# USER DEFINED FUNCTIONS

## What is a function?

1. It is a small program used to do a particular task.
2. It is a small program with in another program.
3. It is a sub program or sub routine.
4. It is a procedure.
5. It is a self-contained block.
6. It is a reusable code component.
7. It is a module.

## Advantages:

1. **Modularity**: Dividing program instructions into small pieces.
2. **Simplicity**: It is easy to understand the program instructions.
3. **Reusability**: Just write once, use several times.
4. **Efficiency**: performance is improved.
5. It is easy to identify the errors.

Entire ‘C’ program is collection of functions. Hence ‘C’ is a function oriented structured programming language.

We are having 2 types of functions.

1. Predefined / library / built-in functions

These are the functions provided with software and ready to use.

Eg: printf(), scanf(),…

2. **User defined functions**: They are created by the user.

Eg: sum(), prime(), factorial(),……

Every user defined function is divided into 3 parts.

1. Function declaration/ proto typing
2. Function calling
3. Function definition

## Function declaration/proto typing:

Generally function declaration is conducted before or within the main(), before function calling.

It tells the compiler that we are going to use this kind of function in our program in future.

**Syntax:**

**[return_datatype] function_name ( [arguments / parameters] );**

**Eg:**

**void sum();**

## Function calling:

It should be conducted within the main() only.

When a function is invoked (called), the compiler will search for the matching function definition and it is available then the program execution is jumped from function calling area to the function definition area.

Linking a function call to the function definition is called **binding.** C language supports only static or compile time binding. But C++ supports static and dynamic binding[run time binding].

Without function call, a function never participates in function execution. Hence it is mandatory to call a function in a program.

**Syntax:**

function_name([arguments]);

Eg: sum();

## Function definition:

It contains the function header and body. The function header should be matched with function declaration.

It is conducted outside the main(). If the definition is conducted before the main(), there is no need of function declaration.

Function header consists of function name and the parameters.

Function body consists of statements related to the task of function.

When a function is invoked, the program execution is shifted from function calling area ( main() ) to function definition area. After executing the function body, the execution is jumped to the next statement after the function call in main().

Function declaration and function definition should be identical.

## Syntax:

**Function Header**

```
[return data type]  Function_name(
[parameters/arguments] )
{
  Statements;
```

**Function Body**

```
 [return value;]
}
```

Here return data type indicates the type of value that the function is returning to main() / called function. It is depended on the return value.

**If the sub program is having return statement then it is called function and without return value it is called procedure.**

The **default return value of a function is integer**. Use the keyword **void** [nothing] when the function is not having any return value. Otherwise integer will become the return value.

A function may have any number of return statements. But only one return statement is executed at a time. The statements after the return statement are not executed and gives compile time warning.

Return data type should be matched with return value data type.

Function name is used to identify a function and it should not be predefined.

Arguments / parameters used to carry the values from function calling area to function definition area. They are optional and of two types.

1. **Actual** arguments/parameters.
2. **Formal** arguments/parameters.

**The arguments we are sending in function calling at main() are called actual parameters and the parameters we are using in function definition to receive the actual argument values is called formal parameters.**

Actual and formal parameter names may be same or different.

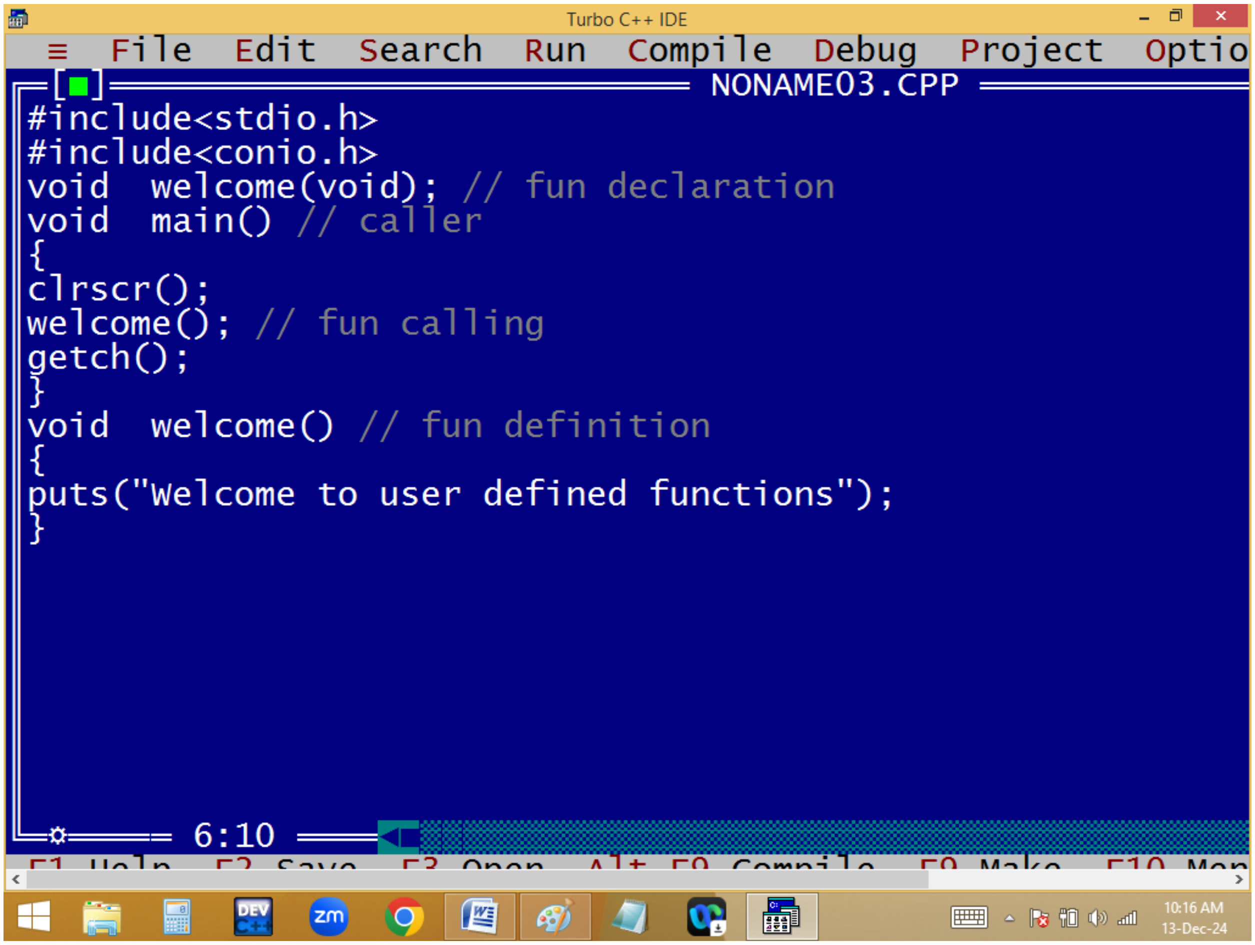
Difference between arguments and parameters is “arguments are nothing but different vehicles which are parked in same parking area and the parking area is the parameters. i.e. arguments are always changed but parameters are fixed.

The calling function arguments should be matched with definition parameters list in quantity, data type and order.

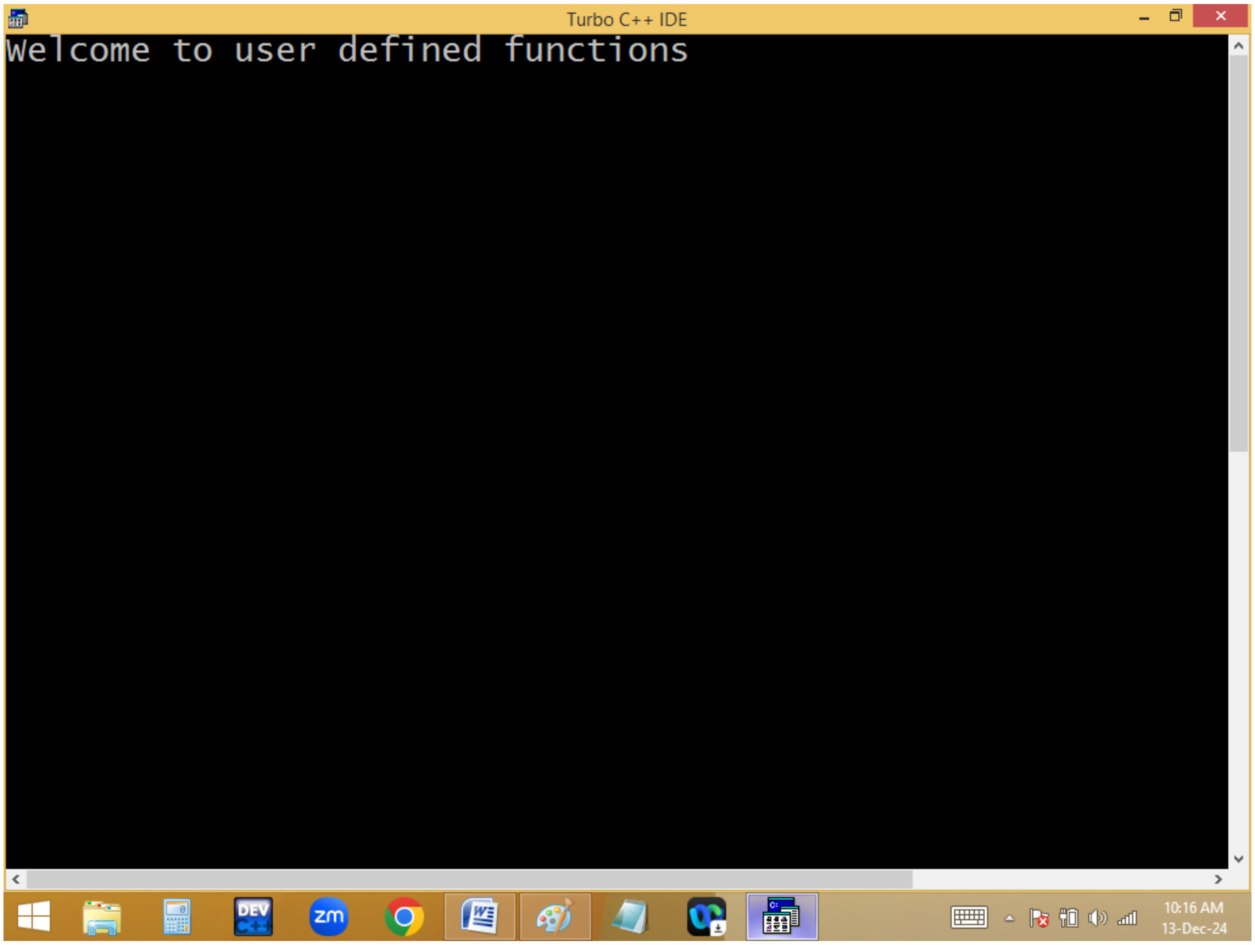
Based on arguments and return values, functions are divided into 4 types.

1. Function without arguments, without return values.
2. Function with arguments and with return values.
3. Function without arguments and with return values.
4. Function with arguments and without return values.

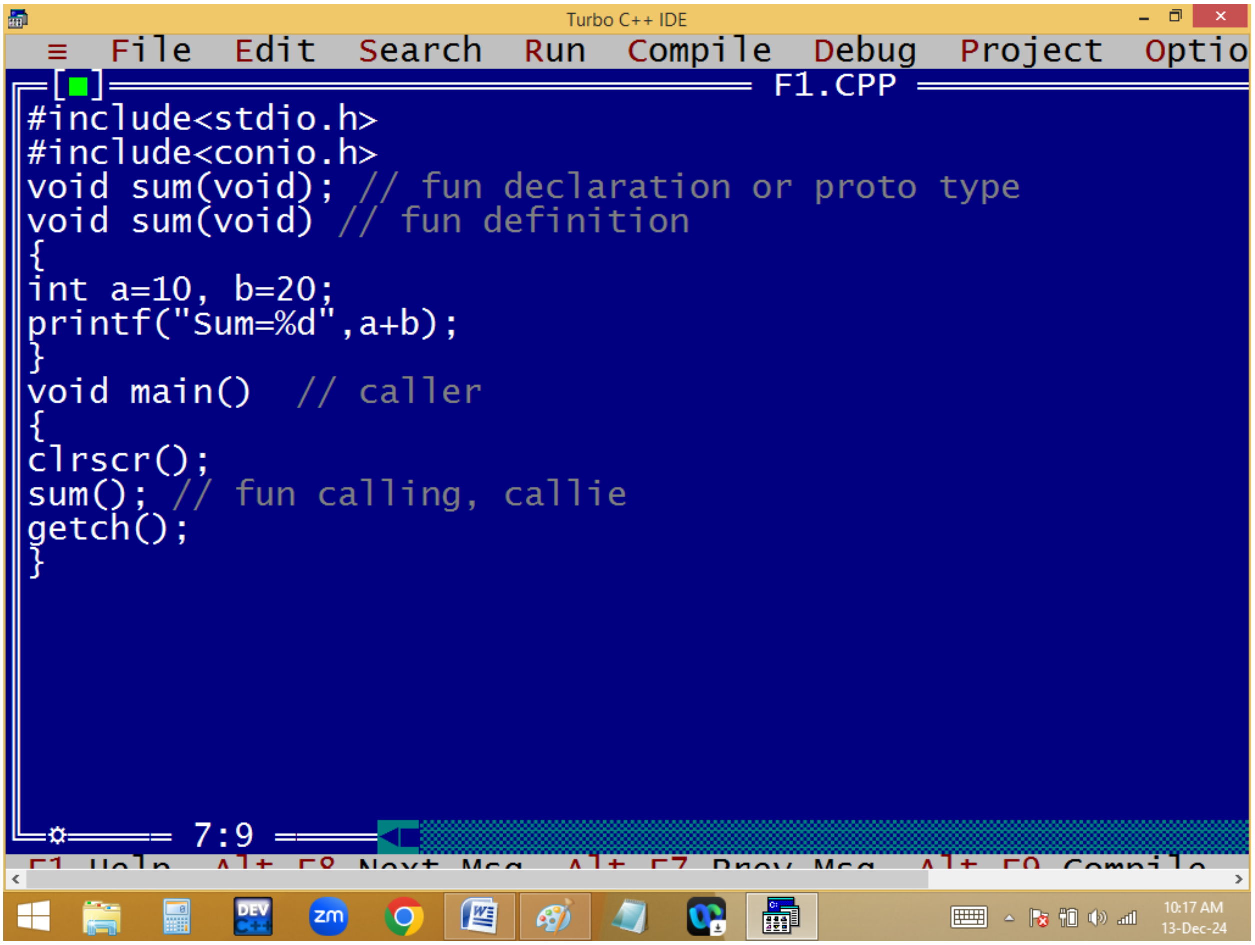
Eg: **function without arguments, without return values**


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
NONAME03.CPP
6:10
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Men
10:16 AM
13-Dec-24

