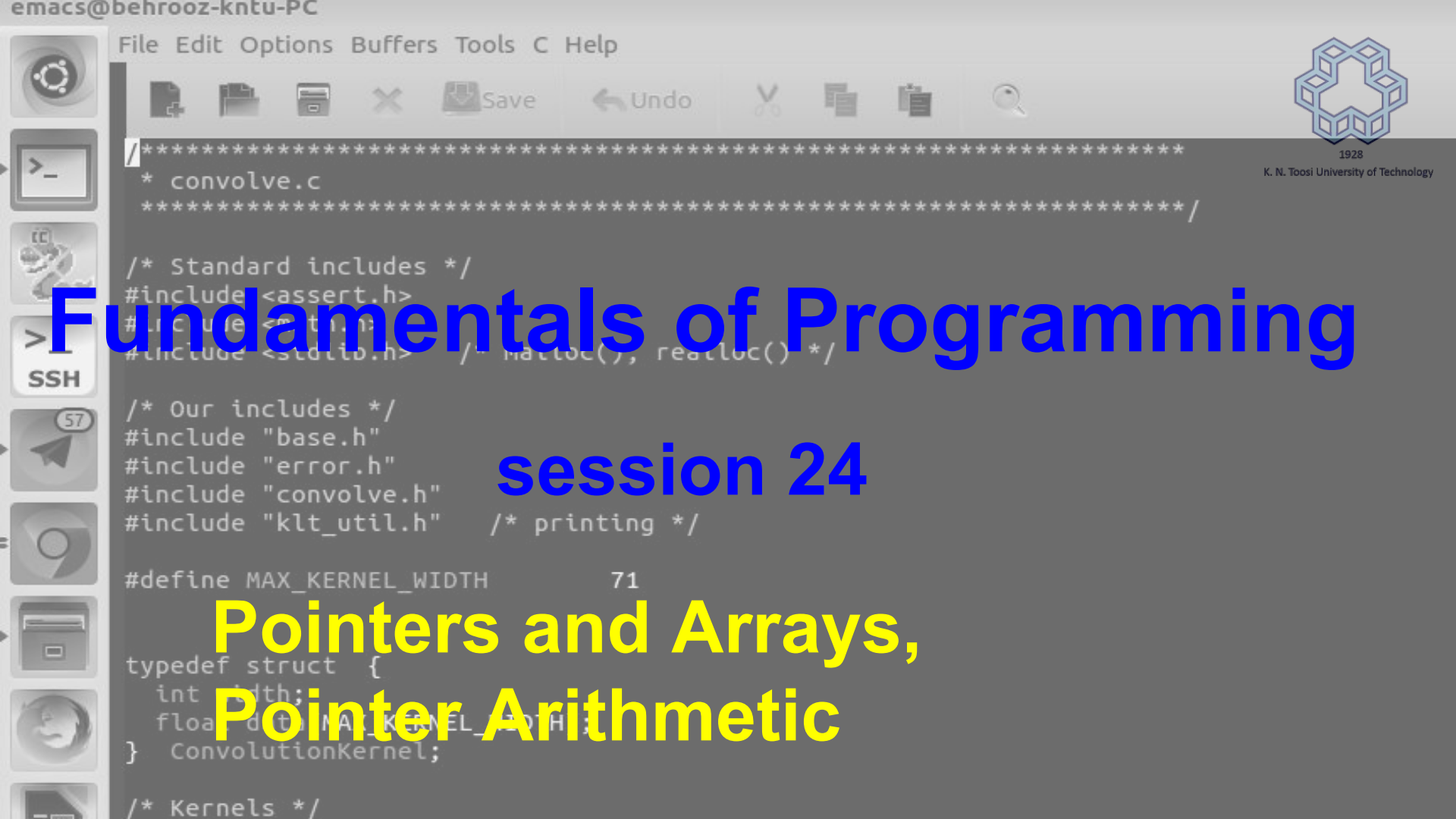
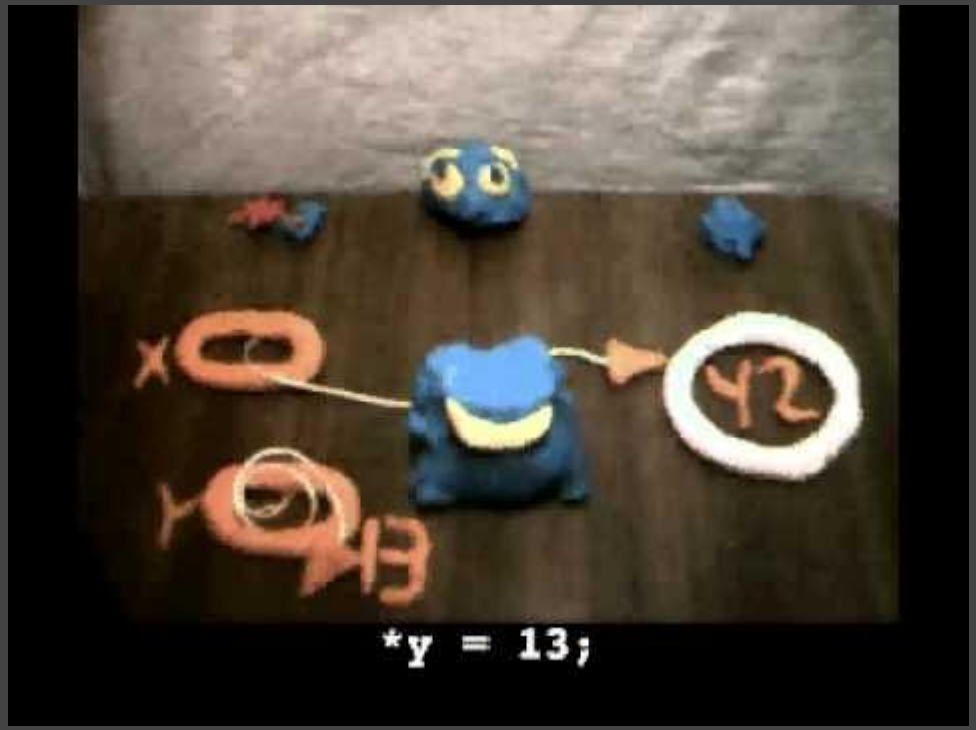
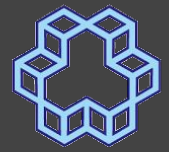

