C - PROGRAM STRUCTURE

It illustrates[Describes] how to write a program in c-language.

Every programming language is having a particular structure and we should have to follow this structure.

C-Programming structure is divided into the following parts.

- **[documentation section]**
- **Header files / Proto types / Preprocessor**
- **[global variables]**
- **[function declarations & definitions]**
- **void main() / main() / int main()**
- **Other statements.**

Generally documentation section consists of program headings, definitions etc and They should be represented with comments.

The statements that are enclosed in between /* and */ are called comments.

Comments never participate in program execution. They are only for user understandability or display purpose.

C-Language supports comment block only.

Eg:

```
/*
………;
………;
*/
```

C++ supports comment block and single line comments.

Eg: // …………………….

Header files consists of function definitions, global variables, macros etc.

We can declare the header files at any place of our program. But before going to use the relevant function, its header file should be declared. It is recommended to declare the header files at the top of the program.

Every header file should be started with **#include**. Here **#** is a **preprocessor** indicator.

We can place header files in angled brackets

< > or double quotes **“ “**.

Header file never ends with **semicolon**(**;**).

Note: In C^{++}, we should have to declare header files at the top only.

The variables that are declared before main() or top of the program are called **global variables** and they can be accessed from anywhere in our program. They are optional.

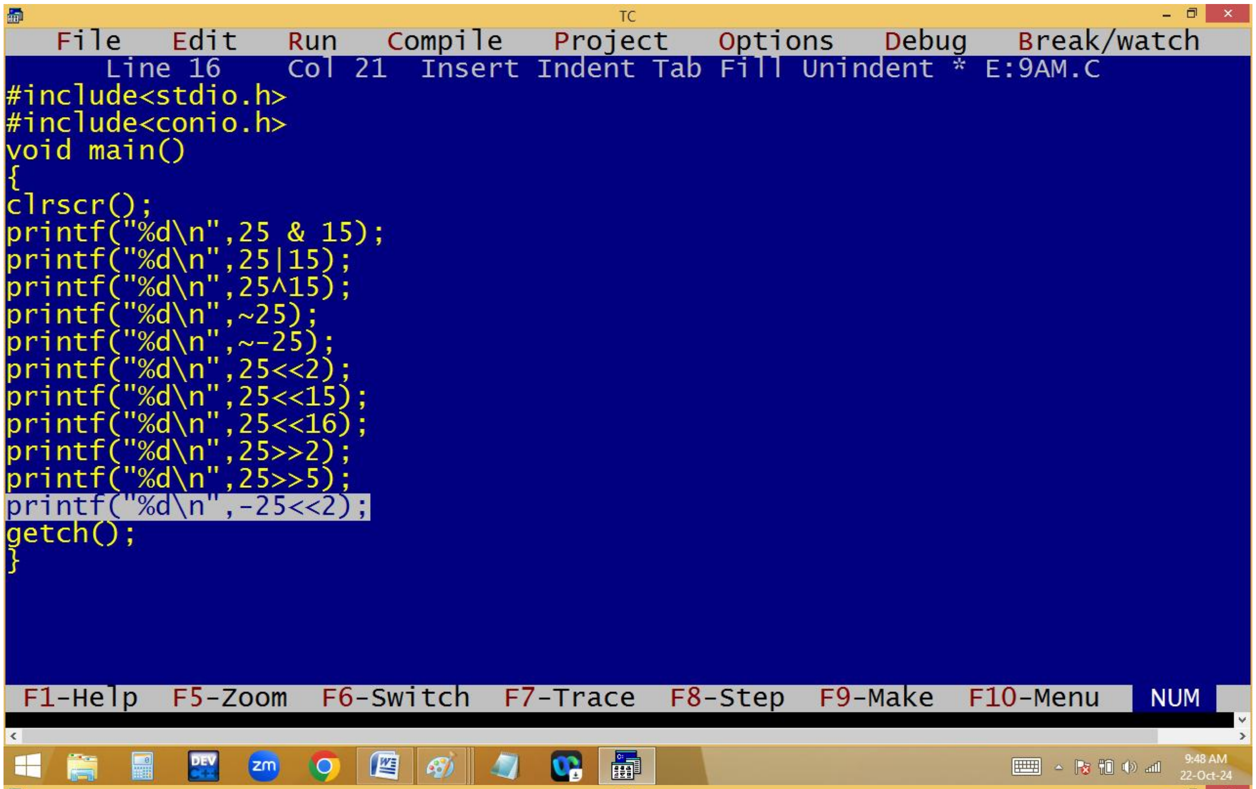
Function declarations and definitions contain function header and body.

* Every C-Program execution starts from main() function and travel towards down. Hence it is also called **top-down** approach.

