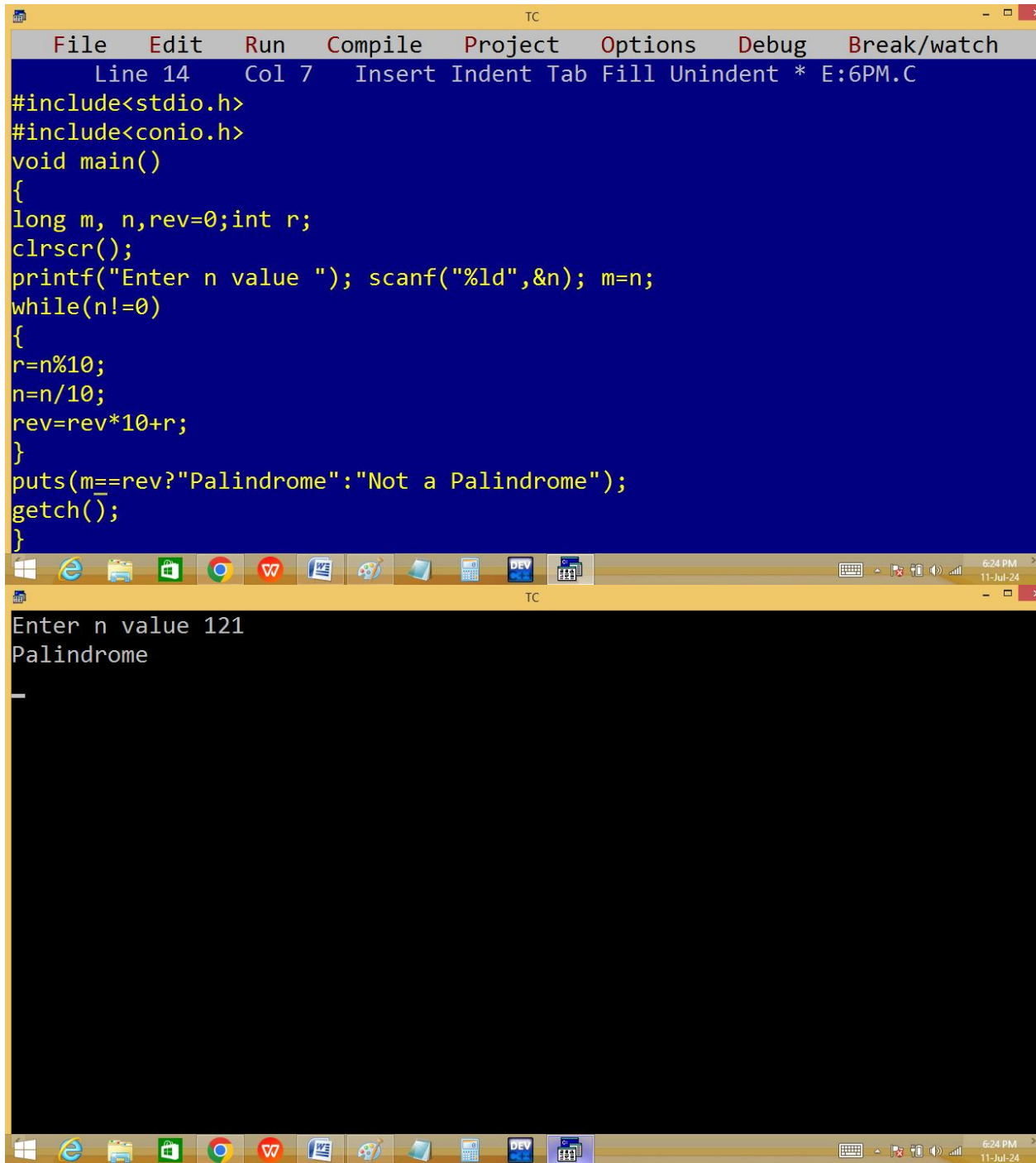


## Finding palindrome or not.

121 reverse is 121

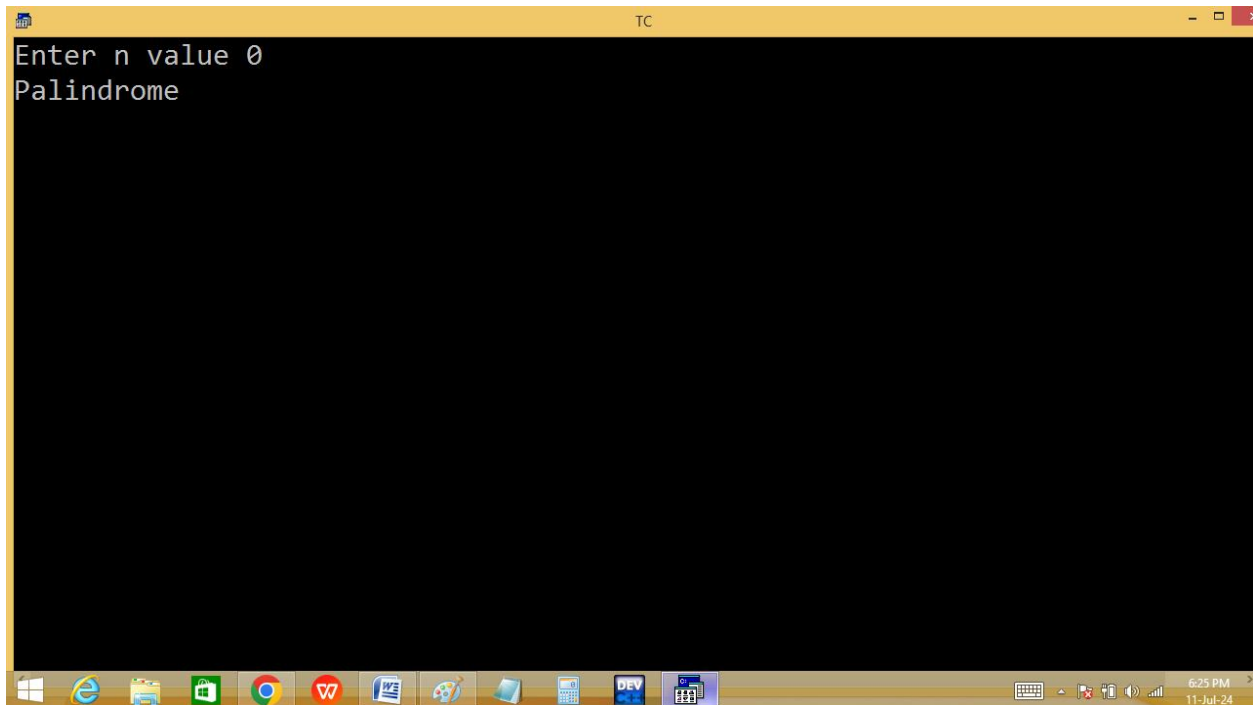


```
TC
File Edit Run Compile Project Options Debug Break/watch
Line 14 Col 7 Insert Indent Tab Fill Unindent * E:6PM.C
#include<stdio.h>
#include<conio.h>
void main()
{
long m, n, rev=0; int r;
clrscr();
printf("Enter n value "); scanf("%ld",&n); m=n;
while(n!=0)
{
r=n%10;
n=n/10;
rev=rev*10+r;
}
puts(m==rev?"Palindrome":"Not a Palindrome");
getch();
}
```

Enter n value 121  
Palindrome

```
TC
Enter n value -121
Palindrome
```

```
TC
Enter n value 1
Palindrome
-
```



$$\begin{array}{l} \text{f} \quad \frac{m}{121} \leftarrow \frac{n}{121} \\ \frac{n}{121} \div 10 = 1 \\ \frac{1}{121} \div 10 = 2 \\ \frac{1}{121} \div 10 = 1 \\ 0 \end{array}$$

$$\begin{array}{l} \text{rev} \\ 0 \times 10 + 1 = 1 \\ 1 \times 10 + 2 = 12 \\ 12 \times 10 + 1 = 121 \end{array}$$

