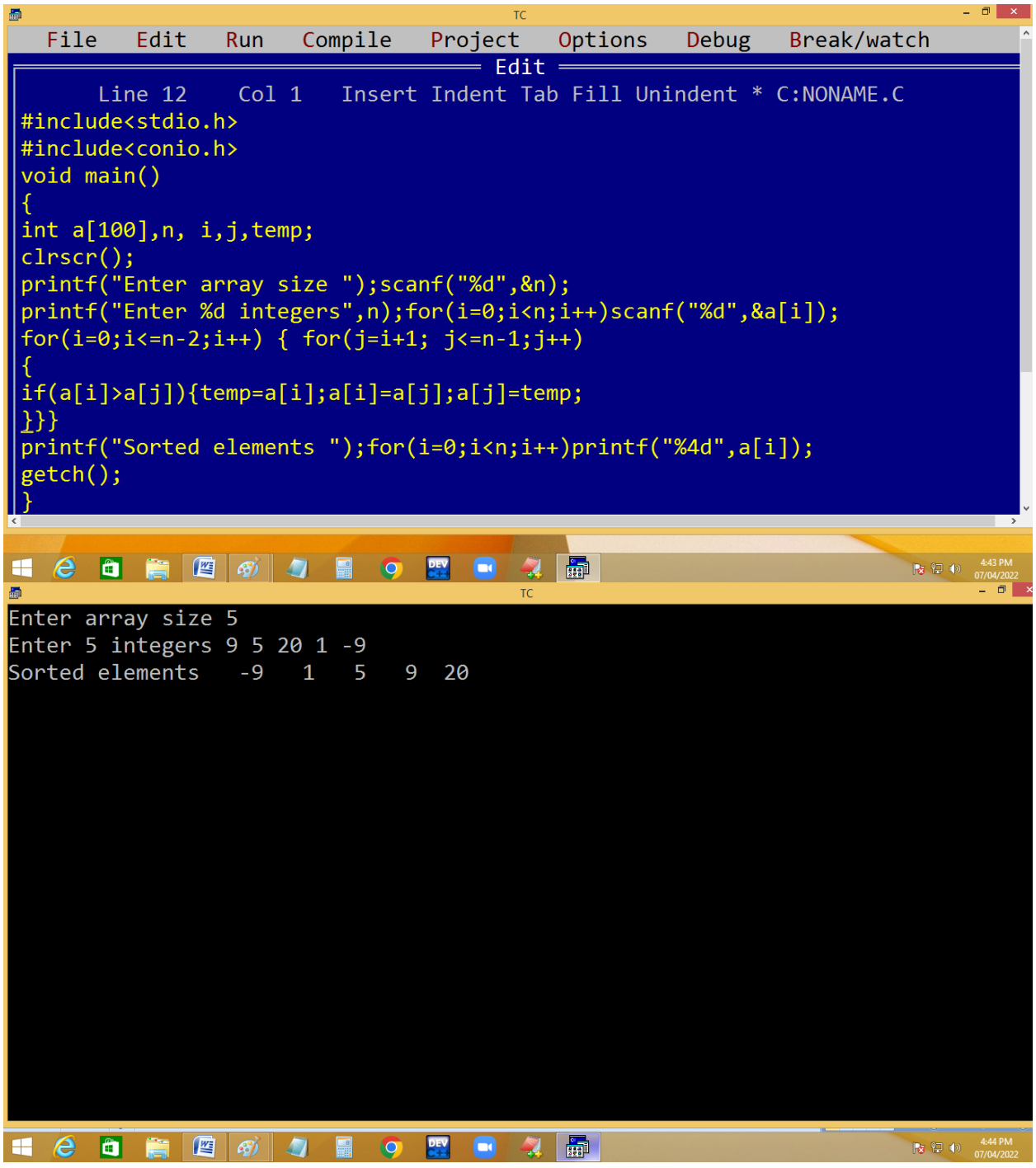
SELECTION SORT:

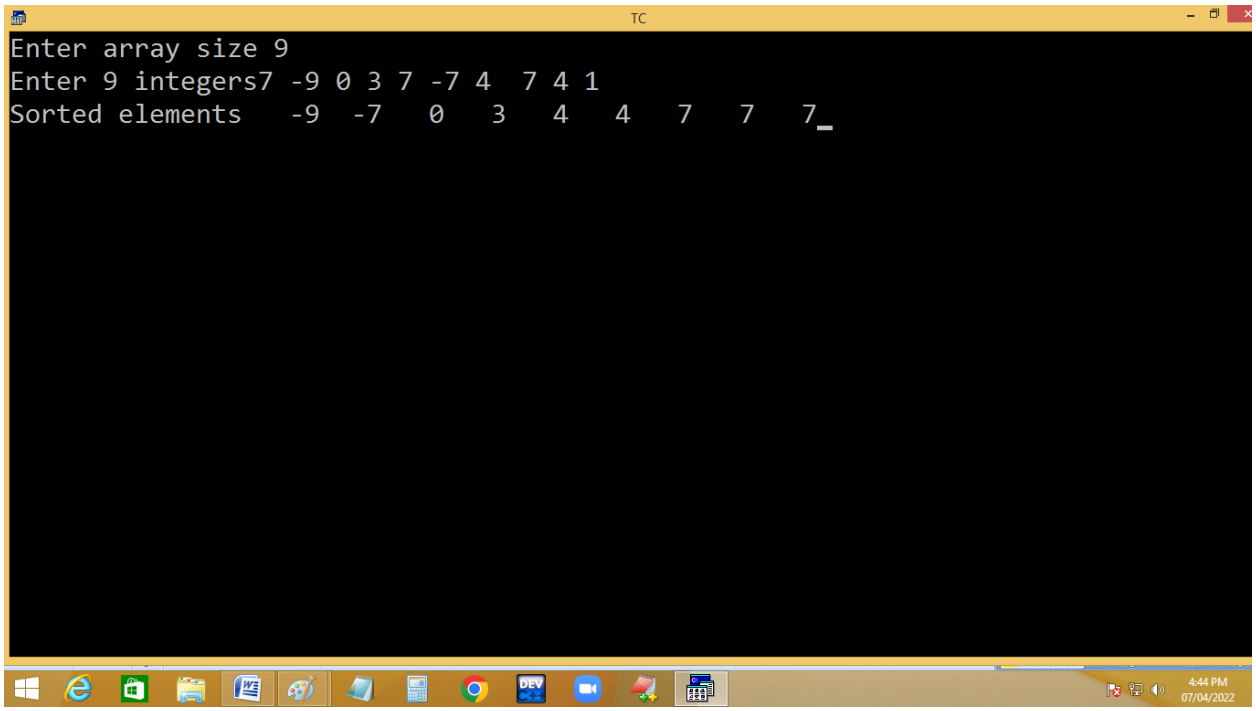
selection sort:

9	5	20	1	-9
0	1	2	3	4

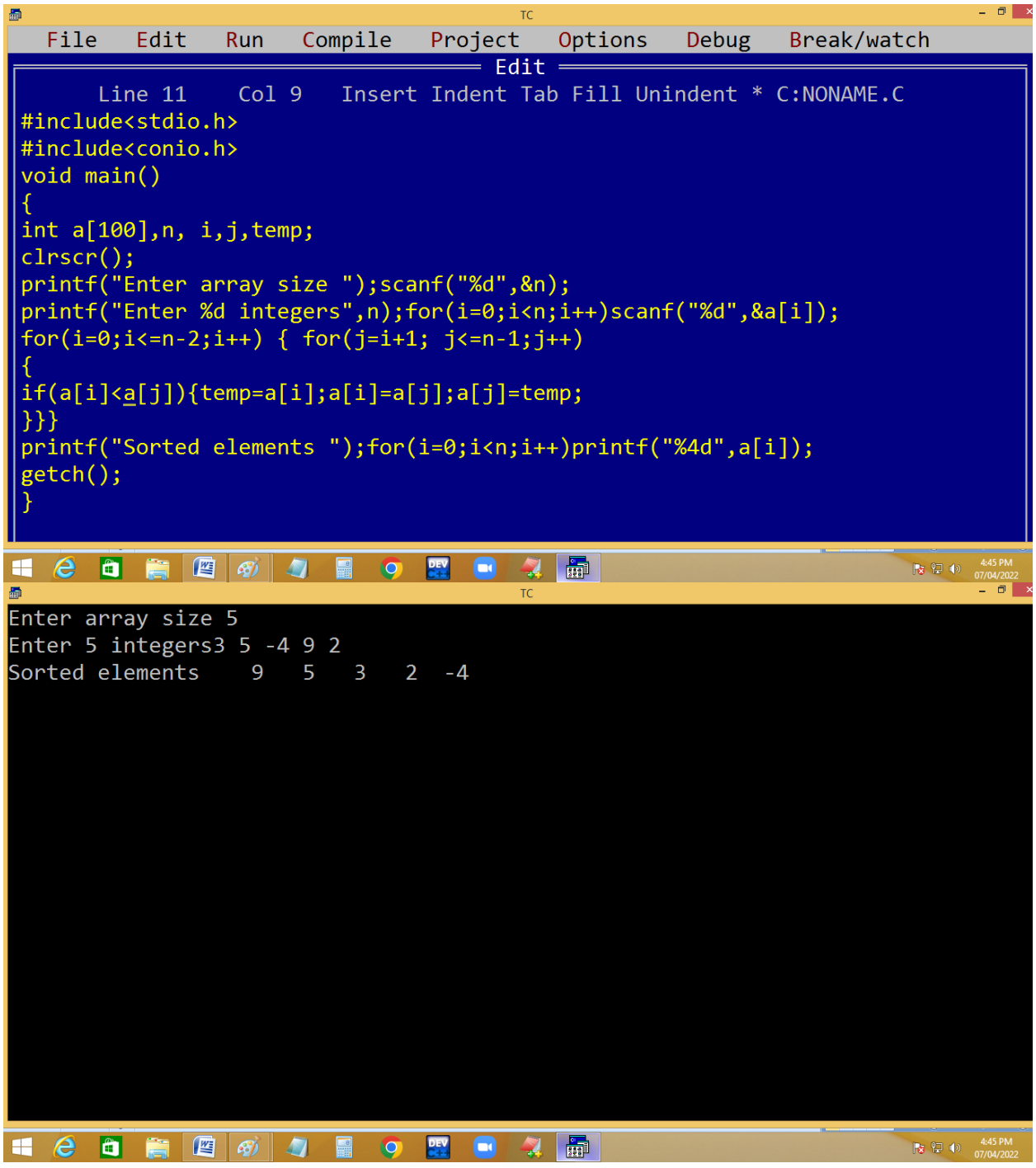
9 5 20 1 -9
5 9 20 1 -9
1 9 20 5 -9
-9 9 20 5 1
-9 5 20 9 1
-9 1 20 9 5
-9 1 9 20 5
-9 1 5 20 9
-9 1 5 9 20

