Embedded C

$\boxed{1}$ C Identifiers

↳ var Name, fun Name

* Lower case → var   * upper case → const

* C Typed language → Types → char
  ↳ Int
  ↳ floating

char → 8 bit

Int → 16 - 32 - 64

float → 32   double 64

⟹ Boolean [Bit var] → 0
  ↳ 1

$\boxed{2}$ Type specifiers

1- signed   2- unsigned

2

↳ can apply to char & Int

$\boxed{1}$ signed → represent negative values

↳ MSB → sign-bit

$\boxed{2}$ unsigned → represent positive

↳ MSB → part of numerical val

* char → unsigned by default
* Int → signed " "

$\boxed{3}$ Type qualifier → const
→ Volatile

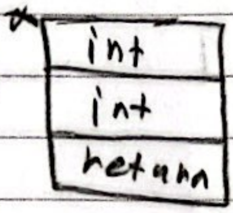
* const → var can't be changed
→ will be stored in ROM

* Volatile → refer to variables may change by hardware

$\boxed{4}$ functions
→ when program call function → executio
Jump to the address associated with fun name

* function prototype → way to let compiler know about fun before it's used
* advantage of using function in other files

→ int sum (int, int) →



* must be on return value [of any type]

$\boxed{5}$ Loops
* else matched with nearest unmatech If

/ /

$\boxed{6}$ Logical operators

↳ C permits short circuiting

↳ يعني شلاً لو داخل condition ولقى أول جزء فيها 0 ومعمول معاه أكثر من جزء تانية مش هيعمل روتنه anding by logic = 0

Bit wise

* BIT wise operators
  ↳ logical operator
  ↳ shift operator

&	And
|	or
^	xor (for toggling)
~	Invert

$\boxed{7}$ shift operator
  ↳ shift Right >>
  ↳ shift Left <<

x = 0b 0000 0110 x >> 1
x = 0b 0000 0011

x << 1 → x = 0b 0000 1100

* Set Bit (1 << bit number) | bit *

* clear Bit ~(1 << bit-number) & bit

C Pre Processor

- → any command begins with #
- → Processed during preprocessing
- → doesn't understand c lang
  - ← علشان كدا ممكن تسبب أخطاء كثير
- → يستخدم / لو هكتب كذا سطر تحت بعض
- → Common Errors
  - → ; لا نستخدمها في نهاية ال macro
  - int x = 5;;; ← لو استخدمتها سي هيحصل compiler error
  - → white spaces are very important to preprocessor خد بالك منهم → will cause error
  - → don't use comments

1. # Include
   - → to include files
   - → تقول ال preprocessor انه يبدل ال directive اللي مكتوب بمحتوى ال file ده

< > → system Library

" " → user Library

$\boxed{2}$ #define → used for text replacement

$$\begin{tikzcd} & \text{\#define} \arrow[dl] \arrow[dr] & \\ \text{object like macro} & & \text{function like macro} \end{tikzcd}$$

* في الـ fun العادية ~~[illegible]~~ ~~call~~ بتاخد مكان في الـ memory مرة واحدة
وبنادي عليها call لما بيرجع بيرد للعنوان بتاعها

* أما في الـ fun like macro ← كل مرة بعمل فيها call بياخد مكان في الـ memory

can be used to define array & switch case

* const qualifier

#define X 3

↳ compile time const
during compilation
↳ can be changed in preprocess

const int max=30 → store in Rom
→ can't be changed
can't be use to define array & switch case

* Macros are faster than fun
because ⇒ لأنه ما بيعملش call للstack
ولا بيعمل push / pop

—— // —— // —— // ——

Scope & ~~Extern~~ Extent

[1] Scope

↳ جزء البرنامج اللي الـ var بتاعي بيكون معرّف فيه

Local   external

[2] Extent
↳ describe Life time for variable [when memory is allocated
and when // // Released

Automatic   Static

Local Scope / Auto Extent

local scope ⇒ fun أي var متعرف داخل
و هيكون local
by default

→ function Arguments have local scope

→ local variable has Auto extent

لـ الـ lifetime بتاعهم بينتهي بانتهاء الدالة وبيبدأ أول ما يتعرفوا
called Auto var