emacs@behrooz-kntu-PC
File Edit Options Buffers Tools C Help
Save
Undo
1928
K. N. Toosi University of Technology
/***
* convolve.c
***/
/* Standard includes */
#include <assert.h>
#include <math.h>
#include <stdlib.h> /* malloc(), realloc() */
/* Our includes */
#include "base.h"
#include "error.h"
#include "convolve.h"
#include "klt_util.h" /* printing */
#define MAX_KERNEL_WIDTH 71
typedef struct {
int width;
float data[MAX_KERNEL_WIDTH];
} ConvolutionKernel;
/* Kernels */
SSH
57
Fundamentals of Programming
session 24
Pointers and Arrays,
Pointer Arithmetic

Pointers

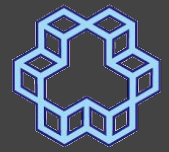


Pointers



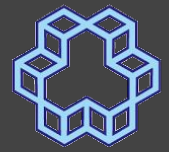

```
a = *(&b);

p = &(*q);
```

Remember: the "swap" function




```c
void swap(int *p, int *q) {
  int temp;
  temp = *p;
  *p = *q;
  *q = temp;
}
```

Remember: bubble sort

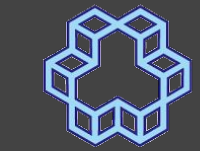



```c
int main() {
  int a[] = {25,19,14,18,27,6,32,18,20,1,21};
  int n = sizeof(a) / sizeof(a[0]);

  printArray(a,n);

  bubbleSort(a,n);

  printArray(a,n);

  return 0;
}
```

```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]) {
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
      }


    }
  }
}
```

```
nasihatkon@kntu:code$ gcc pointersort.c && ./a.out
25, 19, 14, 18, 27, 6, 32, 18, 20, 1, 21,
1, 6, 14, 18, 18, 19, 20, 21, 25, 27, 32,
nasihatkon@kntu:code$
```

