Code Blocks

1- Basic sentences in any C-Program :

```
#include <stdio.h>
int main( )
{
.
.
.
return 0;
}
```

Note: • Each sentence ends with a (;)

• Variables (name) can include letters + numbers + underscore → and it can't start with a number

* to make the program Print any line you want :-

- Printf ("The line \n") ; or -

↳ • you use it to start a new line

* to make the program read anything you want :-

- scanf ("%d", &num);

↳ • Typesof file → name of the variable.

int → d
float → f
double → lf
char → c

* to print the output :-

- Printf ("sum = %d", sum)

↳ int

• Types of files :
int → d
float → f
double → lf
char → c

• in Code Blocks if you make a mistake :-

→ it's either an error or a Warning

error: you can't run the Program with it

Warning: ↳ They don't affect your program But try to fix them

• Types of errors :-

• run-time error: Exp: X=5;
y = X-4;
y = y-1;
Z = 3/y; ⟹ y = 0 so it's a wrong equation

• Logical error : Exp : - If the program is to sum and you put sum = x − y

- or avg = (g1 + g2)/2 ← the answer will be wrong so you put ( )

Note :-

• to define Constants : like: PI = 3.14 , $\epsilon_0 = 8.85 \times 10^{-12}$

```
#include <stdio.h>
#define PI 3.14        (No "=" sign , All Capitals)
int main()
```
(space)

Note :-

• If your want to use a character when you scan it :

scanf ("%d %c", &age, &genelur)
you have to put a space

Note :-

• If you want to write an equation make sure to use Parentheses ( )

* example: X = 5 + 2*3 + 4/2*5
= 5 + 6 + 10 = 21

→ with Parentheses: X = [(5+2) * (3+4)] / (2*5)
= 7 * 7 / 10 = 4.9

How to use Type Casting

Note :-
variables can be:
int, float / double, Char

If you want to use divider (/) Then the answer can be :-

example : X, Z : x is an integer and Z is a float
X = 3/2 The answer is 1
Z = 3/2 " " " 1.0
If you want the answer to be 1.5 you use :-

z = 3/2.0 or 3.0/2 or 3.0/2.0

- لكي تحصل على جواب كسري يجب ان يكون البسط والمقام او الاثنين مكتوبان بصراحة العواصل

- لكي تحول رقم int ← float غيره واحده استعمل Type Casting

x = 3, y = 2 / q y is integers : مثل

Because this is an integer ← (a) = (float) x/y; → 1

z = (float) x/y; → 1.5

Output formatting

float:- Printf ("%10.2f")

f = 73.62 width عدد الخانات (التقريب) number of digits

Ex:- _ _ _ _ 7 3 . 6 2

integer:- Printf ("%5d")

Ex: d = 53 width

_ _ _ 5 3

- If there is a negative sign Ex %-3d (d = 53)
  Then we put numbers left to right 5 3 _

Comments

- There are 2 ways:-

↳ using /* ——— */ /* —— */ Ex: #—— int—— main() {

/* ——— */

↳ or // ——— Ex:- sum = x+y; // added x to y

files :-

How to present a file to the computer ↳

you have to type : FILE * in;*out; (in) either you enter infos or you print info (out)

FILE: ALL capitals — *: Pointer

in = fopen ("data.txt", "r");

"data.txt": name of your file — "r": Mode (read or write)

→ fscanf (in, "%d %d", &n1, &n2) → name of variables

fscanf: To read the data

in: could be out-too

"%d %d": Type of variables

At the end of your program and befor return you Type :-

```
fclose(in);
or fclose(out);
```

functions

① system defined functions

② user defined functions

- They are used to Preform a task inside a Program :-

② First : you name your function with a header :-

Ex : a function called sum :-

```
int sum(int, int);
```
← Prototype

second: you call the function during the program :-

Ex: `S = Sum(X, Y);` ← function call

int int

lastely: you define the function

Ex:

```
    return 0;
}
int sum ( int a, int b)
{
    int result;
    result = a+b;
    return result;
}
```
→ definition of Program

① Ex: Sqrt (square root √) (double) → double

defined by the system

• Pow (int, 2) → (int)^2 (double) → double

2 ↓ Power

• floor (أكبر عدد صحيح) (2.5 → 2) (double) → int

• Ceil (يؤخذ الرقم الصحيح فوق الكسر) (2.5 → 3) (double) → int

• abs | | : خاصة لطباعة integers - (int) → int

• fabs | | : خاصة لطباعة double - (double) → double

• Trigonometric functions : sin, Cos, Tan, Cot, SeC, CsC → (double) → doubl

• functions can :- 1- → Takes & doesn't give back data

2- → doesn't Take and give back data

3- → doesn't Take and doesn't give back data

1- we put void as a proto Type Exp :- void sum (int, int);

when Calling the function :- we don't use ~~void~~ sum (x, y);

when defining the function we put void and we don't put return

Selection :-

- relational operator
  - < less than
  - > more (greather) than
  - <= less or equal
  - >= greater or equal
  - == equal
  - != notequal
- logical operator
  - && and
  - || or
  - ! not

Ex :- Rational operator

```
if (x>5)
  Printf("good\n")
else
  Printf("bael")
```

Notes

- In C-language :-
  0 means false.
  anything else is True (on)

