Pointer to array:

Array is an implicit pointer.

Due to this it holds the base cell addr [0 cell addr] implicitly.

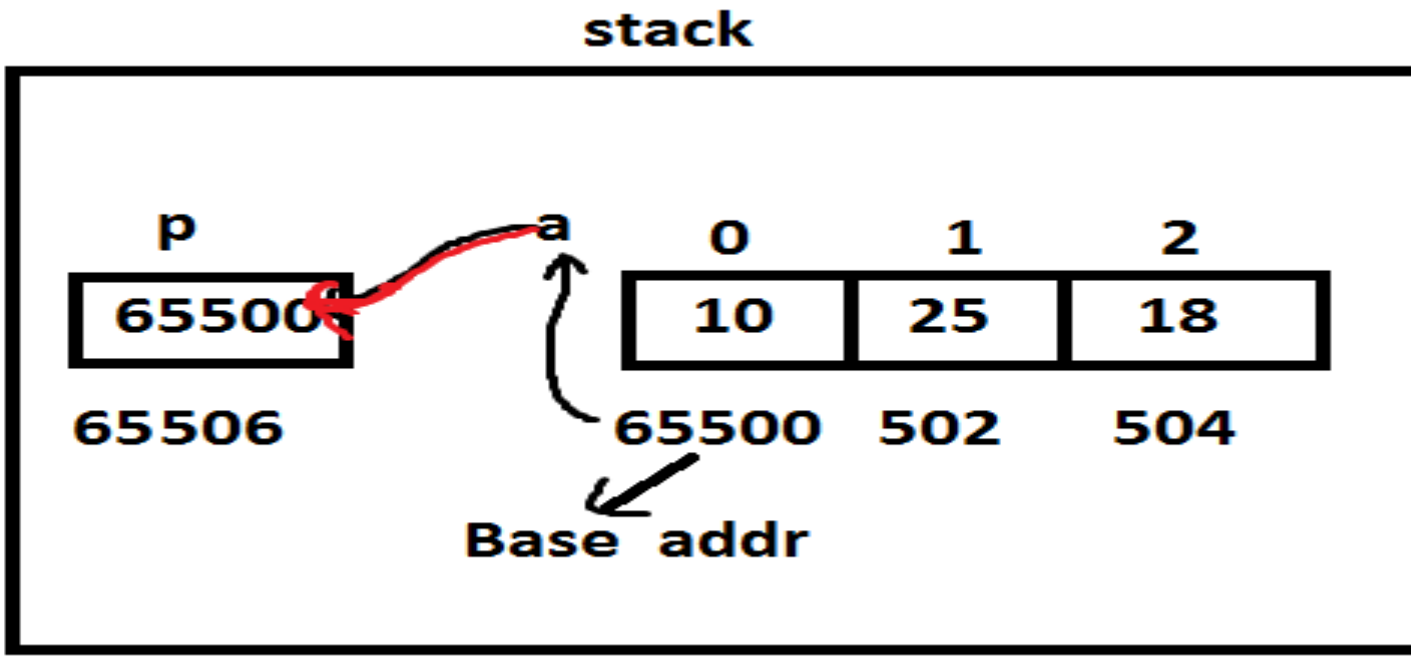
By assigning the array name or 0 cell addr to the pointer, we can handle array elements using the following syntax.

***(ptrvariable + offset/index * sizeof(variable));**

Eg:

int a[3]={10, 25, 18}, *p, i;

p = a ; or p = &a[0]; or p = &a;



for(i=0;i<3;i++)

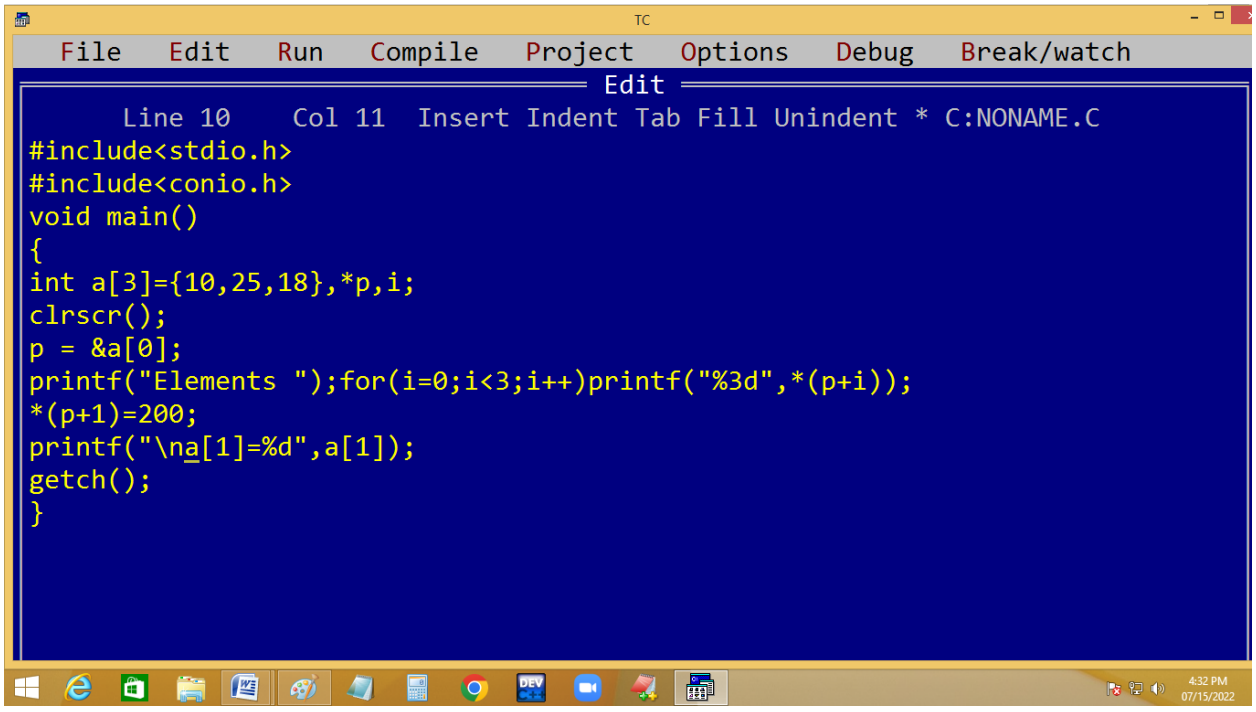
printf("%4d", *(p+i));

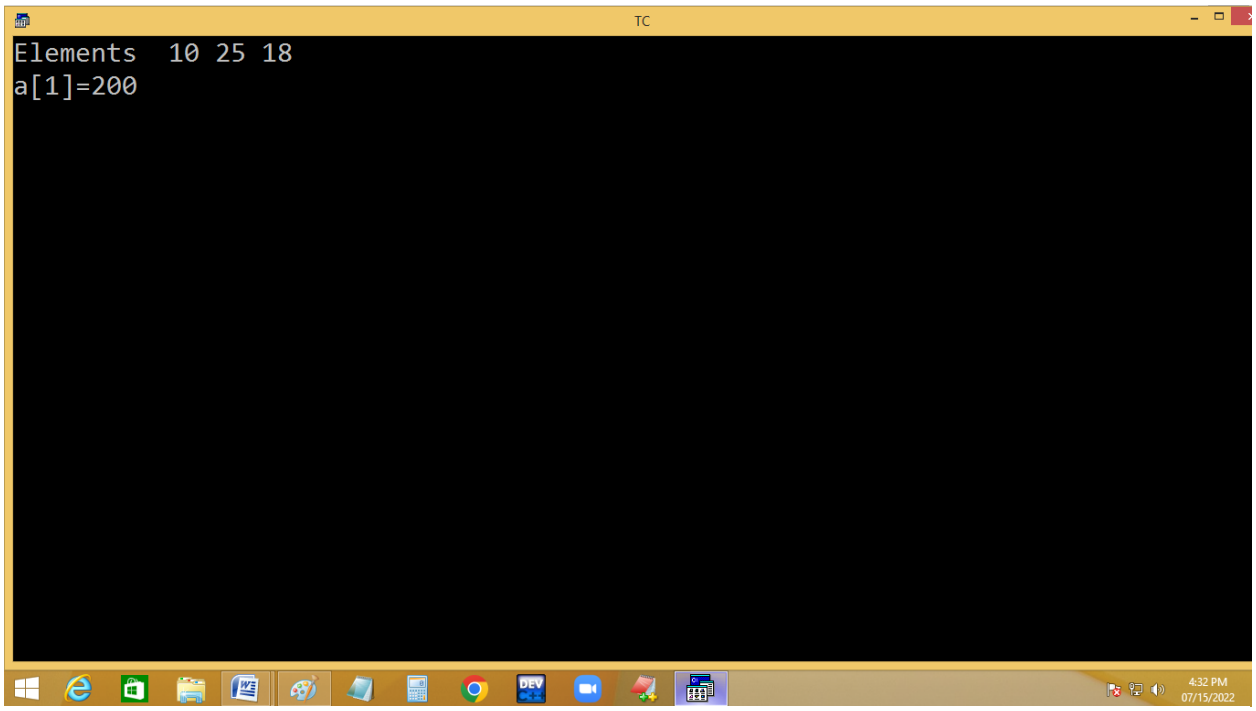
Here *(p+i) meaning is:

