Introduction to programming tutorial class

C / C++

Introduction

> How does a computer work?

> What is a program?

> What is a programming langage?

> Let’s get into it!

A/ Computer Basics

I/ HARDWARE BASICS

i- **C**entral **P**roccessing **S**ystem (CPU)

ii- Memory System

> **R**andom **A**ccess **M**emory (RAM)

> Hard Drive

iii- Input - output (I/O)

> Computer Peripheral *(mouse, keyboard, screen etc)*

Classes of Systems

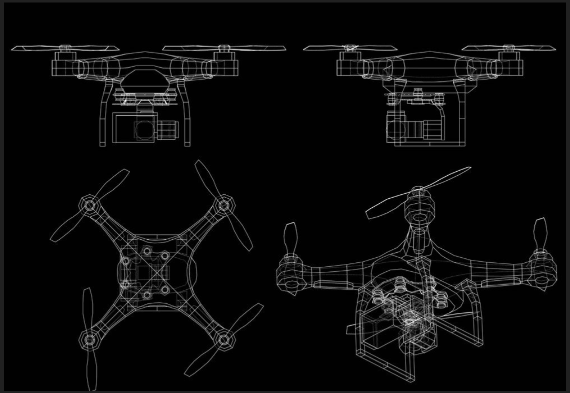
> Client / Work Station *(Personnal computers)* = Single user device

> Server

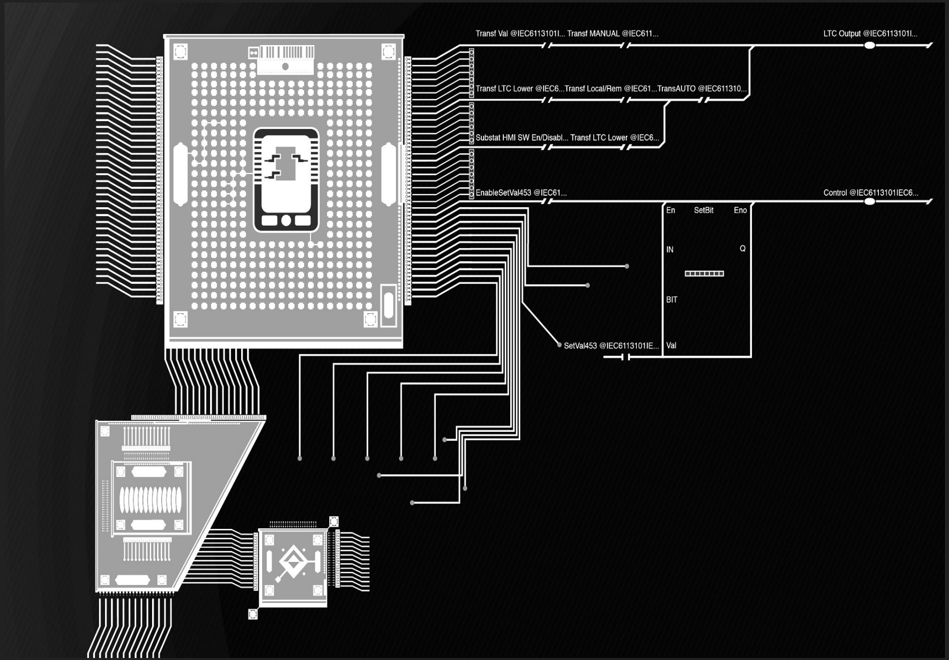
- *responds to request from Network*
- *no direct human interface*
- *use of server to communicate with a Network*

> Hand Held = *Phone, audioplayers, smartwatches etc*

> Embedded Systems = *car, coffee machine etc*



i- Central Proccessing System (CPU)


Transf Val @IEC6113101I... Transf MANUAL @IEC611...
LTC Output @IEC6113101I...
Transf LTC Lower @IEC6...Transf Local/Rem @IEC61...TransAUTO @IEC611310...
Substat HMI SW En/Disabl... Transf LTC Lower @IEC6...
EnableSetVal453 @IEC61...
Control @IEC6113101IEC6...
En
SetBit
Eno
IN
Q
BIT
Val
SetVal453 @IEC6113101IE...

> **copy bytes**

> **arithmetics** *(at least addition and negation)*

> **bit logic** **not / and/or / exclusive or**

(those single instructions help manipulating single bits amongst bytes)

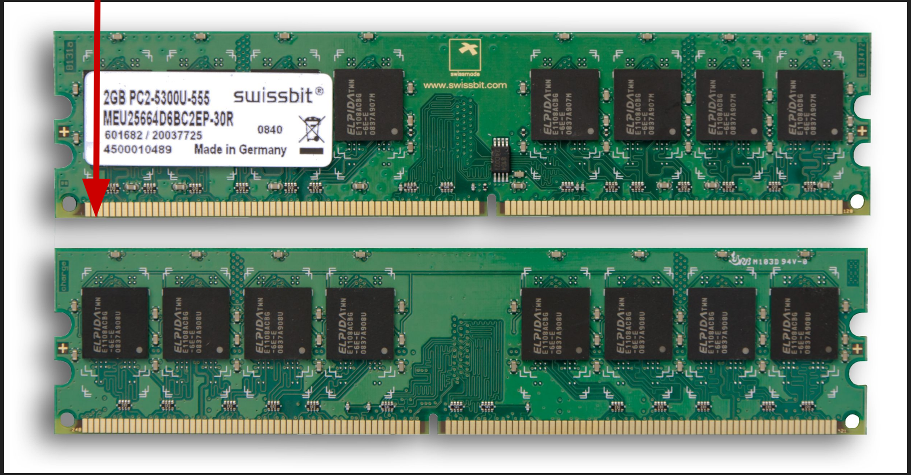
> **jumps** *navigates memory*

CPU is composed two kinds of registers:

- *status:* *stores data affecting the operation of the CPU*
- *general purpose:* *stores any data the CPU needs to compute inside itself, needed data to perfom operation is briefly stored there*

ii- Random Access Memory

this is a byte



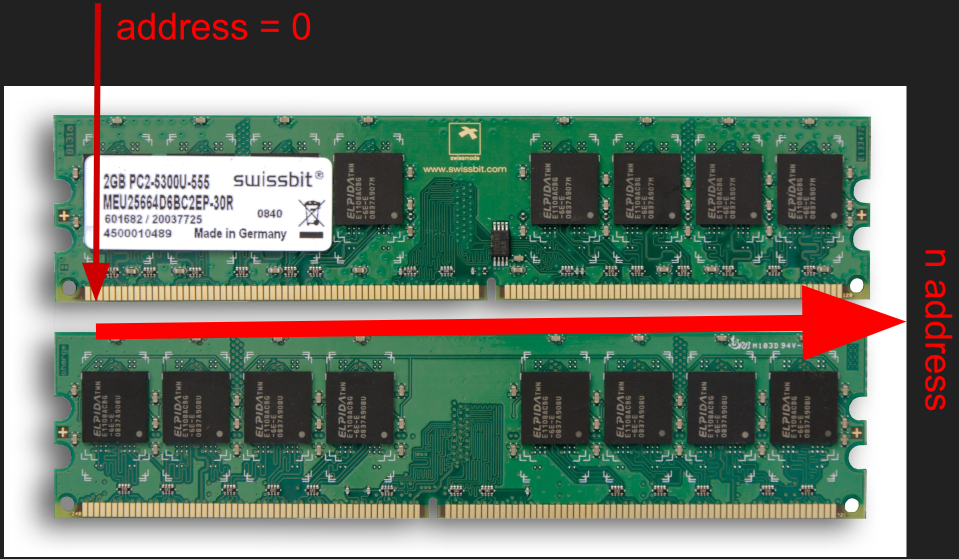
ii- Random Access Memory

> addressable by the CPU

> volatile : *when the power goes off the content of the ram gets erased*

> faster than storage to read and write

> stores code and data of running programs



B/ Operating System Basics

> load and manage processes

> provide interfaces to hardware via system calls

> provide filesystem

> ***provide a basic user interface***

MS-DOS -> **microsft windows** *for pc and windows server for servers*

UNIX = *Linux, BSD(berkley software distribution), OS X (based on BSD)*

MS-DOS -> **microsft windows** *for pc and windows server for servers*

UNIX = *GNU/Linux,* **BSD** *(berkley software distribution), OS X (based on BSD)*

FreeBSD

> First version issued in 1977

> open source

> written in **C**

> modular monolitic core

MS-DOS -> microsft windows

for pc and windows server for servers

UNIX = *Linux, BSD(berkley software distribution), OS X (based on BSD)*

Windows

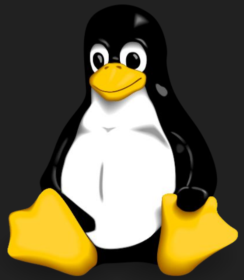
> First version issued in 1985

> written in **C**, **C++** and Assembler

> Core inspired by VAX VMS amd UNIX

MS-DOS -> **microsft windows** *for pc and windows server for servers*

UNIX = **GNU/Linux**, *BSD(berkley software distribution), OS X (based on BSD)*



> First version issued in 1991
>
> open source
>
> written in **C** and Assembler
>
> UNIX core

MS-DOS -> **microsft windows** *for pc and windows server for servers*

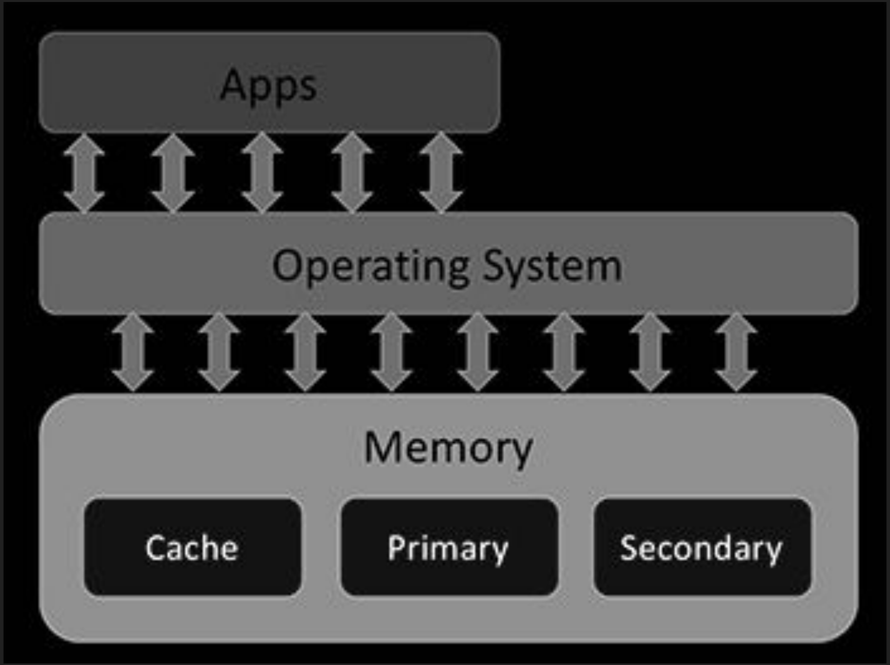
UNIX = *GNU/Linux,* BSD *(berkley software distribution),* **OS X (based on BSD)**

> First version issued in 1994

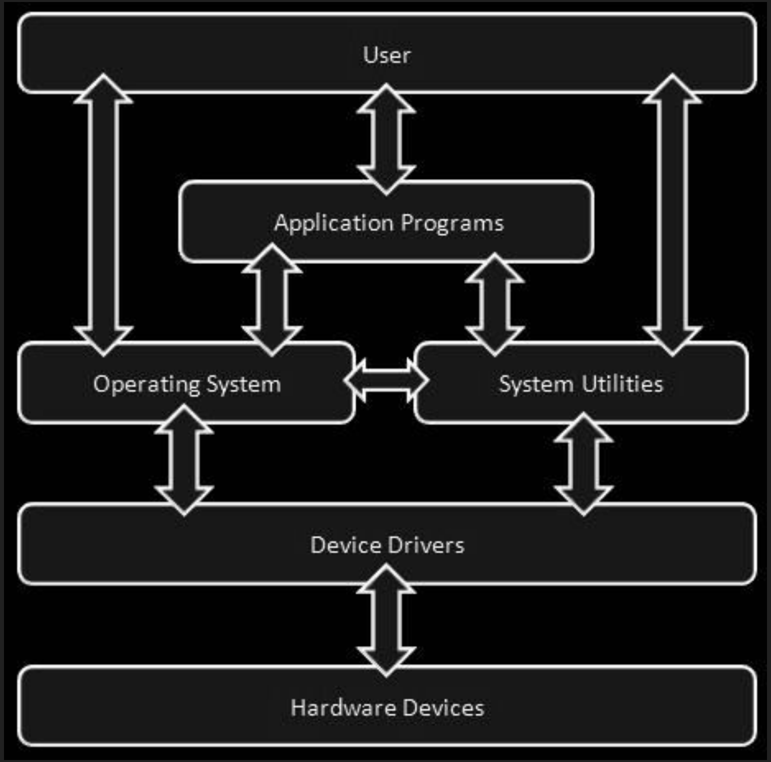
> written in **C, C++,** objective C, Swift, Assembly

