Programming with C++

COMP2011: Introduction

Cecia Chan
Gary Chan
Cindy Li
Wilfred Ng

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
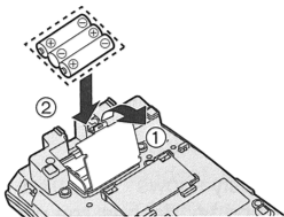
1 Press down in the direction of the arrow and open the cover (①).

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.
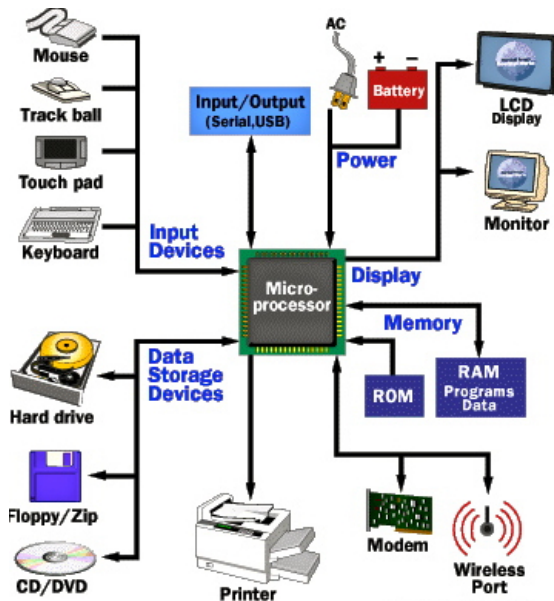 Battery life may depend on usage conditions and ambient temperature.

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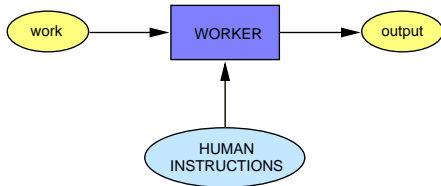
Computer work model
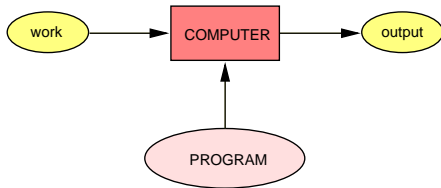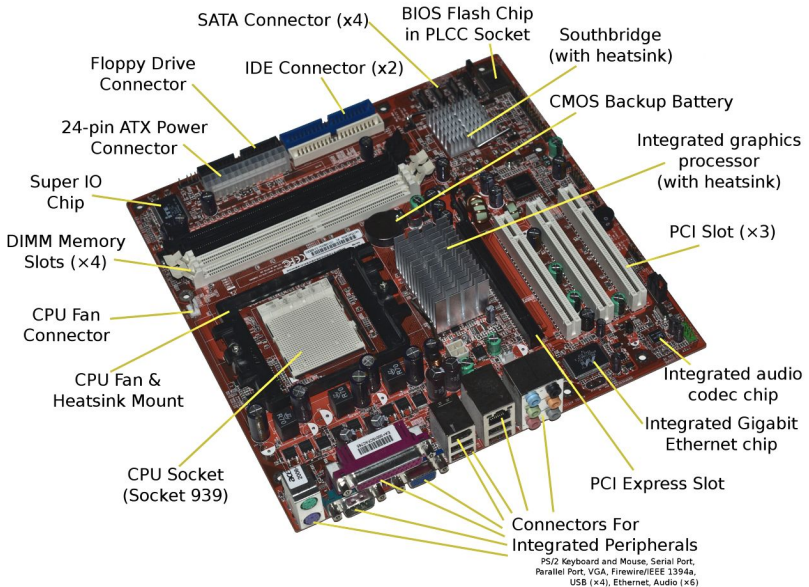
- A computer program is a set of machine-readable instructions that tells a computer how to perform a specific task. (During the execution of the program, it may interact with the users and its environment.)

