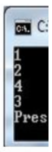
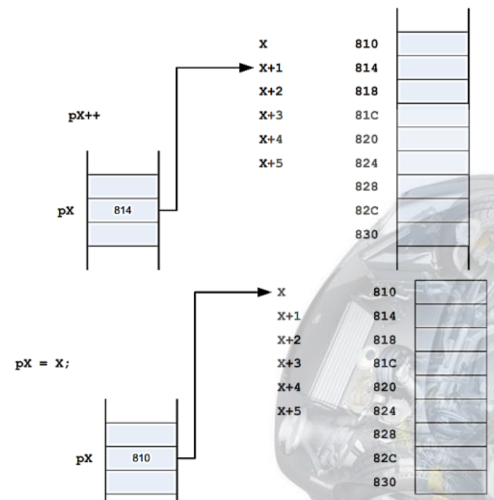
Pointer to Array

```c
#include "stdio.h"

void main()
{
        int x[5] = {1, 2, 3, 4, 5};
        int* p = x;
        printf("%d\r\n", *p);
        p++;
        printf("%d\r\n", *p);
        p = x + 3;
        printf("%d\r\n", *p);
        p--;
        printf("%d\r\n", *p);
```



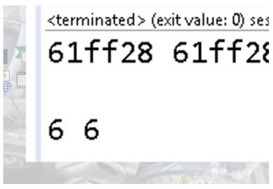
```
1
2
4
3
Pres
```



str[4] =equivalent= *(p1+4)

Pointer Variables

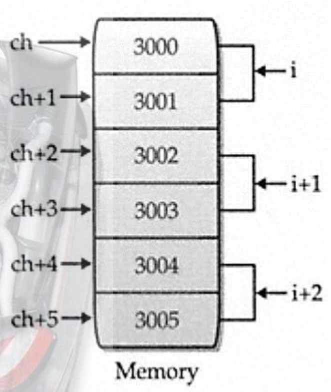
```c
int x = 6;
int* px;
px = &x;
printf("%x %x\r\n", &x, px);
printf("%d %d\r\n", x, *px);
```



```
<terminated> (exit value: 0) ses
61ff28 61ff28

6 6
```

Each time a pointer is incremented, it points to the memory location of the next element of its base type (assume 2-byte integers)

```c
char *ch = (char *) 3000;
int *i = (int *) 3000;
```



Pointer to Structure

```c
//Prepared by Eng.Keroles
#include <stdio.h>
struct SPerson
{
        char m_Name[18];
        int m_ID;
        char m_Age;
        short m_Salary;
        double m_Weight;
};
```

```c
int main(int argt ,char**argv) {
    struct SPerson manager =
    {"Mohamed Hady", 162, 39, 3000, 79.5};
    struct SPerson emploees[] = {
            {"Mostafa Said", 163, 30, 1500, 81.0},
            {"Ahmed Salah", 164, 25, 1200, 91.0},
            {"Safa Fayez", 165, 28, 1400, 65.0}};
    int i;
    struct SPerson* p;
    p = &manager;
    printf("manager: %s, %d, %d, %d, %lf\r\n",
            p->m_Name, p->m_ID, (int)p->m_Age,
            (int)p->m_Salary, p->m_Weight);
    p->m_Salary = 4000;
    printf("manager: %s, %d, %d, %d, %lf\r\n",
            manager.m_Name, manager.m_ID,
            (int) manager.m_Age, (int) manager.m_Sal
            manager.m_Weight);
    p = emploees;
    for(i=0;i<sizeof(emploees)/sizeof(struct SPerson
            i++, p++)
    {
        printf("emploee %d: %s, %d, %d, %d, %lf\r\n"
                i+1, p->m_Name, p->m_ID,
                (int)p->m_Age, (int)p->m_Salary,
                p->m_Weight);
    }
    return 0    ;
```

ENG. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

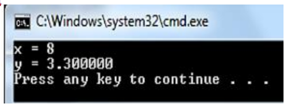
Pointer pass by refrence

`void Sort(int values[], int nValues)`	is equivelant to	`void Sort(int* values, int nValues)`

Programmer is free to choose which notation is sutable, because both methods gives the same behaviour.

