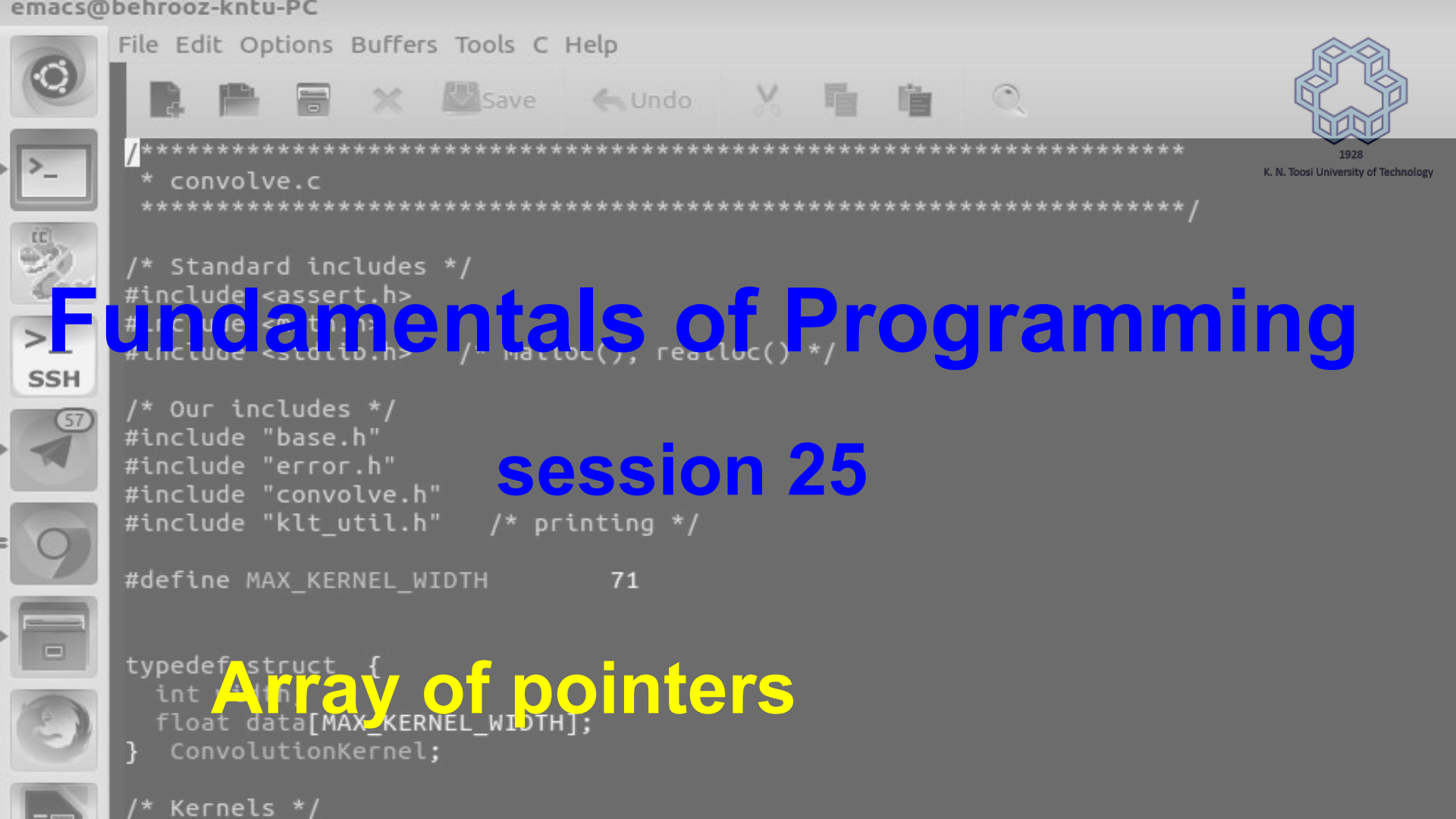
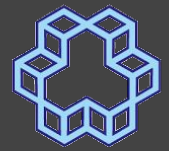
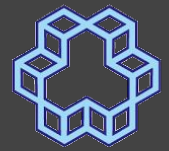

