



```
*****  
* convolve.c  
***** /
```

```
/* Standard includes */  
#include <assert.h>  
#include <math.h>  
#include <stdio.h> /* malloc(), realloc() */
```

```
/* Our includes */  
#include "base.h"  
#include "error.h"  
#include "convolve.h"  
#include "klt_util.h" /* printing */
```

```
#define MAX_KERNEL_WIDTH 71
```

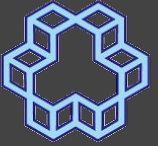
```
typedef struct {  
    int width;  
    float (*k)[MAX_KERNEL_WIDTH];  
} ConvolutionKernel;
```

```
/* Kernels */
```

Fundamentals of Programming

lecture 19

Pointers and Arrays, Pointer Arithmetic



1926

K. N. Toosi University of Technology

Array arguments are addresses

```
void bubbleSort(int a[], int n) {
    int m;

    for (m = n-1; m > 0; m--) {
        for (int i = 0; i < m; i++) {

            if (a[i] > a[i+1]) {
                swap(&a[i], &a[i+1]);
            }

        }
    }
}
```

pointersort.c

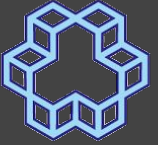
```
void bubbleSort(int *a, int n) {
    int m;

    for (m = n-1; m > 0; m--) {
        for (int i = 0; i < m; i++) {

            if (a[i] > a[i+1]) {
                swap(&a[i], &a[i+1]);
            }

        }
    }
}
```

pointersort2.c



1926

K. N. Toosi University of Technology

Array arguments vs Pointers

```
#include <stdio.h>

void printArray1(int *a, int n);
void printArray2(int a[], int n);

int main() {
    int a[] = {1,2,3,4,5,6};

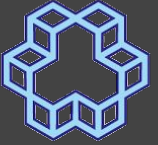
    printArray1(a,6);
    printArray2(a,6);

    return 0;
}

void printArray1(int a[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}

void printArray2(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

pointerarray1.c



1926

K. J. Somaiya Institute of Technology

Array arguments vs Pointers

```
#include <stdio.h>

void printArray1(int *a, int n);
void printArray2(int a[], int n);

int main() {
    int a[] = {1,2,3,4,5,6};

    printArray1(a,6);
    printArray2(a,6);

    return 0;
}

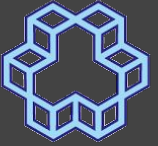
void printArray1(int a[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}

void printArray2(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

pointerarray1.c

```
nasihatkon@kntu:code$ gcc pointerarray1.c && ./a.out
1, 2, 3, 4, 5, 6,
1, 2, 3, 4, 5, 6,
nasihatkon@kntu:code$
```

The following are equivalent

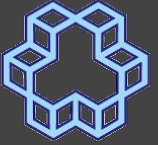


1926

K. J. Somaiya Institute of Technology

```
void printArray1(int a[], int n) {  
    for (int i = 0; i < n; i++)  
        printf("%d, ", a[i]);  
    putchar('\n');  
}
```

```
void printArray2(int *a, int n) {  
    for (int i = 0; i < n; i++)  
        printf("%d, ", a[i]);  
    putchar('\n');  
}
```



1926

K. N. Toosi University of Technology

Arrays vs. Pointers

```
#include <stdio.h>

void printArray(int *a, int n);

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

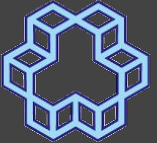
    p = a[0];

    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

pointerarray3.c



1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

```
#include <stdio.h>

void printArray(int *a, int n);

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a[0];

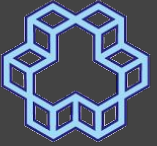
    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
```

pointerarray3.c

```
nasihatkon@kntu:code$ gcc pointerarray3.c && ./a.out
pointerarray3.c: In function 'main':
pointerarray3.c:9:5: warning: assignment makes pointer from integer without a cast [-Wint-conversion]
    p = a[0];
    ^
Segmentation fault (core dumped)
```



1926

K. N. Toosi University of Technology

Arrays vs. Pointers

```
#include <stdio.h> pointerarray4.c

void printArray(int *a, int n);

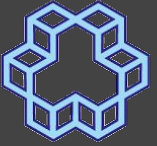
int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = &a[0];

    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

1926

K. N. Toosi University of Technology

Arrays vs. Pointers

```
#include <stdio.h>
pointerarray4.c

void printArray(int *a, int n);

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = &a[0];

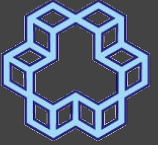
    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

```
nasihatkon@kntu:code$ gcc pointerarray4.c && ./a.out
1, 2, 3, 4,
nasihatkon@kntu:code$
```

Arrays vs. Pointers



1926

K. N. Toosi University of Technology

```
#include <stdio.h> pointerarray5.c

void printArray(int *a, int n);

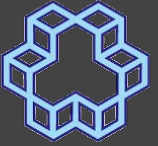
int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = &a[1];