Using swap for bubble sort



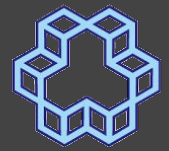

```c
void bubbleSort(int a[], int n) {
  int m;

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

      if (a[i] > a[i+1]) {
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
      }


    }
  }
}
```

```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]);
      }


    }
  }
}
```

pointersort.c

```
nasihatkon@kntu:code$ gcc pointersort.c && ./a.out
25, 19, 14, 18, 27, 6, 32, 18, 20, 1, 21,
1, 6, 14, 18, 18, 19, 20, 21, 25, 27, 32,
nasihatkon@kntu:code$
```

Array arguments are addresses



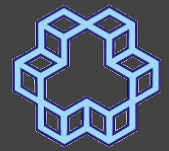

```c
void bubbleSort(int a[], int n) {
  int m;

  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]);
      }


    }
  }
}
```

pointersort.c

```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

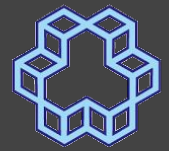
Array arguments vs Pointers



pointerarray1.c

```c
#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');
}
```

Array arguments vs Pointers

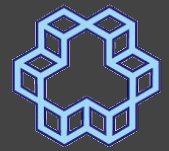


pointerarray1.c

```c
#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');
}
```

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

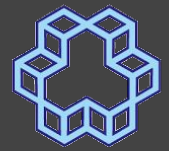
The following are equivalent




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

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

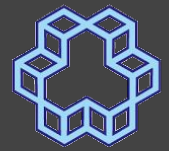
Arrays vs. Pointers




```c
#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

Arrays vs. Pointers

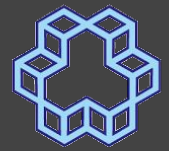


pointerarray3.c

```c
#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) {
```

```
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)
```

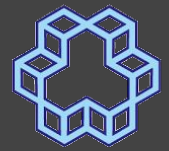
Arrays vs. Pointers




```c
#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');
}
```

pointerarray4.c

Arrays vs. Pointers

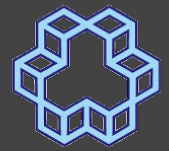


pointerarray4.c

```c
#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');
}
```

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

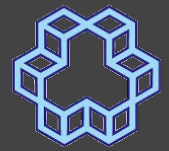
Arrays vs. Pointers




```c
#include <stdio.h>

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');
}
```

pointerarray5.c

Arrays vs. Pointers

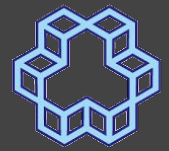


pointerarray5.c

```c
#include <stdio.h>

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,
```

Arrays vs. Pointers



pointerarray4.c

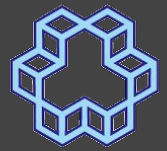
```c
#include <stdio.h>

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');
}
```

pointerarray6.c

```c
#include <stdio.h>

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');
}
```

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

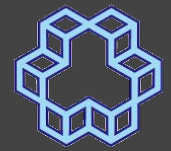
Arrays vs. Pointers



pointerarray7.c

```c
#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');
}
```

Arrays vs. Pointers

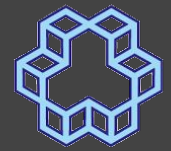


pointerarray7.c

```c
#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');
}
```

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

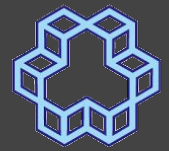
Arrays vs. Pointers



pointerarray8.c

```c
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;
}
```

Arrays vs. Pointers

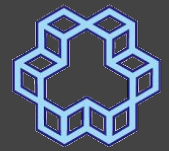


pointerarray8.c

```c
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;
}
```