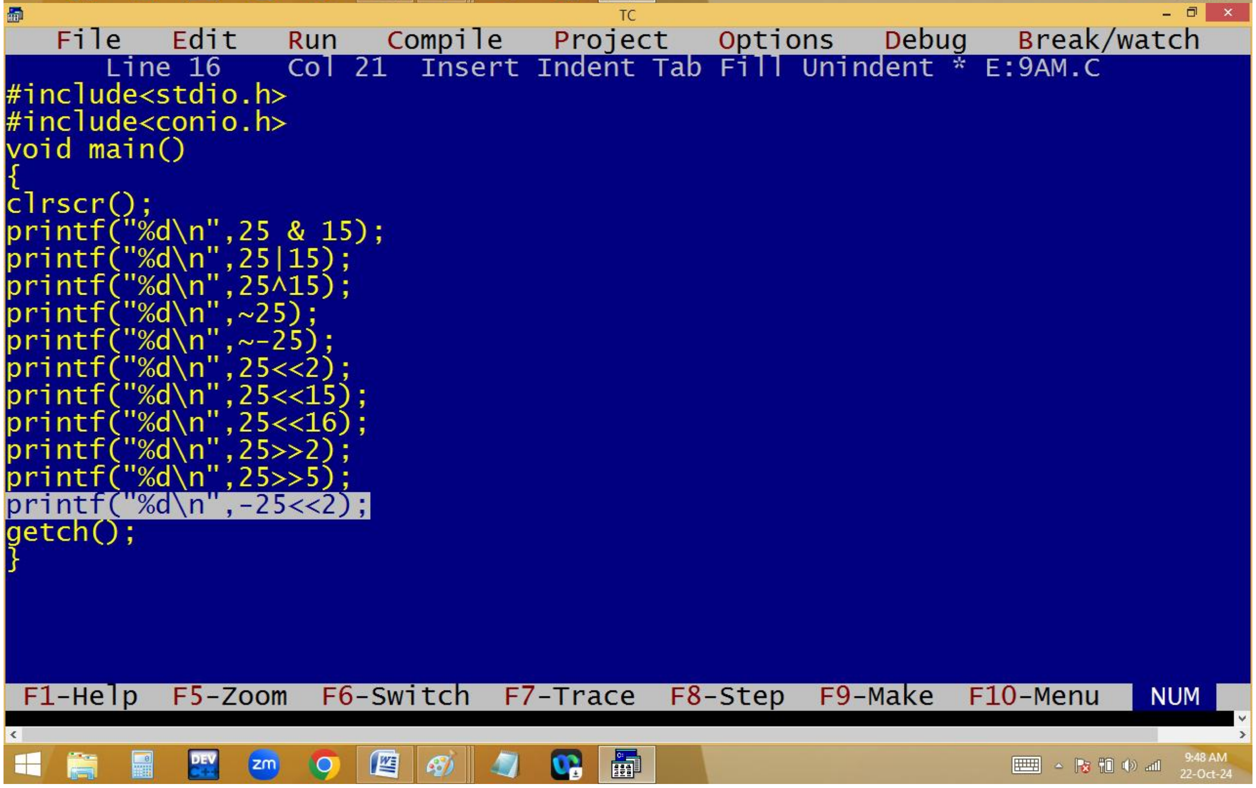
* Without main(), C-Program never executed but compiled.

* main() is predefined function with user defined body. main() doesn't have any header file. One program have to maintain one main() only. We can create alternate for main(). Other statements are changed from program to program.