    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```



1926

K. N. Toosi University of Technology

Arrays vs. Pointers

```
#include <stdio.h>
pointerarray5.c

void printArray(int *a, int n);

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

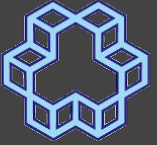
    p = &a[1];

    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

```
nasihatkon@kntu:code$ gcc pointerarray5.c && ./a.out
2, 3, 4, 5,
```



1926

K. N. Toosi University of Technology

Arrays vs. Pointers

```
#include <stdio.h> pointerarray4.c

void printArray(int *a, int n);

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = &a[0];

    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

```
#include <stdio.h> pointerarray6.c

void printArray(int *a, int n);

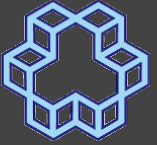
int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```



1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

```
#include <stdio.h> pointerarray4.c

void printArray(int *a, int n);

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = &a[0];

    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

```
#include <stdio.h> pointerarray6.c

void printArray(int *a, int n);

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

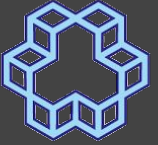
    printArray(p,4);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
```

```
nasihatkon@kntu:code$ gcc pointerarray6.c && ./a.out
1, 2, 3, 4,
```

Arrays vs. Pointers



1926

K. N. Toosi University of Technology

```
#include <stdio.h>
pointerarray7.c

void printArray(int *a, int n);

int main() {
    int n = 6;
    int a[] = {1,2,3,4,5,6};
    int *p;

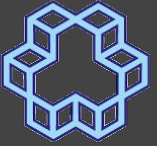
    p = a;

    p = &n;

    printArray(p,1);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```



1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

```
#include <stdio.h>
void printArray(int *a, int n);

int main() {
    int n = 6;
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    p = &n;

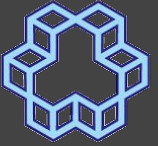
    printArray(p,1);

    return 0;
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

pointerarray7.c

```
nasihatkon@kntu:code$ gcc pointerarray7.c && ./a.out
6,
```



1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

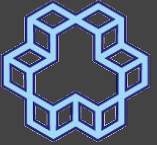
```
int main() { pointerarray8.c
    int n = 6;
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    p = &n;

    printf("%d\n", *p);
    printf("%d\n", p[0]);

    return 0;
}
```

1926

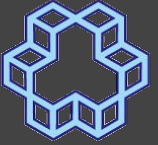
K. J. Somaiya Institute of Technology

Arrays vs. Pointers

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    p = &n;  
  
    printf("%d\n", *p);  
    printf("%d\n", p[0]);  
  
    return 0;  
}
```

pointerarray8.c

```
nasihatkon@kntu:code$ gcc pointerarray8.c && ./a.out  
6  
6
```



1926

K. J. Somaiya Institute of Technology

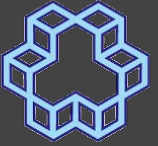
Arrays vs. Pointers

```
pointerarray9.c
int main() {
    int n = 6;
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printf("%d\n", *p);
    printf("%d\n", p[0]);

    return 0;
}
```



1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

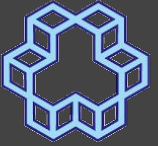
```
pointerarray9.c
int main() {
    int n = 6;
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printf("%d\n", *p);
    printf("%d\n", p[0]);

    return 0;
}
```

```
nasihatkon@kntu:code$ gcc pointerarray9.c && ./a.out
1
1
```



1926

K. N. Toosi University of Technology

Arrays vs. Pointers

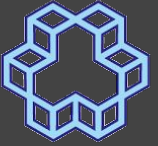
```
int main() {
    int n = 6;
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printf("%d\n", *a);
    printf("%d\n", a[0]);

    return 0;
}
```

pointerarray10.c



1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

```
int main() {
    int n = 6;
    int a[] = {1,2,3,4,5,6};
    int *p;

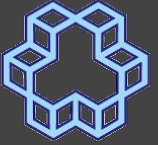
    p = a;

    printf("%d\n", *a);
    printf("%d\n", a[0]);

    return 0;
}
```

pointerarray10.c

```
nasihatkon@kntu:code$ gcc pointerarray10.c && ./a.out
1
1
```



1926

K. J. Somaiya Institute of Technology

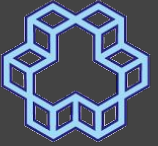
Arrays vs. Pointers

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    printf("%d\n", *a);  
    printf("%d\n", a[0]);  
  
    return 0;  
}
```

pointerarray10.c

pointers and arrays are the same right?

```
nasihatkon@kntu:code$ gcc pointerarray10.c && ./a.out  
1  
1
```



1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

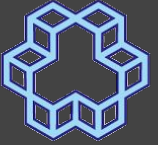
```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    printf("%d\n", *a);  
    printf("%d\n", a[0]);  
  
    return 0;  
}
```

pointerarray10.c

pointers and arrays are the same right?
Well, not quite!

```
nasihatkon@kntu:code$ gcc pointerarray10.c && ./a.out  
1  
1
```

