

Lec 15

What is typedef in C?

- A typedef defines a new name for existing types and doesn't introduce a new type
- it is the (Partial storage-class modifier) compiler directive mainly use with user-defined data types (structure, union or enum)) to reduce their complexity and increase code readability and portability.
- In C Programming, we can use typedef declarations to create shorter or more meaningful names for types already defined by C (inbuilt data types like `int`, `char`, `float`) or for types that you have declared.

Syntax:

`typedef type newTypeName`

Example: `typedef unsigned int UnsignedInt`
`UnsignedInt num = 33;`

*Note

- A typedef creates synonyms or new name for existing types it does not create new type.

الستاريك أو typedef هو مصطلح يختلف عن الـ keyword ونعني به
وأحياناً يُدّعى كـ alias وإن الـ keyword يُعرف بـ token على الأجهزة من
يعتمد على 16bit MC ويرت على 32bit MC
هذا أكمل أو أخطاء

* يستخدم أيضًا لـ cooperation لـ alias تـابـيـب موجود
لـ alias داـرـيـاـ لـ اختصار أو بـ دـيـلـ لـ alias تـابـيـب
Ex- `typedef int i;` int i كـ اـنـوـاـنـوـدـ بـ الـ كـ لـ يـبـ مـوـجـوـدـ
وـ مـكـنـ الـ كـ لـ يـبـ بـ الـ كـ لـ يـبـ مـوـجـوـدـ

Code Portability:-

- We know the size of the int is not specified by C standard.
- The C standard only explains the minimum size of the integer that is 16 bit.
- If you working on a platform for where you want the size of int is always 32 bit, so in that situation typedef is useful.

Portability في البرمجة *

وـ مـكـنـ بـ الـ كـ لـ يـبـ كـ وـ دـعـ فـ الـ كـ لـ يـبـ الـ كـ لـ يـبـ
الـ كـ لـ يـبـ دـاـرـيـاـ لـ اـنـ سـوـقـ حـالـةـ الـ كـ لـ يـبـ الـ كـ لـ يـبـ
وـ بـ عـنـ اـسـاسـ اـنـ اـنـوـاـنـوـدـ الـ كـ لـ يـبـ الـ كـ لـ يـبـ
زـفـ دـيـ بـ نـفـحـ الـ كـ لـ يـبـ الـ كـ لـ يـبـ
typedef لـ long MC بلـ اـنـ اـنـوـاـنـوـدـ الـ كـ لـ يـبـ الـ كـ لـ يـبـ
(bits) او data type

Platform.Types.h

```

#ifndef _PLATFORM_TYPES_H // file guard
#define _PLATFORM_TYPES_H Lec 8

#define CPU_TYPE_8 8
#define CPU_TYPE_16 16
#define CPU_TYPE_32 32
#define CPU_TYPE

#if (CPU_TYPE == CPU_TYPE_32)
typedef unsigned char boolean;
typedef unsigned char uint8; } from data sheet
                                number of bits
#endif (CPU_TYPE == CPU_TYPE_16)
typedef ... }

#elif (CPU_TYPE == CPU_TYPE_8)
typedef ... }

#endif // if guard
#endif // ifndef guard

```

قبل صياغة جزء في عرض Readability و `DS C & AND DS CALL`

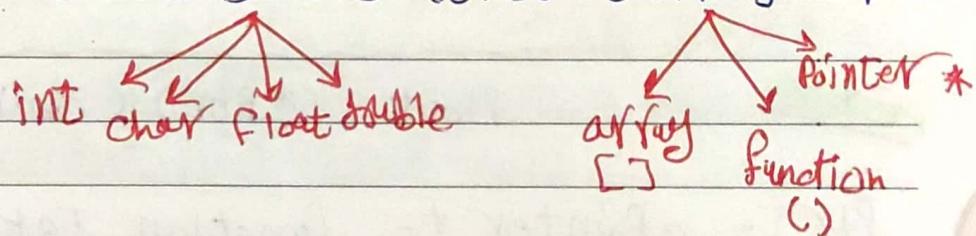
Complex Pointer / complex Declaration

وهي عبارة عن مطابقات مجمعات الابعاد والفرادة فنسان المترتبة لها (ونظرًا لأنها تعرف بعملية التحويل عليها فنسان نعرف أن المطابقات التي تتبعها يسيطر على

