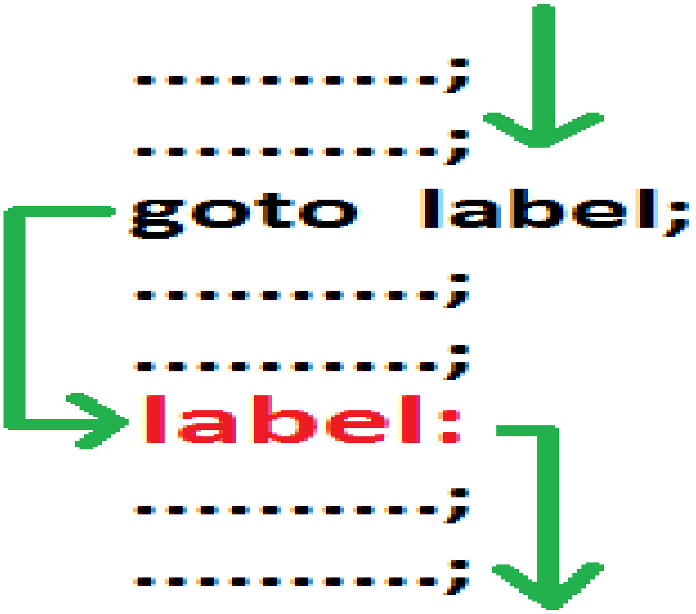
goto label / jumping statement

It is used to transfer program execution

from one place to another place [label].

In this process it is jumping from one area to another without any condition. Hence it is also called **unconditional** jumping statement.

Syntax:



```
..........;
..........;
goto label;
..........;
..........;
label:
..........;
..........;
```

Here **goto** is a keyword.

Label is an identifier is used to identify the area[line].

Every label should be end with : (colon)

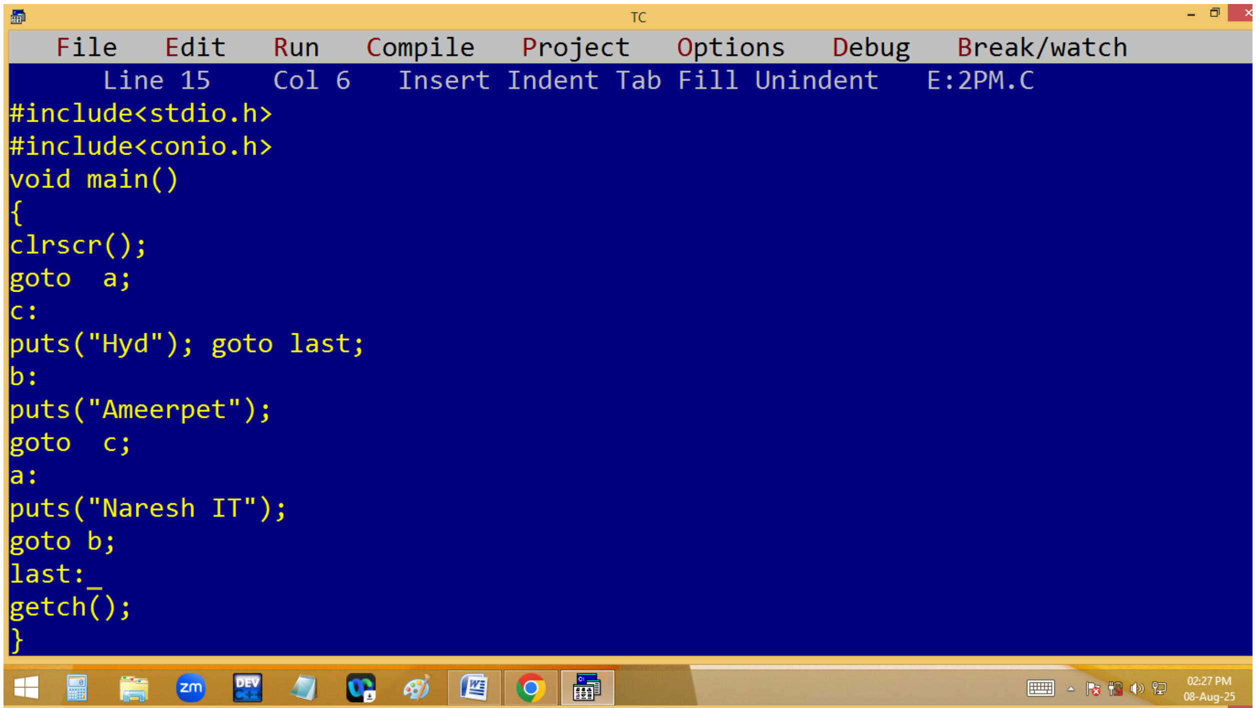
Keywords not allowed in labels i.e. label should be user defined.

