
과제 5

[실습제목: 문제지 8]



과 목 명	C 프로그래밍
교 수 명	김 병 정
학 번	20237107
작 성 자	하 태 영
제 출 일	2025.12.10

한림대학교

문제 C81-0001

```
C C81-0001.c > arr_shift_cnt(int [], int, int)
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  void debug_title(char x_str[]) // 배열 문자열 전달
6  {
7      printf("%s : ", x_str);
8  }
9
10 void debug_title2(char x_str[], int x_val) // 배열 문자열 전달
11 {
12     printf("%s : %d \n", x_str, x_val);
13 }
14
15 void debug_arr(int x_arr[], int x_cnt)
16 {
17     for(int i=0; i<x_cnt; i++)
18         printf("%d ", x_arr[i]);
19     printf("\n");
20 }
```

```
C C81-0001.c > arr_shift_cnt(int [], int, int)
22 void arr_shift(int x_arr[], int x_cnt, char op)
23 {
24     switch(op)
25     {
26         case 'l':
27             for(int i=0; i<x_cnt-1; i++)
28             {
29                 int tmp;
30                 tmp = x_arr[i];
31                 x_arr[i] = x_arr[i+1];
32                 x_arr[i+1] = tmp;
33             }
34             break;
35         case 'r':
36             for(int i=x_cnt-1; i>0; i--)
37             {
38                 int tmp;
39                 tmp = x_arr[i];
40                 x_arr[i] = x_arr[i-1];
41                 x_arr[i-1] = tmp;
42             }
43             break;
44     }
45 }
```

C C81-0001.c > ...

```
48 void arr_shift_cnt(int x_arr[], int x_cnt, int op_cnt)
49 {
50     int shift_count = (op_cnt > 0) ? op_cnt : -op_cnt; // 이동 횟수의 절댓값
51     char direction;
52
53     // 1. 이동 방향 결정
54     if (op_cnt > 0) {
55         direction = 'r'; // 양수면 오른쪽 이동
56     } else if (op_cnt < 0) {
57         direction = 'l'; // 음수면 왼쪽 이동
58     } else {
59         return; // 0이면 이동할 필요 없음
60     }
61
62     // 2. 결정된 횟수(절댓값)만큼 arr_shift 함수 반복 호출
63     for (int i=0; i< shift_count; i++) {
64         // 배열과 크기, 방향을 arr_shift 함수에 전달하여 1회 이동 수행
65         arr_shift(x_arr, x_cnt, direction);
66     }
67 }
68
69 int main()
70 {
71     int arr[10] = {1, 2, 3, 4, 5};
72
73     debug_title2("물리적메모리크기", sizeof(arr));
74     debug_title2("저장공간갯수", sizeof(arr)/sizeof(int));
75
76     int arr_cnt = 5; // 배열의 사용크기를 알려주어야 한다.
77
78     debug_title("초기값");
79     debug_arr(arr, arr_cnt);
80
81     arr_shift_cnt(arr, arr_cnt, 4); // 오른쪽 4번 shift
82     arr_shift_cnt(arr, arr_cnt, -5); // 왼쪽 5번 shift
83
84     debug_title("최종값");
85     debug_arr(arr, arr_cnt);
86 }
```

● (base) hataeyeong@hataeyeong-ui-MacBookPro C기 초 문제지 (8) % cd "/Users/hataeyeon
rs/hataeyeong/Desktop/Study/University/3-2/C프로그래밍/C기 초 문제지 (8)"/"C81-0001
물리적메모리크기 : 40
저장공간갯수 : 10
초기값 : 1 2 3 4 5
최종값 : 2 3 4 5 1

문제 C82-0001

C C82-0001.c > ...

```
1  #include <stdio.h>
2
3  void delete_and_shift(int arr[], int *used_size, int delete_index){
4      // 1. 삭제할 인덱스 뒤의 원소들을 앞으로 한 칸씩 당김
5      for(int i = delete_index; i < (*used_size) - 1; i++) {
6          arr[i] = arr[i+1]; // arr[i+1]의 값을 arr[i]로 복사
7      }
8
9      // 2. 사용 크기 1 감소
10     (*used_size)--;
11 }
12
13 void debug_arr(int arr[], int used_size){
14     for(int i = 0; i < used_size; i++) {
15         printf("%d ", arr[i]);
16     }
17     printf("\n");
18 }
19
20 int main() {
21     int arr[] = {10, 20, 30, 40, 50};
22     int used_size = 5;
23     int delete_index = 2;
24
25     delete_and_shift(arr, &used_size, delete_index);
26     debug_arr(arr, used_size);
27     return 0;
28 }
```

