lesson 33 pointer

إذا كنت تجلس فى المنزل و طلبت من أخيك الصغير أن يحضر لك التفاحة التى على الطاولة، سيذهب أخوك ليحضر لك التفاحة، و لكن ماذا لو لم يجد تفاحة على الطاولة ؟ بالتأكيد لن يحضر لك شئ لأنك طلبت التفاحة.
و لكن لو قمت بتغيير طلبك بحيث طلبت منه أن يحضر لك ما على الطاولة أيا كان، ربما لا يجد تفاحة و يجد برتقالة أو ورقة أو هاتفك المحمول أو أيا كان على هذه الطاولة.
ففى المرة الأولى أنت طلبت منه شئ معين إذا لم يجده لن يحضر شئ و فى المرة الثانية أن أعطيته مرجع أو reference و هو (الطاولة) بحيث أن أى شئ موجود على هذه الطاولة سوف يحضره بغض النظر عن ماهيته.

فى هذا الدرس سنتعلم عن ال **pointers**، و هى التمثيل البرمجى للمثال الذى ذكرناه . ال pointers هى عبارة عن data type يستطيع أن يحمل عناوين فقط (adresses) .

العنوان أو Address :
إذا كان لدينا **variable** من نوع int و ليكن x، إذا أردنا عنوانه فى الذاكرة سنستخدم علامة **&**

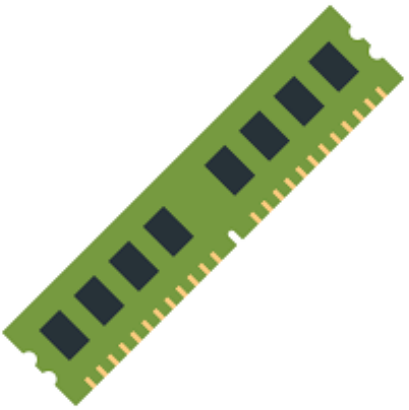
```
int x = 5;
```

لطباعة العنوان نستخدم %p

```
printf("%p \n", &x);
```

هل لاحظت استخدام علامة & فى جملة ال printf مع أننا معتادون على عدم استخدامها، بل نستخدمها فى أمر scanf فقط !
الفرق هو أننا هنا نتعامل مع العنوان، نحن لا نقوم بطباعة قيمة ال x التى تساوى 5 ،
نحن نقوم بطباعة عنوان ال x الذى قد يكون شيئا مثل ذلك : 0x7ffe35ce33ec أو 0x7fffce1caf8c أو 0x7ffebf84ac5c ... و غيرها من الإحتمالات المتغيرة فى كل مرة نقوم بعمل run للبرنامج .
لماذا العنوان يتغير فى كل مرة نقوم بعمل run للبرنامج ؟
لنفهم ذلك علينا فهم ال memory و كيفية عملها .

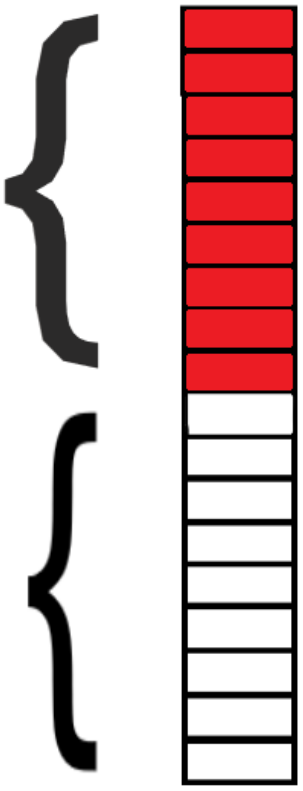
الذاكرة memory :



ذاكرة ال ram (Random Access Memory) و التى تعنى ذاكرة الوصول العشوائى، و هى المكان الذى يتم فيه تخزين البيانات التى نحجزها فى البرنامج مثل ال int و ال char و ال arrays و هكذا ، و كلما قمنا بحجز متغير جديد يتم تخزينه فى المساحة الفارغة من ال ram أيا كان مكان و عنوان هذا المكان .

