C Language

Lecture 6. 배열

BAE SEKANG(배세강)

Contents

Array

♣ 배열

- 여러 개의 값을 연속하여 저장할 수 있는 공간

```
#include <stdio.h>

int main() {
    int arr[5] = {10, 20, 30, 40, 50};

    return 0;
}
```

Array

♣ 배열 선언

- [Type] [배열 이름][# of data] = {};

```
#include <stdio.h>

int main() {
    char vowels[5] = {'a', 'e', 'i', 'o', 'u'};
    for (int i = 0; i < 5; i++) {
        printf("%c ", vowels[i]);
    }

    return 0;
}
```

Array

♣ 배열 초기화

- 이상한 값이 들어있을 수 있기 때문에, 초기화를 해야 한다.
- {0}은 모든 칸을 0으로 초기화, 그 외의 숫자는 적용 되지 않는다. (직접 초기화 해야함: for문 활용)

```
#include <stdio.h>

int main() {
    int arr[5] = {0}; // {0, 0, 0, 0, 0}
    for (int i=0; i<5; ++i) {
        scanf("%d", &arr[i]);
    }
}
```

Array

♣ 인덱스(index)

- 0번부터 시작하는 배열의 주소

```
#include <stdio.h>

int main() {
    char vowels[5] = {'a', 'e', 'i', 'o', 'u'};

    printf("%c", vowels[0]); // Output: a

    return 0;
}
```

Array

♣ 인덱스(index)

- sizeof()를 활용하여 배열의 크기(N)를 구할 수 있다. ➔ $0 \leq \text{index} \leq N - 1$

```
#include <stdio.h>

int main() {
    char vowels[5] = {'a', 'e', 'i', 'o', 'u'};

    unsigned int N= sizeof(vowels) / sizeof(vowels[0]);
    printf("N: %u₩n", N);

    return 0;
}
```

Array

♣ Out of Bounds(OOB)

- 다른 메모리 영역에 침범하면 문제가 발생할 수 있다.
- Data Corruption: 값이 오염되어 추후 문제 발생 가능
- Segmentation Fault: OS가 보호하는 메모리 영역을 건드리면 프로그램 강제 종료
- OOB를 활용한 공격 ➔ Buffer Overflow

- Buffer Overflow: 사용자의 입력을 배열의 크기를 넘치게 입력하여 프로그램 실행 흐름를 바꾸는 공격 기법

Array

♣ 배열 사용

```
#include <stdio.h>

int main() {
    char vowels[5] = {'a', 'e', 'i', 'o', 'u'};
    char a = vowels[0];
    char e = vowels[1];
    vowels[2] = 'I';
    char i = vowels[2];
    char o = (vowels[3] - 32);
    char u = (vowels[4] - 32);

    printf("%c %c %c %c %c", a, e, i, o, u);
}
```

Array

[Practice]

문자열 S를 입력 받을 때, S에 포함된 모음의 개수를 출력한다. (단, S의 길이는 20자 이하로 가정)

[입력 1]
programming

[입력 2]
GOAT

[입력 3]
Hey, what's up?

[출력 1]
3

[출력 2]
2

[출력 3]
3

Array

♣ Memory Structure

- 배열은 메모리 상 옆으로 나란히 붙어있다.
- 주소 계산 방식: target addr = start arr + (index * size of structure)
  예) int인 arr[2] = 100 + (2 * 4) = 108 // 시작 주소를 100으로 가정

- 배열의 이름은 주소와 같다(포인터): arr은 arr[0]을 가리키는 시작 주소값

Array

♣ 배열과 문자열

- 문자열은 char 타입의 배열이다.
- printf()는 ₩0가 나올 때까지 출력한다. ₩0가 없다면 String Overrun
- String Overrun: 문자열 출력 시 범위를 벗어난 뒷 부분의 값까지 출력

```
#include <stdio.h>

int main() {
    char vowels[5] = {'a', 'e', 'i', 'o', 'u'}; // null 문자 없음!
    char vowels2[6] = {'a', 'e', 'i', 'o', 'u', '₩0'};
    printf("%s", vowels);
    printf("%s", vowels2);
    return 0;
}
```