Schematic Diagram of a Personal Computer

SATA Connector (x4)

BIOS Flash Chip
in PLCC Socket

Southbridge
(with heatsink)

Floppy Drive
Connector

IDE Connector (x2)

CMOS Backup Battery

24-pin ATX Power
Connector

Integrated graphics
processor
(with heatsink)

Super IO
Chip

PCI Slot (×3)

DIMM Memory
Slots (×4)

CPU Fan
Connector

CPU Fan &
Heatsink Mount

Integrated audio
codec chip

Integrated Gigabit
Ethernet chip

CPU Socket
(Socket 939)

PCI Express Slot

Connectors For
Integrated Peripherals
PS/2 Keyboard and Mouse, Serial Port,
Parallel Port, VGA, Firewire/IEEE 1394a,
USB (×4), Ethernet, Audio (×6)

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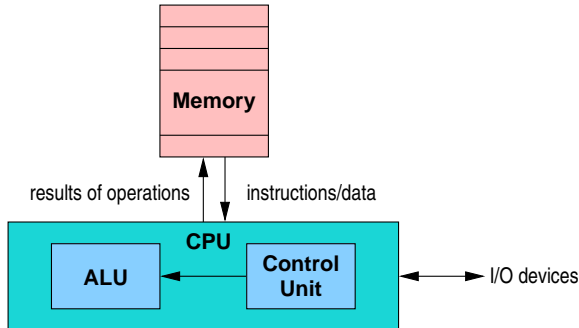
- Designed by John von Neumann, a mathematician, in 1945.
- It is still today's dominant computer architecture.
- CPU = Central Processing Unit
- ALU = Arithmetic Logic Unit.
- For efficiency, many programming languages, including C++, are designed to take advantage of the architecture.
- More on this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

```
00001001001011100110010011001101001011011000110010000100100100100100101100011001010110001101110100
0111
01010111001001001001001001011000100101110011000110010001000001010011001110110001101100011001100100
1011111
01100011011011011110101011000001010010101100101011000110010011001000100010110001110011100100001011
1000111
00110110010101100011011110000110100101011101101110000100100100100100110011101110100100100101101111
1000
01110100001000100000101000001001001011100110000101101100011010010110010011011011000110000001001000
0000
10100000100100101110011011101101010100010111101100100100010010001010110110000110100011000000101001
0110100
01101110000101000001001001011100110000101001110100100011100010001000010100110110001001010011010000
0111
01010101010001010010001110010000000110000000010100000001010100011001011100110110001011101100100000
0010
01011111001011000110010110010110010011010100010011000110010011000110110001001100010001001111010000
0010
11110010000010010010010111101010100001001110011000100001101000101010100110000010100100101000110
0100
00101001001001011001110100010101100001011001110100100110010010010011110110100010001101110110110100
0100
0010001011111000011000011010000111010101101011011100011010010010110011100010001000000111101110011
0001
00110010000000110010010001010000010010110101010110111010011010110110111011000100001011010101000111
0011
0110111010110001001010101010001101011011111100100000010010010011010101100010100100111010101010000
0010
11000001001001010000010100100100111010110010010010101000101010110010100010000010000000010101010111
0101
01011000100110010010010010010111110010001011100011010010010110011011011010011111010101101101100101
01
00010010010010000011000100000100110110010010010110101000100001010110101011100100100011001100000111
00
00011110010010010110110111010110011010011101100100010010001011000100110011010000011010001010101011
10
00011101101100011101000001101010010010011010010100111110110110010011101110110110110110001000010100
11
01000101001110110100011110010010010100100010001001011001010011010101010110100010110010101011111011
0110
11010010010010001111010101010100101011011000110010010011110111011010110001010010010101001101110010
010
10011010001000001110110010010001010111011001010101101001110010010001000100111101110011000110001010
0001
0001011111011000011010000111010101101101100011010001010101101110010010001001111010101101101100101
01
```

