ARRAYS

It is collection of homogeneous [same type] variables.

Array is nothing but collection of contiguous memory locations, where we can store and manage more than one value of same type under one name.

It is a derived data type.

It is an implicit / internal pointer.

It is implicit const pointer

It is one of the data structure.

Advantages:

Generally to store several values of same type, we have to declare several variables. Here we have to remember all these variable names also. When the program is too big, it is very difficult to remember all the variable names. In this situation, the only solution is array.

Array reduce program length.

Array minimize the errors.

In functions to carry several values of same type at a time, we are using arrays.

Disadvantage: Array size is static. i.e. array size should be entered at design time and it is fixed at compile time. Due to this we can't change this array size at runtime, which causes memory wastage some times.

In C language we are using

1. One dimensional arrays
2. Multi dimensional arrays

One dimensional arrays:

- An array with one row and several columns.
- An array with single subscripting operator [ ] is called one dimensional array.
- It is an implicit single pointer.

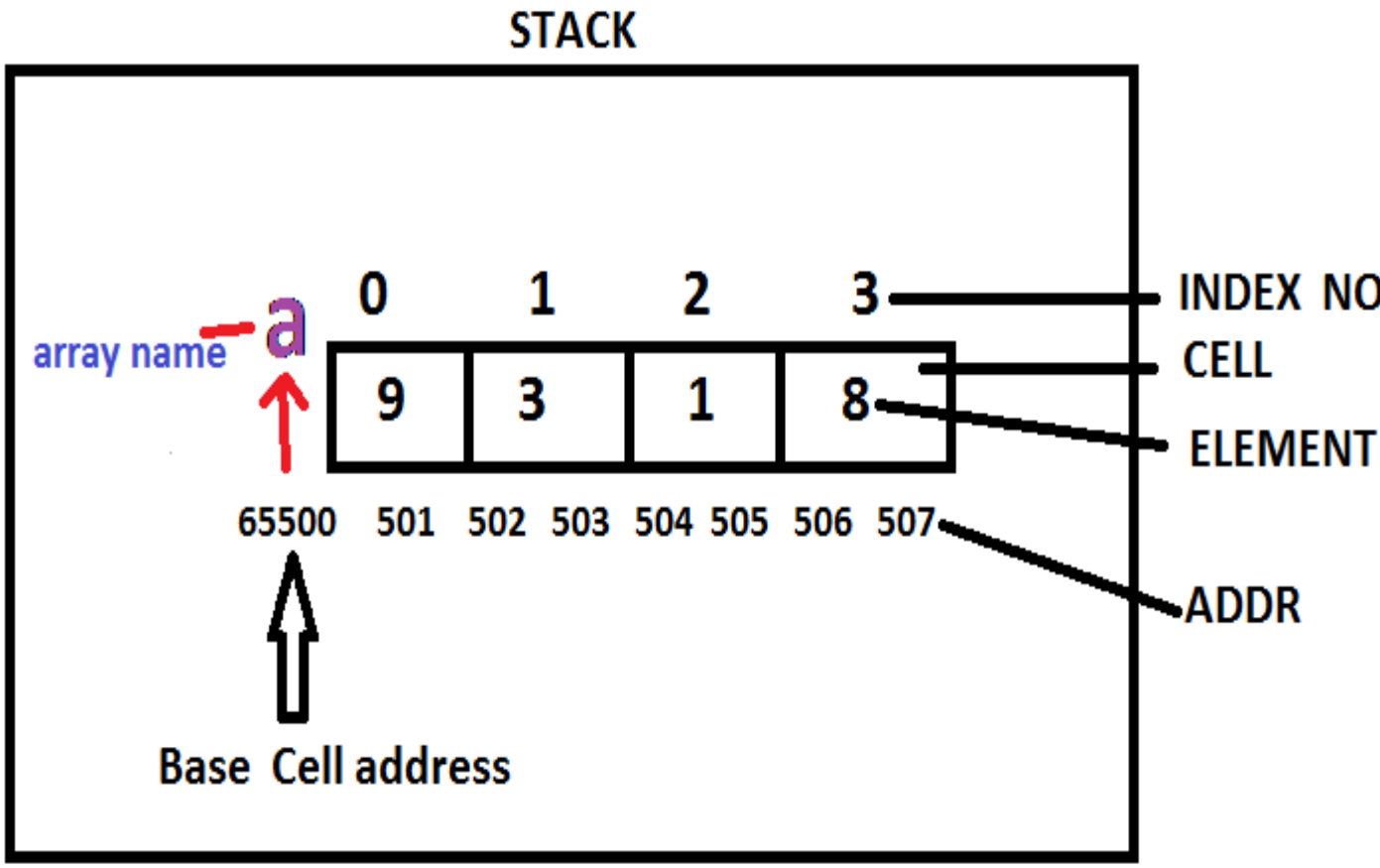
Syntax:

```
datatype  variable[size] = {elements};
```

Eg:

```
int  a[4] = { 9, 3, 1, 8 };
```

Memory allocation for array:



Array is implicit pointer because of array variable stores base cell [0 cell 1st byte] address. Hence array variable value and 0 cell address both are same.

