CS2311 Computer Programming

LT1: Introduction to Programming

What is Programming

- These are NOT programming (IMO)
  - HTML, markdown, or copying code
- 1st aspect: design a solution, or an algorithm, to solve a problem using computers
  - Algorithms, data structures, systems, hardware
- 2nd aspect: write a program to instruct a computer, or computers, to realize the solution
  - Implementation
  - *This course is more about this!*

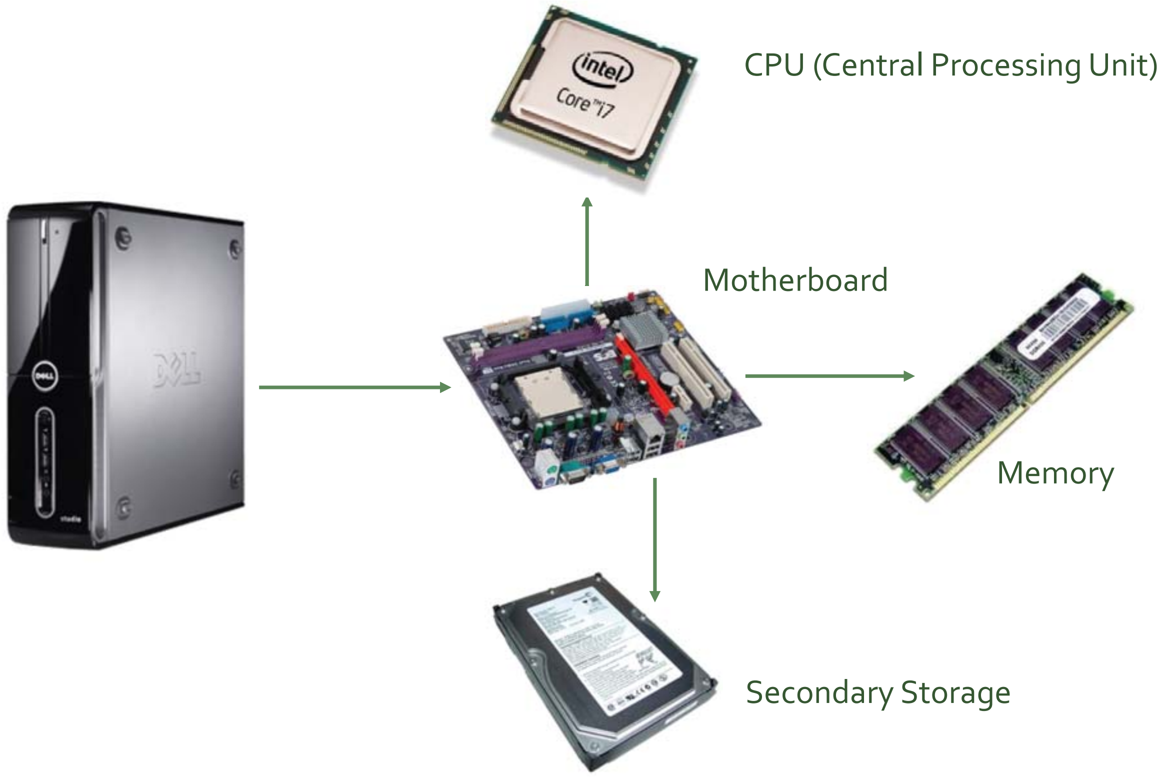
Why Learn Programming

Why Learn Programming

"Everybody in this country should learn how to program a computer, because it teaches you how to think."