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

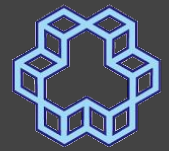
Arrays vs. Pointers



pointerarray9.c

```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;
}
```

Arrays vs. Pointers

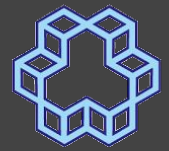


pointerarray9.c

```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
```

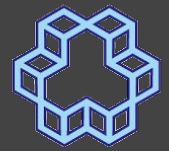
Arrays vs. Pointers



pointerarray10.c

```c
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;
}
```

Arrays vs. Pointers

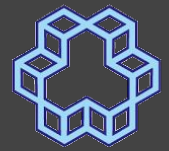


pointerarray10.c

```c
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;
}
```

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

Arrays vs. Pointers



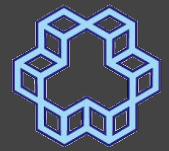
pointerarray10.c

```c
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;
}
```

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

pointers and arrays are the same right?

Arrays vs. Pointers



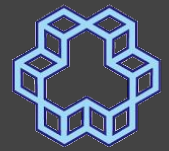
pointerarray10.c

```c
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;
}
```

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

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

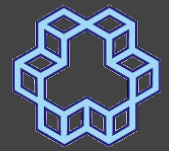
Arrays vs. Pointers



pointerarray11.c

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

  p = a;

  p = &n;
  a = &n;

  return 0;
}
```

Arrays vs. Pointers

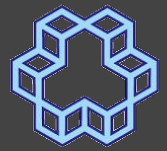


pointerarray11.c

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

  p = a;

  p = &n;
  a = &n;

  return 0;
}
```

```
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;
     ^
```

Arrays vs. Pointers



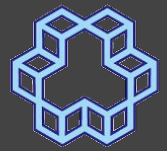
pointerarray11.c

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

  p = a;

  p = &n;
  a = &n;

  return 0;
}
```

```
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;
     ^
```

somehow, similar to `int * const p;`

Arrays vs. Pointers



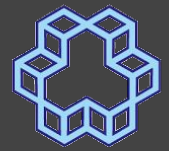
pointerarray11.c

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

  p = a;

  p = &n;
  a = &n;

  return 0;
}
```

```
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;
     ^
```

somehow, similar to `int * const p;`

Arrays vs. Pointers



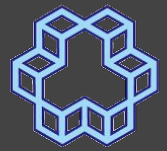
pointerarray11.c

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

  p = a;

  p = &n;
  a = &n;

  return 0;
}
```

```
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;
     ^
```

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

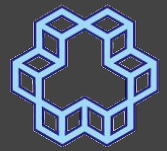
Arrays vs. Pointers: size



pointerarray12.c

```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;
}
```

Arrays vs. Pointers: size



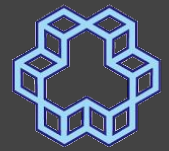
pointerarray12.c

```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;
}
```

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

Addressed are stored in 8 bytes in 64-bit systems

Arrays vs. Pointers: size 32 bit systems



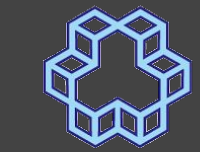
pointerarray13.c

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

  p = a;

  printf("%u\n", sizeof(a));
  printf("%u\n", sizeof(p));

  return 0;
}
```

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

Addressed are stored in 4 bytes in 32-bit systems

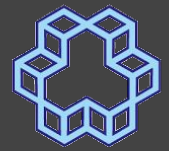
Array arguments: size



pointerarray14.c

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

  p = a;

  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');
}
```

Array arguments: size

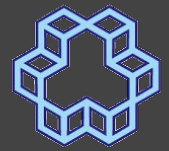


pointerarray14.c

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

  p = a;

  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');
}
```

```
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: size



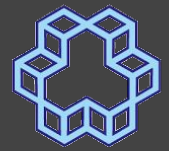
pointerarray14.c

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

  p = a;

  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');
}
```

```
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$
```

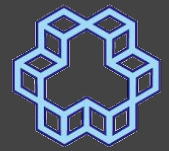
Array arguments are really pointers!