Array declaration methods:

int a[3]; Ok

int a[ ]; No

int a[3]={1,2,3}; Ok

int a[ ]={1,2,3}; Ok

int a[0]={1,2,3}; Ok

int a[-5]; No

int a[5.5]; No

int n = 5, a[n]; No

int a[3]={10,20}; Ok

int a[3]={1, 2, 3, 4}; No

int a[0]; error

#define n 5 /* macro */

int a[n]; Ok

const int n=5, a[n]; No

int a[5>3]; ➔ int a[1]; Ok

int a[3<2]; ➔ int a[0]; No

int a[2+3]; ➔int a[5]; Ok

int a[5%3]; ➔ int a[2]; Ok

int a[5%5]; ➔ int a[0]; No

int a[1,2,3]; ➔ error

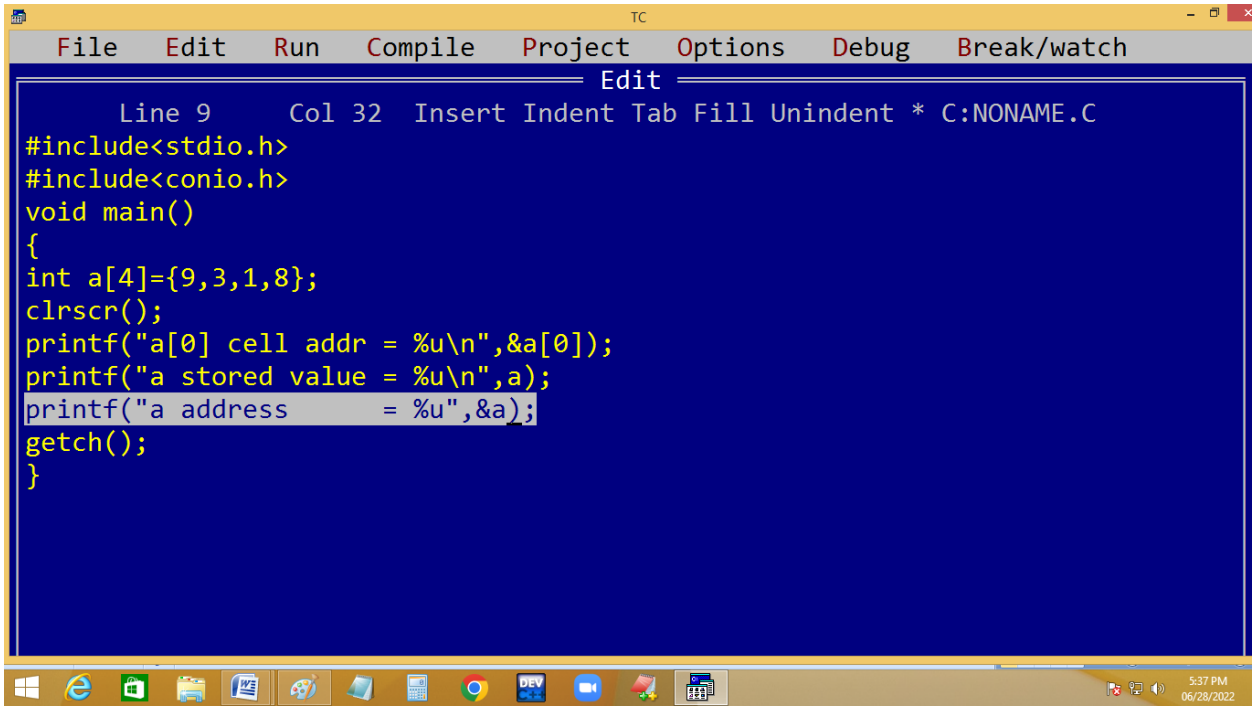
int a[40000]; ➔ 40000 * 2 = 80000 bytes ➔ No