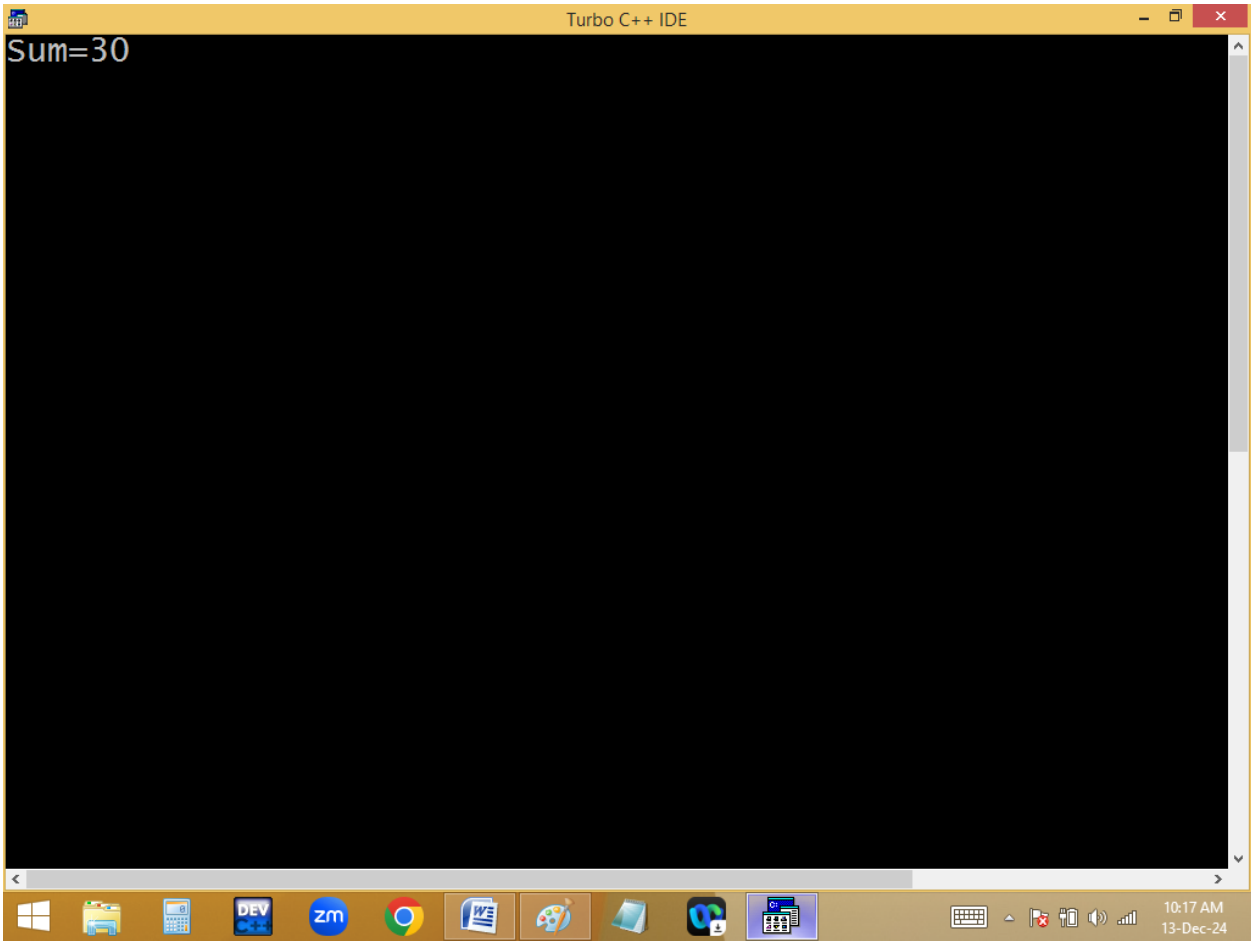
```
#include<stdio.h>
#include<conio.h>
void  welcome(void); // fun declaration
void  main() // caller
{
clrscr();
welcome(); // fun calling
getch();
}
void  welcome() // fun definition
{
puts("Welcome to user defined functions");
}
```



```
Welcome to user defined functions
```

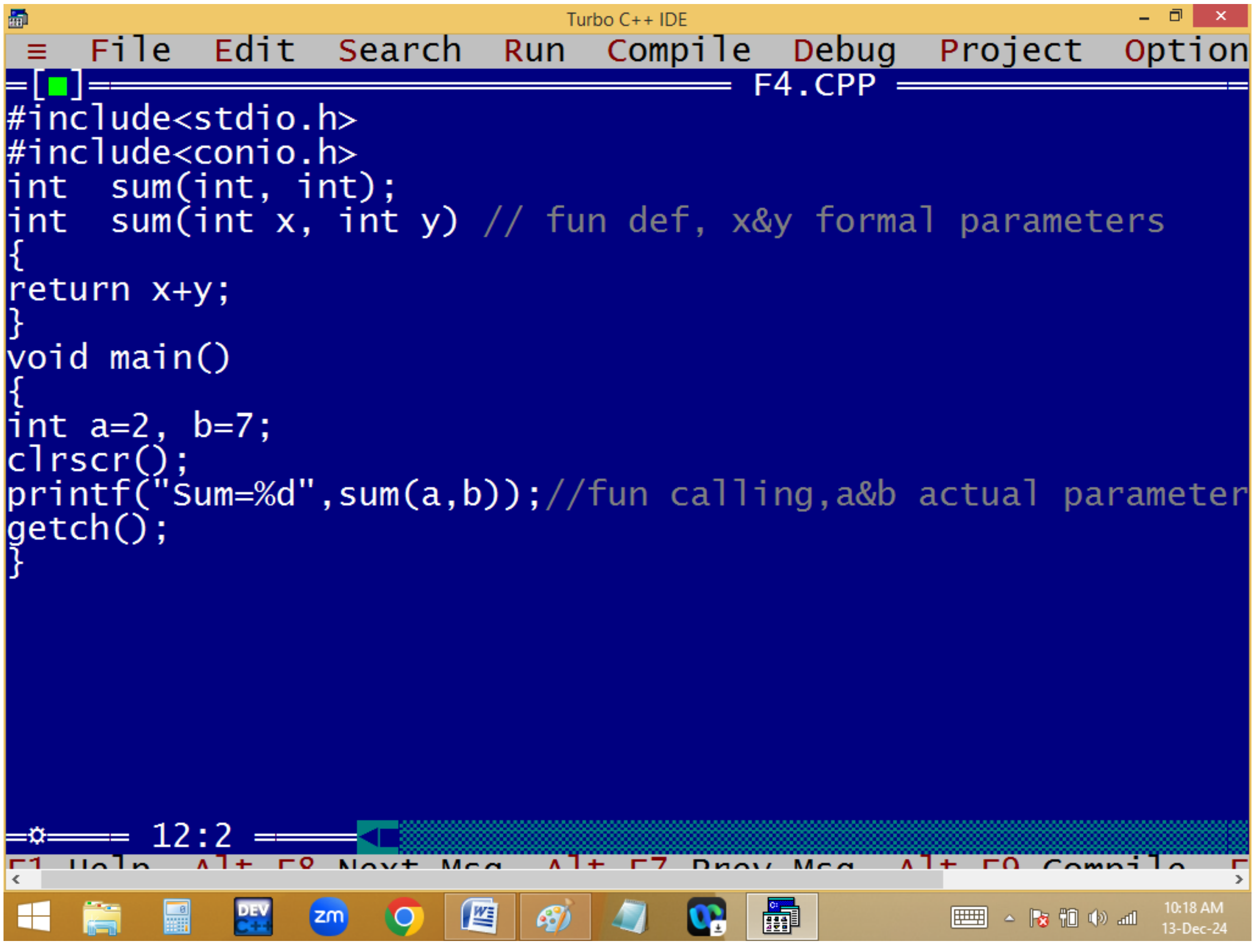

Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F1.CPP
7:9
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile
10:17 AM
13-Dec-24

```
#include<stdio.h>
#include<conio.h>
void sum(void); // fun declaration or proto type
void sum(void) // fun definition
{
int a=10, b=20;
printf("Sum=%d",a+b);
}
void main()  // caller
{
clrscr();
sum(); // fun calling, callie
getch();
}
```

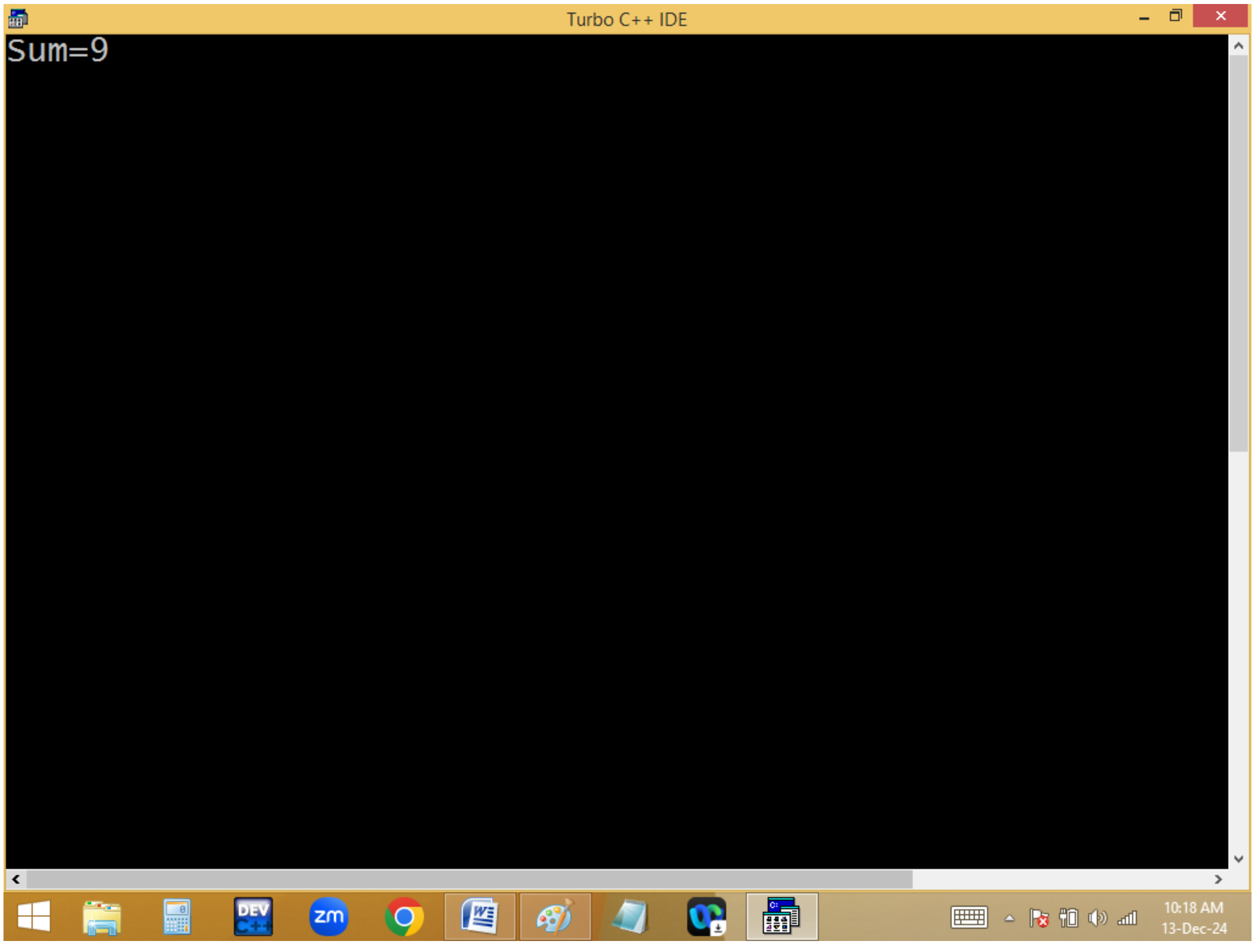


```
Sum=30
```

## Function with arguments, with return value:

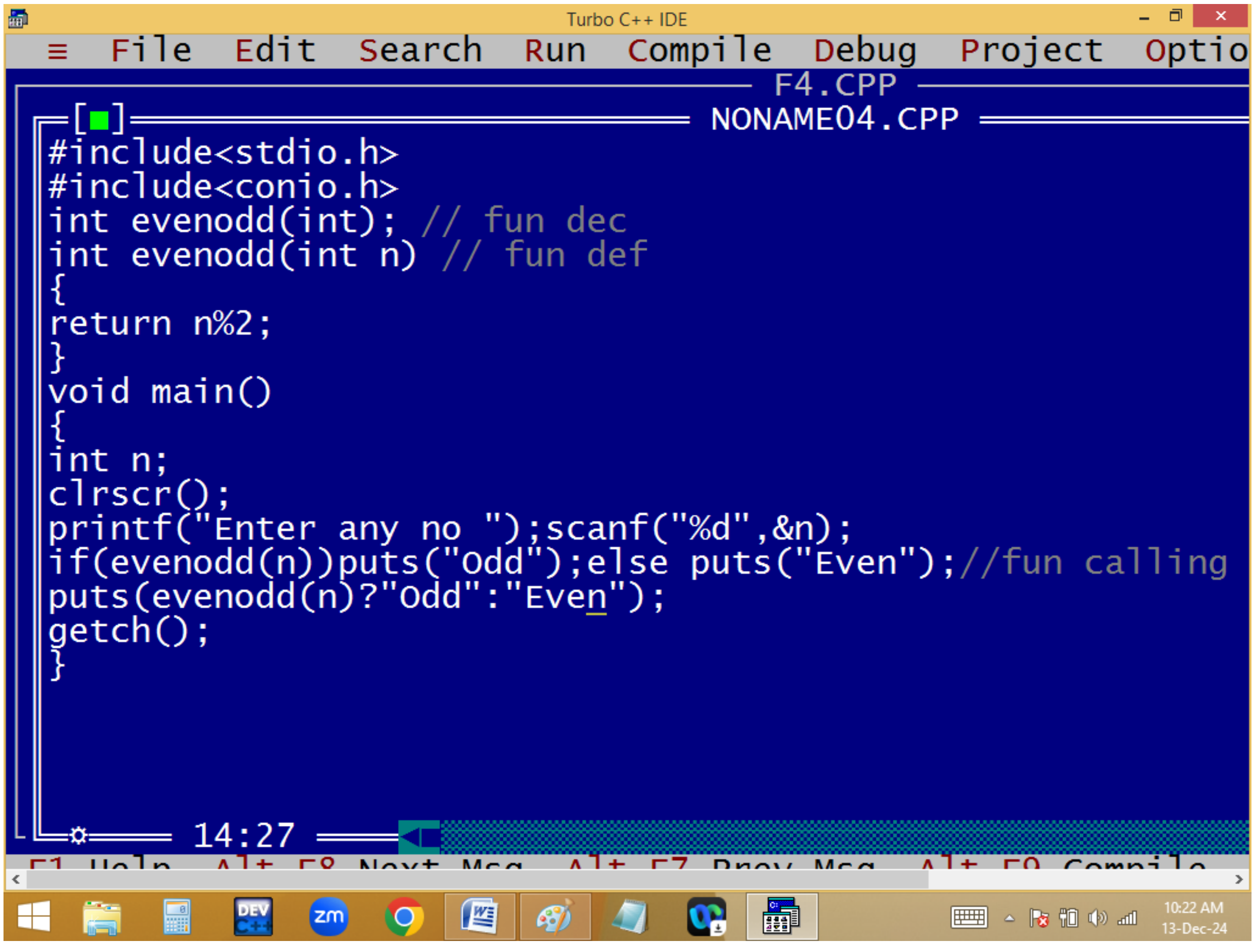


```
#include<stdio.h>
#include<conio.h>
int  sum(int, int);
int  sum(int x, int y) // fun def, x&y formal parameters
{
return x+y;
}
void main()
{
int a=2, b=7;
clrscr();
printf("Sum=%d",sum(a,b));//fun calling,a&b actual parameter
getch();
}
```