Critical thinking: Logic Rigor Clarity

What is a Computer



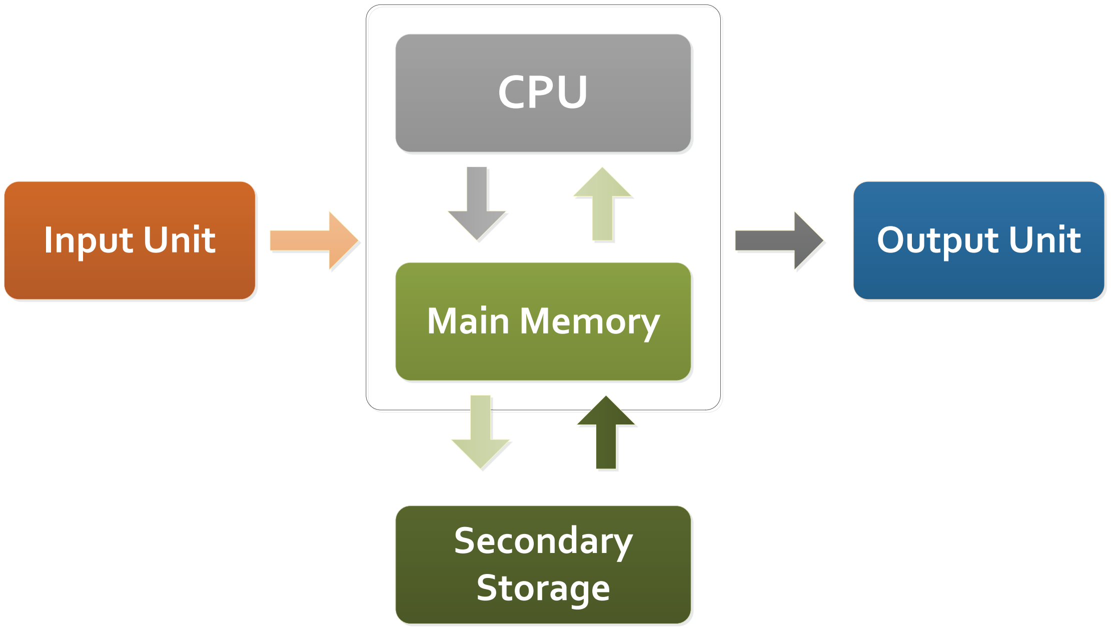
CPU (Central Processing Unit)

Motherboard

Memory

Secondary Storage

Stored Program Computer (Von Neumann Machines)



Input Unit

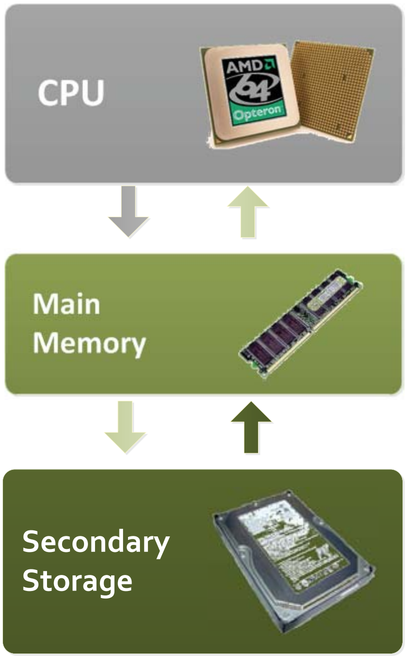
CPU

Main Memory

Output Unit

Secondary Storage

Personal Computer



CPU

Main Memory

Secondary Storage

CPU (Central Processing Unit): Read instruction from main memory and execute the instruction. Update main memory value or send instruction to motherboard

Main Memory: fast storage of program and data in action

Secondary Storage: Storage of program and data files

Personal Computer

Input Unit

Input Unit: Get input from user or external environment

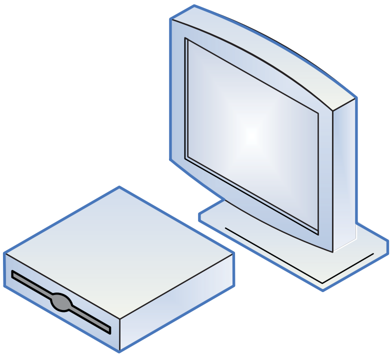
Output Unit

Output Unit: Show result to user or other programs

What is a Computer Program?

- A list of instructions that tells a computer to do something

Timer Recording	Program X
1. Turn on	int x=10;
2. Set Channel to **ch01**	int y=11;
3. Set Date to **1/9/2018**	y+=x;
4. Set Time to **3:00am**	System.out.println(y);
5. Confirm setting	System.out.println(x);



A Computer Program

- A way to communicate with computers
- Written in a programming language

Programming Languages

- To write a program for a computer, we must use a computer language.

Machine Language

Directly understood by the computer

Symbolic Language

English-like abbreviations representing elementary computer operations

High-level Language

Close to human language.