هذه المنطقة هى المساحة الممتلئة
فى الذاكرة نتيجة تخزين بيانات
البرامج المختلفة فى الكمبيوتر
مثل ال int , char , float و هكذا

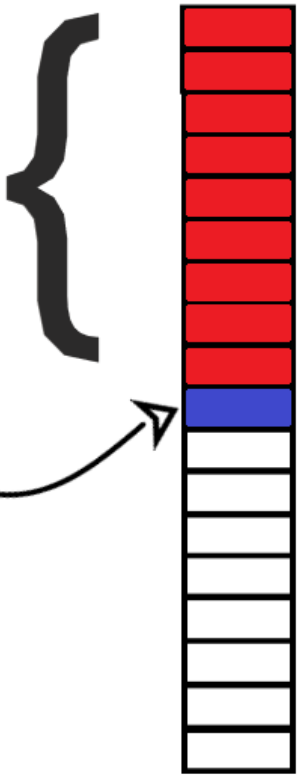
هذه المنطقة هى المساحة الفارغة فى
الذاكرة و التى يمكنها تخزين أى
بيانات جديدة عند تشغيل أى برنامج



Memory

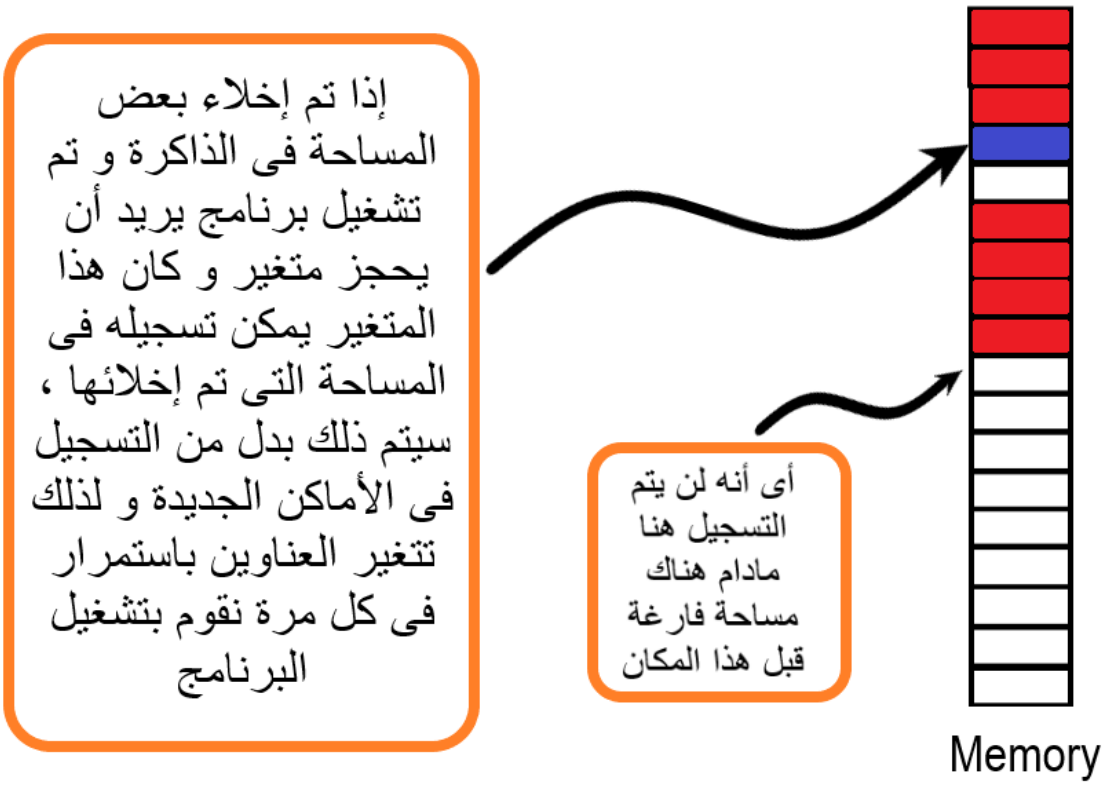
هذه المنطقة هى المساحة الممتلئة
فى الذاكرة نتيجة تخزين بيانات
البرامج المختلفة فى الكمبيوتر
مثل ال int , char ,float و هكذا

عند إنشاء متغير جديد يبدأ
البرنامج فى حجز مساحة له
فى الذاكرة فى أول مكان
فارغ و متاح التخزين فيه

