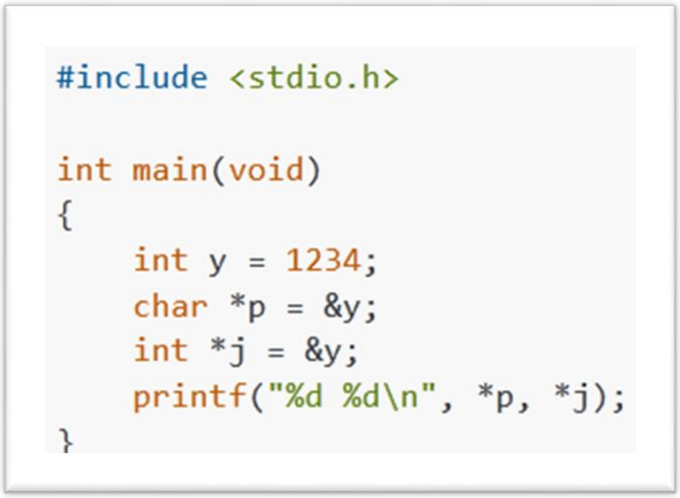
Hints on Pointers in C

BY / SOHAIB DAR



```
#include <stdio.h>

int main(void)
{
    int y = 1234;
    char *p = &y;
    int *j = &y;
    printf("%d %d\n", *p, *j);
}
```

If you have something like….

int y = 1234;
int *p = &x;

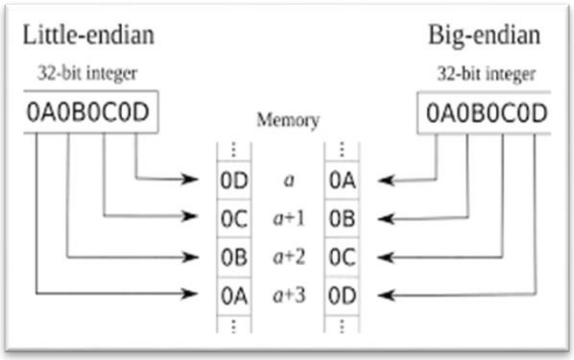
- **If we dereference Pointer p then it will read integer bytes because you declared it to be pointer to int.**
- **size of int is 4 bytes (for 32/64-bit platforms)**
- **but it is machine dependent that is why it will use sizeof() operator to know correct size and will read so.**

int y   = 1234;
char *p = &y;
int *j   = &y;

Pointer p points to y (pointer to a char) so it will only read one byte or whatever byte char is.

- 1234 in binary as
  00000000 00000000 00000100 11010010
- **Now if our machine is little endian it will store the bytes reversing them**

11010010 00000100 00000000 00000000



- **11010010 is at address 00 Hypothetical address,**
- **00000100 is at address 01, and so on.**

```
BE:      00   01   02   03
       +----+----+----+----+
    y: | 00 | 00 | 04 | d2 |
       +----+----+----+----+
```

```
LE:      00   01   02   03
       +----+----+----+----+
    y: | d2 | 04 | 00 | 00 |
       +----+----+----+----+
```

So **pointer p** it will **read only first byte**

The output = **- 46**   in case of **(signed char)** and = **210** in case of **(unsigned char).**

as Byte read would be **11010010** **(signed char).**

- **negative numbers are represented as <u>2's Complement</u> so the most-significant bit is the sign bit.**
- **First bit 1 denotes the sign.**

11010010 = –128 + 64 + 16 + 2 = - 46

if we dereference **pointer j,** it will completely read all bytes of int, so pointer to int and output = **1234**

If we declare **pointer j as int *j** then will **read sizeof(int)** here **4** bytes**(machine dependent).**

char *p = &y is **invalid and a cast is required.**

We need to explicitly **cast to char*** as **char *p = &y** **is a constraint violation**

char * and int * are not compatible types and may have different sizes. Thus, we must explicitly cast the source to the target type:

Write >> char *p = (char *)&y.