> monolitic core

Program memory management



Device drivers

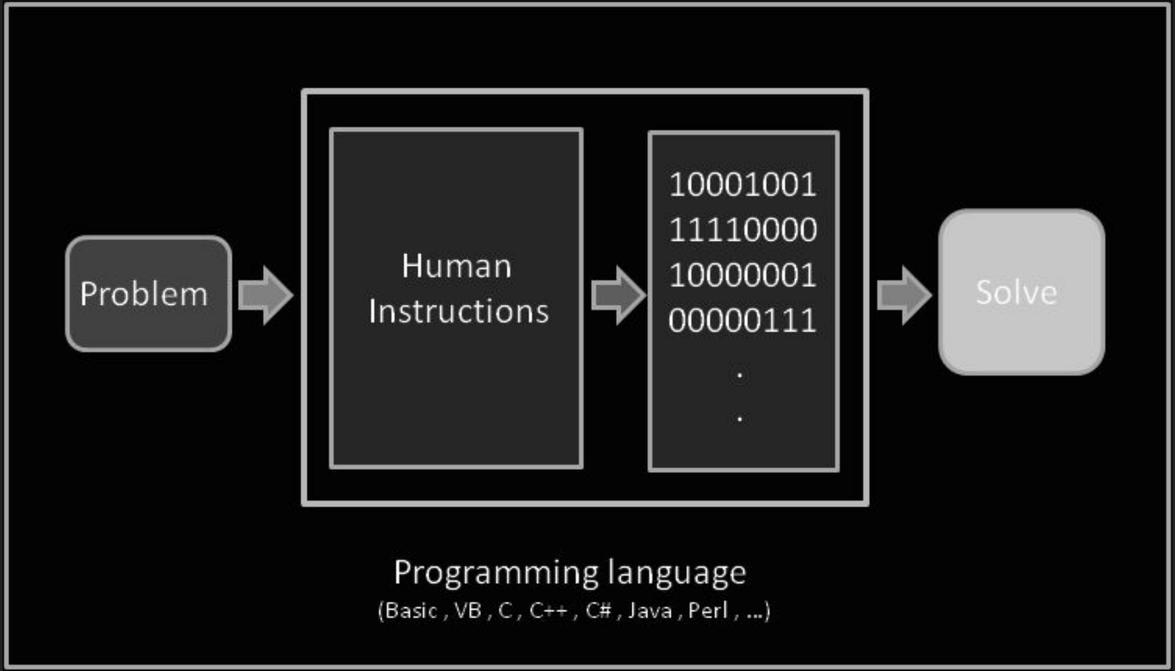


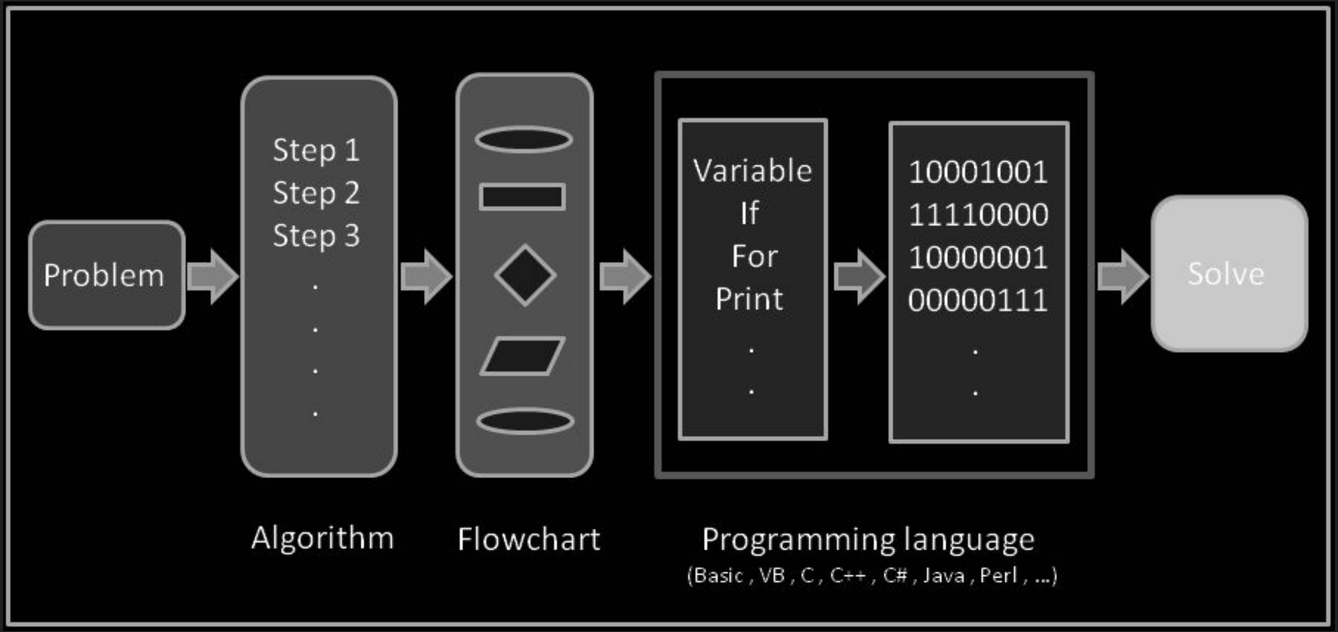
So, what is a *program*?

B/ Programs

A program is a step-by-step list of instructions for the computer to interpret in order to execute tasks aiming to solve a given problem. To interact with the computer you need to speak it's langage.