● (base) hataeyeong@hataeyeong-ui-MacBookPro C기 초 문제 지 (8)
ersity/3-2/C프로그래밍/C기 초 문제 지 (8)/"C82-0001
10 20 40 50

문제 C82-0002

```
C C82-0002.c > ...
1  #include <stdio.h>
2
3  void insert_value(int arr[], int *used_size, int arrCnt, int index, int value){
4      // 1. 삽입 실패 조건 검사 (사용 크기가 물리적 크기와 같을 경우)
5      if (*used_size >= arrCnt) {
6          printf("삽입 실패\n");
7          return;
8      }
9
10     // 2. 인덱스 유효성 검사 (index는 0 이상 used_size 이하)
11     if (index < 0 || index > *used_size) {
12         printf("유효하지 않은 인덱스입니다.\n");
13         return;
14     }
15
16     // 3. 삽입 위치(index)부터 끝까지 원소를 뒤로 한 칸씩 밀 (Shift Right)
17     for (int i = *used_size; i > index; i--) {
18         arr[i] = arr[i - 1];
19     }
20
21     // 4. 새 값을 삽입
22     arr[index] = value;
23
24     // 5. 사용 크기 1 증가
25     (*used_size)++;
26 }
27
28 void debug_arr(int arr[], int used_size){
29     for(int i = 0; i < used_size; i++) {
30         printf("%d ", arr[i]);
31     }
32     printf("\n");
33 }
34
36 int main() {
37     int arr[] = {1, 2, 3, 4};
38     int used_size = 4;
39     int index = 2;
40     int value = 99;
41
42     insert_value(arr, &used_size, 10, index, value);
43     debug_arr(arr, used_size);
44     return 0;
45 }
```

```
● (base) hataeyeong@hataeyeong-ui-MacBookPro C기 초 문제지 (8) % cd "/
iversity/3-2/C프로그래밍/C기 초 문제지 (8)/" && gcc C82-0002.c -o C8
esktop/Study/University/3-2/C프로그래밍/C기 초 문제지 (8)"/C82-0002
1 2 99 3 4
```

문제 C82-0003

```
C C82-0003.c > ...
1  #include <stdio.h>
2
3  // 배열 상태 출력 함수
4  void debug_arr(int arr[], int used_size) {
5      for (int i = 0; i < used_size; i++) {
6          printf("%d ", arr[i]);
7      }
8      printf("\n");
9  }
10
11 // 원소 삭제 및 이동 함수
12 void delete_and_shift(int arr[], int *used_size, int delete_index) {
13     // 1. 유효성 검사 (인덱스 범위 확인)
14     if (delete_index < 0 || delete_index >= *used_size) {
15         printf("삭제 실패: 인덱스 범위 오류\n");
16         return;
17     }
18
19     // 2. Shift Left: 삭제할 위치 뒤의 원소들을 앞으로 한 칸씩 당김
20     // 앞의 원소를 뒤의 원소로 덮어쓰는 방식 (Overwrite)
21     for (int i = delete_index; i < *used_size - 1; i++) {
22         arr[i] = arr[i + 1];
23     }
24
25     // 3. 사용 크기 1 감소 (포인터 사용)
26     (*used_size)--;
27 }
28
29 int main() {
30     // [예시 1]
31     int arr1[10] = {10, 20, 30, 40, 50};
32     int used_size1 = 5;
33
34     // 인덱스 2 (값 30) 삭제
35     delete_and_shift(arr1, &used_size1, 2);
36     debug_arr(arr1, used_size1); // 출력: 10 20 40 50
37
38     // [예시 2]
39     int arr2[10] = {7, 8, 9, 10, 11, 12};
40     int used_size2 = 6;
41
42     // 인덱스 0 (값 7) 삭제
43     delete_and_shift(arr2, &used_size2, 0);
44     debug_arr(arr2, used_size2); // 출력: 8 9 10 11 12
45
46     return 0;
47 }
```

- (base) hataeyeong@hataeyeong-ui-MacBookPro C기 초 문제지 (8) %
sers/hataeyeong/Desktop/Study/University/3-2/C프로그래밍/C기
지 (8)/" && gcc C82-0003.c -o C82-0003 && "/Users/hataeyeong/
/Study/University/3-2/C프로그래밍/C기 초 문제지 (8)"/"C82-0003
10 20 40 50
8 9 10 11 12

문제 C82-0003

1. 문제 풀이 과정

- 로직 설계: 배열의 특정 원소를 삭제하면 그 자리가 비게 되므로, 뒤에 있는 모든 원소를 앞으로 당겨와야 한다는 점을 AI에게 확인했습니다.
- 알고리즘 구현: AI의 조언을 통해 for 반복문을 사용하여 i번째 인덱스에 i+1번째 값을 덮어쓰는 (`arr[i] = arr[i + 1]`) 'Shift Left' 알고리즘을 구현했습니다.
- 변수 관리: 삭제 후 배열의 크기가 줄어들어야 하므로, main 함수의 `used_size` 변수 주소를 전달받아 함수 내에서 값을 감소시키는 포인터 방식을 적용했습니다.

