

```
1 // Created by hengxin on 11/27/2024.
2
3 int main() {
4     char **argv;
5
6     int *names[10];
7
8     int (*musician_score_table)[10];
9
10    int *StrCpyStd(char *dest, const char *src);
11
12    int (*comp)(const void *left, const void *right);
13
14    int atexit(void (*func)(void));
15
16    void (*signal(int sig, void (*handler)(int)))(int);
17
18    char ((*func(int num, char *str))[])( );
19
20    char ((*arr[3])())[5];
21 }
```

```
1  /**
2   * Echo program (command-line) arguments.
3   *
4   * Created by hengxin on 11/27/2024.
5   */
6
7  #include <stdio.h>
8
9  // arg: argument
10 // c: count
11 // v: vector
12
13 // argv[0]: the name of the program
14 // argv[1 .. argc - 1]: command line arguments
15 // argv[argc]: NULL
16 int main(int argc, char *argv[]) {
17     // for version with argv
18     // for (int i = 1; i < argc; ++i) {
19     //     printf("argv[%d] = %s\n", i, argv[i]);
20     // }
21
22     // for version with pointers
23     // for (char **ptr = argv + 1; *ptr != NULL; ptr++) {
24     //     printf("%s\n", *ptr);
25     // }
26
27     // while version
28     // char **ptr = argv + 1;
29     // while (*ptr != NULL) {
30     //     printf("%s\n", *ptr);
31     //     ptr++;
32     // }
33
34     // WRONG
35     // char **ptr = argv + 1;
36     // while (*ptr++ != NULL) {
37     //     printf("%s\n", *ptr);
38     // }
39
40     char **ptr = argv;
41     // ++ptr
42     // *ptr++
43     // --ptr
44     // *ptr--
```

```
45  while (*++ptr != NULL) {  
46      printf("%s\n", *ptr);  
47  }  
48  
49  return 0;  
50 }
```

```
1 // Created by hengxin on 11/27/2024.
2
3 // Python Tutor Visualization:
4 // (1) https://tinyurl.com/scores-of-musicians
5 // (2) https://tinyurl.com/scores-of-musicians-malloc
6 // https://tinyurl.com/
7
8 #include <stdio.h>
9 #include <stdlib.h>
10
11 #define NUM_OF_MUSICIANS 4
12 #define NUM_OF_SCORES 3
13
14 void Print(const int table[][NUM_OF_SCORES], int
    num_of_musicians);
15
16 int main() {
17     // C, Java, Python scores of several musicians
18
19     // TODO: (1) initialize scores with a 2D array
20     // const int scores[NUM_OF_MUSICIANS][NUM_OF_SCORES] = {
21     //     { 0, 10, 20 },
22     //     { 10, 20, 30 },
23     //     { 20, 30, 40 },
24     //     { 30, 40, 50 },
25     // };
26
27     // TODO: (2) Dynamically allocate memory for scores
28     // malloc here
29     int (*scores)[NUM_OF_SCORES] = malloc(NUM_OF_MUSICIANS
    * sizeof(*scores));
30     if (scores == NULL) {
31         return 0;
32     }
33
34     // fill in data here
35     for (int i = 0; i < NUM_OF_MUSICIANS; ++i) {
36         for (int j = 0; j < NUM_OF_SCORES; ++j) {
37             scores[i][j] = i * j;
38         }
39     }
40
41     // print it here
42     Print(scores, NUM_OF_MUSICIANS);
```

```

43
44 // ptr_scores here
45 int (*ptr_scores)[NUM_OF_SCORES] = scores;
46 printf("ptr_scores[3][2] = %d\n",
47        (*(ptr_scores + 3))[2]);
48
49 // do not forget to free it
50 free(scores);
51
52 return 0;
53 }
54
55 // table: int table[][COL]
56 // int table[]: int *table
57 // int table[][COL]: int (*table)[COL]
58 void Print(const int table[][NUM_OF_SCORES], int
num_of_musicians) {
59     for (int i = 0; i < num_of_musicians; i++) {
60         for (int j = 0; j < NUM_OF_SCORES; j++) {
61             // table[i][j]: (*(table + i) + j)
62             // table: int (*)[COL]
63             // table + i: int (*)[COL]
64             // *(table + i): int *
65             // *(table + i) + j: int *
66             // (*(table + i) + j): int
67             printf("table[%d][%d]: %d\n\n",
68                    i, j, table[i][j]);
69
70             printf("table: %p\n", table);
71             printf("table + %d: %p\n", i, table + i);
72             printf("*(table + %d): %p\n", i, *(table + i));
73             printf("*(table + %d) + %d: %p\n", i, j, *(table + i
) + j);
74             printf("*(*(table + %d) + %d): %d\n", i, j, (*(
table + i) + j));
75         }
76         printf("\n\n");
77     }
78 }

```

```
1 # 10-double-pointers
2
3 ## `selection-sort-strings.c`
4
5 - `const`
6
7 ## `echo.c`
8
9 - Linux `echo`
10 - C standard
11 - `printf("%s\n", argv[i])`: printf the nullptr
12
13 ## `scores.c`
14
15 - `student_score_table`: as a 2D array
16 - `Print`
17   - `int table[][COLS]` vs. `int (*table)[COLS]`
18 - `malloc`
19   - `int *`
20   - `int (*)[COLS]`
```

```
1 add_executable(selection-sort-ints selection-sort.c)
2 add_executable(selection-sort-strings selection-sort-
  strings.c)
3
4 add_executable(scores scores.c)
5
6 add_executable(echo echo.c)
7
8 add_executable(pointers-malloc pointers-malloc.c)
9
10 add_executable(decl decl.c)
```

```
1 // Created by hfwei on 2024/11/27.
