Function without arguments, with return value.

Finding factorial using a user defined function.

TC

File Edit Run Compile Project Options Debug Break/watch

Edit

Line 16 Col 2 Insert Indent Tab Fill Unindent * C:NONAME.C

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

TC

```
Enter a no 4
Factorial=24
```

n=4 3 2 1

```
while(n>1) f=f*n--;

return f;

main() { p("%ld", fact()); }
```

f
1×4=4
4×3=12
12×2=24

24

Eg. Finding strong no.

1 ! = 1

2! = 2

145 ➔ 1! + 4! + 5! = 1 + 24 + 120 = **145**

```c
#include<stdio.h>
#include<conio.h>
int strong(); /* fun dec */
void  main()
{
clrscr(); puts(strong()?"Strong no":"Not Strong no"); /* fun calling */
getch();
}
int  strong()  /* fun def */
{
int  n,m,f,r,s=0; printf("Enter a no "); scanf("%d",&n);
for(m=n;m!=0;m=m/10)
{
for(r=m%10,f=1; r>1; r-- ) f=f*r;
s=s+f;
}
return  n==s?1:0;
}
```

```
Enter a no 145
Strong no
```

TC

```
Enter a no 1
Strong no
```

6:40 PM
07/26/2022

TC

```
Enter a no 2
Strong no
```

6:41 PM
07/26/2022

```
Enter a no 3
Not Strong no
```

```
Enter a no 143
Not Strong no
```

$$\overset{m}{\overline{1\,4\,\cancel{5}}} \;\%\; 10 = \overset{r}{\overline{5}}\,! = \overset{s}{\overline{120}}$$

$$1\,\cancel{4} \;\%\; 10 = 4\,! = 24$$

$$1 \;\%\; 10 = 1\,! = 1$$

$$\overset{m}{\overline{145}}$$

```
File   Edit   Run   Compile   Project   Options   Debug   Break/watch
     Line 13    Col 8   Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
int  show()
{
return 1,2,3;
}
void main()
{
clrscr();
printf("%d",show());
getch();
}
/* 3 */_
```

```
File   Edit   Run   Compile   Project   Options   Debug   Break/watch
     Line 13    Col 5   Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
int  show()
{
return 1; return 2; return 3;
}
void main()
{
clrscr();
printf("%d",show());
getch();
}
/* 1_*/
```

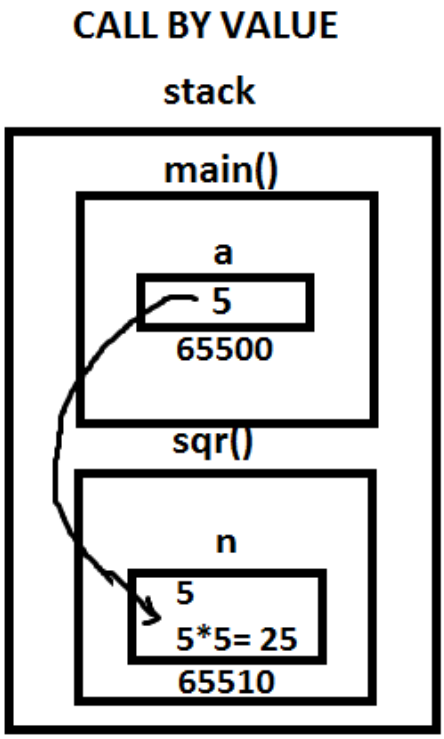
Passing parameters to the functions: [parameter passing techniques]

In C-Language, we can send the arguments to the functions in 2 ways.

1. Call by value / pass by value.
2. Call by address / pass by address. [call by reference]

<u>Call by value / pass by value:</u>

In call by value we are sending actual parameter value to the formal parameter. Later there is no relation is maintained in between actual and formal parameters. Due to this any change in formal parameter doesn't effects the value of actual parameter.



