(note) Function can't be defined inside anther Function but can be called inside anther Function.

What is the meaning of Following Function prototype with empty parameter list??

answer

Function can be called with any number of parameters of any types.

```
Void Fun( )
{
  /*____*/
}
```

Explanation :- empty list in C mean that the parameter list is not specified and Function can be called with any number of parameters of any types.
but in C++ empty list means Function can only called without any Parameters.

in C++ → Void Fun() = Void Fun(Void) [are same]

in C → Void Fun() ≠ Void Fun(Void) [not are same]

(note) Sizeof() operator is evaluted at Runtime
its result → in bytes.

How many Times the program will print "IndiBix"??

answer

Till stack over Flows

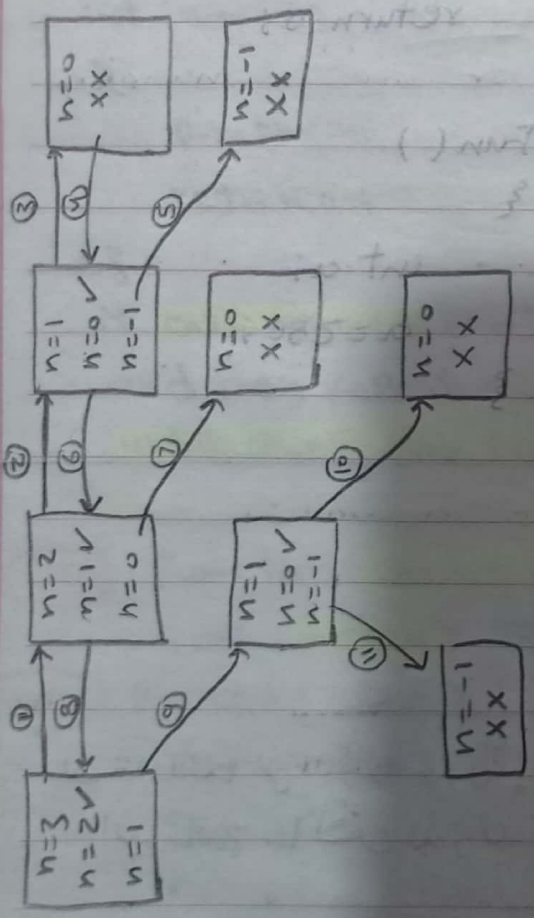
هنا هتفضل تطبع ويعمل يستدعي الـ main لحد ما مساحة الـ stack تملي فده هيعمل error الي هو stack over Flows

```
int main()
{
  printF("IndiaBix");
  main();
  return 0;
}
```

---

what will be The out Put of the Program??

```
#include <stdio.h>
void Fun(int);
int main()
{
  int a=3;
  Fun(a);
  return 0;
}
void Fun(int n)
{
    iF (n>0)
    {
      Fun(--n);
      PrintF("%d, ",n);
      Fun(--n);
    }
}
```



output ⇒ 0 1 2 0

note → ✓ معناه إن الرقم ده هيتطبع

xx معناه إن شرط (n>0) مش متحقق

Can you put Function definition in main Function??

Yes → in C99 you can do it but,
You should put auto before it to tell the compiler this Function is local in main and You can't use it out main.
in C89 → This will make error.

What is the output??

```c
#include <stdio.h>
int Fun();
main()
{
    int i;
    i = Fun();
    printf("%d", i);
    return 0;
}
int Fun()
{
    int a;
    a=250;
}
```

(int ← مش لومش مكتوبة)

output → 1

- Return datatype of any Function by default is int
- When Function excuted successfully and we don't return anything, by default C compilers returns 1 For its successful execution.

عشانها الـ Fun هتتنفذ كدا بدون أي مشاكل بس مطبعش جملة return جوها فالـ compiler هيرجع 1 بشكل تلقائي
: i=1
output → 1

note exit(); لو قابلتنا في الكود فدي بتقوم بقفل البرنامج خالص بدون تكملة الكود

important note

Can Array pass by Value??

yes when we put the array in structure and we pass structure by value To a function then we copy all elements of array and this pass the array by value

هيتم شرحها بالتفصيل في الـ structure

---

note ex:- a>50? return 1 : return 0; → error

In aternary operator we cannot use return it requires expressions not code.

what is the output??

```
#include<stdio.h>
int main()
{
    printf("%p", main());
    return 0;
}
```

it calls main() function and then repeat infinite untill stack over flow

```
#include<stdio.h>
int main()
{
    printf("%p", main);
    return 0;
}
```

output → it will print address of pointer of the main() function.

---

note Function cannot return more than one value at a time. True (✓)

note Function can be called either by value or reference. True (✓)

ex
```
c=add(a,b); ✓
c=add(&a,&b); ✓
```

التاريخ : الموضوع :

note A Function may have any number of return statements each returning different values.

True ⇒ example

```
int Fun(int a)
{
  if(a == 5)
    return 1;
  else
    return 0;
}
```

note Names of Functions in Two different files linked together must be unique

and if two Function are declared in a same name it gives error

Multiple declaration of Function_name().

note usually Recursion works slower than loops.

why لأن لما بيحصل recursion و ال Fun بتستدعي نفسها بيحصل push في ال stack بتاع ال Fun وده هياخد وقت وميموري أكتر أما في حالة ال loops مش هيحصل الكلام ده و بالتالي in loops we save a lot of time and space.

What can functions return in C ??

answer Functions can return any type expect array and function

هنا قصده بال return إنه يرجع القيم كـ value

فالبتالى ال array & function مش نرجعهم

بس نقدر نحل المشكلة دى ازاى ??

by returning → Pointer to array.
→ Pointer to Function.