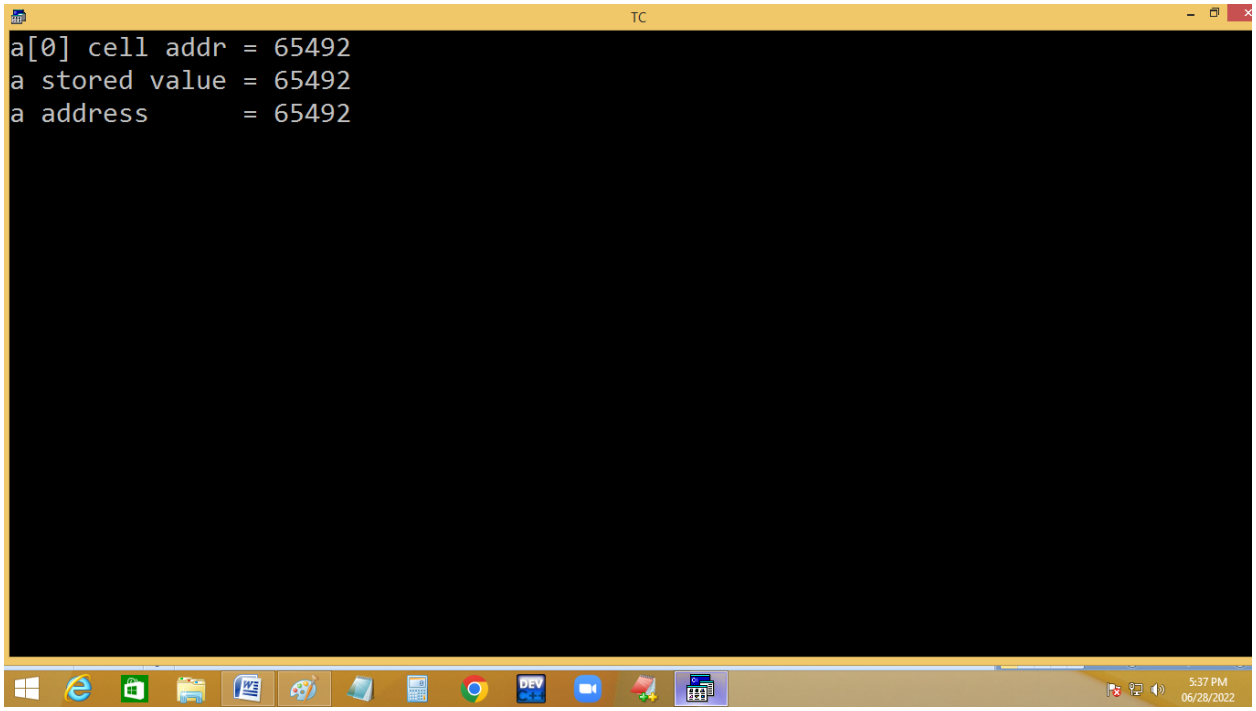
Note: Stack size is 65536 bytes(64kb) Only.

float a[10000]; Ok ➔ 10000 * 4 = 40000 bytes

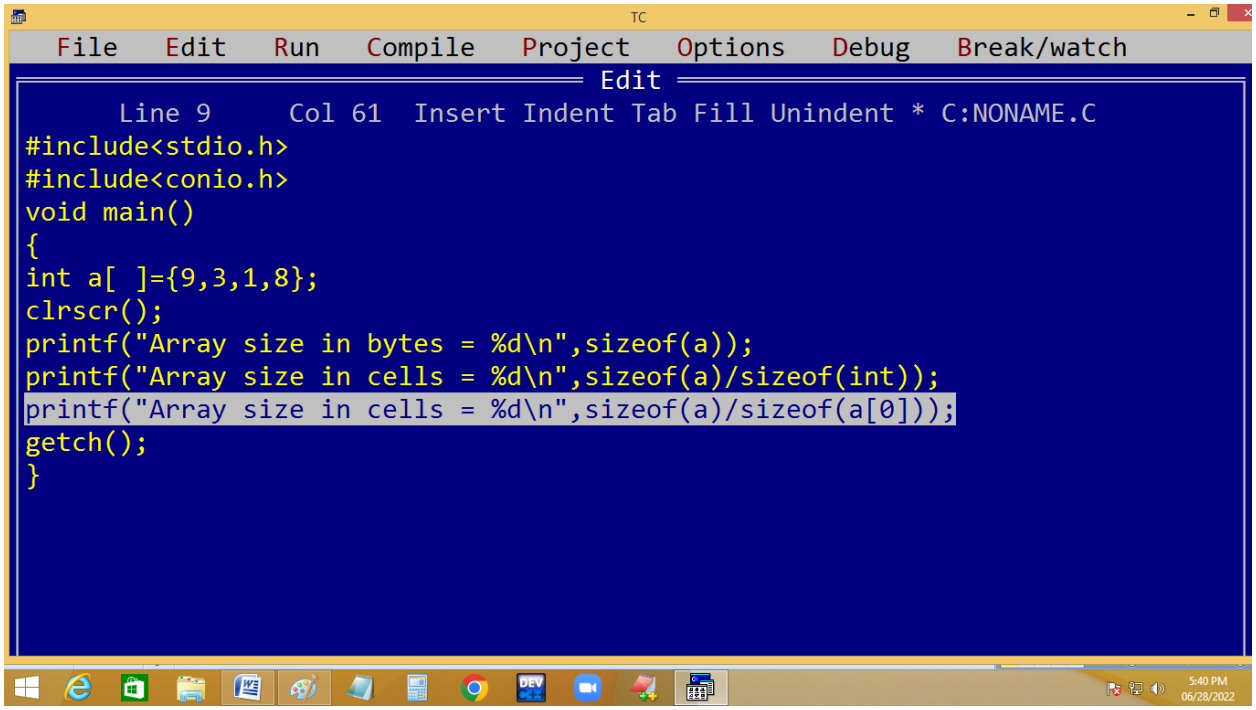
float a[20000]; No ➔ 20000 * 4 = 80000 bytes