Eg: 1

```
#include<stdio.h>
#include<conio.h>

void  sqr(int n)

{

n = n * n;

}  /* n  deleted after the function execution
*/

void main()

{

int a=5;

clrscr();

printf("Before function call a = %d\n" ,a);

sqr(a);  /* fun calling */
```

```
printf("After function call a = %d", a):
getch();
}
```

Output:

Before function call a = 5

After function call a = 5

Eg: 2 swapping of two integers

```
#include<stdio.h>
#include<conio.h>
void  swap(int  a, int  b)
{
int  temp=a;
a=b;
b=temp;
}
void  main()
{
int  a=5, b=7;
clrscr();
```

```
printf("Before fun call a=%d, b=%d\n" , a , b);
swap(a, b);
printf("After fun call a=%d, b=%d", a , b);
getch();
}
```

Output:

```
Before fun call a=5, b=7
After fun call a=5, b=7
```

TC

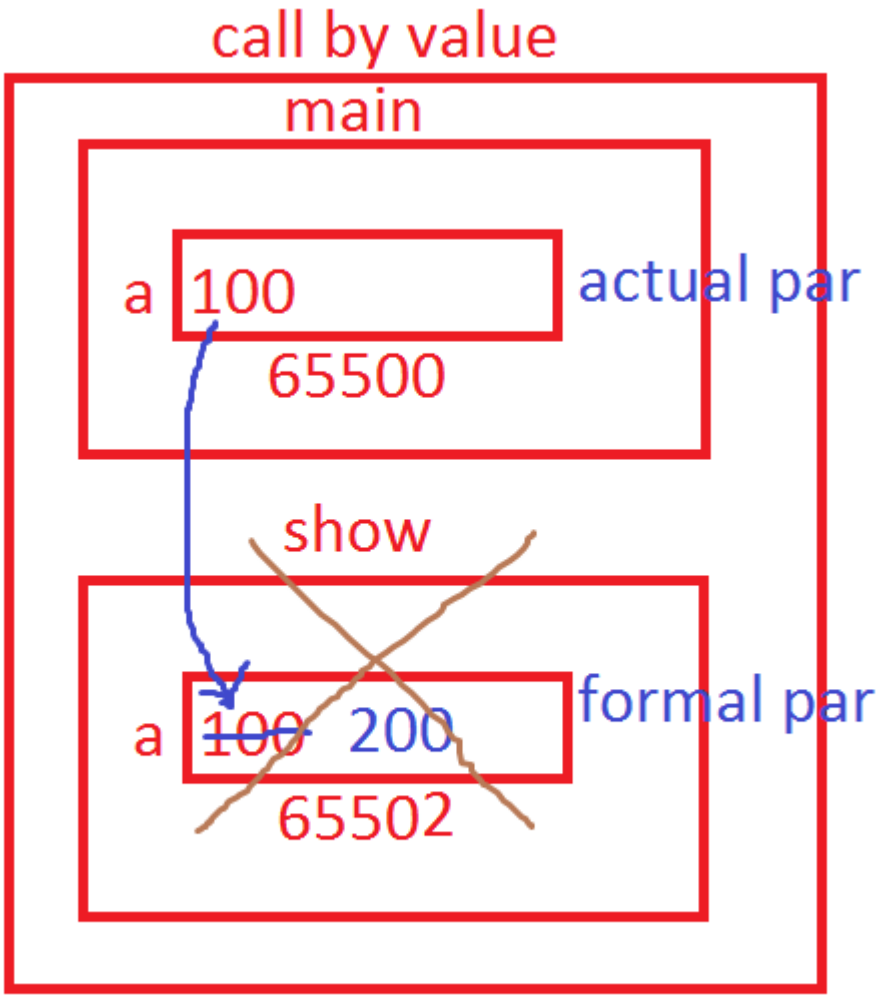
File Edit Run Compile Project Options Debug Break/watch

Line 5 Col 1 Insert Indent Tab Fill Unindent * C:NONAME.C

```c
#include<stdio.h>
#include<conio.h>
void  show(int  a) /* fun def */
{
a=200;
}

void main()
{
int a=100; /* local var */
clrscr();
printf("before fun call a=%d\n",a);
show(a); /* fun calling */
printf("after  fun call a=%d",a);
getch();
}
```