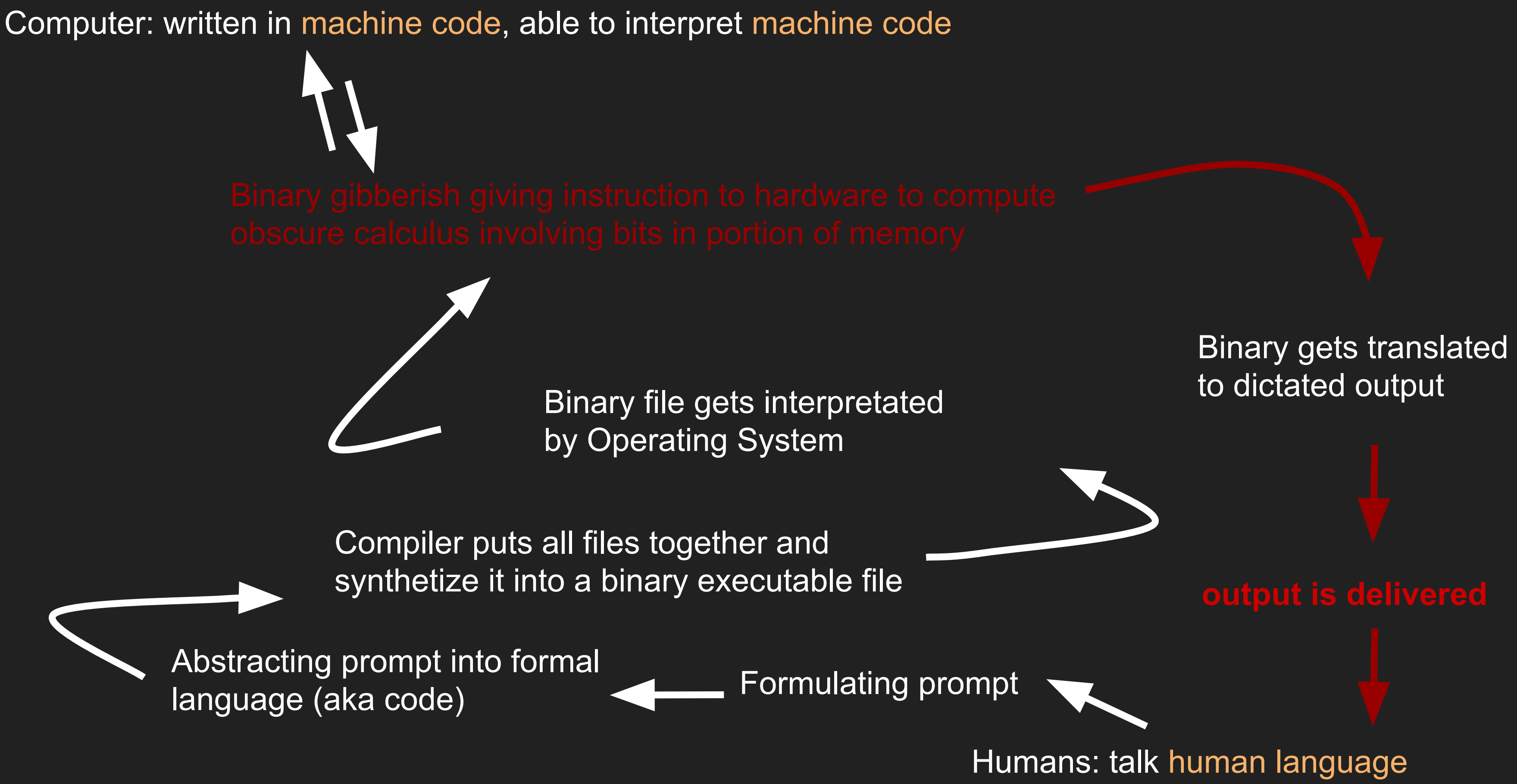
> Read input
> Parse input
> Process data
> Store data
> Perform tasks in order to provide the desired output


Problem
Human Instructions
10001001
11110000
10000001
00000111
.
.
Solve
Programming language
(Basic , VB , C , C++ , C# , Java , Perl , ...)


Problem
Step 1
Step 2
Step 3
Variable
If
For
Print
10001001
11110000
10000001
00000111
Solve
Algorithm
Flowchart
Programming language
(Basic , VB , C , C++ , C# , Java , Perl , ...)

Then, how do we *talk* to the computer?

C/ Programming languages



Computer: written in machine code, able to interpret machine code

Binary gibberish giving instruction to hardware to compute obscure calculus involving bits in portion of memory

Binary gets translated to dictated output

Binary file gets interpretated by Operating System

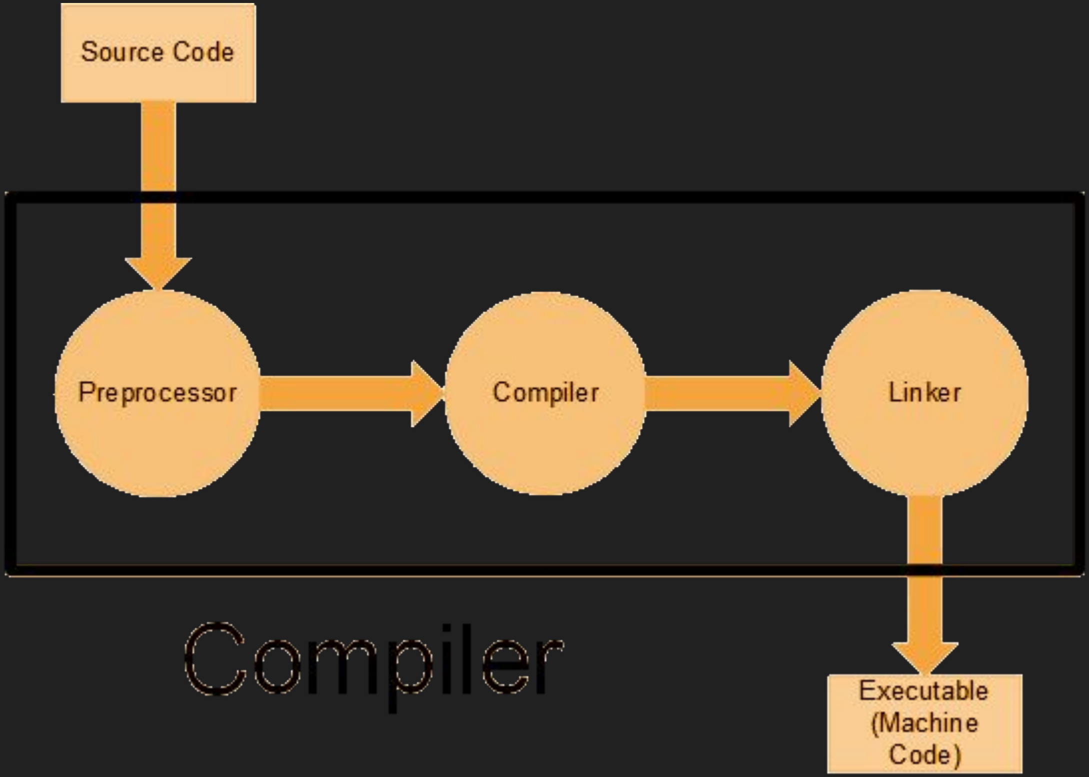
Compiler puts all files together and synthetize it into a binary executable file

output is delivered

Abstracting prompt into formal language (aka code)

