

```
1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4 #include <stdbool.h>
5
6 bool IsLeapYear(int year);
7
8 int main(void) {
9     // local variable
10    // scope: block scope
11    int year = 0;
12    scanf("%d", &year);
13
14    // caller
15    // callee
16    bool leap = IsLeapYear(year);
17
18    if (leap) {
19        printf("%d is a leap year\n", year);
20    } else {
21        printf("%d is a common year\n", year);
22    }
23
24    return 0;
25 }
26
27 // year: formal parameter
28 // block scope
29 bool IsLeapYear(int year) {
30    // local variable
31    // scope: block scope
32    bool leap = (year % 4 == 0 && year % 100 != 0) || (year
33    % 400 == 0);
34    return leap;
35 }
```

```
1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4 #include <time.h>
5 #include <stdlib.h>
6
7 int main(void) {
8     int high = 100;
9     int number_of_tries = 7;
10
11     /*
12      * (1) print the rules of the game to players
13      */
14     printf("Let us play the Guess the Number game.\n"
15           "The computer will generate a random number
16           between 1 and %d.\n"
17           "You have %d tries.\n",
18           high, number_of_tries);
19
20     /*
21      * (2) generate a random number between 1 and high
22      */
23     srand(time(NULL));
24     int secret = rand() % high + 1;
25     printf("Test: secret = %d\n", secret);
26
27     while (number_of_tries > 0) {
28         /*
29          * (3) ask the player to enter his/her guess
30          */
31         printf("Please enter your guess number.\n"
32               "You still have %d tries.\n", number_of_tries);
33
34         /*
35          * (4) obtain the guessed number, compare it with the
36          *      secret number,
37          *      and inform the player of the result.
38          */
39         int guess = 0;
40         scanf("%d", &guess);
41         printf("Test: guess = %d\n", guess);
42
43         if (guess == secret) {
44             printf("You Win!\n");
45         }
46     }
47 }
```

```
43     break;
44 } else if (guess > secret) {
45     printf("guess > secret\n");
46 } else {
47     printf("guess < secret\n");
48 }
49
50 /*
51  * (5) repeat (3) and (4) until the player wins or
    loses.
52  */
53     number_of_tries--;
54
55     if (number_of_tries == 0) {
56         printf("You Lose!\n");
57     }
58 }
59
60 return 0;
61 }
```

```
1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4
5 #define LEN_L 5
6 #define LEN_R 6
7
8 int L[LEN_L] = { 1, 3, 5, 7, 9 };
9 int R[LEN_R] = { 0, 2, 4, 6, 8, 10 };
10
11 void Merge(const int left[], int left_len,
12           const int right[], int right_len);
13
14 int main(void) {
15     int l = 0;
16     int r = 0;
17
18     while (l < LEN_L && r < LEN_R) {
19         if (L[l] <= R[r]) {
20             printf("%d ", L[l]);
21             l++;
22         } else {
23             printf("%d ", R[r]);
24             r++;
25         }
26     }
27
28     while (r < LEN_R) {
29         printf("%d ", R[r]);
30         r++;
31     }
32
33     while (l < LEN_L) {
34         printf("%d ", L[l]);
35         l++;
36     }
37
38     return 0;
39 }
```

```
1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4
5 void Print(char ch, int count);
6 void NewLine(void);
7
8 int main(void) {
9     int lines = 0;
10    scanf("%d", &lines);
11
12    for (int i = 0; i < lines; ++i) {
13        Print(' ', lines - 1 - i);
14        Print('*', 2 * i + 1);
15
16        if (i < lines - 1) {
17            printf("\n");
18            NewLine();
19        }
20    }
21
22    return 0;
23 }
24
25 void Print(char ch, int count) {
26     for (int i = 0; i < count; ++i) {
27         printf("%c", ch);
28     }
29 }
30
31 void NewLine(void) {
32     printf("\n");
33 }
```

```
1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4 #include <stdbool.h>
5
6 bool IsPrime(int number);
7
8 int main(void) {
9     int max = 0;
10    scanf("%d", &max);
11
12    int count = 0;
13
14    for (int number = 2; number <= max; number++) {
15        if (IsPrime(number)) {
16            count++;
17            printf("%d ", number);
18        }
19    }
20
21    printf("\ncount = %d\n", count);
22
23    return 0;
24 }
25
26 bool IsPrime(int number) {
27     for (int factor = 2; factor * factor <= number; factor
28         ++ ) {
29         if (number % factor == 0) {
30             return false;
31         }
32     }
33     return true;
34 }
```

```
1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4 #include <string.h>
5 #include <stdbool.h>
6
7 #define LEN 21
8 char string[LEN] = "";
9
10 bool IsPalindrome(const char str[]);
11
12 int main() {
13     printf("Input a string containing at most 20 characters.