TC

```
before fun call a=100
after  fun call a=100
```



Call by address [Reference]:

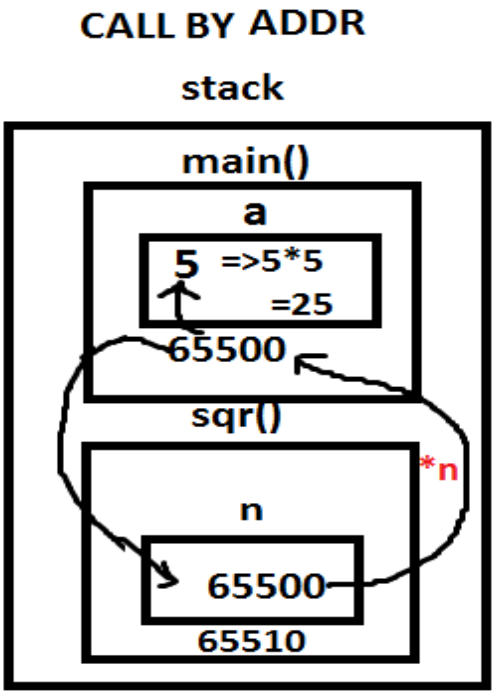
In call by address, the address of actual parameter is passed to formal parameter. Due to this the formal parameter should be declared as a pointer. Then only the formal parameter receives the actual parameter address. Due to this any changes in formal parameter effects in actual parameter address i.e. actual parameter value.

Hence pointers allows the local variables to access outside the functions and this process is called call by address / reference.

It is very much useful in handling the strings, arrays etc outside the functions.

Eg: 1

```
#include<stdio.h>
#include<conio.h>

void  sqr(int *n)

{

*n = *n   *   *n;

}

void main()

{

int a=5;

clrscr();

printf("Before function call a = %d\n " , a);

sqr(&a);  /* fun calling with address */

printf("After function call a = %d " , a);

getch();

}
```



Output:

Before function call a = 5

After function call a = 25

Eg: 2 Swap of two integers

```
#include<stdio.h>
#include<conio.h>
void swap(int *a, int *b)
{
int temp=*a;  *a = *b;  *b=temp;
}
void main()
{
int a=5, b=7;
clrscr();
printf("Before fun call a=%d, b=%d\n", a ,b);
swap(&a, &b);
printf("After fun call a=%d, b=%d", a ,b);
getch();
}
```

Output:

Before function call a=5, b=7

After function call a=7, b=5

Eg.