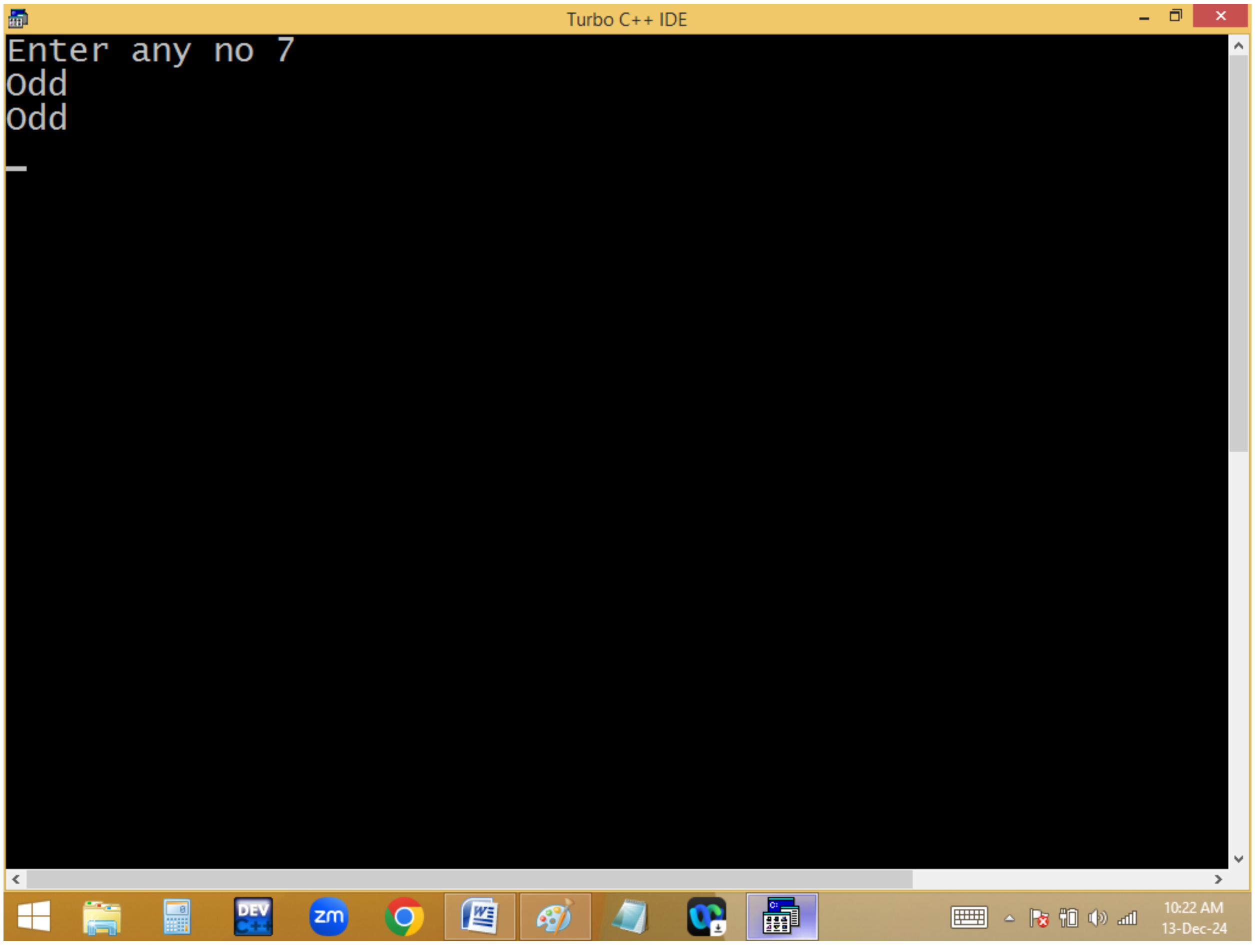


```
Sum=9
```

## Finding even/odd using user defined function?

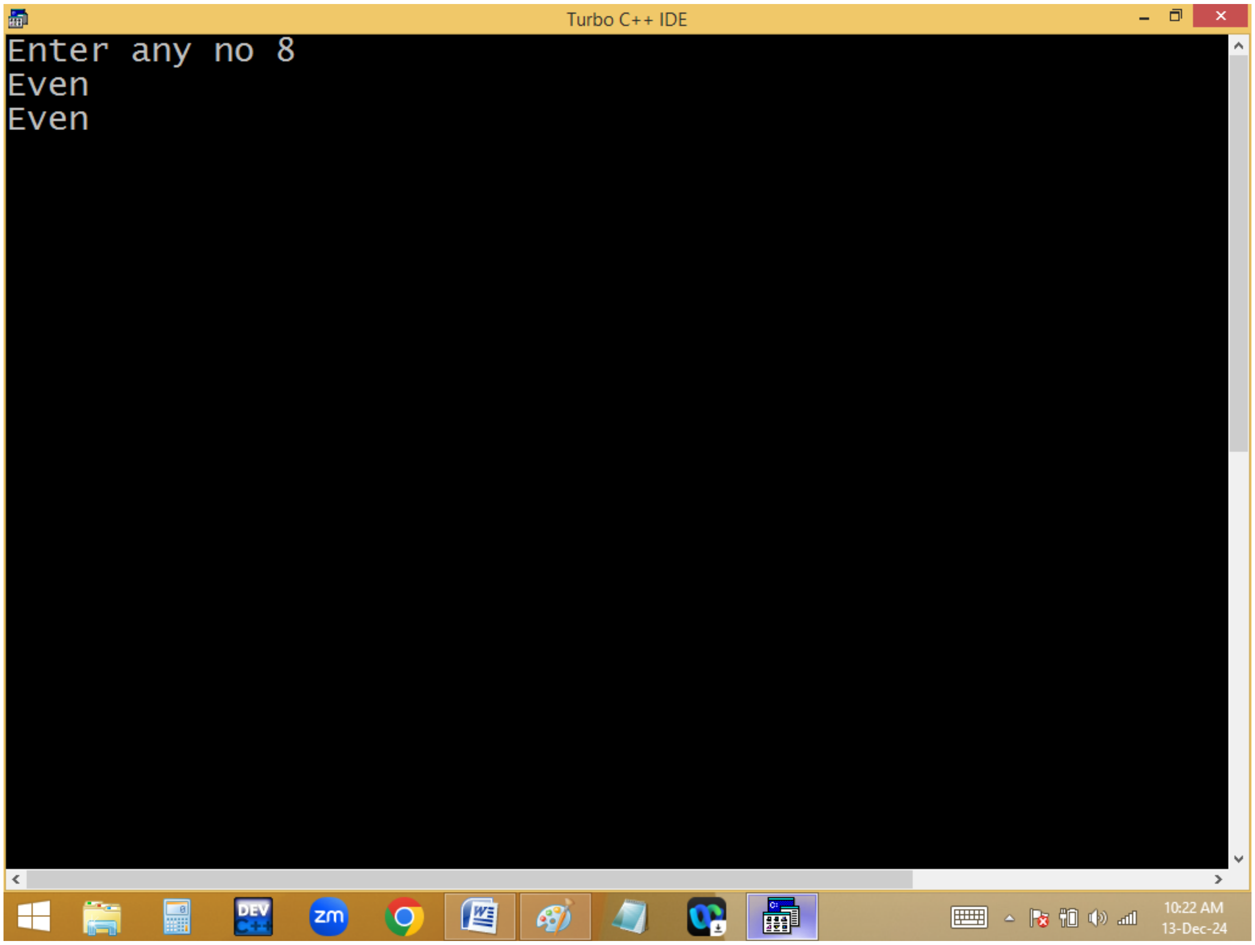

Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F4.CPP
[■]
NONAME04.CPP
14:27
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile
10:22 AM
13-Dec-24

```
#include<stdio.h>
#include<conio.h>
int evenodd(int); // fun dec
int evenodd(int n) // fun def
{
return n%2;
}
void main()
{
int n;
clrscr();
printf("Enter any no ");scanf("%d",&n);
if(evenodd(n))puts("Odd");else puts("Even");//fun calling
puts(evenodd(n)?"Odd":"Even");
getch();
}
```



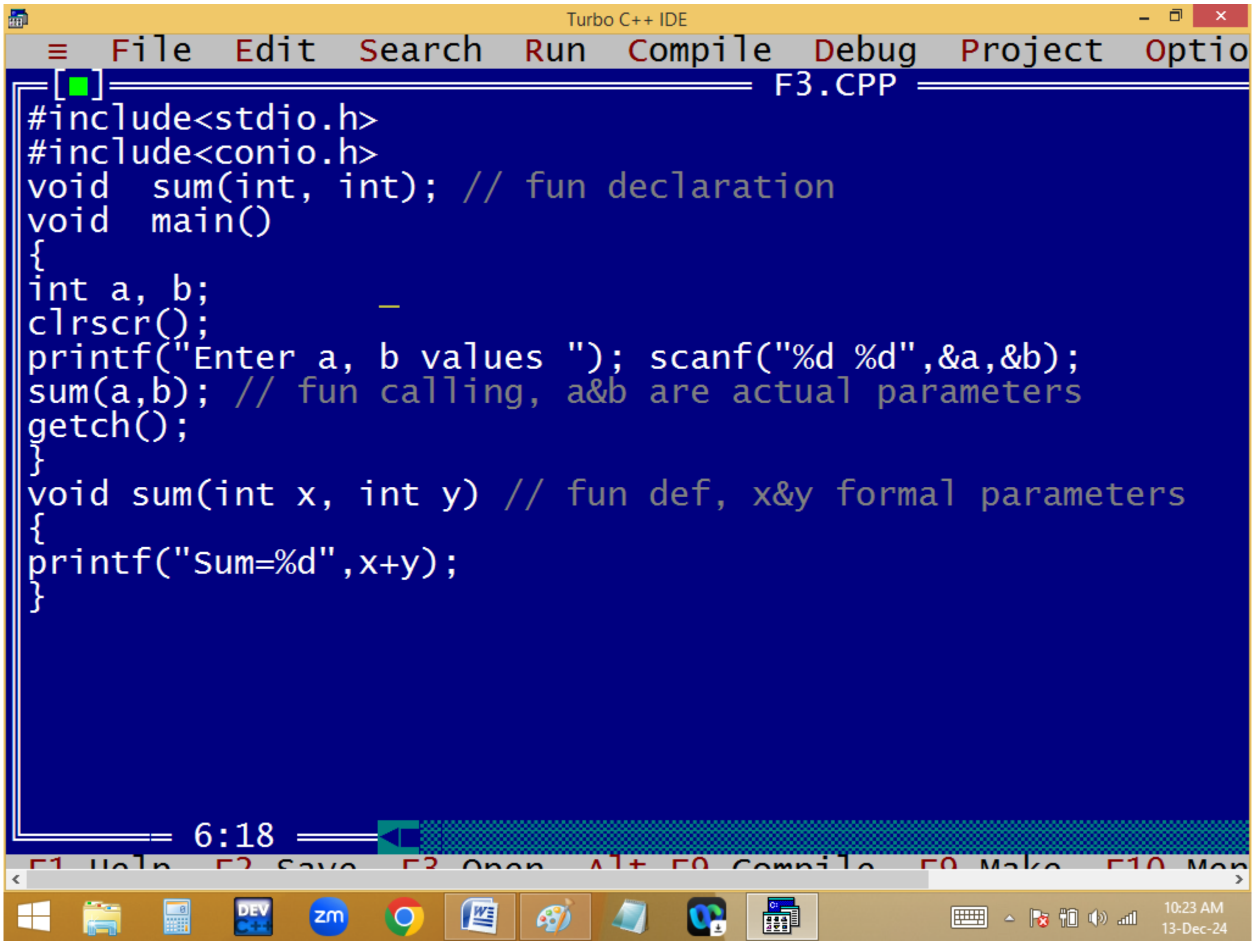
```
Enter any no 7
Odd
Odd

_
```

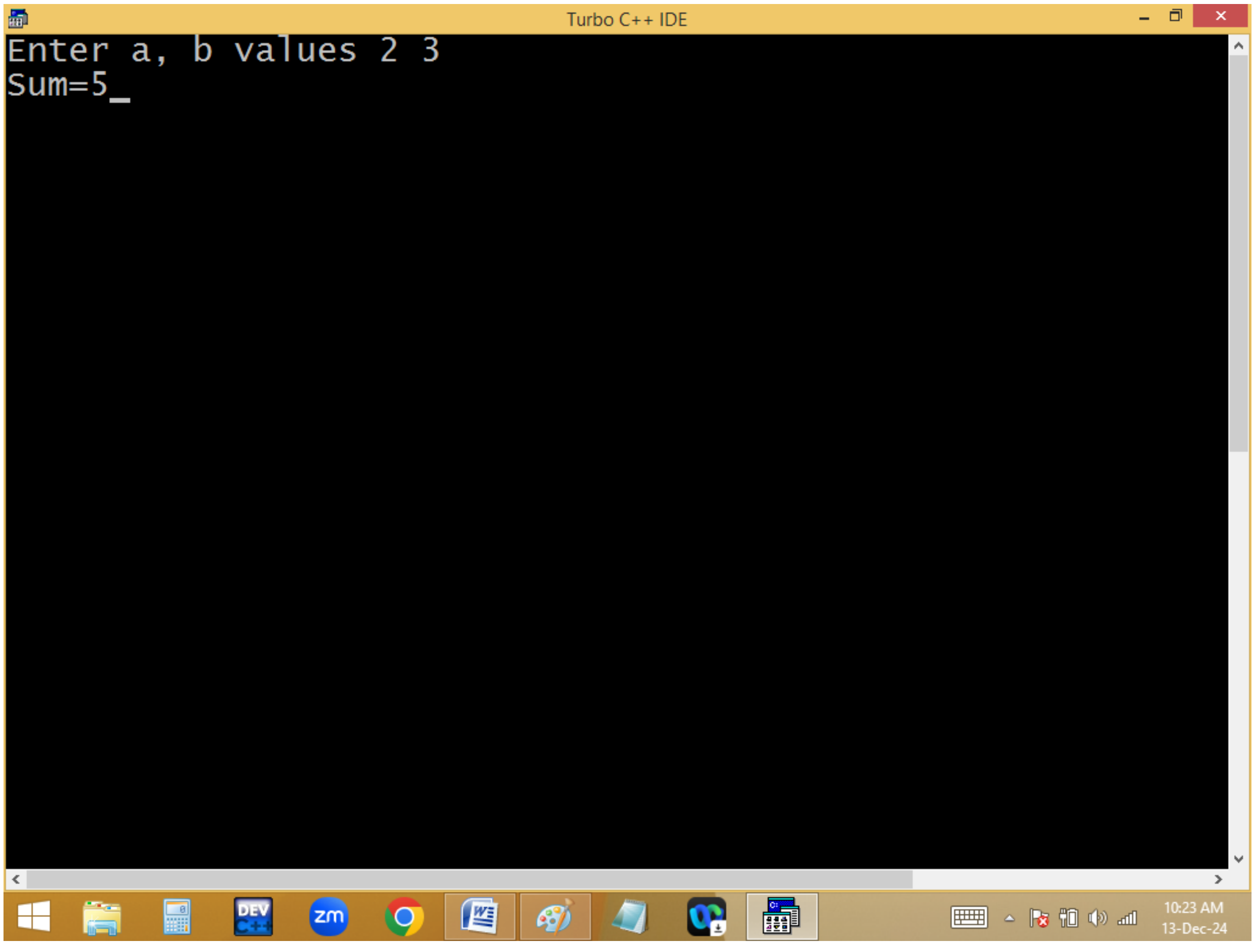


```
Enter any no 8
Even
Even
```

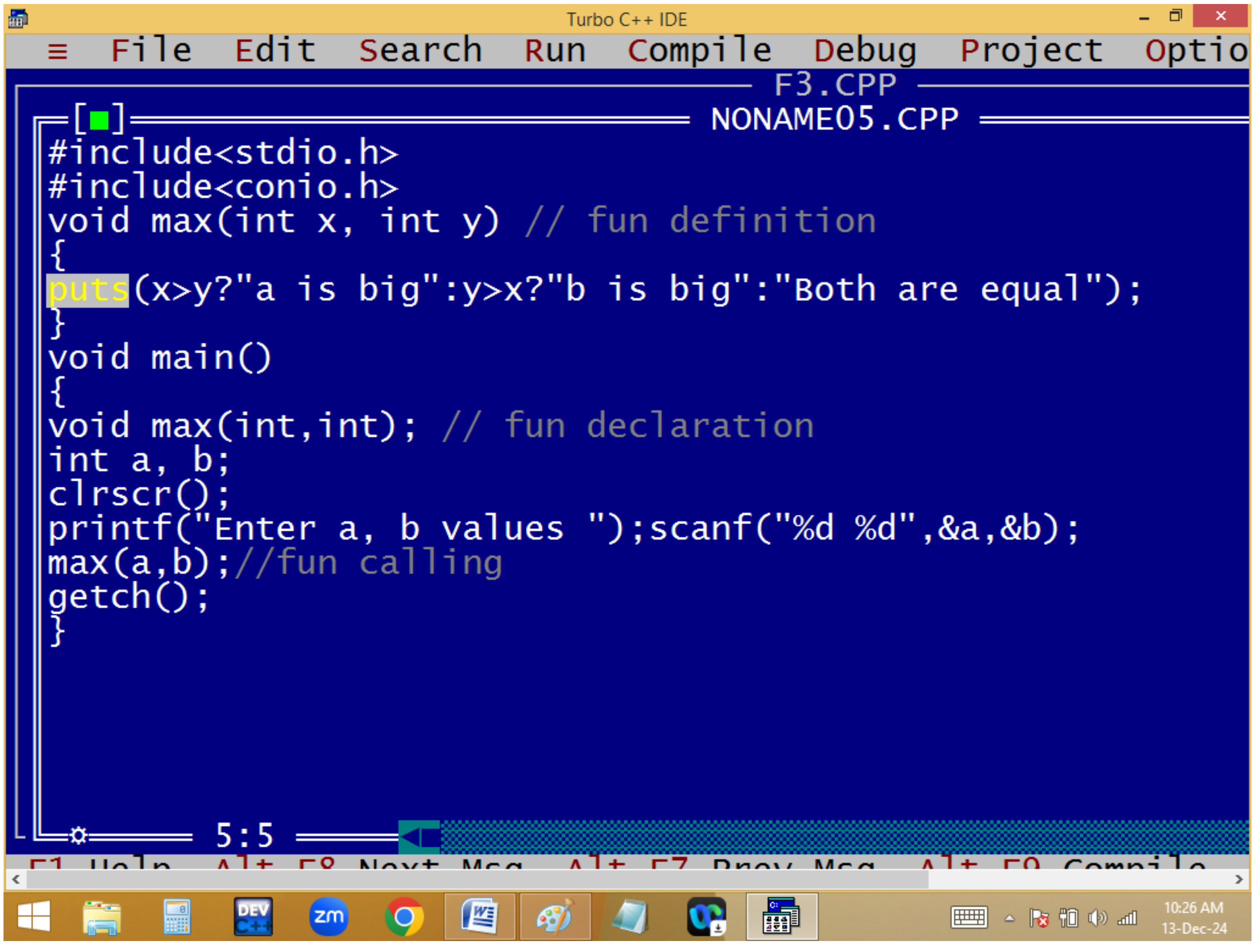
## Function with arguments, without return value:


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F3.CPP
6:18
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Men
10:23 AM
13-Dec-24

```
#include<stdio.h>
#include<conio.h>
void  sum(int, int); // fun declaration
void  main()
{
int a, b;
clrscr();
printf("Enter a, b values "); scanf("%d %d",&a,&b);
sum(a,b); // fun calling, a&b are actual parameters
getch();
}
void sum(int x, int y) // fun def, x&y formal parameters
{
printf("Sum=%d",x+y);
}
```

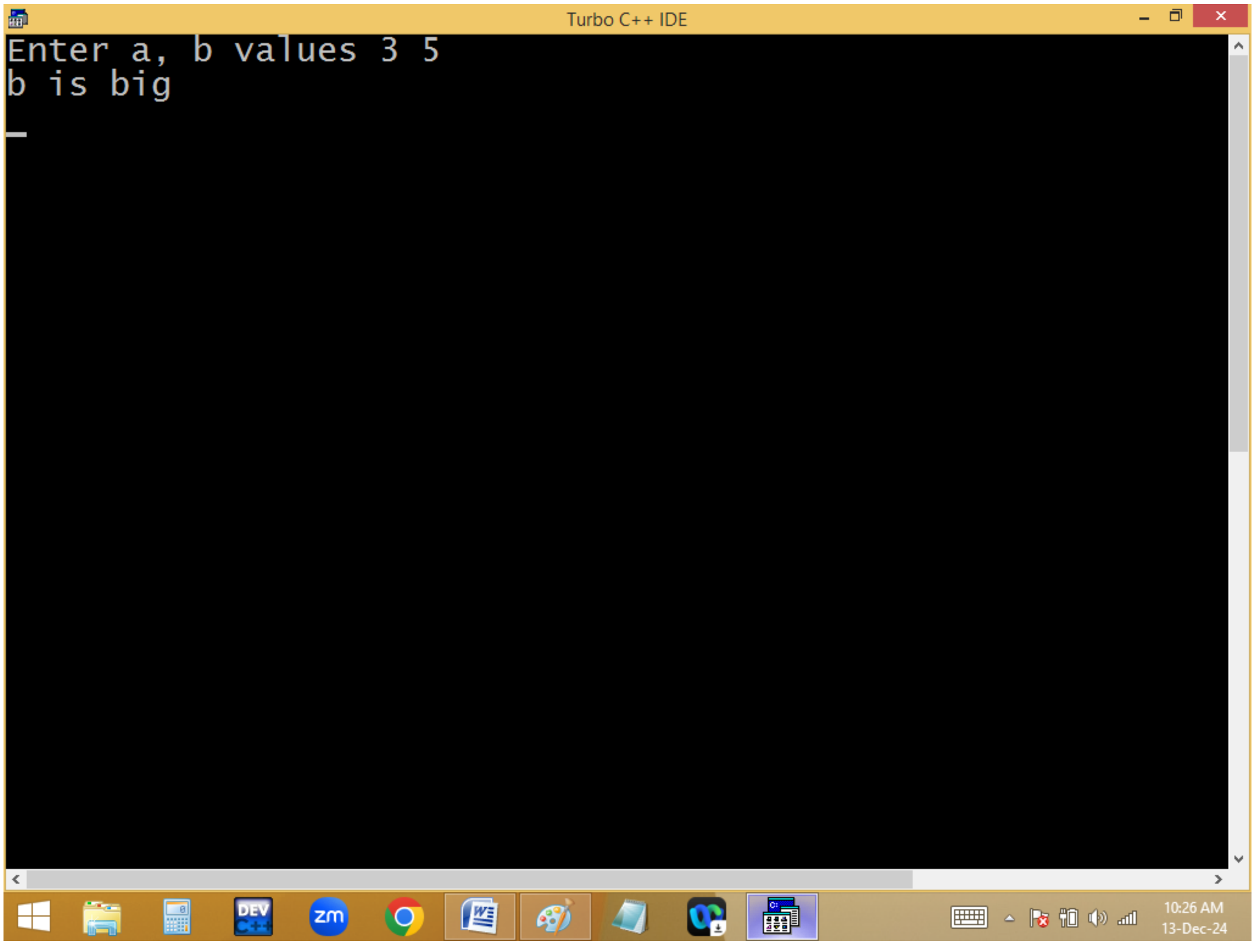


```
Enter a, b values 2 3
Sum=5_
```

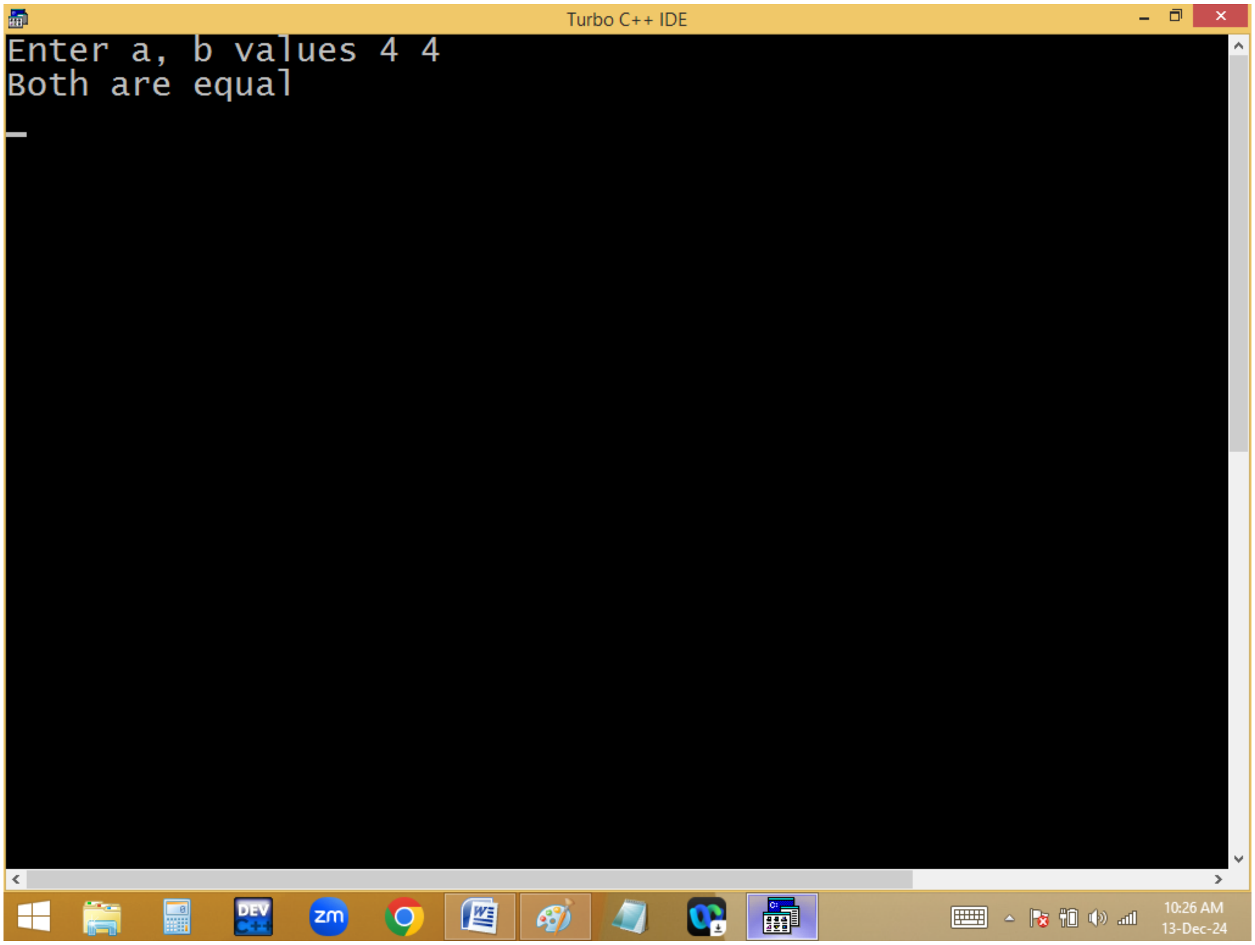


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F3.CPP
NONAME05.CPP
5:5
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile
10:26 AM
13-Dec-24

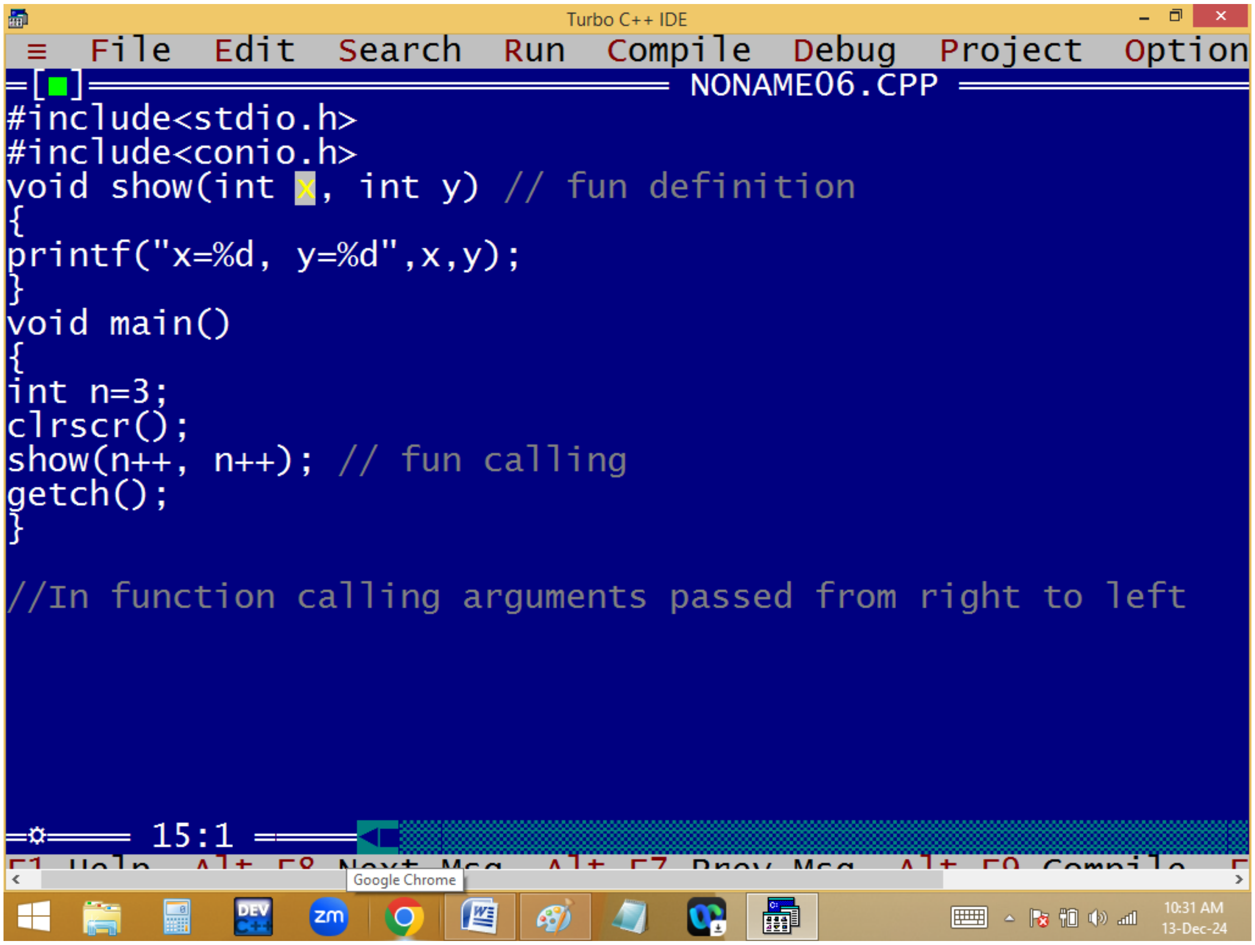
```
#include<stdio.h>
#include<conio.h>
void max(int x, int y) // fun definition
{
puts(x>y?"a is big":y>x?"b is big":"Both are equal");
}
void main()
{
void max(int,int); // fun declaration
int a, b;
clrscr();
printf("Enter a, b values ");scanf("%d %d",&a,&b);
max(a,b);//fun calling
getch();
}
```



```
Enter a, b values 3 5
b is big
```

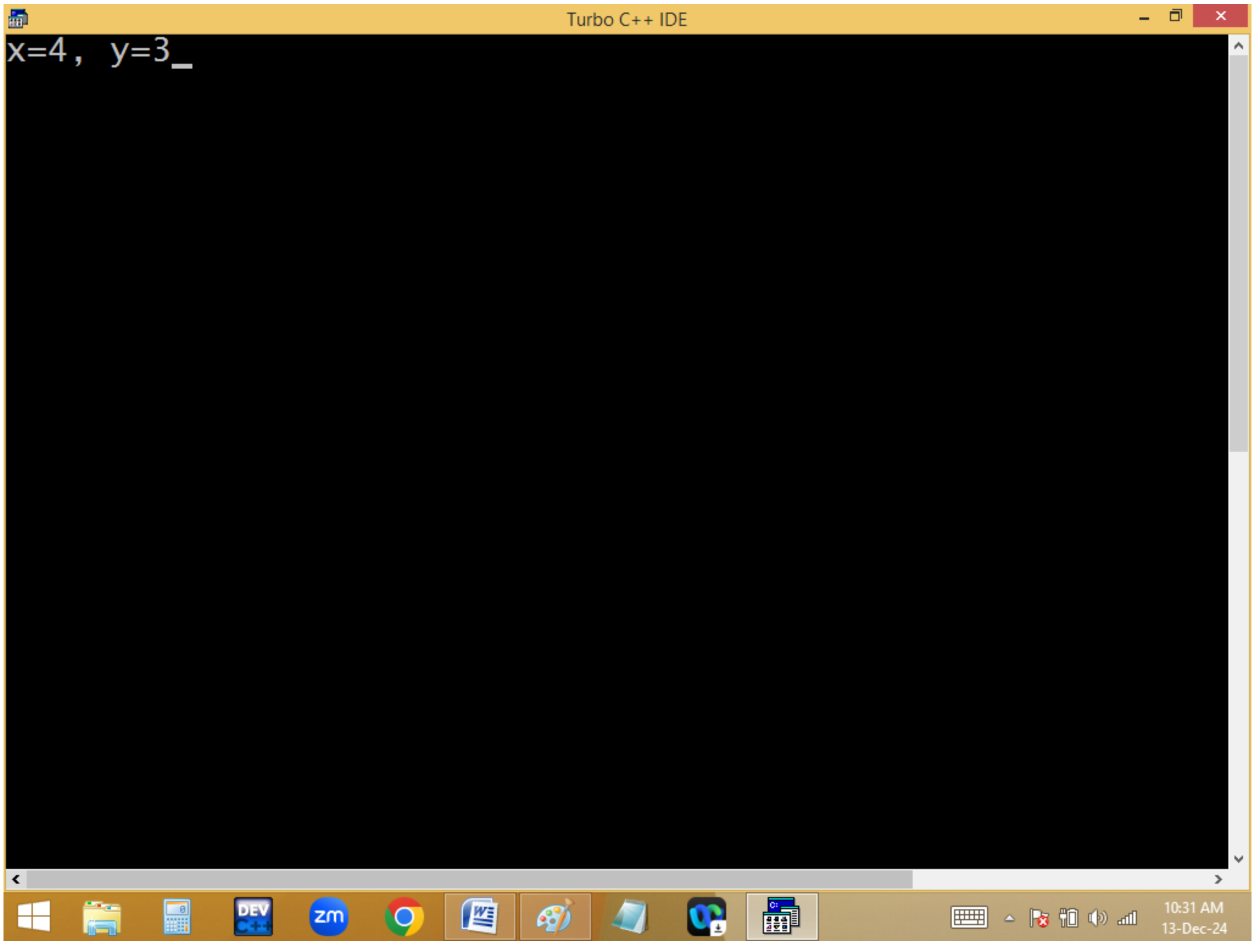


```
Enter a, b values 4 4
Both are equal
_
```