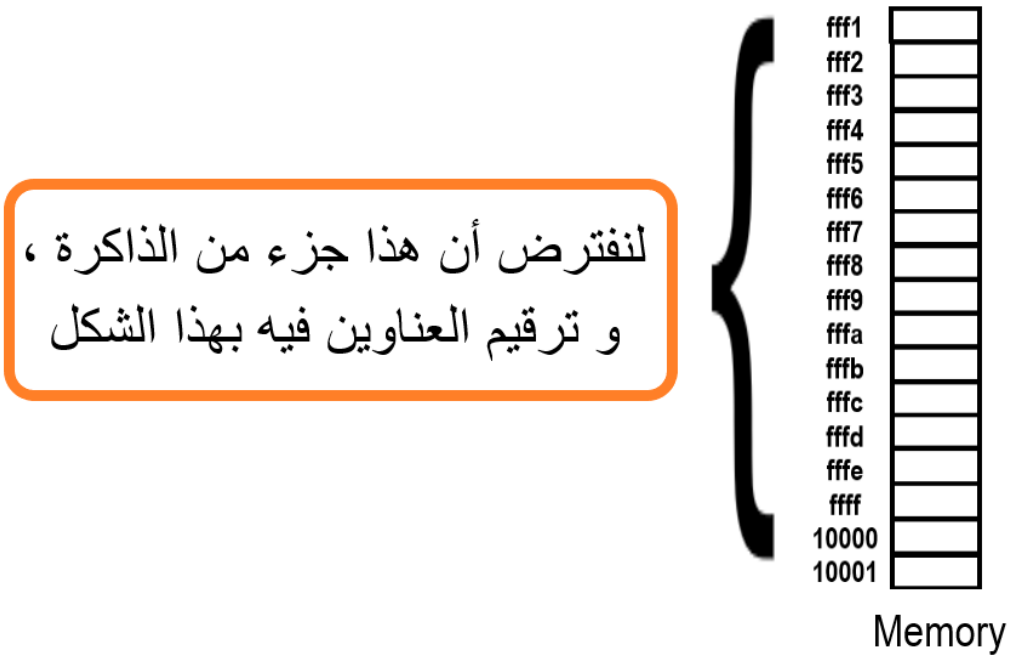


Memory

و لكن ليس بالضرورة أن يتم تخزين البيانات فى المساحة الجديدة فى الmemory و لكن كلما كانت هناك مساحة فارغة يتم ملؤها بالبيانات المراد تخزينها

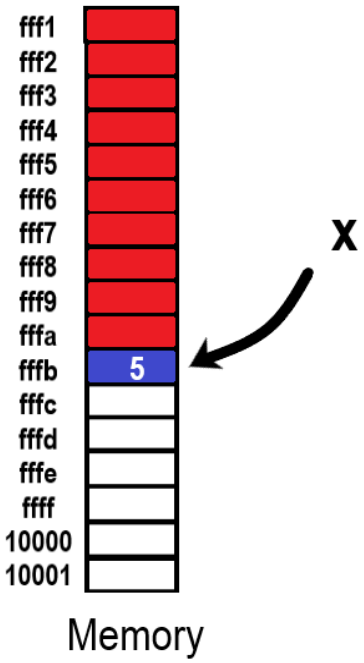


فمن اسمها (ذاكرة الوصول العشوئى) أى أن يمكن تخزين البيانات في أي مكان بطريقة عشوائية و يمكن الوصول إلى هذه البيانات المخزنة فى أى عنوان (Adress) بمجرد معرفته .



عند تسجيل متغير جديد يتم حجز مكان له في الذاكرة في أقرب مكان فارغ و متاح التسجيل فيه

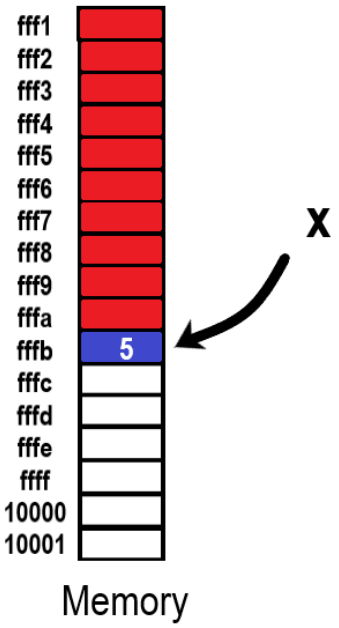
```
int x = 5;
```



في هذه الحالة يمكننا طباعة قيمة ال x بطريقتين مختلفتين، الأولى عن طريق اسم المتغير نفسه و الثانية عن طريق عنوانه :

```
int x = 5;
printf ( " %d \n",x );
printf ( " %d \n", *(&x) );
```

```
output :
        5
        5
```



لنفهم ال pointersبشكل أوضح علينا أولا أن نفرق بين شيئين :

- **عنوان** متغير ما و الذى يمكننا الحصول عليه عن طريق علامة **&**
- و **القيمة المخزنة داخل هذا العنوان** و التي نستطيع الحصول عليها عن طريق وضع علامة * قبل العنوان .