L i	R j
0	1,2,3,4
1	2,3,4
2	3,4
3	4


TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 12 Col 1 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],n, i,j,temp;
clrscr();
printf("Enter array size ");scanf("%d",&n);
printf("Enter %d integers",n);for(i=0;i<n;i++)scanf("%d",&a[i]);
for(i=0;i<=n-2;i++) { for(j=i+1; j<=n-1;j++)
{
if(a[i]>a[j]){temp=a[i];a[i]=a[j];a[j]=temp;
}}}
printf("Sorted elements ");for(i=0;i<n;i++)printf("%4d",a[i]);
getch();
}
TC
Enter array size 5
Enter 5 integers 9 5 20 1 -9
Sorted elements -9 1 5 9 20

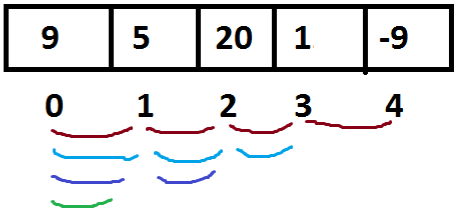


Descending order:

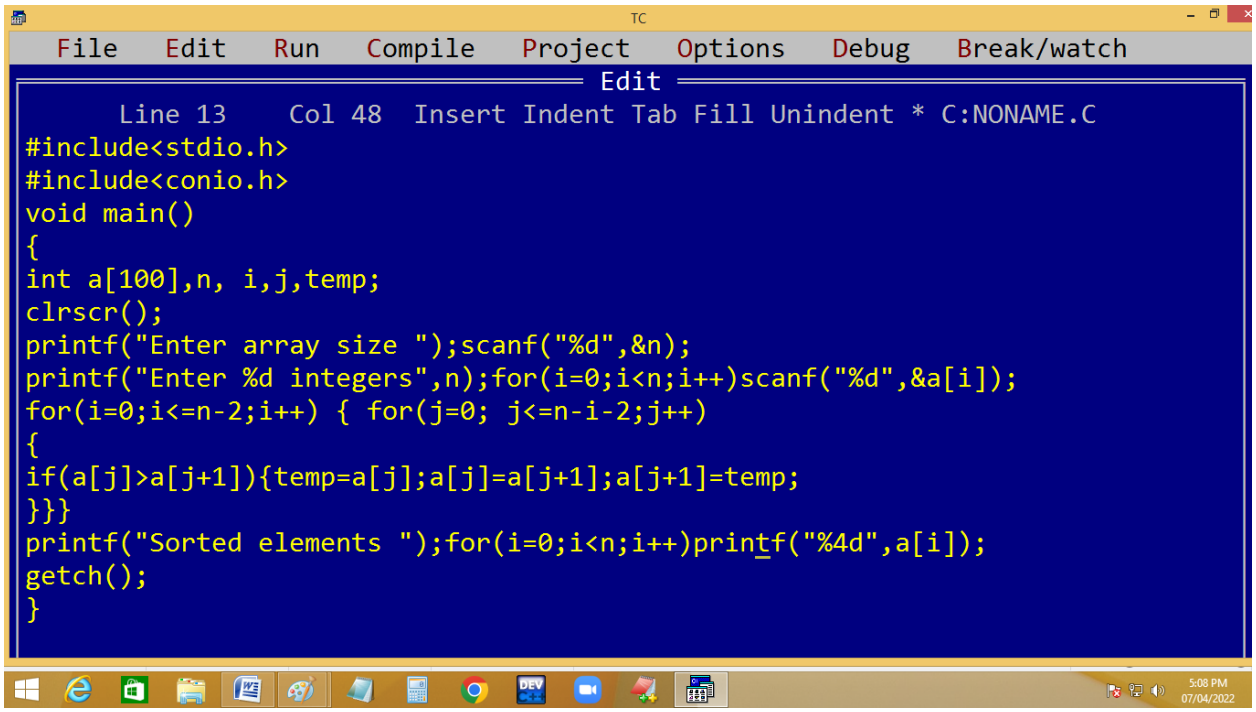

TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 11 Col 9 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],n, i,j,temp;
clrscr();
printf("Enter array size ");scanf("%d",&n);
printf("Enter %d integers",n);for(i=0;i<n;i++)scanf("%d",&a[i]);
for(i=0;i<=n-2;i++) { for(j=i+1; j<=n-1;j++)
{
if(a[i]<a[j]){temp=a[i];a[i]=a[j];a[j]=temp;
}}}
printf("Sorted elements ");for(i=0;i<n;i++)printf("%4d",a[i]);
getch();
}
TC
Enter array size 5
Enter 5 integers3 5 -4 9 2
Sorted elements 9 5 3 2 -4