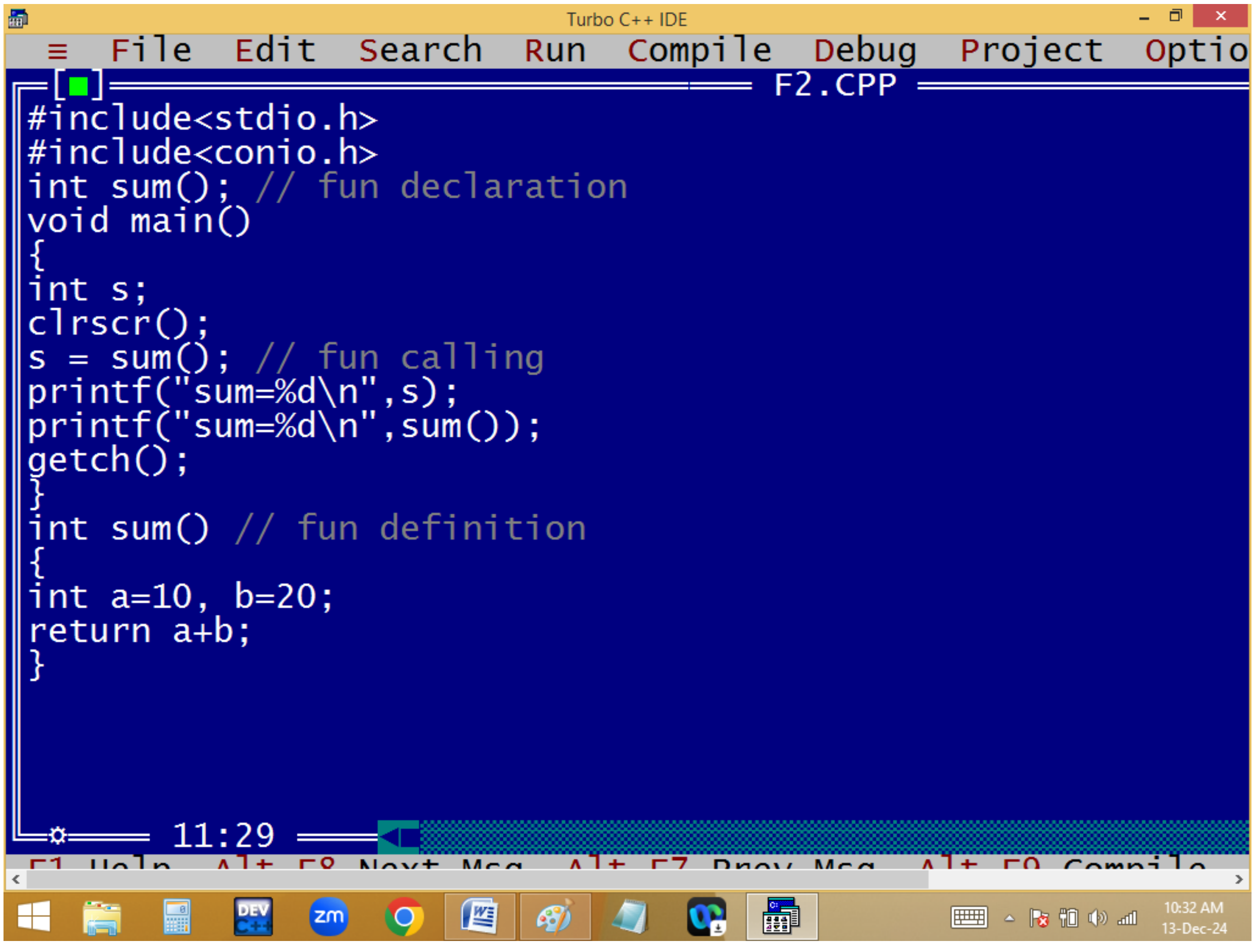
```
#include<stdio.h>
#include<conio.h>
void show(int x, int y) // fun definition
{
printf("x=%d, y=%d",x,y);
}
void main()
{
int n=3;
clrscr();
show(n++, n++); // fun calling
getch();
}

//In function calling arguments passed from right to left
```

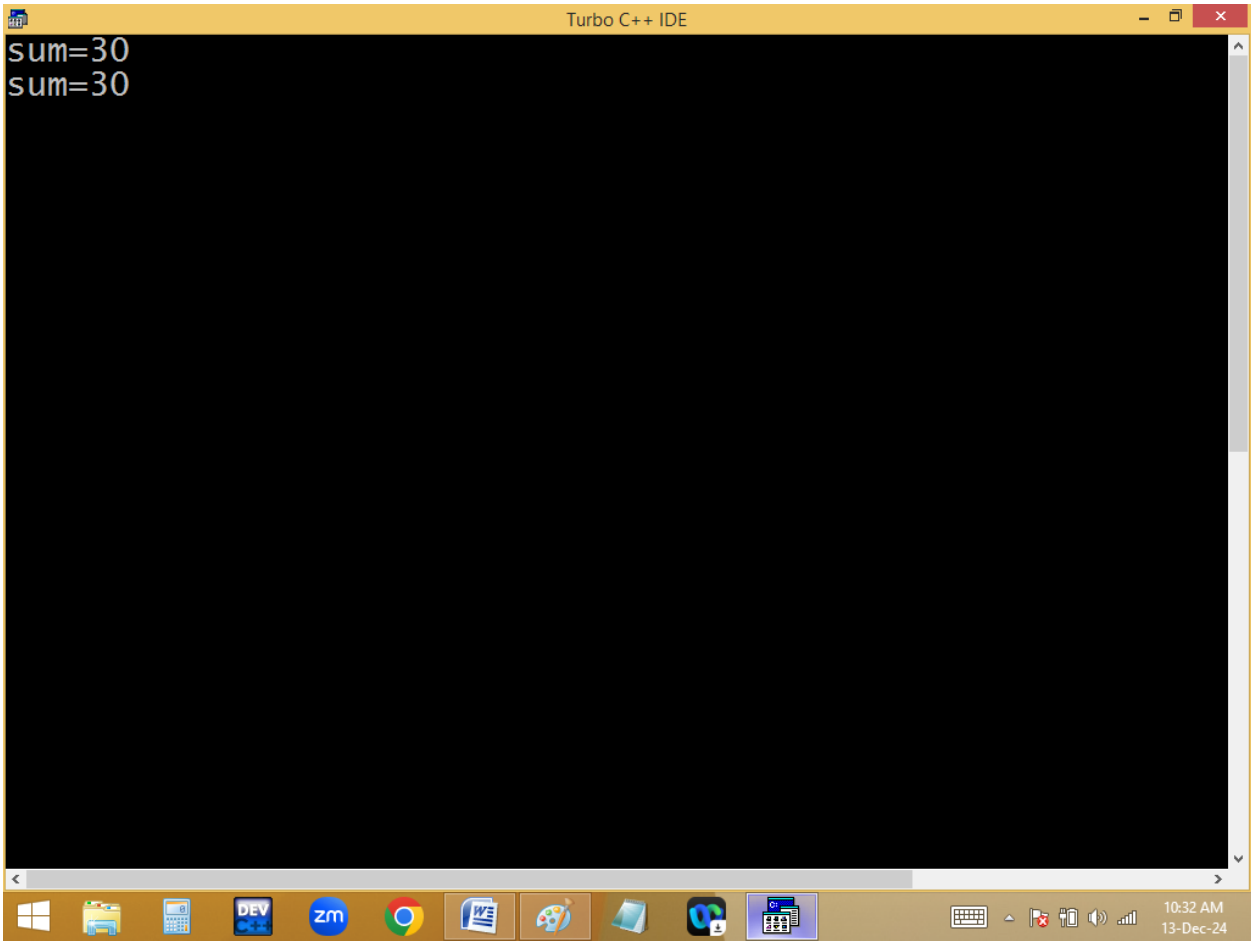


```
x=4, y=3_
```

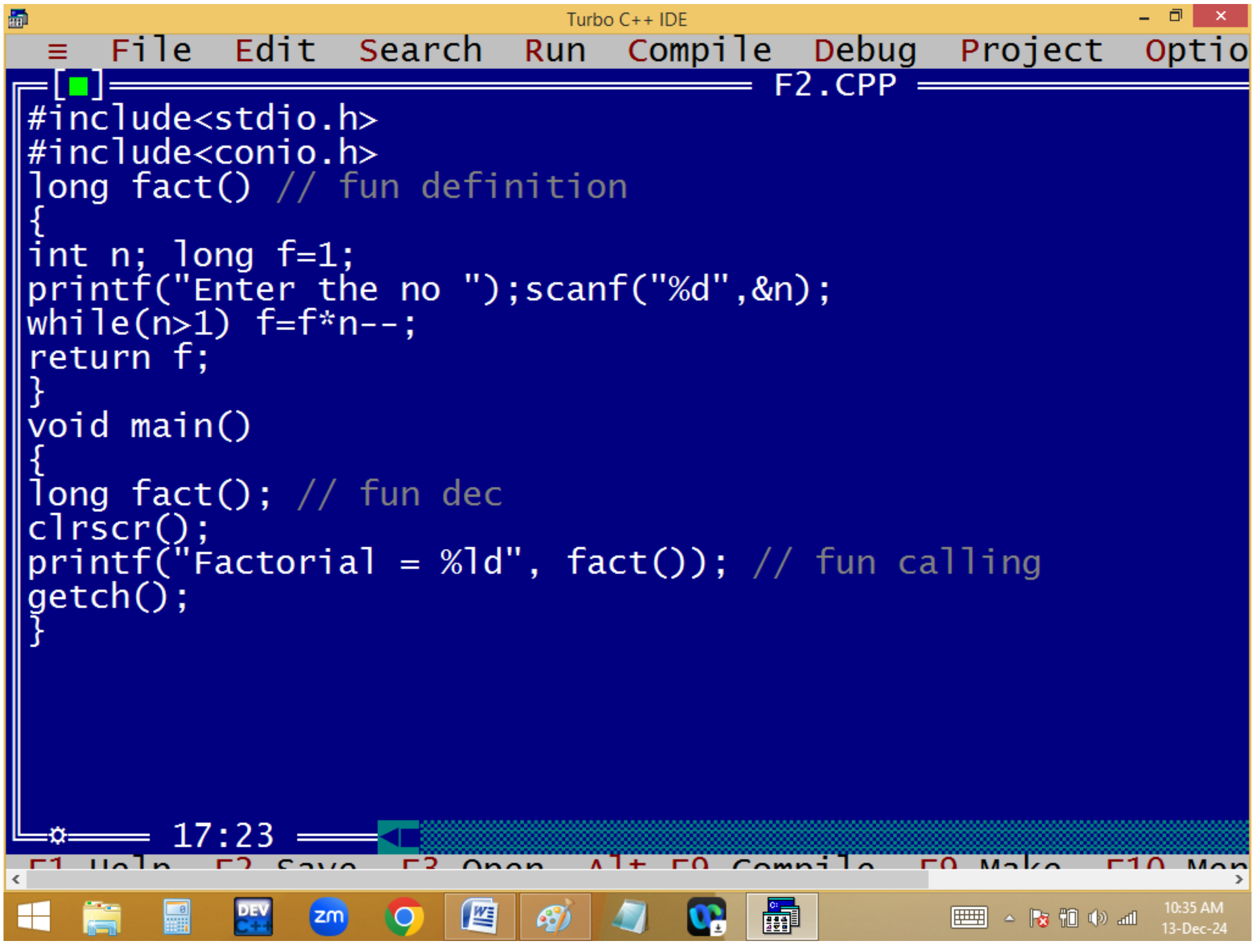
## Function without arguments, with return value:


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F2.CPP
11:29
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile
10:32 AM
13-Dec-24

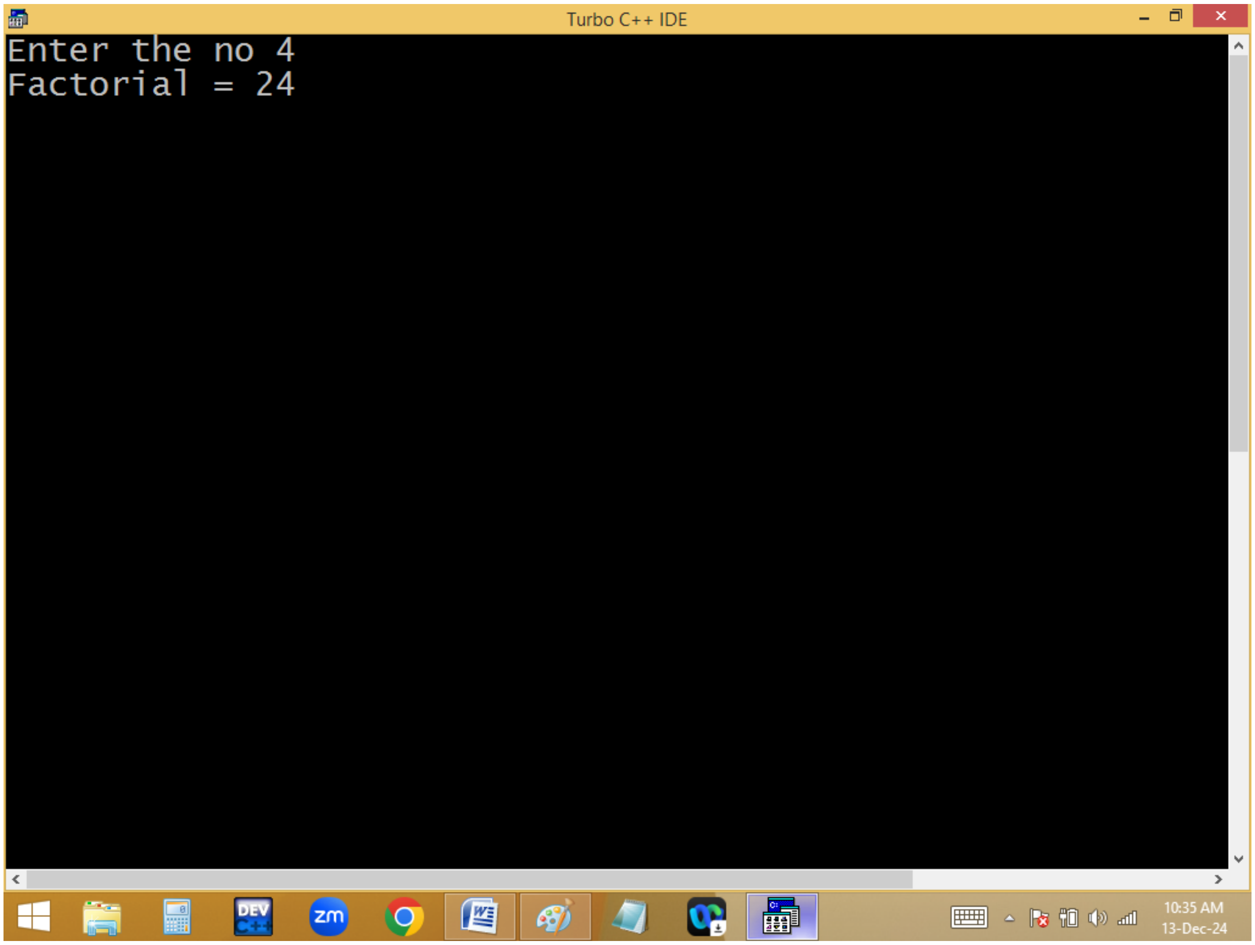
```
#include<stdio.h>
#include<conio.h>
int sum(); // fun declaration
void main()
{
int s;
clrscr();
s = sum(); // fun calling
printf("sum=%d\n",s);
printf("sum=%d\n",sum());
getch();
}
int sum() // fun definition
{
int a=10, b=20;
return a+b;
}
```