Arrays vs. Pointers

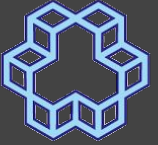


1926

K. J. Somaiya Institute of Technology

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    p = &n;  
    a = &n;  
  
    return 0;  
}
```

pointerarray11.c



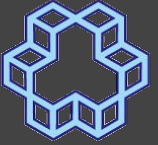
1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

pointerarray11.c

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    p = &n;  
    a = &n; ←  
  
    return 0;  
}
```



1926

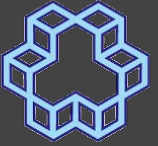
K. J. Somaiya Institute of Technology

Arrays vs. Pointers

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    p = &n;  
    a = &n; ←  
  
    return 0;  
}
```

pointerarray11.c

```
nasihatkon@kntu:code$ gcc pointerarray11.c && ./a.out  
pointerarray11.c: In function 'main':  
pointerarray11.c:13:5: error: assignment to expression with array type  
    a = &n;  
    ^
```



1926

K. J. Somaiya Institute of Technology

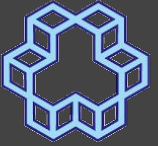
Arrays vs. Pointers

pointerarray11.c

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    p = &n;  
    a = &n; ←  
  
    return 0;  
}
```

somehow, similar to `int * const p;`

```
nasihatkon@kntu:code$ gcc pointerarray11.c && ./a.out  
pointerarray11.c: In function 'main':  
pointerarray11.c:13:5: error: assignment to expression with array type  
    a = &n;  
      ^
```



1926

K. J. Somaiya Institute of Technology

Arrays vs. Pointers

```
int main() {
    int n = 6;
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

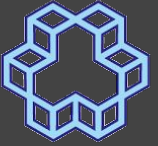
    p = &n;
    a = &n;

    return 0;
}
```

pointerarray11.c

somehow, similar to `int * const p;`
but not the same!

```
nasihatkon@kntu:code$ gcc pointerarray11.c && ./a.out
pointerarray11.c: In function 'main':
pointerarray11.c:13:5: error: assignment to expression with array type
    a = &n;
    ^
```



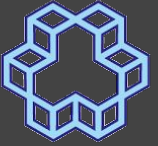
1926

K. N. Toosi University of Technology

Arrays vs. Pointers: size

pointerarray12.c

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    printf("%lu\n", sizeof(a));  
    printf("%lu\n", sizeof(p));  
  
    return 0;  
}
```



1926

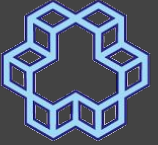
K. J. Somaiya Institute of Technology

Arrays vs. Pointers: size

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    printf("%lu\n", sizeof(a));  
    printf("%lu\n", sizeof(p));  
  
    return 0;  
}
```

pointerarray12.c

```
nasihatkon@kntu:code$ gcc pointerarray12.c && ./a.out  
24  
8  
nasihatkon@kntu:code$
```



1926

K. J. Somaiya Institute of Technology

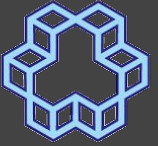
Arrays vs. Pointers: size

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    printf("%lu\n", sizeof(a));  
    printf("%lu\n", sizeof(p));  
  
    return 0;  
}
```

pointerarray12.c

Addressed are stored in 8 bytes in
64-bit systems

```
nasihatkon@kntu:code$ gcc pointerarray12.c && ./a.out  
24  
8  
nasihatkon@kntu:code$
```



1926

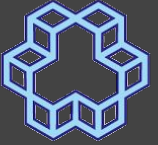
K. J. Somaiya Institute of Technology

Arrays vs. Pointers: size

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    printf("%lu\n", sizeof(a));  
    printf("%lu\n", sizeof(p));  
  
    return 0;  
}
```

pointerarray12.c

```
nasihatkon@kntu:code$ gcc -m32 pointerarray13.c && ./a.out  
24  
4
```

1926

K. J. Somaiya Institute of Technology

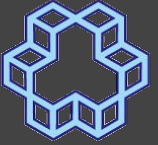
Arrays vs. Pointers: size

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    p = a;  
  
    printf("%lu\n", sizeof(a));  
    printf("%lu\n", sizeof(p));  
  
    return 0;  
}
```

pointerarray12.c

Addressed are stored in 4 bytes in
32-bit systems

```
nasihatkon@kntu:code$ gcc -m32 pointerarray13.c && ./a.out  
24  
4
```



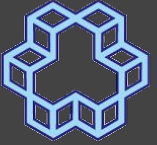
1926

K. N. Toosi University of Technology

Array arguments: size

pointerarray14.c

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    printf("sizeof(a)= %lu\n", sizeof(a));  
  
    printArray(a,6);  
  
    return 0;  
}  
  
void printArray(int a[], int n) {  
    printf("sizeof(a)= %lu\n", sizeof(a));  
  
    for (int i = 0; i < n; i++)  
        printf("%d, ", a[i]);  
    putchar('\n');  
}
```



