp-24]  
ld [%fp-20],%o0  
ld [%fp-24],%o1  
add %o0,%o1,%o0  
st %o0,[%fp-28]  
mov 0,%i0  
nop
```

# Is This Better Now?

```
int main( )  
{  
    int x, y, z;  
  
    x = 1;  
    y = 2;  
    z = x+y;  
  
    return 0;  
}
```

## Example: Write a Program to Sum 2 Numbers

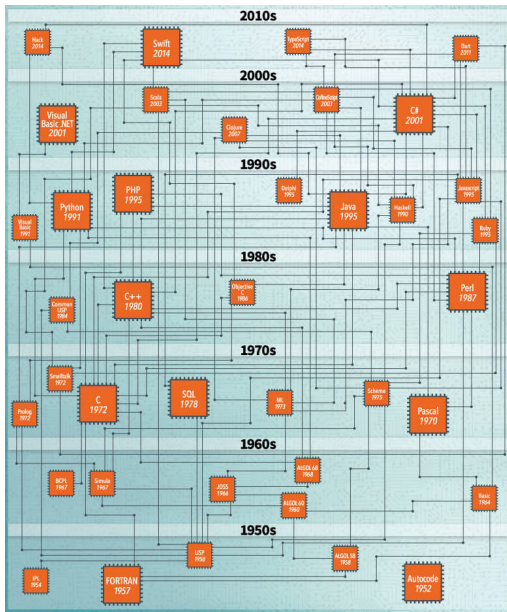
- There are 3 integer-value-holding objects: x, y, and z.
- x and y have the value of 1 and 2 respectively.
- z's value is the sum of x's and y's.

```
int main( )  
{  
    int x, y, z;  
  
    x = 1;  
    y = 2;  
    z = x+y;  
  
    return 0;  
}
```

# Levels of Programming Languages

- machine (binary) language is unintelligible
- assembly language is low level
  - mnemonic names for machine operations
  - explicit manipulation of memory addresses/contents
  - machine-dependent
- high level language
  - readable
    - instructions are easy to remember
    - faster coding
    - less error-prone (fewer bugs?)
    - easier to maintain
  - no mention of memory locations
  - machine-independent = portable

# Chronology of Some Programming Languages



More details in  
this  
computer history  
website, and  
the following  
infographics.

# PYPL PopularitY: Most Popular Programming Languages

Worldwide, Aug 2022 compared to a year ago:

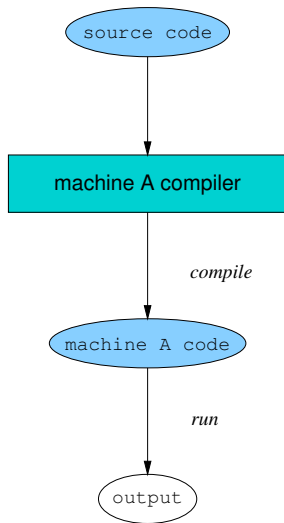
Rank	Change	Language	Share	Trend
1		Python	28.11 %	-2.6 %
2		Java	17.35 %	-0.9 %
3		JavaScript	9.48 %	+0.2 %
4		C#	7.08 %	+0.1 %
5		C/C++	6.19 %	-0.3 %
6		PHP	5.47 %	-0.8 %
7		R	4.35 %	+0.6 %
8	↑↑	TypeScript	2.79 %	+1.1 %
9	↑↑	Swift	2.09 %	+0.5 %
10	↓↓	Objective-C	2.03 %	+0.2 %

It is based on the number of Google searches on the languages' tutorials.

# Mostly Used Programming Languages in Github

Year		Quarter	
2022		1	
# Ranking	Programming Language	Percentage (YoY Change)	YoY Trend
1	Python	16.689% (+0.061%)	^
2	JavaScript	14.270% (-4.486%)	v
3	Java	13.075% (+1.394%)	
4	TypeScript	9.105% (+2.501%)	^
5	Go	7.885% (+0.056%)	v
6	C++	6.950% (-0.035%)	
7	Ruby	6.179% (-1.408%)	v
8	PHP	5.260% (+0.179%)	
9	C#	3.059% (-0.555%)	
10	C	3.023% (-0.230%)	

# Compilation: From Source to Runnable Program



A **compiler** translates **source programs** into **machine codes** that run directly on the target computer.

For example,  
`a.cpp`  $\longrightarrow$  `a.out` (or `a.exe`).

Some C++ compilers:  
`gcc/g++`, `VC++`.

- static codes
- compile once, run many
- optimized codes  
 $\Rightarrow$  more efficient
- examples: FORTRAN, Pascal, C++



# Programming as Problem Solving

- **Understand** and **define** the problem clearly.
  - What are the input(s) and output(s)?
  - Any constraints?
  - Which information is essential?
- **Develop** a solution.
  - Construct an algorithm.
- **Translate** the algorithm into a C++ program.
- **Compile** the program.
- **Test** the program.
- **Debug** the program.
- **Document** the program as you write the program.
- **Maintain** the program
  - modify the codes when conditions change.
  - enhance the codes to improve the solution.

- Why C++?

Read the FAQ from the designer of C++, Bjarne Stroustrup.

- Which C++?

- The language has been **evolving**:

C++ 1983  $\Rightarrow$  C++ 1998  $\Rightarrow$  C++ 2003  $\Rightarrow$  C++ 2011  $\Rightarrow \dots$

- We will learn C++11 (but not all the new features).

- Which compiler?

**GNU gcc/g++**. It is free.

(The compiler you will use in CSE lab is C++11-compliant.)

- Which IDE (integrated development environment) for writing programs?

**VS Code**. It is free and supported by many operating systems such as Windows, Mac OS, and Linux. Other freeware like **Eclipse** is also fine.