

COAL ASSIGNMENT# 03

RECURSION AND STACKFRAMES

Topic: Counting Total Products in a warehouse through counting total rows (recursively).

COAL ASSIGNMENT # 03

Date 03/12/2020

RECURSION AND STACKFRAMES

- Application / System where recursion is used/implemented:-

The real life scenario that I have choose to implement through recursion is calculating the total number of rows of any large auditorium, or any warehouse holding thousands of products for a company and an owner wants to calculate total stock left, he just knows how many products are there in single row. So my recursion implementation will count the total no of products in his ware house.

- It is required to clarify running time requirements.

∴ Though we know number of rows can be very large so let's assume/sec implementation / logic first.

~~the~~ ALGORITHM:-

"Program works something like this:"

Every row will is given with any non zero number or considered atleast but only the first row is given a number 0. Since there are no rows in front of it.

- It will check every front row until it finds zero which means this is the last row.

- will return to the previous row by just incrementing count.

Logic:- (in high level)

```
int func(int arr[], int size, int i) {
```

```
    if (arr[i] == 0)
```

```
        return i;
```

```
    else
```

```
        return 1 + func(arr, size, i+1);
```

```
}
```

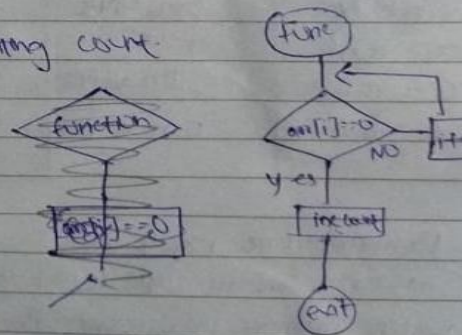
```
main() {
```

```
    arr[] = {7, 6, 5, -1, 2, 3, 6, 4, 0}; // rows
```

```
    ans = func(arr, size, 0);
```

```
    ans * row products;
```

```
}
```



Date _____

STACK FRAME

The example in my code contains array of row numbers as total ⁶ rows. assume esp starting from stack and only first 6 stack frames are displayed here.

0000 5000	Ret address	To The System	main
0000 4FFC	0000 0000	←ebp	Return main
0000 4FF8	0	arg from main	1 st (count total rows)
0000 4FF4	Ret address	To main	
0000 4FE0	0000 4FFB	←ebp	
0000 4FDC	1	Arg from func	2 nd (count total rows)
0000 4FD8	Ret address	To Countline-1	
0000 4FD4	0000 4FE0	←ebp	
0000 4FD0	2	Arg from func	3 rd (count total rows)
0000 4FCC	Ret address	To countline-2	
0000 4FC8	0000 4FD4	←ebp	
0000 4FC4	3	Arg from func	4 th (count total rows)
0000 4FC0	Ret address	To countline-3	
0000 4FB8	0000 4FC8	←ebp	
0000 4FB4	4	Arg from func	5 th (count total rows)
0000 4FB0	Ret address	To Countline-4	
0000 4FA8	0000 4FB8	←ebp	
0000 4FA4	5	Arg from func	6 th (count total rows)
0000 4FA0	Ret address	To countline-5	
0000 4FA4	0000 4FB0	←ebp	

Running time requirements

my code contains byte array of length 6, 2 variables of dword so runtime memory requirement is ^{14 byte} 14 bytes (no local data is used).
 but stack requirement depends upon length of rows till it finds 0
 but this is the stack frame of 1 procedure that code will require so 12 bytes * x length = total stack requirement.

12 * 6 = 72 bytes of stack & 14 of data seg.

Arguments	1
return address	
ebp	

ASM CODE:

TITLE Application (code.asm)

INCLUDE Irvine32.inc

.data

Rows byte 7,6,5,1,2,0

ProductsInRow dword 50

Count dword ?

message byte "The total number of products in warehouse : ",0

.code

main proc

mov ebx, 0

push ebx

call CountTotalRows

add esp, 4

mov eax, Count

mov edx, 0

mul ProductsInRow

mov edx, offset message

call writeString

call WriteDec

call crlf

call crlf

call dumpregs

exit

main endp

CountTotalRows proc

push ebp

mov ebp, esp

mov ebx, [ebp+8]

cmp Rows[ebx], 0

JE outside

```
add ebx, type byte
push ebx
```

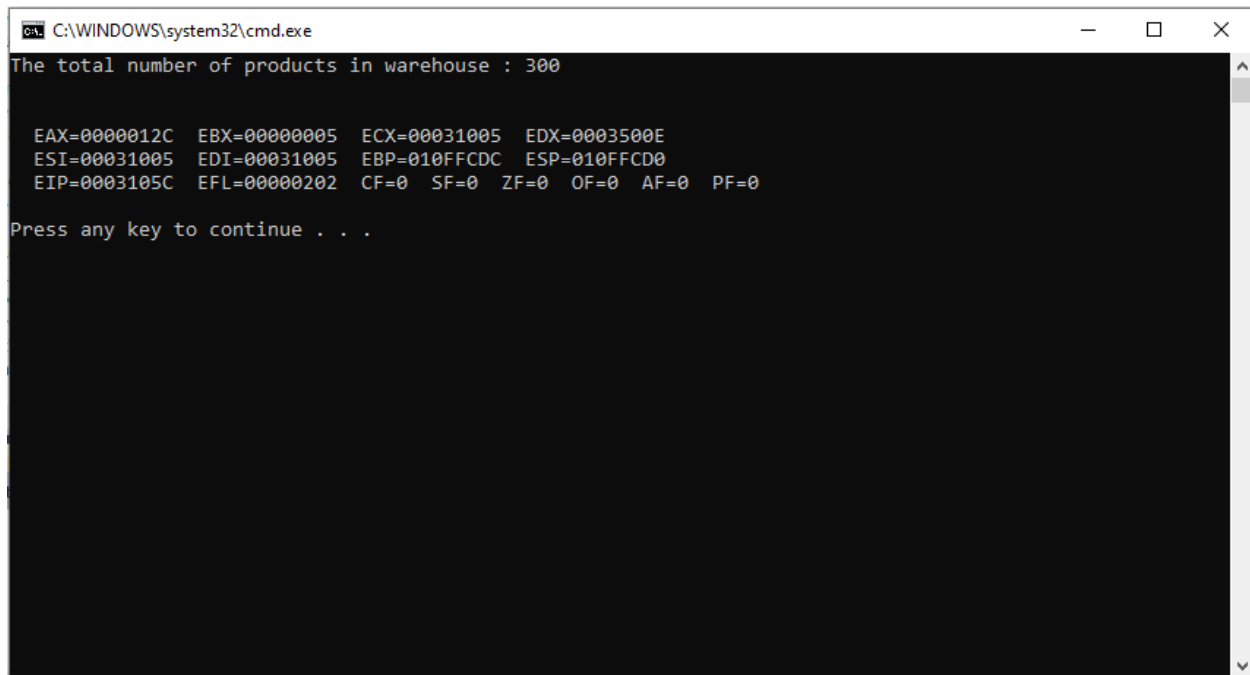
```
call CountTotalRows
add esp, 4
```

```
outside:
```

```
    inc Count
    pop ebp
    ret
```

```
CountTotalRows endp
end main
```

OUTPUT:



```
C:\WINDOWS\system32\cmd.exe
The total number of products in warehouse : 300

EAX=0000012C  EBX=00000005  ECX=00031005  EDX=0003500E
ESI=00031005  EDI=00031005  EBP=010FFCDC  ESP=010FFCD0
EIP=0003105C  EFL=00000202  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=0

Press any key to continue . . .
```