**No to text conversion:**

**102 → One Zero Two**

**#include<stdio.h>**

**#include<conio.h>**

**void main()**

```
{  
long m, n, rev=0; int r;  
clrscr();  
printf("Enter n value "); scanf("%ld", &n);  
if(n<0) printf("-", n=-n);  
m=n;  
while(m!=0){r=m%10; m=m/10; rev=rev*10+r;} /*rev*/  
do  
{  
switch(rev%10)  
{  
case 0: printf("Zero"); break;  
case 1: printf("One"); break;  
case 2: printf("Two"); break;  
case 3: printf("Three"); break;  
case 4: printf("Four"); break;  
case 5: printf("Five"); break;
```

```
case 6: printf("Six");break;
case 7: printf("seven");break;
case 8: printf("Eight");break;
case 9: printf("Nine");
}
rev=rev/10; printf(" ");
}while(rev);
while(n%10==0&& n!=0) printf("Zero ",n/=10);
getch();
}
```

```
TC
Enter n value 123
One Two Three _
```

```
TC
Enter n value 0
Zero
```

```

Enter n value -1200340000
-One Two Zero Zero Three Four Zero Zero Zero Zero

```

```

do
{
switch(rev%10)
{
case 0: p(zero);b; ✓
case 1: p(one);b; ✓
case 2: p(two);b; ✓
case 9: p(nine);
}
rev=rev/10; p("/") ✓;
}while(rev!=0);

```

$$\frac{n}{102} \quad \frac{m}{102} \rightarrow \frac{rev}{201 \div 10 = 1 \text{ one} \checkmark}$$

$$20 \div 10 = 0 \text{ Zero} \checkmark$$

$$2 \div 10 = 0 \text{ Zero} \checkmark$$

$$\frac{n}{100} \div 10 = 0$$

$$10 \div 10 = 0$$

one - Zero Zero

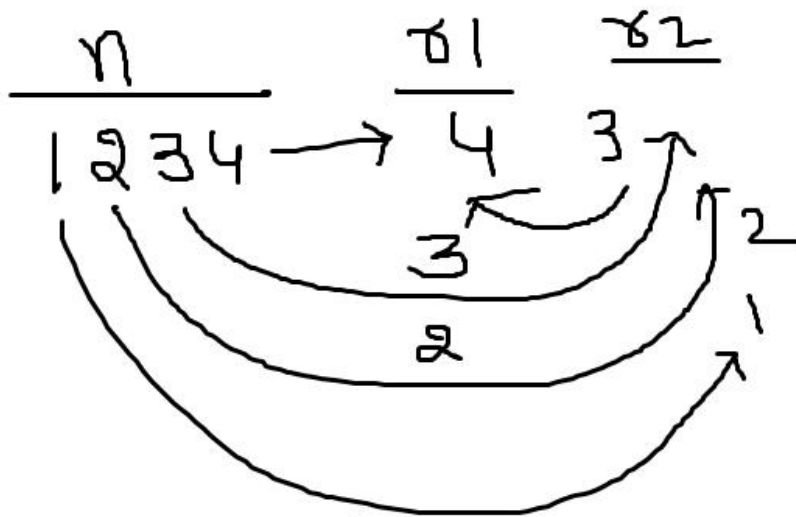
while(n%10==0 && n!=0) p(zero,n=n/10);

$$\frac{n}{0 \div 10 = 0}$$

Finding step no.