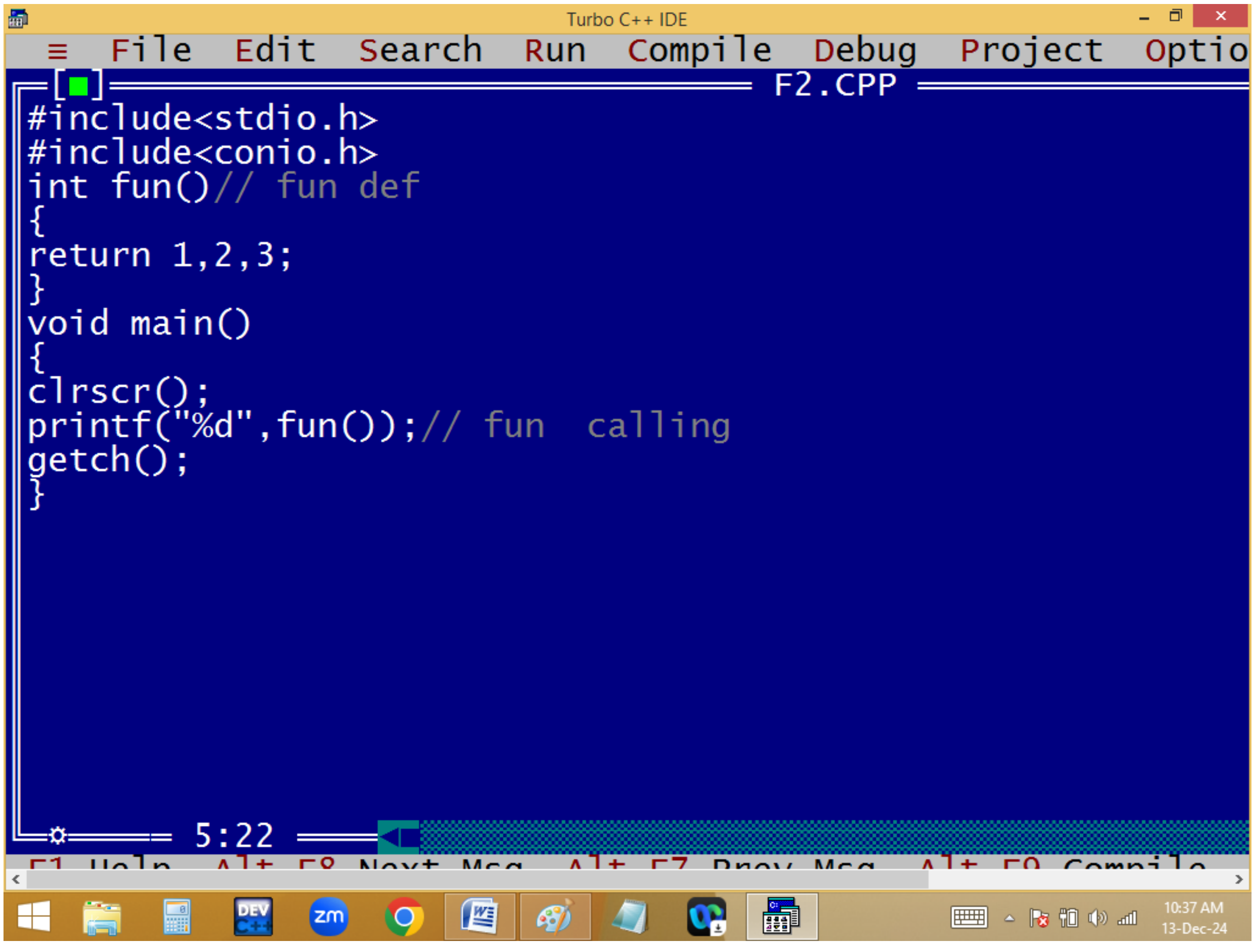
```
sum=30
sum=30
```


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F2.CPP
17:23
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Men
10:35 AM
13-Dec-24

```
#include<stdio.h>
#include<conio.h>
long fact() // fun definition
{
int n; long f=1;
printf("Enter the no ");scanf("%d",&n);
while(n>1) f=f*n--;
return f;
}
void main()
{
long fact(); // fun dec
clrscr();
printf("Factorial = %ld", fact()); // fun calling
getch();
}
```

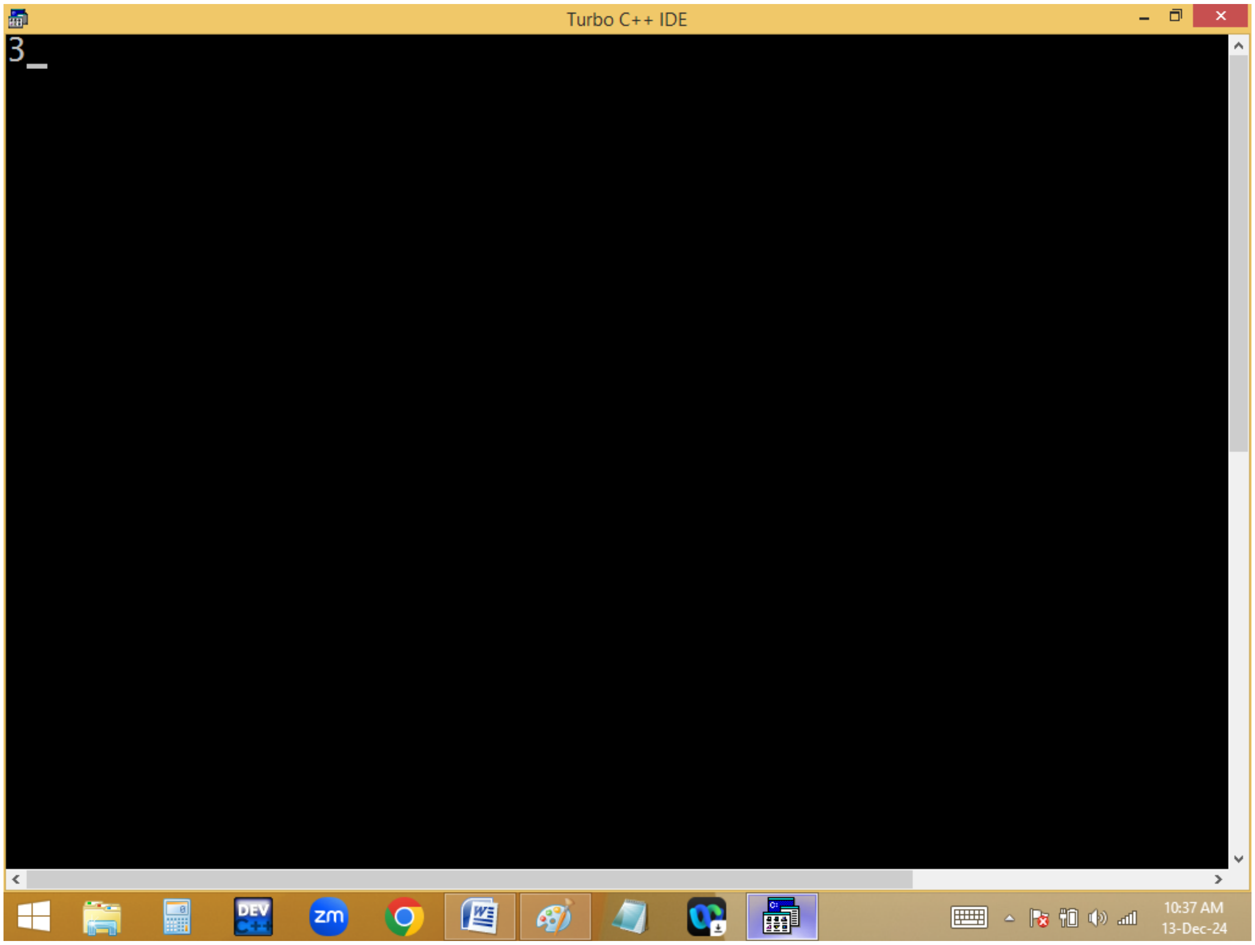


```
Enter the no 4
Factorial = 24
```

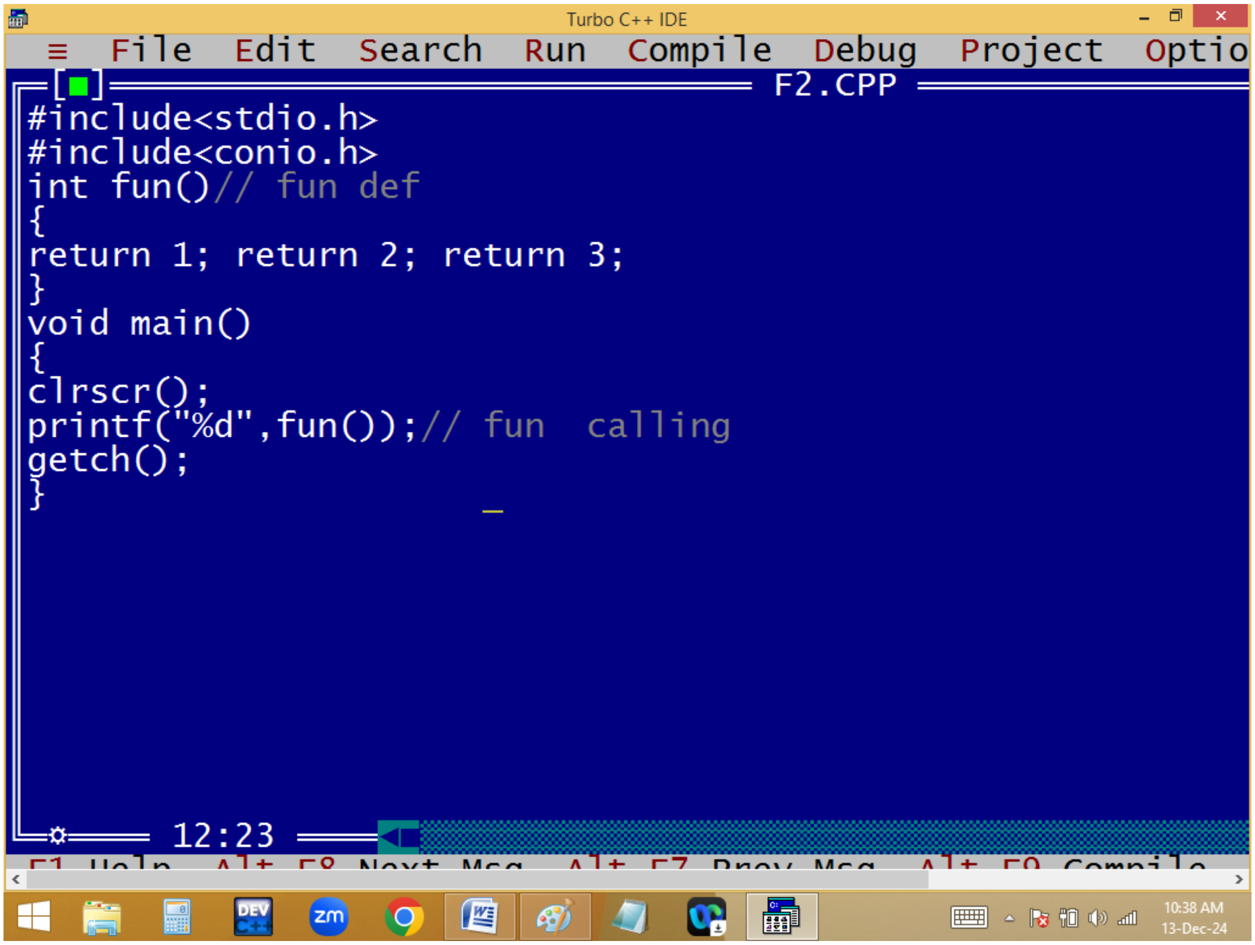


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F2.CPP
5:22
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile
10:37 AM
13-Dec-24

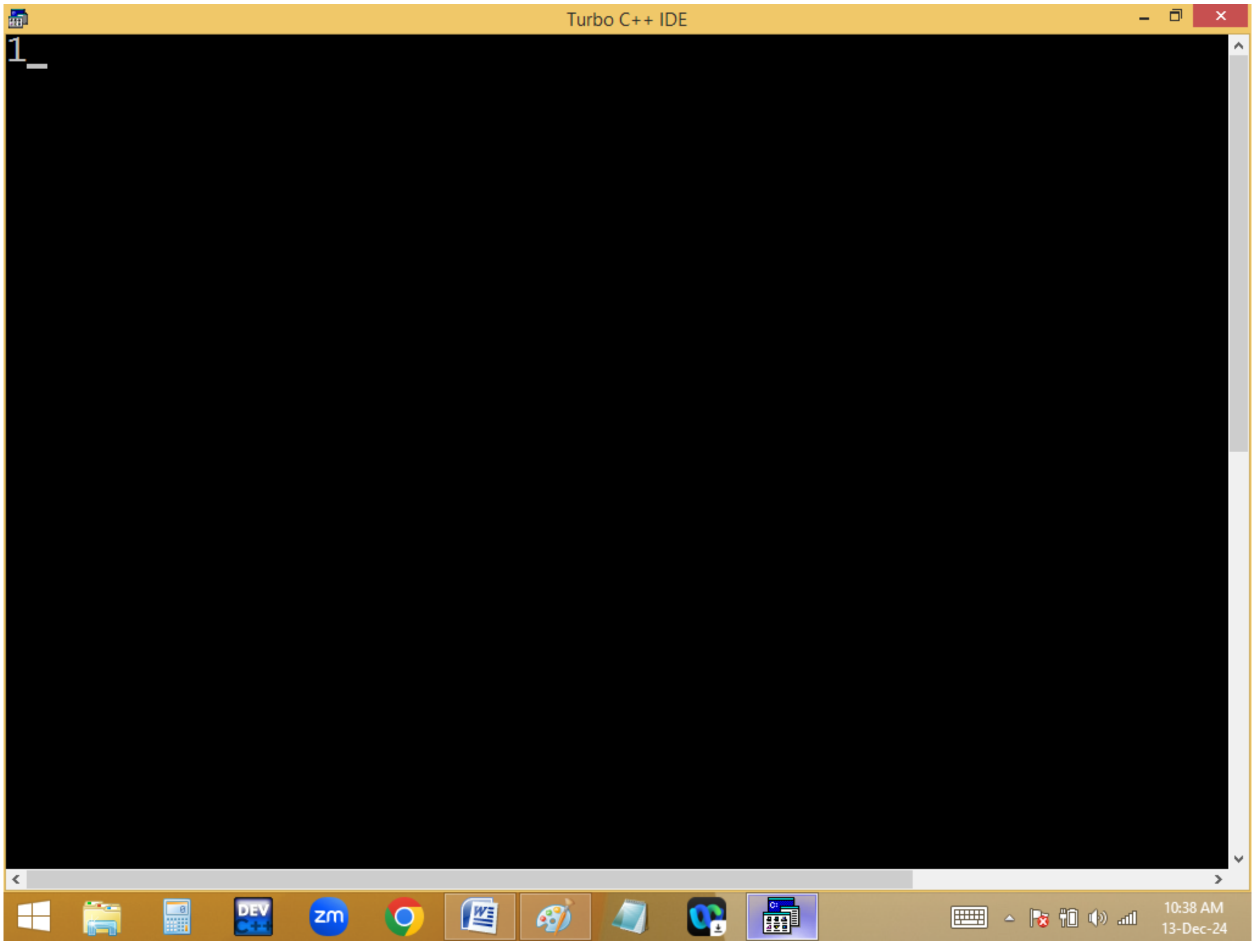
```
#include<stdio.h>
#include<conio.h>
int fun()// fun def
{
return 1,2,3;
}
void main()
{
clrscr();
printf("%d",fun());// fun  calling
getch();
}
```



```
3_
```


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F2.CPP
12:23
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile
10:38 AM
13-Dec-24