Pointer arithmetic




```c
#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;
}
```

Pointer arithmetic: pointer + integer

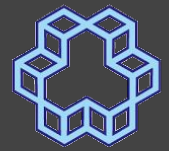


pointerarithmetic1.c

```c
#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;
}
```

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

Pointer arithmetic: pointer + integer



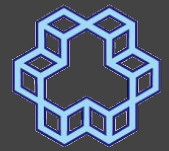
pointerarithmetic1.c

```c
#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;
}
```

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

Why?

Pointer arithmetic: pointer + integer

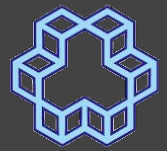


Why?

pointerarithmetic2.c

```c
#include <stdio.h>

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));

}
```

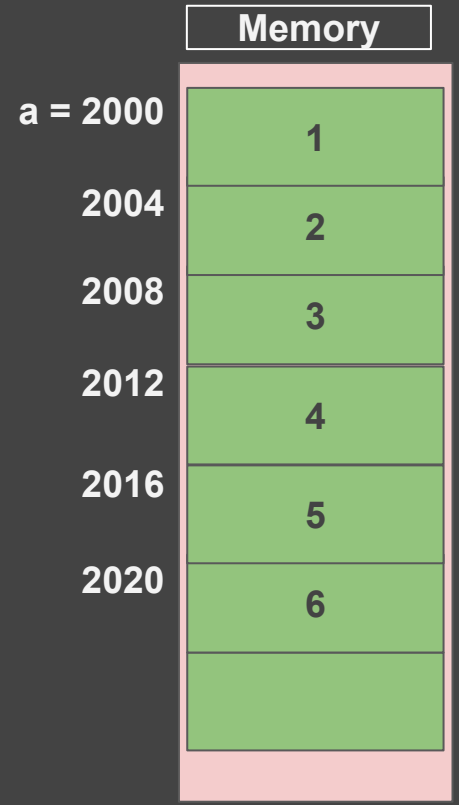
Pointer arithmetic: pointer + integer



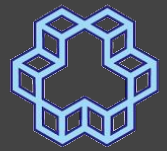
Why?

pointerarithmetic2.c

```c
#include <stdio.h>

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));

}
```



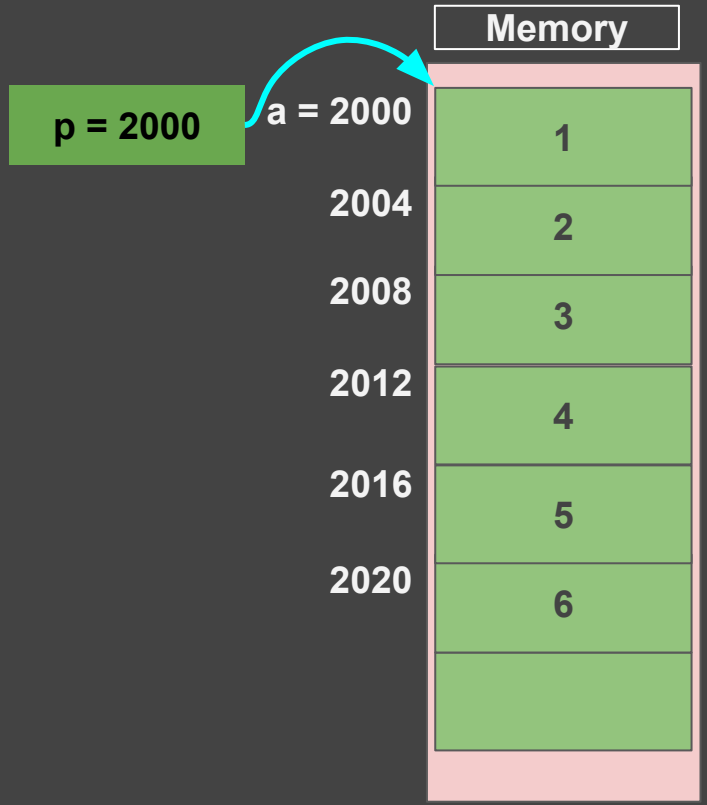
Pointer arithmetic: pointer + integer



Why?

```c
#include <stdio.h>

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));

}
```

pointerarithmetic2.c



Pointer arithmetic: pointer + integer

Why?