Array

♣ 배열과 문자열

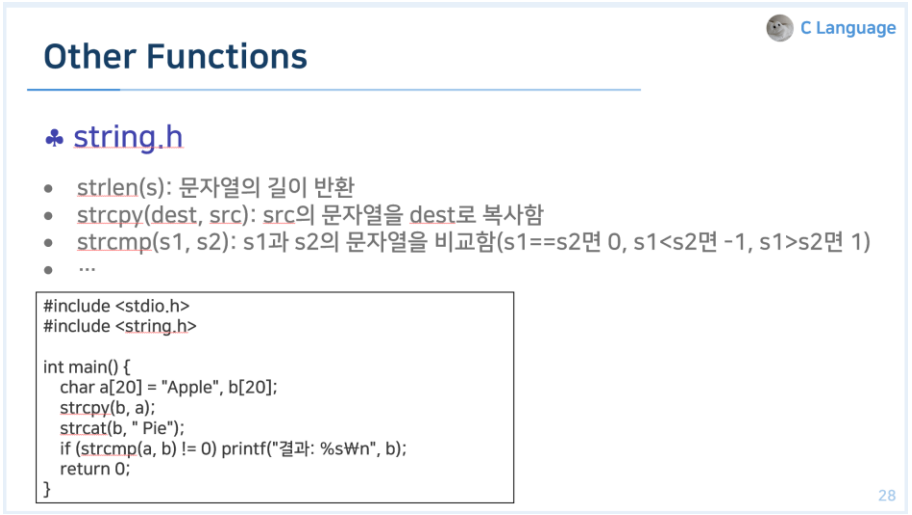
- 저장할 글자 수보다 크게 범위를 잡아야 함: (글자수 + 1)

```
#include <stdio.h>

int main() {
    char word[6] = {'h', 'e', 'l', 'l', 'o', '₩0'};
    printf("%s", word);
}
```

Array

♣ string.h 활용

```
#include <stdio.h>
#include <string.h>

int main() {
    char name[20];
    strcpy(name, "Alice"); // name = "Alice " 로 하면 에러 발생
    printf("%s₩n", name);
    return 0;
}
```



Lecture 7: string.h 참고

Array

[Practice]

자연수 N을 입력 받은 후, N개의 정수를 입력 받는다. N개의 정수 중 최댓값을 출력한다. (단 1 <= N <= 100이라 가정한다.)

[입력 1]
3
20 10 5

[입력 2]
5
1 -5 100 999 998

[출력 1]
20

[출력 2]
999

Multi-dimensional Array

♣ 2차원 배열

- 동일한 자료형을 가진 1차원 배열을 요소(element)로 갖는 배열
- [Type] [배열 이름][row][col];

```
#include <stdio.h>

int main() {
   int arr[3][4] = {
      {1, 2, 3, 4},
      {5, 6, 7, 8},
      {9, 10, 11, 12}
   };
   for (int i=0; i<3; ++i) {
      for (int j=0; j<4; ++j) {
         printf("%d ", arr[i][j]);
      }
      printf("₩n");
   }
}
```

Multi-dimensional Array

♣ 2차원 배열 활용 – Matrix Addition

```
#include <stdio.h>
int main() {
    int N, M;
    int A[100][100], B[100][100];
    scanf("%d %d", &N, &M);
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            scanf("%d", &A[i][j]);
        }
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            scanf("%d", &B[i][j]);
        }
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            printf("%d ", A[i][j] + B[i][j]);
        }
        printf("₩n");
    }
    return 0;
}
```

Exercise

[Problem 1]

문자열 S를 입력 받을 때, S를 뒤집은 후, S가 팰린드롬이라면 (O)를, 아니라면 (X)를 뒤에 붙여 출력한다.

[입력 1]
level

[입력 2]
apple

[출력 1]
level (O)

[출력 2]
elppa (X)

Exercise

[Problem 1] - Answer

```
#include <stdio.h>
#include <string.h>

int main() {
    char word[101];
    char reversed[101];

    int len = strlen(word);

    for (int i = 0; i < len; i++) reversed[i] = word[len - 1 - i];
    reversed[len] = '₩0'; // 중요: 직접 대입한 배열이므로 ₩0를 임의로 넣어주어야 한다!

    printf("%s", reversed);

    if (strcmp(word, reversed) == 0) {
        printf(" (O)₩n");
    } else {
        printf(" (X)₩n");
    }

    return 0;
}
```

Exercise

[Problem 2]

3x3 격자판에 1~9의 숫자가 채워져 있다. 이 격자판이 스도쿠가 되려면 1부터 9까지의 숫자가 중복 없이 딱 1번만 나타나야 한다. 입력 받은 격자판이 스도쿠이면 YES, 아니면 NO를 출력한다.

[입력 1]
1 2 3
4 5 6
7 8 9

[입력 2]
1 2 3
1 3 2
2 1 1

[입력 3]
9 8 7
1 2 3
6 5 4

[출력 1]
YES

[출력 2]
NO

[출력 3]
YES

Exercise

[Problem 2] - Answer

```
#include <stdio.h>
int main() {
    int board[3][3], check[10] = {0};
    int isSudoku = 1;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) scanf("%d", &board[i][j]);
    }
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            int num = board[i][j];
            if (num < 1 || num > 9 || check[num] == 1) {
                isSudoku = 0;
                break;
            }
            check[num] = 1;
        }
        if (!isSudoku) break;
    }
    if (isSudoku) printf("YES₩n");
    else printf("NO₩n");
    return 0;
}
```

THANK YOU!

Any Question?