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 #define LEN 5
7
8 void SelectionSort(int *arr, int len);
9 int GetMinIndex(const int *arr, int begin, int end);
10 void Swap(int *left, int *right);
11 void Print(const int *arr, int len);
12
13 int main(void) {
14     int len = 0;
15     scanf("%d", &len);
16
17     int *numbers = malloc(len * sizeof(*numbers));
18
19     if (numbers == NULL) {
20         return EXIT_FAILURE;
21     }
22
23     for (int i = 0; i < len; ++i) {
24         scanf("%d", &numbers[i]);
25     }
26
27     Print(numbers, len);
28     SelectionSort(numbers, len);
29     Print(numbers, len);
30
31     free(numbers);
32 }
33
34 void SelectionSort(int *arr, int len) {
35     for (int i = 0; i < len; i++) {
36         int min_index = GetMinIndex(arr, i, len);
37         Swap(arr + i, arr + min_index);
38     }
39 }
40
41 int GetMinIndex(const int *arr, int begin, int end) {
42     int min = arr[begin];
43     int min_index = begin;
44 }
```



```
45     for (int i = begin + 1; i < end; ++i) {
46         if (arr[i] < min) {
47             min = arr[i];
48             min_index = i;
49         }
50     }
51
52     return min_index;
53 }
54
55 void Swap(int *left, int *right) {
56     int temp = *left;
57     *left = *right;
58     *right = temp;
59 }
60
61 void Print(const int *arr, int len) {
62     printf("\n");
63     for (int i = 0; i < len; i++) {
64         printf("%d ", arr[i]);
65     }
66     printf("\n");
67 }
```

```
1 // Created by hfwei on 2024/11/27.
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 #define LEN 3
7 #define ROW 3
8 #define COL 4
9
10 int main(void) {
11     // (1) One-Dimensional Array
12     // Visualization: https://tinyurl.com/pointers-malloc-
    int
13
14     // malloc
15     int *array = malloc(LEN * sizeof *array);
16     if (array == NULL) {
17         printf("malloc failed\n");
18         return EXIT_FAILURE;
19     }
20
21     // fill in
22     for (int i = 0; i < LEN; ++i) {
23         array[i] = i * i;
24     }
25
26     // print
27     for (int i = 0; i < LEN; ++i) {
28         printf("%d ", array[i]);
29     }
30     printf("\n\n");
31
32     // free
33     free(array);
34
35     // (2) Two-Dimensional Array
36     // Visualization: https://tinyurl.com/pointers-malloc-
    int-array
37
38     // malloc
39     int (*table)[COL] = malloc(ROW * sizeof *table);
40
41     // fill in
42     for (int i = 0; i < ROW; ++i) {
```

```
43     for (int j = 0; j < COL; ++j) {
44         table[i][j] = i * j;
45     }
46 }
47
48 // print
49 for (int i = 0; i < ROW; ++i) {
50     for (int j = 0; j < COL; ++j) {
51         printf("%d ", table[i][j]);
52     }
53     printf("\n");
54 }
55 printf("\n");
56
57 // free
58 free(table);
59
60 // (3) One-Dimensional Array of Pointers
61 // Visualization: https://tinyurl.com/pointers-malloc-arraypointers
62
63 // malloc and fill in
64 int *array_of_pointers[LEN];
65 for (int i = 0; i < LEN; ++i) {
66     array_of_pointers[i] = malloc((i + 1) * sizeof *
array_of_pointers[i]);
67     for (int j = 0; j < i + 1; ++j) {
68         array_of_pointers[i][j] = i * j;
69     }
70 }
71
72 // print
73 for (int i = 0; i < LEN; ++i) {
74     for (int j = 0; j < i + 1; ++j) {
75         printf("%d ", array_of_pointers[i][j]);
76     }
77     printf("\n");
78 }