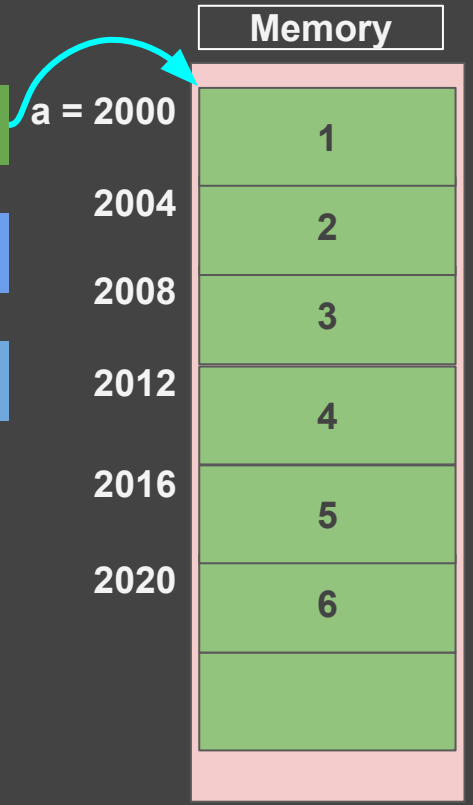
```c
#include <stdio.h>

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));

}
```

pointerarithmetic2.c

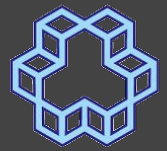
p = 2000

p+1 = ?

p+2 = ?



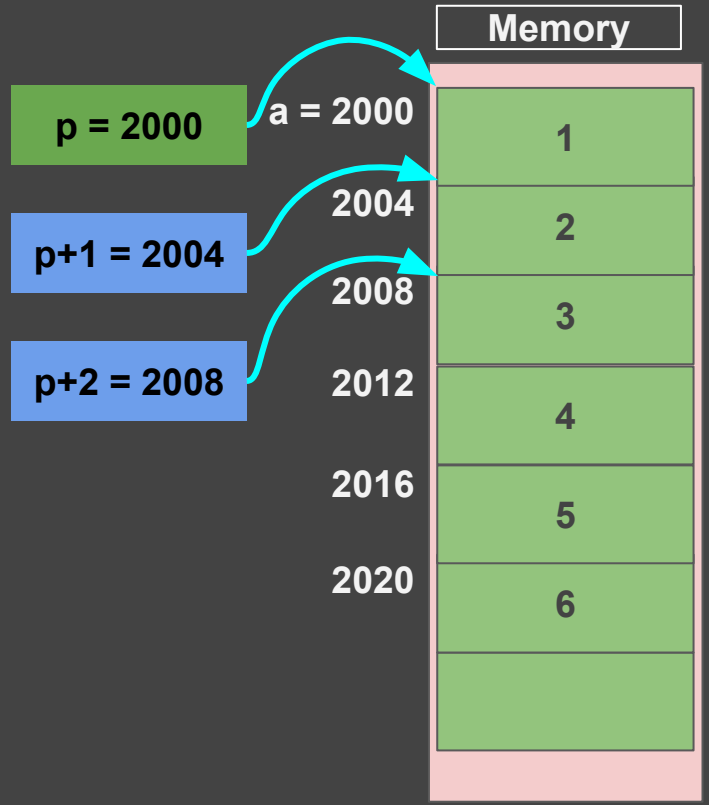
Pointer arithmetic: pointer + integer



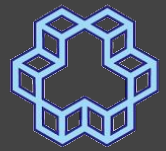
Why?

```c
#include <stdio.h>

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));

}
```