---

what is the output ??

output → 2 1

```
#include <stdio.h>
void dynamic(int s, ...)
{
    printf("%d ", s);
}
int main()
{
    dynamic(2,4,6,8);
    dynamic(1,3,5);
    return 0;
}
```

in C, three continuous dots is known as ellipsis which is variable number of arguments of function. The values to parameters are assined one by one.

هنا احنا هناخد اول
رقم هيتبعت لل dynamic
وهنخزنه فى ال s جديد
هنطبع ال s وبالتالى
output will be 2 1. #

Important note

Can a Program be compiled without a main Function??

answer → True

How

1. using a macro that defines main

```
#define macro main
int macro(void)
{
    printf("Hello");
    return 1;
}
```

هيشتغل تمام هو بس هيشيل macro ويبدلو بكلمة main

2. using Token-pasting operator

```
#include<stdio.h>
#define macro m##a##i##n
int macro()
{
    printf("Hello");
    return 0;
}
```

في الحالة دي هو هيشيل كل ال ## وبعد كدا هيبدل كلمة macro بـ main ويكمل هيشتغل عادي

طب حاليا في أول حالتين إحنا كنا بنكتب main بردوا طب مفيش حل من غير ما نكتب main ؟؟ 😐

الحل هو إننا نغير ال entry Point بتاعت البرنامج ونخليها أي حاجة تانية غير main طب هتشتغل كدا إزاي ؟ 😐

تعال نشوف الكود ←

التاريخ :

الموضوع :

3 Modify the enty point during compilation

```
int nomain();
// The symbol _start is the entry point of the program.
void _start()
{
    nomain();
    exit(0);
}
int nomain()
{
    puts("Hi");
    return 0;
}
```

by default
يبقى مكتوب هنا
دالة اسمها main()

الموضوع :

التاريخ :

4 Functions in math.h

1 round (double num)

هنا الـ round بتشوف الـ Fractional comp بتاعت الرقم

لو هي أكبر من أو تساوى 0.5 فهي هتقرب الرقم لأقرب رقم أبعد من الصفر.

ex round (3.5) = 4.00

round (-3.5) = -4.00 ← صابقت 4- عشان هي أبعد عن الصفر

لو هي أقل من 0.5 فالرقم هيفضل زي ما هو بدون الـ Fractional component

ex round (3.1) = 3.00

---

2 ceil (double num)

هنا الـ ceil بتقرب الرقم دائما لـ The next highest integer

مش بتتأثر بالقيم بتاعت الـ Fractional component.

ex ceil (3.5) = 4.00 because 4 > 3

ceil (-3.5) = -3.00 because -3 > -4

---

3 Floor (double num)

هنا الـ Floor بتقرب الرقم دائما لـ The next lowest integer

مش بتتأثر بالقيم بتاعت الـ Fractional component

ex Floor (3.5) = 3.00 because 3 < 4

Floor (-3.5) = -4.00 because -4 < -3

---

4 trunc (double num)

الـ trunc هنا بتلغي قيمة الـ Fractional component وتخلي قيمته تساوى صفر

ex trunc (3.5) = 3.00

trunc (-3.5) = -3.00

---

note How to check For even or odd number by using bitwise operator ??

Solution :-

```
(num & 1) ? printf("odd") : printf("even");
```

ex num = 3

011 & 001 ⇒ 001 (True) → odd number

ex num = 2

010 & 001 ⇒ 000 (False) → even number

another method using arithmetic operator :

```
((num%2) != 0) ? printf("odd") : printf("even");
```

---

note How To check number is power of 2 ??

Solution :

```
if ( (num & (num-1)) == 0)
        printf(" True");
else
        printf(" False");
```

ex num = 8 num-1 = 7

1000 & 0111 = zero

← True ✓

ex num = 6 num-1 = 5

110 & 101 = 100 ≠ zero

← False ✓

How to check number is power of [3] ??

```
double i ;
i = log(num) / log(3)
(trunc(i) == i) ? printf("true") : printf("false");
```

ex num = 9

$$i = \frac{\log(9)}{\log(3)} = \frac{\log(3^2)}{\log(3)} = \frac{2\log(3)}{\log(3)}$$

i = 2

trunc(i) = trunc(2) = 2

(2 == 2) → True

ex num = 6

$$i = \frac{\log(6)}{\log(3)} = 1.63$$

trunc(i) = trunc(1.63) = 1.00

(1.00 == 1.63) → False not power of [3]

What is the output??

```
char ch = 'a' ;
printf("%d %d %d %d", sizeof(int) ,sizeof(ch) ,sizeof('a')
                    , sizeof("a"));
```

output → 4 1 4 2

explantion →
- sizeof(int) = 4 byte
- sizeof(ch) = sizeof(char) = 1 byte
- sizeof('a') ⇒ 'a' convert to it's ASCII so it is int
  ∠ sizeof('a') = sizeof(int) = 4 byte
- sizeof("a") ⇒ "a" is a string = 'a' + '\0'
  sizeof("a") = size of two char = 2 byte

note
```
int x,y,z;
z = scanf("%d %d", &x, &y);
printf("%d", z);
```

output → 2 Why ?

scanf() returns number of inputs

so the input x,y so z = 2

printf() returns number of characters that is print

ex x = printf("Mostafa Edrees");

x = 14 بتعد الـ space

```
int main()
{
    int x;
    int x= 5;
    printf("%d", x);
    return 0;
}
```

output → Compilation error

why → Redeclaration of 'x'

```
int x;
int x=5;
int main()
{
  printf("%d", x);
  return 0;
}
```

output → 5

why → C allows a globel variable to be declared again when First declartion doesn't initialize the variable.

```
int x=5;
int x=10;
int main()
{
   printf("%d", x);
   return 0;
}
```

output → Compilation error

why → Redeclartion of 'x'