Duplicate labels not allowed.

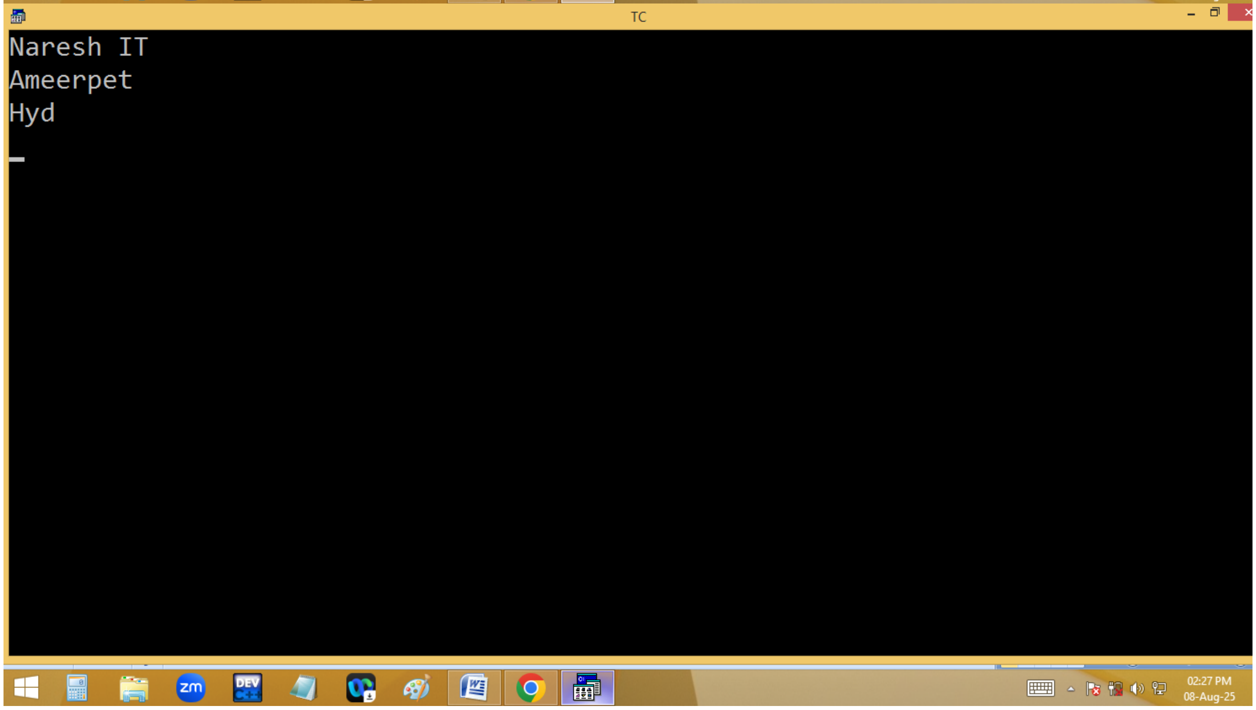
There is no space between go and to.

Label naming rules are similar to the identifier rules.