emacs@behrooz-kntu-PC
File Edit Options Buffers Tools C Help
Save
Undo
1928
K. N. Toosi University of Technology
/***
* convolve.c
***/
/* Standard includes */
#include <assert.h>
/* Our includes */
#include "base.h"
#include "error.h"
#include "convolve.h"
#include "klt_util.h" /* printing */
#define MAX_KERNEL_WIDTH 71
typedef struct {
float data[MAX_KERNEL_WIDTH];
} ConvolutionKernel;
/* Kernels */
Fundamentals of Programming
session 25
Array of pointers

Array of pointers




```
int *a[10];

char *s[20];
```

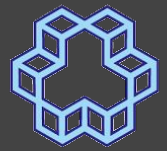
Array of pointers




```c
#include <stdio.h>

int main() {
  char *seasons[4];
  char s0[] = "Spring";
  char s1[] = "Summer";
  char s2[] = "Fall";
  char s3[] = "Winter";

  seasons[0] = s0;
  seasons[1] = s1;
  seasons[2] = s2;
  seasons[3] = s3;

  return 0;
}
```

arrayofpointers0.c

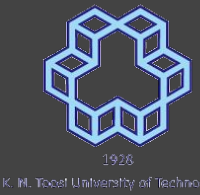
Array of pointers




```c
#include <stdio.h>

int main() {
  char *seasons[4];

  seasons[0] = "Spring";
  seasons[1] = "Summer";
  seasons[2] = "Fall";
  seasons[3] = "Winter";

  return 0;
}
```

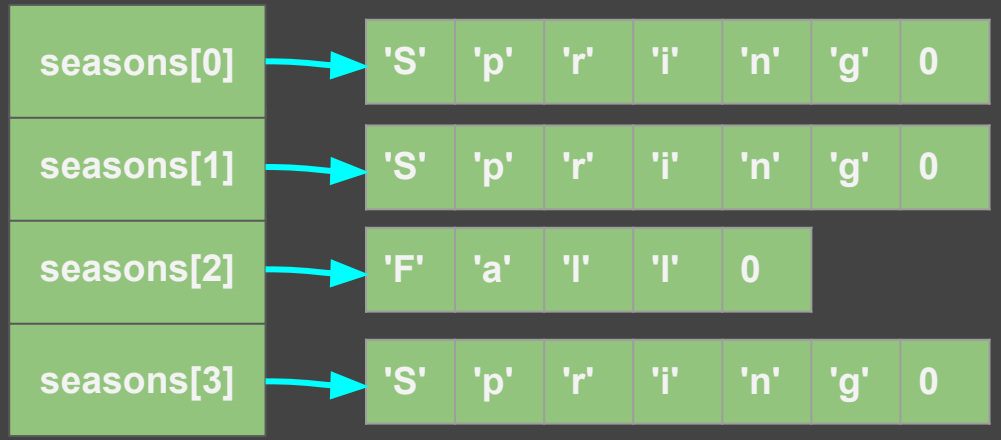
arrayofpointers1.c

Array of pointers

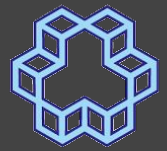



```c
#include <stdio.h>

int main() {
  char *seasons[4];

  seasons[0] = "Spring";
  seasons[1] = "Summer";
  seasons[2] = "Fall";
  seasons[3] = "Winter";

  return 0;
}
```

