

scanf ("%c",&var);

والأنبوب كان ب2

if(x==var)

بتادي نفس الوظيفة بتاعه: الـ scanf نفس القيمة

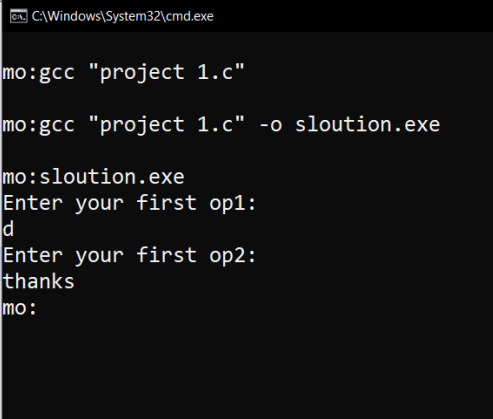
if(x=='2')

الي بيحصل انه في الاولي الكومبيلر بياخد القيمة المتغير ويشوفها في الـ ASCII table وبيحفظ الرقم الي ظهر له في الـ ASCII table وده نتيجة للاستخدام الـ %c

لكن في الحاله الثانيه هو بياخد الرقم الناتج من تحويل الـ '2' لـ char بالتالي هيروح يشوف القيمة بتاعته في الـ ASCII table

مثال:

```
1 #include <stdio.h>
2 int main()
3 {
4     char op1,op2;
5     printf("Enter your first op1: \n");
6     scanf("%c",&op1);
7     printf("Enter your first op2: \n");
8     scanf("%c",&op2);
9     printf("thanks");
10 }
11
```



ايه الي حصل هنا؟؟

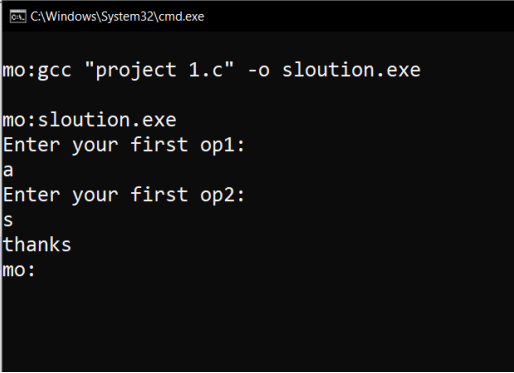
اول حاجه احنا كتبنا حرف ع الكيبورد d ثم ضغطنا Enter علشان نكمل البرنامج

نتيجه لكده ان البروسيوسور دخله الـ d خذنها في op1 والـ Enter خذنها برقمها في الـ ASCII table في المتغير op2

وبكده مسمش ليك انك تعمل أي حاجه غير انك تدخل اول حرف وخلص البرنامج لانه اخذ الي هو محتاجه وده بسبب الـ printf

وهناك عدده حلول ابسطها واسهلها نضع مسافه قبل الـ %c بس كالمثال الاتي:

```
1 #include <stdio.h>
2 int main()
3 {
4     char op1,op2;
5     printf("Enter your first op1: \n");
6     scanf(" %c",&op1);
7     printf("Enter your first op2: \n");
8     scanf(" %c",&op2);
9     printf("thanks");
10 }
11
```