Pointer with Unknown Type (void)*

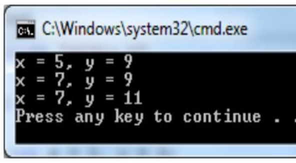
```c
#include "stdio.h"

void main()
{
        int x = 5;
        double y = 10.3;
        void* p;

        p = &x;
        *(int*)p = 8;
        printf("x = %d\r\n", x);

        p = &y;
        *(double*)p = 3.3;
        printf("y = %lf\r\n", y);
}
```



```
x = 8
y = 3.300000
Press any key to continue . . .
```

Pointer to Pointer

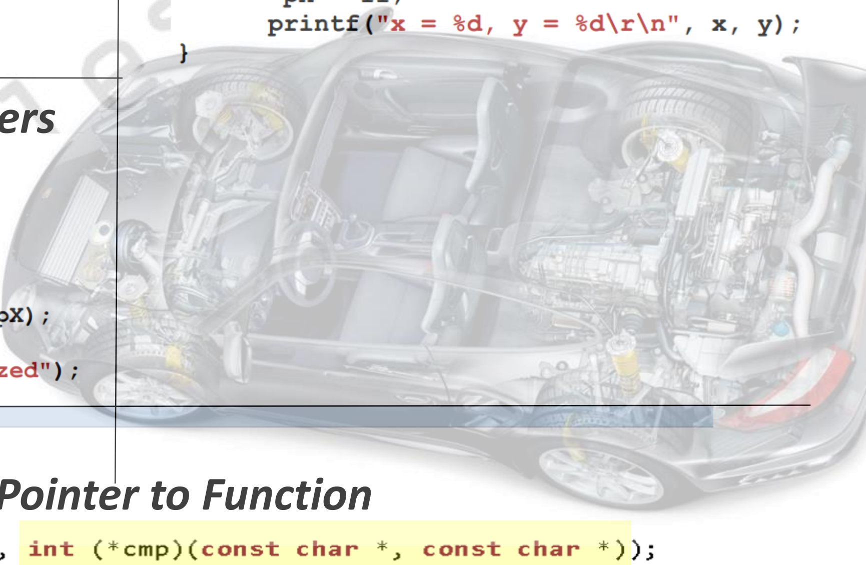
```c
#include "stdio.h"

void main()
{
        int x = 5, y = 9;
        int* pX = &x; //Pointer
        int** ppX = &pX; //Pointer to Pointer

        printf("x = %d, y = %d\r\n", x, y);

        **ppX = 7;
        printf("x = %d, y = %d\r\n", x, y);

        *ppX = &y;

        *pX = 11;
        printf("x = %d, y = %d\r\n", x, y);
}
```



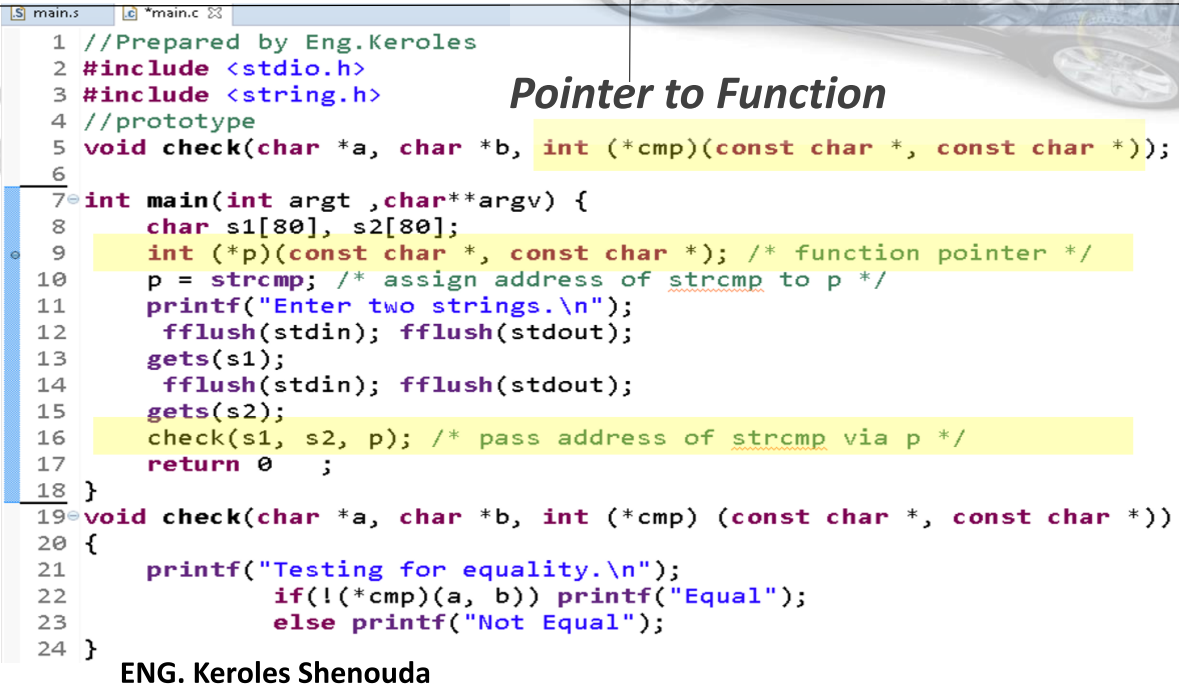
```
x = 5, y = 9
x = 7, y = 9
x = 7, y = 11
Press any key to continue .
```