pointerarithmetic2.c



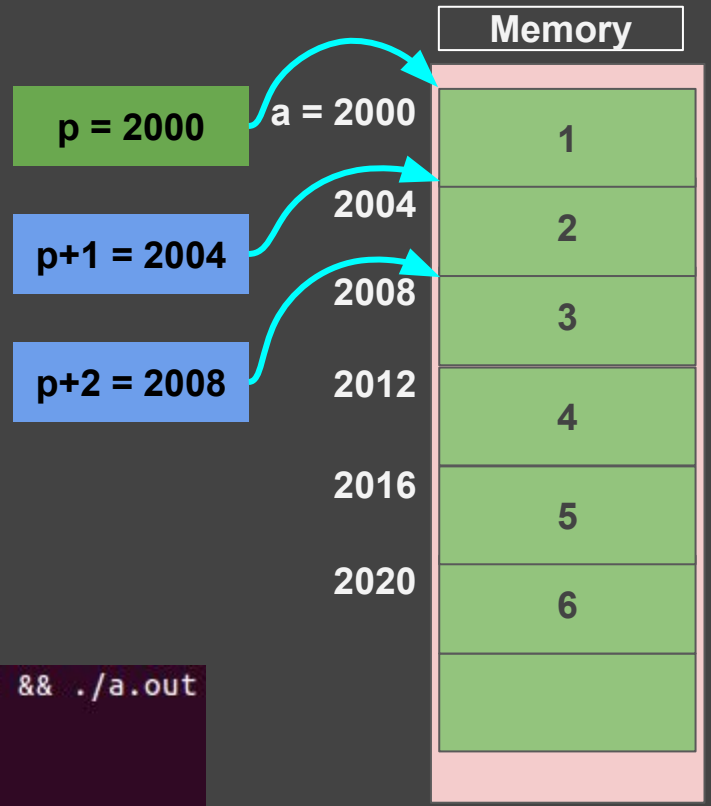
Pointer arithmetic: pointer + integer



Why?

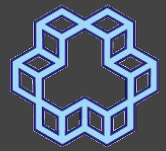
```c
#include <stdio.h>

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));

}
```

pointerarithmetic2.c




```
nasihatkon@kntu:code$ gcc pointerarithmetic2.c && ./a.out
1
2
3
```

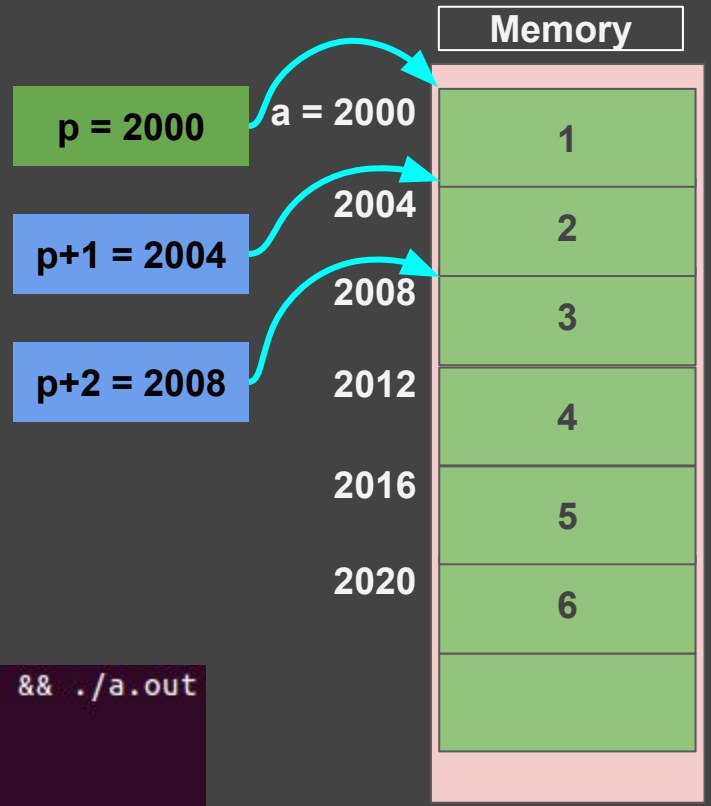
Pointer arithmetic: pointer + integer



Why?

```c
#include <stdio.h>

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));

}
```

pointerarithmetic2.c




```
nasihatkon@kntu:code$ gcc pointerarithmetic2.c && ./a.out
1
2
3
```