2. 새롭게 이해하거나 어려웠던 내용

- Shift Left의 원리: 삭제는 단순히 값을 지우는 것이 아니라, 뒤의 값들을 앞으로 이동시켜 빈 공간을 메우는 과정임을 이해했습니다. 특히 반복문의 범위를 `used_size - 1`까지만 설정해야 마지막 원소를 당겨올 때 인덱스 초과 오류가 발생하지 않는다는 점을 배웠습니다.
- 포인터의 필요성: 함수 내부에서 단순히 `int used_size` 값을 줄이면 지역 변수만 변경되고 끝납니다. 따라서 `int *used_size` 포인터를 사용하여 원본 변수의 값을 직접 수정해야 한다는 'Call by Reference' 개념을 확실히 익혔습니다.

문제 C82-0004

C C82-0004.c > ...

```
1  #include <stdio.h>
2
3  // 배열 상태 출력 함수
4  void debug_arr(int arr[], int used_size) {
5      for (int i = 0; i < used_size; i++) {
6          printf("%d ", arr[i]);
7      }
8      printf("\n");
9  }
10
11 // 원소 삽입 및 이동 함수
12 void insert_value(int arr[], int *used_size, int arrCnt, int index, int value) {
13     // 1. 유효성 검사 (배열 꽉 참 또는 인덱스 범위 오류)
14     if (*used_size >= arrCnt) {
15         printf("삽입 실패\n");
16         return;
17     }
18     if (index < 0 || index > *used_size) {
19         printf("유효하지 않은 인덱스입니다.\n");
20         return;
21     }
22
23     // 2. Shift Right: 삽입할 위치 확보를 위해 뒤에서부터 원소를 뺀다
24     // 주의: 데이터를 덮어쓰지 않기 위해 반드시 '뒤에서 앞으로' 순회해야 함
25     for (int i = *used_size; i > index; i--) {
26         arr[i] = arr[i - 1];
27     }
28
29     // 3. 값 삽입 및 사용 크기 증가
30     arr[index] = value;
31     (*used_size)++;
32
33     // 결과 출력
34     debug_arr(arr, *used_size);
35 }
```

```

37  int main() {
38      // [예시 1] 정상 삽입
39      int arr1[10] = {1, 2, 3, 4};
40      int used_size1 = 4;
41
42      // 인덱스 2 위치에 99 삽입
43      insert_value(arr1, &used_size1, 10, 2, 99);
44      // 출력: 1 2 99 3 4
45
46      // [예시 2] 삽입 실패 (공간 부족)
47      int arr2[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
48      int used_size2 = 10;
49
50      // 인덱스 15 위치에 999 삽입 시도
51      insert_value(arr2, &used_size2, 10, 15, 999);
52      // 출력: 삽입 실패
53
54      return 0;
55  }

```

```

● (base) hataeyeong@hataeyeong-ui-MacBookPro C기 초 문 제 지 (8) % cd
iversity/3-2/C프로그래밍/C기 초 문 제 지 (8)/" && gcc C82-0004.c -o
top/Study/University/3-2/C프로그래밍/C기 초 문 제 지 (8)/"C82-0004
1 2 99 3 4
삽입 실패

```

1. 문제 풀이 과정

- 로직 설계: 배열 중간에 값을 넣으려면 기존 값들을 뒤로 밀어 공간을 만들어야 합니다. 이 과정에서 물리적 크기(arrCnt)를 초과하지 않는지 먼저 검사하는 로직을 AI와 함께 설계했습니다.
- 알고리즘 구현: 데이터를 뒤로 미는 과정에서 데이터 손실을 방지하기 위해, 배열의 끝에서부터 역순으로 반복문을 돌리는 'Shift Right' 방식을 적용했습니다.
- 오류 수정: 처음에는 반복문을 앞에서부터 돌려 값이 덮어쓰워지는 오류가 있었으나, AI의 피드백을 통해 인덱스를 감소(i--)시키는 방향으로 수정하여 해결했습니다.

