

```

int index;
while (start <= end)
{
    Middle = (start + end) / 2;
    if (search == Arr[Middle])
    {
        index = Middle;
        break;
    }
    else if (search > Arr[Middle])
    {
        start = Middle + 1;
    }
    else
    {
        end = Middle - 1;
    }
}

```

ده الكود بتاع ال binary search

شرحه: هو عبارته عن انك بتدور في ال Array عن عنصر معين

فتشوف ال middle هل بيساوي لو

بيساوي يبقى تمام لقيناه لو لا بنسأل هل اكبر ولا اصغر لو اكبر هسيرش في النص الكبير لو اصغر هسيرش في الجزء الاصغر

مثال توضيحي: فرضا انت معاك كتاب

وبتسيرش علي صفحه معينه كتاب 1000

صفحه وانت عايز الرقم مثلا 503 انت بس

بتشوف النص (500) هي اقل من 503 يبقى

هي مستحيل تكون في النص الأول وبالتالي

هتجاهل النص الأول كله ويبقى كده السيرش

قل من 1000 ل 500 صفحه

نكرر ثاني 750 اكبر من 503 يبقى نتجاهل

ال 250 صفحه الي في الآخر

وببقى كده السيرش بقي من 500 صفحه

ل 250 صفحه

نكرر ثاني 250 < 125 < 63 < 32 < 15 < 7 < 4

انت حرفيا بصيت في ال 1000 صفحه خلال 7 خطوات تقريبا

بينما لو كنت شغال ال liner search الي هو عبارته عن بشوف صفحه بصفحه كنت اخدت 503 خطوه علشان توصل لنفس النتيجة.

**ملحوظة** ال break بتاثر علي ال loop مش علي if .

## String

الحروف احنا عارفين انها بيتيجي من ال ASCII table

فال string بيبقي عبارته عن ال array من ال char بس المميز في ال string انها بتختم ال array بتاعتها برمز مميز يميزها عن أي Array ثاني

يبقي عندنا كده حاجتين مختلفتين عن بعض وهما ال String وال Array of char

مثال:

```
char name [7]={'m', 'o', 'h', 'a', 'm', 'e', 'd'};
```

بالشكل ده هيتخزن في ال memory char ورا cahr عادي زيه أي Array عادي

انما ال String

```
char name [8]= "mohamed";
```

ممکن نخطها بین {} عادي

او ممکن نسیب [] فاضیه

الفرق هنا هو بیزود ال null character في اخر الاراي {\0}

یعني بتتمثل في ال memory زي الي فوق بالظبط لكن بیزود في الاخر ال null character الي هي 0 بس.

%s ده ممکن نستخدمه مع ال printf او scanf علشان اتعامل مع ال string

في ال scanf بيفضل یستقبل ال char لحد ما یلاقي مسافه او enter

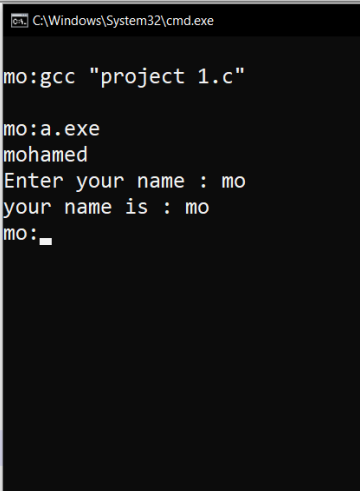
```
scanf ("%s",arr);
```

مش بنحط علامه ال & في ال scanf()

في ال printf بيفضل ی loop علي ال char ویطبع لحد ما یقابل ال '\0' ویقف

مهم :مفیش حاجه في ال C اسمها String هو بس char Array بس اخره \0

```
1 #include <stdio.h>
2 //#include "functions.h"
3
4 int main(void)
5 {
6     char name[]="";
7     char arr[]="mohamed\n";
8     printf("%s",arr);
9     printf("Enter your name : ");
10    scanf("%s",name);
11    printf("your name is : %s",name);
12    return 0;
13 }
```



عندنا أنواع ال Array

one dimensional array:

```
int num [3]= {1,2,3};
```

دي اراي كل عناصرها data type عاديه زي المثال ده فيه كل عناصره int

Multi dimensional array: (matrix)

```
Int nums[3] [2] = { { 1 , 2 } , { 3 , 4 } , { 5 , 6 } };
```