TC

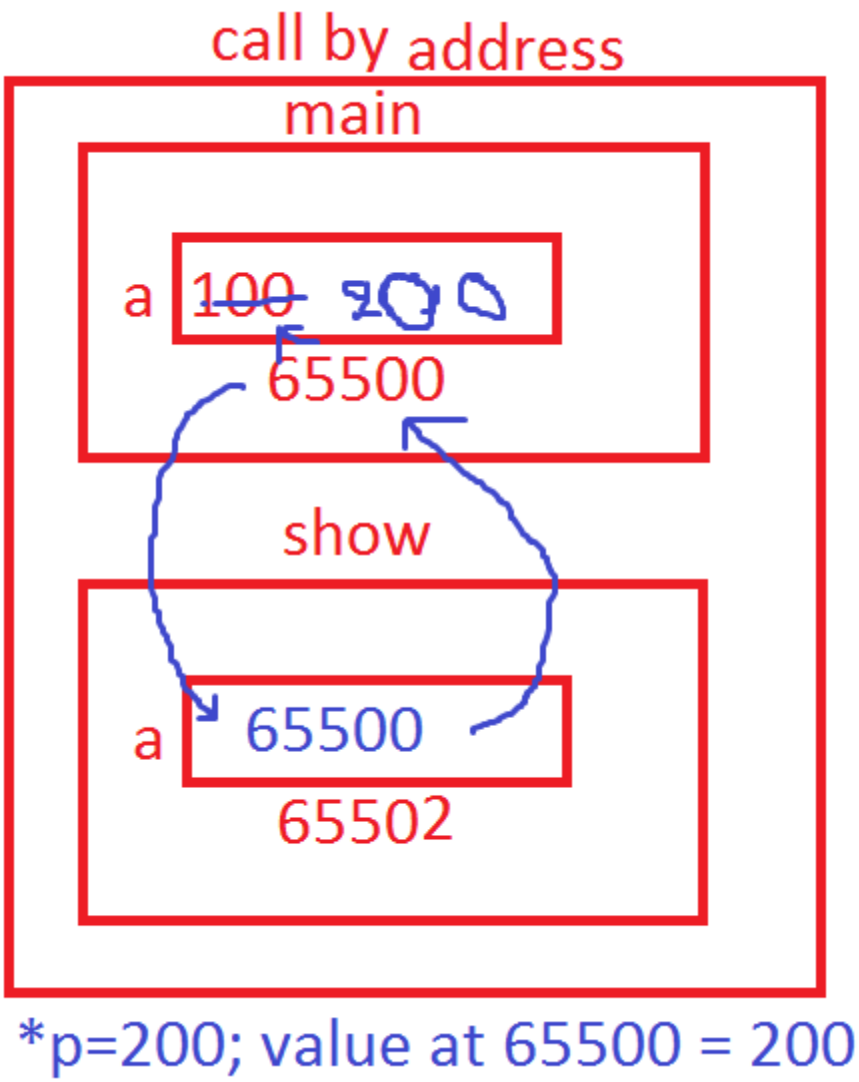
File Edit Run Compile Project Options Debug Break/watch

Line 9 Col 27 Insert Indent Tab Fill Unindent * C:NONAME.C

```
#include<stdio.h>
#include<conio.h>
void  show(int * a) /* fun def */
{
*a=200;
}

void main()
{
int a=100; /* local var */
clrscr();
printf("before fun call a=%d\n",a);
show(&a); /* fun calling */
printf("after  fun call a=%d",a);
getch();
}
```

TC

```
before fun call a=100
after  fun call a=200
```



PASSING STRING / ARRAY TO FUNCTION

String/array is implicit pointer i.e. string / array variable stores base address. Due to this when string/array is passed to a function, implicitly base address is passed and formal parameter becomes pointer and it receives this address. Hence any change occurred in formal parameter, effects on actual parameter value also.

We can declare string / array formal parameter in 3 ways.