1926

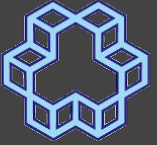
K. J. Somaiya Institute of Technology

Array arguments: size

```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    printf("sizeof(a)= %lu\n", sizeof(a));  
  
    printArray(a,6);  
  
    return 0;  
}  
  
void printArray(int a[], int n) {  
    printf("sizeof(a)= %lu\n", sizeof(a));  
  
    for (int i = 0; i < n; i++)  
        printf("%d, ", a[i]);  
    putchar('\n');  
}
```

pointerarray14.c

```
nasihatkon@kntu:code$ gcc pointerarray14.c && ./a.out  
pointerarray14.c: In function 'printArray':  
pointerarray14.c:20:36: warning: 'sizeof' on array function parameter  
'a' will return size of 'int *' [-Wsizeof-array-argument]  
    printf("sizeof(a)= %lu\n", sizeof(a));  
                                ^  
pointerarray14.c:19:21: note: declared here  
    void printArray(int a[], int n) {  
                        ^  
sizeof(a)= 24  
sizeof(a)= 8  
1, 2, 3, 4, 5, 6,  
nasihatkon@kntu:code$
```



1926

K. J. Somaiya Institute of Technology

Array arguments: size

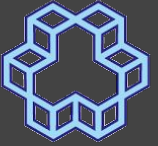
```
int main() {  
    int n = 6;  
    int a[] = {1,2,3,4,5,6};  
    int *p;  
  
    printf("sizeof(a)= %lu\n", sizeof(a));  
  
    printArray(a,6);  
  
    return 0;  
}  
  
void printArray(int a[], int n) {  
    printf("sizeof(a)= %lu\n", sizeof(a));  
  
    for (int i = 0; i < n; i++)  
        printf("%d, ", a[i]);  
    putchar('\n');  
}
```

pointerarray14.c

```
nasihatkon@kntu:code$ gcc pointerarray14.c && ./a.out  
pointerarray14.c: In function 'printArray':  
pointerarray14.c:20:36: warning: 'sizeof' on array function parameter  
'a' will return size of 'int *' [-Wsizeof-array-argument]  
    printf("sizeof(a)= %lu\n", sizeof(a));  
                                ^  
pointerarray14.c:19:21: note: declared here  
    void printArray(int a[], int n) {  
                        ^  
sizeof(a)= 24  
sizeof(a)= 8  
1, 2, 3, 4, 5, 6,  
nasihatkon@kntu:code$
```

Array arguments are really pointers!

Pointer Arithmetic



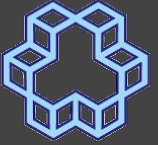
1926

K. J. Somaiya Institute of Technology

```
#include <stdio.h>
```

pointerarithmetic1.c

```
int main() {  
    int n = 6;  
    int *p;  
  
    p = &n;  
  
    printf("Address0= %p\n", p);  
    printf("Address1= %p\n", p+1);  
    printf("Address2= %p\n", p+2);  
  
    return 0;  
}
```



1926

K. J. Somaiya Institute of Technology

Pointer Arithmetic

```
#include <stdio.h>

int main() {
    int n = 6;
    int *p;

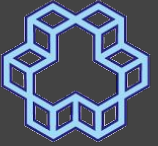
    p = &n;

    printf("Address0= %p\n", p);
    printf("Address1= %p\n", p+1);
    printf("Address2= %p\n", p+2);

    return 0;
}
```

pointerarithmetic1.c

```
nasihatkon@kntu:code$ gcc pointerarithmetic1.c && ./a.out
Address0= 0x7ffda15ea93c
Address1= 0x7ffda15ea940
Address2= 0x7ffda15ea944
```



1926

K. J. Somaiya Institute of Technology

Pointer Arithmetic

```
#include <stdio.h>

int main() {
    int n = 6;
    int *p;

    p = &n;

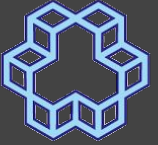
    printf("Address0= %p\n", p);
    printf("Address1= %p\n", p+1);
    printf("Address2= %p\n", p+2);

    return 0;
}
```

pointerarithmetic1.c

Why?

```
nasihatkon@kntu:code$ gcc pointerarithmetic1.c && ./a.out
Address0= 0x7ffda15ea93c
Address1= 0x7ffda15ea940
Address2= 0x7ffda15ea944
```

1926

K. N. Toosi University of Technology

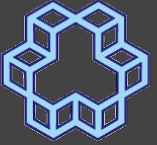
Pointer arithmetic: pointer + integer

```
#include <stdio.h> pointerarithmetic2.c

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printf("%d\n", *p);
    printf("%d\n", *(p+1));
    printf("%d\n", *(p+2));
}
```