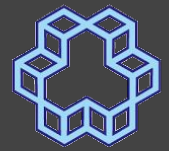
The precedence table

```
*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

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm

pointer definition



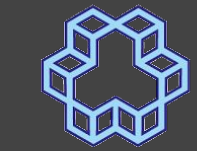

```c
char c;
char *cp;

short int s;
short int *sp;

int i;
int *ip;

long int l;
long int *lp;
```

pointerarithmetic3.c

```c
char c, *cp;

short int s, *sp;

int i, *ip;

long int l, *lp;
```

pointerarithmetic4.c

pointer definition

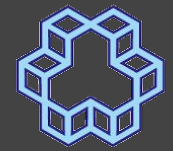



```
char        c, *cp;
short int   s, *sp;
int         i, *ip;
long int    l, *lp;

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);
```

pointer definition

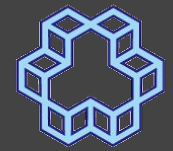



```
printf("cp=%p\n"  , cp);
cp++;
printf("cp=%p\n\n", cp);

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 definition

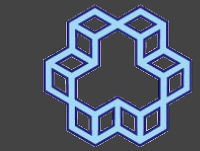



```
printf("cp=%p\n"  , cp);
cp++;
printf("cp=%p\n\n", cp);

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

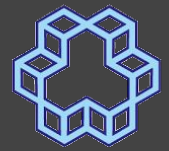
pointer arithmetic & arrays




```c
#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

pointer arithmetic & arrays



pointerarithmetic5.c

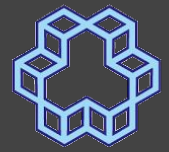
```c
void printArray(int *a, int n) {
  for (int i = 0; i < n; i++)
    printf("%d, ", a[i]);
  putchar('\n');
}
```

pointerarithmetic6.c

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

a[0] ≡ *a

pointer arithmetic & arrays



pointerarithmetic5.c

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

pointerarithmetic6.c

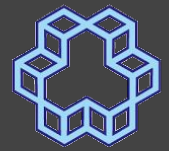
```c
void printArray(int *a, int n) {
  for (int i = 0; i < n; i++)
    printf("%d, ", *(a+i));
  putchar('\n');
}
```

$a[0] \equiv *a$

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

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

pointer arithmetic & arrays



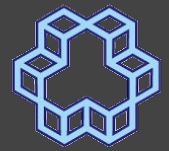
pointerarithmetic6.c

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

pointerarithmetic7.c

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

pointer arithmetic & arrays



pointerarithmetic7.c

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

pointerarithmetic8.c

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

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

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

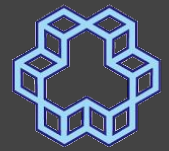
Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
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

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm

pointerarithmetic8.c

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

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

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

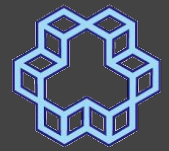
Exercise



pointerarithmetic10.c

```c
#include <stdio.h>

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;
}
```

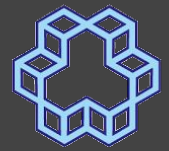
Exercise



pointerarithmetic11.c

```c
#include <stdio.h>

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;
}
```

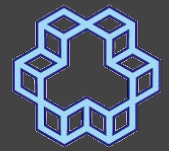
Subtracting two pointers




```
int a[10];

int *p = a;

int *q = &a[6];

int i = q - p;
```

what is the value of i?

Subtracting two pointers




```
int a[10];

int *p = a;

int *q = &a[6];

int i = q - p;
```

what is the value of i? 6