2. 새롭게 이해하거나 어려웠던 내용

- 반복문의 방향성: 삽입 시 공간을 만들기 위해 데이터를 이동할 때, arr[i] = arr[i - 1] 로직을 사용합니다. 이때 앞에서부터 이동하면 이전 값이 다음 값을 덮어쓰버리게 됩니다. 따라서 반드시 뒤에서부터(역순으로) 이동해야 데이터가 안전하게 보존된다는 핵심 원리를 깨달았습니다.
- 물리적 크기와 논리적 크기: 배열이 메모리에 할당된 최대 크기(물리적 크기)와 현재 데이터가 들어있는 크기(논리적 크기)는 다르며, 삽입 전 반드시 if (*used_size >= arrCnt) 조건을 통해 오버플로우를 방지해야 함을 배웠습니다.

문제 C83-0001

```
C C83-0001.c > ...
1  #include <stdio.h>
2
3  void minmax(int arr[], int size, int *min_ptr, int *max_ptr)
4  {
5      *min_ptr = arr[0];
6      *max_ptr = arr[0];
7
8      for (int i = 1; i < size; i++) {
9          if (arr[i] < *min_ptr) {
10             *min_ptr = arr[i];
11         }
12         if (arr[i] > *max_ptr) {
13             *max_ptr = arr[i];
14         }
15     }
16 }
17 void io_main()
18 {
19     int arr[10];
20     int arr_cnt;;
21
22     int min = 100;
23     int max = 0;
24
25     printf("숫자 5개를 입력해주세요 ? "); //6 8 5 2 9
26     for(int i=0; i<5; i++)
27     {
28         if (scanf("%d", &arr[i]) != 1) {
29             printf("\n입력 오류가 발생했습니다.\n");
30             return;
31         }
32     }
33     printf("입력된 숫자 : ");
34     for(int i=0; i<5; i++)
35         printf("%d ",arr[i]);
36     printf("\n");
37
38     minmax(arr, 5, &min, &max);
39
40     printf("min :%d max : %d \n", min, max);
41 }
42 int main()
43 {
44     io_main();
45 }
```

```
● (base) hataeyeong@hataeyeong-ui-MacBookPro C기 초 문제지 (8) % cd
iversity/3-2/C프로그래밍 /C기 초 문제지 (8)/" && gcc C83-0001.c -o
top/Study/University/3-2/C프로그래밍 /C기 초 문제지 (8)/"C83-0001
숫자 5개를 입력해주세요 ? 6 8 5 2 9
입력된 숫자 : 6 8 5 2 9
min :2 max : 9
```

문제 C84-0001

C C84-0001.c > ...

```
1  #include <stdio.h>
2  #include <string.h> //strlen()
3
4  int find_dic(char* x_dic, int x_dic_cnt, char x_ch){
5      for(int j=0; j<x_dic_cnt; j++)
6      {
7          if(x_dic[j] == x_ch)
8              return j; //딕셔너리 내 위치 발견
9      }
10     return 0; //없음
11 }
12
13 void stack_dic(char* x_dic, int* x_dic_cnt, char x_ch) {
14     // 딕셔너리 배열의 현재 마지막 위치(*x_dic_cnt)에 새 문자를 저장합니다.
15     x_dic[*x_dic_cnt] = x_ch;
16
17     // 문자 개수를 1 증가시켜 다음 문자가 들어갈 위치를 갱신합니다.
18     (*x_dic_cnt)++;
19 }
20
21 void print_dic(char* x_dic, int x_dic_cnt){
22     for(int i=0; i<x_dic_cnt; i++)
23     {
24         printf("%c", x_dic[i]);
25
26         if(i==x_dic_cnt-2)
27         {
28             printf("\n");
29             break;
30         }
31         else
32             printf(",");
33     }
34 }
```

```

36 void local_main(){
37     char str[1024];
38
39     while (1)
40     {
41         char dic[1024] = {'\0'}; //NULL
42         int dic_cnt = 0;
43         dic_cnt = 0;
44         printf("공백 문자가 포함된 문자열을 입력하시오 ? ");
45         fgets(str, sizeof(str), stdin);
46
47         for (int i = 0; i < strlen(str); i++)
48         {
49             if (find_dic(dic, dic_cnt, str[i]) == 0)
50                 stack_dic(dic, &dic_cnt, str[i]);
51         }
52
53         print_dic(dic, dic_cnt);
54     }
55 }
56
57 int main() {
58     local_main();
59 }
60

```

```

○ (base) hataeyeong@hataeyeong-ui-MacBookPro C기 초 문 제 지 (8)
ersity/3-2/C프로그래밍 /C기 초 문 제 지 (8)/"C84-0001
공백 문자가 포함된 문자열을 입력하시오 ? Hello World
H,e,l,o, ,W,r,d
공백 문자가 포함된 문자열을 입력하시오 ? Hello Python
H,e,l,o, ,P,y,t,h,n
공백 문자가 포함된 문자열을 입력하시오 ? 1234 Hallym
1,2,3,4, ,H,a,l,l,y,m

```