arrayofpointers1.c

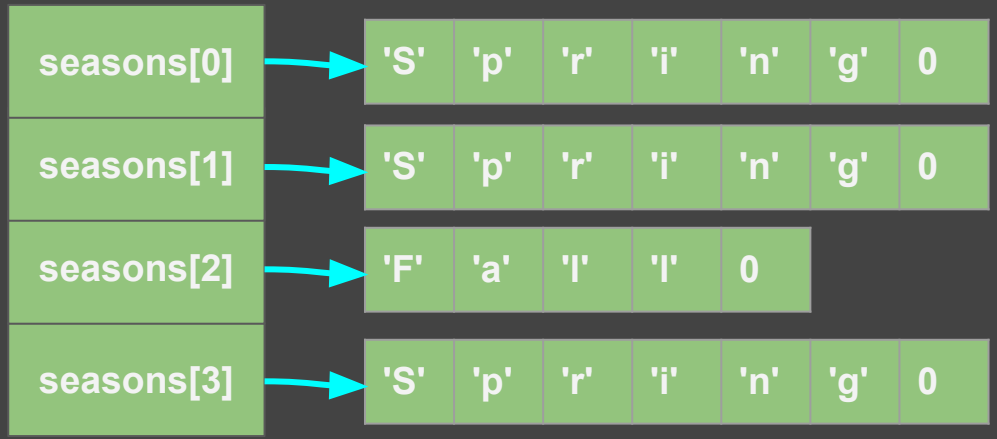


Array of pointers

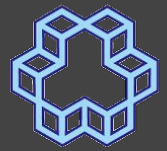



```c
#include <stdio.h>

int main() {
  char *seasons[4] = {"Spring", "Summer", "Fall", "Winter"};

  return 0;
}
```

arrayofpointers2.c



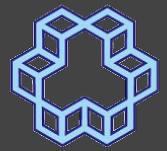
Array of pointers




```c
#include <stdio.h>

int main() {
  char *seasons[4] = {"Spring", "Summer", "Fall", "Winter"};
  char *order[4]   = {"1st",    "2nd",    "3rd",  "4th"};

  for (int i = 0; i < 4; i++)
    printf("%s season is %s\n", order[i], seasons[i]);

  return 0;
}
```

arrayofpointers3.c

Array of pointers

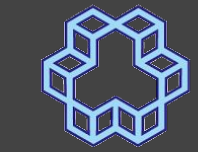



```c
#include <stdio.h>

int main() {
  char *seasons[4] = {"Spring", "Summer", "Fall", "Winter"};
  char *order[4]   = {"1st",    "2nd",    "3rd",  "4th"};

  for (int i = 0; i < 4; i++)
    printf("%s season is %s\n", order[i], seasons[i]);

  return 0;
}
```

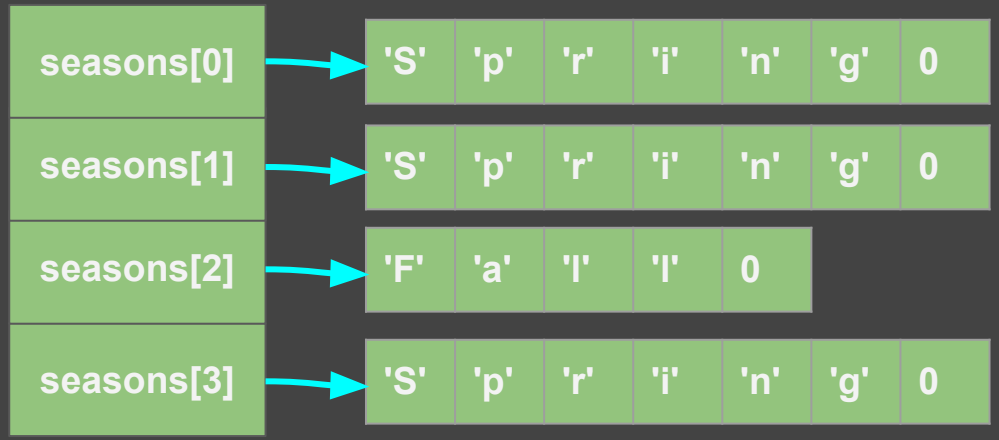
arrayofpointers3.c

```
nasihatkon@kntu:code$ gcc arrayofpointers3.c && ./a.out
1st season is Spring
2nd season is Summer
3rd season is Fall
4th season is Winter
```

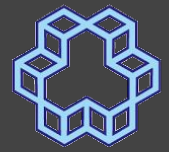
Referencing pointers of arrays



seasons[1][3] = ?