لكن دي عباره عن Array كل عنصر فيها عباره عن Array وكل Array تعتبر Array عاديه

ممکن نعتبرها بردو عبارہ عن rows و columns

طیب کده فهمنا شکلها طیب عایزین نجیب عنصر معین منها فرضا نجیب ال 1 هتقی ازای: nums[0][0]

ودی معناها عایز ال Array رقم 0 عایز العنصر رقم 0 فیها

مثال تانی رقم 4 هتقی : [1] [1] nums

وبتمثل فی المیموری زی ال array العادیه 1 و 2 و 3 و 4 و 5 و 6 ورا بعض

ممکن تبقی اکثر من 2 diminutions

مثال :

```
Int arr3D [2][3][2];
```

کده ارای بتتكون من 2 ارای کل ارای بتتكون من 3 ارای کل ارای فیهم فیها رقمیین int و ممکن تزود اکثر حسب الاحتیاج

طیب الارای بشكل عام بتتخط فی المیموری ازای بشكل عام

انا عرفت متغیر Arr ده عبارہ عن ارای بتخزن بمعلومیہ اول عنصر فیہ وبالتالي لما احب اجیب القیمہ بتاع المتغیر الی اسمہ Arr هیقی تساوی ال address الی متخزن فیہ اول عنصر فی ال Arr

وبتقی Read only مینفعش اگیرها بالبوینتر

طیب هو ازای بیعرف ال index بتاع العناصر التانیہ طالما هو مش مسجلها ومسجل ال Address اول

عنصر بس (alias of first elment)؟؟؟

Alias = ده اسم مستعار ل Address بتاع اول عنصر فی الارای

لان ال Array زی ما قولنا بیخزن العناصر ورا بعض فهو بس مجرد بیطبق معادلہ بسیطہ

ال Address الجدید = ال Address الأول + (ال index الی انا عایزه \* datatype size )

مثال:

```
double arr[3]={1,2,3};
```

```
Arr[2]
```

ال Address الجدید = ال Address الأول + (2 \* 8 بایت ) (ال double 8 بایت)

وبکده هیجیب العنصر التالت فی ال Array

## Pointer

هو عبارة عن datatype بتستخدم لحفظ address متغير ثاني

هنا في المثال المتغير X من النوع int بيتخزن فيه قيمة 5 في ال Address

رقم 0x70 الي هو 70 بس بال Hexadecimal



Pointe data type \* name;

المسافات بين \* وال Pointe data type وال name ممكن تكون موجوده وممكن لا

Ex: int \* pointer\_to\_interger ;

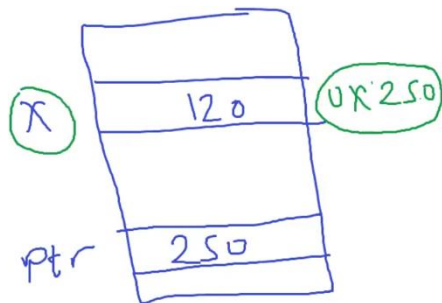
كده انا امرته بانه يعمل متغير بيشيل Address بس من نوع int

خلاص احنا كده عيايزين نخزن بقي ال address :

Int x=120;

Int \*ptr;

ptr=&x;



احنا استخدمنا ال Address operator علشان نجيب ال Address بتاع ال X

او ممكن في سطر واحد

Int \*ptr=&x;

في مصطلح اسمه ال Dereferencing

هو انا أوصل لمكان للميموري من خلال ال Address بتاعها

\*ptr=value;

هنا ال \* معاناها ادخل في المكان الي عنوانه متخزن في المتغير ptr وغير ال value الموجوده في ال Address ده

طب لزمه الشقلبه دي 😊؟؟

### calling by Value

```
void func (int Number)
{
    Number ++;
}
main ()
{
    int x=10;
    func (x);
    printf ("%d", x);
}
```

calling by value , calling by reference في ال

في العادي calling by value :

لما بنستخدمها في الحاله دي الي بيحصل ان ال function بتاخذ

Copy من المتغير x وبتعدل علي النسخه والاصل بيفضل زي ما هو

وبعدين بتستخدم النسخه في المثال الي قدمنا ال x=10

طيب انا اخدت مكان في الميموري زياده وعدلت علي المتغير الجديد وكل ده

ولسه مستخدمتش المتغير الي عندي

احنا حرفيا لو الكود ده جواه لوب علي Array فيها 100 عنصر الزاما هعمل

اراي ثانيه فيها 100 عنصر تانيين علشان ازود كل عنصر في الاراي واحد تضيق موارد كثير بالذات ان احنا في الامبيد بنحتاج كل bit عندنا في الميموري وبنحتاج يكون البرنامج سريع