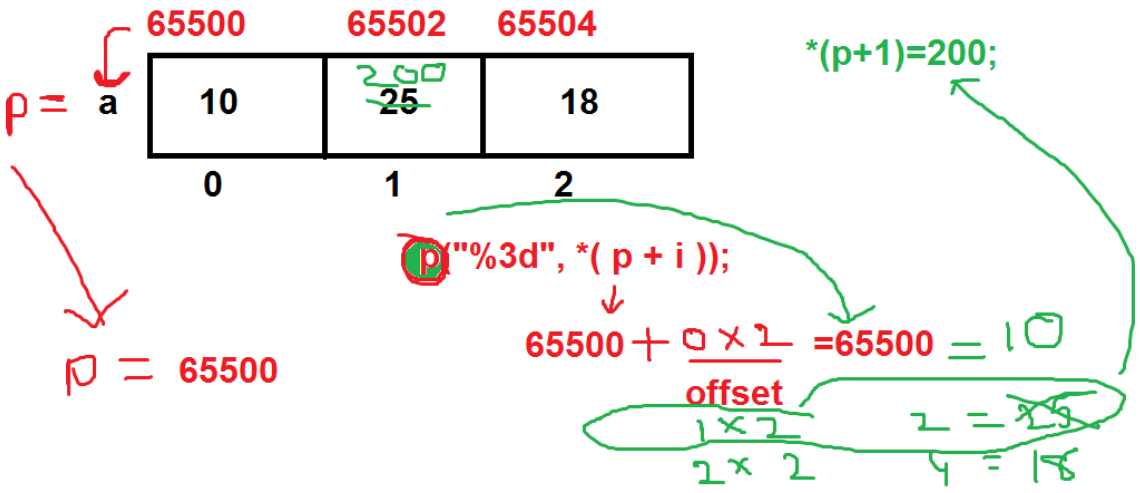
p is 65500

1. *(p+0*2)➔*65500➔value at 65500➔10
2. *(p+1*2)➔*65502➔value at 65502➔25
3. *(p+2*2)➔*65504➔value at 65504➔18

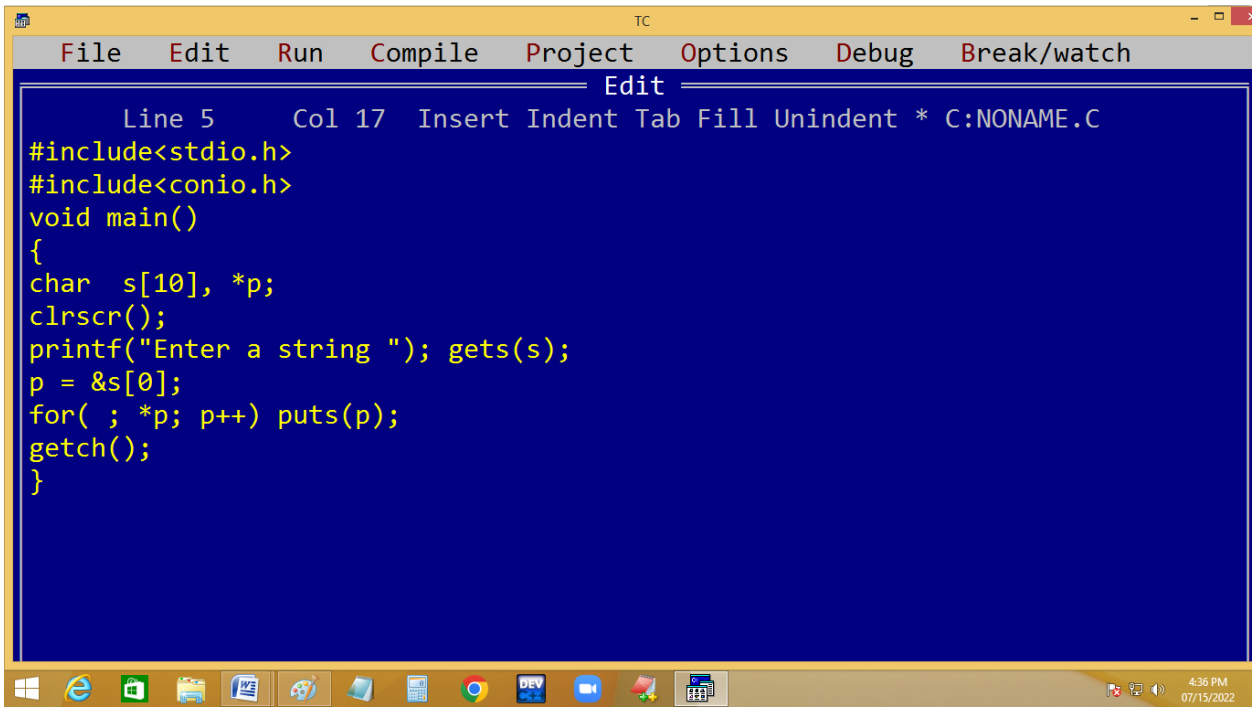
Note: Here 2 is int size.

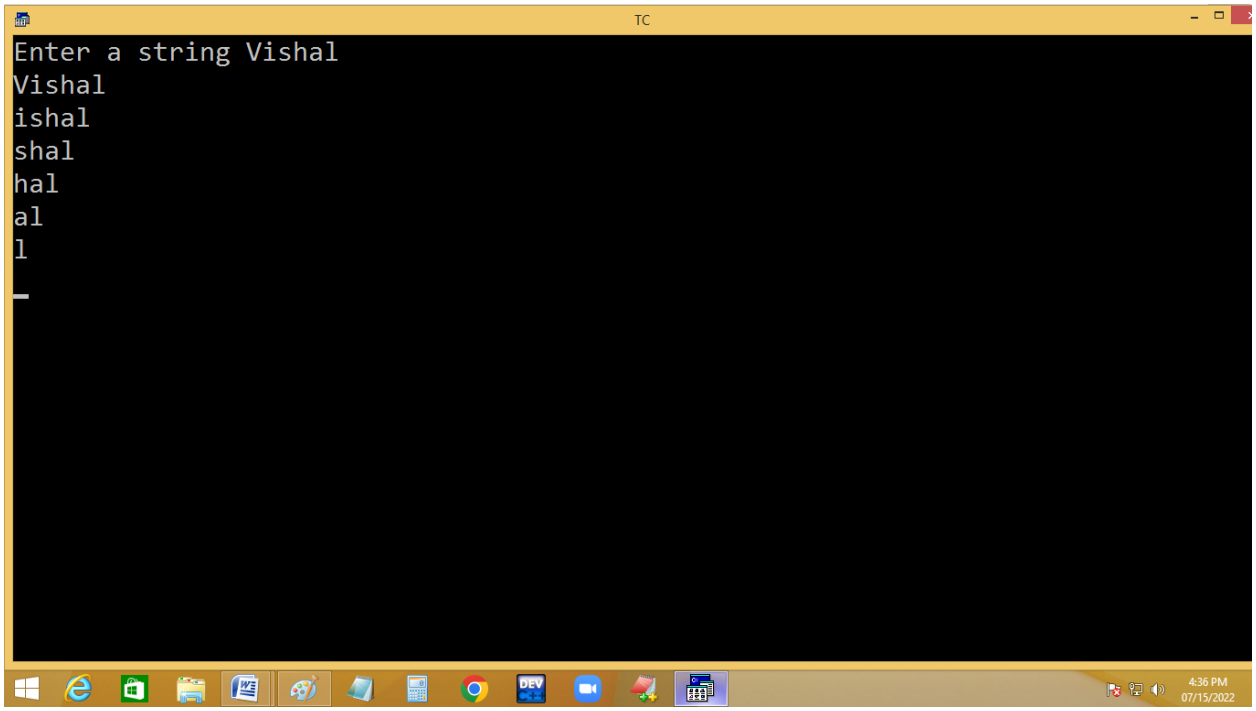


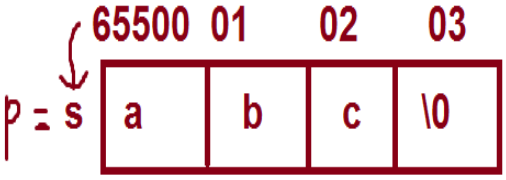




Pointer to string:


TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 5 Col 17 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
char s[10], *p;
clrscr();
printf("Enter a string "); gets(s);
p = &s[0];
for(; *p; p++) puts(p);
getch();
}