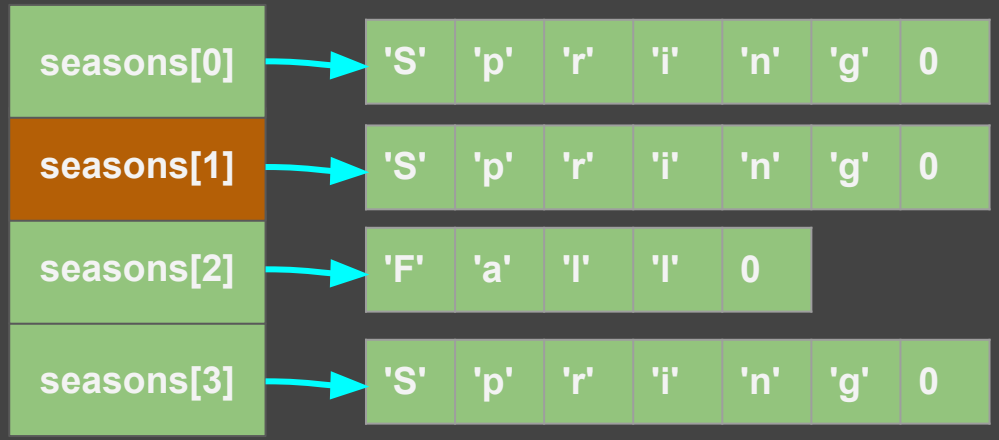


Referencing pointers of arrays

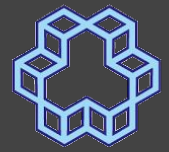


seasons[1][3]

= (seasons[1]) [3]

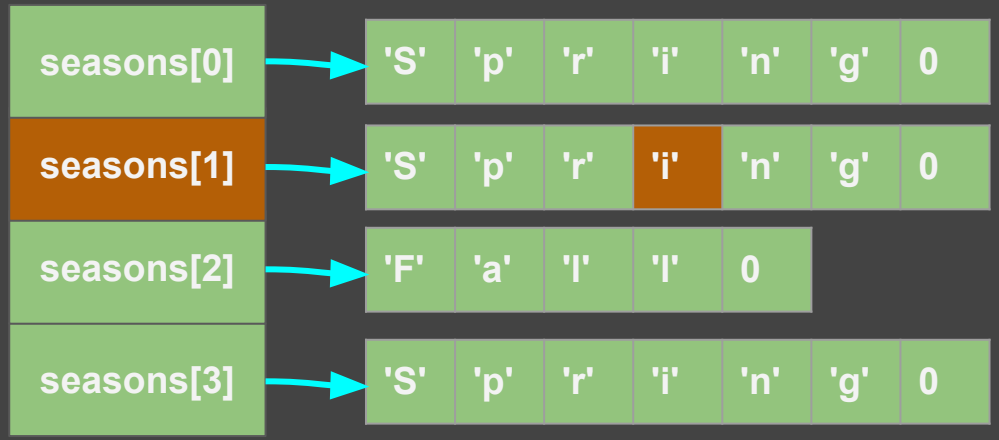


Referencing pointers of arrays

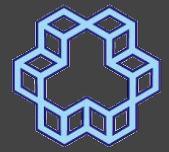


seasons[1][3]

= (seasons[1]) [3] = 'i'