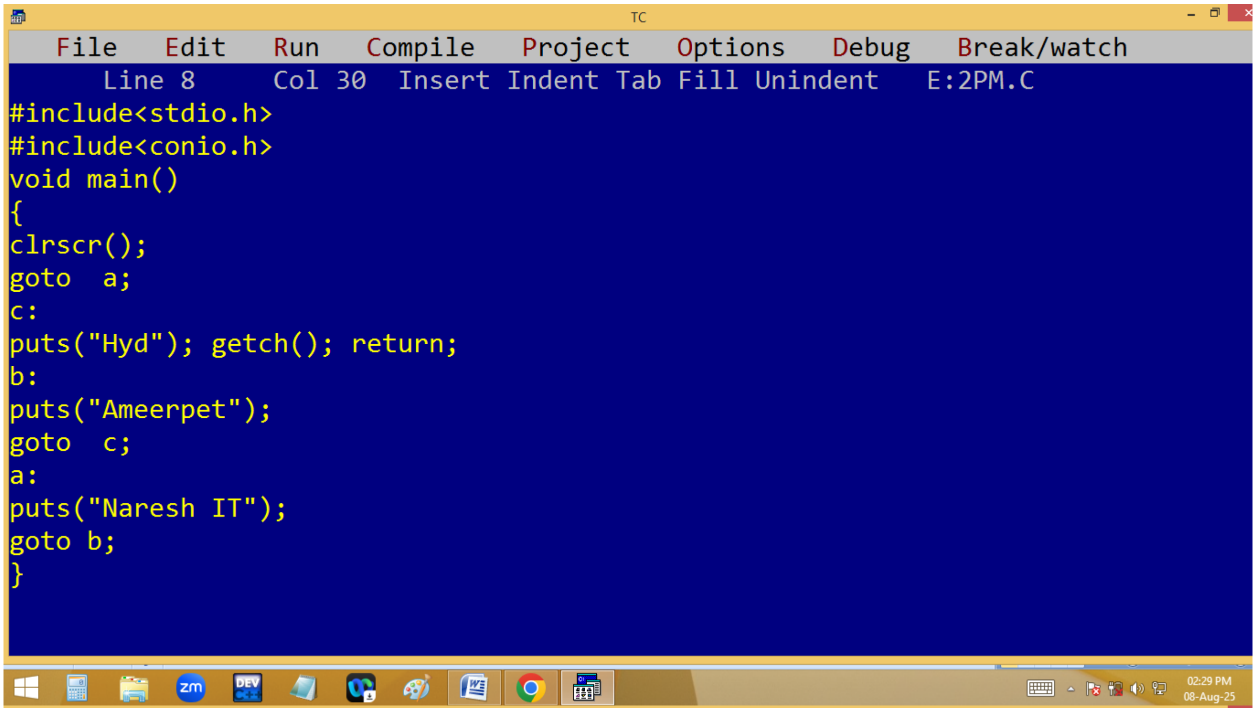
Note: goto label working style is similar to loops some times.



