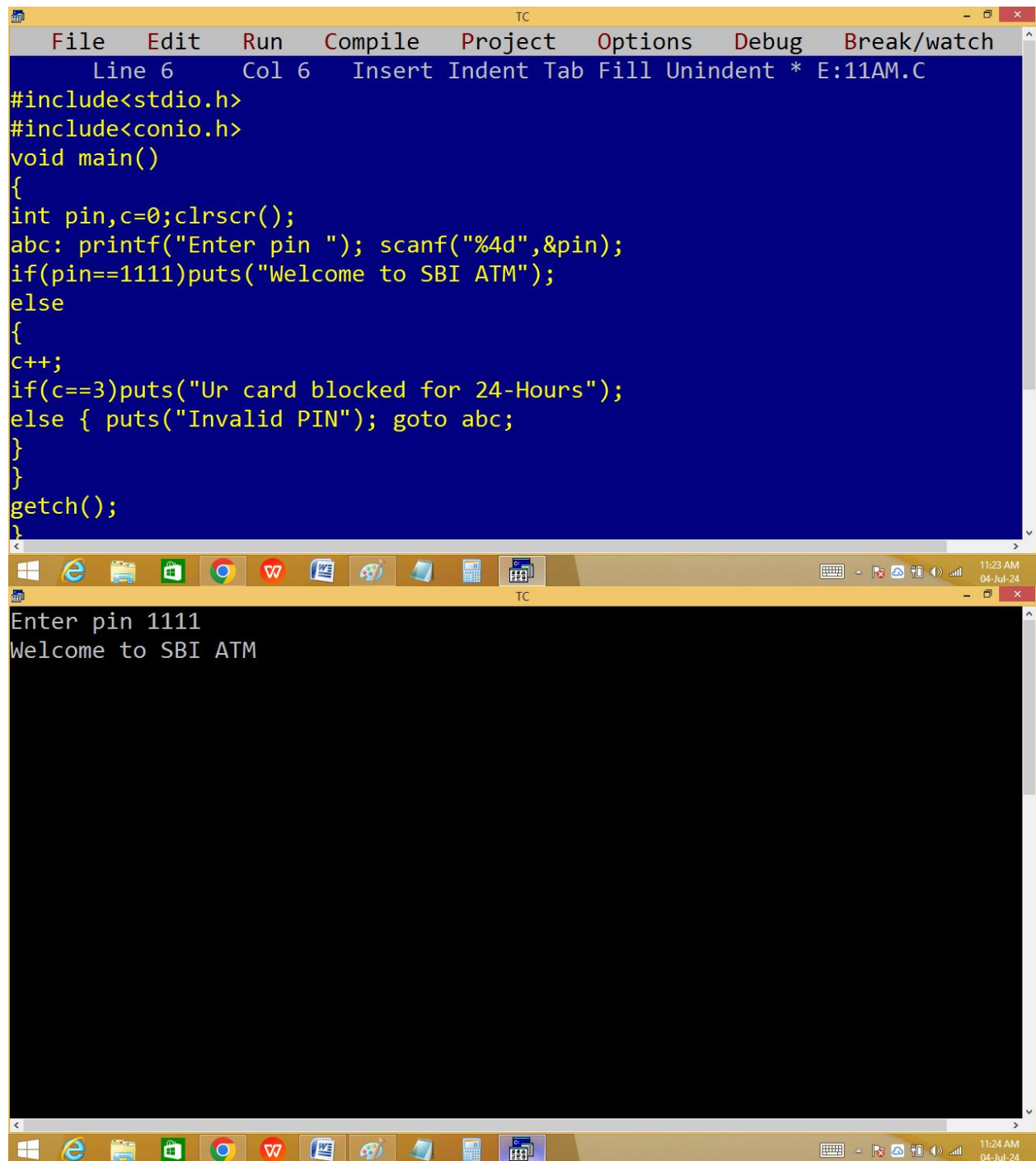


## ATM Pin Validation:

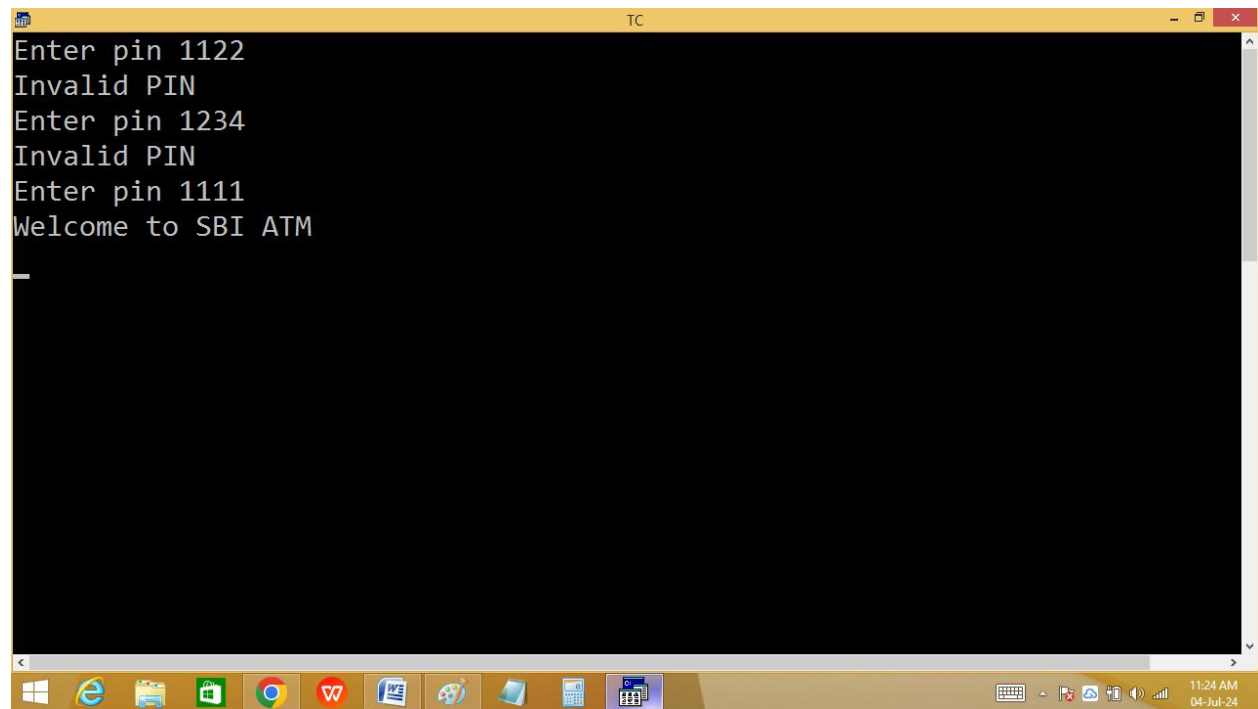


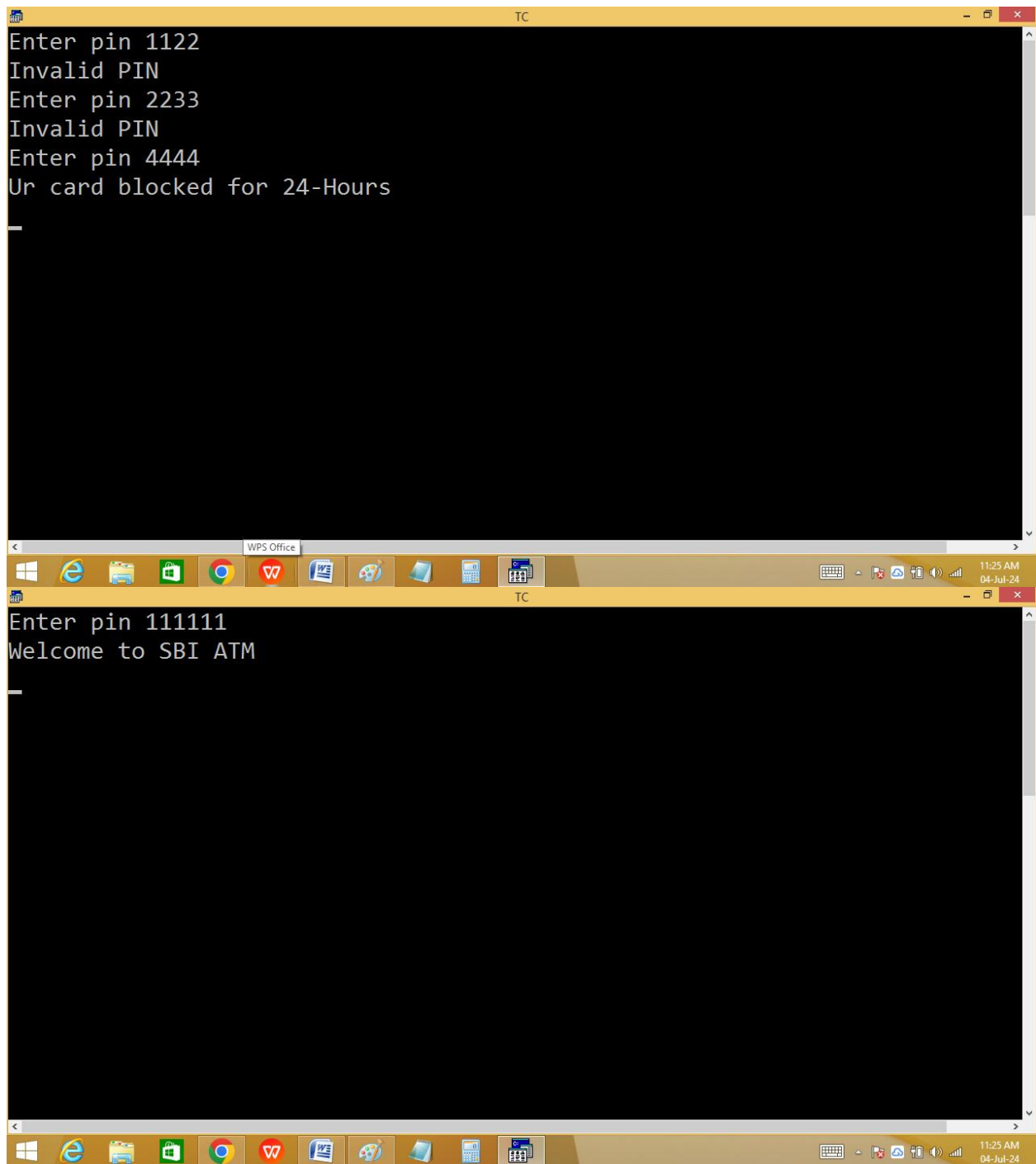
The image shows a screenshot of a Turbo C++ (TC) IDE. The top window displays the source code for an ATM pin validation program. The code includes headers for `stdio.h` and `conio.h`, and defines a `main` function. Inside `main`, it declares `pin` and `c`, clears the screen, and prompts the user to enter a pin. It uses `scanf` to read the pin. If the pin is 1111, it prints "Welcome to SBI ATM". Otherwise, it increments `c` and checks if `c` is 3. If `c` is 3, it prints "Ur card blocked for 24-Hours". Otherwise, it prints "Invalid PIN" and jumps to the `abc` label. The program ends with `getch()`.

The bottom window shows the program's execution. It displays the prompt "Enter pin 1111" and the output "Welcome to SBI ATM".

```
File Edit Run Compile Project Options Debug Break/watch
Line 6 Col 6 Insert Indent Tab Fill Unindent * E:11AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
int pin,c=0;clrscr();
abc: printf("Enter pin "); scanf("%d",&pin);
if(pin==1111)puts("Welcome to SBI ATM");
else
{
c++;
if(c==3)puts("Ur card blocked for 24-Hours");
else { puts("Invalid PIN"); goto abc;
}
}
getch();
}
```