1926

K. N. Toosi University of Technology

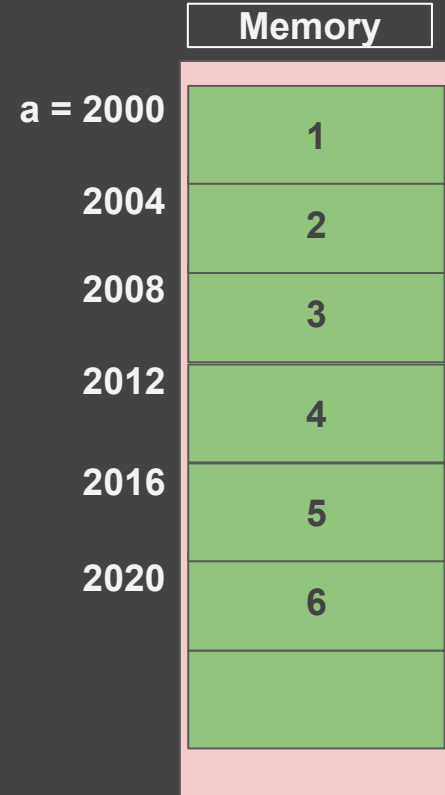
Pointer arithmetic: pointer + integer

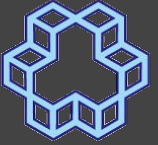
```
#include <stdio.h> pointerarithmetic2.c

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printf("%d\n", *p);
    printf("%d\n", *(p+1));
    printf("%d\n", *(p+2));
}
```





1926

K. N. Toor University of Technology

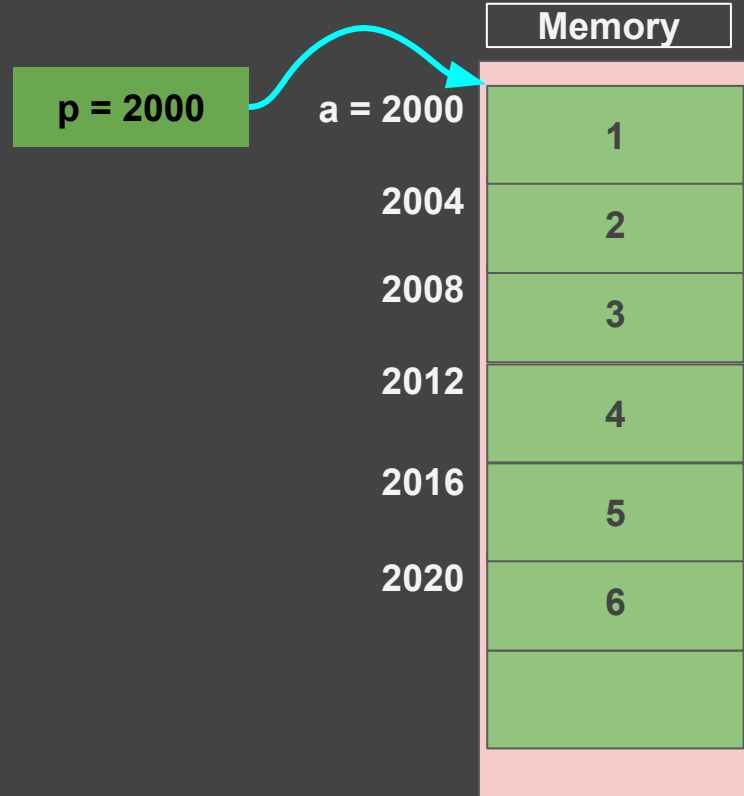
Pointer arithmetic: pointer + integer

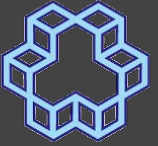
```
#include <stdio.h> pointerarithmetic2.c

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printf("%d\n", *p);
    printf("%d\n", *(p+1));
    printf("%d\n", *(p+2));
}
```





