

4장 프로그래밍 실습_C 코드

[프로그래밍 실습 1]

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

#pragma warning(disable : 4996)

void printPowerSet(int* set, int set_size);
void main()
{
    int i, set_size, *set;
    system("cls");
    // 사용자로부터 집합의 크기가 될 자연수를 입력받음
    printf("\n원소의 개수(자연수)를 입력하라.: ");
    scanf("%d", &set_size);
    set = (int*)malloc(set_size * sizeof(int));
    printf("\n집합 : {");

    // set 배열을 1부터 n까지의 자연수들로 초기화 및 원소 출력
    for (i = 0; i < set_size; i++) {
        if (i == set_size - 1)
            printf("%d", i + 1);
        else
            printf("%d,", i + 1);
        *(set + i) = i + 1;
    }
    printf("}\n\n{n} 는 공집합의 표시\n\n");
    printPowerSet(set, set_size);

    free(set);
    system("pause");
}

// printPowerSet 함수가 멍집합을 계산하면서 출력
void printPowerSet(int* set, int set_size) {
    int pow_set_size = 1 << set_size;    // 2^{set_size}로 멍집합 크기 정의 및 초기화
    int counter;
    int j = 0;

    // 집합을 다시 한 번 출력
    printf("{");
    for (int i = 0; i < set_size; i++) {
        if (i == set_size - 1)
            printf("%d", i + 1);
        else
            printf("%d,", i + 1);
    }
    printf("} 의 멍집합 : \n{ { } ");

    for (counter = 1; counter < pow_set_size; counter++) {
        printf(", { ");
        for (int j = 0; j < set_size; j++) {
            // counter의 (오른쪽부터) j번째 비트가 1이면 set[j]를 출력
            if ((counter & (1 << j)) > 0) {
                printf("%d,", set[j]);
            }
        }
    }
}
```



```

        printf("\n집합 B가 중복된 원소를 갖고 있다. 원소를 하나 더 입력하라.: ");
        i--;
        continue;
    }
    B[input] = true;
}

// 집합 A를 출력
printf("\nA = {");
setout(A);

// 집합 B를 출력
printf("\nB = {");
setout(B);

// 집합 A와 B의 합집합을 계산하고 출력
printf("A \u222A B = {");
set_union(A, B);

// 집합 A와 B의 교집합을 계산하고 출력
printf("A \u2229 B = {");
set_intersect(A, B);

// 집합 A와 B의 차집합(A-B)을 계산하고 출력
printf("A \u2216 B = {");
set_minus(A, B);

// 집합 A와 B의 차집합(B-A)을 계산하고 출력
printf("B \u2216 A = {");
set_minus(B, A);

free(A), free(B);
system("PAUSE");
}

// setout 함수가 주어진 집합을 출력
void setout(bool* A) {
    int count = 0, i;
    for (i = 0; i < MAX; i++) {
        if (A[i] == 1)
            break;
    }
    if (i == MAX) {
        printf("}\n");
        return;
    }

    for (i = 0; i < MAX; i++) {
        if (A[i] == 1) {
            if (count != 0) {
                printf(",");
            }
            printf("%3d", i);
            count = 1;
        }
    }
    printf(" }\n");
}

// Union 함수가 집합 A와 B의 합집합을 계산해서 출력

```

```

void set_union(bool* A, bool* B) {
    bool* result = (bool*)calloc(MAX, sizeof(bool));
    for (int i = 0; i < MAX; i++) {
        if (A[i] || B[i])
            result[i] = true;
    }

    setout(result);
    free(result);
}

// Inter 함수가 집합 A와 B의 교집합을 계산해서 출력
void set_intersect(bool* A, bool* B) {
    bool* result = (bool*)calloc(MAX, sizeof(bool));

    for (int i = 0; i < MAX; i++) {
        if (A[i] && B[i])
            result[i] = true;
    }

    setout(result);
    free(result);
}

// Minus 함수가 집합 A와 B의 차집합(A-B)을 계산해서 출력
void set_minus(bool* A, bool* B) {
    bool* result = (bool*)calloc(MAX, sizeof(bool));

    for (int i = 0; i < MAX; i++) {
        if (A[i] == true && B[i] == false)
            result[i] = true;
    }

    setout(result);
    free(result);
}

```