79 printf("\n");
80
81 // free
82 for (int i = 0; i < LEN; ++i) {
83     free(array_of_pointers[i]);
84 }
```

```

85
86  // (4) Two-Dimensional Array with Potentially Non-
    Contiguous Memory
87  // Visualization: https://tinyurl.com/pointers-malloc-
    int-pp
88
89  // malloc
90  int **matrix = malloc(ROW * sizeof *matrix);
91  for (int i = 0; i < ROW; ++i) {
92      matrix[i] = malloc(COL * sizeof *matrix[i]);
93  }
94
95  // fill in
96  for (int i = 0; i < ROW; ++i) {
97      for (int j = 0; j < COL; ++j) {
98          matrix[i][j] = i * j;
99      }
100 }
101
102 // print
103 for (int i = 0; i < ROW; ++i) {
104     for (int j = 0; j < COL; ++j) {
105         printf("%d ", matrix[i][j]);
106     }
107     printf("\n");
108 }
109 printf("\n");
110
111 // free
112 for (int i = 0; i < ROW; ++i) {
113     free(matrix[i]);
114 }
115 free(matrix);
116
117 // (5) Two-Dimensional Array with Contiguous Memory
118 // Visualization: https://tinyurl.com/pointers-malloc-
    int-p
119
120 // malloc
121 int *matrix_contiguous = malloc(ROW * COL * sizeof *
    matrix_contiguous);
122
123 // fill in
124 for (int i = 0; i < ROW; ++i) {

```

```
125     for (int j = 0; j < COL; ++j) {
126         matrix_contiguous[i * COL + j] = i * j;
127     }
128 }
129
130 // print
131 for (int i = 0; i < ROW; ++i) {
132     for (int j = 0; j < COL; ++j) {
133         printf("%d ", matrix_contiguous[i * COL + j]);
134     }
135     printf("\n");
136 }
137
138 // free
139 free(matrix_contiguous);
140
141 return 0;
142 }
```

```
1 // Created by hfwei on 2024/11/27.
2
3 // Python Tutor Visualization: https://tinyurl.com/array-
  of-musicians (LEN 3)
4 // https://tinyurl.com/
5
6 #include <stdio.h>
7 #include <string.h>
8
9 #define LEN 10
10
11 void SelectionSort(const char *arr[], int len);
12 int GetMinIndex(const char *arr[], int begin, int end);
13 void Swap(const char **left, const char **right);
14 void Print(const char *arr[], int len);
15
16 int main(void) {
17     const char *musicians[LEN] = {
18         "Luo Dayou",
19         "Cui Jian",
20         "Dou Wei",
21         "Zhang Chu",
22         "Wan Qing",
23         "Li Zhi",
24         "Yao",
25         "ZuoXiao",
26         "ErShou Rose",
27         "Hu Mage",
28     };
29
30     Print(musicians, LEN);
31     SelectionSort(musicians, LEN);
32     Print(musicians, LEN);
33
34     return 0;
35 }
36
37 void SelectionSort(const char *arr[], int len) {
38     for (int i = 0; i < len; i++) {
39         int min_index = GetMinIndex(arr, i, len);
40
41         // arr: char *[]
42         // char**
43         // const char **
```

```
44     Swap(arr + i, arr + min_index);
45 }
46 }
47
48 int GetMinIndex(const char *arr[], int begin, int end) {
49     const char *min = arr[begin];
50     int min_index = begin;
51
52     for (int i = begin + 1; i < end; ++i) {
53         if (strcmp(arr[i], min) < 0) {
54             min = arr[i];
55             min_index = i;
56         }
57     }
58
59     return min_index;
60 }
61
62 void Swap(const char **left, const char **right) {
63     const char *temp = *left;
64     *left = *right;
65     *right = temp;
66 }
67
68 void Print(const char *arr[], int len) {
69     for (int i = 0; i < len; i++) {
70         // arr[i]
71         printf("%s\n", arr[i]);
72     }
73     printf("\n");
74 }
```