1926

K. N. Toosi University of Technology

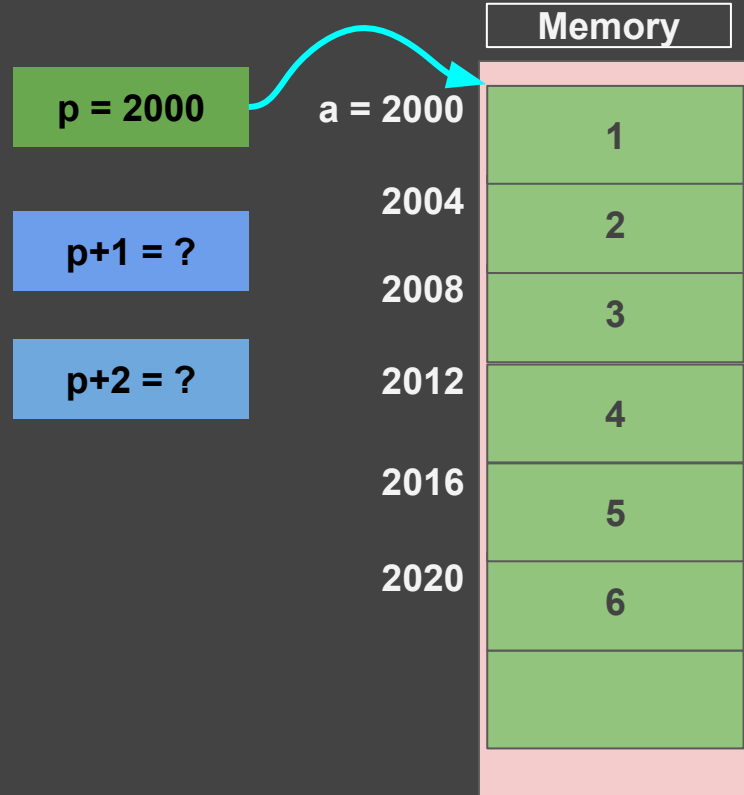
Pointer arithmetic: pointer + integer

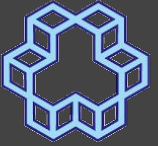
```
#include <stdio.h> pointerarithmetic2.c

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printf("%d\n", *p);
    printf("%d\n", *(p+1));
    printf("%d\n", *(p+2));
}
```





1926

K. N. Toosi University of Technology

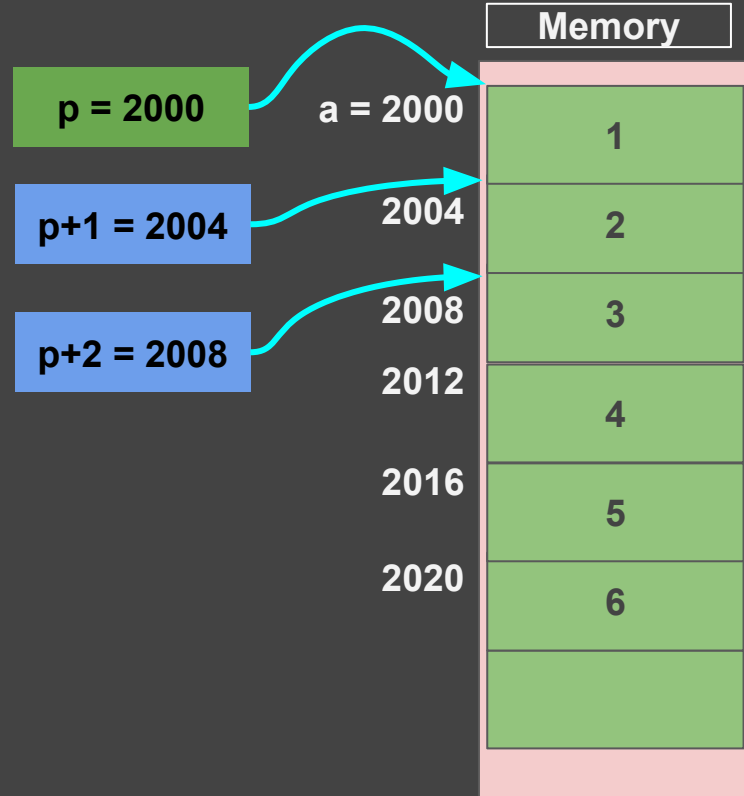
Pointer arithmetic: pointer + integer

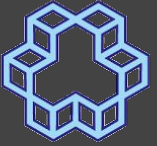
```
#include <stdio.h> pointerarithmetic2.c

int main() {
    int a[] = {1,2,3,4,5,6};
    int *p;

    p = a;

    printf("%d\n", *p);
    printf("%d\n", *(p+1));
    printf("%d\n", *(p+2));
}
```





1926

K. J. Somaiya Institute of Technology

Pointer arithmetic: pointer + integer

```
#include <stdio.h> pointerarithmetic2.c
```

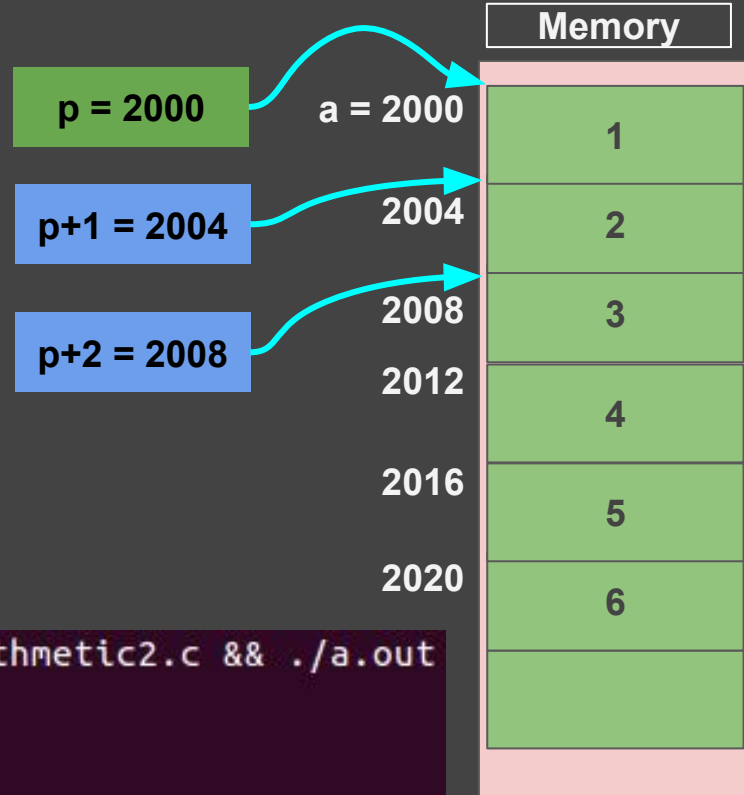
```
int main() {  
    int a[] = {1,2,3,4,5,6};  
    int *p;
```