---

notice

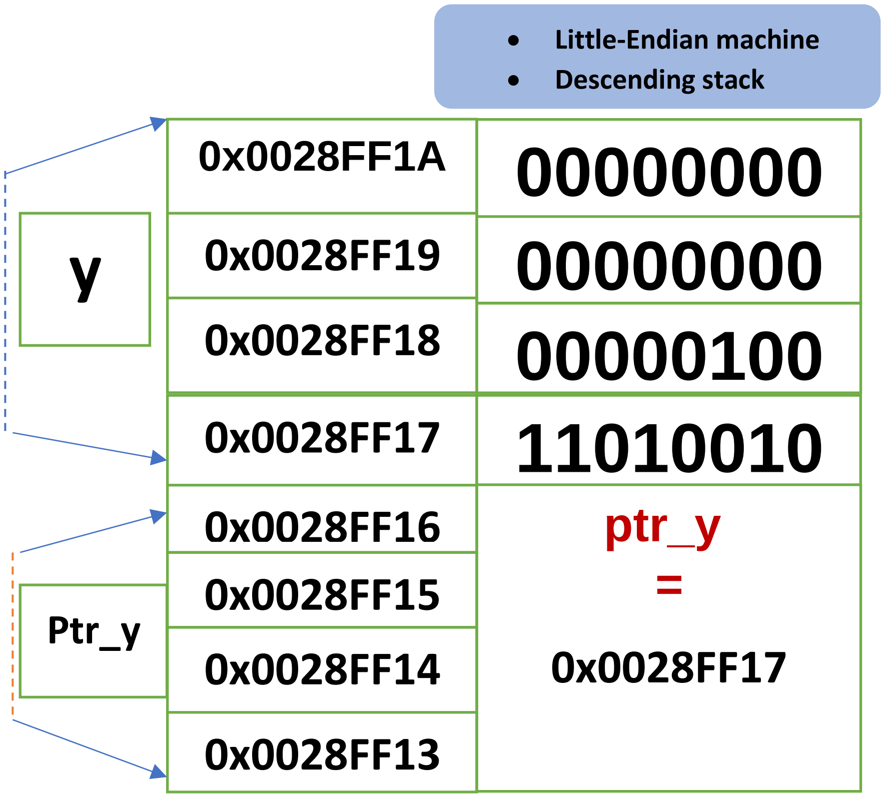
The dereferencing looks like a command to the processor to get Specific width of data (step size of the pointer data)

Such as:

```
char *p;    //step size = 1byte
int *p;     //step size = 4byte
```

For Example

```
int y       = 1234;     //1234 = 00000000 00000000 00000100 11010010
char *ptr_y = &y;
```



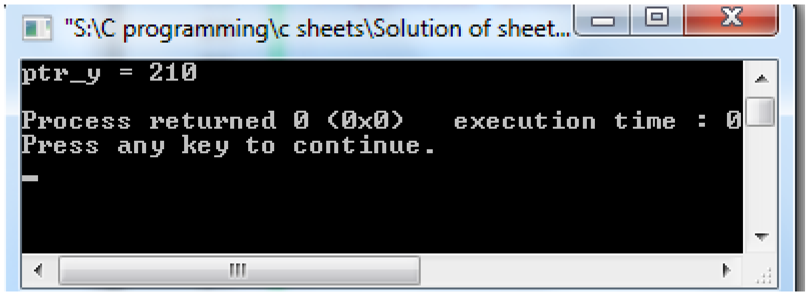
➢ **Output of …** printf("ptr_y = %d\n",*ptr_y); **??**

Ptr_y = (ptr_y+0)

***ptr_y = Bsae address + (0* (char step size))**

= 0x0028FF17 + 0 *1

= *(0X0028FF17) = 11010010 = 210


"S:\C programming\c sheets\Solution of sheet...