1. With size eg: char st[50] / int a[3]
2. Without size eg: char st[ ] / int a[ ]

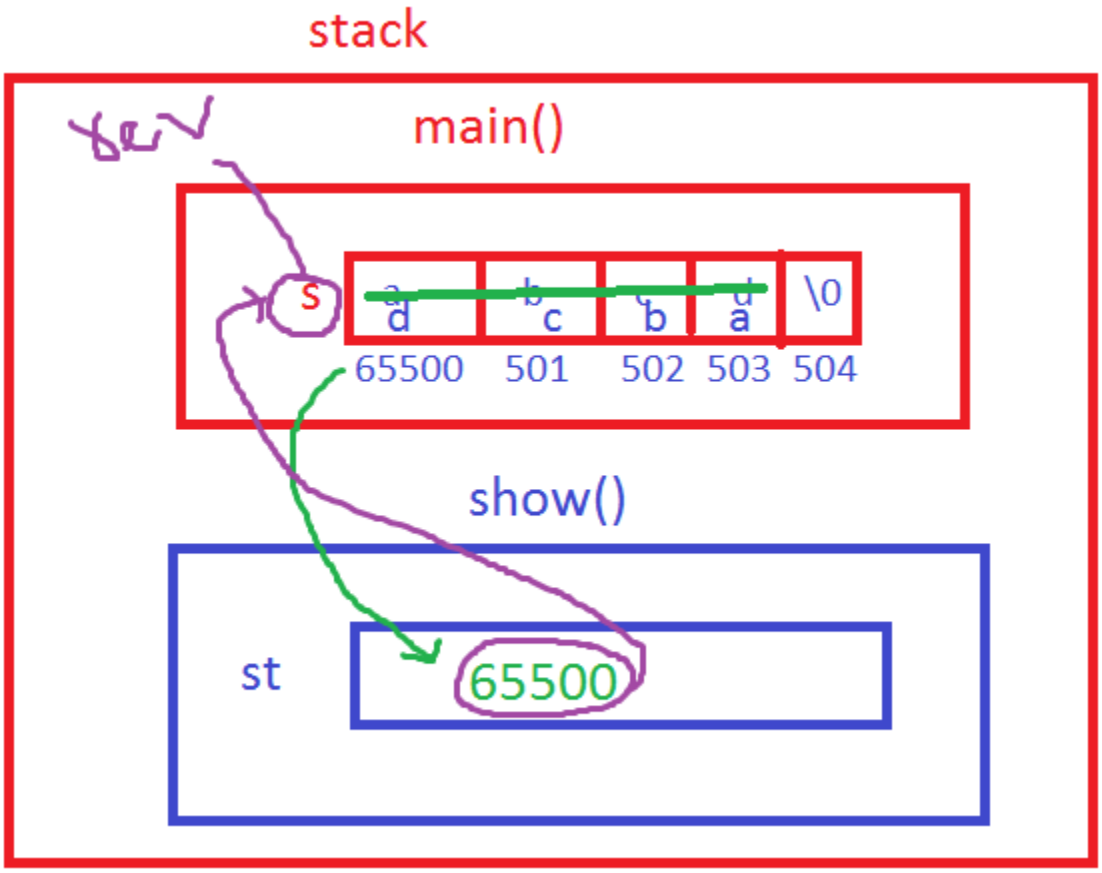
3. As a pointer eg: char * st / int *a

We can pass string / array actual parameter with or without address.

Eg:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void  reverse( char st[10] )    or  st[ ]  or  *st
{
strrev(st);
}
void  main()
{
char  s[10]="abcd”;
clrscr();
reverse(s);  or  reverse(&s);
printf("String = %s", s);
getch();
}
```

O/P: String = dcba



Passing array to function:

```
#include<stdio.h>

#include<conio.h>

void show(int a[3])  or  a[ ]  or  *a
```

```
{
a[0]=100; a[1]=200; a[2]=300;
}
void  main()
{
int  a[3]={10,20,30};
clrscr();
show(a);   or  show(&a);
printf("Array elements %d %d %d",a[0],a[1],a[2]);
getch();
}
```

O/P: Array elements 100   200   300

Passing two – dimensional array to function.