Example: *a* = *a* + *b*

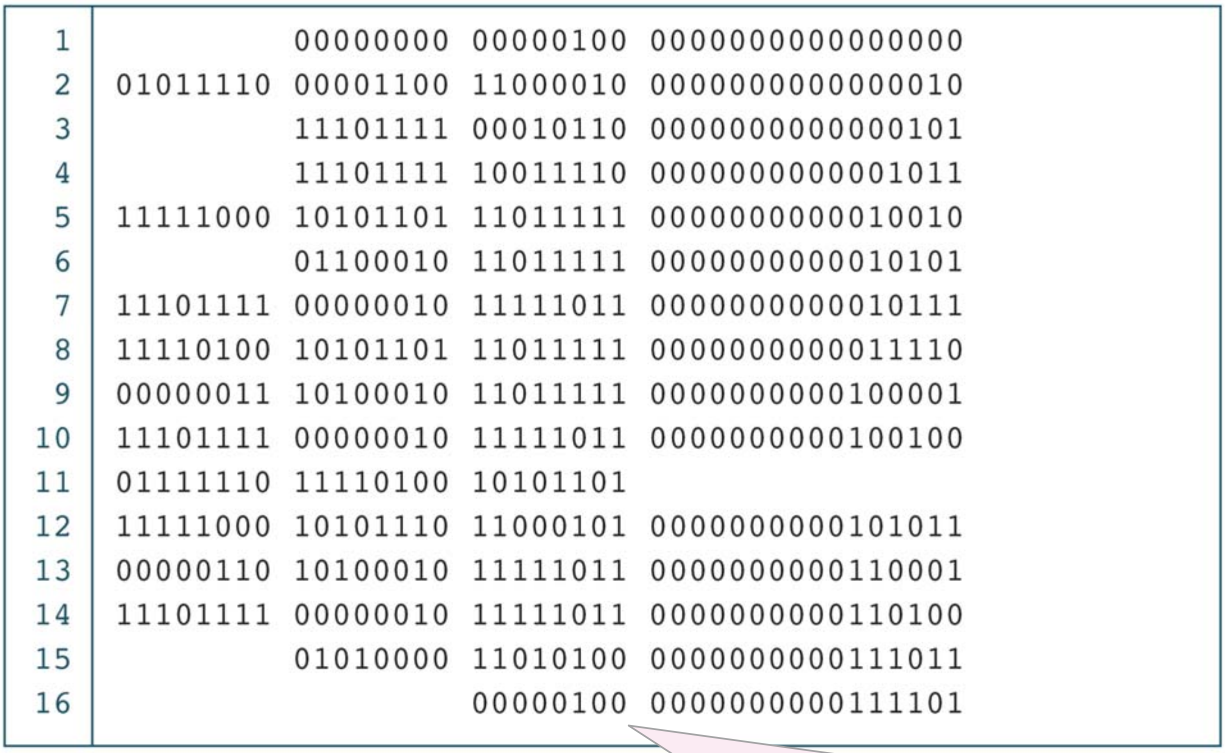
[add values of *a* and *b*, and store the result in *a*, replacing the previous value]