Enter pin 1111  
Welcome to SBI ATM





**Read a stu id, name, 6 sub marks and find the tot, avg and grade.**

**#include<stdio.h>**

```
#include<conio.h>

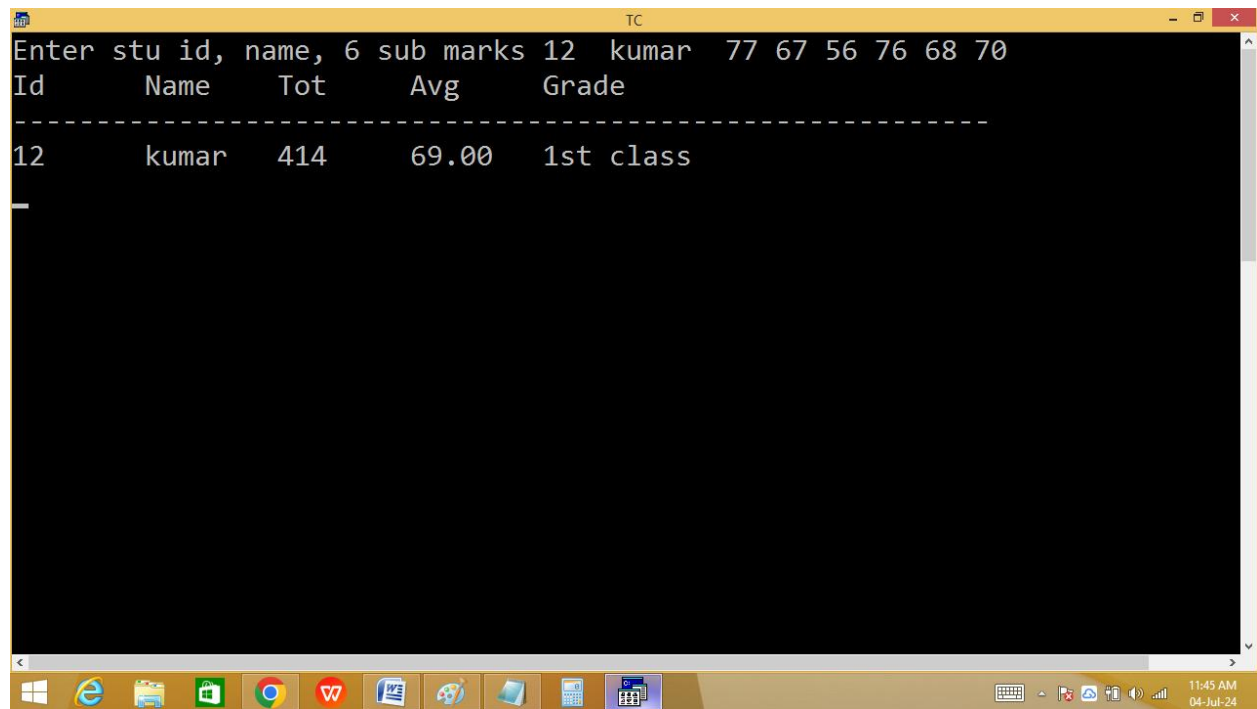
void main()
{
    int id, tel,eng,hin,mat,sci,soc,tot;
    char name[20];
    float avg;
    clrscr();
    printf("Enter stu id, name, 6 sub marks ");
    scanf("%d%s%d%d%d%d%d", &id, name, &tel, &eng, &hin, &mat, &sci, &soc);
    tot=tel+eng+hin+mat+sci+soc;
    avg=tot/6.0;
    puts("Id\tName\tTot\tAvg\tGrade");
    puts("-----");
    printf("%d\t%s\t%d\t%.2f\t", id, name, tot, avg);
    if(tel>=35&&eng>=35&&hin>=35&&mat>=35&&sci>=35
    &&soc>=35)
```

```
{  
    if(avg>=75)puts("Distinction");  
    else if(avg>=60)puts("1st class");  
    else if(avg>=50)puts("2nd class");  
    else puts("3rd class");  
}  
else puts("Fail");  
getch();  
}
```

```
TC
Enter stu id, name, 6 sub marks 2 kumari 66 56 30 45 23 55
Id      Name    Tot      Avg      Grade
-----
2       kumari  275     45.83    Fail

TC
Enter stu id, name, 6 sub marks 1 krish 88 99 90 98 98 99
Id      Name    Tot      Avg      Grade
-----
1       krish   572     95.33    Distinction
```

```
TC
Enter stu id, name, 6 sub marks 12 kumar 77 67 56 76 68 70
Id      Name    Tot      Avg      Grade
-----
12      kumar    414      69.00    1st class
```



```
TC
Enter stu id, name, 6 sub marks 3 bablu 55 65 45 55 51 61
Id      Name    Tot    Avg    Grade
-----
3      bablu    332    55.33  2nd class

TC
Enter stu id, name, 6 sub marks 4 pinky 35 40 41 44 50 40
Id      Name    Tot    Avg    Grade
-----
4      pinky    250    41.67  3rd class
```

Ternary / Conditional operator( ? : )



It is similar to if else / ladder if in working style.

It allows to complete if else / ladder if in a single statement.

When we are working with if else/ladder if it is going to take more than one line of statements. Ternary operator is going to finish the same task in a single statement.

**But the difference between if ...else and ternary operator is ternary operator supports only one statement at a time and if supports any number of statements.**

It is having 3 expressions. Hence it is called ternary operator.

It is starting with a condition. Hence it is called conditional operator.

## Syntax:

condition ? true statement : false statement ;  
          ↑                  ↑                  ↑  
exp1/op1    exp2/op2    exp3/op3

If condition true, statement after ? executed.

If condition false, statement after : is executed.

When compared with if else, conditional operator **performance is high**.

**Eg:** Finding big in two no's using ternary operator.

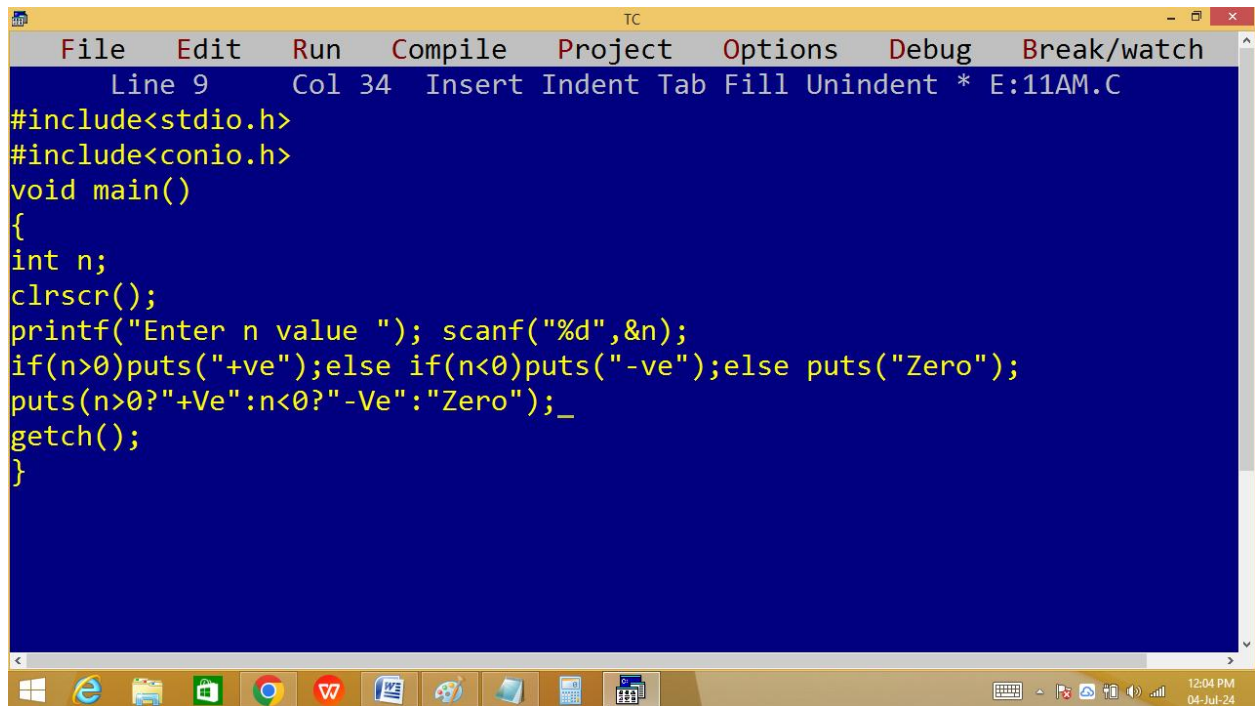
```
TC
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,big;
clrscr();
printf("Enter a, b values "); scanf("%d %d",&a, &b);
printf("%d is big\n",a>b?a:b);
puts(a>b?"a is big":"b is big");
a>b?puts("a is big"):puts("b is big");
big = a>b?a:b;
printf("%d is big\n",big);
puts(a>b?"a is big":b>a?"b is big":"Both are equal");
getch();
}
```

```
TC
Enter a, b values 4 4
4 is big
b is big
b is big
4 is big
Both are equal
```