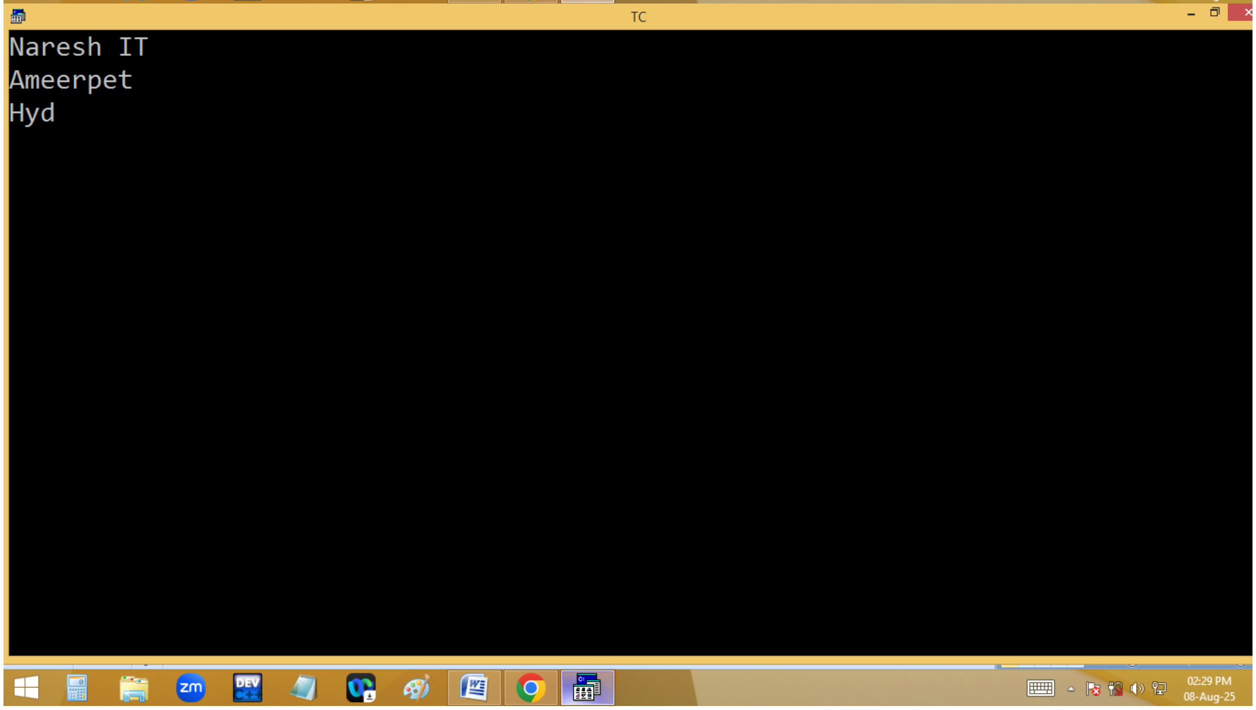
```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
goto  a;
c:
puts("Hyd"); goto last;
b:
puts("Ameerpet");
goto  c;
a:
puts("Naresh IT");
goto b;
last:
getch();
}
```