binary code

assembly language

C, C++, Java, Python

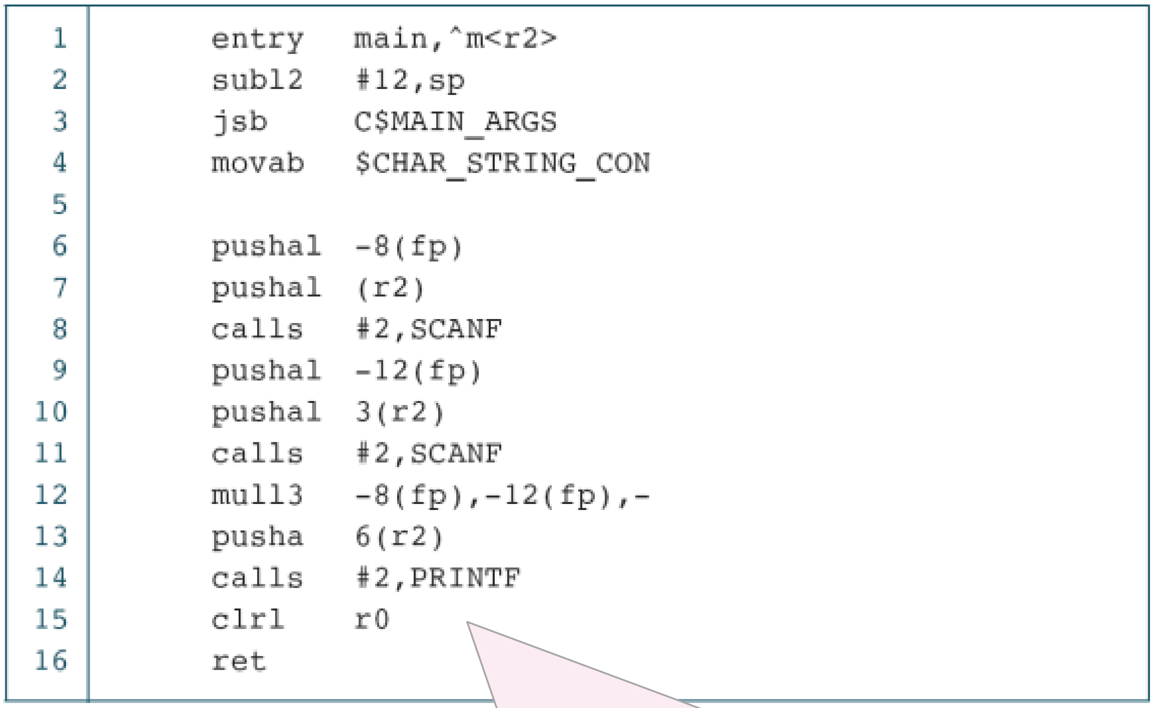
PROGRAM 1-1 The Multiplication Program in Machine Language



```
         00000000 00000100 0000000000000000
01011110 00001100 11000010 0000000000000010
         11101111 00010110 0000000000000101
         11101111 10011110 0000000000001011
11111000 10101101 11011111 0000000000010010
         01100010 11011111 0000000000010101
11101111 00000010 11111011 0000000000010111
11110100 10101101 11011111 0000000000011110
00000011 10100010 11011111 0000000000100001
11101111 00000010 11111011 0000000000100100
01111110 11110100 10101101
11111000 10101110 11000101 0000000000101011
00000110 10100010 11111011 0000000000110001
11101111 00000010 11111011 0000000000110100
         01010000 11010100 0000000000111011
                  00000100 0000000000111101
```

The only language understood by computer hardware is machine language.

PROGRAM 1-2 The Multiplication Program in Symbolic Language

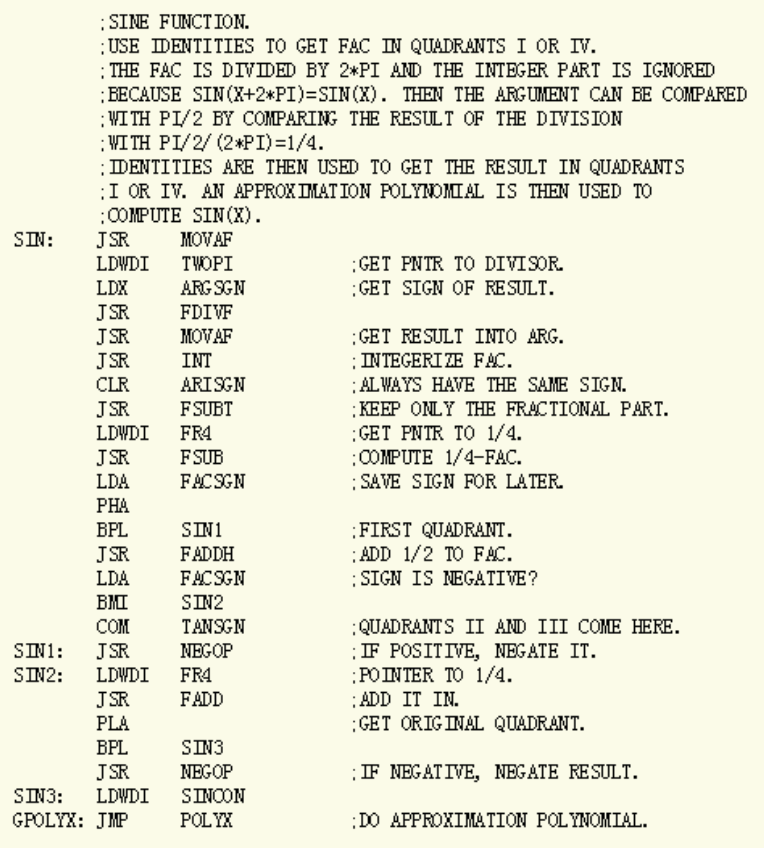


```
        entry   main,^m<r2>
        subl2   #12,sp
        jsb     C$MAIN_ARGS
        movab   $CHAR_STRING_CON

        pushal  -8(fp)
        pushal  (r2)
        calls   #2,SCANF
        pushal  -12(fp)
        pushal  3(r2)
        calls   #2,SCANF
        mull3   -8(fp),-12(fp),-
        pusha   6(r2)
        calls   #2,PRINTF
        clrl    r0
        ret
```