Ex:- `char * (*(*arr[])[10])(())[];`

علاقة رأسية ترتيب المترتبة فنسان فرق ما ملح لزم لفرق سوء توكن (إذن) `arr` `*` `function`

complex Declaration = Basic types + Derived types

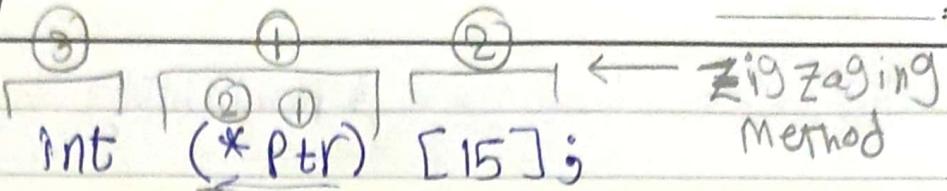


precedence المطابقين تدرك لفظاً يدهم أول طرفة نظر بـ Priority بالترتيب

[1] () , [] → highest Priority
↳ Left to Right

[2] * , Identifier
↳ Right to Left

[3] Data types

Example 1

`ptr` is a pointer to an array of 15 elements of `int`.

→ دالة الـ `ptr` تأخذ مدخلات على شكل طرفة و تخرج على شكل سpiral outwards

`int (*ptr)[15];` Anti clockwise (soAC)

Loyarray `(*ptr)`: size of this is 15
int size of each is 15

Ex 2

`float (*ptr)(char);`

`ptr` is a pointer to function takes a `char` argument and return `float`.

Ex 3

`char * (*ptr)(int, float*);`

`ptr` is a pointer to function that takes (two arguments ^(IA) integer and ^(SA) pointer to `float`) and return pointer to `char`.

EX4

```
Void (*ptr)( int(*)[2], int(*)(void));
```

ptr is a pointer to function takes two argument
 (first argument is a pointer to an array of two elements
 of integer)

(Second argument is a pointer to function takes
 no argument and return integer
 and returning void.

EX5

```
Void (*(*a[]))();
```

a is array of pointers to a function taking no
 argument and returning a pointer to a function taking
 int variable argument and return void

EX6

```
Int (*(*ptr)[6])();
```

ptr is a pointer
 to an array of 6 elements of pointers points
 to function taking no argument and return int.

EX7

```
char *(*(*(*a[])[12]))[];
```

2D array

a is an array of array of size 12 of pointer to pointer
 to function taking no argument and return pointer
 to an array of pointer to char

Ex 8

Void (*func())(Void (*)(int*, void**), int(*)
(void**, int*))

- Fun is function taking no element and return pointer to function taking two argument
- first argument
is a pointer to function taking two element or argument
 - .first argument is a pointer to integer.
 - second argument is a pointer to pointer void. and return void.
- second argument
is a pointer to function taking two argument
 - .first argument is a pointer to pointer void
 - .second argument is a pointer to integer and return integer.
 - and return void

اراء احسن ال اراء

`uint32 (*Function Ptr) (void);`

جذب:-

دعا عن اداء من \leftarrow Function to Pointer

`typedef uint32 (*functionptr) (void);`

function

فما زلت اذكر

عند زيارتي لجامعة الامارات ورثيتي كابحه العراده ورثيتي
وأنا أعلم أن variable داخل الـ function ينتمي إلى \leftarrow Pointer to
function

وـ complex pointer مثل typedef \leftarrow function

Union :-

-The C Language provides the flexibility to the programmer to create their own data types as per their requirement.

→ These data type is called user defined data type or non primitive data type.

What is Union:-

- A union is used to store different data types in same memory location.

- In union we can define many members as per requirement but here we need to remember that stored value is shared by all members.

- it means a union provides an efficient way of using the same memory location but care fully.

٨. Union اتحاد

هو باردة لبيان نوع البيانات التي تمثلها في الذاكرة كل مخزن حاجز يأخذ
نوع البيانات مخزون في الميموري طبقاً لبيانات الميموري
الكل مخزن مع بعض سطر كل المخزن ليغير المخزن تكون عبارة
ومن هنا نستخدم المخزن الذي اعد حاجزه كذا.

Syntax :-

Union (name) → optional

* member that the union can contain

3:

NOTE

- A union type declaration is a template only.
- There is no memory reserved for the union until a variable is declared.