⇒ memory ↑ managed by compiler

unintialized local var ⇒ undefined value

So it's a good practice ←
to Initialize var when it's declared

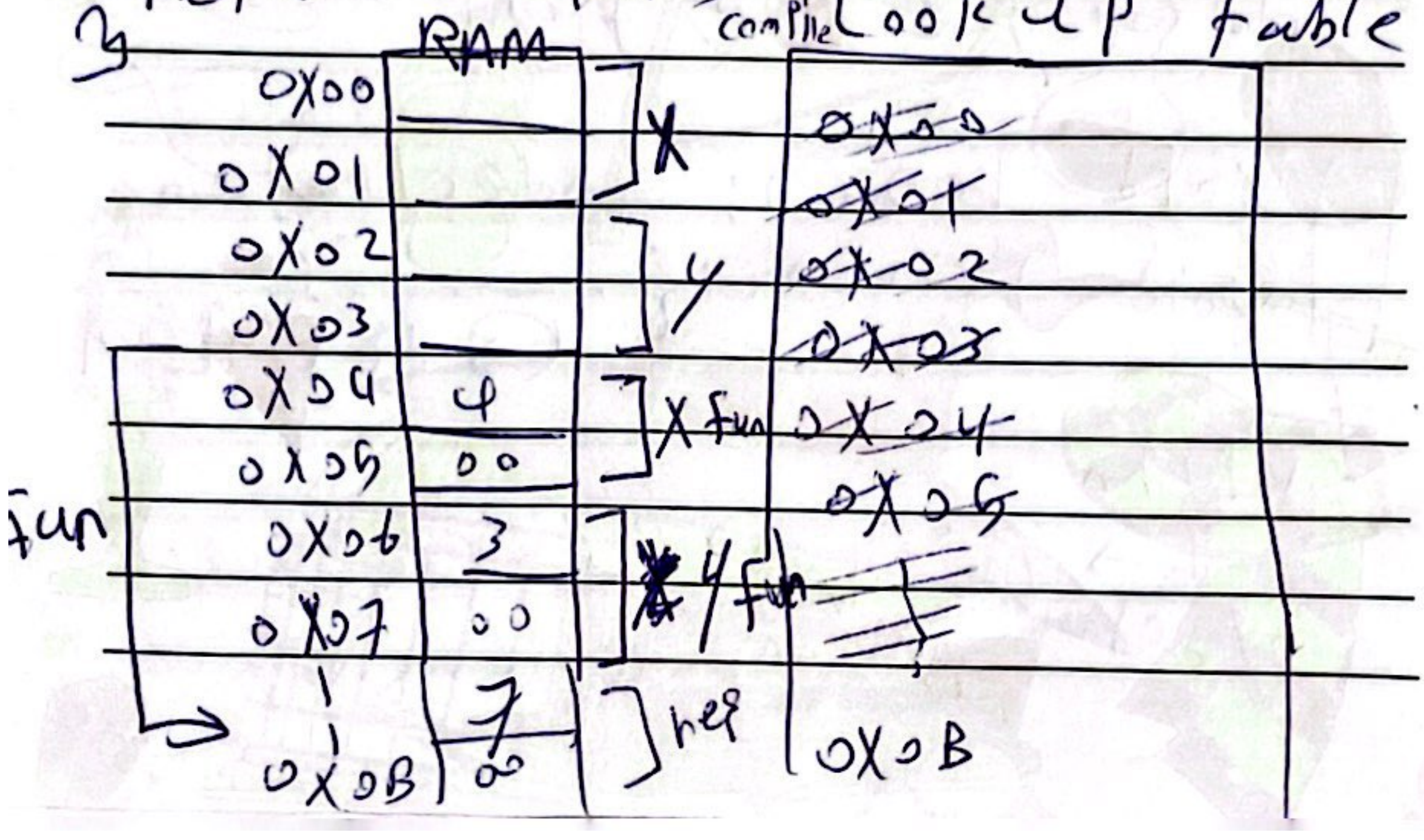
⇐ في نهاية الـ block ← var is destroyed
& memory recovered

دا معناه إنه إمكانات المتغير ← لكن بيه X إن X متهيأ و يتعلق فيه value

```
void Add(int,int)
main()
{ int x,y;
  ~~result = 10;~~ → error
  add(4,3);
}
void Add(int x,int y)
{
  result = x + y;
  return result;
}
```

lifetime لـ

الكومبيلر بعد ما ذفذ الدالة كاملة بيمسح الداتا بتاعتها ن ال memory, وتاخد فاضية يعتبر/ بيستغلها تاني



أكثر سؤال

Declration vs Defintion

* declration
  ↳ var exist in the phogram but no memony allocation
  int Add (int, int)

* definition ⟹ variable is defined
  ↳ has memony locatio

[illegible]
```
int Add (int x, int y)
{

}
```

extern → used to declare var

file 1 . c
```
main() intglobal;
{ Inc();
  Inc();
}
void Inc(void)
{ global++;
}
```

file2.c
```
exten. int global;
void inc2(void)
{
   global += 2;
}
```

* لو عرفت global Int بره في الملفين وخلاص
هيطلع error

* لو مشتركها بين استخدم extern أعمل في كل ملف
static ~~[illegible]~~ int x ⇒ global static

* الدالة ⟵ by default extern يعني
متعرفة في ELF, معمولها define في مكان

extern ⇒ Sharing the scope

but before using you must
declare [var/fun]

```
void  LCD();  ⟹ declration
void  LCD(){ ———— }
             ↳ defintion
```

↳ extern
  ↳ بس مش أكتبها لأنها
by default extern

[illegible] ٨ [illegible] X ، [illegible]

Static with Global variable

```
int a;
main()
{
 int x; → local var → init val garbage
        ↳ Auto extent

}
```

→ global var → كل البرنامج شايفة
 ↳ Init value zero
 ↳ scope → all program

↓ بيكون مرئي في كل فايلات البرنامج بتاعي

Static global
 ↳ بيخلي المتغير مرئي لكل الفايل اللي أنا فيه