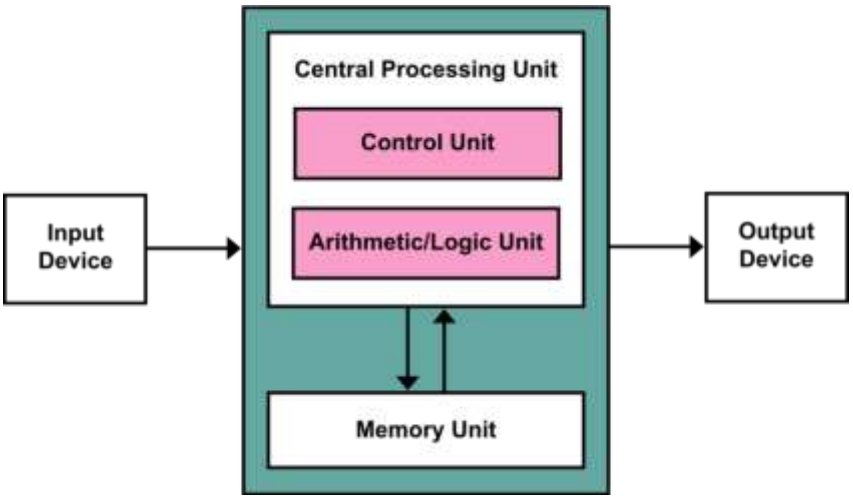
What is a C program?

- It's Just a Text File
- Understandable format language
- What we refer to as High-Level language OR Human readable language

von Neumann diagram



Not accurate computer diagram, but is a good enough model for computers

1. Memory
2. Control Unit
3. Arithmetic Logic Unit
4. Input / Output

Memory

- **Store data** and **instructions** for program execution

1. **What is Instructions**
    a. Instructions are basic expression of the basic operations
    b. E.g. + , - , *, /
    c. " { } " -> body of a function
    d. " ( ) " -> parenthesis parameter
    e. E.g. $x + y \rightarrow z$
        i. Operation = '+'
        ii. Source operands = x, y
        iii. Destination operands = z
    f. Op-code -> representation of operator in bits
        i. Example "+" is represent as 01100 in bits
2. **RAM**
    a. Basic cycle of CPU
        i. Fetch -> Decode -> Execute -> Write Back -> Fetch

ii. Fetch
    1. Fetch from memory to control unit
iii. Decode
    1. Breaking down the information to op-code and operands
iv. Execute
    1. Processed the information in the ALU for result
v. Write Back
    1. Write Back the result to memory and return the cycle

3. **What is ASCII?**
    - Idea to represent each character with 8 bit (0-255)
    - 0-127 (Readable Character)
    - 128-255 (Unreadable Character)

Directives

- Lines in a C program that begins with a '#' symbol
    - #include <stdio.h>
        - $\text{Command} \rightarrow \text{'include'}$
        - $\text{Arguments} \rightarrow \text{' stdio.h'}$
    - <stdio.h> is to be found in the library-include in the user directories
    - "stdio.h" is in the user-defined directories, default is in the same directory as the C file
- During Pre-processing
    - Read libraries and remove comments

Variable

- Objects that represent real world thing/idea (simulation / representation)
- Reality : computers only read one or zero
- Assign values to variable, so that the value mean something in the program

- If a program is successfully compiled and linked, each DECLARED variable is assigned a specific memory location during Runtime

"string literally" is whatever in between the "" with an un-escaped character

Identifies

- Name
- Variables and function have names
- Only these alphabets can be used
    - Digits (0-9)
    - Letters (a-z, A-Z)
    - Underscore (_)

A statement is a Line ends with ;

e.g. compiled statement:

```
int main(){
}

if (expression){
}
```

else is paired with the closest **un-paired if**

```
if() //#1
        if() //#2

else{} //paired with #2
```

Syntax:

```
int I_AM_a_VAR;
int _IAMOK_2;
int _1234_OK;

int I-am-Error;        //compile_error, '-' is an operator
int 2_AsaC_ERROR; //compile_error, '2' is a number at the front
```

Declaration/un-Initialize:

```
int a;
float b;
```

Initialization:

```
int a = 10;
float b = 10.0f;
```

Assignment/Assignment:

```
a = 20;
b = 20.0f;
```

Declaration + Definition

in .h file:

```
extern int a;                          // Declaration
extern static float b;

struct A
{
        static int iampublicdefault;                // Users have access
```

```
private:
	static int iamprivate;			// Users have no access
};

class B
{
	static int iamprivatedefault;	// Users have no access
public:
	static int iampublic;			// Users have access
};
```

in .cpp file

```
int a = 10;					// Definition
static float b = 10.0f;
static int A::iampublicdefault = 10;
static int A::iamprivate = 10;
static int B::iamprivatedefault = 10;
static int B::iampublic = 10;
```