TC
Enter a string Vishal
Vishal
ishal
shal
hal
al
l

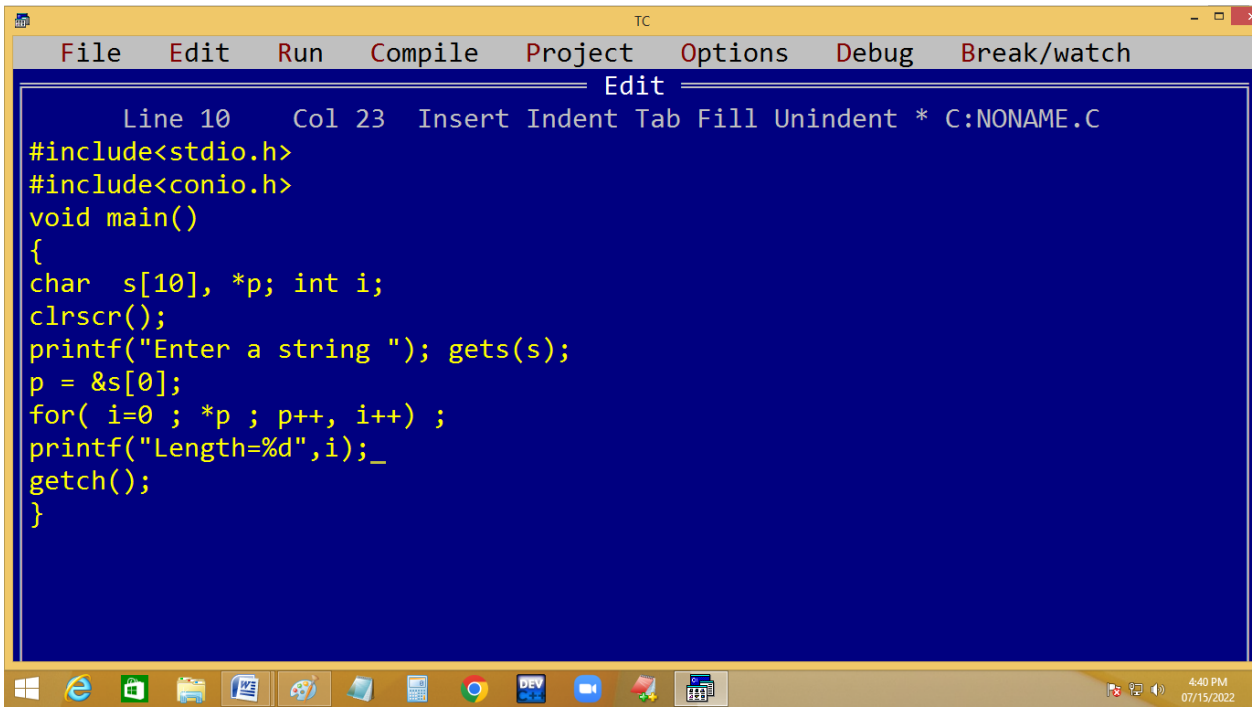


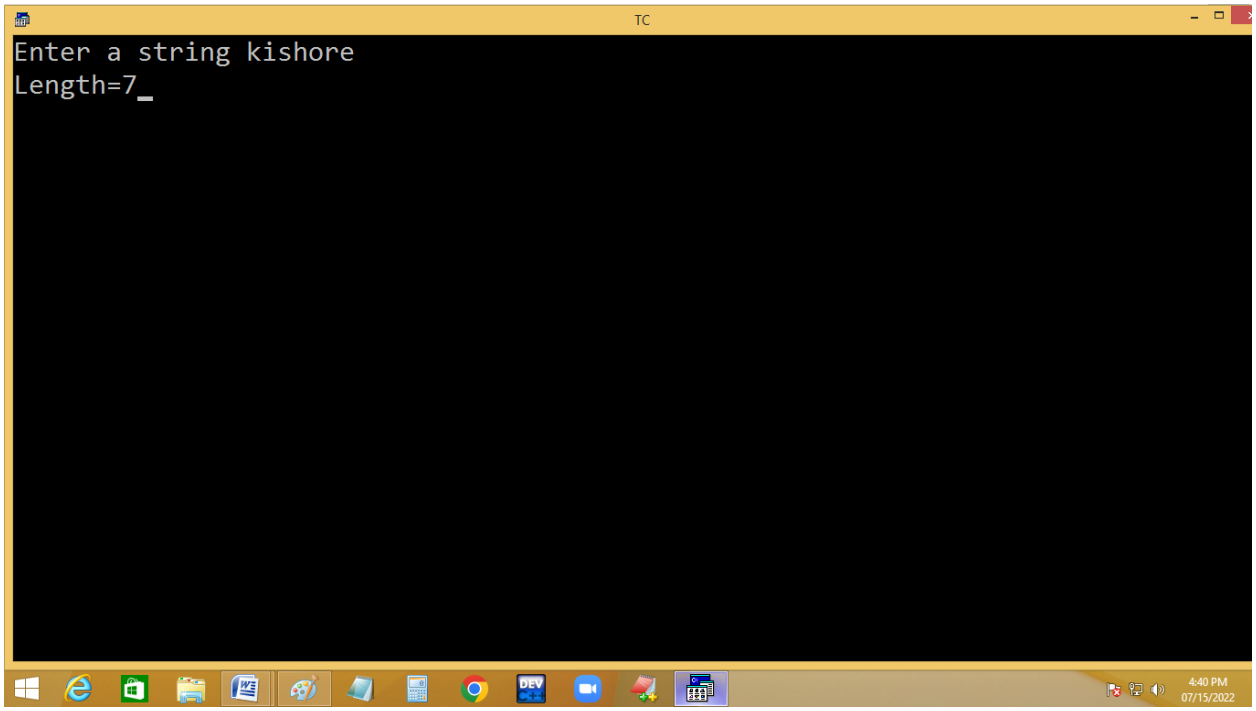
*p != '\0'

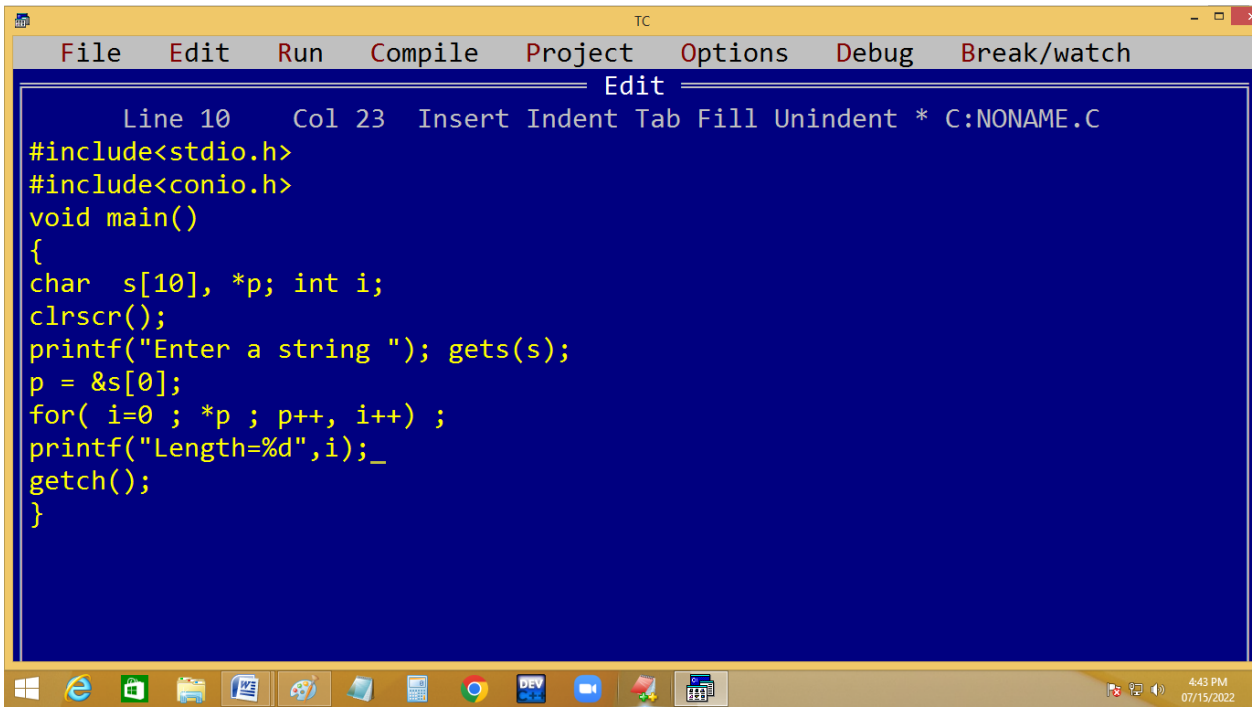
for (; *p ; p++) puts(p);

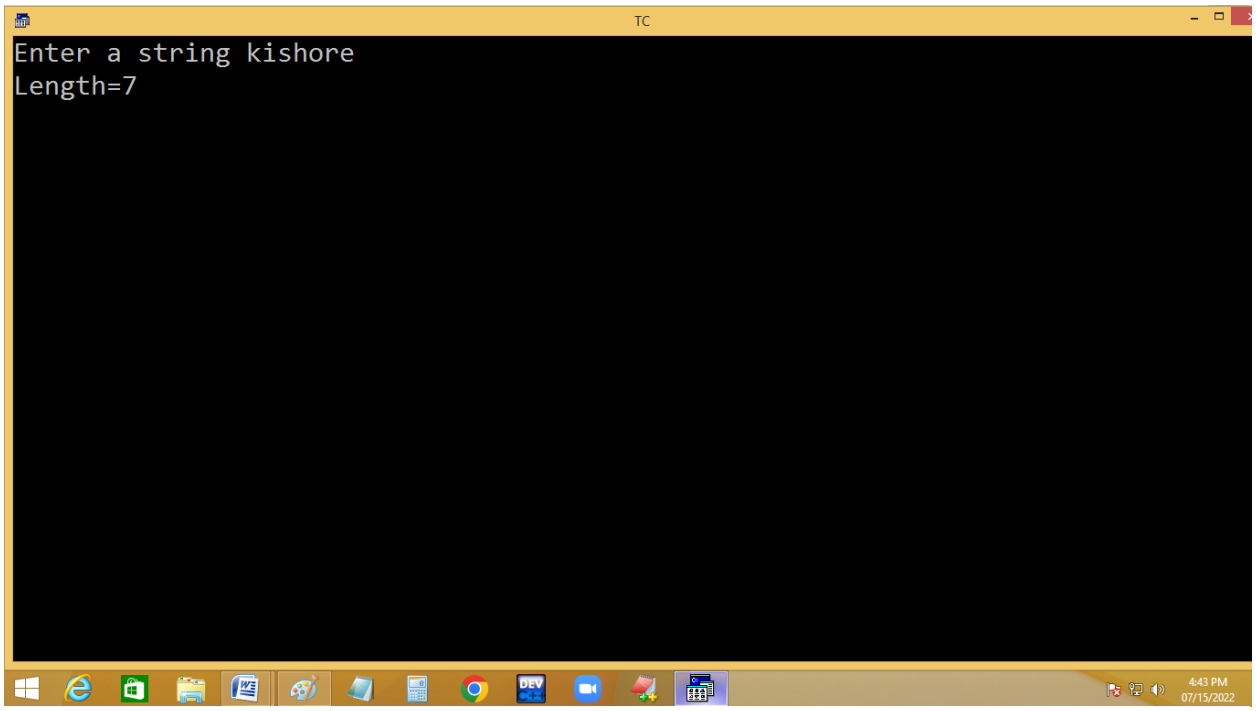
65500+0*1==>65500 to \0 ==>abc
65500+1*1==>65501 to \0 ==>bc
65500+2*1==>65502 to \0 ==>c

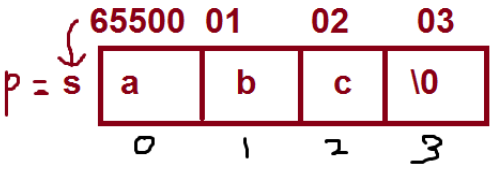
Eg. finding string length using pointer.


TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 10 Col 23 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
char s[10], *p; int i;
clrscr();
printf("Enter a string "); gets(s);
p = &s[0];
for(i=0 ; *p ; p++, i++) ;
printf("Length=%d",i);
getch();
}


TC
Enter a string kishore
Length=7


TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 10 Col 23 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
char s[10], *p; int i;
clrscr();
printf("Enter a string "); gets(s);
p = &s[0];
for(i=0 ; *p ; p++, i++) ;
printf("Length=%d",i);_
getch();
}
4:43 PM
07/15/2022





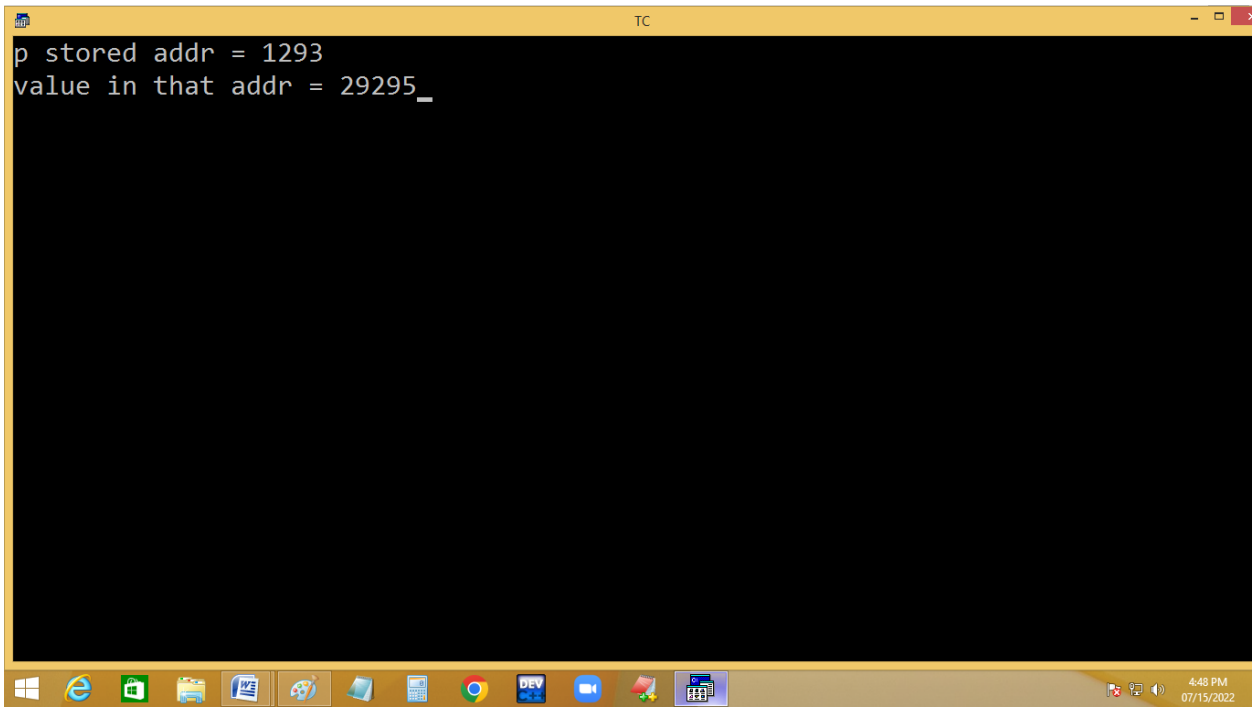

```
for( i=0; *p !='\0';  p++, i++ ) ;

p( i ); ==> 3
```

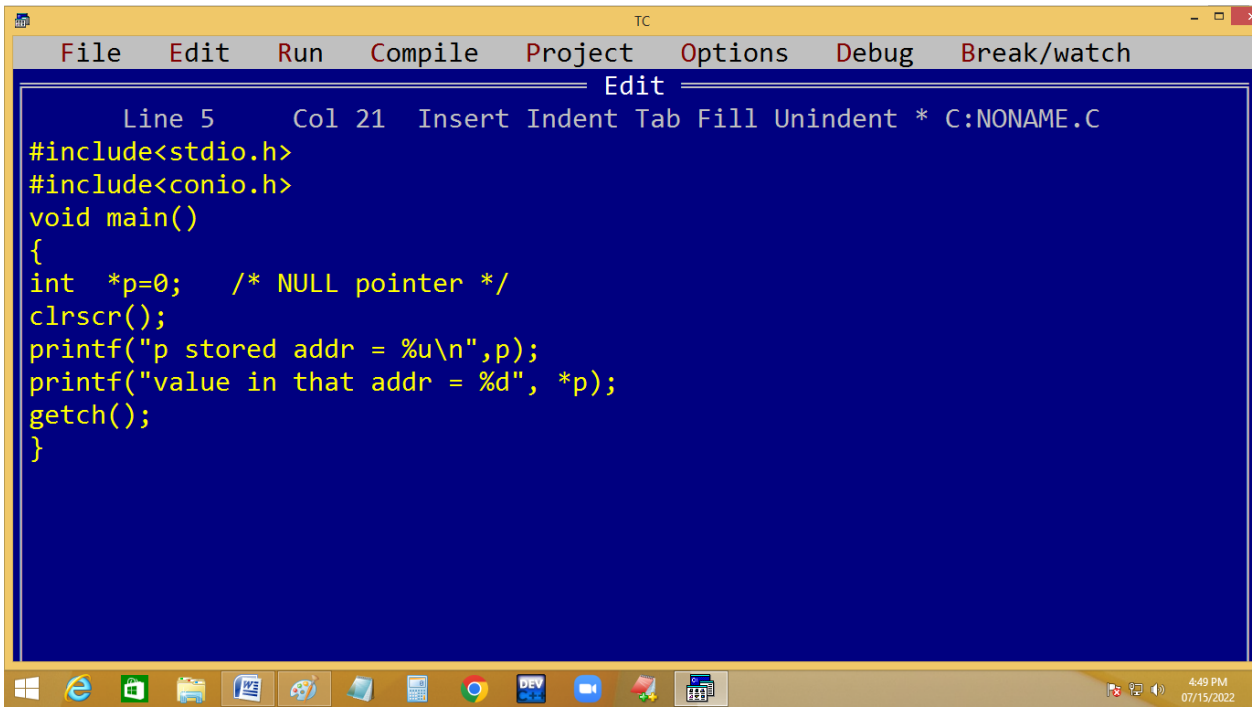
i	对P
0	65500 value - a
1	65501 value - b
2	65502 value - c
3	65503 value - '\0'

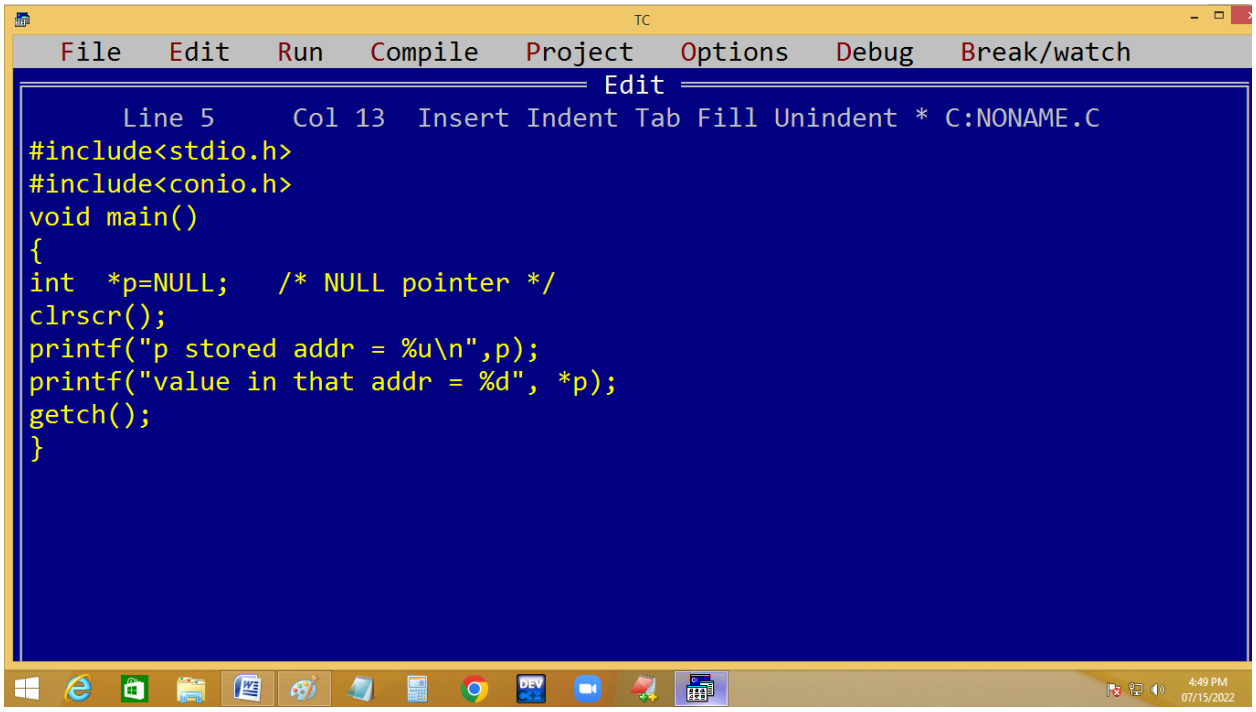
Bad / wild pointer:

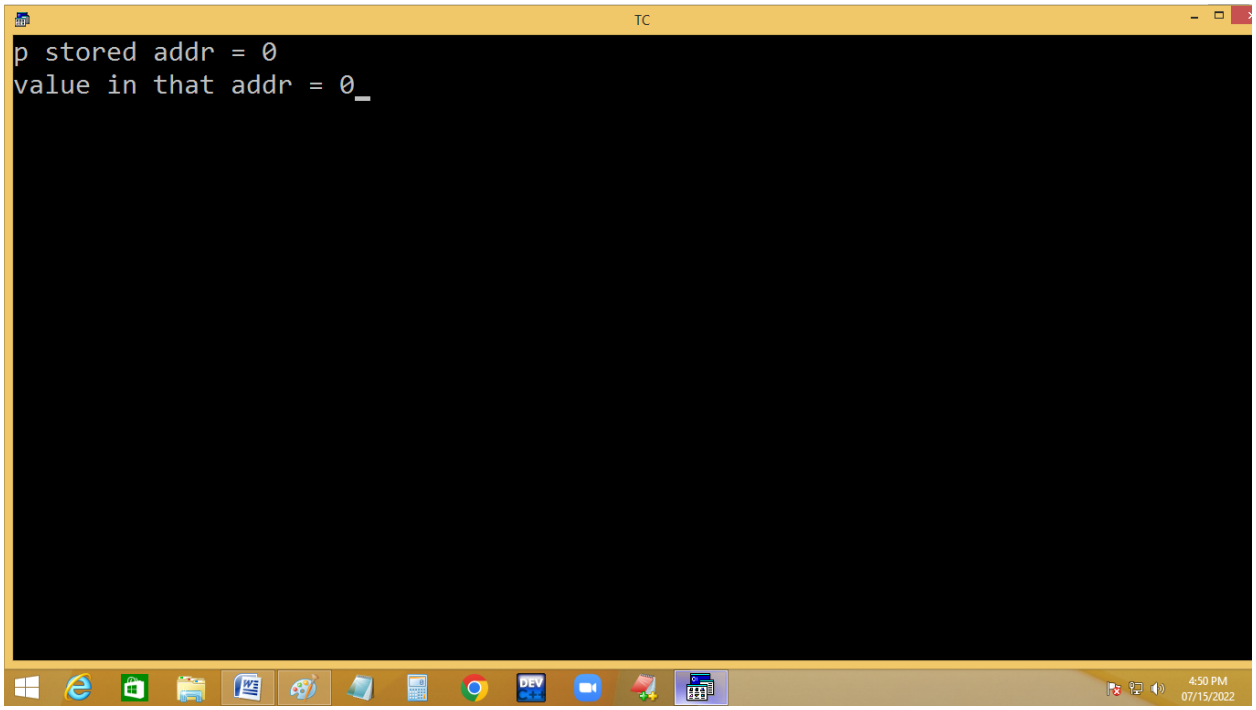
A pointer is declared but not initialized. In this situation the pointer is taking some unknown address and value. This kind of pointer is called bad / wild pointer.