ففي المثال السابق يمكننا الوصول إلى القيمة المخزنة في x عن طريق وضع علامة * قبل عنوان x ، و يمكننا أن نحصل على عنوان x عن طريق وضع علامة & قبله .

إذا كيف نقوم بتعريف متغير من نوع pointer و ما هو الذى يمكنه أن يحمله ال pointer هل يستطيع تخزين عنوان المتغيرات أم القيم التي بداخلها ؟

ال pointers :

هي متغيرات خاصة تستخدم **لتخزين العناوين** بدلاً من القيم و يتم حجزها بهذا الشكل :

```
int *p;
```

فهكذا حجزنا pointer من نوع int أي أنه يشير إلى عنوان متغير من نوع int :

```
int x = 2;
int *p;
p = &x;
```

فهكذا ال pointer الذى اسمه p حاليا يحمل عنوان ال x و يمكن طباعته بهذا الشكل :

```
printf("%p", p);
```

و يمكن طباعة قيمة ال x عن طريق الp (pointer) بوضع علامة * قبله هكذا

```
printf("%d", *p);
```

output:

2

و يمكن تغيير قيمة ال x عن طريق الp (pointer) هكذا :

```
int x = 2;
int *p;
p = &x;
*p = 5;
printf("%d", x);
```

output:

5

لأنه في هذه الحالة لم يكترث ال p إلى أن المتغير الذى يغير قيمته هو ال x ، هو كل ما لديه العنوان الذى يستطيع تغيير ما بداخله ففي السطر

```
*p = 5;
```

يأخذ البرنامج أمر أن يذهب إلى العنوان المُخزّن داخل p و يقوم بتغير القيمة التي بداخله فبالتالى تتغير قيمة ال x تباعا لذلك .

ملحوظة يمكن أن يتم إعطاء قيمة لل pointer عند تعريفه

```
int *p = &x ;
```

و في هذه الحالة بالرغم من أن علامة ال * موجودة قبل اسم ال pinter إلا أنه يتم تسجيل العنوان فيه و ليس القيمة، هذه الحالة عند تعريفه فقط و لكن في منتصف الكود نضع ال * عندما نريد أن نستخدم القيمة المخزنة داخل العنوان، و لا نضع * إذا كنا نريد استخدام العنوان نفسه أو تخزين عنوان أخر :

```
int y = 10, x = 5;
int *p = &x;
p = &y;
*p = 20;
```

لذلك لا يجب أن نكتب

*p= &x; هذا خطأ

قد يبدو لك أن هذا المشوار الطويل ليس ضرورى و أن كان بإمكاننا أن نغير قيمة ال x مباشرة بدل من كل ذلك، و لكن هناك إستخدامات كثيرة لل pointers التي تجعلها واحدة من أهم المفاهيم و الأدوات في لغة ال C

مثال أخر لعمل المؤشر والمتغير :

```c
#include <stdio.h>
int main(  ) {
  int* p, x;
  x = 10;
  printf("Address of x: %p\n", &x);
  printf("Value of x: %d\n\n", x); // 10
  p  = &x;
  printf("Address of pointer p: %p\n", p);
  printf("Content of pointer p: %d\n\n", *p); // 10
  x = 15;
  printf("Address of pointer p: %p\n", p);
  printf("Content of pointer p: %d\n\n", *p); // 15
  *p = 25;
  printf("Address of x: %p\n", &x);
  printf("Value of x: %d\n\n", x); // 25
}
```

output:

Address of x: 004FFC38

Value of x: 10

Address of pointer p: 004FFC38

Content of pointer p: 10

Address of pointer p: 004FFC38

Content of pointer p: 15

Address of x: 004FFC38

Value of x: 25

(قم بتجربة الكود بنفسك واضغط هنا)

مثال على أهمية استخدام ال pointer :