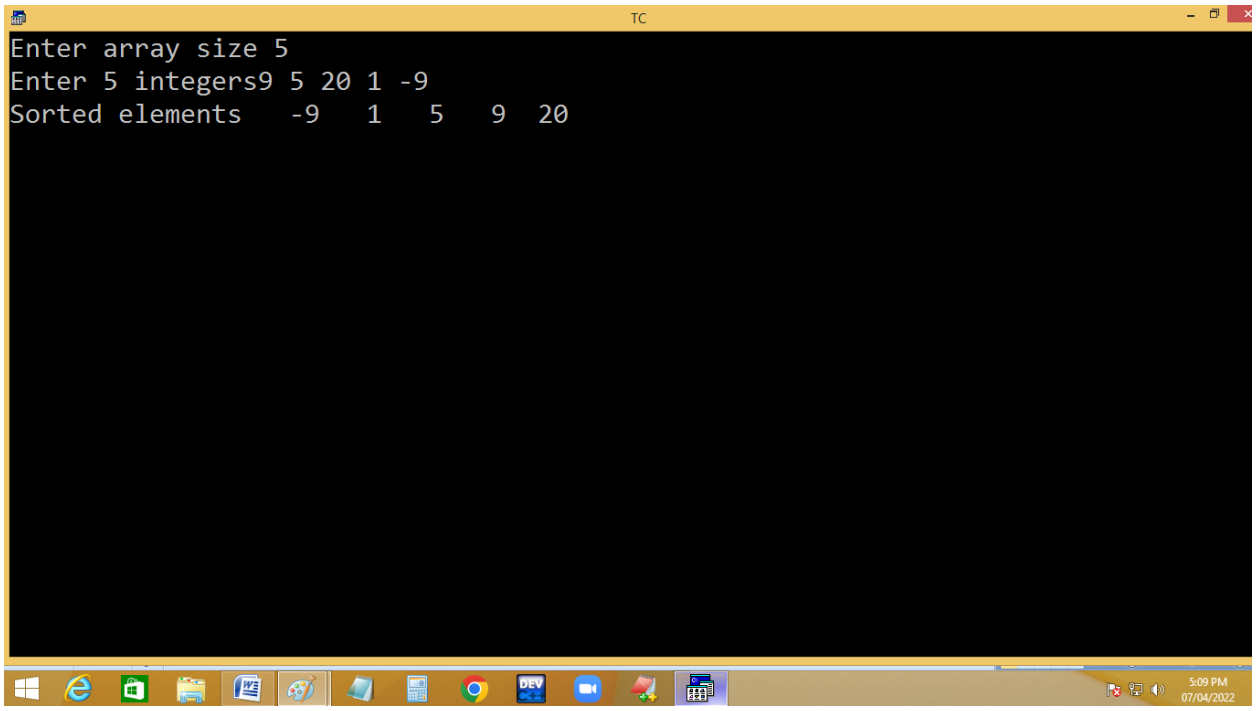
Bubble sort:

```
9  5  20 1  -9
5  9  20 1  -9
5  9  1  20 -9
5  9  1  -9 20
5  1  9  -9 20
5  1  -9 9  20
1  5  -9 9  20
1  -9 5  9  20
-9  1 5  9  20
```



i	j	j+1
0	0-1,1-2,2-3,3-4	
1	0-1,1-2,2-3	
2	0-1,1-2,	
3	0-1	






```
for(i=0;i<=n-2;i++)
{
for( j=0; j<=n-i-2; j++)
{
if(a[ j]>a[ j+1] )
{
temp=a[ j];a[ j]=a[ j+1];a[ j+1]=temp;
}
}
}
```

i	j	j+1	n-i-2
0	0	1,1-2,2-3,3-4	5-0-2=3
1	0	1,1-2,2-3	5-1-2=2
2	0	1,1-2,	5-2-2=1
3	0	1	5-3-2=0

Two dimensional arrays:

Array with several rows and columns.

Array with two subscripting operators [ ] [ ].

It is array of arrays. i.e. collection of one-dimensional arrays.

It is implicit double pointer.

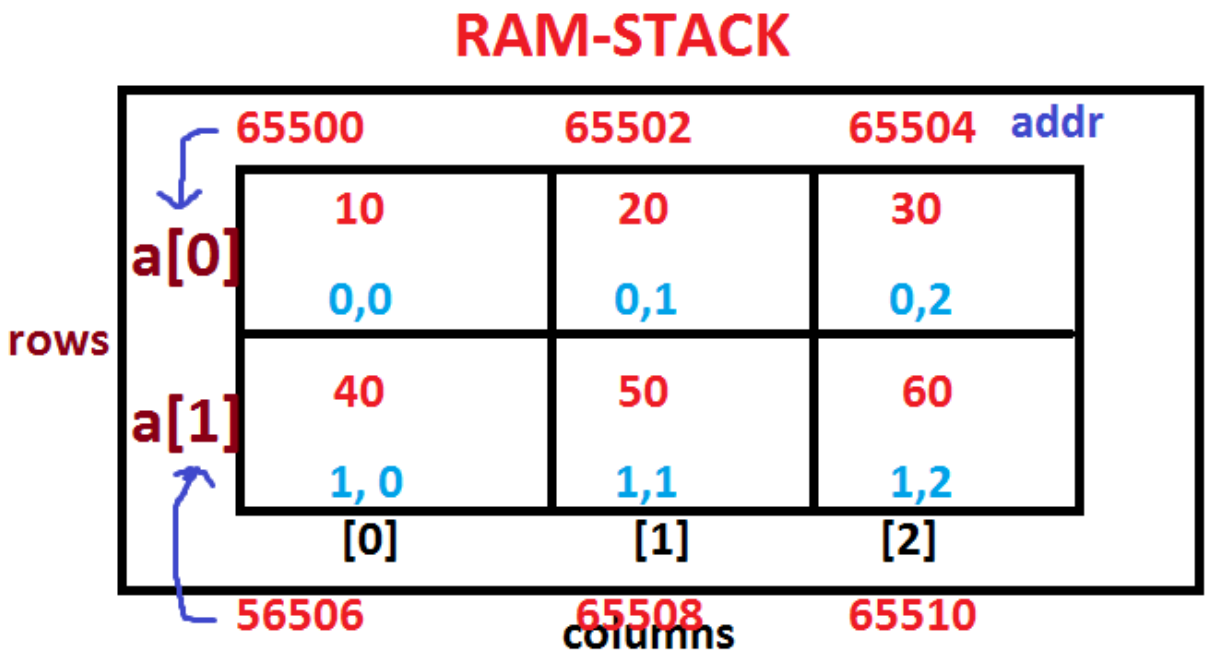
It is a n*n matrix.