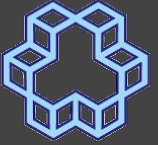
```
    p = a;
```

```
    printf("%d\n", *p);  
    printf("%d\n", *(p+1));  
    printf("%d\n", *(p+2));
```

```
}  
nasihatkon@kntu:code$ gcc pointerarithmetic2.c && ./a.out
```

```
1  
2  
3
```





1926

K. J. Somaiya Institute of Technology

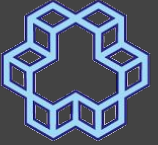
The precedence table

*p+i

*(p+i)

Category	Operator	Associativity
Postfix	() [] -> . ++ --	Left to right
Unary	+ - ! ~ ++ -- (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= =	Right to left
Comma	,	Left to right

pointer definition



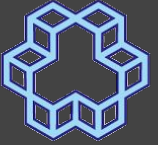
1926

K. N. Toosi University of Technology

```
char c; pointerarithmetic3.c  
char *cp;  
  
short int s;  
short int *sp;  
  
int i;  
int *ip;  
  
long int l;  
long int *lp;
```

```
char c, *cp; pointerarithmetic4.c  
  
short int s, *sp;  
  
int i, *ip;  
  
long int l, *lp;
```

pointer definition



1926

K. N. Toosi University of Technology

```
char      c, *cp;
short int s, *sp;
int       i, *ip;
long int  l, *lp;

cp = &c;
sp = &s;
ip = &i;
lp = &l;
```

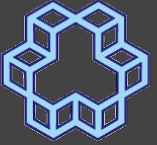
pointerarithmetic4.c

```
printf("cp=%p\n" , cp);
cp++;
printf("cp=%p\n\n", cp);

printf("sp=%p\n" , sp);
sp++;
printf("sp=%p\n\n", sp);

printf("ip=%p\n" , ip);
ip++;
printf("ip=%p\n\n", ip);

printf("lp=%p\n" , lp);
lp++;
printf("lp=%p\n\n", lp);
```

1926

K. J. Somaiya Institute of Technology

pointer definition

```
printf("cp=%p\n" , cp);  
cp++;  
printf("cp=%p\n\n", cp);  
  
printf("sp=%p\n" , sp);  
sp++;  
printf("sp=%p\n\n", sp);  
  
printf("ip=%p\n" , ip);  
ip++;  
printf("ip=%p\n\n", ip);  
  
printf("lp=%p\n" , lp);  
lp++;  
printf("lp=%p\n\n", lp);
```

```
nasihatkon@kntu:code$ gcc pointerarithmetic4.c && ./a.out  
cp=0x7ffce23085c9  
cp=0x7ffce23085ca  
  
sp=0x7ffce23085ca  
sp=0x7ffce23085cc  
  
ip=0x7ffce23085cc  
ip=0x7ffce23085d0  
  
lp=0x7ffce23085d0  
lp=0x7ffce23085d8
```

pointerarithmetic4.c



1926

K. J. Somaiya Institute of Technology

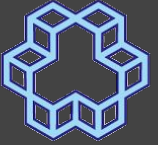
pointer definition

```
printf("cp=%p\n" , cp);  
cp++;  
printf("cp=%p\n\n", cp);  
  
printf("sp=%p\n" , sp);  
sp++;  
printf("sp=%p\n\n", sp);  
  
printf("ip=%p\n" , ip);  
ip++;  
printf("ip=%p\n\n", ip);  
  
printf("lp=%p\n" , lp);  
lp++;  
printf("lp=%p\n\n", lp);
```

```
nasihatkon@kntu:code$ gcc pointerarithmetic4.c && ./a.out  
cp=0x7ffce23085c9  
cp=0x7ffce23085ca  
  
sp=0x7ffce23085ca  
sp=0x7ffce23085cc  
  
ip=0x7ffce23085cc  
ip=0x7ffce23085d0  
  
lp=0x7ffce23085d0  
lp=0x7ffce23085d8
```

