

```
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 int main(int argc, char *argv[]) {
10     // for version with argv
11
12     // for version with pointers
13
14     // while version
15
16     return 0;
17 }
```

```
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
21     // TODO: (2) Dynamically allocate memory for scores
22     // malloc here
23
24     // fill in data here
25
26     // print it here
27
28     // ptr_scores here
29 // int (*ptr_scores)[NUM_OF_SCORES] = scores;
30 // printf("ptr_scores[3][2] = %d\n",
31 //         (*(ptr_scores + 3))[2]);
32
33     // do not forget to free it
34
35     return 0;
36 }
37
38 void Print(const int table[][NUM_OF_SCORES], int
    num_of_musicians) {
39     for (int i = 0; i < num_of_musicians; i++) {
40         for (int j = 0; j < NUM_OF_SCORES; j++) {
41             printf("table[%d][%d]: %d\n\n",
42                 i, j, table[i][j]);
```

```
43
44 //      printf("table: %p\n", table);
45 //      printf("table + %d: %p\n", i, table + i);
46 //      printf("*(table + %d): %p\n", i, *(table + i));
47 //      printf("*(table + %d) + %d: %p\n", i, j, *(table
    + i) + j);
48 //      printf("*(*(table + %d) + %d): %d\n", i, j, (*(
    table + i) + j));
49     }
50     printf("\n\n");
51 }
52 }
53
54 //      { 0, 10, 20 },
55 //      { 10, 20, 30 },
56 //      { 20, 30, 40 },
57 //      { 30, 40, 50 },
```

```
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 // TODO: Explain "const"
42 int GetMinIndex(const int *arr, int begin, int end) {
43     int min = arr[begin];
44     int min_index = begin;
```

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



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

```
44
45 int GetMinIndex(const int arr[], int begin, int end) {
46     int min = arr[begin];
47     int min_index = begin;
48
49     for (int i = begin + 1; i < end; ++i) {
50         if (arr[i] < min) {
51             min = arr[i];
52             min_index = i;
53         }
54     }
55
56     return min_index;
57 }
58
59 void Swap(int *left, int *right) {
60     int temp = *left;
61     *left = *right;
62     *right = temp;
63 }
64
65 void Print(const int arr[], int len) {
66     printf("\n");
67     for (int i = 0; i < len; i++) {
68         printf("%d ", arr[i]);
69     }
70     printf("\n");
71 }
72
73 // "Luo Dayou",
74 // "Cui Jian",
75 // "Dou Wei",
76 // "Zhang Chu",
77 // "Wan Qing",
78 // "Li Zhi",
79 // "Yao",
80 // "ZuoXiao",
81 // "ErShou Rose",
82 // "Hu Mage",
```