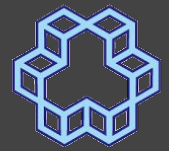
Array of pointers




```c
#include <stdio.h>

int main() {
  char *seasons[4] = {"Spring", "Summer", "Fall", "Winter"};

  for (int i = 0; i < 4; i++) {
    for (int j = 0; seasons[i][j] != 0; j++)
      putchar(seasons[i][j]);
    putchar('\n');
  }
  return 0;
}
```

arrayofpointers4.c

Array of pointers

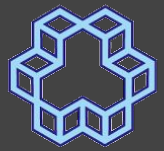



```c
#include <stdio.h>

int main() {
  char *seasons[4] = {"Spring", "Summer", "Fall", "Winter"};

  for (int i = 0; i < 4; i++) {
    for (int j = 0; seasons[i][j] != 0; j++)
      putchar(seasons[i][j]);
    putchar('\n');
  }
  return 0;
}
```

arrayofpointers4.c

```
nasihatkon@kntu:code$ gcc arrayofpointers4.c && ./a.out
Spring
Summer
Fall
Winter
```

Array of pointers vs. 2D arrays



Memory

0
1
2
3
10
11
12
13
20
21
22
23

```
int a[3][4] = {{ 0,  1,  2,  3},
               {10, 11, 12, 13},
               {20, 21, 22, 23}};
```

```
int *a[3];

int array0[] = { 0,  1,  2,  3};
int array1[] = {10, 11, 12, 13};
int array2[] = {20, 21, 22, 23};

a[0] = array0;
a[1] = array1;
a[2] = array2;
```

a[0]
a[1]
a[2]
0
1
2
3
10
11
12
13
20
21
22
23

Array of pointers vs. 2D arrays

Memory

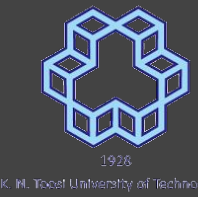
0
1
2
3
10
11
12
13
20
21
22
23

```
int a[3][4] = {{ 0,  1,  2,  3},
               {10, 11, 12, 13},
               {20, 21, 22, 23}};
```

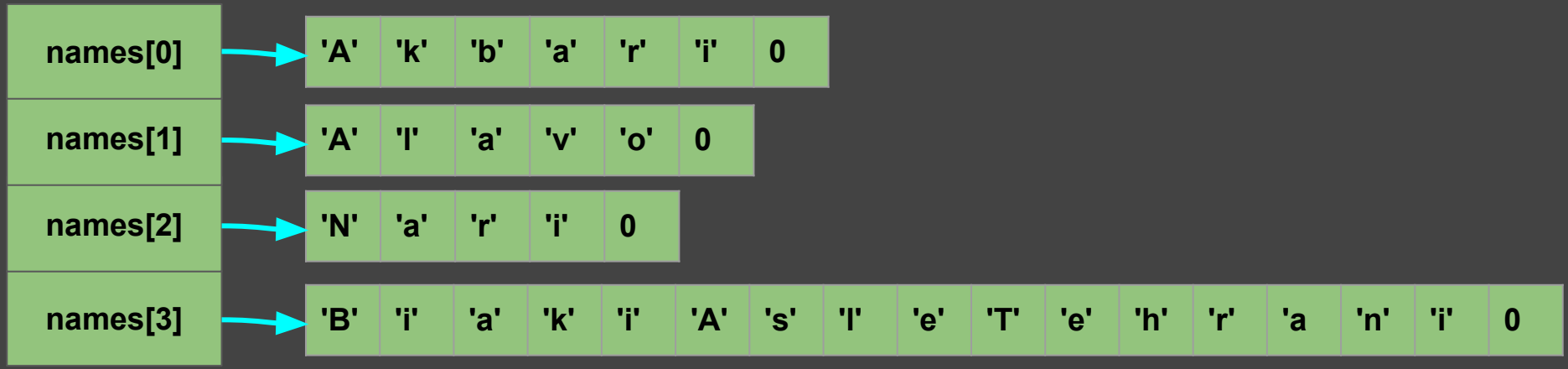
```
int *a[3];

a[0] = (int[]) { 0,  1,  2,  3};
a[1] = (int[]) {10, 11, 12, 13};
a[2] = (int[]) {20, 21, 22, 23};
```