### calling by reference (by Address)

```
void func (int *Number)
{
    *Number = 30;
}
main ()
{
    int x=10;
    func (&x);
}
```

لكن ندخل ال pointer في القصة:

الي حصل هنا عملنا func بيدخلها pointer to integer كانبوت ثم

في ال main بعطنا ال address بتاع المتغير من خلال ال & وبالتالي المتغير الي اسمه number موجود فيه حاليا ال Address بتاع المتغير X

دخلنا ال func ال \* هنا بتعمل dereferencing وبتغير قيمه ال X الفعليه وبتخليها 30

يعني بعد تنفيذ البرنامج الي قدمنا حاليا قيمه ال x=30

امثله: 1:

:

```
1 #include <stdio.h>
2 //include "functions.h"
3 int main(void)
4 {
5     int x=2;
6     int * ptr=&x;
7     printf("x before changes:%d\n", x);
8     *ptr=30;
9     printf("x after changes:%d", x);
10    return 0;
11 }
```

```
C:\Windows\System32\cmd.exe
mo: gcc "project 1.c"
mo: a.exe
x before changes:2
x after changes:30
mo:
```

## Expected Output

Write a C code that ask the user to enter 2 values and save them in two variables, then the program sends these variables by address to a function that returns the summation of them. The program then prints the result.

```
Please Enter Value 1: 10
Please Enter Value 2: 30
The result is: 40
```

```
1 #include <stdio.h>
2 //#include "functions.h"
3 int func(int *,int *);
4 int main(void)
5 {
6     int x,y,z;
7     printf("Enter first num: ");
8     scanf("%d",&x);
9     printf("Enter second num: ");
10    scanf("%d",&y);
11    z=func(&x,&y);
12    printf("the sum of two values is :%d",z);
13    return 0;
14 }
15 int func(int * num1,int * num2)
16 {
17     return *num1+*num2;
18 }
```

```
C:\Windows\System32\cmd.exe
mo: gcc "project 1.c"
mo: a.exe
Enter first num: 20
Enter second num: 30
the sum of two values is :50
mo:
```

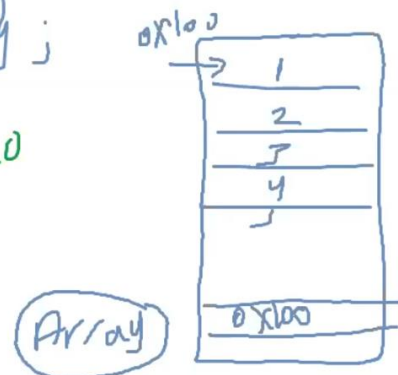
نأخذ بالنا هنا ملحوظة في ال proto type او في ملف ال h. ممكن نسيب ال input متغير اسم عادي

نرجع للاري ثاني: وافكر ان ال Array مجرد متغير بيثيل ال Address بتاع اول عنصر في ال Array (alies) ممكن اقرءه لكن

مستحيل اعدله

$\text{int Array}[5] = \{1, 2, 3, 4, 5\};$

$*\text{Array} = 20 \equiv \text{Array}[0] = 20$



وَممكن نعتبرهم ال pointer وال array بيادو نفس الوظيفة

```
int Array[20] = { . . . } ;  
int *ptr = Array ;  
ptr[0]      ptr[3]      ptr[7]  
Normal[0]    Array[3]    Array[7]
```

من الحاجات الي ممكن اعمل عليها ال pointer function

ال function بطبيعته الحال بيشارو علي جزء من الكود بيتم تنفيذه لما اعمله call

فانا عايز اعمله pointer

هو زي ال function بالظبط في ال Declaration لكن بس بزود حته ال (\*name) حوالين الاسم

Declaration of pointer to function:

Pointer type (\* name ) (inputs);

وعشان assignه الفانكشن (ممكن بال & او منغيره):

```
void Normal_func(void)
```

```
{
```

```
    Statements;
```

```
}
```

```
void(*ptr)(void)= & Normal_func;
```

```
void(*ptr)(void)= Normal_func;
```