Formulating prompt

Humans: talk human language


Source Code
Preprocessor
Compiler
Linker
Compiler
Executable (Machine Code)

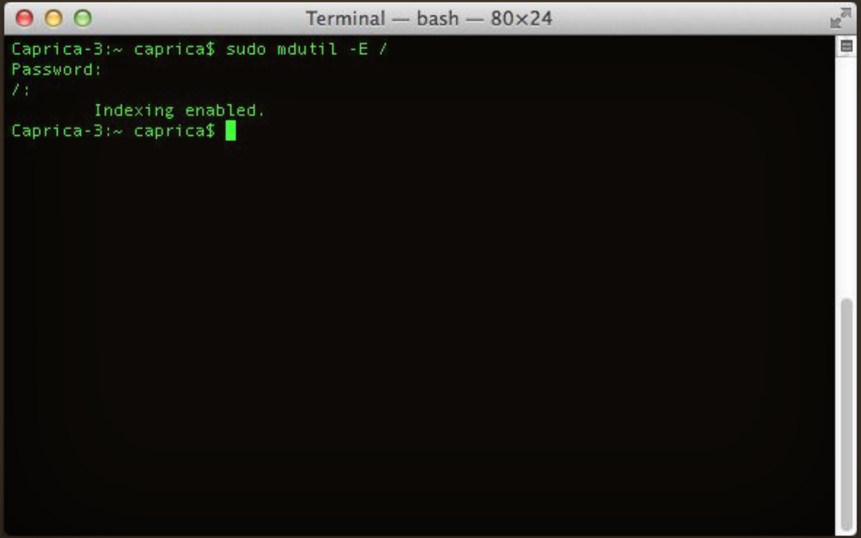
Integrated Developpement Environement (IDE): set up tool that helps write and debug

terminal: text input/output environment

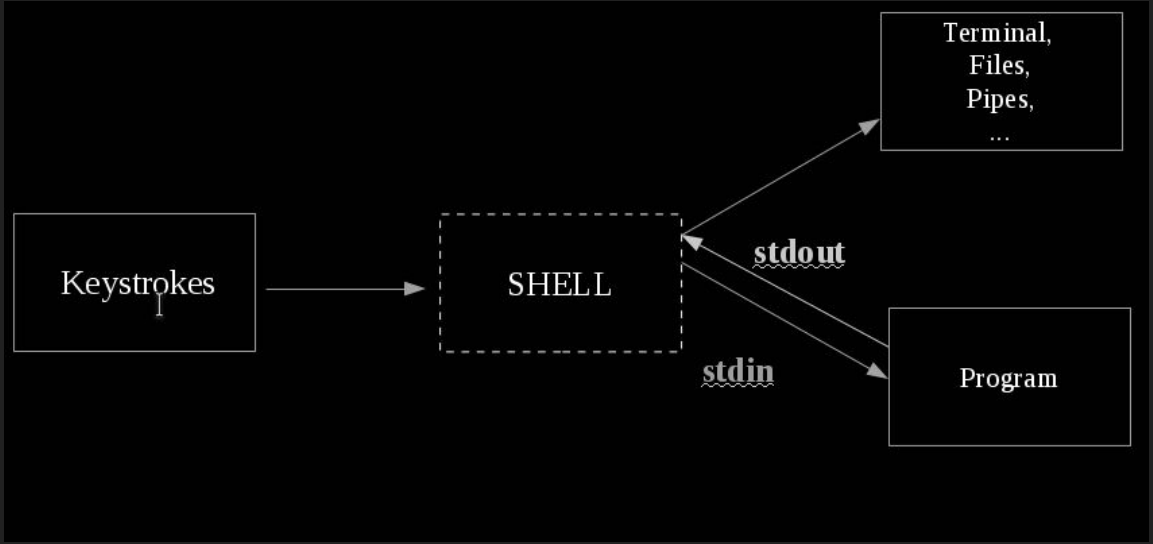
console: physical terminal

shell: command line interpreter

command line: instruction

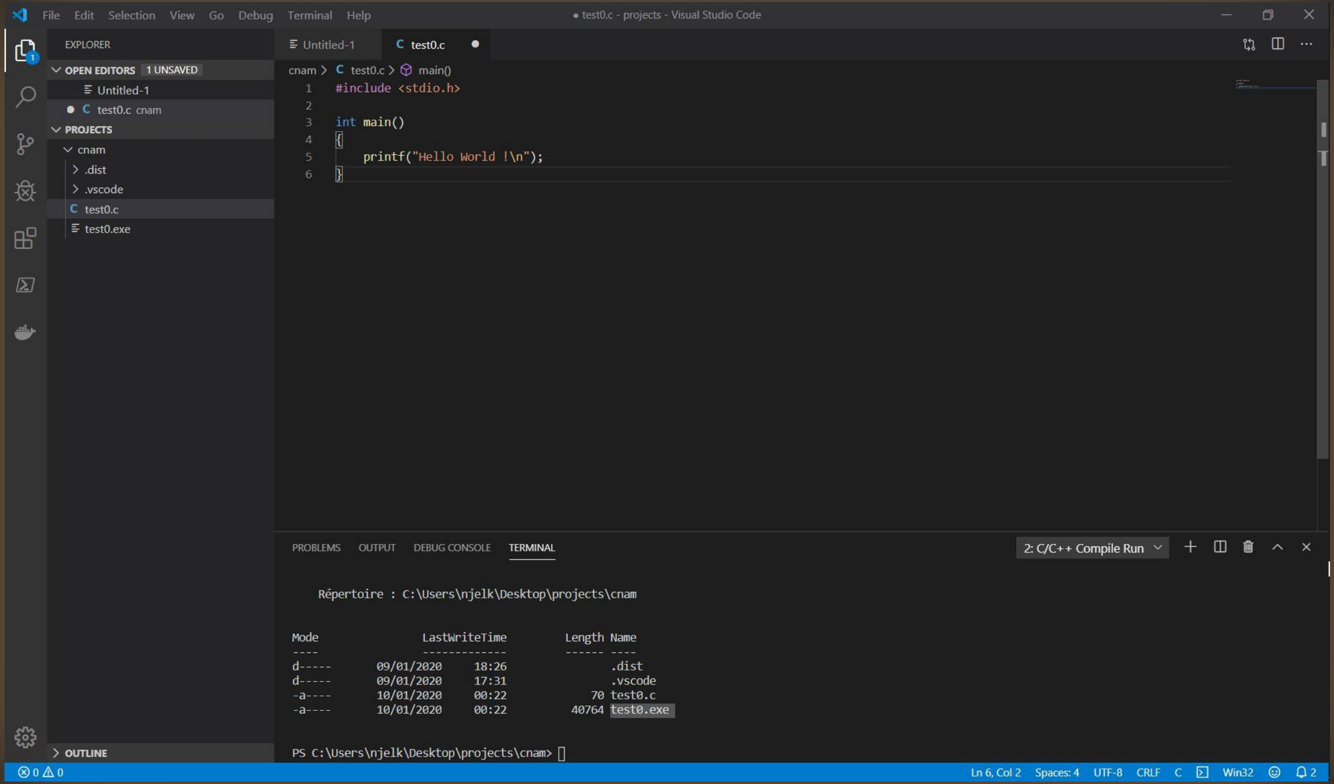


```
Caprica-3:~ caprica$ sudo mdutil -E /
Password:
/:
        Indexing enabled.
Caprica-3:~ caprica$ 
```