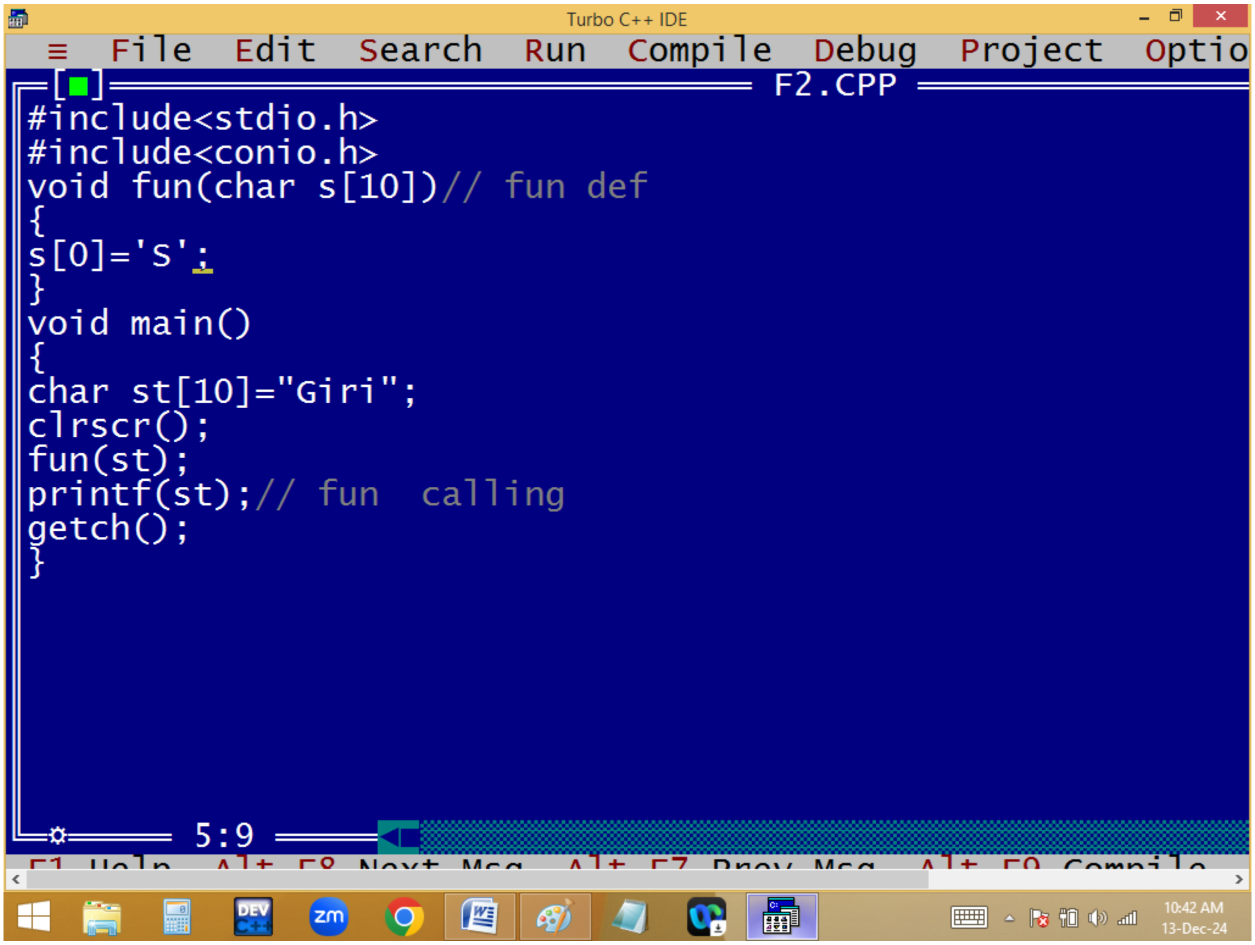
```
#include<stdio.h>
#include<conio.h>
int fun()// fun def
{
return 1; return 2; return 3;
}
void main()
{
clrscr();
printf("%d",fun());// fun  calling
getch();
}
```



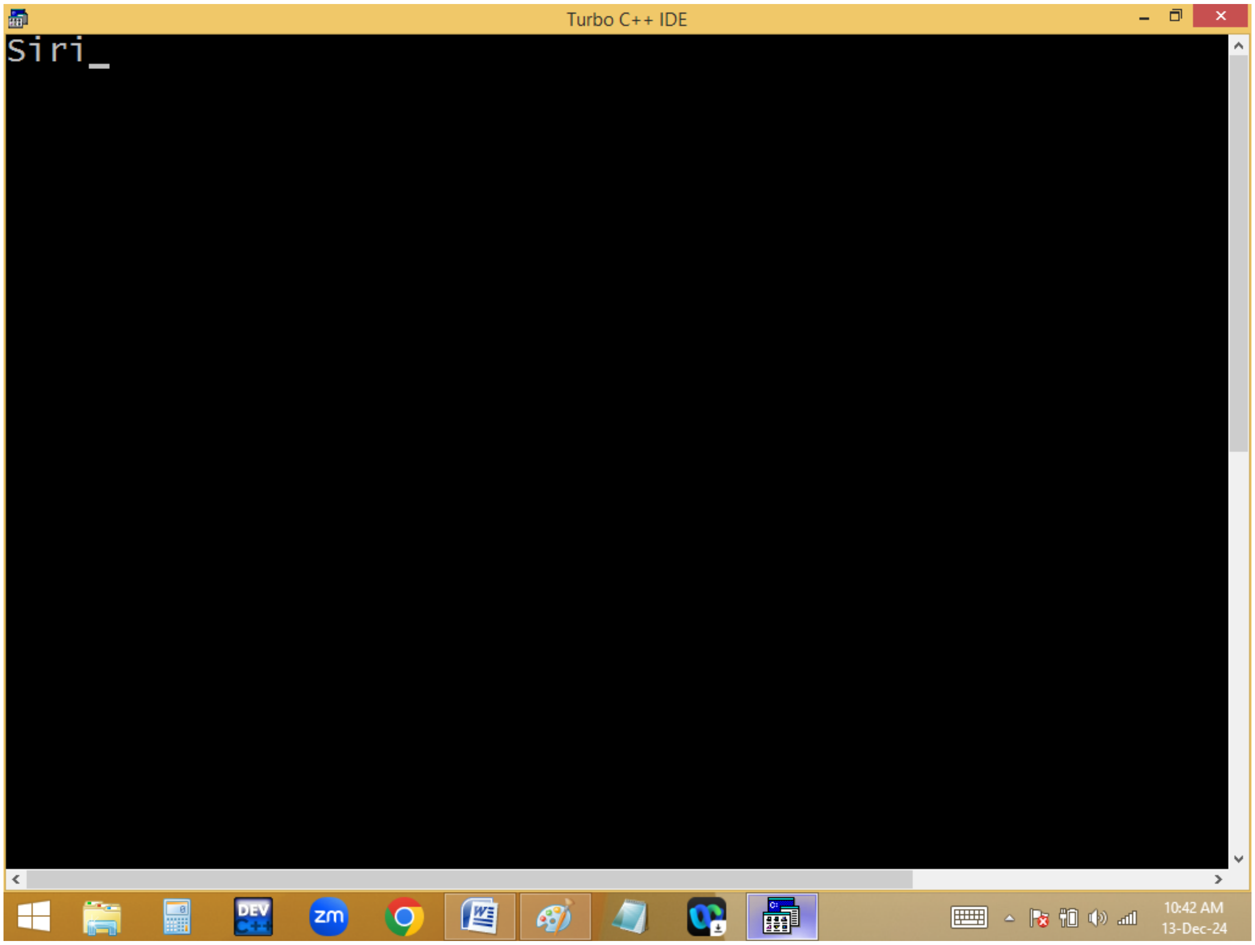
```
1
```

## Passing array / string to function:

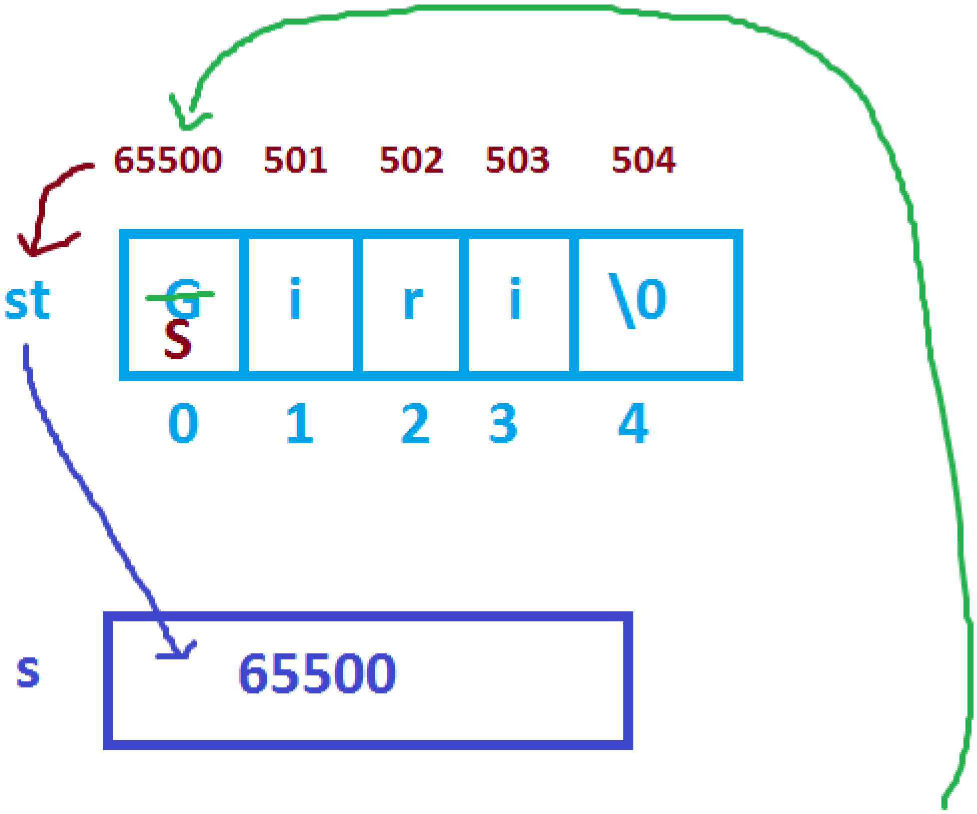
### Call by reference / call by address:



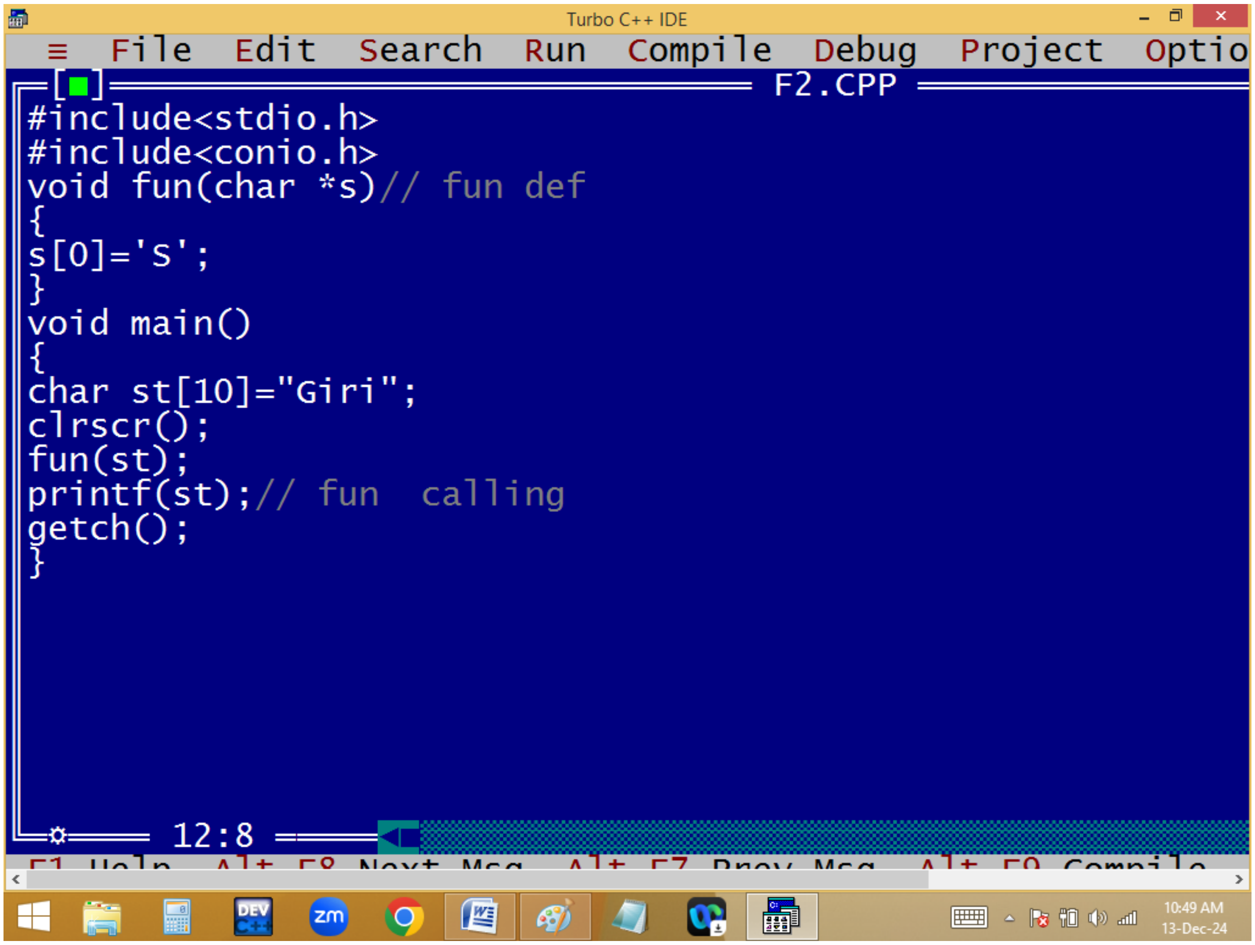
```
#include<stdio.h>
#include<conio.h>
void fun(char s[10])// fun def
{
s[0]='S';
}
void main()
{
char st[10]="Giri";
clrscr();
fun(st);
printf(st);// fun  calling
getch();
}
```