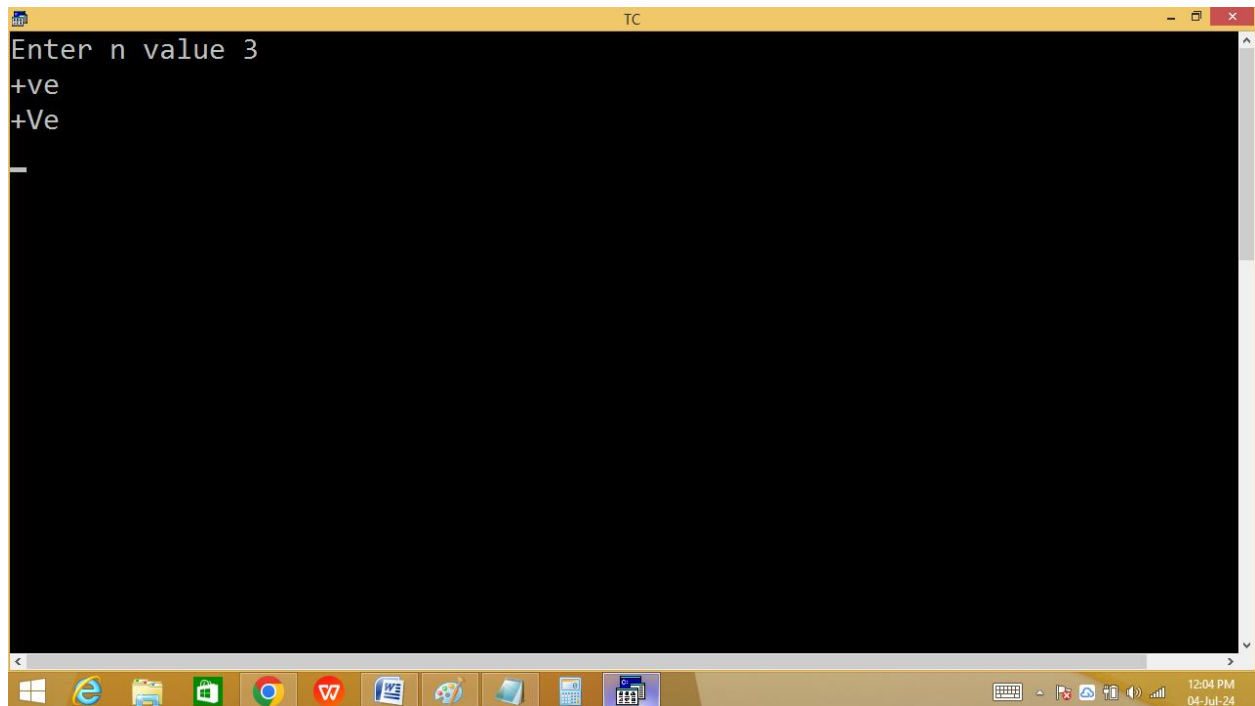
```
TC
Enter a, b values 1 2
2 is big
b is big
b is big
2 is big
b is big
```

```
TC
Enter a, b values 2 1
2 is big
a is big
a is big
2 is big
a is big
```

## Finding +ve /-ve / 0 using ternary op:

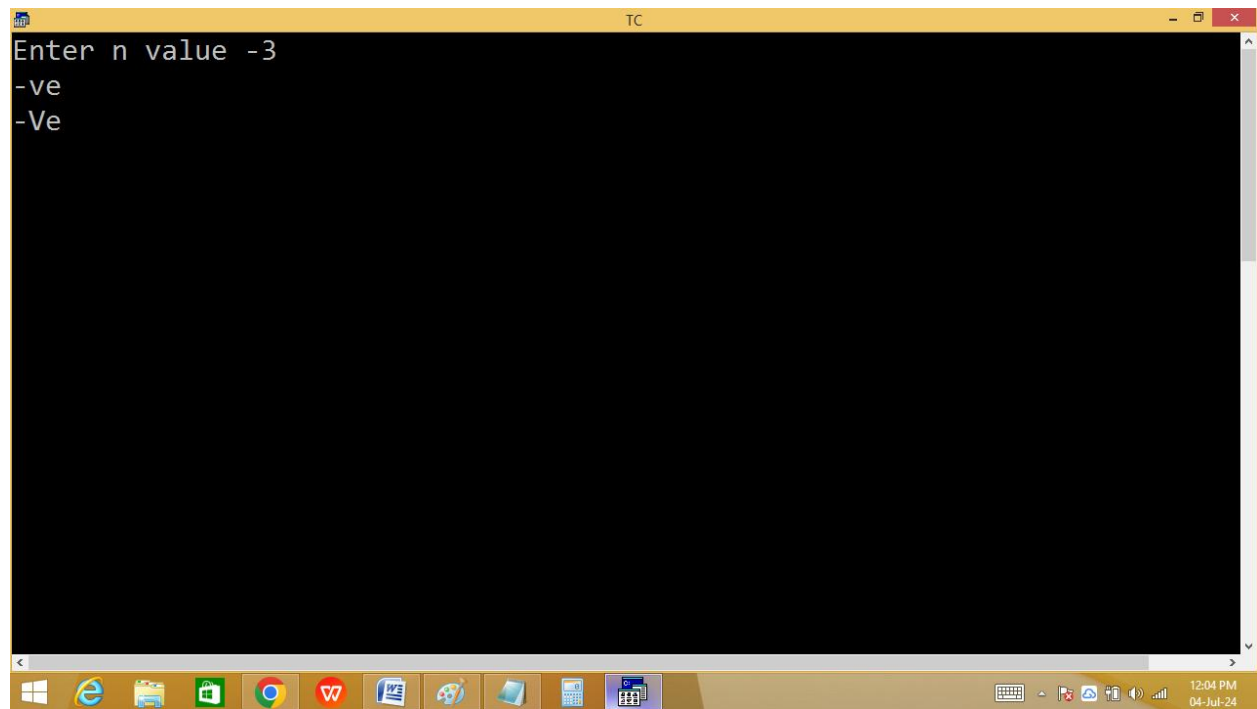


```
TC
File Edit Run Compile Project Options Debug Break/watch
Line 9 Col 34 Insert Indent Tab Fill Unindent * E:11AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
clrscr();
printf("Enter n value "); scanf("%d",&n);
if(n>0)puts("+ve");else if(n<0)puts("-ve");else puts("Zero");
puts(n>0?" +Ve":n<0?" -Ve":"Zero");_
getch();
}
```