Keystrokes
SHELL
Terminal,
Files,
Pipes,
...
stdout
stdin
Program

Visual Code Studio

- runs under Windows, Linux and Mac OS
- Most used IDE under Windows
- Opensource (= free!)


File Edit Selection View Go Debug Terminal Help
● test0.c - projects - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
Untitled-1
● test0.c cnam
PROJECTS
cnam
.dist
.vscode
test0.c
test0.exe
OUTLINE
Untitled-1
test0.c
cnam > test0.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: C/C++ Compile Run
⊗ 0 ⚠ 0
Ln 6, Col 2 Spaces: 4 UTF-8 CRLF C Win32 2

```c
#include <stdio.h>

int main()
{
    printf("Hello World !\n");
}
```

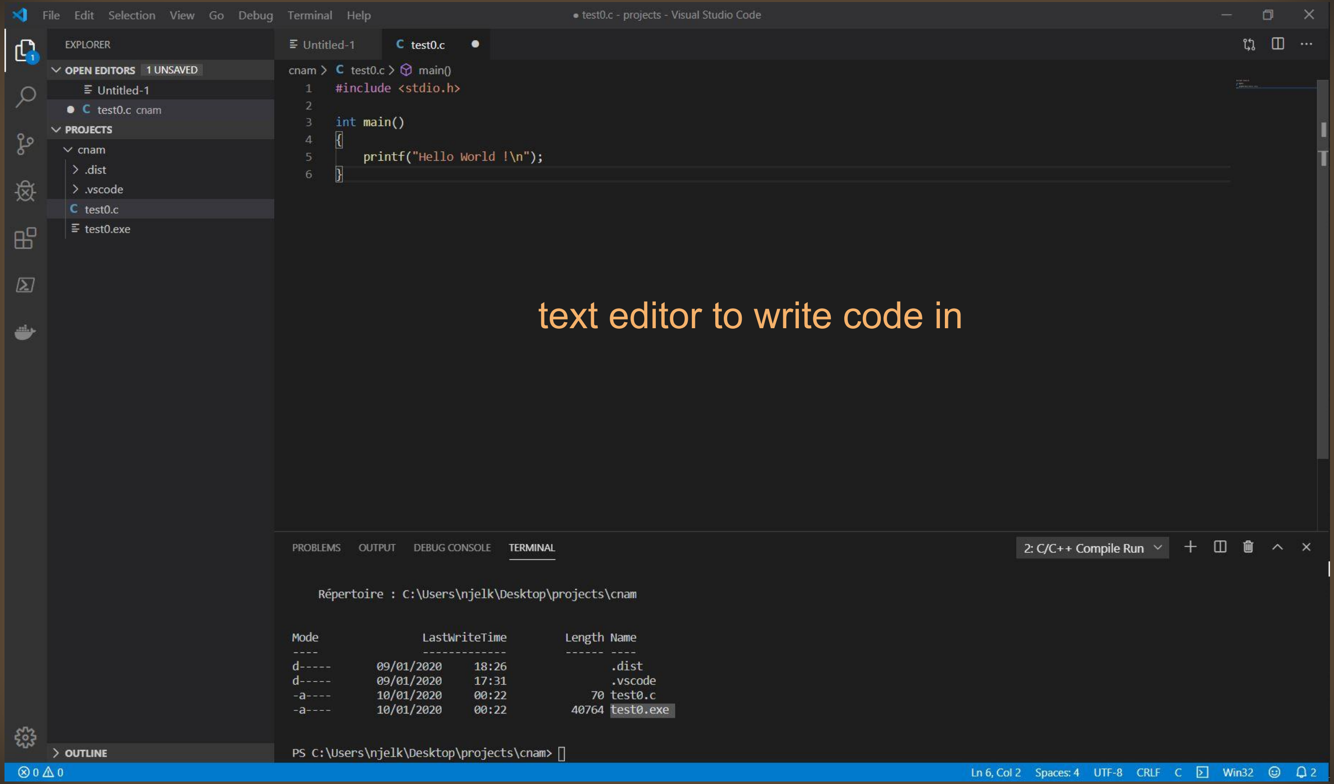
```
    Répertoire : C:\Users\njelk\Desktop\projects\cnam


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       09/01/2020     18:26                .dist
d-----       09/01/2020     17:31                .vscode
-a----       10/01/2020     00:22             70 test0.c
-a----       10/01/2020     00:22          40764 test0.exe


PS C:\Users\njelk\Desktop\projects\cnam> 
```

Visual Code Studio

- runs under Windows, Linux and Mac OS
- Most used IDE under Windows
- Opensource (= free!)