NULL pointer: when a pointer is initialized with 0 or NULL then it is called NULL pointer. To avoid bad and dangling pointers we are using NULL pointer.

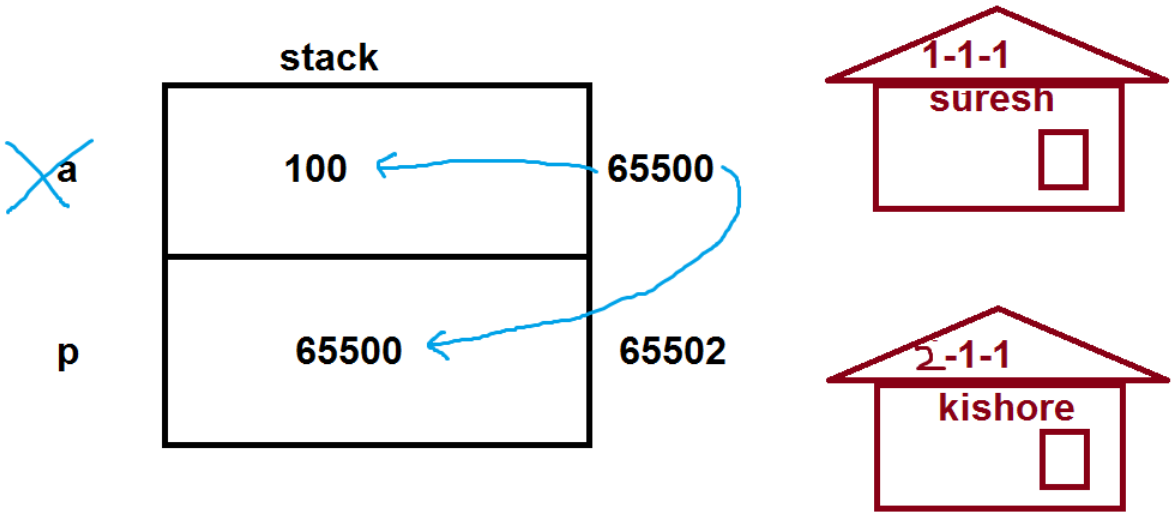

TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 5 Col 21 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int *p=0; /* NULL pointer */
clrscr();
printf("p stored addr = %u\n",p);
printf("value in that addr = %d", *p);
getch();
}
4:49 PM
07/15/2022


TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 5 Col 13 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int *p=NULL; /* NULL pointer */
clrscr();
printf("p stored addr = %u\n",p);
printf("value in that addr = %d", *p);
getch();
}
4:49 PM
07/15/2022



Dangling pointer:

A pointer is declared and some variable / memory address is assigned. After some time that variable or memory released [deleted]. But still the pointer is storing the deleted variable or memory address. This kind of pointer is called dangling pointer. To avoid this we are using NULL pointer.



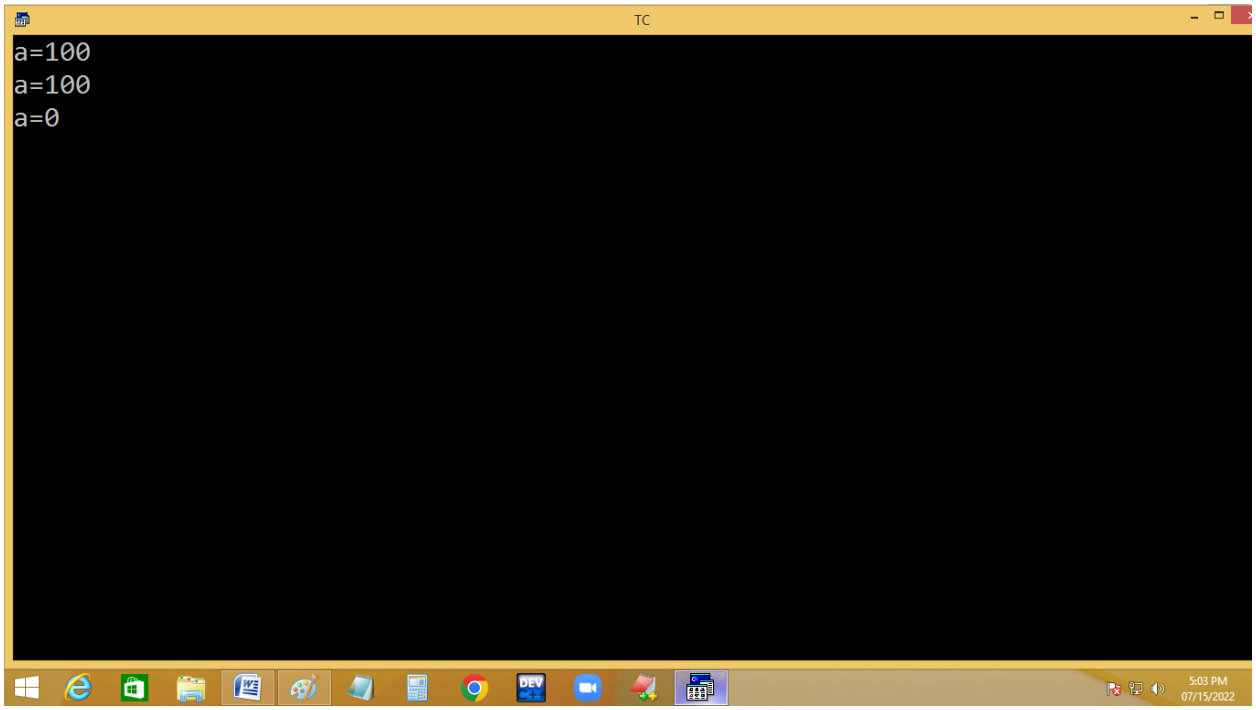
TC

File Edit Run Compile Project Options Debug Break/watch

Edit

Line 1 Col 19 Insert Indent Tab Fill Unindent * C:NONAME.C

```
#include<stdio.h>
#include<conio.h>
void main()
{
int  *p;   /* bad pointer */
clrscr();
{
int a=100; /* local var */
p = &a;
printf("a=%d\n",*p);
} /* a deleted */
printf("a=%d\n", *p);
p=NULL;
printf("a=%d",*p);
getch();
}
```