Symbolic language uses symbols, or mnemonics, to represent the various machine language instructions.

Part of Assembly Code for Microsoft BASIC, by Bill Gates



```
        ;SINE FUNCTION.
        ;USE IDENTITIES TO GET FAC IN QUADRANTS I OR IV.
        ;THE FAC IS DIVIDED BY 2*PI AND THE INTEGER PART IS IGNORED
        ;BECAUSE SIN(X+2*PI)=SIN(X). THEN THE ARGUMENT CAN BE COMPARED
        ;WITH PI/2 BY COMPARING THE RESULT OF THE DIVISION
        ;WITH PI/2/(2*PI)=1/4.
        ;IDENTITIES ARE THEN USED TO GET THE RESULT IN QUADRANTS
        ;I OR IV. AN APPROXIMATION POLYNOMIAL IS THEN USED TO
        ;COMPUTE SIN(X).
SIN:    JSR     MOVAF
        LDWDI   TWOPI           ;GET PNTR TO DIVISOR.
        LDX     ARGSGN          ;GET SIGN OF RESULT.
        JSR     FDIVF
        JSR     MOVAF           ;GET RESULT INTO ARG.
        JSR     INT             ;INTEGERIZE FAC.
        CLR     ARISGN          ;ALWAYS HAVE THE SAME SIGN.
        JSR     FSUBT           ;KEEP ONLY THE FRACTIONAL PART.
        LDWDI   FR4             ;GET PNTR TO 1/4.
        JSR     FSUB            ;COMPUTE 1/4-FAC.
        LDA     FACSGN          ;SAVE SIGN FOR LATER.
        PHA
        BPL     SIN1            ;FIRST QUADRANT.
        JSR     FADDH           ;ADD 1/2 TO FAC.
        LDA     FACSGN          ;SIGN IS NEGATIVE?
        BMI     SIN2
        COM     TANSGN          ;QUADRANTS II AND III COME HERE.
SIN1:   JSR     NEGOP           ;IF POSITIVE, NEGATE IT.
SIN2:   LDWDI   FR4             ;POINTER TO 1/4.
        JSR     FADD            ;ADD IT IN.
        PLA                     ;GET ORIGINAL QUADRANT.
        BPL     SIN3
        JSR     NEGOP           ;IF NEGATIVE, NEGATE RESULT.
SIN3:   LDWDI   SINCON
GPOLYX: JMP     POLYX           ;DO APPROXIMATION POLYNOMIAL.
```

http://www.pagetable.com/docs/M6502.MAC.txt

PROGRAM 1-3 The Multiplication Program in C

```
/* This program reads two integers from the keyboard
   and prints their product.
      Written by:
      Date:
*/
#include <stdio.h>

int main (void)
{
// Local Definitions
   int number1;
   int number2;
   int result;

// Statements
   scanf  ("%d", &number1);
   scanf  ("%d", &number2);
   result = number1 * number2;
   printf ("%d", result);
   return 0;
}  // main
```

high-level languages are easier for us to understand.