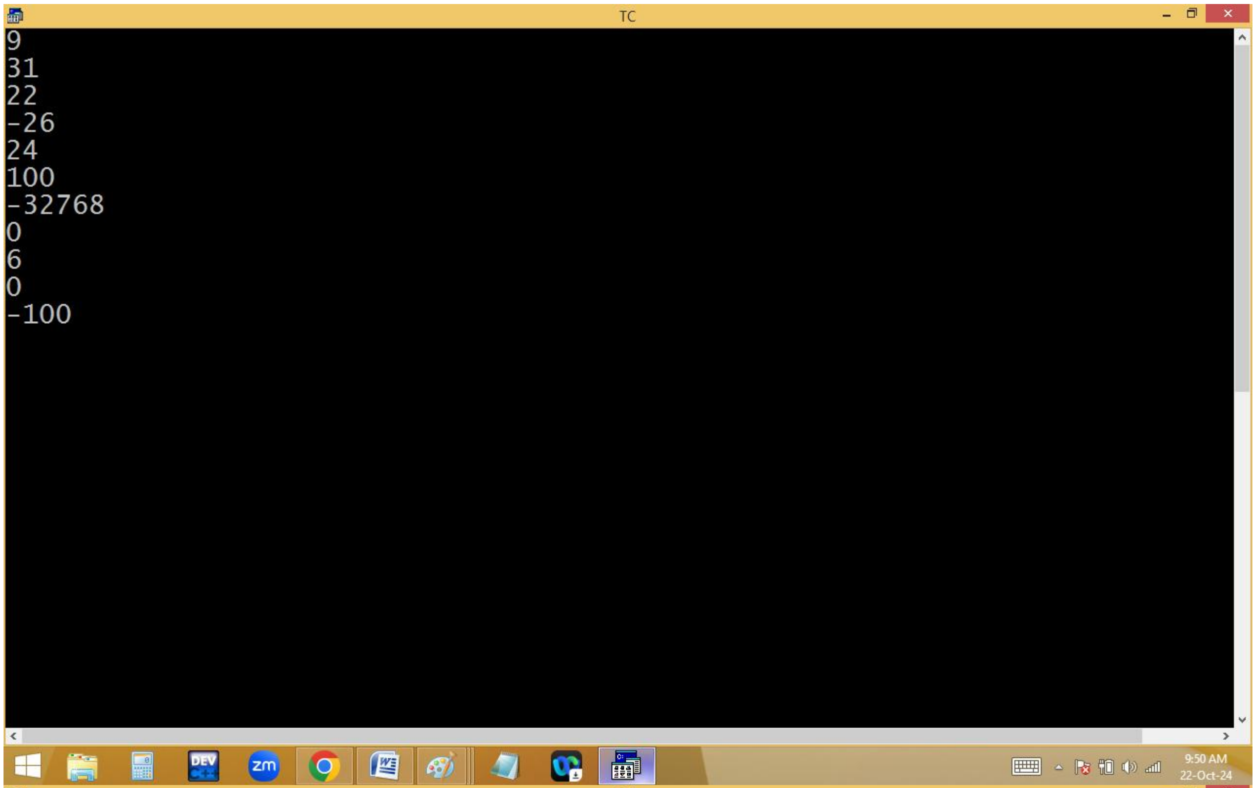
Note: It is recommended to write C programs in lower case only. Every statement should have to end with semicolon main().



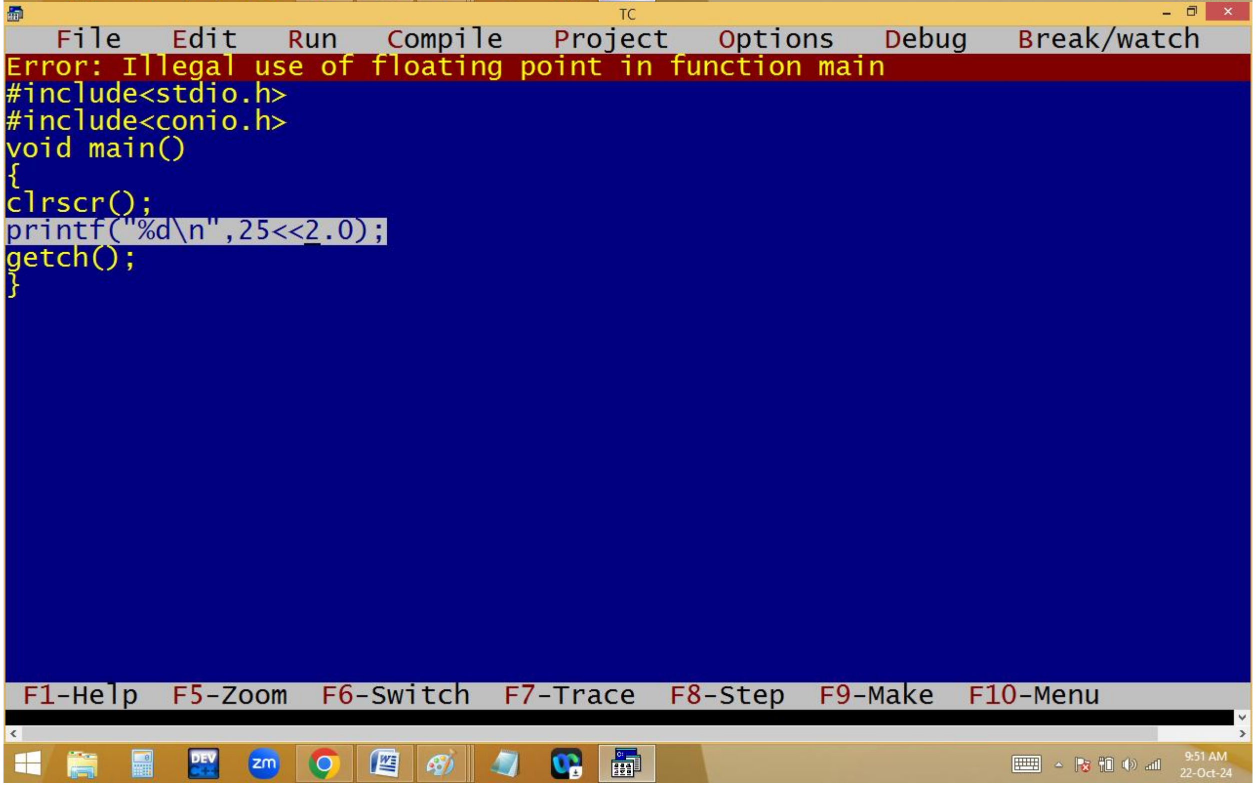
```
      Line 16    Col 21  Insert Indent Tab Fill Unindent * E:9AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("%d\n",25 & 15);
printf("%d\n",25|15);
printf("%d\n",25^15);
printf("%d\n",~25);
printf("%d\n",~-25);
printf("%d\n",25<<2);
printf("%d\n",25<<15);
printf("%d\n",25<<16);
printf("%d\n",25>>2);
printf("%d\n",25>>5);
printf("%d\n",-25<<2);
getch();
}
```