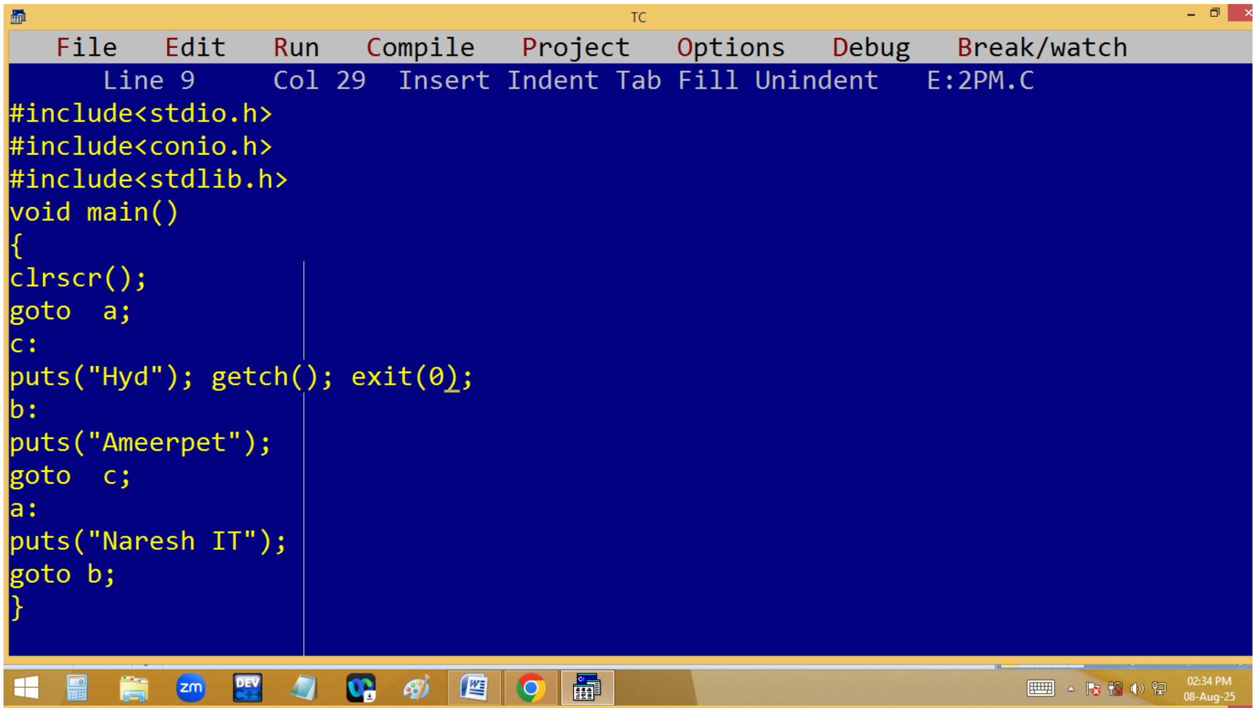
```
Naresh IT
Ameerpet
Hyd
```


TC
File Edit Run Compile Project Options Debug Break/watch
Line 8 Col 30 Insert Indent Tab Fill Unindent E:2PM.C
02:29 PM
08-Aug-25

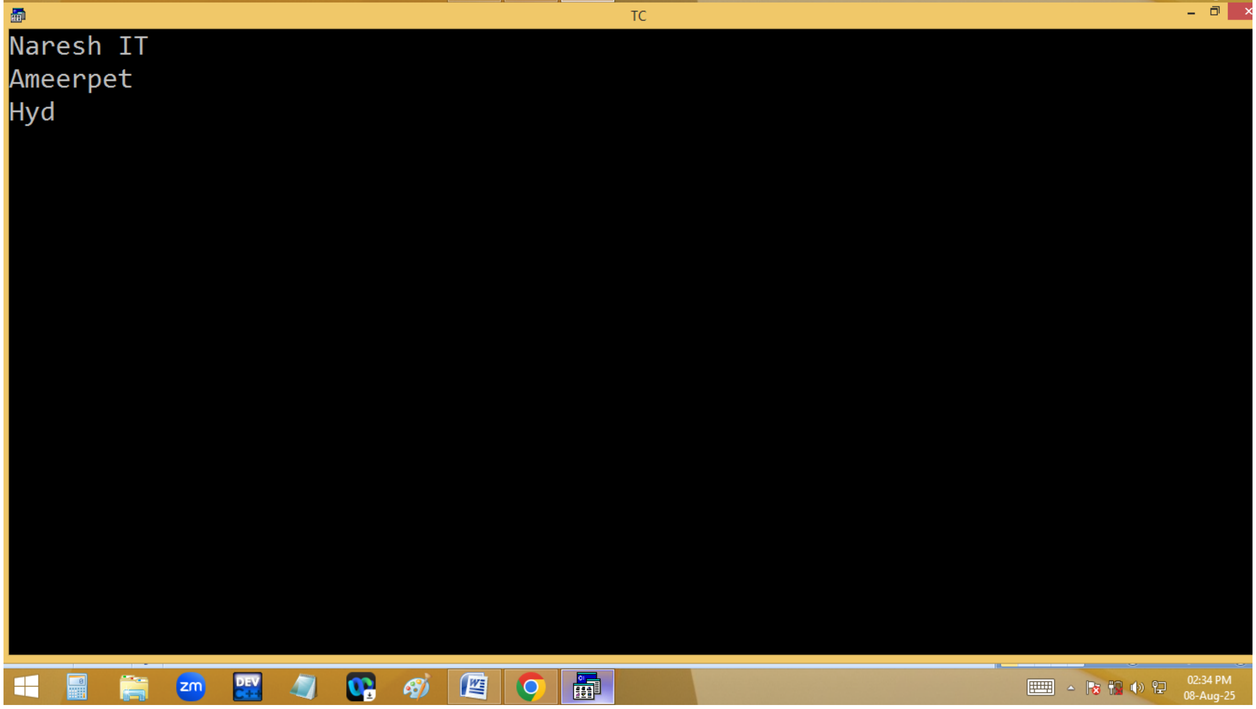
```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
goto  a;
c:
puts("Hyd"); getch(); return;
b:
puts("Ameerpet");
goto  c;
a:
puts("Naresh IT");
goto b;
}
```



```
Naresh IT
Ameerpet
Hyd
```