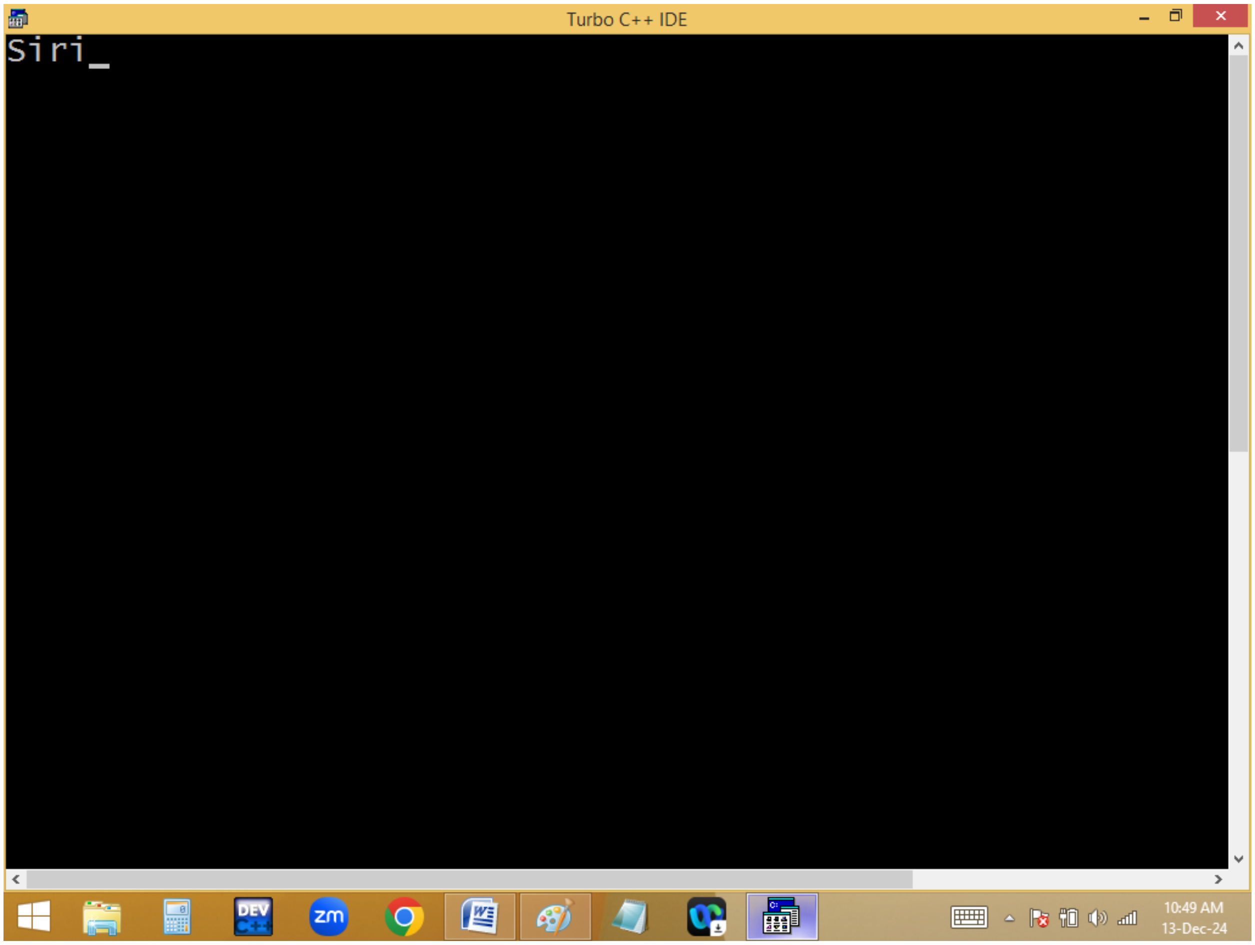
```
Siri_
```



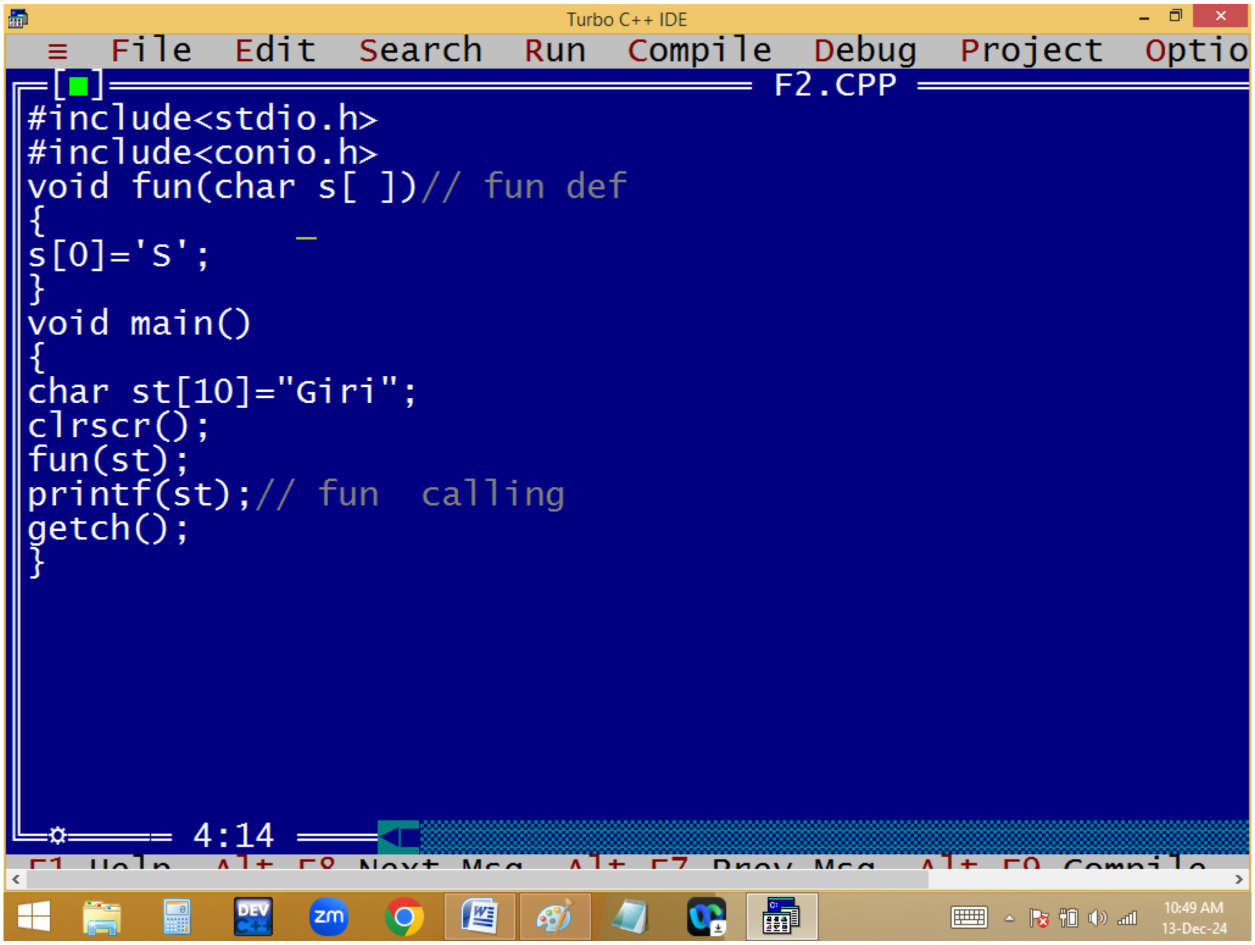
s[0]='G'; 65500+0*1=65500 = 'S';



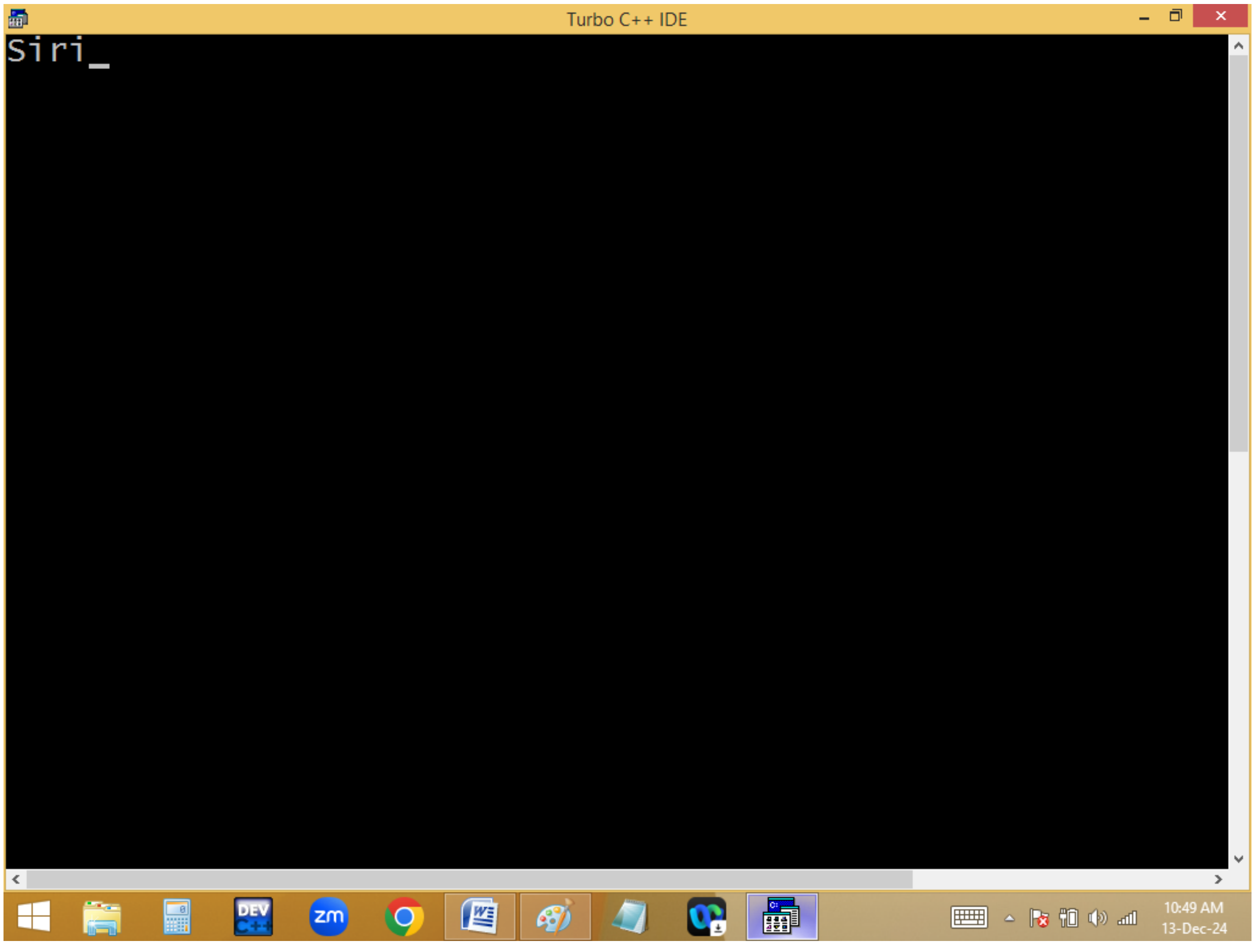
```
#include<stdio.h>
#include<conio.h>
void fun(char *s)// fun def
{
s[0]='S';
}
void main()
{
char st[10]="Giri";
clrscr();
fun(st);
printf(st);// fun  calling
getch();
}
```



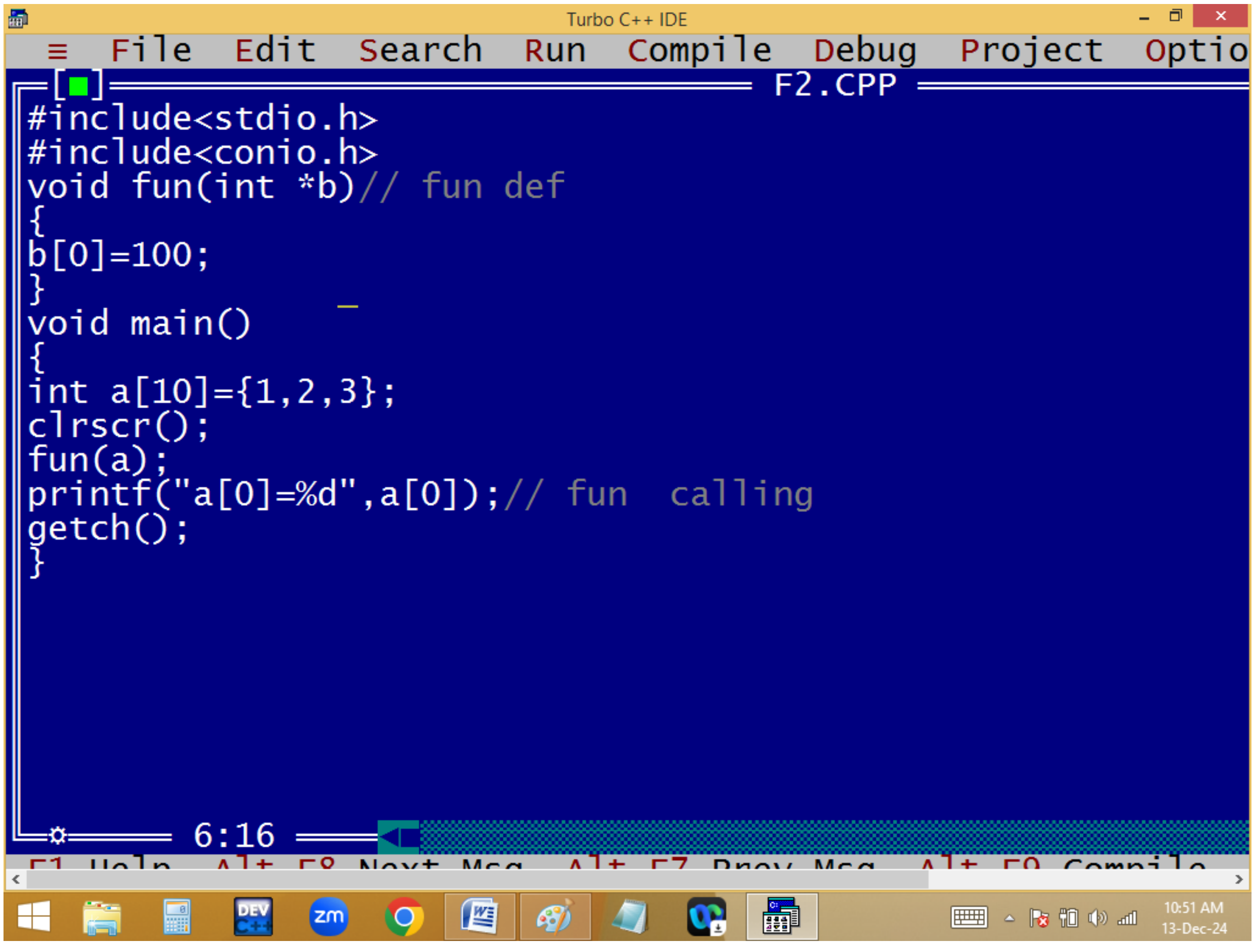
```
Siri_
```


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F2.CPP
4:14
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile
10:49 AM
13-Dec-24

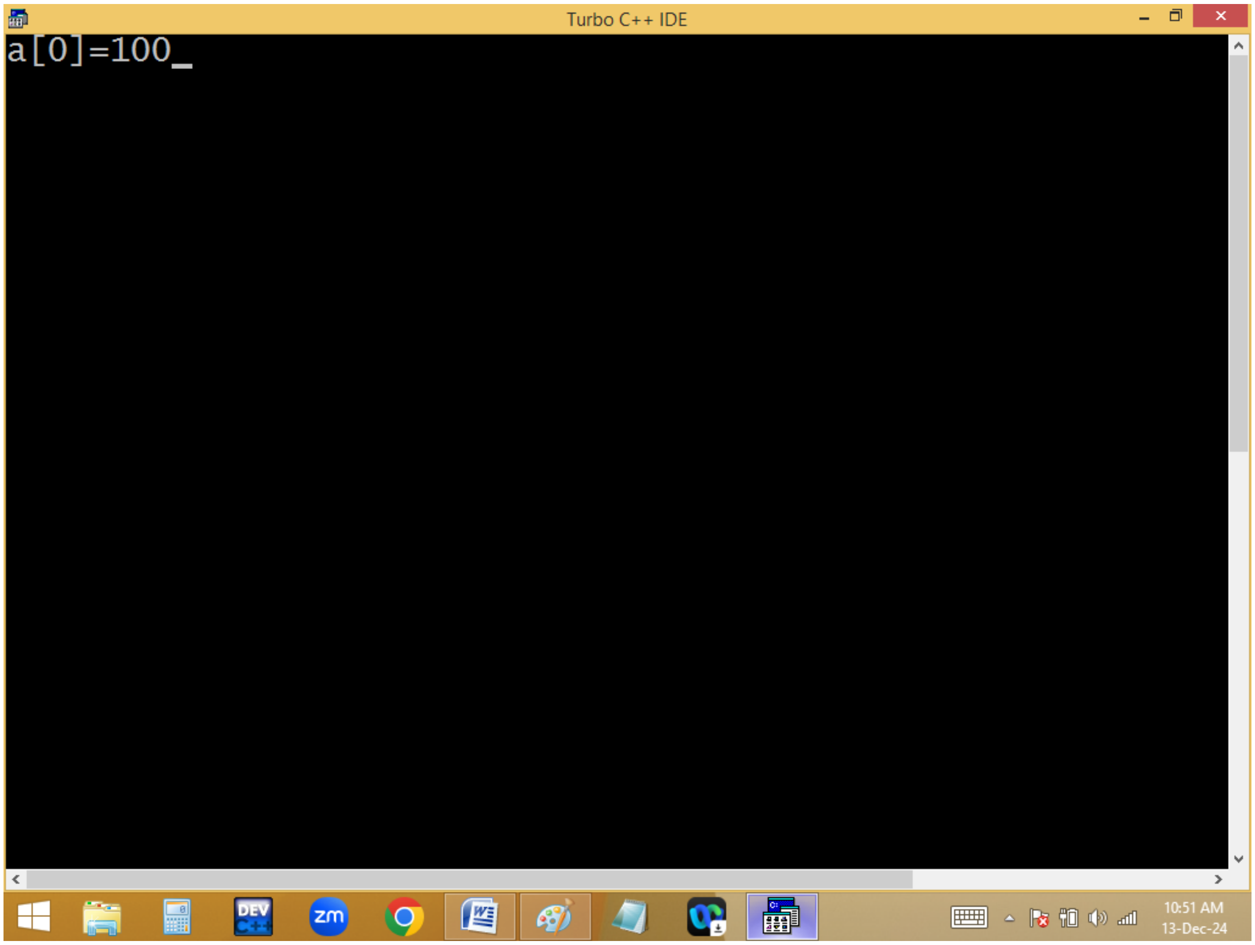
```
#include<stdio.h>
#include<conio.h>
void fun(char s[ ])// fun def
{
s[0]='S';
}
void main()
{
char st[10]="Giri";
clrscr();
fun(st);
printf(st);// fun  calling
getch();
}
```



```
Siri_
```


Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project Optio
F2.CPP
6:16
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile
10:51 AM
13-Dec-24

```
#include<stdio.h>
#include<conio.h>
void fun(int *b)// fun def
{
b[0]=100;
}
void main()
{
int a[10]={1,2,3};
clrscr();
fun(a);
printf("a[0]=%d",a[0]);// fun  calling
getch();
}
```



```
a[0]=100_
```