TC
File Edit Run Compile Project Options Debug Break/watch
F1-Help F5-Zoom F6-Switch F7-Trace F8-Step F9-Make F10-Menu NUM
9:48 AM
22-Oct-24

```
      Line 16    Col 21  Insert Indent Tab Fill Unindent * E:9AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("%d\n",25 & 15);
printf("%d\n",25|15);
printf("%d\n",25^15);
printf("%d\n",~25);
printf("%d\n",~-25);
printf("%d\n",25<<2);
printf("%d\n",25<<15);
printf("%d\n",25<<16);
printf("%d\n",25>>2);
printf("%d\n",25>>5);
printf("%d\n",-25<<2);
getch();
}
```

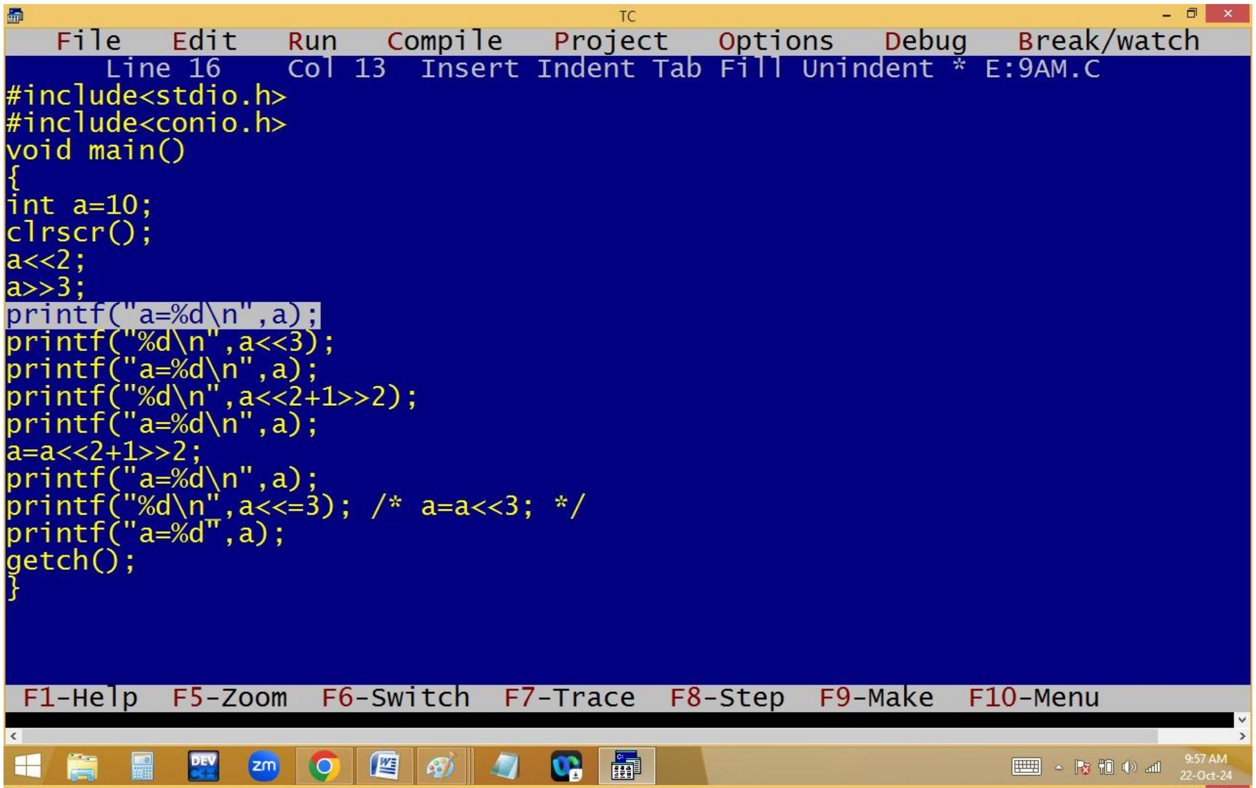


```
9
31
22
-26
24
100
-32768
0
6
0
-100
```