There are Many Programming Languages in the World

Ada Assembly Basic C C++ C# Cobol Cobra CODE ColdFusion Delphi Eiffel Fortran FoxPro GPSS J# J++ Java JavaScript LISP Logo LUA MEL Modula-2 Miranda Objective-C Perl PHP Prolog Python SQL Visual Basic Swift

Programming Languages

- Programming languages usually differ in two aspects
  - Language Syntax
  - Standard libraries / SDKs / functions
- Java

```
if (a>b) {
     System.out.println("a is larger than b");
} else {
     System.out.println("a is smaller than or equal to b");
}
```

- Pascal

```
if a>b then
     writeln('a is larger than b');
else
     writeln('a is smaller than or equal to b');
```

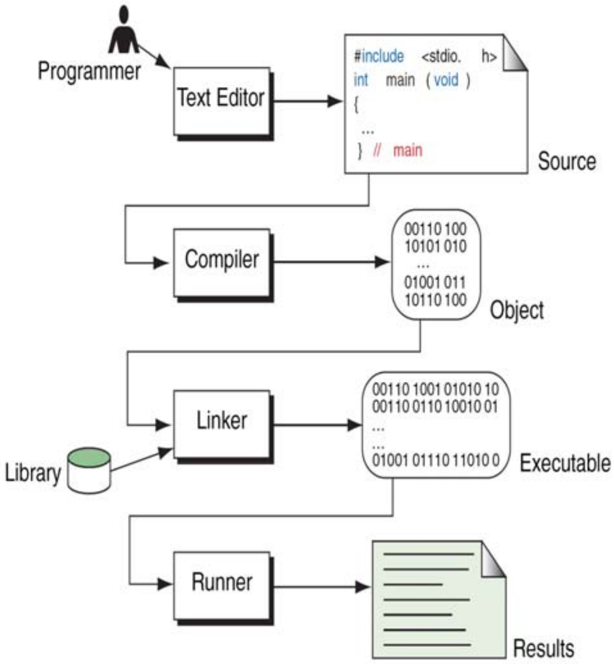
Programming Languages

- Syntax is well-defined, no exception
  - if (…){…} else {…};
  - for (;;;) {…}
  - while () {…}

- Basic Components:
  - Variables / structures / function declaration
  - Variables / structures / function access
  - Conditional statements
  - Iteration statements
  - SDK/built-in functions

Building a C++ Program

- **Writing** source code in C++
  - e.g. hello.cpp
- **Preprocessing**
  - Processes the source code for compilation
- **Compilation**
  - Checks the grammatical rules (syntax)
  - Source code is converted to object code in machine language (e.g. hello.obj)
- **Linking**
  - Combines object code and libraries to create an executable (e.g. hello.exe)
  - Library: common functions (input, output, math, etc.)



Being a Programmer

Some Difficulties

- Computer only follows instructions. It won't solve problems by itself
- A programmer needs to:
  1. develop an appropriate solution (logic)
  2. express the solution in a programming language (implementation)
  3. validate the logic and implementation (testing)

Requirements

- Correct syntax
- Correct logic
- Efficient
- Running properly under various constraints
- Scalability, Maintainability
- Platform independent

Do the thing right, Do the right thing

- It's a lot easier to learn how to do things right
  - Syntax is easy to learn (as long as you want to learn…)
- Takes many failures to learn what's the right thing to do
  - The correct logic, way of solving the problem
  - Then try to make it more efficient

Career Prospects

- Software engineer
  - Tencent, Alibaba, Microsoft, Google
- Data scientist
  - Tencent, Alibaba, Apple, Airbnb, Instagram, Tesla
- ML architect
  - Amazon, Google, Facebook, Microsoft, BAT in China
- Chip architect:
  - Qualcomm, Cambricon in China
- Financial engineer:
  - Banks, hedge funds
- Researcher, professor

Basic Concepts of Programming

CONTROL FLOW
DATA REPRESENTATION
LANGUAGE SYNTAX
LANGUAGE SEMANTICS
PRE-PROCESSOR DIRECTIVES
FUNCTIONS
LIBRARY

A Computer Program

- Instructions
  - A set of predefined action that a computer can perform
    - E.g. addition, subtraction, read , write
- Logic Flow
  - Arrangement of Instructions
    - E.g. Calculate BMI
      1. Read weight from keyboard
      2. Read height from keyboard
      3. Weight x weight/height
      4. Write BMI to screen
- Variable (data)
  - A space for temporarily store value for future process
- Constant (data)
  - A value that will not be changed for the whole processing

A Simple Program

```
/* The first program in honor of Dennis Ritchie who invented C at Bell Labs in
   1972 */

#include <iostream>
using namespace std;

int main()
{
   cout << "Hello, world!\n";
   return 0;
}
```

Comments

```
/* The traditional first program in honor of
  Dennis Ritchie who invented C at Bell Labs
  in 1972 */
```