Syntax:

datatype variable [rows] [columns] = {elements} ;

Eg:

int a[2][3] = { { 10, 20 , 30 } , { 40 ,50, 60 } };



In two dimensional array the rows/first subscript is working as array of pointers and they stores first column address of each row. Hence it is an implicit/internal double pointer.

In the above example, To print the first row, first column value, we have to use

printf("%d", a[0] [0]); ➔ 10

Internally how this statement is working ?

a[0] means value at a[0] i.e. 65500.

65500 + [0] col ➔ 65500 + 0 * 2 ➔ 65500 ➔ value at 65500 is 10.

Index no

Int size

row col

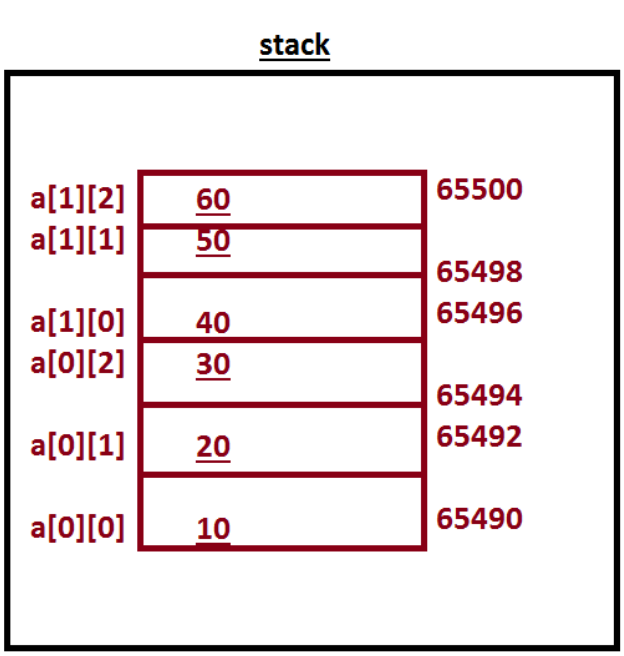
p(a[0] [0]);

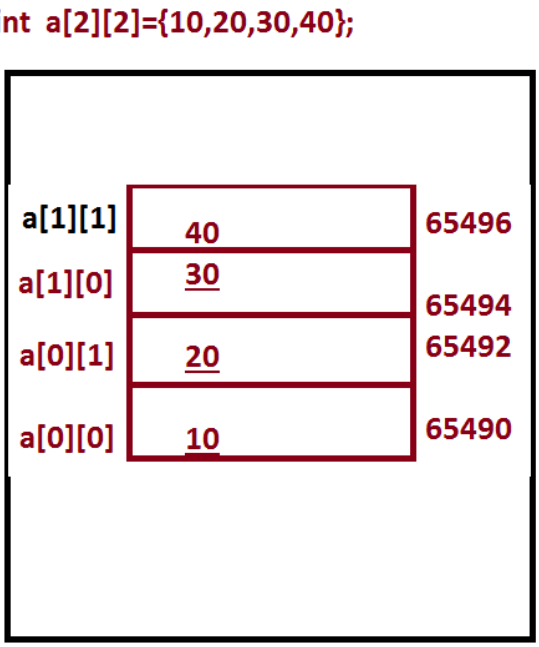
65500 + 0*2 = 65500==>print value at 65500=10

p[1][2]=60;