File Edit Selection View Go Debug Terminal Help
● test0.c - projects - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
Untitled-1
test0.c cnam
PROJECTS
cnam
.dist
.vscode
test0.c
test0.exe
OUTLINE
Untitled-1
test0.c
cnam > test0.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: C/C++ Compile Run
Ln 6, Col 2 Spaces: 4 UTF-8 CRLF C Win32

```c
#include <stdio.h>

int main()
{
    printf("Hello World !\n");
}
```

text editor to write code in

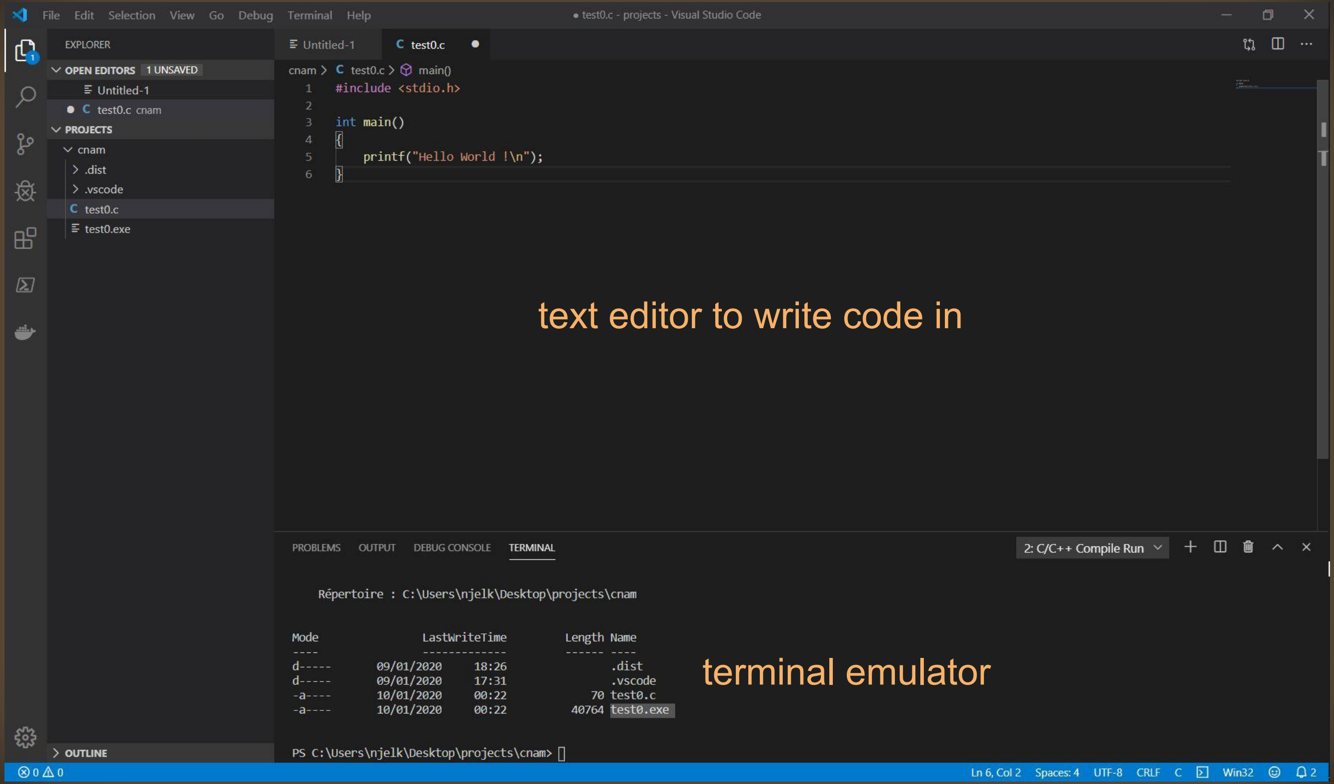
```
    Répertoire : C:\Users\njelk\Desktop\projects\cnam


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       09/01/2020     18:26                .dist
d-----       09/01/2020     17:31                .vscode
-a----       10/01/2020     00:22             70 test0.c
-a----       10/01/2020     00:22          40764 test0.exe


PS C:\Users\njelk\Desktop\projects\cnam>
```

Visual Code Studio

- runs under Windows, Linux and Mac OS
- Most used IDE under Windows
- Opensource (= free!)



File Edit Selection View Go Debug Terminal Help
● test0.c - projects - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
Untitled-1
● test0.c cnam
PROJECTS
cnam
.dist
.vscode
test0.c
test0.exe
OUTLINE
Untitled-1
test0.c
cnam > test0.c > main()
text editor to write code in
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: C/C++ Compile Run
terminal emulator
⊗ 0 ⚠ 0
Ln 6, Col 2 Spaces: 4 UTF-8 CRLF C Win32 🔔 2

```c
#include <stdio.h>

int main()
{
    printf("Hello World !\n");
}
```

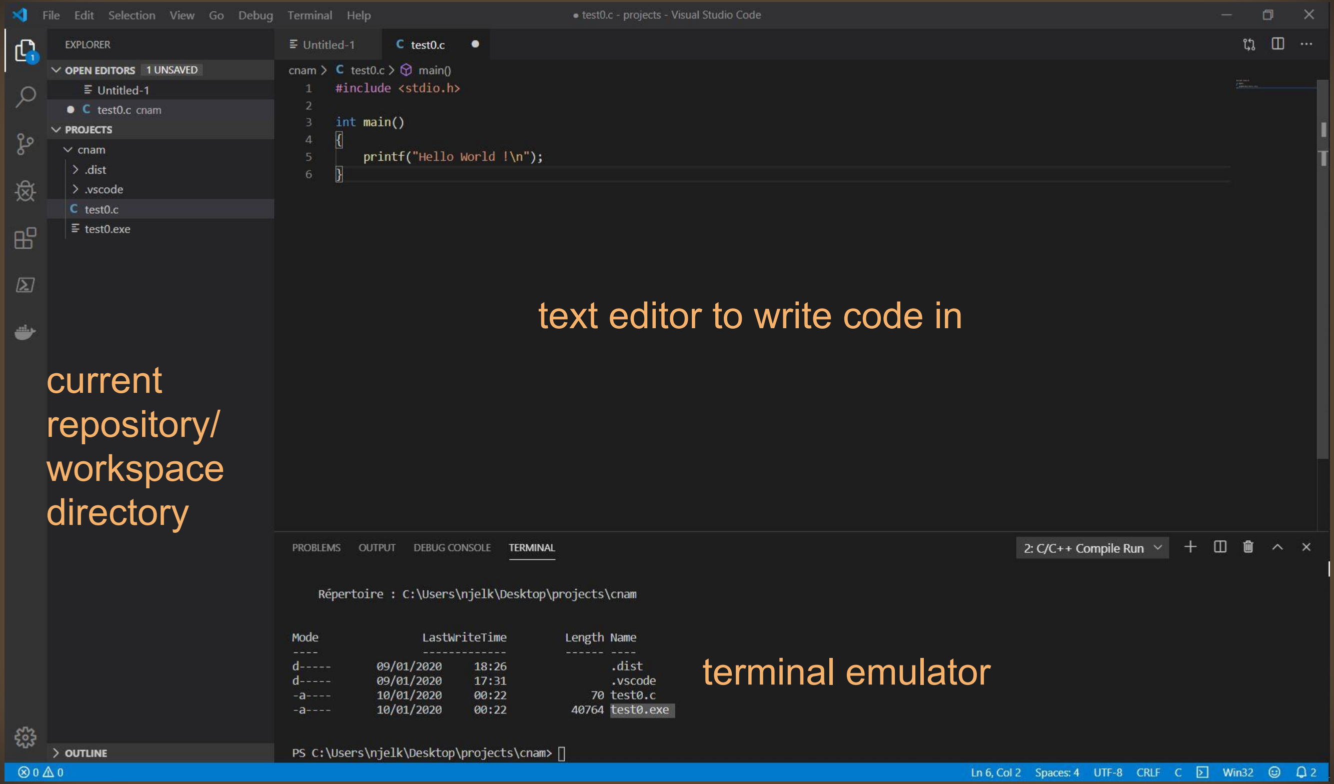
```
    Répertoire : C:\Users\njelk\Desktop\projects\cnam


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       09/01/2020     18:26                .dist
d-----       09/01/2020     17:31                .vscode
-a----       10/01/2020     00:22             70 test0.c
-a----       10/01/2020     00:22          40764 test0.exe


PS C:\Users\njelk\Desktop\projects\cnam> 
```