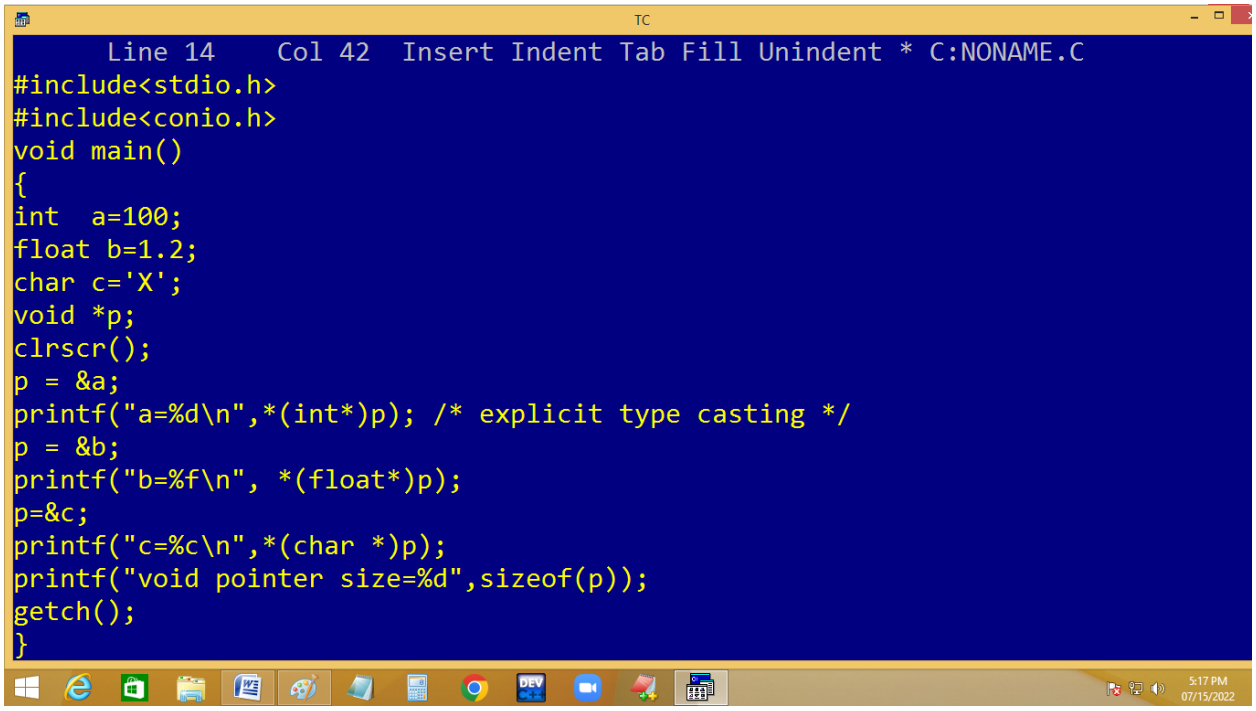


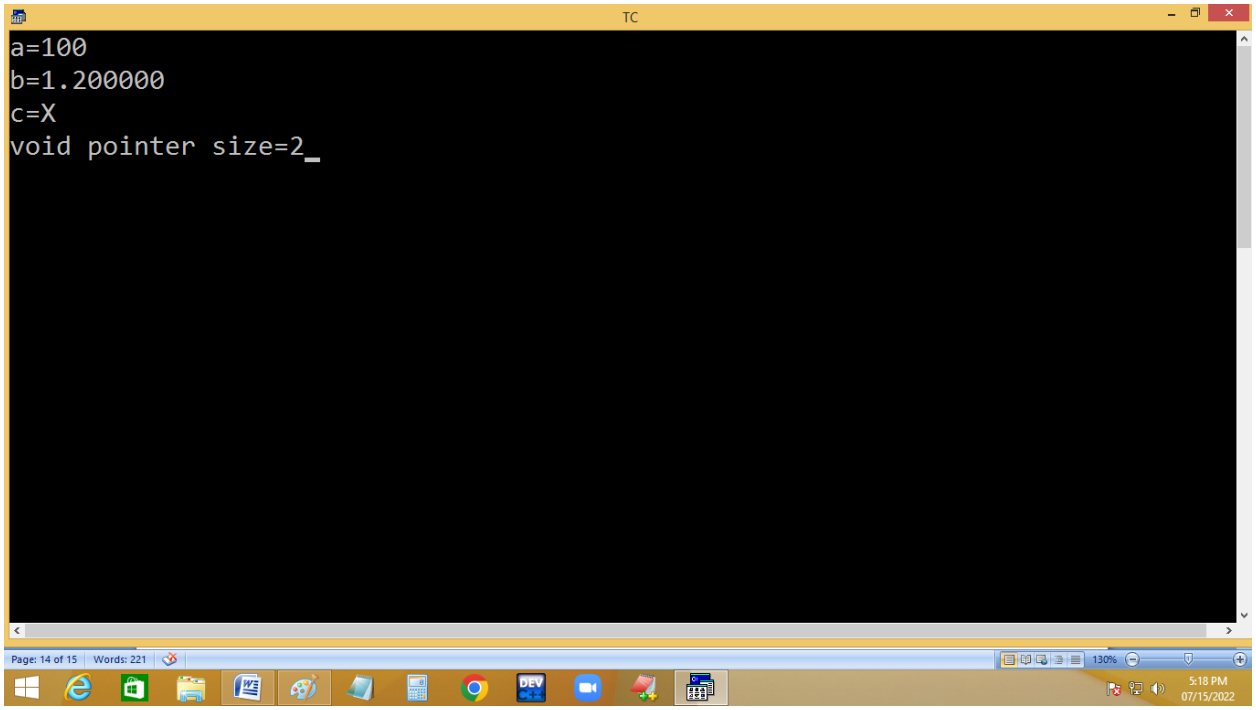
void/generic pointer:

pointer is used to store the address of another variable / memory. When several variables with different type, we have to declare several pointers.

void pointer stores any type of address. But before going to access void pointer, explicit type casting should be provided.

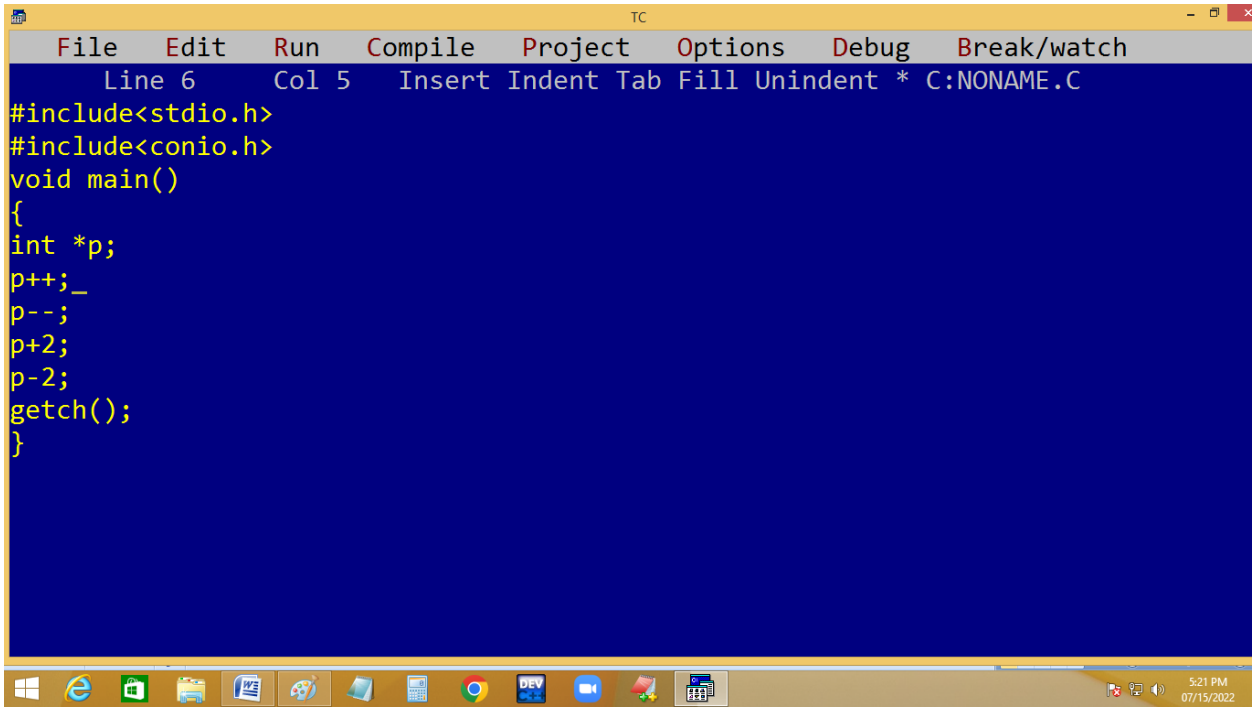
Void pointer takes 2 bytes. It is very much used in dynamic memory allocation.