→ If you print :-
x=1 x=0 x=5
if (x=5)
Then - -
else
- - - -

→ It will print good in all Cases Because = is to define

If you print
if (x=0)
The- - -
else- - -

→ It will print bael in all cases Because zero means fals so the program jumps to else

Ex: logical operator

```
if ( (age>20) && (gender=='F')
        ||(avg>85))
```

If we enter age = 15, gender = 'F', avg = 70

= (F && T || F)
= (F || F)
= (F)

- !() is the most important
- && is more important than ||
- paraenthesis ( ) is the most important

Example :- A program to know wheather a letter is a vowel or not :-

```
Char letter;
Printf("enter letter")
Scanf("%c", &letter);
```

→ `if (letter=='a' || letter=='o' || letter=='i' ||---- )`

• Using switch :-

- it works only with == (Doesn't work for < >)
- it works with integers & Asccicode (a, b, c, ——)
  (But doesn't work with float and string)
- break; is used to stop the program

• The Main structure:-

```
switch (Variable)
{
Case [      ] : printf / Do an operation ;
            break ;
Case [      ] : printf / Do an operation ;
            break;

default : Do another thing ;
}
```

• Example :- A program to know if a letter is a ~~to~~ vowel or not

```
switch (letter)
{
 Case 'a' : case 'i' : Case 'o' : case 'u' : Case 'e'
   Printf ("%c is a vowel", letter );
    break;
default : printf ("%c is not a vowel", letter);
}
```

Nested if :-

it's like putting an if inside another if

Ex:- `x = 4 , y = 3 ;`

```
if (x > y);
{ if (x == 4);
    printf ("good\n");
  The second if
}
else
{ printf("bad\n");
  printf ("bye\n");
}
```

إضافة قبل الأقواس In this Case This else is for

يلزم تحقيق الشرط الأول ثم تحقيق الشرط الثاني لكي يطبع good ، هنا يطبع good bye

• بإضافة الأقواس الزرقاء تصبح else تابعة لـ if الأولى تطبع good bye

• لو كانت x = 2 يطبع bad bye

• إذا لم تضع هذه الأقواس وتحقق الشرط يطبع bad و bye

• وإذا تحقق يطبع good فقط

Boolean function

(True or false) / (yes or no) هو function لا ترجع عدد وإنما ترجع إجابة.

Ex:- `int isEven (int n);` ← it's like asking is this even?

```
{
if (n % 2 == 0)
    return 1; ← True
else
    return 0; ← false
```

loops :-

- while
- for
- do/while

While loops

متناهما مع بكون العمليات ثم زد ١ آخر شيء

متناهما زد ١ على المتغير ثم قم بأي عملية أخرى

- Increament by one → i++ or ++i
- decrement by one → i-- or --i

* i++ and ++i are different

Ex:
```
X=5
X++
print X
```
→ here it prints 6

```
X=5
++X
Print X
```
→ here it prints 6

But :-
```
X=5
y=++X
Print X
```
→ it prints 6 and y=6

في الحالتين تزيد x ولكن في الحالة الأولى عوضت قيمة الزيادة في y أما في الحالة الثانية عوضت قيمة x القديمة في y ثم زدنا واحد لذلك تبقى y = 5

```
X=5
y=X++
Print X
```
→ it prints 6 and y=5

for loops :-

- The same as while
- تختصر كل العملية في سطر واحد
- Ex:1- `for (x=1 ; x<=5; x++)` printf("good\n");

لا تضع فاصلة منقوطة لهذا ذلك يجعله يدخل في loop دون ان يقوم بأي شيء

2- `for (x=1; y=10; (x<=5) && (y>=5); x++, y--)` printf("hi\n")

- Note :

If more than one variable control the loop we use && such as:-

```
x=1 ; y=10;
while (x<=5) && (y>=5)
{ printf--- x++; y--;
```

do / while

- هي اللوب الوحيدة التي يجب ان تدخلها مرة واحدة على الاقل لأن التحقق من الشرط موجود بآخرها

- Where you put ; after while

Ex: a program that counts number of digits

```
X= 12345;
Count = 0;
do
{
X=X/10
Count ++;
}
while (X > 0);
```

break and Continue

- break is used to stop the loop

ex:

```
X=3
while (X<7)
{ printf("%d", X);
if (X==5)
  break;
  X++;
}
printf("bye\n");
```

files & loops :-

- برنامج يستقبل المعلومات من file ويتوقف باشارة معينة

Ex1 Grades

```
int g1, g2, status;
float avg;
FILE *in;
in = fopen("data.txt", "r");
status = fscanf(in, "%d%d", &g1, &g2);
while (status != EOF)
  {
    avg = (float)(g1+g2)/2
    printf("avg = %.2f\n", avg);
    status = fscanf(in, "%d%d", &g1, &g2);
  }
```

اشارة التوقف

Note!

- في حالة Nested loops عدد مرات اللوب تساوي عدد مرات اللوب الأول ضرب عدد مرات اللوب الثانية

مثال :-

```
for (int a=1; j=1; j<=5; j++)
  for(i=1; i<=5; i++)
    {
      printf("%d\n", a);
      a++;
    }
```

في هذه الحالة يطبع : 25

يدخل في هذه اللوب ٥ مرات قبل الانتقال للوب الأكبر مرة أخرى