يختلف struct وال联合 في طريقة التعامل بالضبط
الهدف من联合 هو إداره الذاكرة وتخزين
(Passing by value) لبيانات الـ struct

Enum → Enumeration.

- is user defined data type and it consists set of named constant integer.

Why we use enum?

- 1- increase the code readability.
- 2- Easy to debug the code as compares to symbolic constant macro.

* لـ enum بمقدمة عنوان يُعرف فيه الأبيات.

* طلب التوكيل في وقت الحاجة باستخدام

* خالص يعني أنه أولاديهم، فيه يطلب أمن صدر ويعينه البعد واحد وهو هنا
بعنوان مكتن لـ enum العلامة بتات - أولاديهم، أو قيمه لأى عنصر وأخليه
بالعلامة للـ enum أداء لدوره أو اللـ بعد غيره واحذر على العلامة دعوه هنا.

* التوكيل دعوه

Syntax:-

enum Enumerating_Tag { Enumeration_List } ;

example:-

enum Day

Sat, // = 0

Sun, // = 1

Mon, // = 2

Tue, // = 3

Wed, // = 4

Thu, // = 5

Fri, // = 6

} ;

* مسماً اختياري مسماً ملائمه لـ Day

* والأد لو عرفت ما هو يبحث جزءه

* في المعياري

Automatic

The most important Property of Enum?

- It follows the scope rule. Main() begins in Main() and ends in Main()
- The compiler automatically assigns the value to its Member constant

Ex:-

enum Day day_1 = tue ✓ // day_1 = 3



العنصر يأخذ قيمة المتغير

enumerator element , int number → warning
Not error

انواع enum ادوات معرفة وادوات

للحاسوب برمجيات

enum Days

Mon = 1,

tue, ← = 2

Wed, ← = 3

Thu = 4,

Fri, ← = 5

Sat = 6,

Sun ← = 7

انواع enum ادوات معرفة وادوات

القليل برمجيات

} ;

Using enum with for loop :-

enum year { Jan, Feb, Mar, APR, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec };

int main()

enum year mon = Jan;

for (mon = Jan; mon <= Dec; mon++)

{

printf ("%i\n", mon);

}

outPut

0

1

2

3

4

5

6

Using enum with switch case & function "Parameter"?

enum month *

11

void print_month_name (enum year mon)

{

switch (mon) {

case Jan : printf ("January"); break;

case Feb : printf ("February"); break;

default : printf ("INVALID!!"); break;

}

int main()

print_month_name (Jan); //January

print_month_name (Feb); //February

print_month_name (APR); //INVALID!!

}

Value ليمثل enum نفس الـ two element ممكن بعدهما حمل لوكانو ورا يعني

main أو Main لا ينفع لـ enum الـ rule scope global \hookrightarrow local

element ينبع اعرف الاسرار لوكانه two enum re declaration \hookrightarrow effort main أو main أو main

Dynamic Memory Allocation

malloc function:-

١- هرم مساحة عن فاتلشن يستخدمها اثنان لجزءها في الميموري
وخصوصاً خسلشن ال heap

٢- يتأخر عدد ال bytes المطلوب ويتجزأ void ←
ويتأتي في ادراهم explicit casting اعلاه نوع البروينتر الميتحزن
فيه العنوان.

٣- لازم اعمل Validate على العنوان الراجع دا لوب NULL بما
الفاتلشن دعه رسم لجزءها في heap لورجع عنوان
ليما تمام

`int * PTR = NULL;`

`PTR = (int *) malloc (size)` و

٤- size يبقى كما حسب اذ ما ذكر حجز كام بایت

يعنى ممكن يبقا كذا (`malloc (5 * sizeof(int))`)

لأن دو و هي حجز دين ال sizeof يرجع 4 وبالنطري
 $5 * 4 = 20$

٥- البوينتر ال مستقبل فيه الارجع يفضل تكون

`int * const PTR = (int *) malloc (5 * sizeof(int))` و

٦- عيادة افتر العامل مع البوينتر

`* (PTR + 1) = 0x11;`

`PTR++` → error

calloc function :-

- * تقريبا نفس كل حالات malloc
- 1- لما يجرب اماكن بها بذى قيم الماكنات في_malloc
- 2- تجربة بحث قيم عوائده
- 3- هناك اثنان من parameters