TC
File Edit Run Compile Project Options Debug Break/watch
Line 9 Col 29 Insert Indent Tab Fill Unindent E:2PM.C
02:34 PM
08-Aug-25

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
clrscr();
goto  a;
c:
puts("Hyd"); getch(); exit(0);
b:
puts("Ameerpet");
goto  c;
a:
puts("Naresh IT");
goto b;
}
```



```
Naresh IT
Ameerpet
Hyd
```

Conditional control statements:

They are working based on a condition.

C provides two types of conditional control statements.

1. If
2. Switch

if:

if is a keyword.

It is a decision making statement.

It is used to check the given condition / expression is true or false.

Note: In C & C++ other than 0 anything is 1 i.e. true.

We are having 4 types of if conditions.

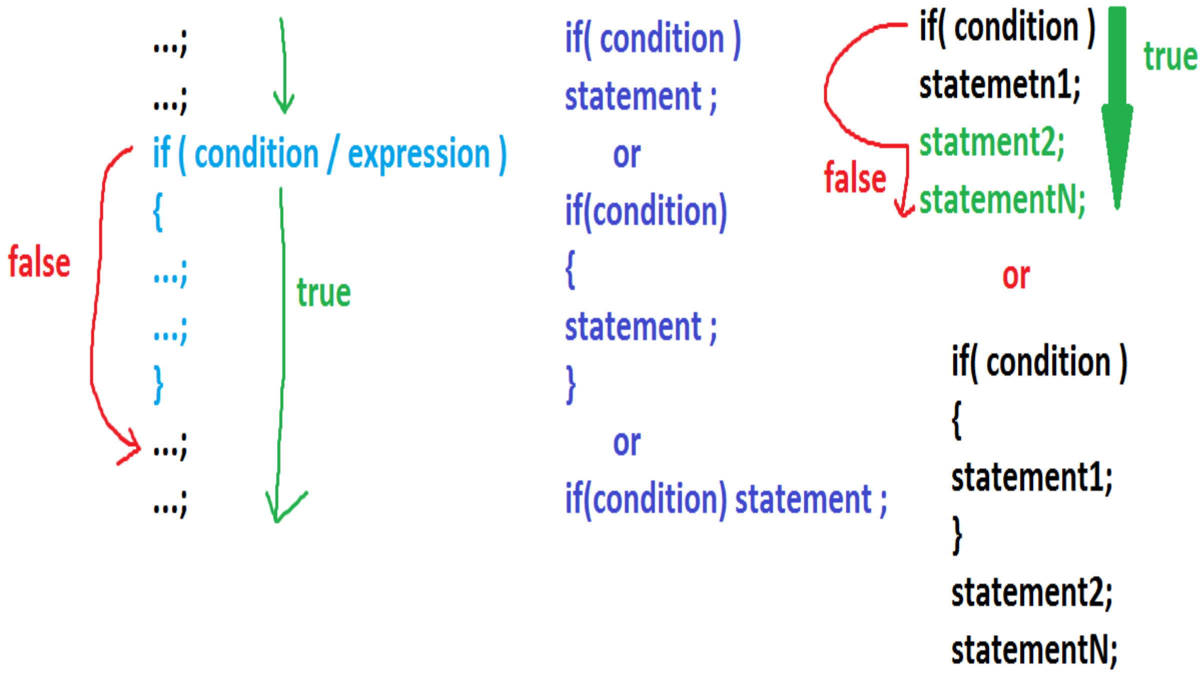
1. Simple if
2. If..else
3. If..else if ladder
4. Nested if

Simple if: When the program is having only one condition then prefer simple if.

If condition true statements in if block { } are executed and later outside statements executed.

If condition false only the outside statements are executed.

Syntax:



```
...;
...;
if ( condition / expression )
{
...;
...;
}
...;
...;
```

```
if( condition )
statement ;
    or
if(condition)
{
statement ;
}
    or
if(condition) statement ;
```

```
if( condition )
statemetn1;
statment2;
statementN;
```

or

```
if( condition )
{
statement1;
}
statement2;
statementN;
```