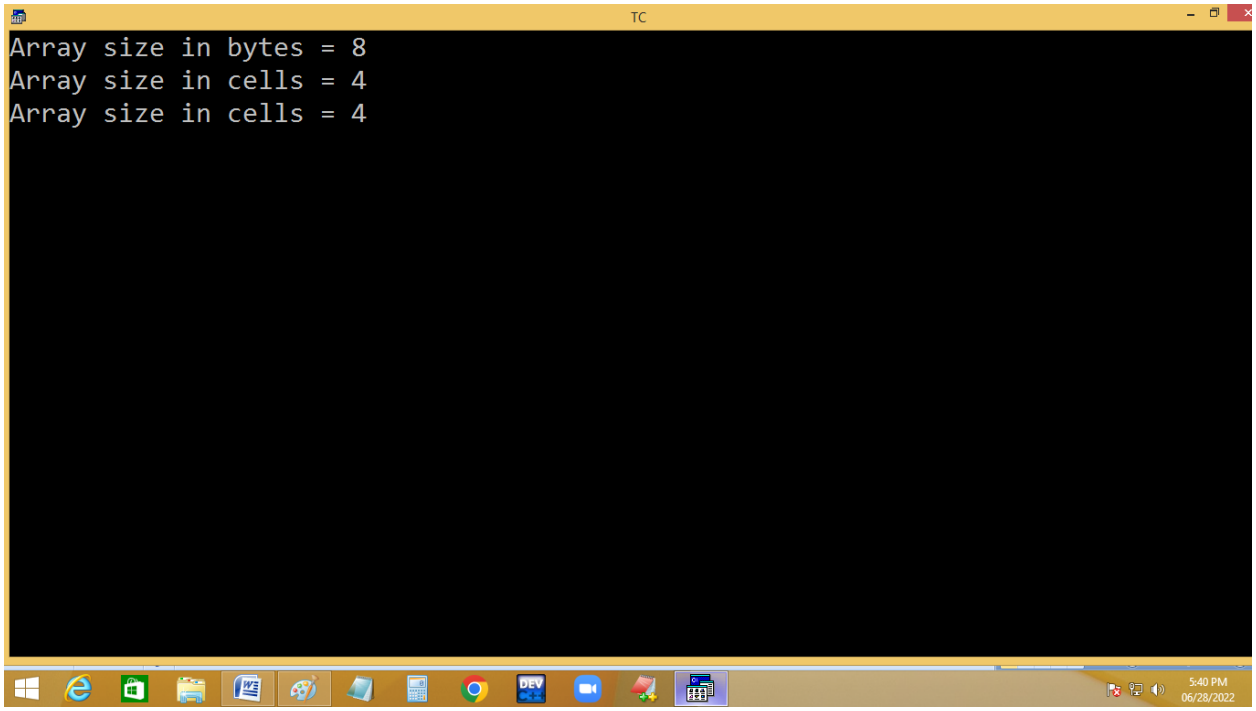
Eg. finding array address.