14     \n");
15     scanf("%20s", string);
16     printf("\'%s\' is %s a palindrome.\n", string,
17           IsPalindrome(string) ? "" : "not");
18
19     return 0;
20 }
21
22 bool IsPalindrome(const char str[]) {
23     int len = strlen(str);
24
25     for (int i = 0, j = len - 1; i < j; i++, j--) {
26         if (str[i] != str[j]) {
27             return false;
28         }
29     }
30
31     return true;
32 }
```

```
1 # 0-intro
2 add_executable(guess-func guess.c)
3
4 # 2-if-for-array
5 add_executable(leap-func leap.c)
6
7 # 3-for-a-while
8 add_executable(stars-func stars.c)
9 add_executable(primes-func primes.c)
10 add_executable(binary-search-func binary-search.c)
11 add_executable(palindrome-func palindrome.c)
12 add_executable(selection-sort-func selection-sort.c)
13
14 # 4-loops
15 add_executable(merge-func merge.c)
16 add_executable(game-of-life-func game-of-life.c)
17 add_executable(game-of-life-transformed game-of-life-
    transformed.c)
18
19 add_executable(insertion-sort-func insertion-sort.c)
20 add_executable(binary-insertion-sort-func binary-insertion
    -sort.c)
```



```

1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <unistd.h>
6 #include <synchapi.h>
7
8 #define SIZE 6
9 const int board[SIZE][SIZE] = {
10     { 0 },
11     { 0, 1, 1, 0, 0, 0 },
12     { 0, 1, 1, 0, 0, 0 },
13     { 0, 0, 0, 1, 1, 0 },
14     { 0, 0, 0, 1, 1, 0 },
15     { 0 }
16 };
17
18 //const int board[SIZE][SIZE] = {
19 //    [1][1] = 1, [1][2] = 1,
20 //    [2][1] = 1, [2][2] = 1,
21 //    [3][3] = 1, [3][4] = 1,
22 //    [4][3] = 1, [4][4] = 1
23 //};
24
25 int main() {
26     // extended board
27     int old_board[SIZE + 2][SIZE + 2] = { 0 };
28
29     for (int row = 1; row <= SIZE; row++) {
30         for (int col = 1; col <= SIZE; col++) {
31             old_board[row][col] = board[row - 1][col - 1];
32         }
33     }
34
35     // print the original board
36     for (int row = 1; row <= SIZE; row++) {
37         for (int col = 1; col <= SIZE; col++) {
38             printf("%c ", old_board[row][col] ? '*' : ' ');
39         }
40         printf("\n");
41     }
42     system("clear"); // clear the screen/terminal
43
44     int new_board[SIZE + 2][SIZE + 2] = { 0 };

```

```

45
46     for (int round = 1; round < 10; round++) {
47         for (int row = 1; row <= SIZE; row++) {
48             for (int col = 1; col <= SIZE; col++) {
49                 // count the number of neighbours of old_board[row
50                 ][col]
51                 int neighbours =
52                     old_board[row - 1][col - 1] +
53                     old_board[row - 1][col] +
54                     old_board[row - 1][col + 1] +
55                     old_board[row][col - 1] +
56                     old_board[row][col + 1] +
57                     old_board[row + 1][col - 1] +
58                     old_board[row + 1][col] +
59                     old_board[row + 1][col + 1];
60
61                 // evaluate the new board
62                 if (old_board[row][col]) { // old_board[row][col
63                     ] is alive
64                     new_board[row][col] = (neighbours == 2 ||
65                     neighbours == 3);
66                 } else { // old_board[row][col] is dead
67                     new_board[row][col] = (neighbours == 3);
68                 }
69             }
70         }
71     }
72     // print the new board
73     for (int row = 1; row <= SIZE; row++) {
74         for (int col = 1; col <= SIZE; col++) {
75             printf("%c ", new_board[row][col] ? '*' : ' ');
76         }
77         printf("\n");
78     }
79
80     // sleep for a while
81     // Linux: #include <unistd.h>
82     sleep(1);
83     // Windows: #include <windows.h>: Sleep(ms)
84     // Sleep(1000);
85
86     // clear the screen
87     // Linux: #include <stdlib.h>
88     system("clear");

```

```
86      // Windows: #include <stdlib.h> system("clr");
87      // system("clr");
88
89      // start the next round
90      for (int row = 1; row <= SIZE; row++) {
91          for (int col = 1; col <= SIZE; col++) {
92              old_board[row][col] = new_board[row][col];
93          }
94      }
95  }
96
97  return 0;
98 }
```

```
1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4
5 #define LEN 10
6
7 // global variable
8 // file scope
9 // const int dictionary[LEN] = { 1, 1, 2, 3, 5, 8, 13, 21
10 // , 34, 55 };
11 // int dict[]: the address of the first element of the
12 // array
13
14 int BinarySearch(int key, const int dict[], int len);
15
16 int main(void) {
17     const int dictionary[LEN] = { 1, 1, 2, 3, 5, 8, 13, 21,
18     34, 55 };
19
20     int key = 0;
21     scanf("%d", &key);
22
23     // dictionary (actual argument): const int[]
24     // dict (formal parameter): int[]
25     int index = BinarySearch(key, dictionary, LEN);
26     if (index == -1) {
27         printf("Not found!\n");
28     } else {
29         printf("The index of %d is %d.\n", key, index);
30     }
31
32     return 0;
33 }
34
35 int BinarySearch(int key, const int dict[], int len) {
36     int low = 0;
37     int high = len - 1;
38
39     while (low <= high) {
40         int mid = (low + high) / 2;
41
42         if (key > dict[mid]) {
43             low = mid + 1;
44         } else if (key < dict[mid]) {
45             high = mid - 1;
46         } else {
47             return mid;
48         }
49     }
50     return -1;
51 }
```

```
42     high = mid - 1;
43 } else { // key == dict[mid]
44     return mid;
45 }
46 }
47
48 return -1;
49 }
```

```
1 // Created by hfwei on 2024/10/16.