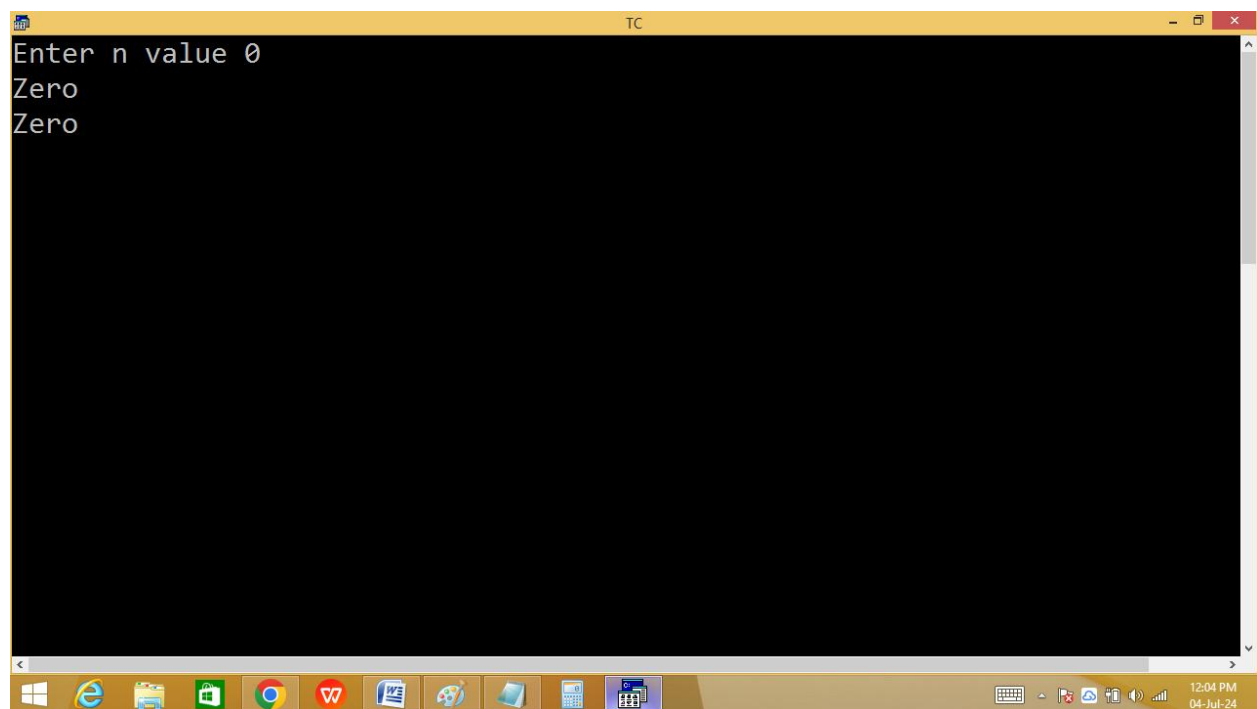


```
TC
Enter n value 3
+ve
+Ve
```

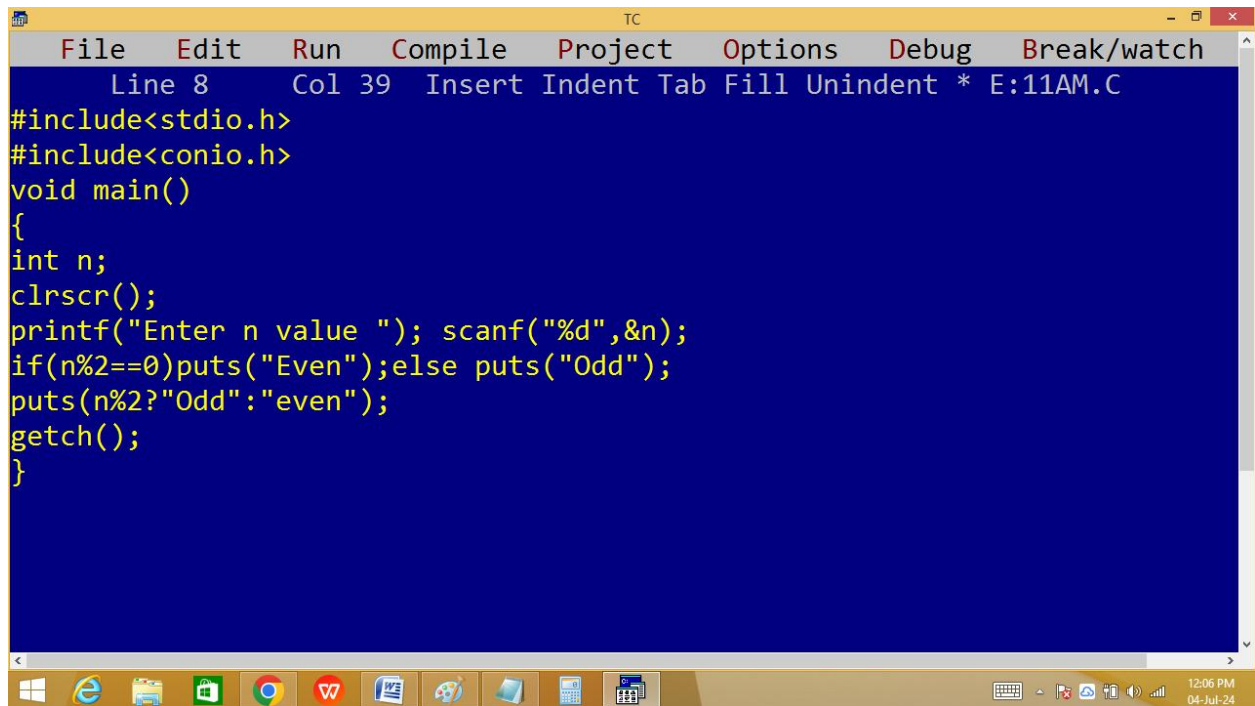
```
TC
Enter n value -3
-ve
-Ve
```



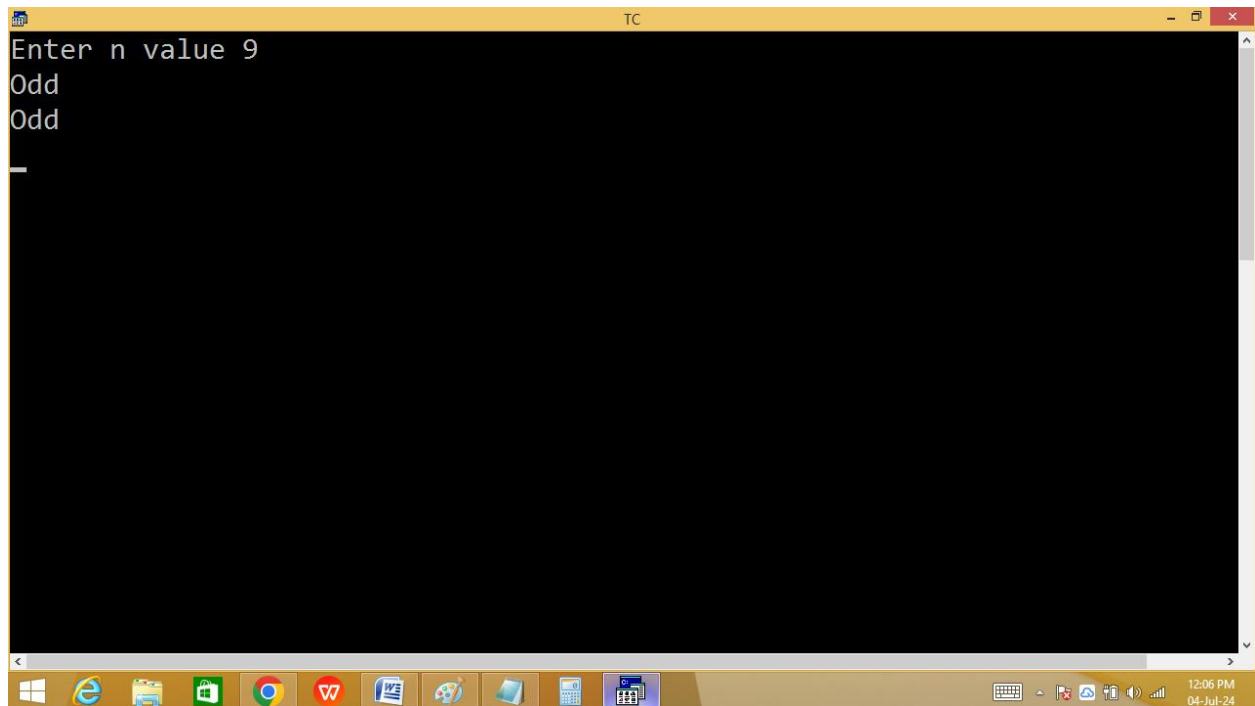
```
TC
Enter n value 0
Zero
Zero
```



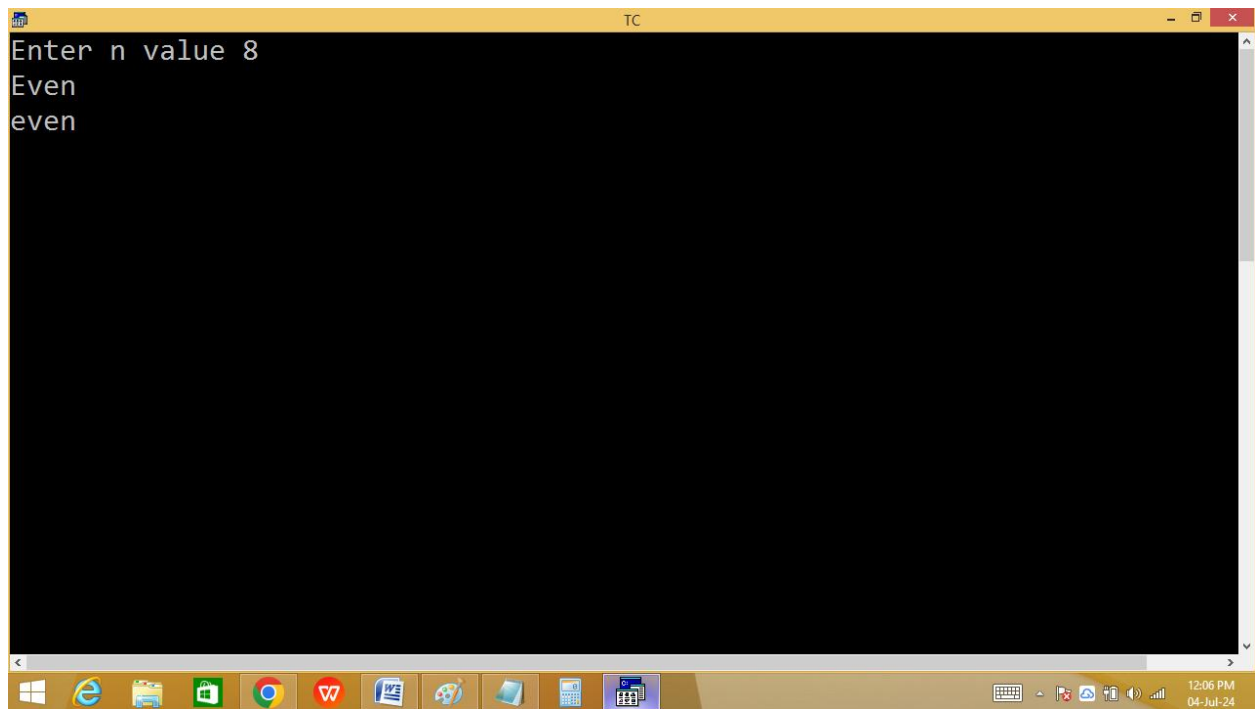
## Finding even/odd using ternary op:



```
TC
File Edit Run Compile Project Options Debug Break/watch
Line 8 Col 39 Insert Indent Tab Fill Unindent * E:11AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
clrscr();
printf("Enter n value "); scanf("%d",&n);
if(n%2==0)puts("Even");else puts("Odd");
puts(n%2?"Odd":"even");
getch();
}
```