1234 → step no

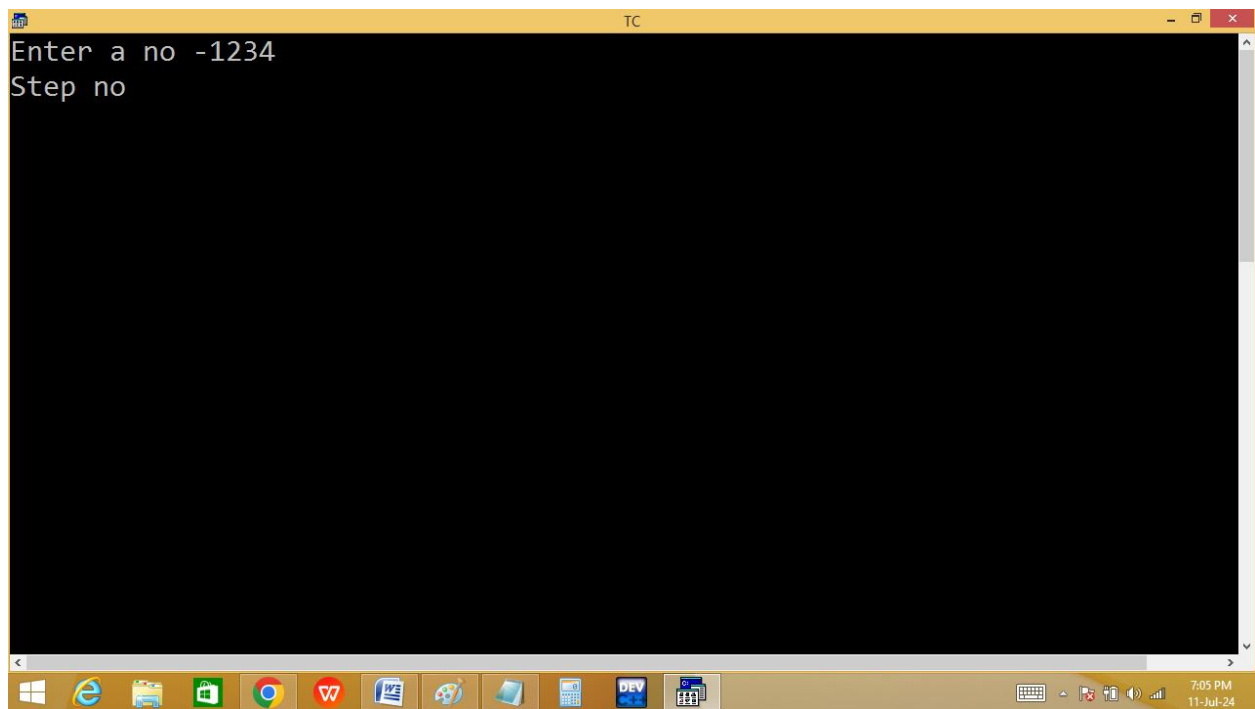
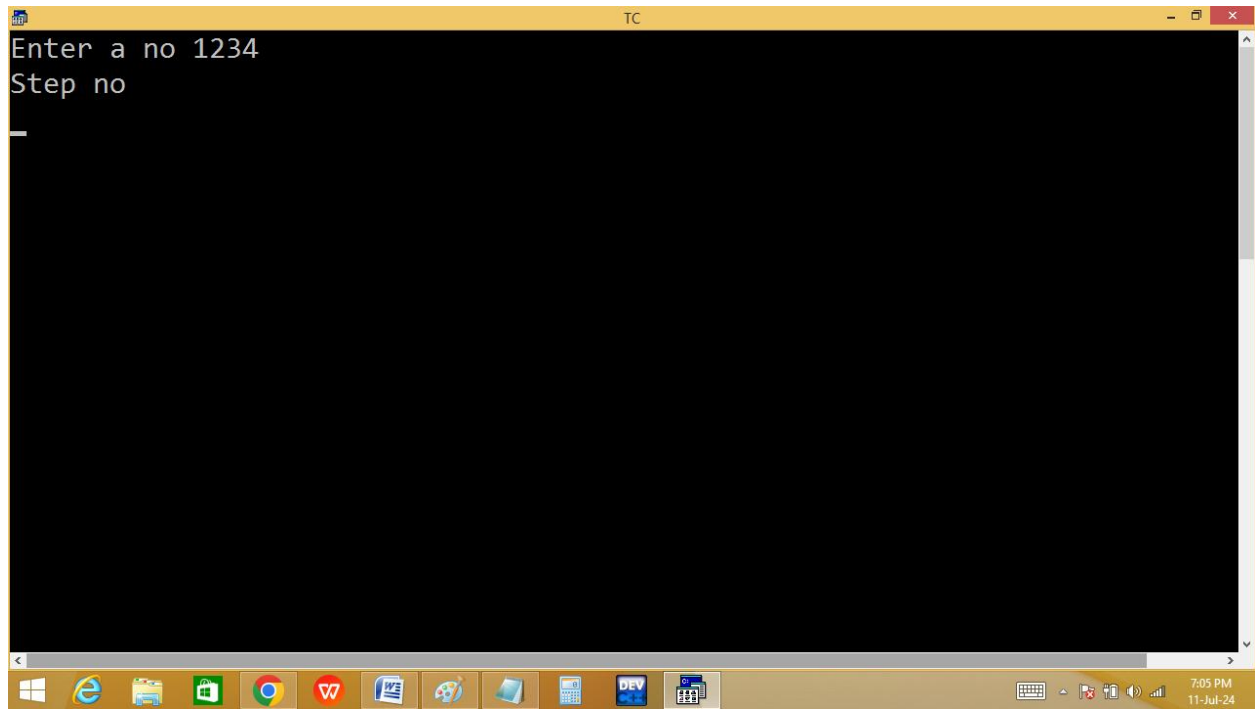
2356 → not step no





```
TC
Line 17 Col 2 Insert Indent Tab Fill Unindent * E:6PM.C
#include<stdio.h>
#include<conio.h>
void main()
{
long n,r1,r2;
clrscr();
printf("Enter a no "); scanf("%ld",&n);
r1=n%10; n=n/10;
while(n)
{
r2=n%10;
if(r1-r2==1 || r2-r1==1) { r1=r2; n=n/10; }
else { puts("Not a step no"); getch(); return; }
}
puts("Step no");
getch();
}
```

```
TC
Enter a no 1245
Not a step no
_
```



```

r1=n%10;
n=n/10;