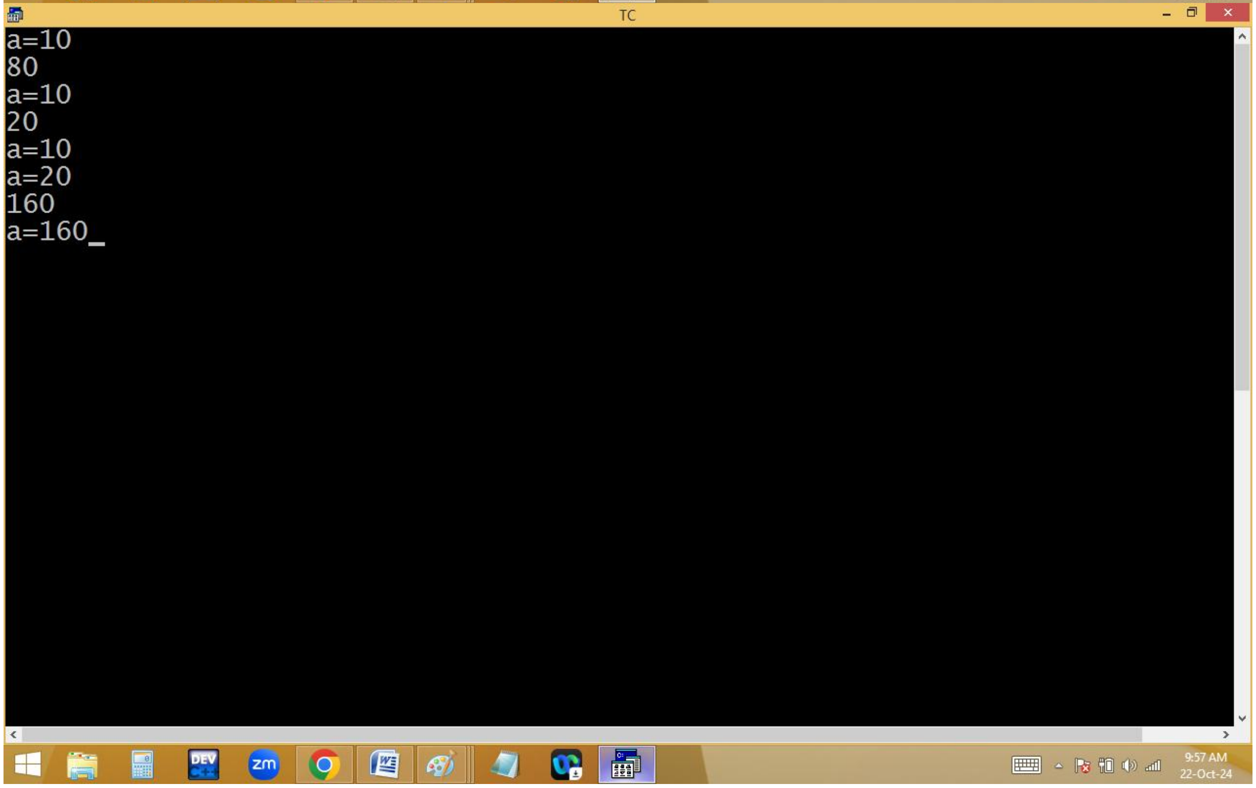

TC
File Edit Run Compile Project Options Debug Break/watch
F1-Help F5-Zoom F6-Switch F7-Trace F8-Step F9-Make F10-Menu
9:51 AM
22-Oct-24

```
Error: Illegal use of floating point in function main
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("%d\n",25<<2.0);
getch();
}
```


TC
File Edit Run Compile Project Options Debug Break/watch
Line 16 Col 13 Insert Indent Tab Fill Unindent * E:9AM.C
F1-Help F5-Zoom F6-Switch F7-Trace F8-Step F9-Make F10-Menu
9:57 AM
22-Oct-24

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=10;
clrscr();
a<<2;
a>>3;
printf("a=%d\n",a);
printf("%d\n",a<<3);
printf("a=%d\n",a);
printf("%d\n",a<<2+1>>2);
printf("a=%d\n",a);
a=a<<2+1>>2;
printf("a=%d\n",a);
printf("%d\n",a<<=3); /* a=a<<3; */
printf("a=%d",a);
getch();
}
```