- Enclosed by "**/***" and "***/**" or begin with "**//**"
- **//** single line comments

```
// this is a single line comment
// each line must begin with "//" sign
```

Preprocessor Directive

- Give information / instruction to compiler for program creation

  #include <iostream>
  - Preprocessor directive
  - Tells computer to load contents of a certain file/library
  - In this program, we include the library **iostream** into the program as it contains the definition of **cout** which is used to print something to the screen.
  - No semi-colon at the end of the include directive

  using namespace std;
  - Specifying that the standard (**std**) namespace is used such that we can use a shorthand name for the object **cout**
    - **std::cout <-> cout**

Functions

- When writing a program, the programmer usually group related code into functions for easy design and maintenance

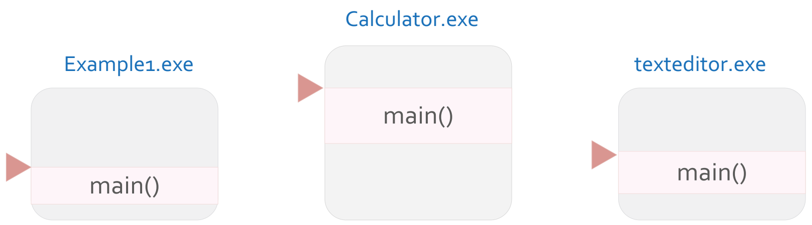
- We will talk about functions and how to write your own functions in later lectures

Functions – the main function

```
int main()
{

    return 0;
}
```

- The starting point of program
  - the first function called by the computer



Function – main

- **int main()**
  - **int** means the return value of the function is an integer
  - no semi-colon after **main()**
  - C++ is case sensitive
  - **void main()** works for some compilers
  - Incorrect: **int Main()**, **Int main()**, …
- **{ }**
  - Braces: left brace begins the body of a function. The corresponding right brace must end the function
- **return 0**
  - The **main()** function has to return an integer upon completion
  - **0** is returned to indicate the program exits successfully

Function – **main**

- Sometimes it’s also okay to have the following, for certain compilers
- **void main()**
  - **void** means there is no return value
  - In this course, we stick to **int main()**

Simple Program

```
/* The traditional first program in honor of
  Dennis Ritchie who invented C at Bell Labs
  in 1972 */

#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, world!\n";
    return 0;
}
```

Library / SDK / Package

- Normally, we won't write a program all by ourselves. Instead, we will reuse the code written by ourselves / other developers Especially for the repeating tasks or low-level operation like disk I/O
- The reusing code is well designed and packed as libraries / SDK / packages
- Standard C++ program comes with a set of packages to make programmer task easier
- **iostream** is one example

Object – **cout**

```
cout << "Hello, world!\n";
```

- Object is a programming unit that store values (attributes) and provide functions (methods) to manipulate the values (we will elaborate this concept in future classes)
- **cout**: object provided by **iostream** library (package) for screen (console) output
- **<<**: output (also called insertion) operator that outputs values to an output device. In this case, the output device is **cout** (the screen)
- The value on the right hand side of the operator is the string you want to output

Object – **cout**

- **\n**
  - escape sequence: the character following \ is not interpreted in the normal way
  - represents a newline character: the effect is to advance the cursor on the screen to the beginning of the next line
  - newline: position the character to the beginning of next line
- \\
  - backslash: Insert the backslash character \ in a string
- **\\"**
  - double quote: Insert the double quote character **"** in a string

Summary

- Basic components of a computer program are:
  - Instructions
  - Logic Flow
  - Variables and Constants
- A correct logic is important in programming
- Programmer usually reuse code written by the others and the code is commonly in form of library / SDK / packages
- **cout** is an object provided by **iostream** package for screen output

Summary

- A simple C++ program will have

```
#include <iostream>        //A preprocessor
using namespace std;       //namespace declaration
int main() {
        /* the starting point of program execution  */

        return 0;
}
```

Summary

- Development cycle
  - Write a program in plan text via
    - Text editor
      - Notepad, UltraEdit
      - Vim, Sublime Text
    - Integrated Development Environment (IDE)
      - E.g. Visual Studio, NetBean, Eclipse
  - Compile the program
    - IDE / ANSI C++
  - Execute the program
    - IDE / Console shell
  - Debug the program