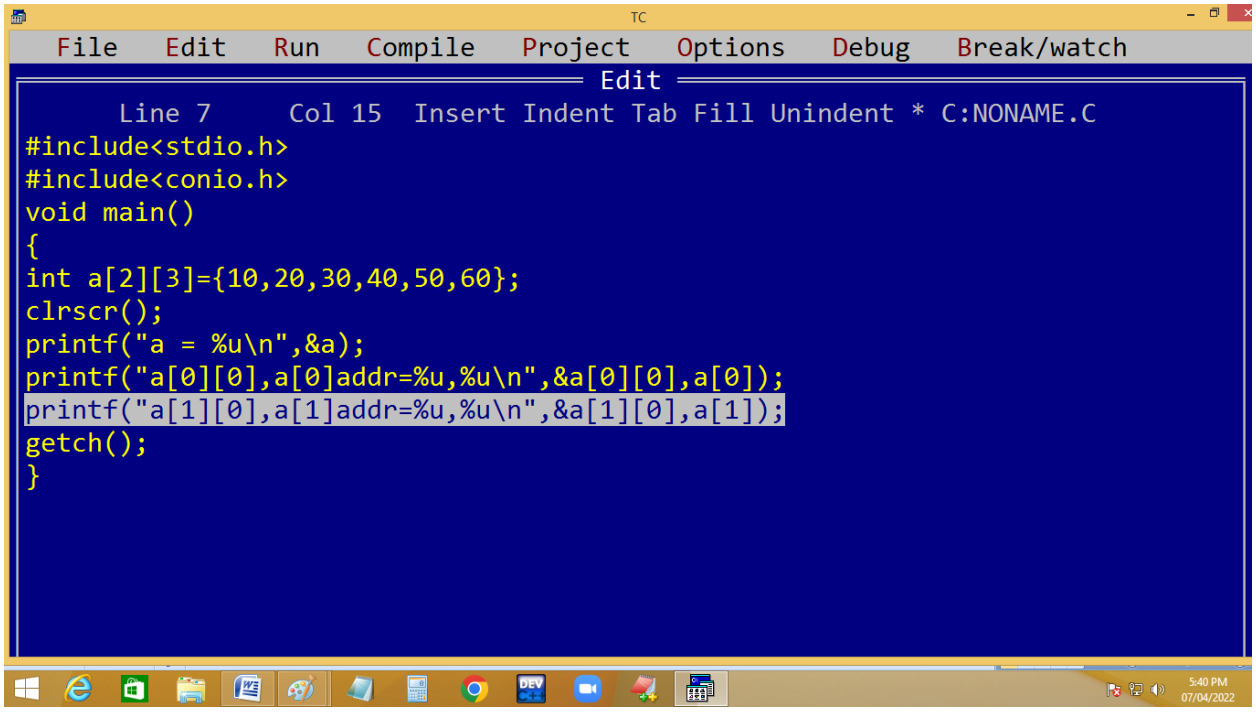
65506+2*2=65510==>60

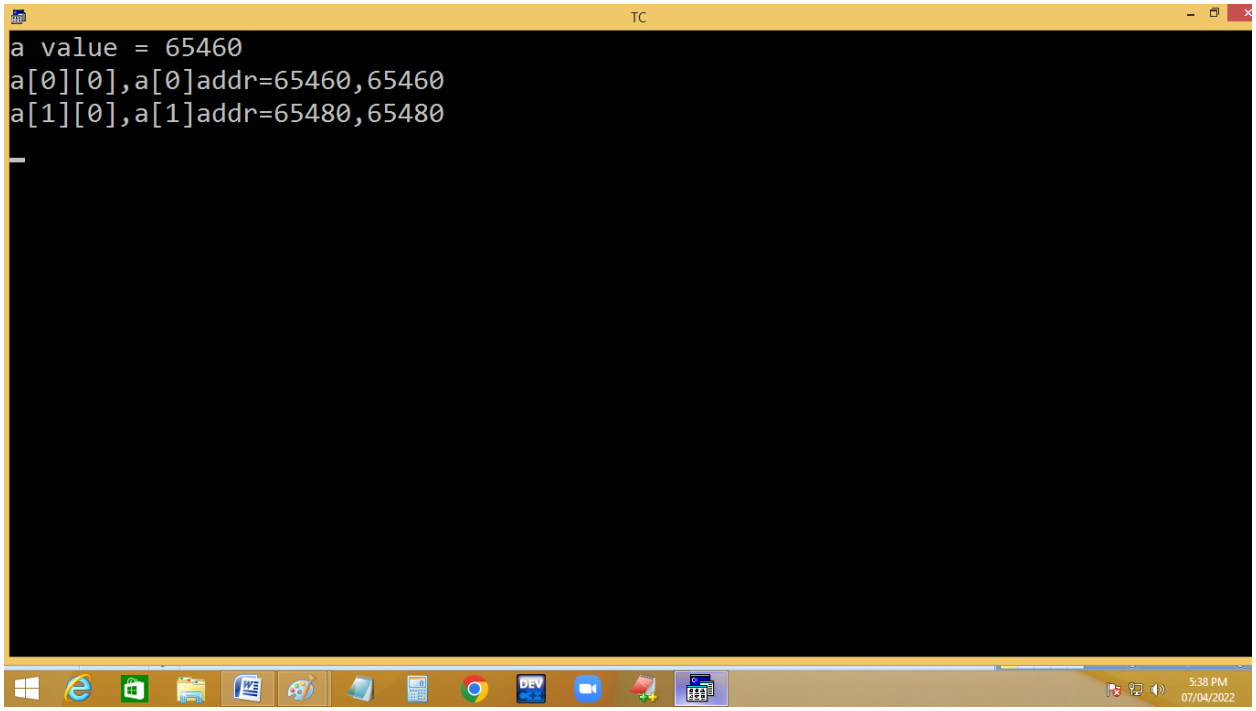
offset