NULL and Unassianed Pointers

```c
#include "stdio.h"

void main()
{
        int* pX = NULL;
        if(pX!=NULL)
                printf("pX point to %d", *pX);
        else
                printf("pX is not initialized");
}
```



Pointer to Function



```c
//Prepared by Eng.Keroles
#include <stdio.h>
#include <string.h>
//prototype
void check(char *a, char *b, int (*cmp)(const char *, const char *));

int main(int argt ,char**argv) {
    char s1[80], s2[80];
    int (*p)(const char *, const char *); /* function pointer */
    p = strcmp; /* assign address of strcmp to p */
    printf("Enter two strings.\n");
     fflush(stdin); fflush(stdout);
    gets(s1);
     fflush(stdin); fflush(stdout);
    gets(s2);
    check(s1, s2, p); /* pass address of strcmp via p */
    return 0    ;
}
void check(char *a, char *b, int (*cmp) (const char *, const char *))
{
    printf("Testing for equality.\n");
            if(!(*cmp)(a, b)) printf("Equal");
            else printf("Not Equal");
}
```

ENG. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

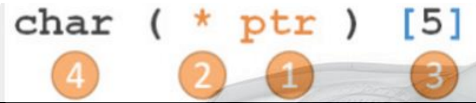
Pointers Tricks

How to Read C complex pointer

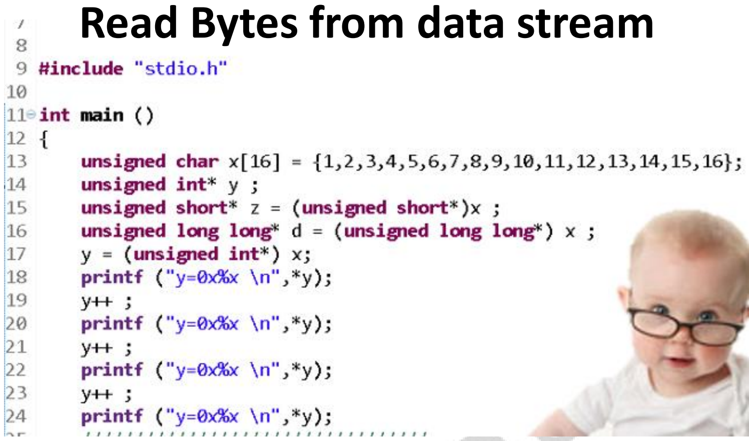
Operator	Precedence	Associative
(),[]	1	Left to Right
*,Identifier	2	Right to Left
Data Type	3	–