نريد عمل برنامج فيه متغيران و المطلوب أن يقوم بتبديل قيمتهما، لكن باستخدام **function**

```
#include <stdio.h>
void fun(int x, int y) {
  int t = x;
  x = y;
  y = t;
}
int main(  ) {
  int x = 5, y = 20;
  fun(x, y);
  printf("x=%d  y=%d", x, y);
}
```

output:

```
x=5 y=20
```

خطوات تنفيذ البرنامج :

البرنامج يرسل قيم ال x و y من ال main إلى fun ليتم تبديلهم

المتغير t يحتفظ بقيمة ال x مؤقتا حتى لا تضيع عندما تأخذ ال x قيمة ال y

يأخذ ال y قيمة ال t التي هي في الأساس كانت قيمة ال x و هكذا فعليا تكون قيمة ال x و y قد تم تبديلهم .

لكن هناك شيء يبدو غريب !

عند طباعة الناتج وجدنا أن القيمتين لم يتغيروا، فلماذا حدث ذلك ؟

لأن ال x و y داخل ال main مختلفين عن ال x و y الموجودان داخل fun

فبالرغم من تبديل قيمتى x و y داخل ال fun إلا أنهم لم يتغيروا داخل ال main

وبالتالى لحل هذه المشكلة نقوم بإرسال عناوين المتغيرات من ال **main** إلى ال **function** لتقوم بتغيير القيمة التى بداخل العنوان وعند استدعاء **fun** داخل **main** يصبح قيم المتغيرات فى **main** تغيرت بالفعل كما نريد.

```
#include <stdio.h>
void fun(int* x, int* y) {
  int t = *x;
  // xالمتغير t يحتفظ بالقيمة التي في العنوان
  *x = *y;
  // x القيمة التي في العنوان y يتم حفظها في العنوان
  *y = t;
  // y القيمة التي في المتغير t يتم حفظها في العنوان

  // main من الfun هنا تم تغيير القيم التى بداخل العناوين المرسلة لل
}

int main(  ) {
  int x = 5, y = 20;
  fun(&x, &y);
  // هنا يرسل عناوين المتغيرات بدل من القيم
  printf("x=%d  y=%d", x, y);
}
```

output:

```
x=20 y=5
```

بعض الأخطاء عند كتابة المؤشر **pointer** :

```
#include <stdio.h>
int main(  ) {
  int x, * p;
  //تشير إلى القيمة x
  //تشير الى العنوان p
  p = x; // خطأ

  //تشير الى العنوان  &x
  //تشير إلى القيمة  *p
  *p = &x; // خطأ

  //الاثنان يشيروا الى العنوان  &x and p
  p = &x; // ليس خطأ

  //الاثنين يشيروا إلى القيمة  x and *p
  *p = x; // ليس خطأ
}
```

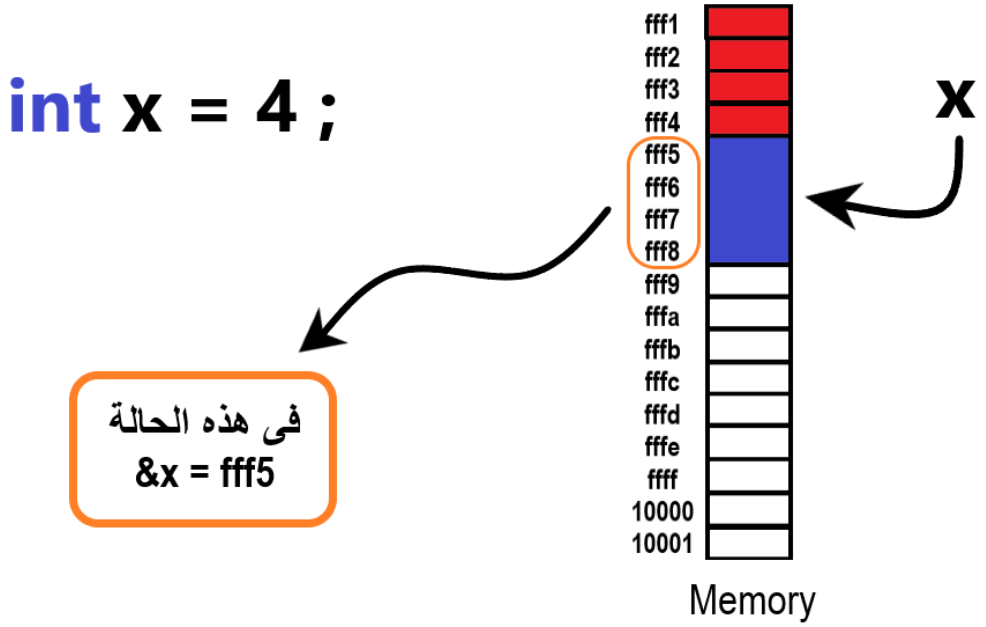
(قم بتجربة الكود بنفسك واضغط هنا)