لكن ال Derefrncing المره دي مش هيخليك تعدل الكود لكن هيخليك تنفذه وحاجه زي دي بنستخدمها في ال call back وهنحتاجها قدام

```
ptr();
```

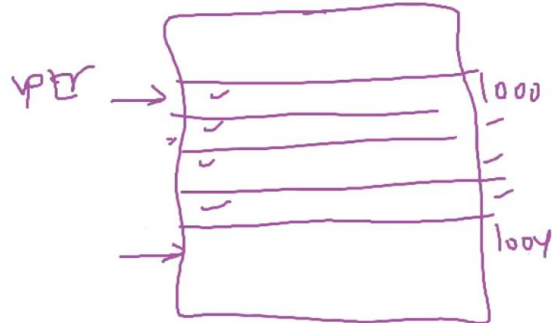
وال call بتاعها عادي كانك غيرت بس اسم ال function

ايه هي العمليات المسموحه علي ال pointer

### 1. increment

1) Increment Address direct

$\text{int}^* \text{ptr} = 1000$  ;  
 $\text{ptr}++$  ;  
1004



ال increment بتبقى بال step يعني ال int عنا بزيد 4 بايت لكن لو كان Double هزيد 8 وهكذا.....

### 2. Decrement نفس الحاجه

### 3. Subtraction

بيطرح Step مش رقم

$\text{int}^* \text{ptr} = 1000$  ;  
 $\text{ptr}--$  ;  
996

$\text{char}^* \text{ptr} = 1000$  ;  
 $\text{ptr}--$  ;  
996

### 4. Addition نفس الحاجه

### 5. Subtraction pointer from pointer

$\text{int}^* \text{ptr}_1 = 2000$  ;  
 $\text{int}^* \text{ptr}_2 = 1000$  ;  
 $\text{int } X = \text{ptr}_1 - \text{ptr}_2$  ;  
250

ازاي 1000-2000 هيطلع ب250 الفكره انه بيتعامل بال step واحنا عارفين ان ال step بتاعه ال int عبارته عن 4 بايت



يبقي 2000 ← 500 ستيب بعد ما قسمنا علي 4

وال 1000 ← 250 ستيب بعد ما قسمنا علي 4 والفرق ما بينهم 250 بس محدش بيستخدمها لان ملهاش تطبيق

## 6. Additon ( invalid) مينفعش

ملحوظات مهمه:

```
int* func (void)
{
    int x = 20;
    return &x;
}
main ()
{
    int* var = func ();
}
```

### • Dangling pointer :

احنا هنا عرفنا x في الفانكشن واتحجز لها مكان في ال Stack الوقتي عملنا return لل Address الخاص بال x والفانكشن خلصت فالحجات الي كانت موجوده في ال Stack كلها اتمسحت من ضمنهم ال x فبالتالي ال Address الي رجع من ال function ببشاور علي حاجه ثانيه او مكان فاضي

حاجه زي دي ممكن تسبب مشاكل اثناء ال run

الحل: ممكن نخليها gloabl

او نخليها pointer ببيعتها اليوزر وتفضل موجوده

او نخليها static دي بتمنع ان ال variable يتمسح بعد ما الفانكشن تخلص بس متشاف بس بس local

### • Wild pointer :

هو بوينتر عادي عملتيه Derefrecing منغير ما اديله Address

```
int * ptr ;
print ("%d", *ptr); → runtime
```

مثال: واحد عامل فانكشن صح وانت جيت تستخدمها و عملت الشكل الي علي الشمال ده

```
int *ptr ;  
  
func (ptr) ;  
  
func (int * Arr)  
{  
    *Arr = 20 ;  
    *Arr + 1 = 20 ;  
}
```

حل بقي مشكله زي دي ايه

انت لازم تعرف ال pointer وتديله قيمه علطول طيب ايه القيمه الي هحطها؟؟

لو مشعارف تحط ايه حط ال null ودي عبارته عن مكان بره الميموري بتاعتك أصلا او Zero عاديته او معمولوها تحويل لبوينتر علي حسب ال Stader الي اتفق عليه المبرمجين ما بينهم كده 😊

طيب الرجل الطيب الي عامل الفانكشن ده يعمل ايه

حاجه بسيطه لو البوينتر ببساوي null متشتغلش غير كده اشتغل