Different Terms From Table –

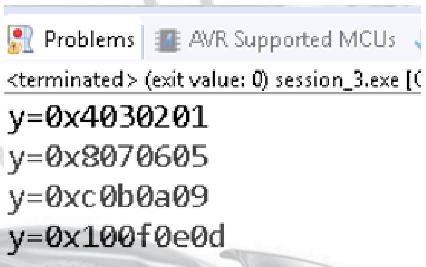
()	Bracket operator **OR** function operator.
[]	Array subscription operator
*	Pointer operator
Identifier	**Name of Pointer Variable**
Data type	Type of Pointer



Read Bytes from data stream



```
#include "stdio.h"

int main ()
{
    unsigned char x[16] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
    unsigned int* y ;
    unsigned short* z = (unsigned short*)x ;
    unsigned long long* d = (unsigned long long*) x ;
    y = (unsigned int*) x;
    printf ("y=0x%x \n",*y);
    y++ ;
    printf ("y=0x%x \n",*y);
    y++ ;
    printf ("y=0x%x \n",*y);
    y++ ;
    printf ("y=0x%x \n",*y);
```

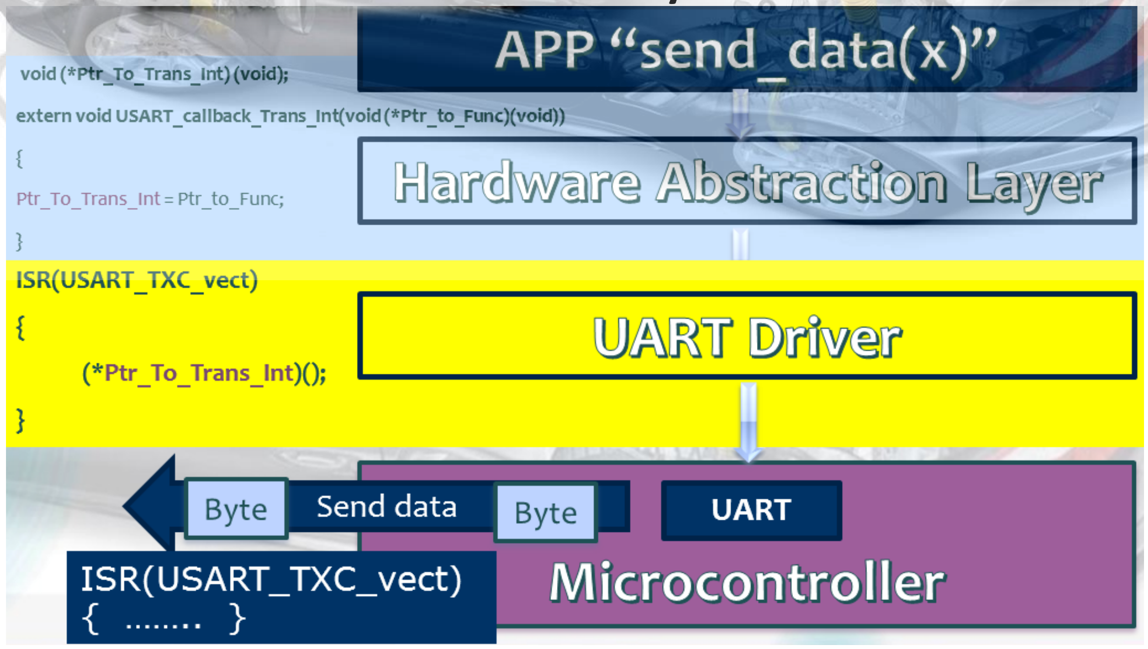


```
<terminated> (exit value: 0) session_3.exe [C
y=0x4030201
y=0x8070605
y=0xc0b0a09
y=0x100f0e0d
```

Pointer with Constant

Example	Value constant	Pointer Constant
char *ptr	No	No
const char *ptr	Yes	No
char const *ptr	Yes	No
char* const ptr	No	Yes
const char *const ptr	Yes	Yes

Modularity



```
void (*Ptr_To_Trans_Int)(void);
extern void USART_callback_Trans_Int(void (*Ptr_to_Func)(void))
{
Ptr_To_Trans_Int = Ptr_to_Func;
}
ISR(USART_TXC_vect)
{
        (*Ptr_To_Trans_Int)();
}
```

ENG. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/