a[0]
a[1]
a[2]
0
1
2
3
10
11
12
13
20
21
22
23

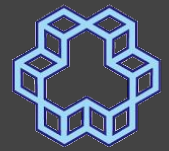
Array of pointers vs. 2D arrays



- Advantages of array of pointers
  - Rows of different length (more flexibility, saving memory)



Array of pointers vs. 2D arrays



- Advantages of array of pointers
  - Rows of different length (more flexibility, saving memory)

Array of pointers vs. 2D arrays

Memory

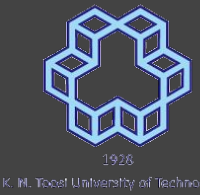
0
1
2
3
10
11
12
13
20
21
22
23

```
int a[3][4] = {{ 0,  1,  2,  3},
               {10, 11, 12, 13},
               {20, 21, 22, 23}};
```

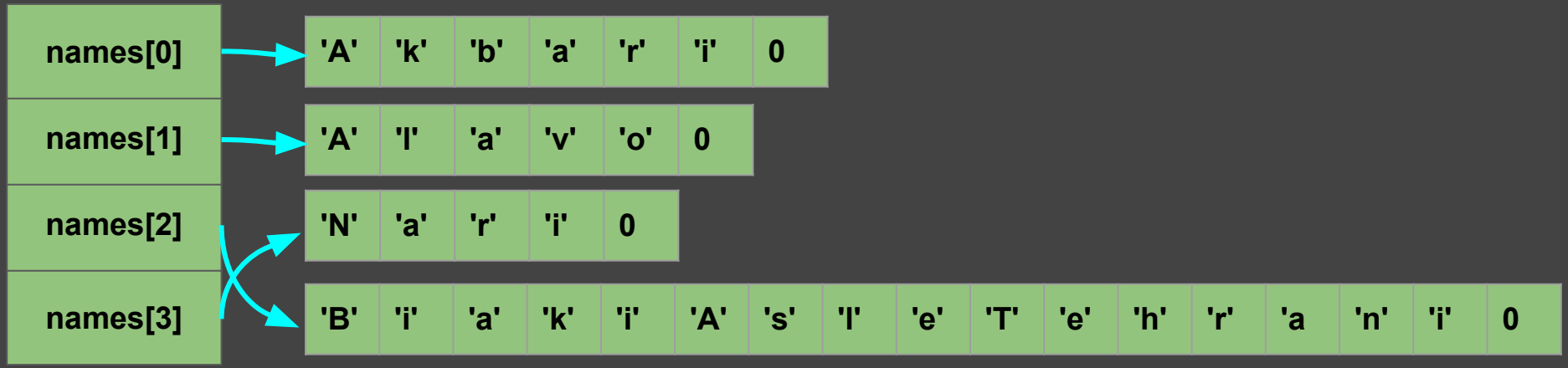
```
int *a[3];

a[0] = (int[]) { 0,  1,  2,  3};
a[1] = (int[]) {10, 11, 12, 13, 14};
a[2] = (int[]) {20, 21, 22};
```

a[0]
a[1]
a[2]
0
1
2
3
10
11
12
13
14
20
21
22

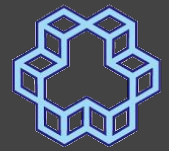
Array of pointers vs. 2D arrays



- Advantages of array of pointers
  - Rows of different length (more flexibility, saving memory)
  - Swapping two rows efficiently



Array of pointers vs. 2D arrays



- Advantages of 2D arrays
  - Efficiency (in accessing elements)