2 // Code generated by ChatGPT.
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <time.h>
7
8 #define MAX_LEN 10000
9 #define RANGE 10
10
11 int main(void) {
12     int numbers[MAX_LEN] = { 0 };
13
14     int size = 0;
15     scanf("%d", &size);
16
17     // generate an array of random integers between 0 and
18     RANGE - 1
19     srand(time(NULL));
20     for (int i = 0; i < size; i++) {
21         numbers[i] = rand() % RANGE;
22     }
23     // print the original array
24     for (int i = 0; i < size; i++) {
25         printf("%d ", numbers[i]);
26     }
27     printf("\n");
28
29     // TODO: insertion sort
30     for (int i = 1; i < size; i++) {
31         // numbers[0 .. i - 1] is already sorted
32         int key = numbers[i];
33
34         int j = i - 1;
35         while (j >= 0 && numbers[j] > key) {
36             numbers[j + 1] = numbers[j];
37             j--;
38         }
39         numbers[j + 1] = key;
40
41         // numbers[0 .. i] is already sorted
42         for (int i = 0; i < size; i++) {
43             printf("%d ", numbers[i]);
```

```
44     }
45     printf("\n");
46 }
47 // i = size
48 // numbers[0 .. size - 1] is already sorted
49
50 // print the sorted array
51 for (int i = 0; i < size; i++) {
52     printf("%d ", numbers[i]);
53 }
54 printf("\n");
55
56 return 0;
57 }
```

```
1 // Created by hfwei on 2024/10/23.
2
3 #include <stdio.h>
4
5 #define LEN 20
6 int numbers[LEN] = { 0 };
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 = -1;
15     while (scanf("%d", &numbers[++len]) != EOF);
16
17     Print(numbers, len);
18     SelectionSort(numbers, len);
19     Print(numbers, len);
20
21     return 0;
22 }
23
24 void SelectionSort(int arr[], int len) {
25     for (int i = 0; i < len; i++) {
26         int min_index = GetMinIndex(arr, i, len);
27
28         // swap arr[i] and arr[min_index]
29         // Swap(left, right)
30         // arr[i] = 3    arr[min_index] = 5
31         // left = 3    right = 5
32         // Swap(arr[i], arr[min_index]);
33
34         int temp = arr[i];
35         arr[i] = arr[min_index];
36         arr[min_index] = temp;
37     }
38 }
39
40 int GetMinIndex(const int arr[], int begin, int end) {
41     int min = arr[begin];
42     int min_index = begin;
43
44     for (int j = begin + 1; j <= end - 1; ++j) {
```



```
45     if (arr[j] < min) {
46         min = arr[j];
47         min_index = j;
48     }
49 }
50
51 return min_index;
52 }
53
54 // left = 3    right = 5
55 void Swap(int left, int right) {
56     int temp = left;
57     left = right;
58     right = temp;
59     // left = 5 right = 3
60 }
61
62 void Print(const int arr[], int len) {
63     printf("\n");
64     for (int i = 0; i < len; i++) {
65         printf("%d ", arr[i]);
66     }
67     printf("\n");
68 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 #define MAX_LEN 10000
6 #define RANGE 10
7
8 int main() {
9     int numbers[MAX_LEN] = { 0 };
10
11     int size = 0;
12     scanf("%d", &size);
13
14     srand(time(NULL));
15     for (int i = 0; i < size; i++) {
16         numbers[i] = rand() % RANGE;
17     }
18
19     // print the original array
20     for (int i = 0; i < size; i++) {
21         printf("%d ", numbers[i]);
22     }
23     printf("\n");
24
25     // TODO
26     for (int i = 1; i < size; i++) {
27         int key = numbers[i];
28         int low = 0;
29         int high = i - 1;
30
31         while (low <= high) {
32             int mid = (low + high) / 2;
33             if (key >= numbers[mid]) {
34                 low = mid + 1;
35             } else {
36                 high = mid - 1;
37             }
38         }
39
40         for (int j = i - 1; j >= low; j--) {
41             numbers[j + 1] = numbers[j];
42         }
43         numbers[low] = key;
44     }
```

```
45     for (int i = 0; i < size; i++) {
46         printf("%d ", numbers[i]);
47     }
48     printf("\n");
49 }
50
51 // Print the sorted array
52 for (int i = 0; i < size; i++) {
53     printf("%d ", numbers[i]);
54 }
55 printf("\n");
56
57 return 0;
58 }
```

```
1 // Created by hfwei on 2024/10/23.