Visual Code Studio

- runs under Windows, Linux and Mac OS
- Most used IDE under Windows
- Opensource (= free!)


File Edit Selection View Go Debug Terminal Help
● test0.c - projects - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
Untitled-1
test0.c cnam
PROJECTS
cnam
.dist
.vscode
test0.c
test0.exe
OUTLINE
Untitled-1
test0.c
cnam > test0.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: C/C++ Compile Run
Ln 6, Col 2 Spaces: 4 UTF-8 CRLF C Win32

```c
#include <stdio.h>

int main()
{
    printf("Hello World !\n");
}
```

```
    Répertoire : C:\Users\njelk\Desktop\projects\cnam


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       09/01/2020     18:26                .dist
d-----       09/01/2020     17:31                .vscode
-a----       10/01/2020     00:22             70 test0.c
-a----       10/01/2020     00:22          40764 test0.exe


PS C:\Users\njelk\Desktop\projects\cnam>
```

text editor to write code in

current repository/ workspace directory

terminal emulator

Denis Ritchie and Ken Thompson

Massachusetts
Institute of
Technology

Imperative and procedural

Statically typed *(everything as to be stated, types, returns etc)*

Weakly typed *(not memory safe)*

primary goal is high performance

Compilers:

GCC (GNU compiler collection)

Clang / LLVM (low level virtual machine)

MSVC (microsoft visual C++)

Applications:

> Software design
> Operating System design
> Graphical render
> Compiler design
> Compute mathematical prompts fast
> Embedded system

Use

>Systems that require fast and direct

>access to hardware

> Systems with limited resources (like memory)

> Systems where performance is the most important attribute

Procedural programming

> Each task that has to be computed is prompted logical step by logical step

> Data and procedures are treated has two distinct entities

Procedural programming

Writing down a list of instruction to tell the computer what it should do step by step

Object oriented programming

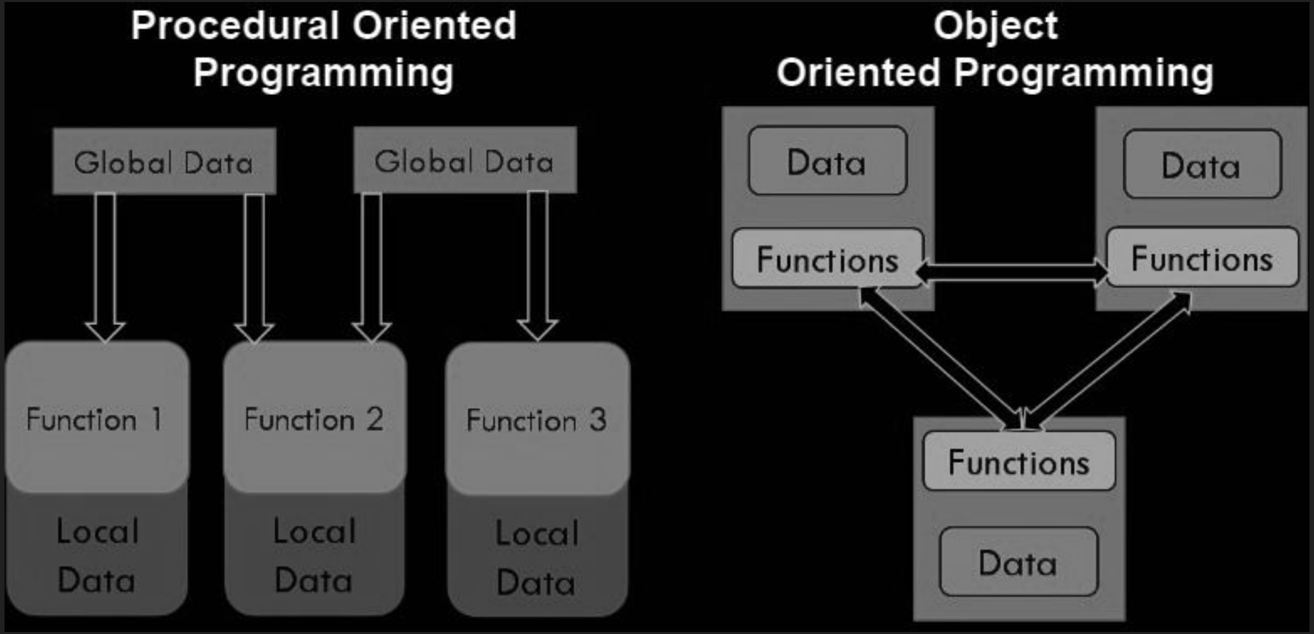
>works with a collection of objects, working in tandem with each other to solve a particular problem at hand

> OOP mimics real world, less abstract

> Every Object is self sustainable

Procedural programming

Writing down a list of instruction to tell the computer what it should do step by step


Procedural Oriented Programming
Global Data
Global Data
Function 1
Local Data
Function 2
Local Data
Function 3
Local Data
Object Oriented Programming
Data
Functions
Data
Functions
Functions
Data