pointerarithmetic4.c

pointer arithmetic & arrays



1926

K. N. Toosi University of Technology

```
#include <stdio.h>

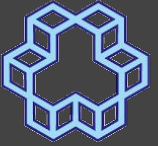
void printArray(int *a, int n);

int main() {
    int a[] = {1,2,3,4,5,6};

    printArray(a, 6);
}

void printArray(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d, ", a[i]);
    putchar('\n');
}
```

pointerarithmetic5.c



1926

K. J. Somaiya Institute of Technology

pointer arithmetic & arrays

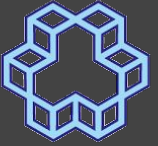
pointerarithmetic5.c

```
void printArray(int *a, int n) {  
    for (int i = 0; i < n; i++)  
        printf("%d, ", a[i]);  
    putchar('\n');  
}
```

pointerarithmetic6.c

```
void printArray(int *a, int n) {  
    for (int i = 0; i < n; i++)  
        printf("%d, ", *(a+i));  
    putchar('\n');  
}
```

$a[0] \equiv *a$



1926

K. J. Somaiya Institute of Technology

pointer arithmetic & arrays

pointerarithmetic5.c

```
void printArray(int *a, int n) {  
    for (int i = 0; i < n; i++)  
        printf("%d, ", a[i]);  
    putchar('\n');  
}
```

pointerarithmetic6.c

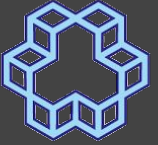
```
void printArray(int *a, int n) {  
    for (int i = 0; i < n; i++)  
        printf("%d, ", *(a+i));  
    putchar('\n');  
}
```

$a[0] \equiv *a$

$a[i] \equiv *(a+i)$

$\& a[i] \equiv a+i$

pointer arithmetic & arrays



1926

K. J. Somaiya Institute of Technology

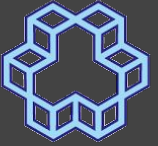
pointerarithmetic6.c

```
void printArray(int *a, int n) {  
    for (int i = 0; i < n; i++)  
        printf("%d, ", *(a+i));  
    putchar('\n');  
}
```

pointerarithmetic7.c

```
void printArray(int *a, int n) {  
    for (int i = 0; i < n; i++, a++)  
        printf("%d, ", *a);  
    putchar('\n');  
}
```

pointer arithmetic & arrays



1926

K. N. Toosi University of Technology

pointerarithmetic7.c

```
void printArray(int *a, int n) {  
    for (int i = 0; i < n; i++, a++)  
        printf("%d, ", *a);  
    putchar('\n');  
}
```

pointerarithmetic8.c

```
void printArray(int *a, int n) {  
    int *endp = a+n;  
  
    while (a < endp)  
        printf("%d, ", *a++);  
  
    putchar('\n');  
}
```



1926

K. J. Somaiya Institute of Technology

Category	Operator	Associativity
Postfix	() [] -> . ++ --	Left to right
Unary	+ - ! ~ ++ -- (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= =	Right to left
Comma	,	Left to right

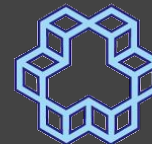
pointerarithmetic8.c

```
void printArray(int *a, int n) {
    int *endp = a+n;

    while (a < endp)
        printf("%d, ", *a++);

    putchar('\n');
}
```


Exercise



1926

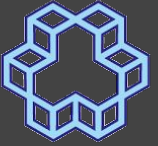
K. N. Toosi University of Technology

```
#include <stdio.h>
```

pointerarithmetic10.c

```
int main() {  
    char s[] = "Salam! Chetori?";  
    char t[100];  
    char *p, *q;  
  
    for (p = s, q = t; *p != '\0'; p++, q++)  
        ;  
  
    *q-- = '\0'; // *q = '\0'; q--;  
  
    for (p = s; *p != '\0'; p++, q--)  
        *q = *p;  
  
    puts(s);  
    puts(t);  
  
    return 0;  
}
```

Exercise



1926

K. N. Toosi University of Technology

```
#include <stdio.h> pointerarithmetic11.c

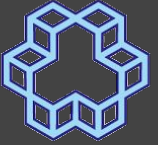
void mystrcpy(char *t, char *s) {
    while ( (*t++ = *s++) != '\0' )
        ;
}

int main() {
    char s[] = "Salam! Chetori?";
    char t[100];

    mystrcpy(t,s);

    puts(s);
    puts(t);

    return 0;
}
```



1926

K. J. Somaiya Institute of Technology

Subtracting two pointers

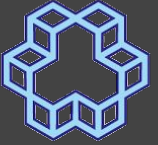
```
int a[10];
```

```
int *p = a;
```

```
int *q = &a[6];
```

```
int i = q - p;
```

what is the value of i?



1926

K. J. Somaiya Institute of Technology

Subtracting two pointers

```
int a[10];
```

```
int *p = a;
```

```
int *q = &a[6];
```

```
int i = q - p;
```

what is the value of i? 6