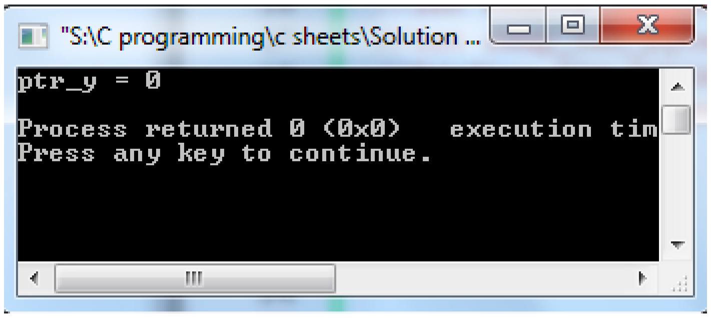
```
ptr_y = 210

Process returned 0 (0x0)   execution time : 0
Press any key to continue.
```

➢ **Output of …** printf("ptr_y = %d\n",*(ptr_y+2)); **???**

***(ptr_y+2) = Base address + 2(char step size)**

= 0x0028FF17 + 2 * 1

= *(0x0028FF19) = 0

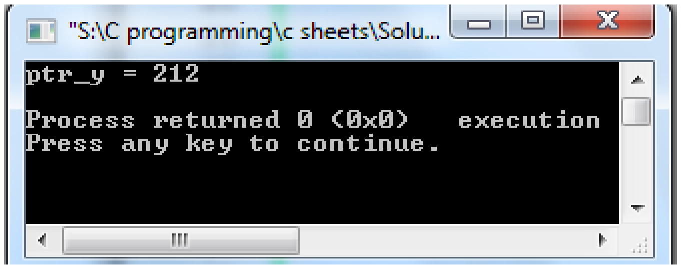

"S:\C programming\c sheets\Solution ...

```
ptr_y = 0

Process returned 0 (0x0)   execution tim
Press any key to continue.
```

- **Take care of the brackets and dereferencing sign(*), again take care.**

Output of … printf("ptr_y = %d\n",*ptr_y+2); **???**

= 210 + 2 = 212


"S:\C programming\c sheets\Solu...

```
ptr_y = 212

Process returned 0 (0x0)   execution
Press any key to continue.
```

Another simple syntax

*(Ptr_y + n)   =   ptr_y[n]

*(Ptr_y +2)   =   ptr_y[2]   ……. and so on .

Look here

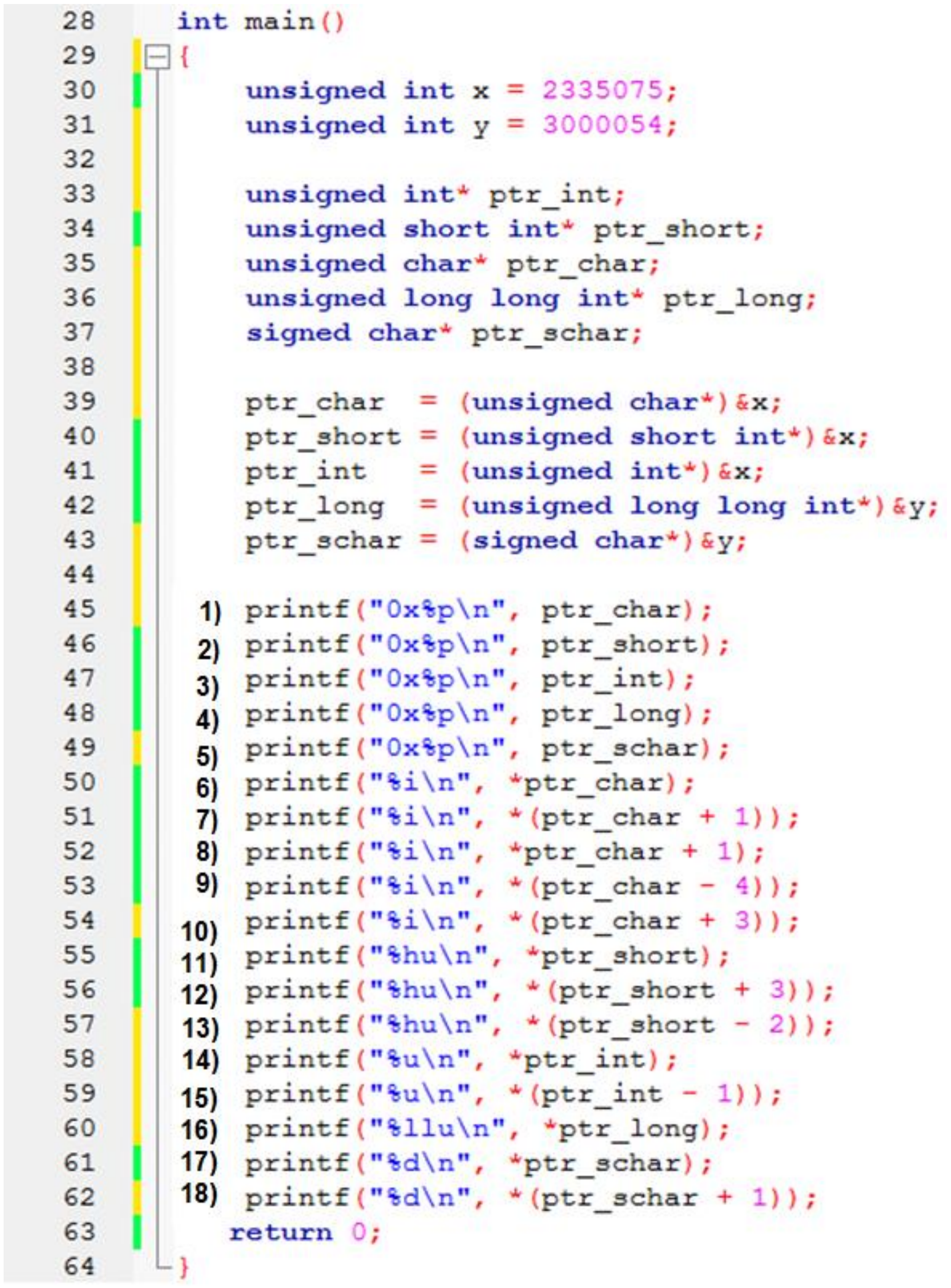
Output of … printf("ptr_y = %d\n", (-1)[ptr_y]); ???