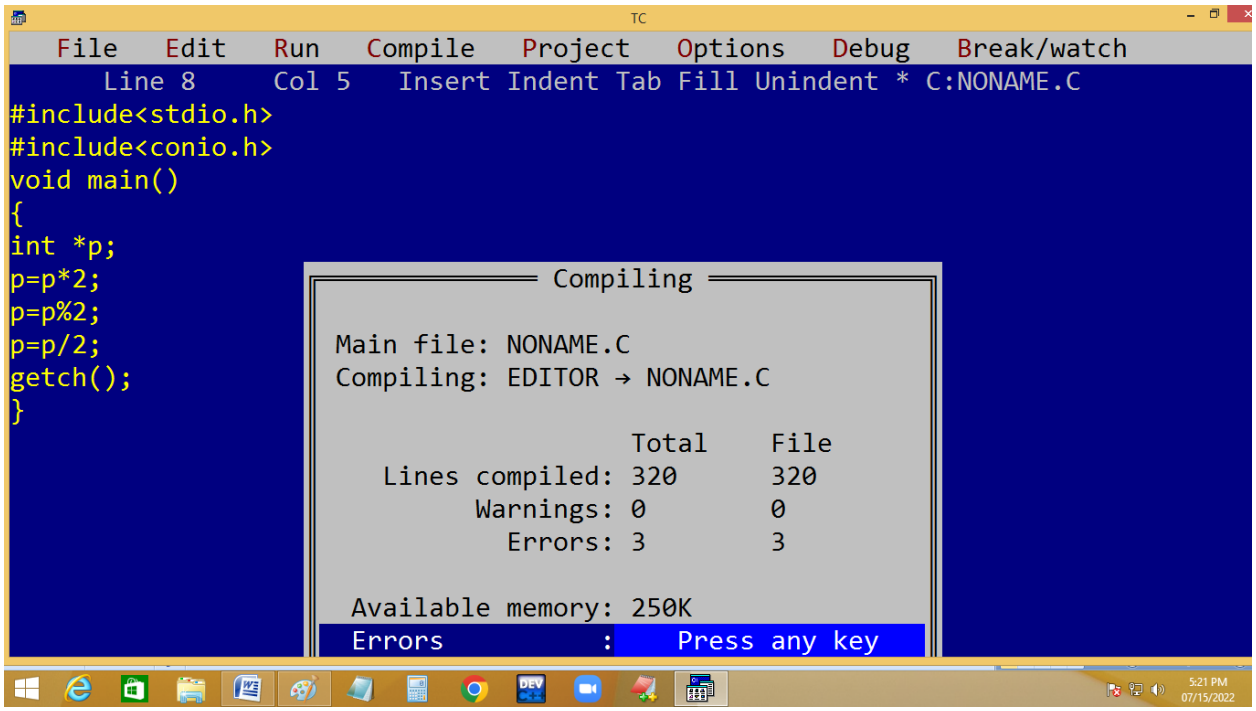


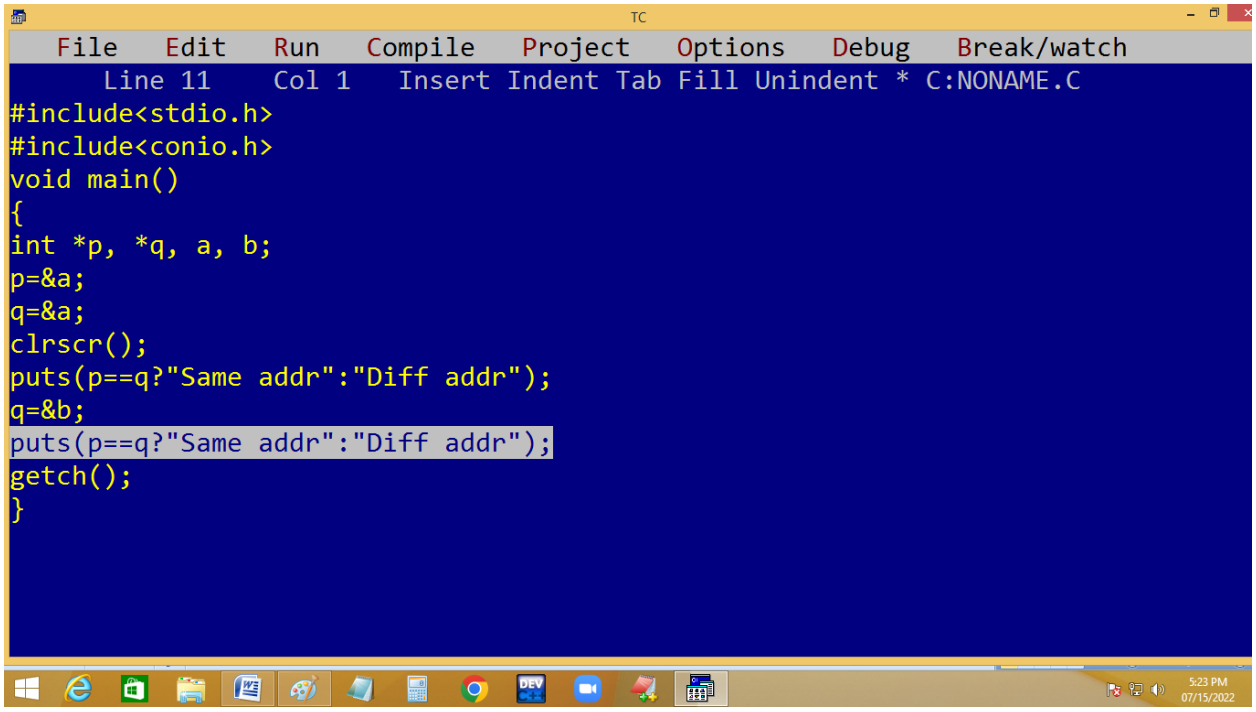


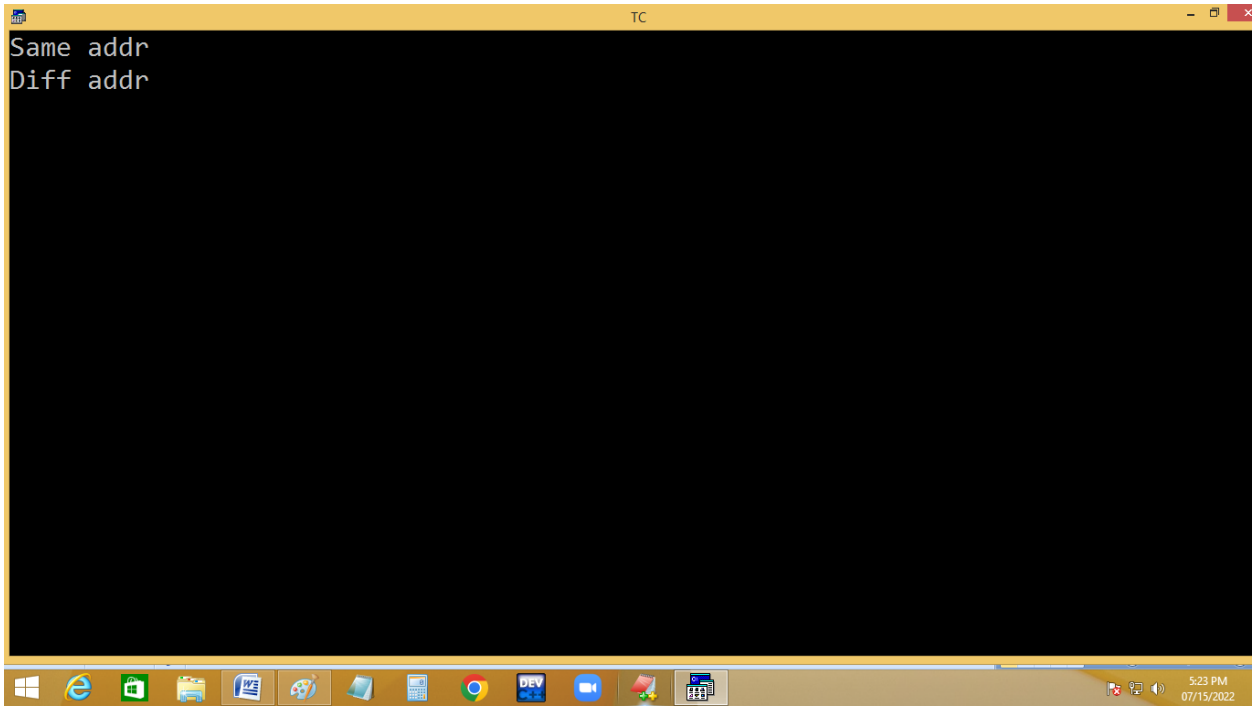
Pointer arithmetic:

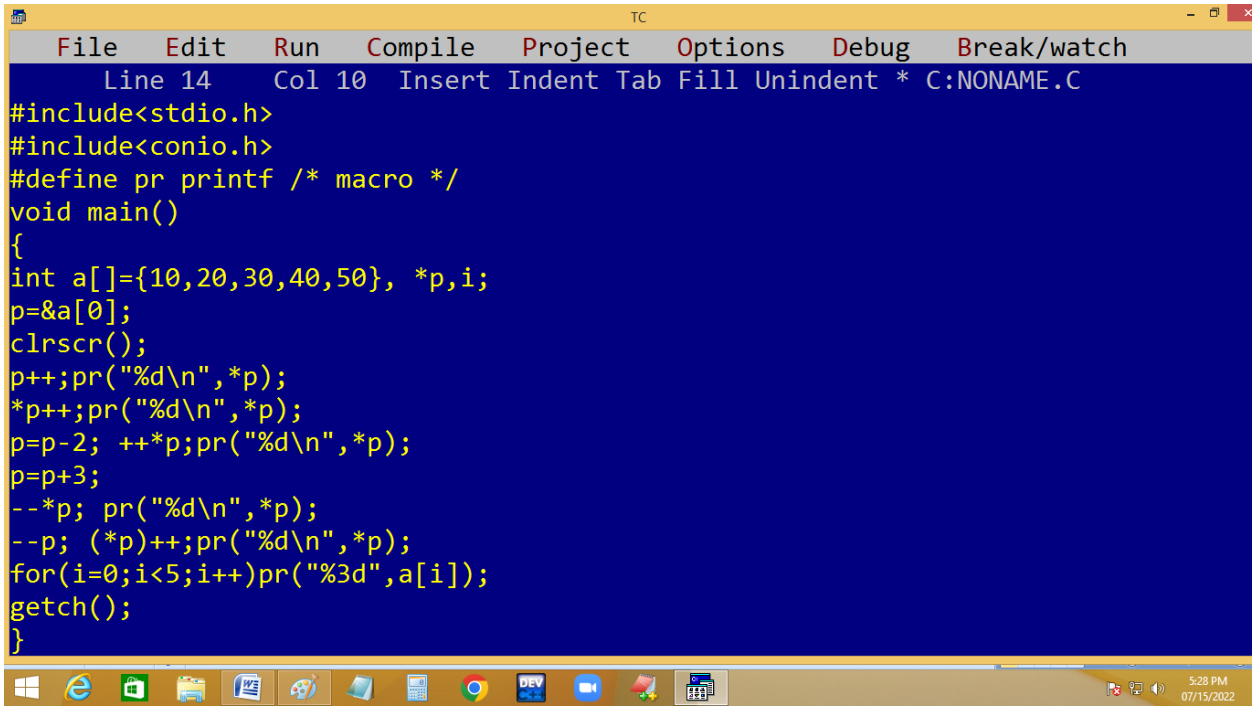
Like general variables we can compare, +, -, ++, -- the pointer values. But we can't perform %, / , * on pointers.