وده مثال جيد للتطبيق :

```
1 #include <stdio.h>
2 int main()
3 {
4     char operand='r',repet='y';
5     float op1,op2;
6
7     while(repet=='y')
8     {
9         printf("Enter your first operand: ");
10        scanf("%f",&op1);
11        printf("Enter your second operand: ");
12        scanf("%f",&op2);
13        printf("Enter your operand : ");
14        scanf(" %c",&operand);
15        if(operand=='+')
16        {
17            printf("the result is : %0.2f + %0.2f = %0.2f \n",op1,op2,(op1+op2));
18        }
19        if(operand=='-')
20        {
21            printf("the result is : %0.2f - %0.2f = %0.2f \n",op1,op2,(op1-op2));
22        }
23        if(operand=='*')
24        {
25            printf("the result is : %0.2f * %0.2f = %0.2f \n",op1,op2,(op1*op2));
26        }
27        if(operand=='/')
28        {
29            printf("the result is : %0.2f / %0.2f = %0.2f \n",op1,op2,(op1/op2));
30        }
31        printf("do you want repet y to yes n to no : ");
32        scanf(" %c",&repet);
33    }
34 }
35
```

```
C:\Windows\System32\cmd.exe
mo: gcc "project 1.c"
mo: a.exe
Enter your first operand: 2.3
Enter your second operand: 5.25
Enter your operand : +
the result is : 2.30 + 5.25 = 7.55
do you want repet y to yes n to no : y
Enter your first operand: 2.3
Enter your second operand: 6.5
Enter your operand : -
the result is : 2.30 - 6.50 = -4.20
do you want repet y to yes n to no : y
Enter your first operand: 4.2
Enter your second operand: 5.6
Enter your operand : *
the result is : 4.20 * 5.60 = 23.52
do you want repet y to yes n to no : y
Enter your first operand: 2.3
Enter your second operand: 5.69
Enter your operand : /
the result is : 2.30 / 5.69 = 0.40
do you want repet y to yes n to no : n
mo:
```

تطبيقها اسهل بال switch case

## Ternary operator

### Syntax:

Var =condition ? A:B;

هي عبارة عن if else ولكن في شكل ابسط بس مش لتنفيذ مجموعه أوامر

هي بس لمجرد تحديد قيمه بناء علي شرط

```
1 #include <stdio.h>
2 int main()
3 {
4     float x,y,z,var;
5     printf("Enter 3 nums :");
6     scanf ("%f %f %f",&x,&y,&z);
7     var =x>y? x:y;
8     z = z>var? z:var;
9     printf("the greater is = %.2f ",z);
10 }
```

```
C:\Windows\System32\cmd.exe
mo: gcc "project 1.c"
mo: a
Enter 3 nums :5.2 5.1 5.3
the greater is = 5.30
mo:
```

## Go to(unconditional branch)



متستخدمهاش لو حتي تحت تهديد السلاح(دي نصيحه من اخوك 😊) في طرق اسهل واحسن : بس لازم تعرفها برنو.

syntax:

goto label1

label :  
=

```
main () {  
    go to label1:  
    Statements  
    Label1:  
    Statements  
    go to exit:  
    Statements  
    exit:  
    statements  
}
```

مثال :

---

## Functions

جزئيه من الكود بنفذها بتكرار بدل ما اقعد اكتبهم كذا مره انا بس مجرد بستدعيها وهي بتنفذ الكود الي جواها

بتتكون من ايه الفانكشن او دوره حياتها ايه:

1. Declaration(prototype)
2. Implementation(definition)
3. Call

ال Declaration عبارته عن :

`Return_type function_name (input1_type input1_name, input2_type input2_name ) ;`

EX:

`Int add(int x , int y);`

مممكن يكتب بس فوق قبل ال main او File ثاني

ال implementation عبارته عن:

يبقى بعد ال main او file ثاني

Return\_type function\_name (input1\_type input1\_name, input2\_type input2\_name )

```
{
    statements;

    return var
```

ال Return بتبقى نفس نوع ال function وغيها يبقى النوع void او سيبها فاضيه

```
}
```

ملحوظه :

```
main ( )
{
    int op1=10, op2=20;
    int Addition=Add ( op1, op2 ) ;
    printf ("%d", Addition) ;
}
```

- ال printf ب return عدد الحروف الي هتطبعتها
- ال main هي فانكشن وهي اول فانكشن بتننفيذ في الكود

```
/*proptotype or declaration*/
int Add(int x, int y);

int main()
{
    int op1=20, op2=30;
    int addition=Add(op1, op2);    /*Function call*/
    printf("%d", addition);
}

/*Implementation or definition*/
int Add(int x, int y)
{
    int result= x+y;
    return result;
}
```

هي بقي بتشتغل ازاى بعد ما عملنا call زي ما في الرسمه الي فوق

ايه الي بيحصل في الكومبايلر اول ما يشوف ال call بيروح علي مكان الميموري الي متخزن فيه ال function و لو معاه متغيرات بيبقي شايفهم في ال Function وبيدء بتنفيذها بيطلع معاه ناتج لو فيه return هيرجع بنسخه منه لنفس المكان الي كان فيه قبل الاستدعاء.