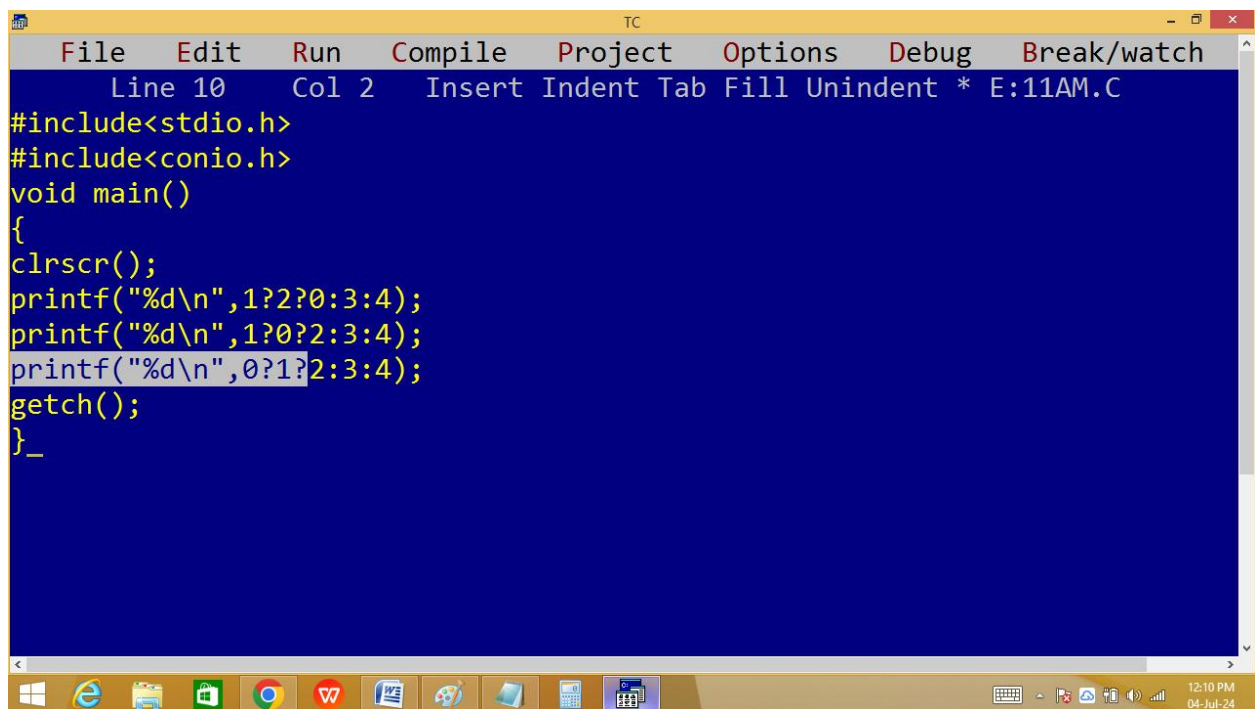


```
TC
Enter n value 9
Odd
Odd
```



A screenshot of a Turbo C++ (TC) console window. The window has a yellow title bar with the text "TC". The main area is black with white text. The text displayed is: "Enter n value 8", "Even", and "even". The Windows taskbar is visible at the bottom with various icons and a system clock showing 12:06 PM on 04-Jul-24.