`int *ptr = (int *) calloc(5, sizeof(int));`

٣- وائل يرجح الالكت بخلاف الـ malloc

realloc function :-

- 1- يستخدم لاعطان البر أو اصغر حجم الـ bytes للجزء

2- لما يكتب الـ size يتحمل copy الماكن القديمة الجديدة ويزود بعدها

3- لو الحجم اصغر يعمد الى الجدير ثم يفرهن القديم بـ `free()` ثم يأخذ الجديد

4- دخانه `two parameter ← void pointer` ← الـ pointer وينتزع بـ `address`

`realloc(pointer, size)`

٤- عينه يرجع `void pointer`

`ptr = (int *) realloc(ptr, (5 * sizeof(int)));`

٥- `NULL` يرجع realloc او `ptr` لو `ptr`

free function :

العنوان الذي أكتبه هنا deallocate لـ free مسحه

لـ free مسحه لـ free مسحه

لـ free مسحه لـ free مسحه

Syntax: free(nameof Pointer);

→ Pointer

لو الـ Pointer داير NULL او فانشن من هتجل illegal.

لو الـ Pointer دايرش uninitialized او رعن حاجه في الميموري او undefined behavior.

deallocate داير Pointer بـ free لـ free مسحه لـ free مسحه لـ free مسحه

فـ free مسحه لـ free مسحه لـ free مسحه

استخدام الـ Array مع الـ Dynamic memory بين أنواعها:

يتفع العامول معاه Pointer لـ 1D array التي يستخدم Pointer [] Subscription operator []

Pointer to Pointer او double Pointer لـ 2D Array يعملها باستخراج []

Steps to creating 2D dynamic array in c using Pointer to Pointer

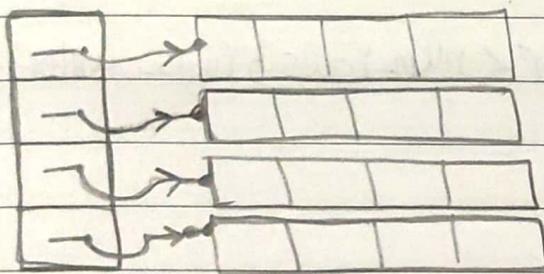
1- Create Pointer to Pointer:-

`int **ptr = NULL` ;

2- Determine size of Rows :-

`ptr = (int **) malloc(num_rows * sizeof(int*))` ;
 from user
 عدد المعرف

* هنا نحن لو عد المعرف 5 يعني اهابن يقوله روح اجزء خالصوري
 5 اما كل عنصر المعرف كل مكان عبارة عن مساحة من الذاكرة
 والا address المخرج من malloc يرجع الى اول عنوان بيسأول
 عنوان آخر.



* ساجزى الرسم ودى كما

3- Determine size of columns :-

`for(int cnt = 0 ; cnt < num_rows ; cnt++)`

{

`ptr[cnt] = (int*) malloc(num_of_col * sizeof(int)) ;`

}

4- Access 2D Array using nested 2 for loop :-

```
int rows_counter=0, cols_counter=0;
for (rows_counter=0; rows_counter<num_rows; rows_counter++)
{
    for (cols_counter=0; cols_counter<num_cols; cols_counter++)
    {
        // operation here
    }
}
```

$\text{ptr}[rows_counter][cols_counter] = \text{ox33};$

}

}

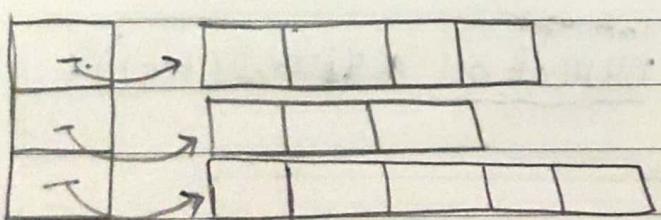
↑ **أذاجي //**
scanf سواد كان
و printf أو.
operation

5- How to free :-

```
for (rows_counter=0; rows_counter<num_rows; rows_counter++)
{
    free(ptr[rows_counter]);
}
```

free(ptr);

مهم اعمل صفر 2D array بس عدد المصفوفات مختلف في كل صف



from user

```
ptr[0] = malloc(col1 * sizeof(int));
ptr[1] = malloc(col2 * sizeof(int));
ptr[2] = malloc(col3 * sizeof(int));
```