(-1)[ptr_y] = *(ptr_y + (-1)) = *(0x0028FF17 - 1) = *(0x0028FF16)

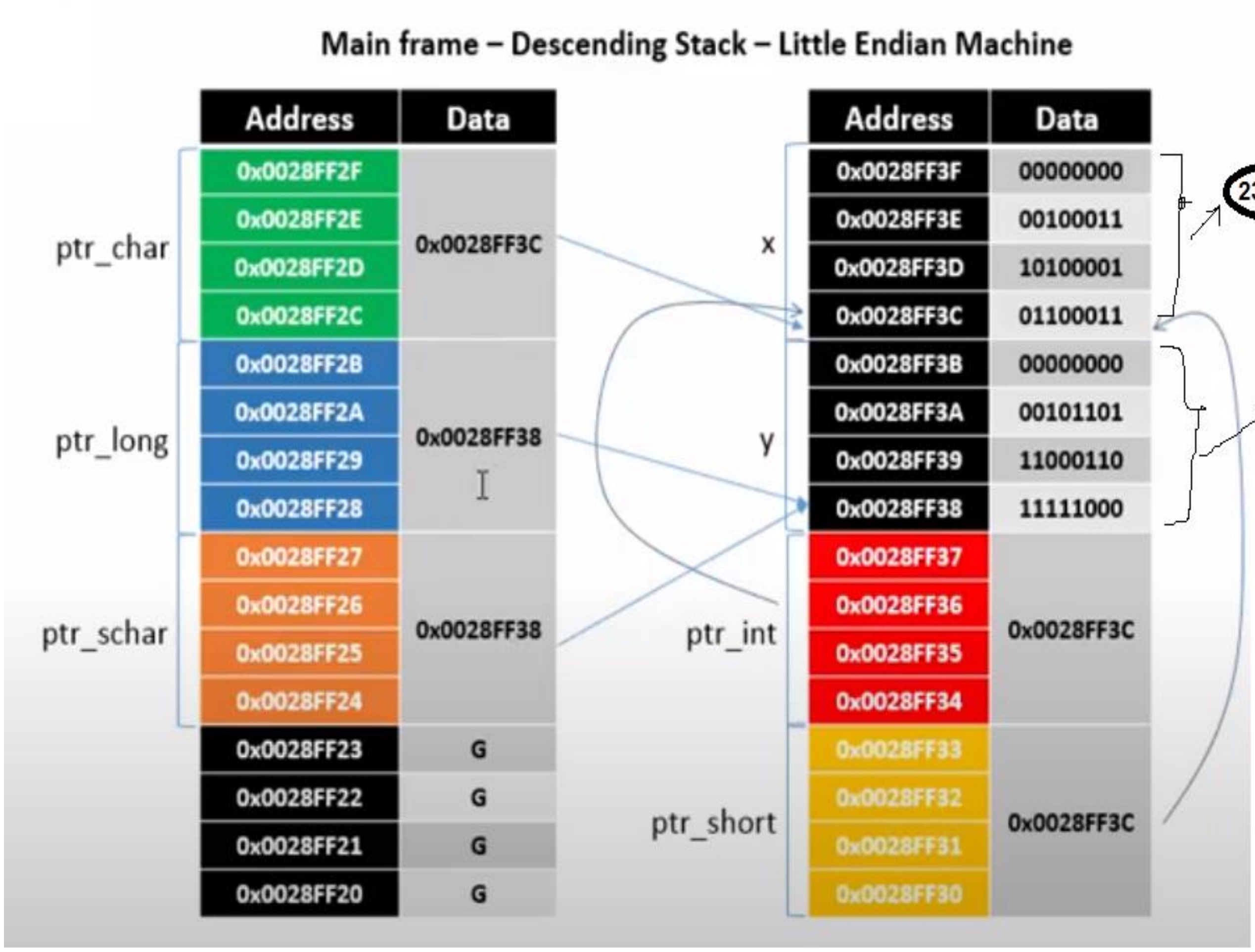
Output of … printf("ptr_y = %d\n", -1[ptr_y]); ???