بعض الملاحظات :

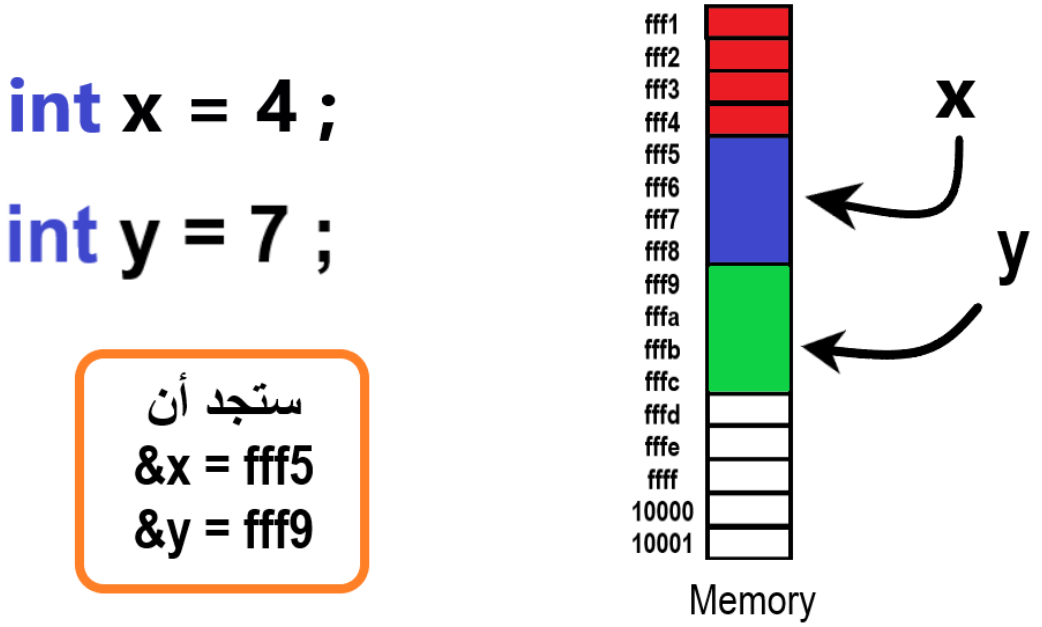
ذاكرة ال ram تحتوى على الكثير من الخانات و أصغر وحدة فيها هي ال byte ، و لكل نوع من أنواع البيانات مساحة محددة من ال bytes الذى يستحوذها عند حجزه ، و هذه المساحة متغيرة حسب نوع النظام الذى تستعمله و هي كالآتى :

Data Type	32-bit size	64-bit size
char	1 byte	1 byte
short	2 bytes	2 bytes
int	4 bytes	4 bytes
long	4 bytes	8 bytes
long long	8 bytes	8 bytes

كل byte في الذاكرة يكون له عنوان، لذلك عند حجز int مثلا ، يتم دمج 4 bytes معا ، و يكون عنوان أول byte فقط هو عنوان ال int



حتى إذا قمنا بحجز متغير من نوع int ثم متغير أخر أيضا من نوع int ، سيكون الفرق بين قيمة عنوان الأول و الثانى = 4



ربما قد تذكرت أن علامة & كنا نستخدمها في scanf ، فما علاقتها بال pointers ؟

في الحقيقة scanf عبارة عن function جاهزة داخل ال stdio.h كما ذكرنا سابقا، و أمر scanf الذى نكتبه هو عبارة عن ال call الخاص بهذه ال function و كما ذكرنا في درس ال functions أن هناك arguments يتم إرسالهم لل function للعمل عليهم .

ففي scanf ال arguments عبارة عن string يحتوى على format specifier مثلا %f% , %s , %c , d و هكذا و ال argument الثانى هو عنوان المكان الذى نريد تخزين القيمة فيه .

لذلك كنا نرسل العنوان إلى scanf function

هل أدركت مدى تأثير ال pointers !