```
a=10
80
a=10
20
a=10
a=20
160
a=160_
```

printf():

It is the major output function in c.

It is a predefined function available in stdio.h

It always refers standard output device. i.e. monitor.

In printf, f means formatted.

Syntax:

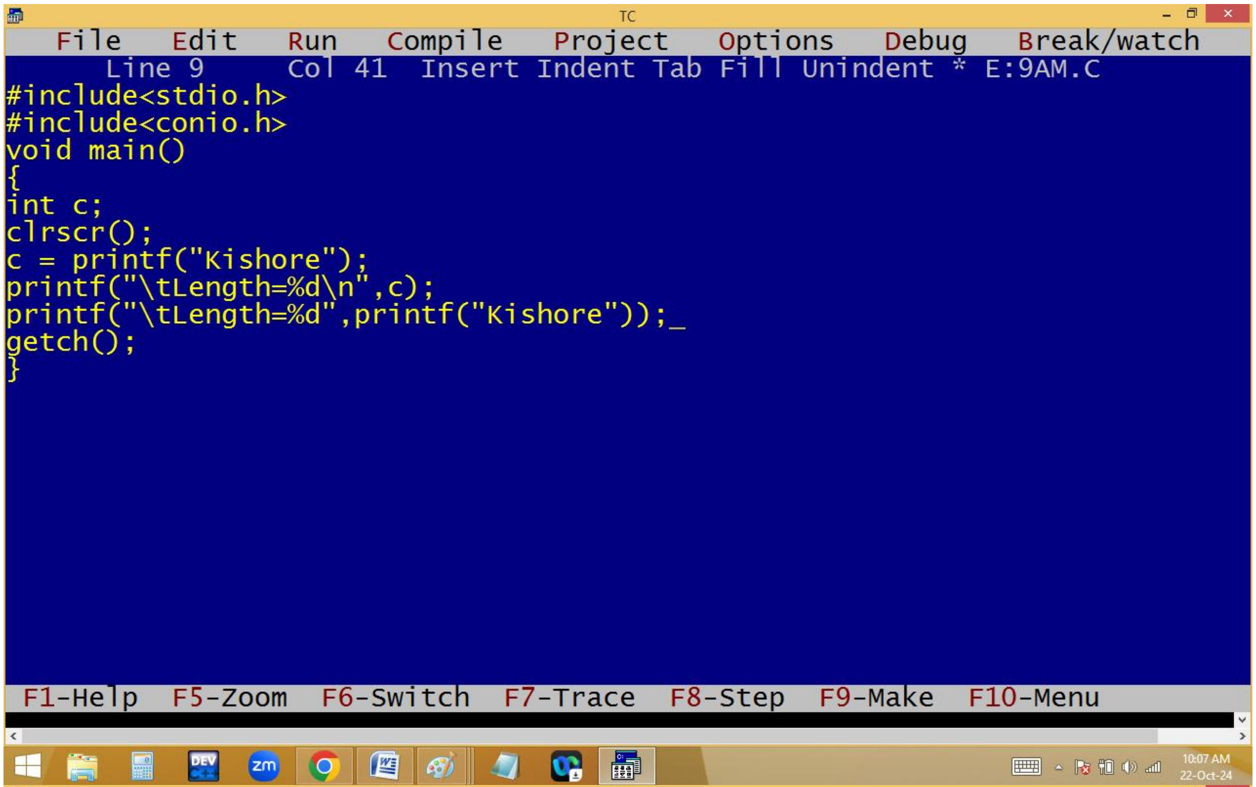
int printf(" [text] [conversion characters / format specifiers / format strings] " [, variables] [, expressions]);

Note:

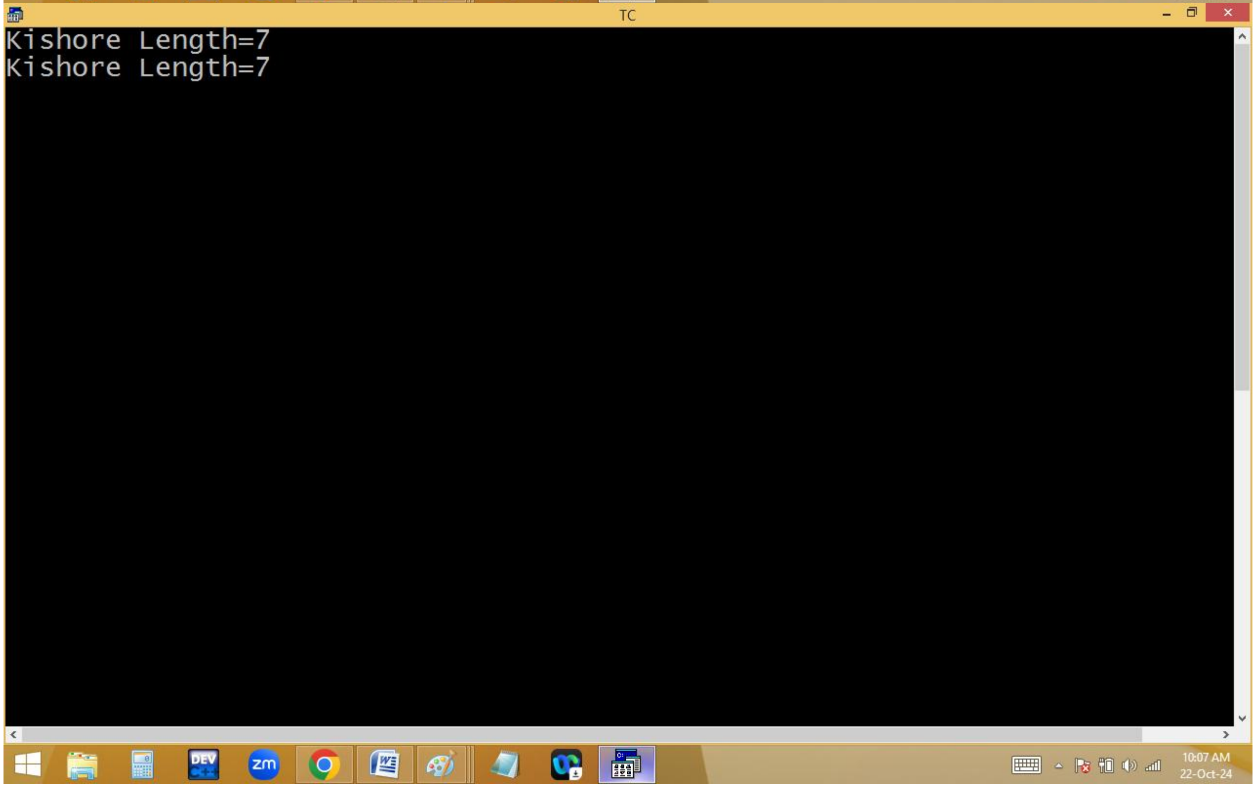
1. Printf always return int that indicates the no of printable characters on the screen.
2. In printf the first argument should be in " ".

3. **Printf can perform both formatted and unformatted outputs.**