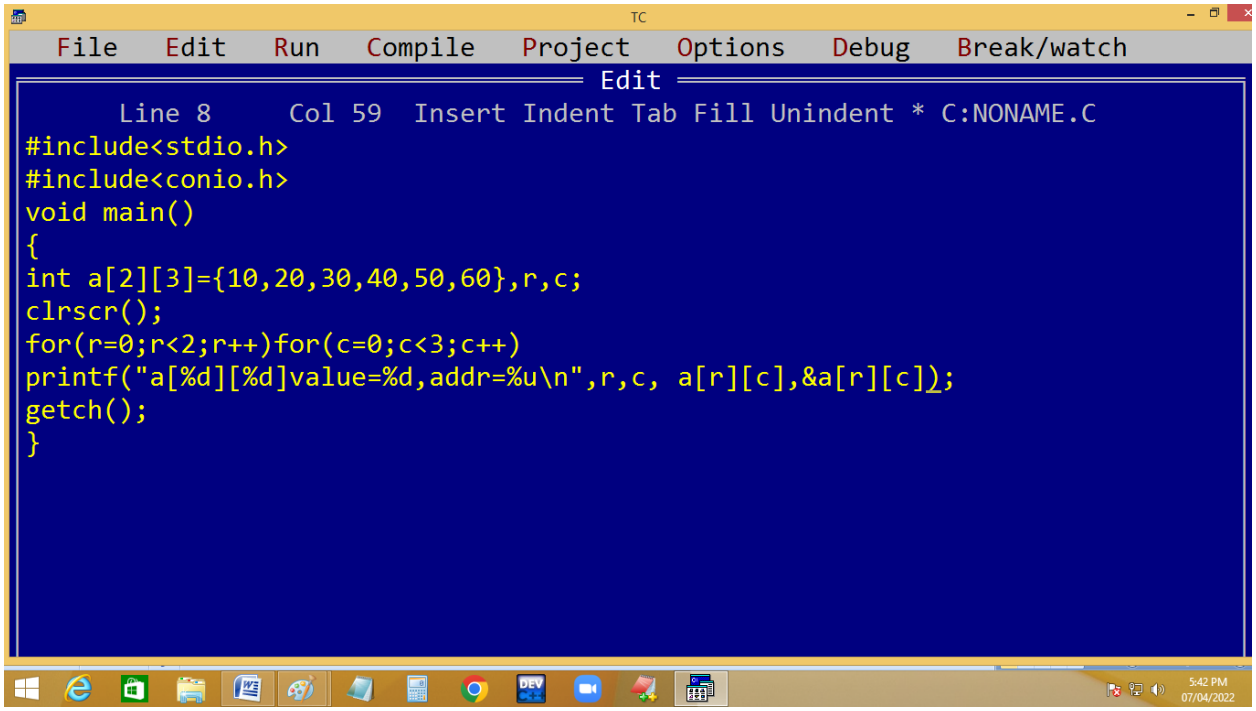


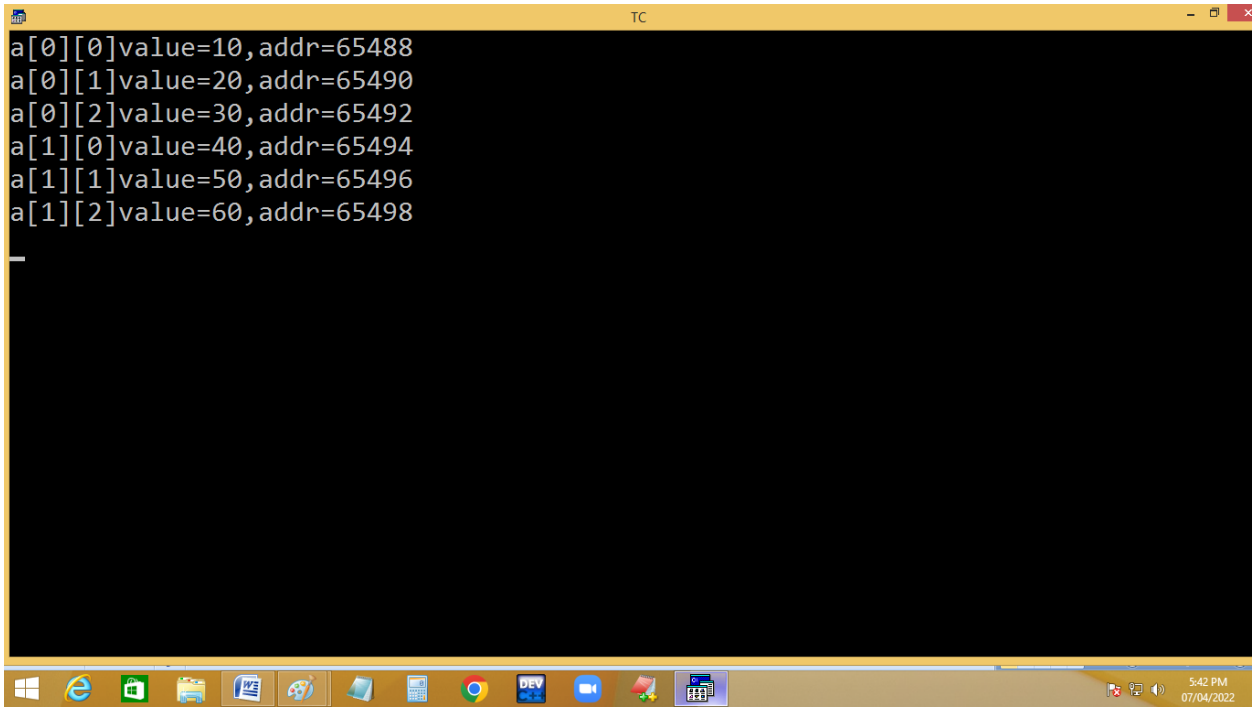
Eg. Finding address of two dimensional array.