-1[ptr_y] = - *(ptr_y + 1) = - *(0x0028FF17 + 1) = - *(0x0028FF18) = - 4

One more Example:-



```
int main()
{
    unsigned int x = 2335075;
    unsigned int y = 3000054;

    unsigned int* ptr_int;
    unsigned short int* ptr_short;
    unsigned char* ptr_char;
    unsigned long long int* ptr_long;
    signed char* ptr_schar;

    ptr_char  = (unsigned char*)&x;
    ptr_short = (unsigned short int*)&x;
    ptr_int   = (unsigned int*)&x;
    ptr_long  = (unsigned long long int*)&y;
    ptr_schar = (signed char*)&y;

 1) printf("0x%p\n", ptr_char);
 2) printf("0x%p\n", ptr_short);
 3) printf("0x%p\n", ptr_int);
 4) printf("0x%p\n", ptr_long);
 5) printf("0x%p\n", ptr_schar);
 6) printf("%i\n", *ptr_char);
 7) printf("%i\n", *(ptr_char + 1));
 8) printf("%i\n", *ptr_char + 1);
 9) printf("%i\n", *(ptr_char - 4));
10) printf("%i\n", *(ptr_char + 3));
11) printf("%hu\n", *ptr_short);
12) printf("%hu\n", *(ptr_short + 3));
13) printf("%hu\n", *(ptr_short - 2));
14) printf("%u\n", *ptr_int);
15) printf("%u\n", *(ptr_int - 1));
16) printf("%llu\n", *ptr_long);
17) printf("%d\n", *ptr_schar);
18) printf("%d\n", *(ptr_schar + 1));
    return 0;
}
```