2 // Run it with "Terminal"
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <unistd.h>
7
8 #define SIZE 6
9
10 void ExtendBoard(const int origin_board[][SIZE],
11                 int extended_board[][SIZE + 2]);
12 void PrintExtendedBoard(const int extended_board[][SIZE +
13 2]);
14 void GenerateNewBoard(const int old_board[][SIZE + 2],
15                       int new_board[][SIZE + 2]);
16 void CopyExtendedBoard(const int src_board[][SIZE + 2],
17                         int dest_board[][SIZE + 2]);
18 void SleepAndClear(int sec);
19
20 int main() {
21     const int board[SIZE][SIZE] = {
22         { 0 },
23         { 0, 1, 1, 0, 0, 0 },
24         { 0, 1, 1, 0, 0, 0 },
25         { 0, 0, 0, 1, 1, 0 },
26         { 0, 0, 0, 1, 1, 0 },
27         { 0 }
28     };
29     int old_board[SIZE + 2][SIZE + 2] = { 0 };
30     ExtendBoard(board, old_board);
31     PrintExtendedBoard(old_board);
32     SleepAndClear(1);
33
34     int new_board[SIZE + 2][SIZE + 2] = { 0 };
35     for (int round = 0; round < 10; round++) {
36         GenerateNewBoard(old_board, new_board);
37         SleepAndClear(1);
38         PrintExtendedBoard(new_board);
39         CopyExtendedBoard(new_board, old_board);
40     }
41
42     return 0;
43 }
```

```

44
45 void ExtendBoard(const int origin_board[][SIZE],
46                 int extended_board[][SIZE + 2]) {
47     for (int row = 1; row <= SIZE; row++) {
48         for (int col = 1; col <= SIZE; col++) {
49             extended_board[row][col] = origin_board[row - 1][col
50             - 1];
51         }
52     }
53
54 void PrintExtendedBoard(const int extended_board[][SIZE +
55 2]) {
56     for (int row = 1; row <= SIZE; row++) {
57         for (int col = 1; col <= SIZE; col++) {
58             printf("%c ", extended_board[row][col] ? '*' : ' ');
59         }
60     }
61 }
62
63 void GenerateNewBoard(const int old_board[][SIZE + 2],
64                      int new_board[][SIZE + 2]) {
65     for (int row = 1; row <= SIZE; row++) {
66         for (int col = 1; col <= SIZE; col++) {
67             // count the number of neighbours of old_board[row][
68             col]
69             int neighbours =
70                 old_board[row - 1][col - 1] +
71                 old_board[row - 1][col] +
72                 old_board[row - 1][col + 1] +
73                 old_board[row][col - 1] +
74                 old_board[row][col + 1] +
75                 old_board[row + 1][col - 1] +
76                 old_board[row + 1][col] +
77                 old_board[row + 1][col + 1];
78
79             // evaluate the new board
80             if (old_board[row][col]) { // old_board[row][col]
81                 is alive
82                 new_board[row][col] = (neighbours == 2 ||
83                 neighbours == 3);
84             } else { // old_board[row][col] is dead

```

```
82         new_board[row][col] = (neighbours == 3);
83     }
84 }
85 }
86 }
87
88 void CopyExtendedBoard(const int src_board[][SIZE + 2],
89                        int dest_board[][SIZE + 2]) {
90     for (int row = 1; row <= SIZE; row++) {
91         for (int col = 1; col <= SIZE; col++) {
92             dest_board[row][col] = src_board[row][col];
93         }
94     }
95 }
96
97 void SleepAndClear(int sec) {
98     sleep(sec);
99     system("clear");
100 }
```