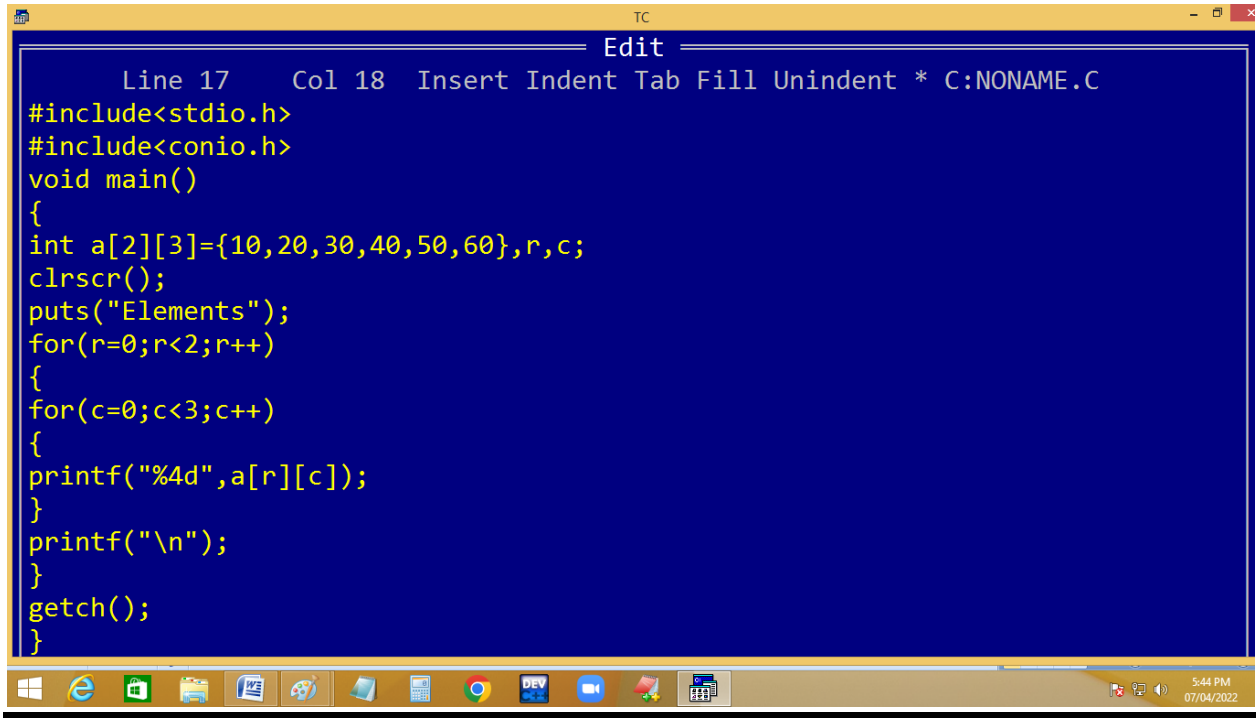


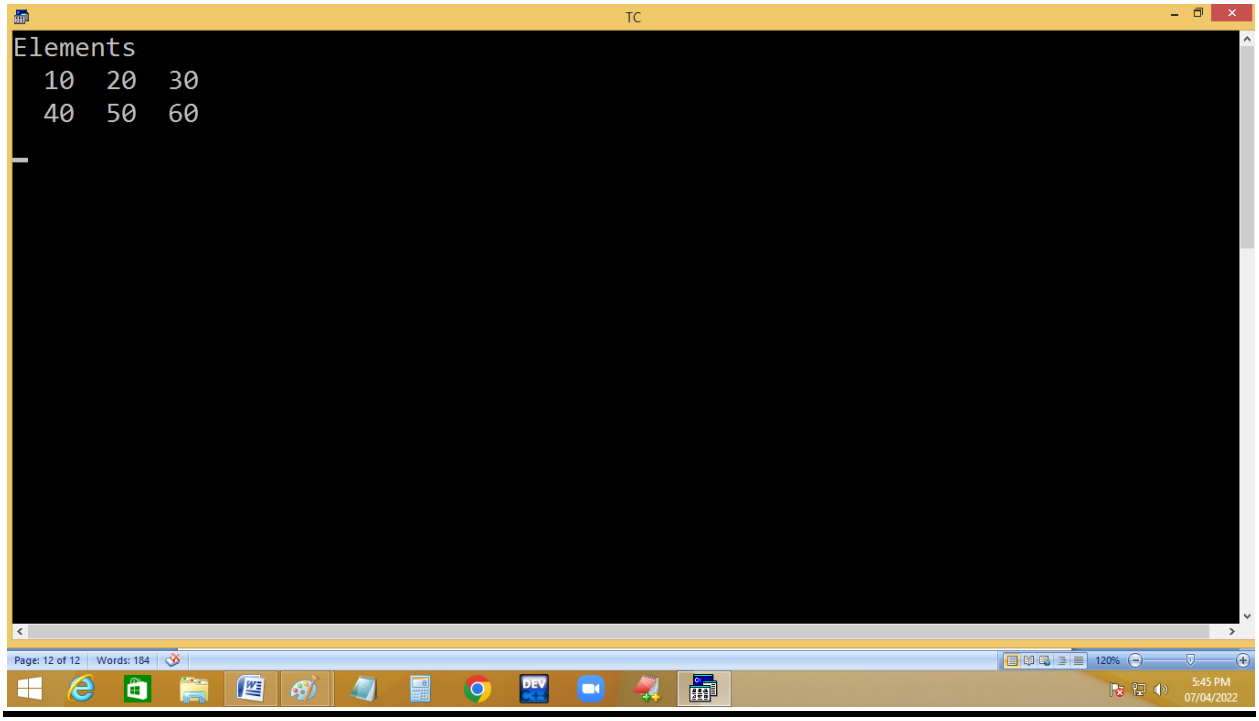
Finding array elements address:





Eg: Direct initialization of a 2 * 3 matrix:


TC
Edit
Line 17 Col 18 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a[2][3]={10,20,30,40,50,60},r,c;
clrscr();
puts("Elements");
for(r=0;r<2;r++)
{
for(c=0;c<3;c++)
{
printf("%4d",a[r][c]);
}
printf("\n");
}
getch();
}


TC
Elements
10 20 30
40 50 60