<u>the output</u>

1) 0x0028FF3C

2) 0x0028FF3C

3) 0x0028FF3C

4) 0x0028FF38

5) 0x0028FF38

6) 99

7) 161

8) 99 + 1 = 100

9) 248

10) 0

11) 41315

12) garbage

13) 50936

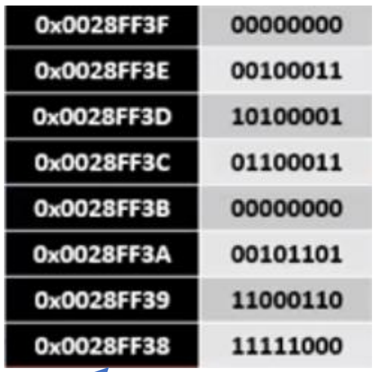
14) 2335075

15) 3000056

16)<u>1002907076170725б</u>

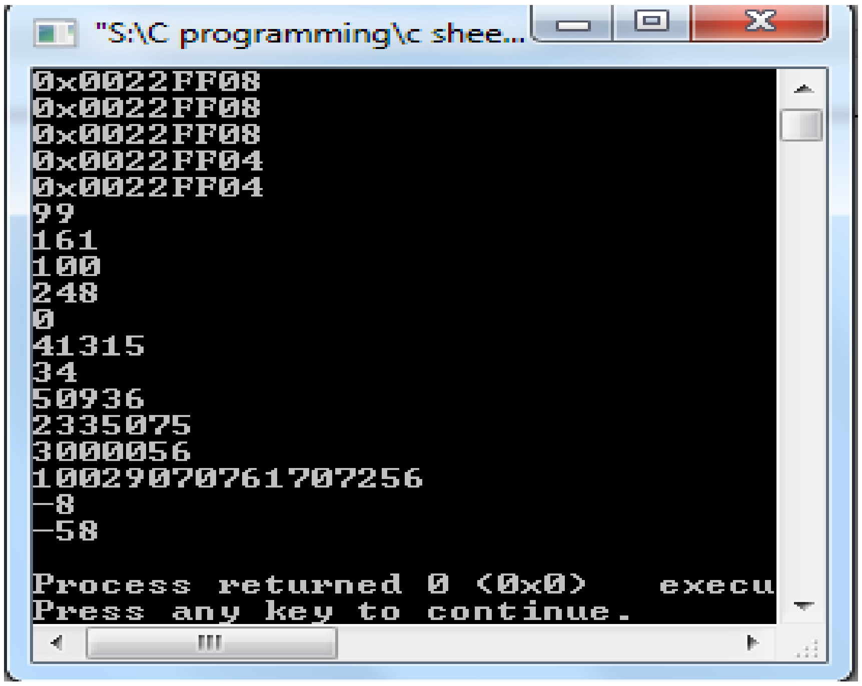
17) 8+16+32+64-128 = -8

18) 2+4+64 – 128 = -58



0x0028FF3F	00000000
0x0028FF3E	00100011
0x0028FF3D	10100001
0x0028FF3C	01100011
0x0028FF3B	00000000
0x0028FF3A	00101101
0x0028FF39	11000110
0x0028FF38	11111000

8-btyes of long long data type



```
0x0022FF08
0x0022FF08
0x0022FF08
0x0022FF04
0x0022FF04
99
161
100
248
0
41315
34
50936
2335075
3000056
1002907076 1707256
-8
-58

Process returned 0 (0x0)   execu
Press any key to continue.
```

Take care of format specifiers

Format Specifier	Type
%c	Character
%d	Signed integer
%e or %E	Scientific notation of floats
%f	Float values

Format Specifier	**Type**
%g or %G	Similar as %e or %E
%hi	Signed integer (short)
%hu	Unsigned Integer (short)
%i	integer
%l or %ld or %li	Long
%lf	Double
%Lf	Long double
%lu	Unsigned int or unsigned long
%lli or %lld	Long long
%llu	Unsigned long long
%o	Octal representation

Format Specifier	**Type**
%p	Pointer
%s	String
%u	Unsigned int
%x or %X	Hexadecimal representation
%n	Prints nothing
%%	Prints % character

The source Code is on My Github

- **Sheet #2 : https://github.com/SohaibDar61/Promblem-Solving-in-C/tree/main/Sheet%20%232**
- **sheet #1 : https://github.com/SohaibDar61/Codeforces-promblem-solving-/tree/main/Sheet%20%231**

Data Structure & Algorithms

- **https://github.com/SohaibDar61/Data_Structures**

E-mail: eng.sohaibdar@gmail.com
Phone:+201018741441
Upwork: https://www.upwork.com/freelancers/~019e71f2adc499c4e8
LinkedIn: https://www.linkedin.com/in/sohaibdar61

- **Anyone can edit or optimize the code , and I hope that revision helps and gets better.**

Thank You

#s