4. **In printf everything printed as it is except conversion characters and back slash characters.**
5. **In printf execution order is right to left and printing is left to right.**

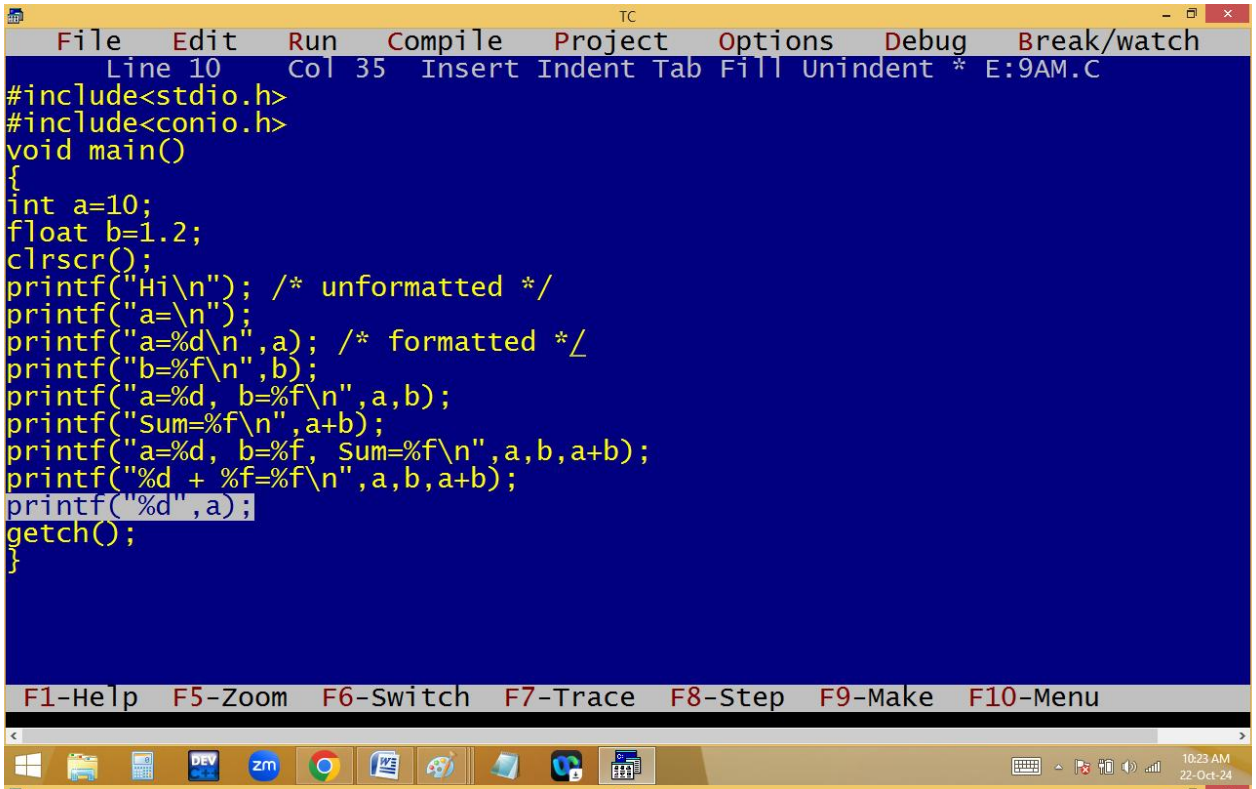
Eg. Write a c program to find string length without using loop / strlen()



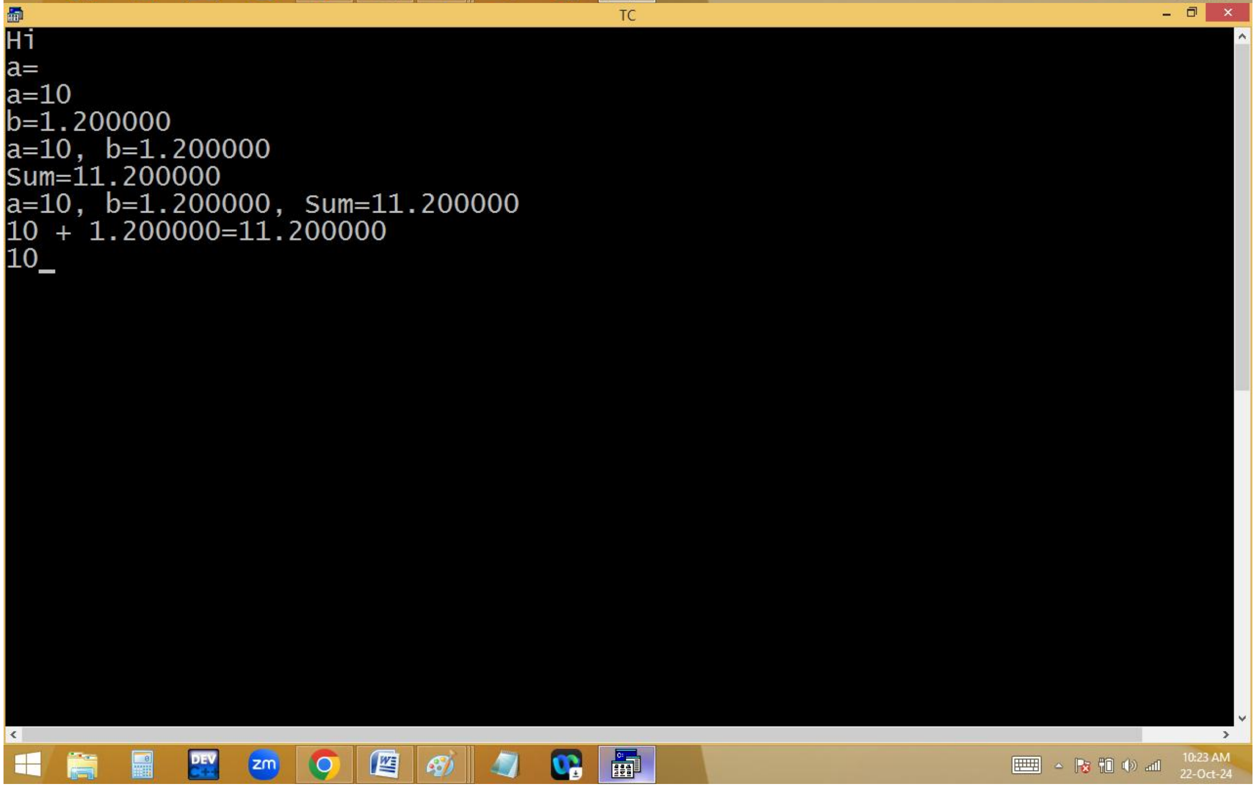
```
     Line 9     Col 41  Insert Indent Tab Fill Unindent * E:9AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
int c;
clrscr();
c = printf("Kishore");
printf("\tLength=%d\n",c);
printf("\tLength=%d",printf("Kishore"));
getch();
}
```



```
Kishore Length=7
Kishore Length=7
```


TC
File Edit Run Compile Project Options Debug Break/watch
Line 10 Col 35 Insert Indent Tab Fill Unindent * E:9AM.C
F1-Help F5-Zoom F6-Switch F7-Trace F8-Step F9-Make F10-Menu
10:23 AM
22-Oct-24

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=10;
float b=1.2;
clrscr();
printf("Hi\n"); /* unformatted */
printf("a=\n");
printf("a=%d\n",a); /* formatted */
printf("b=%f\n",b);
printf("a=%d, b=%f\n",a,b);
printf("Sum=%f\n",a+b);
printf("a=%d, b=%f, Sum=%f\n",a,b,a+b);
printf("%d + %f=%f\n",a,b,a+b);
printf("%d",a);
getch();
}
```



```
Hi
a=
a=10
b=1.200000
a=10, b=1.200000
Sum=11.200000
a=10, b=1.200000, Sum=11.200000
10 + 1.200000=11.200000
10_
```