ملحوظه :

- ممكن تستخدم Functions جوا function
- ممكن نكتب void في الجزء بتاع ال prameters لو مش عايز ابعث حاجه مع ال function

## Local variable & global variables

من الاسم ال local variable دي متغيرات بتبقى متشابهه بس في الاسكوب الي متعرفه فيه زي { } بتوع ال function او ال for loop او ال if

بينما ال global variables دي متغيرات متعرفه بره ال main وبتبقى متشافه في كل البرنامج

ناخد بالنا ان ال local variables بتتخذن في ال Stack في ال memory وده بيكون كل مره بيشتغل في أي function \

وكمال ال gloabl variables بيتخذن في جزء تاني

يعني مثال:

بيبدء تنفيذ البرنامج فيبضيف داتا في ال stack وهو اثناء ما هو شغال ظهر function وأضافت بعض المتغيرات فيبخرن الجزء الجديد و اول ما ال function تنتهي بيمسح الداتا بتاعتها ومعتدش موجوده

ملحوظه : في ال Embedded في ال main بنكتبها void لان مفيش OS هيتعامل مع المومري وحاجات تانيه فمفيش حاجه هت return لكن في ال Computer العادي بنستخدم علشان يتعامل مع الميموري ويقول ان الأمور انتهت تمام ومفيش مشاكل حصلت بسبب البرنامج فب 0 return علشان اتأكد ان كل حاجه مشيت صح وده اختلاف مهم ناخد بالنا منه

فيه مصطلح تاني اسمه : recursion

وده معناه اني بستدعي function جو نفسها تاني ولازم يكون في شرط للخروج علشان مدخولش في infinity loop

project1.c

functions.c

functions.h

1 #include <stdio.h>  
2 int add(int x1,int x2)  
3 {  
4 return x1+x2;  
5 }  
6  
7 void printName(void)  
8 {  
9 printf("yes it work!!!!!!");  
10 }  
11

project1.c

functions.c

functions.h

1 int add(int x1,int x2);  
2 void printName(void);

project1.c

functions.c

functions.h

1 #include <stdio.h>  
2 #include "functions.h"  
3 int main(void)  
4 {  
5 printf("the result is = %d\n",add(3,2));  
6 printName();  
7 return 0;  
8 }

C:\Windows\System32\cmd.exe

mo: gcc "project 1.c" functions.c  
mo: a.exe  
the result is = 5  
yes it work!!!!!!  
mo:

نلاحظ حاجه مهمه ان الملفين ال .c لازم يتعملهم compile علشان البرنامج يشوفهم

```
mo: gcc "project 1.c" functions.c
```

غير كده مش هيشوفها:

project1.c

functions.c

functions.h

1 #include <stdio.h>  
2 #include "functions.h"  
3 int main(void)  
4 {  
5 printf("the result is = %d\n",add(3,2));  
6 printName();  
7 return 0;  
8 }

C:\Windows\System32\cmd.exe

mo: gcc "project 1.c"  
C:\Users\Original\AppData\Local\Temp\ccw0jd2G.o:project 1.c:(.text+0x1e): undefined reference to 'add'  
C:\Users\Original\AppData\Local\Temp\ccw0jd2G.o:project 1.c:(.text+0x33): undefined reference to 'printName'  
collect2.exe: error: ld returned 1 exit status  
mo:

نحاول نطبق علشان الفكره تثبت