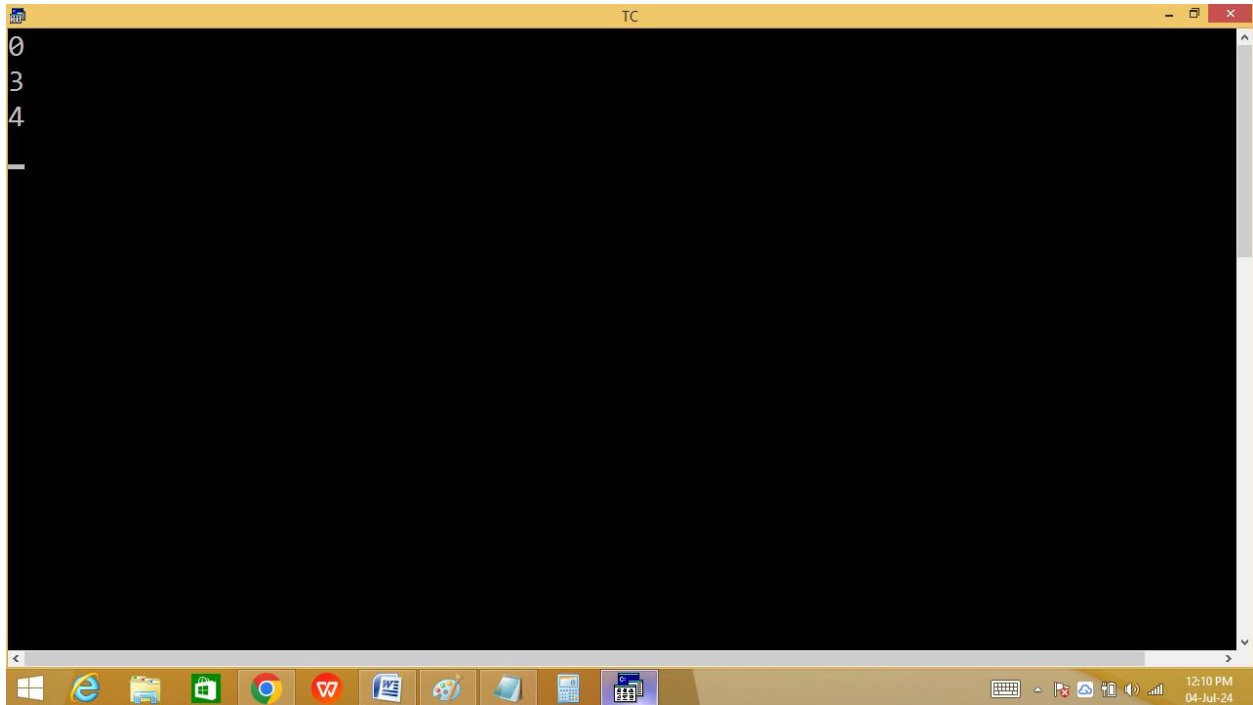
```
Enter n value 8
Even
even
```



A screenshot of a Turbo C++ (TC) editor window. The window has a yellow title bar with the text "TC". Below the title bar is a menu bar with options: File, Edit, Run, Compile, Project, Options, Debug, and Break/watch. Below the menu bar is a status bar showing "Line 10 Col 2 Insert Indent Tab Fill Unindent \* E:11AM.C". The main area is blue with yellow text. The code displayed is: "#include<stdio.h>", "#include<conio.h>", "void main()", "{", "clrscr();", "printf(\"%d\\n\",1?2?0:3:4);", "printf(\"%d\\n\",1?0?2:3:4);", "printf(\"%d\\n\",0?1?2:3:4);", "getch();", "}\_". The Windows taskbar is visible at the bottom with various icons and a system clock showing 12:10 PM on 04-Jul-24.

```
File Edit Run Compile Project Options Debug Break/watch
Line 10 Col 2 Insert Indent Tab Fill Unindent * E:11AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("%d\\n",1?2?0:3:4);
printf("%d\\n",1?0?2:3:4);
printf("%d\\n",0?1?2:3:4);
getch();
}_
```



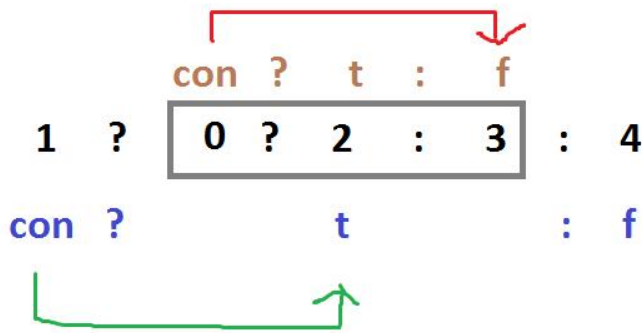


con ? t : f  
1 ? 2 ? 0 : 3 : 4  
con ? t : f

The diagram illustrates the evaluation of a nested ternary operator expression. The expression is written as 'con ? t : f' on the first line, '1 ? 2 ? 0 : 3 : 4' on the second line, and 'con ? t : f' on the third line. A green arrow starts from the 'con' in the first line, points down to the '2' in the second line, and then continues to the 't' in the third line. Another green arrow starts from the 't' in the first line, points down to the '0' in the second line, and then continues to the 'f' in the third line. A third green arrow starts from the '1' in the second line, points down to the 'con' in the third line, and then continues to the 't' in the third line. A fourth green arrow starts from the '2' in the second line, points down to the '0' in the third line, and then continues to the 'f' in the third line. A fifth green arrow starts from the '0' in the second line, points down to the 't' in the third line, and then continues to the 'f' in the third line. A sixth green arrow starts from the '3' in the second line, points down to the 'f' in the third line, and then continues to the 'f' in the third line. A seventh green arrow starts from the '4' in the second line, points down to the 'f' in the third line, and then continues to the 'f' in the third line.

if( 1 )  
{  
if( 2 ) p( 0 );  
else p( 3 );  
}  
else p( 4 );

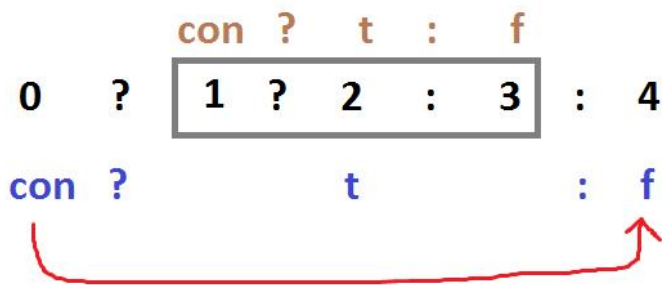
The diagram illustrates the evaluation of a nested if-else statement. The code is written as 'if( 1 )' on the first line, '{' on the second line, 'if( 2 ) p( 0 );' on the third line, 'else p( 3 );' on the fourth line, '}' on the fifth line, and 'else p( 4 );' on the sixth line. A green arrow starts from the '1' in the first line, points down to the '{' in the second line, and then continues to the 'if( 2 )' in the third line. Another green arrow starts from the '2' in the third line, points down to the 'p( 0 );' in the third line, and then continues to the 'p( 3 );' in the fourth line. A third green arrow starts from the '3' in the fourth line, points down to the 'p( 3 );' in the fourth line, and then continues to the 'p( 4 );' in the sixth line. A fourth green arrow starts from the '4' in the sixth line, points down to the 'p( 4 );' in the sixth line, and then continues to the 'p( 4 );' in the sixth line.



```

if( 1 )
{
    if( 0 ) p( 2 );
    else    p( 3 );
}
else      p( 4 );

```



```

if( 0 )
{
    if( 1 ) p( 2 );
    else    p( 3 );
}
else      p( 4 );

```

## Switch statement:

It is a selection statement.

It is used to execute one case of statements from no of cases according to the switch expression value matched with case expression value. In switch the program is jumped to matching case like the go to label.

It is similar to ladder if in working style.

Switch performance is high when compared with ladder if because of it jumps to matching case.

### Syntax:

```
switch(condition / expression)
```

```
{
```

```
case constexp1:
```

```
statements;
```

```
break;
```

```
case constexp2:
```

```
statements;
```

```
break;
```

```
case constexprN:  
statements;  
break;  
[ default: statements; ]  
}
```

Here switch, case, break, default are the keywords.

In between case and case expression / value at least one space should be provided. **Otherwise it will become a label.**

case expression/value should be a **constant integer/char value**. i.e. float / string not allowed.

One case contains one expression only.

**case expression doesn't contain any separators like , . etc.**

case expression should be end with **: (colon)**

Each case should be separated with break keyword. Otherwise remaining cases also executed.

**Duplicate cases not allowed.**

**default is similar to the else and all cases are failed then default statements are executed. Default is**

optional and we can declare it anywhere in our switch.

**Outside case expressions not considered in switch.**