```

$$\begin{array}{r}
 n \\
 \hline
 1234 \div 10 = 4
 \end{array}
 \quad
 \begin{array}{r}
 r1 \\
 \hline
 4
 \end{array}
 \quad
 \begin{array}{r}
 r2 \\
 \hline
 3
 \end{array}$$

An arrow points from the result '4' of the first division to the 'r1' box, and another arrow points from the 'r1' box to the 'r2' box, illustrating the iterative process of extracting digits from a number.

```

while( n )
{
r2=n%10;

if(r1-r2==1 || r2-r1==1){r1=r2; n=n/10;
}
else p("Not a step no"); return;
}
p(step no);
}

```

## for loop:

It is an entry control loop.

for is a keyword.

It is also used to repeat a program several times based on a condition.

When compared with while and do while, for loop is looking to be smart. In for it is compulsory to maintain two semicolons.

**For works without condition also and default condition is always 1 i.e. true.**

**Generally for loop is having 3 expressions.**

- 1. Initialization**
- 2. Test condition / expression**
- 3. Increment/decrement / updation**

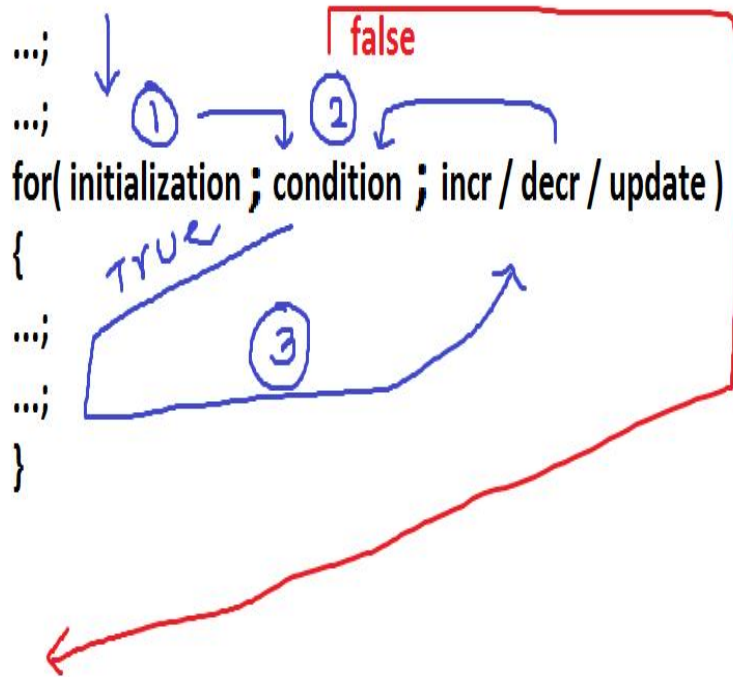
**At first entry of for loop the initialization part is executed and later the test condition is checked. If the condition is true then the for block statements are executed. After completion of the block, the increment or decrement part is executed. Later once again the test condition is evaluated. If it is true then once again for block statements are executed. Like this the process is continued until the condition becomes false. Here the**

**initialization part is executed only once, at the time of loop beginning.**

**It is mandatory to maintain 2 semicolon ( ; ) in a for loop.**

**If the for loop is having more than three expressions, it is mandatory to separate the expressions with , separator.**

**If the for loop is having less than three expressions, then leave the expressions with empty semicolon.**



```
for(exp ; exp ; exp )
```

```
{  
}
```

```
for(exp, exp ; exp ; exp,exp)
```

```
{  
}
```

```
for( ; exp ; )
```

```
{  
}
```

```
for( ; ; )
```

```
{  
}
```