```
#include<stdio.h>
#include<conio.h>
void  show( int (*a)[3] ) or a[2][3] or a[ ][3]
{
a[0][0]=10;  a[1][2]=60;
}
void  main()
{
int a[2][3]={1,2,3,4,5,6};
show(a); /* fun calling */
printf("a[0][0]=%d, a[1][2]=%d",a[0][0],a[1][2]);
getch();
}
```

Output: a[0][0]=10, a[1][2]=60;

Function returning address [pointer]

TC

File Edit Run Compile Project Options Debug Break/watch

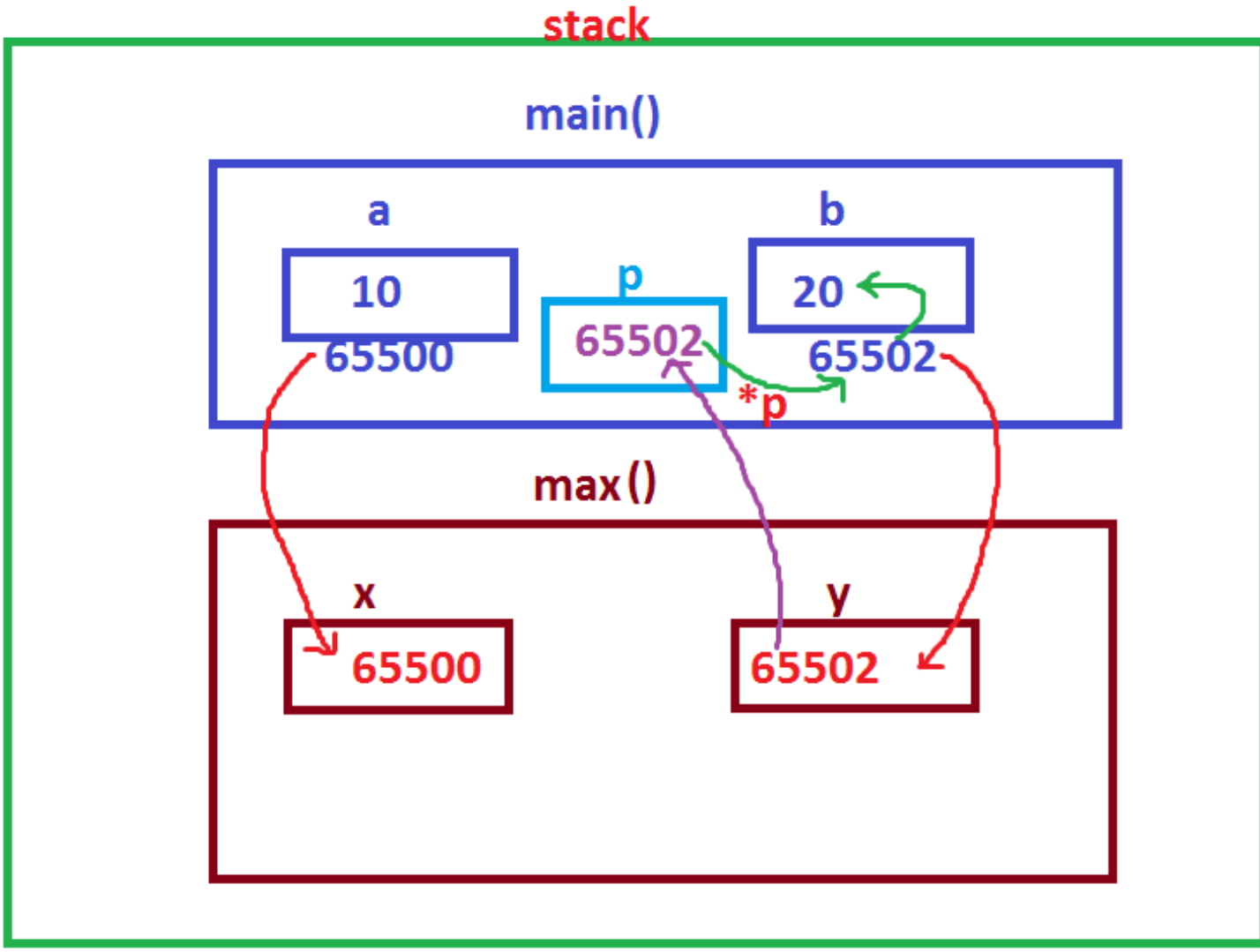
Edit

Line 11 Col 25 Insert Indent Tab Fill Unindent * C:NONAME.C

```
#include<stdio.h>
#include<conio.h>
int * max(int  *x, int *y)
{
return  *x > *y ? x : y;
}
void  main()
{
int  a=10,b=20, *p=max(&a, &b);
clrscr();
printf("%d is big", *p);
getch();
}
```

Watch