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?

```c
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.

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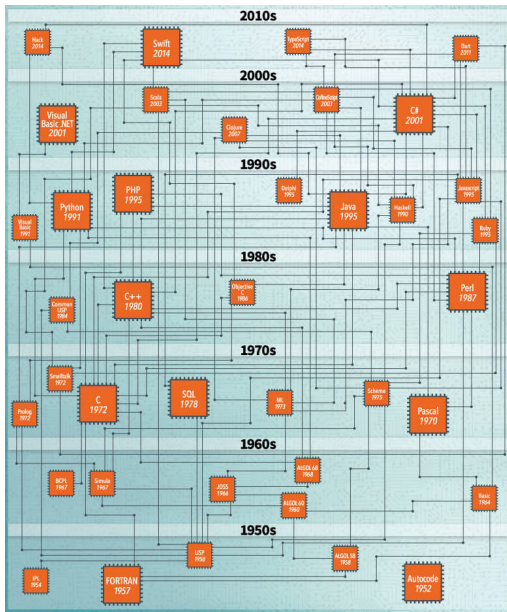
More details in this computer history website, and the following infographics.

TIOBE Index: Most Popular Programming Languages

TIOBE Programming Community Index
Source: www.tiobe.com
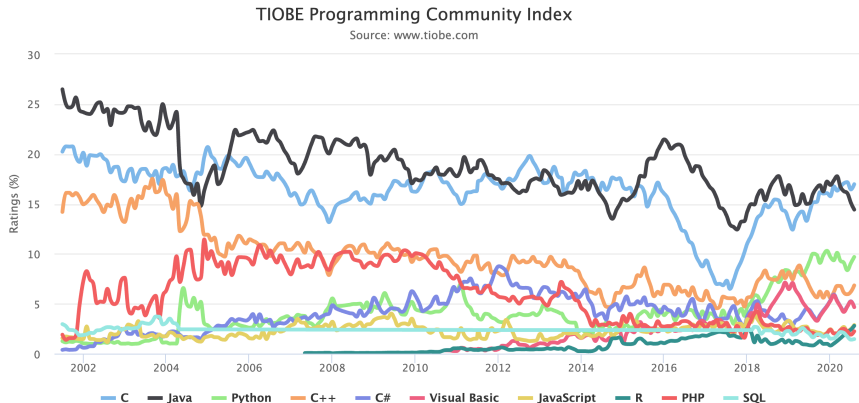
It is based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Baidu, etc. are used to calculate the ratings.

PYPL PopularitY: Most Popular Programming Languages

Worldwide, Sept 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	31.56 %	+2.9 %
2		Java	16.4 %	-3.1 %
3		Javascript	8.38 %	+0.3 %
4		C#	6.5 %	-0.8 %
5		PHP	5.85 %	-0.5 %
6		C/C++	5.8 %	+0.0 %
7		R	4.08 %	+0.3 %
8		Objective-C	2.79 %	+0.2 %
9		Swift	2.35 %	-0.1 %
10		TypeScript	1.92 %	+0.1 %

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

Mostly Used Programming Languages in Github

# Ranking	Programming Language	Percentage (Change)
1	JavaScript	18.789% (-1.133%)
2	Python	16.108% (-1.694%)
3	Java	10.731% (+0.250%)
4	Go	8.922% (+1.006%)
5	C++	7.636% (+0.383%)
6	TypeScript	7.334% (+1.919%)
7	Ruby	6.492% (+0.196%)
8	PHP	5.198% (-0.318%)
9	C#	3.797% (-0.204%)
10	C	3.320% (+0.130%)

Compilation: From Source to Runnable Program
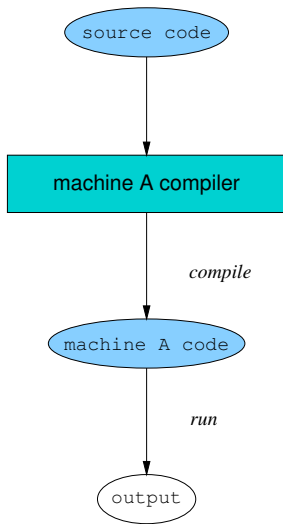
A compiler translates source programs into machine codes that run directly on the target computer.

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++

```
source code
```

```
machine A compiler
```

compile

```
machine A code
```

run

```
output
```

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.

Summary

- 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 $\Rightarrow$ $\cdots$
  - 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.