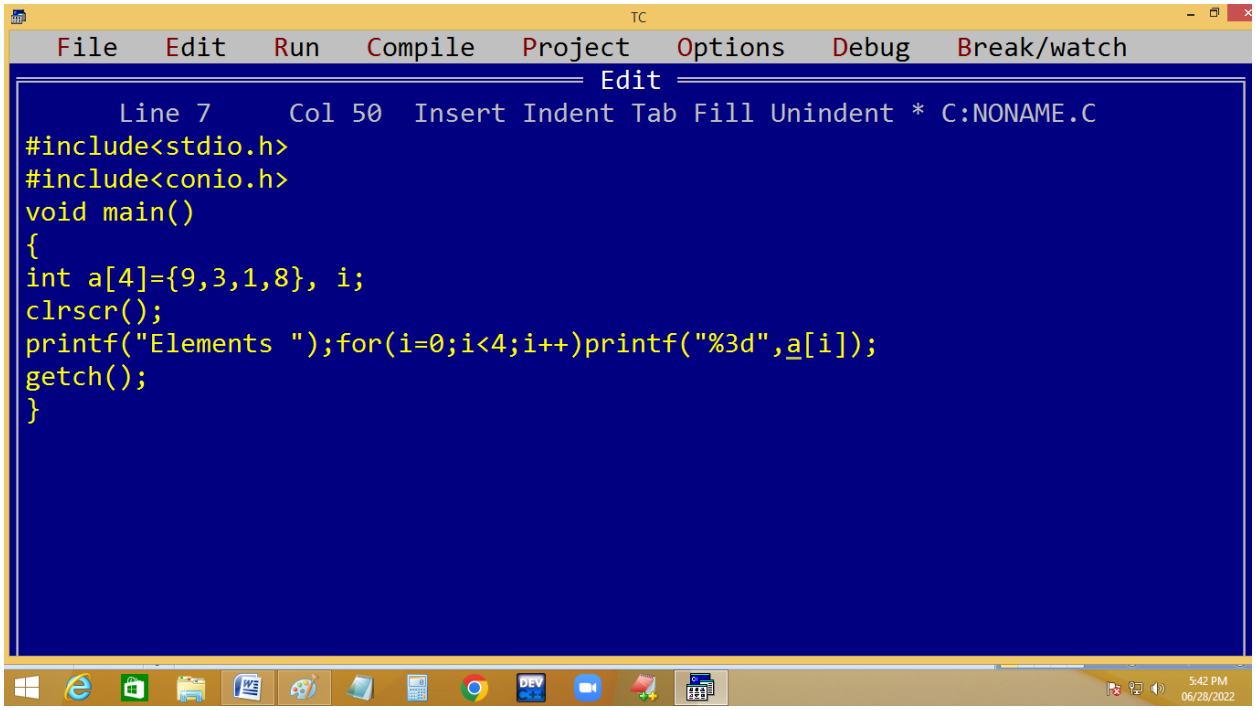


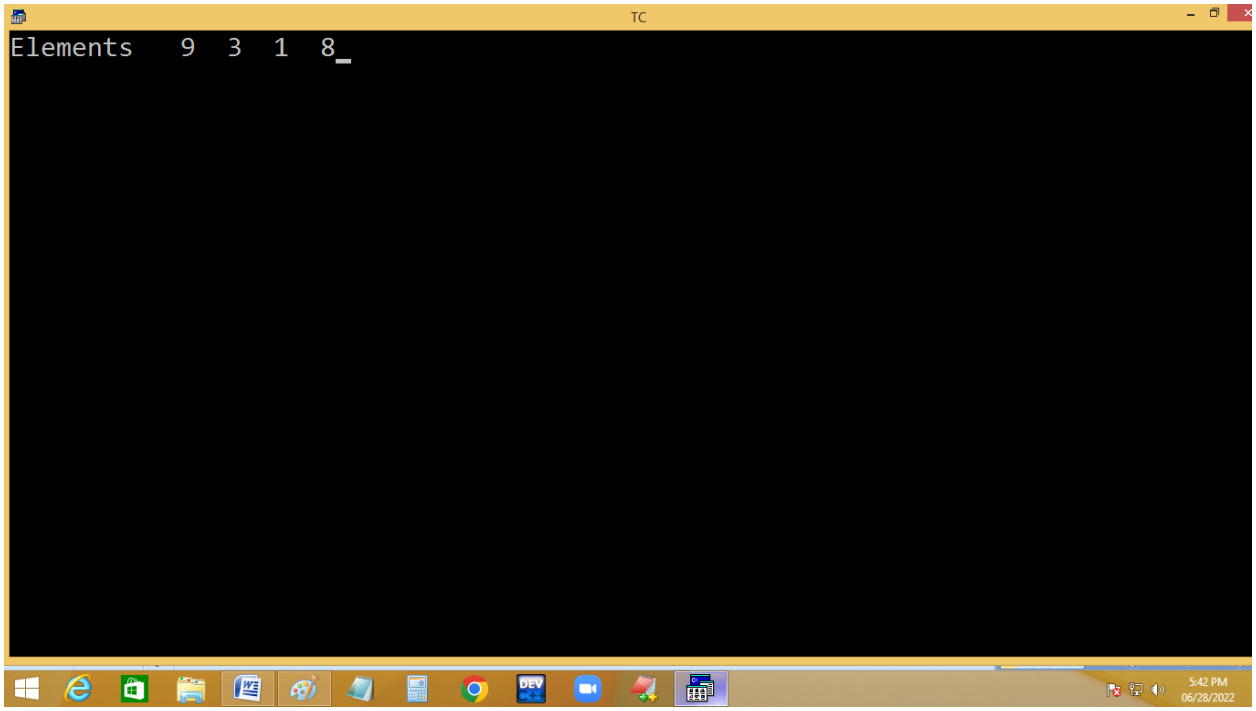
Eg. Finding array size:

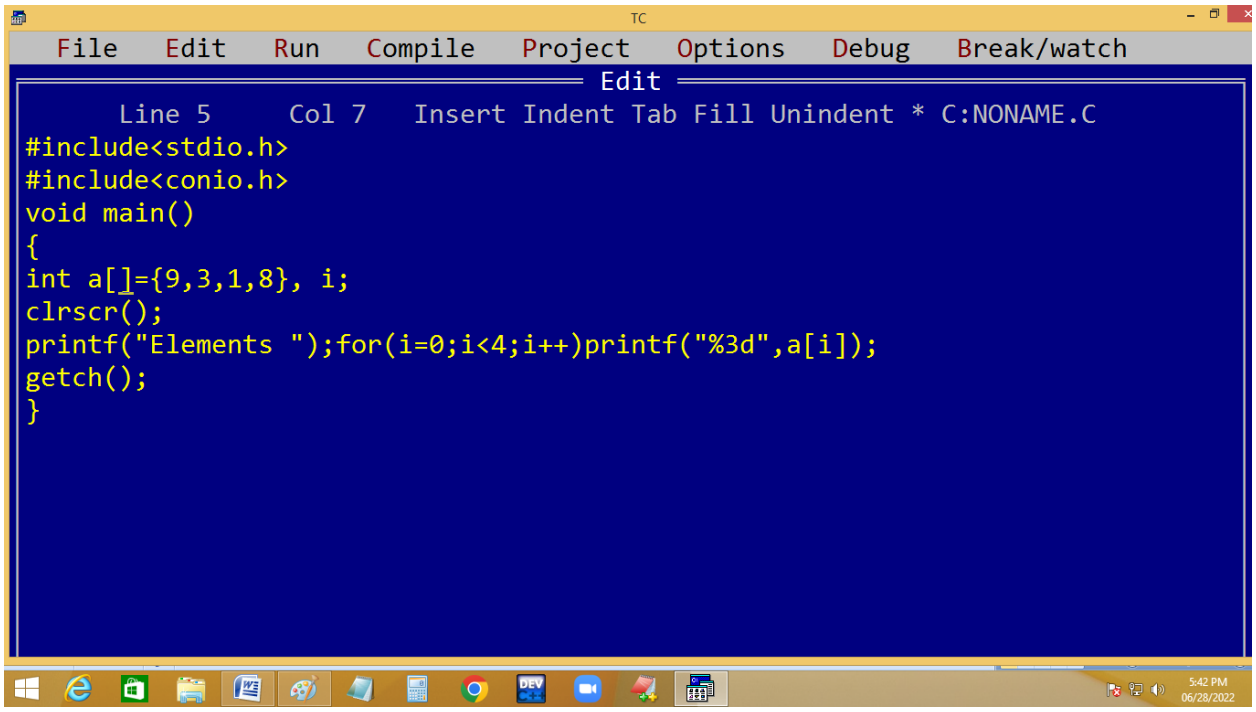


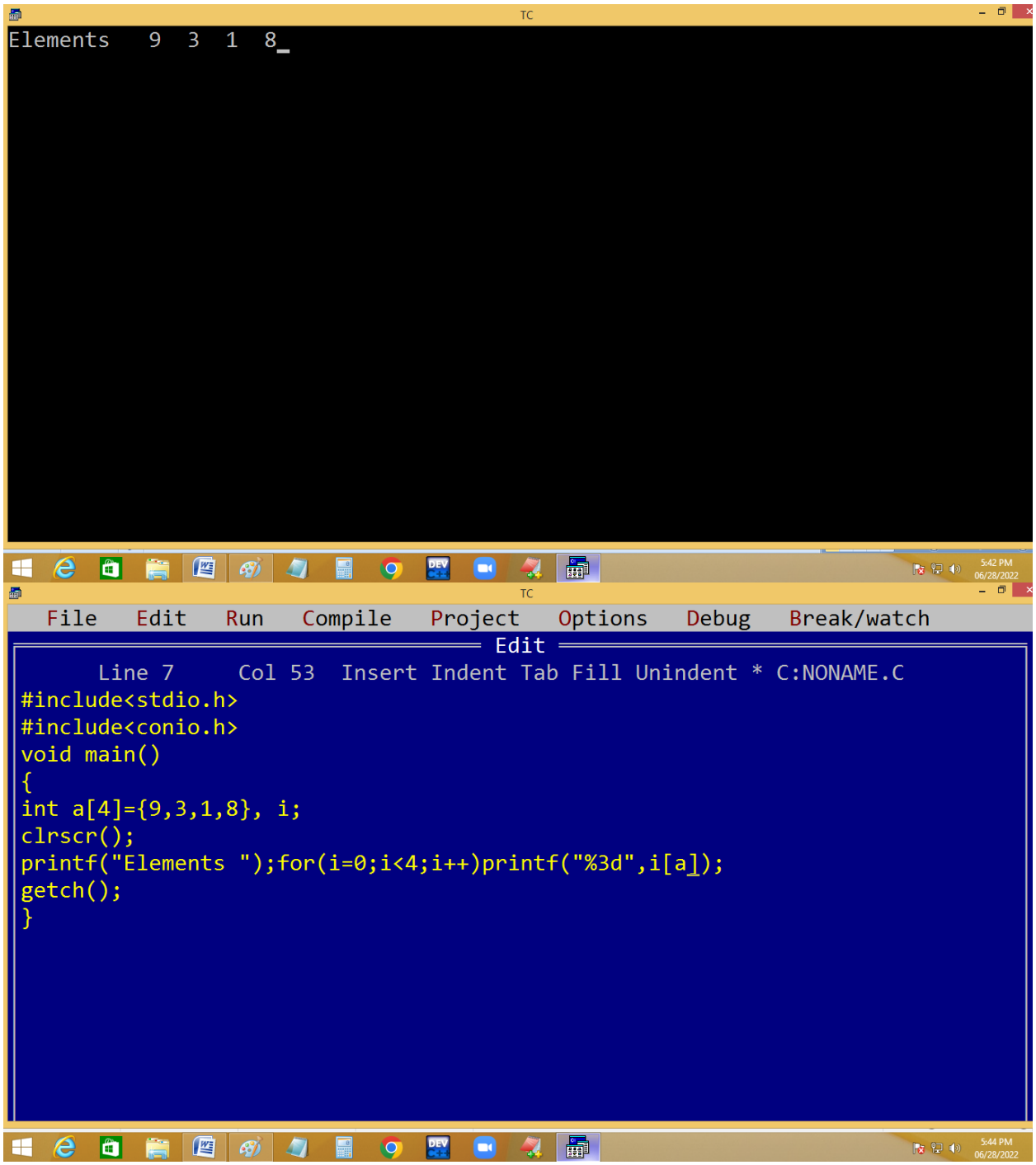


Direct initialization of array elements:

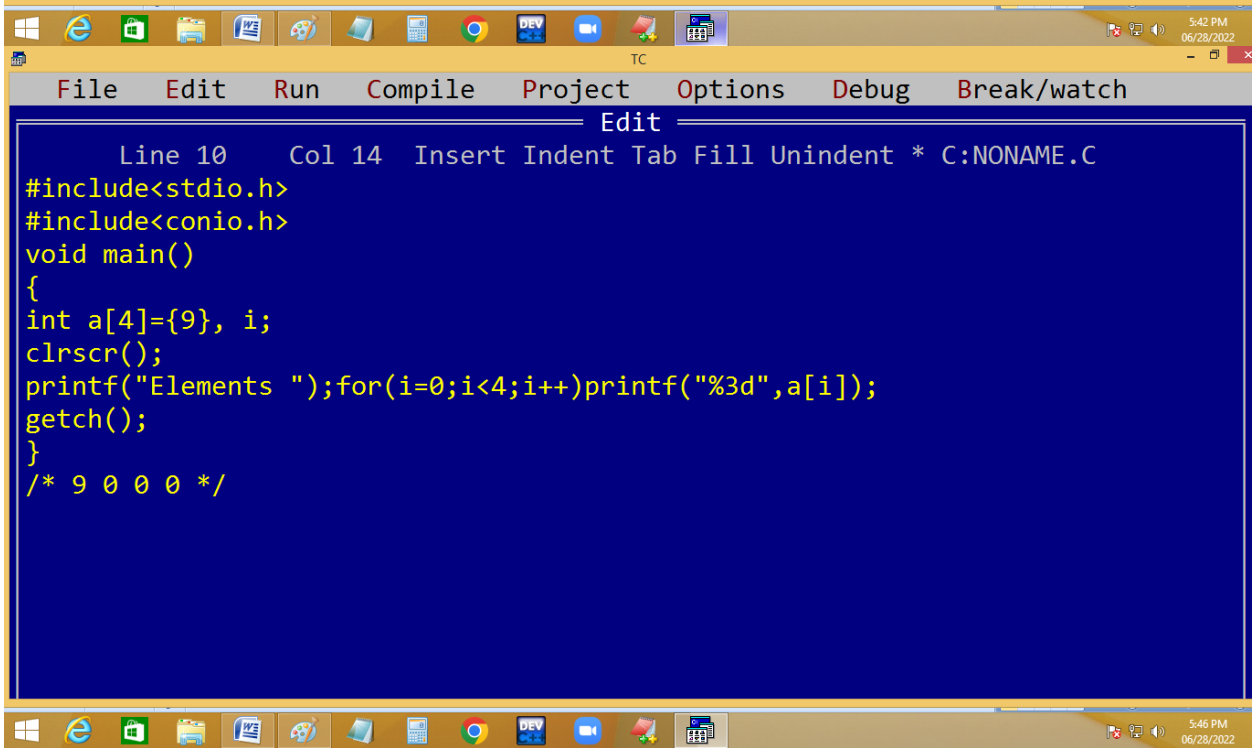


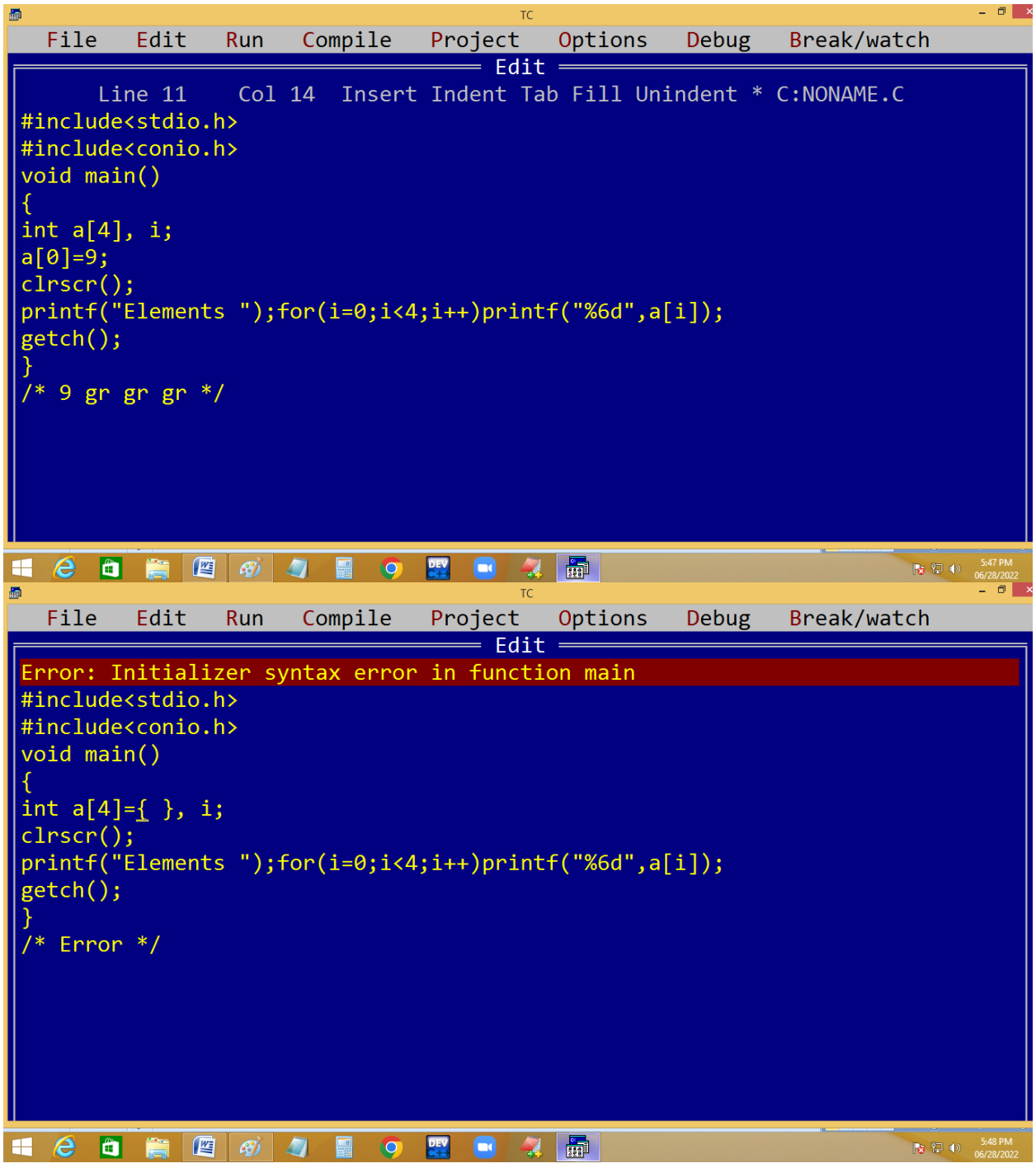



TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 5 Col 7 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a[]={9,3,1,8}, i;
clrscr();
printf("Elements ");for(i=0;i<4;i++)printf("%3d",a[i]);
getch();
}
5:42 PM
06/28/2022


TC
Elements 9 3 1 8
TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 7 Col 53 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a[4]={9,3,1,8}, i;
clrscr();
printf("Elements ");for(i=0;i<4;i++)printf("%3d",i[a]);
getch();
}


TC
Elements 9 3 1 8


TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 10 Col 14 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a[4]={9}, i;
clrscr();
printf("Elements ");for(i=0;i<4;i++)printf("%3d",a[i]);
getch();
}
/* 9 0 0 0 */


TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 11 Col 14 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a[4], i;
a[0]=9;
clrscr();
printf("Elements ");for(i=0;i<4;i++)printf("%6d",a[i]);
getch();
}
/* 9 gr gr gr */
5:47 PM
06/28/2022
TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Error: Initializer syntax error in function main
#include<stdio.h>
#include<conio.h>
void main()
{
int a[4]={ }, i;
clrscr();
printf("Elements ");for(i=0;i<4;i++)printf("%6d",a[i]);
getch();
}
/* Error */
5:48 PM
06/28/2022