```
20 is big
```

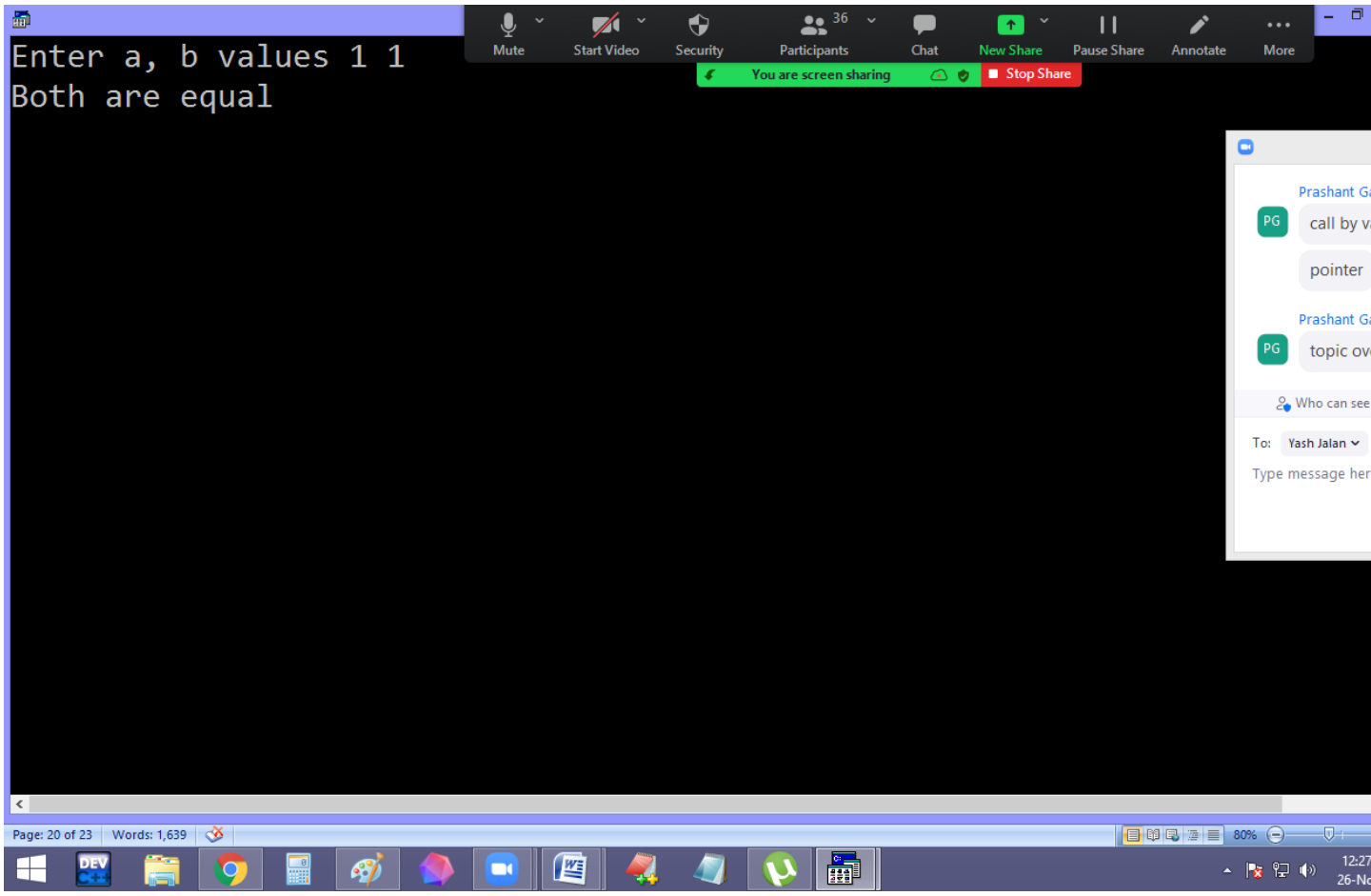

stack
main()
a
10
65500
p
65502
b
20
65502
*p
max()
x
65500
y
65502

File Edit Run Compile Project Debug Break/watch

Line 7 Col 1 Insert Indent Tab Fill Unindent * E:REV.C

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

Page: 16 of 16 Words: 658 100%


Enter a, b values 1 1
Both are equal
Mute
Start Video
Security
Participants
Chat
New Share
Pause Share
Annotate
More
You are screen sharing
Stop Share
Prashant Ga
call by va
pointer
Prashant Ga
topic ove
Who can see
To: Yash Jalan
Type message her
Page: 20 of 23
Words: 1,639
80%