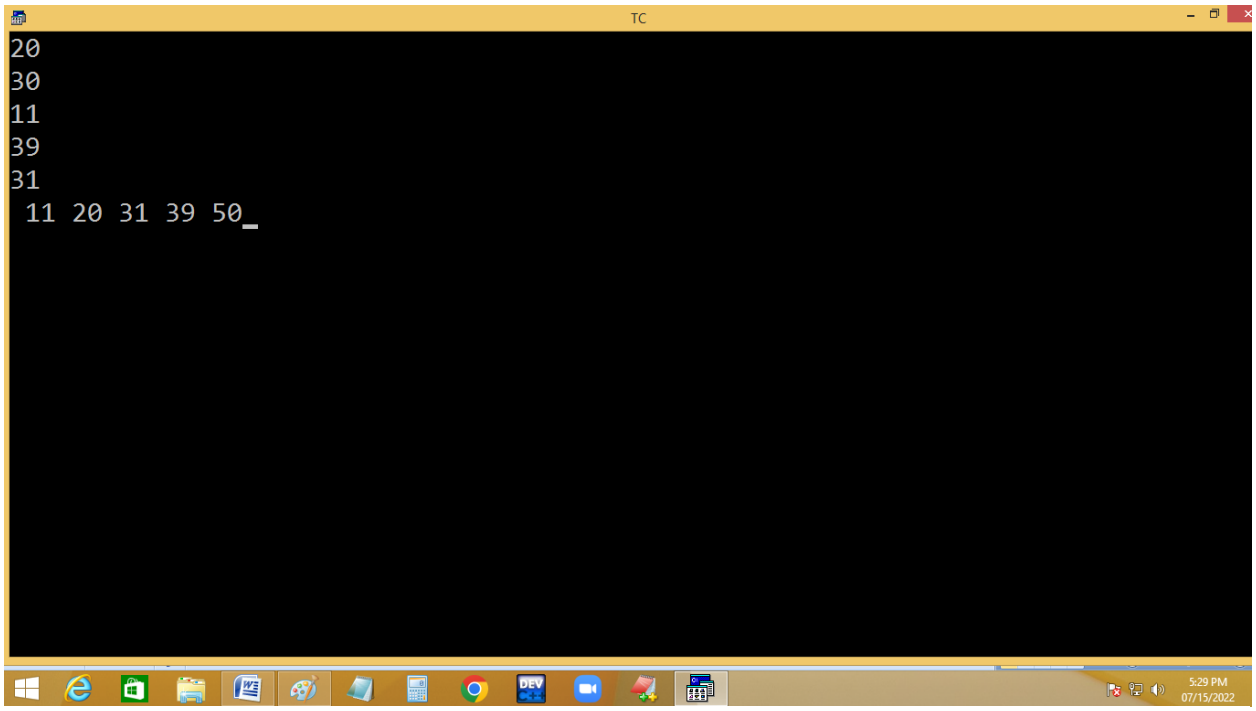


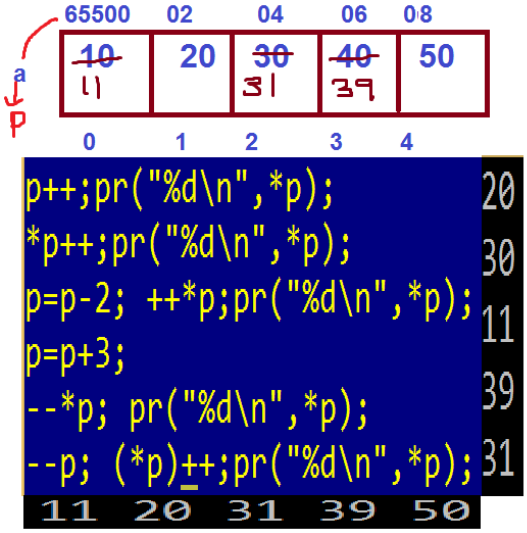

TC
File Edit Run Compile Project Options Debug Break/watch
Line 8 Col 5 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int *p;
p=p*2;
p=p%2;
p=p/2;
getch();
}
Compiling
Main file: NONAME.C
Compiling: EDITOR → NONAME.C
Total File
Lines compiled: 320 320
Warnings: 0 0
Errors: 3 3
Available memory: 250K
Errors : Press any key


TC
File Edit Run Compile Project Options Debug Break/watch
Line 11 Col 1 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
void main()
{
int *p, *q, a, b;
p=&a;
q=&a;
clrscr();
puts(p==q?"Same addr":"Diff addr");
q=&b;
puts(p==q?"Same addr":"Diff addr");
getch();
}


TC
Same addr
Diff addr


TC
File Edit Run Compile Project Options Debug Break/watch
Line 14 Col 10 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
#define pr printf /* macro */
void main()
{
int a[]={10,20,30,40,50}, *p,i;
p=&a[0];
clrscr();
p++;pr("%d\n",*p);
*p++;pr("%d\n",*p);
p=p-2; ++*p;pr("%d\n",*p);
p=p+3;
--*p; pr("%d\n",*p);
--p; (*p)++;pr("%d\n",*p);
for(i=0;i<5;i++)pr("%3d",a[i]);
getch();
}



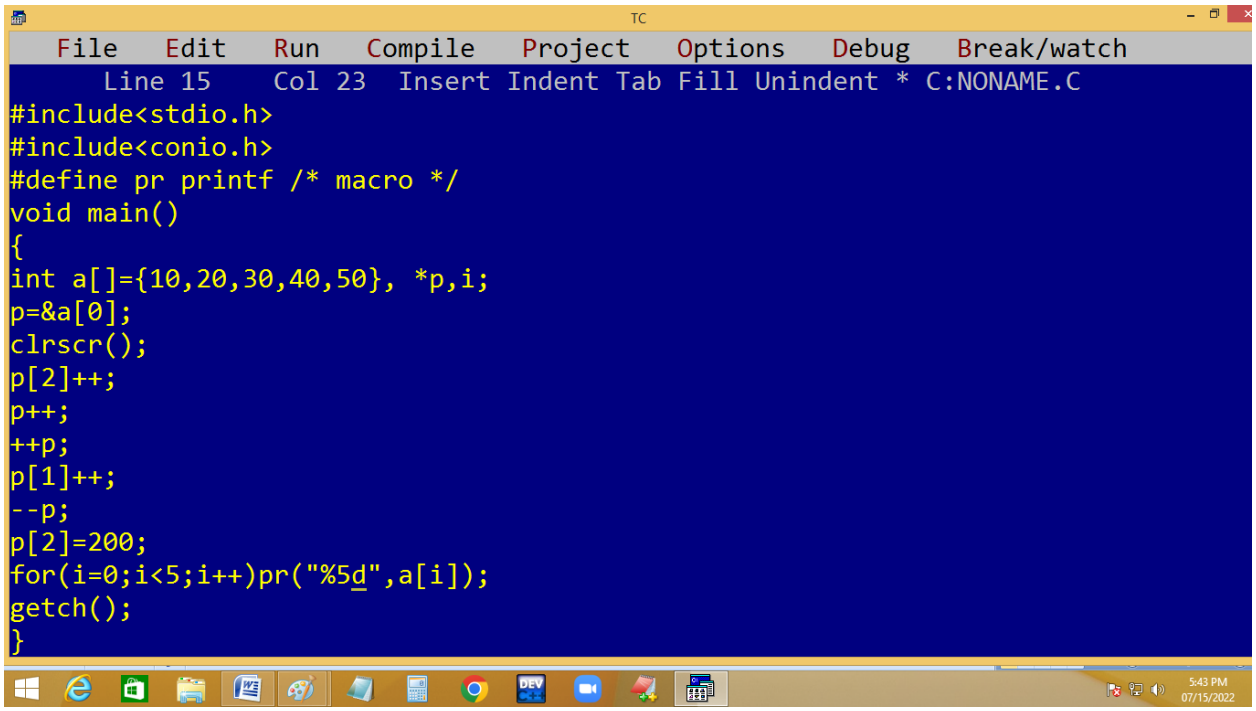


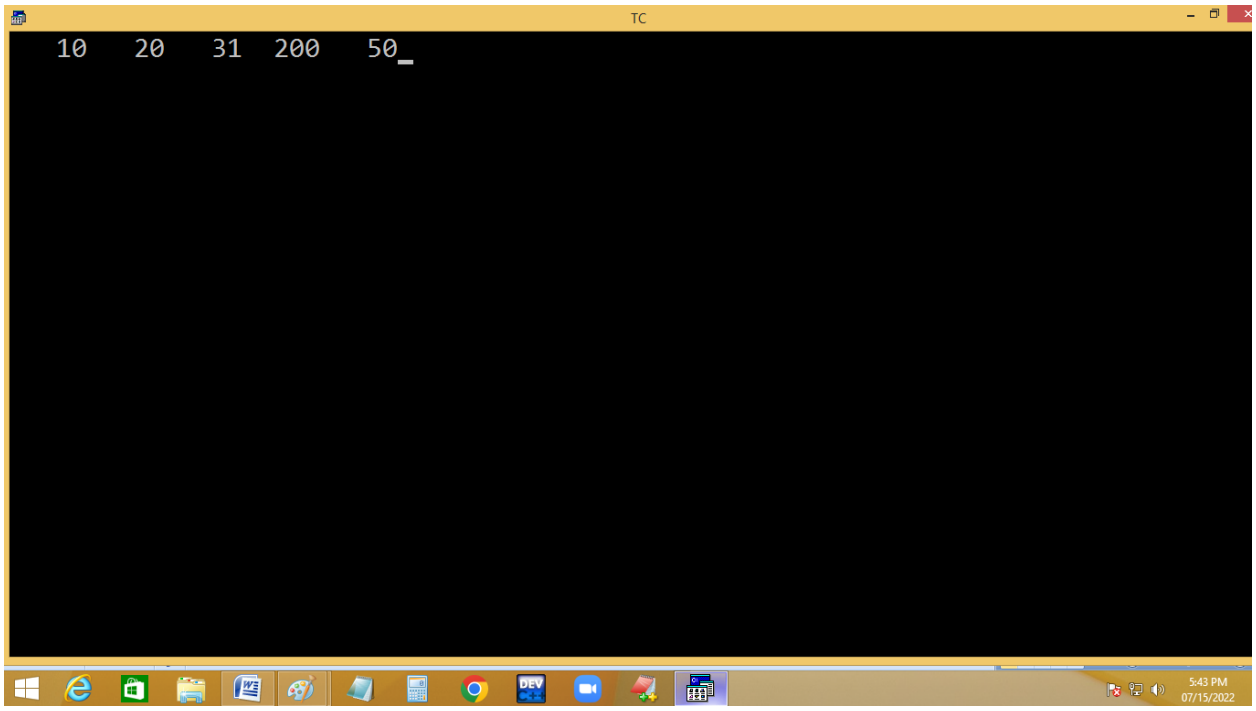
p++==>65500+1*2=65502;
pr(*p) ==>value at 65502==>20
*p++==>65502+1*2=65504
pr(*p) ==>value at 65504==>30

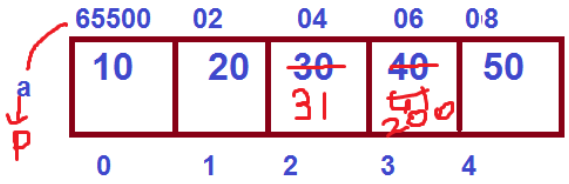
p=p-2==>65504-2*2==>65500
++*p ==> value at 65500++==>10++==>11

pr(*p)==>value at 65500 ==> 11
p=p+3==>65500+3*2==>65506
--*p==>value at 65506--==>40--==>39

pr(*p)==>value at 65506 ==> 39
--p ==>65506-1*2==>65504
(*p)++==>value at 65504++==>30++==>31
pr(*p)==>value at 65504 ==> 31


TC
File Edit Run Compile Project Options Debug Break/watch
Line 15 Col 23 Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
#define pr printf /* macro */
void main()
{
int a[]={10,20,30,40,50}, *p,i;
p=&a[0];
clrscr();
p[2]++;
p++;
++p;
p[1]++;
--p;
p[2]=200;
for(i=0;i<5;i++)pr("%5d",a[i]);
getch();
}






```
p[2]++;
p++;
++p;
p[1]++;
--p;
p[2]=200;
```

```
10   20   31   200   50
```

p = 65500
p[2]++==>30++ =>31

p = 65500

p++==>65502
++p==>65504
p[1]++==> 65506++ ==>40++ ==>41

p=65504
--p; 65502
p[2]=200;
65502+2*2==>65506==> 41 ==> 200