

C - 알고리즘 1

Algorithm



알고리즘 기초

- 알고리즘

어떤 문제를 해결하기 위한 절차나 방법이다. 주어진 입력을 출력으로 만드는 과정을 구체적이고 명료하게 표현한 것이다.

분류	세부 내용
수학 알고리즘	덧셈 알고리즘, 세 수의 최대값
재귀 알고리즘	팩토리얼, 피보나치 수열, 이진수
정렬 알고리즘	순위 정하기, 버블 정렬, 선택 정렬, 삽입 정렬
탐색 알고리즘	순차 탐색, 이분 탐색, 문자열



수학 알고리즘

- 덧셈 알고리즘
 - 1부터 5까지의 합

```
//단순 합계
printf("%d\n", 1 + 2 + 3 + 4 + 5);

//for문 사용
int i, sum = 0;

for (i = 1; i <= 5; i++) {
    sum += i;
}
printf("%d\n", sum);
```



수학 알고리즘

- 1부터 n까지의 합 계산하기

$$\boxed{1} + \boxed{2} + \boxed{3} + \cdots + \boxed{n} \longrightarrow \text{방법 1}$$

$$\boxed{n} \times (\boxed{n} + \boxed{1}) \div \boxed{2} \longrightarrow \text{방법 2}$$

101
101
101

1 + 2 + 3 + ... + 98 + 99 + 100

101 X 50 = 5050



수학 알고리즘

- 1부터 n까지의 합 계산하기

```
int sumN(int n) {  
    int i, sum = 0;  
  
    for (i = 1; i <= n; i++) {  
        sum += i;  
    }  
    return sum;  
}
```

```
int sumN2(int n) {  
    int sum = 0;  
    sum = (n * (n + 1)) / 2;  
  
    return sum;  
}
```



수학 알고리즘

- 1부터 n까지의 합 계산하기

```
//합계 1
int result1;
result1 = sumN(10);

printf("합계: %d\n", result1);

//합계 2
int result2;
result2 = sumN2(10);

printf("합계: %d\n", result2);
```



계산 복잡도 – 빅오(Big O)

- 계산 복잡도

- 입력 크기와 계산 횟수

- 첫 번째 알고리즘 : 덧셈 n 번
 - 두 번째 알고리즘 : 덧셈, 곱셈, 나눗셈(총 3번)

- 대문자 O표기법(Big O) : 계산 복잡도 표현

- $O(n)$: 필요한 계산횟수가 입력 크기 n 과 비례할 때
 - $O(1)$: 필요한 계산횟수가 입력 크기 n 과 무관할 때

- 판단

- 두 번째 방법이 계산 속도가 더 빠름



알고리즘 기초

- 평균 구하기

```
//평균 구하기
int a[] = { 70, 80, 65, 90 };
int sum = 0;
int i;

//단순 평균
printf("%d\n", (a[0] + a[1] + a[2] + a[3]) / 4);

//for문 사용
for (i = 0; i < 4; i++) {
    sum += a[i];
}
printf("평균은 %.1f입니다.\n", sum / 4.0);
```



막대 그래프 그리기

- 막대 그래프 그리기

```
//int arr[] = { 3, 6, 4, 2 };
int arr[4];
int i, j;

//사용자 입력
for (i = 0; i < 4; i++) {
    printf("arr[%d] 입력: ", i);
    scanf("%d", &arr[i]);
}
printf("\n");

//막대 그래프 출력
for (i = 0; i < 4; i++) {
    printf("arr[%d]=%d|", i, arr[i]);
    for (j = 1; j <= arr[i]; j++) {
        printf("*");
    }
    printf("\n");
}
```

```
arr[0] 입력: 3
arr[1] 입력: 6
arr[2] 입력: 4
arr[3] 입력: 2

arr[0]=3|***
arr[1]=6|*****
arr[2]=4|****
arr[3]=2|**
```



막대 그래프 그리기

- 막대 그래프 그리기

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h> //malloc(), free()
#include <stdbool.h> //true/false 사용
```

```
//동적 메모리 할당
int* arr = NULL; //포인터 선언
int size; //배열의 크기
int i, j;

printf("배열의 크기 입력: ");
scanf("%d", &size);

arr = (int*)malloc(sizeof(int) * size);
if (arr == NULL) {
    puts("메모리 할당에 실패했습니다.\n");
    return 1;
}
```



막대 그래프 그리기

- 막대 그래프 그리기

```
// 사용자 값 입력 및 유효성 검사
for (i = 0; i < size; i++) {
    while (true) {
        printf("arr[%d] 값 입력 (0 이상 정수): ", i);
        if (scanf("%d", &arr[i]) != 1 || arr[i] < 0) {
            printf("잘못된 입력입니다. 다시 입력하세요.\n");

            // 입력 버퍼 비우기
            while (getchar() != '\n');
        }
        else { //scanf("%d", &arr[i]) == 1
            break; //값 입력후 while문 빠져나옴
        }
    }
}
```



막대 그래프 그리기

- 막대 그래프 그리기

```
/*
    i=0, arr[0]=3, i++ , ***
    i=1, arr[1]=6, i++ , *****
    i=2, arr[1]=a, 잘못된 입력입니다.
    i=2, arr[2]=4, i++ , ****
    i=3, arr[3]=2, i++ , **
    i=4, 반복 종료
*/

//막대 그래프 출력
for (i = 0; i < size; i++) {
    printf("arr[%d]=%d|", i, arr[i]);
    for (j = 1; j <= arr[i]; j++) {
        printf("*");
    }
    printf("\n");
}
free(arr); //메모리 반납
```

```
배열의 크기 입력 : 4
arr[0] 값 입력 (0 이상 정수): 3
arr[1] 값 입력 (0 이상 정수): 6
arr[2] 값 입력 (0 이상 정수): f
잘못된 입력입니다. 다시 입력하세요.
arr[2] 값 입력 (0 이상 정수): 4
arr[3] 값 입력 (0 이상 정수): 2

arr[0]=3|***
arr[1]=6|*****
arr[2]=4|****
arr[3]=2|**
```



최대값 구하기

- 최대값 구하기

```
// 두 수 중 큰 값 계산
int x = 10, y = 20;
int result;

result = (x > y) ? x : y;
printf("두 수중 큰 수: %d\n", result);

// 세 수중 큰 수
int a = 10, b = 20, c = 30;
int max;

max = a; //a를 최대값 설정

if (b > max)
    max = b;
if (c > max)
    max = c;

printf("최대값은 %d입니다.\n", max);
```



최대값 구하기

- 최대값 구하기

```
int max3(int a, int b, int c) {  
    int max = a; //최대값 설정  
    if (b > max) max = b;  
    if (c > max) max = c;  
    return max;  
}  
  
int main()  
{  
    printf("max3(%d, %d, %d) = %d\n", 3, 2, 1, max3(3, 2, 1));  
    printf("max3(%d, %d, %d) = %d\n", 3, 1, 2, max3(3, 1, 2));  
    printf("max3(%d, %d, %d) = %d\n", 2, 1, 3, max3(2, 1, 3));  
    printf("max3(%d, %d, %d) = %d\n", 2, 3, 1, max3(2, 3, 1));  
  
    return 0;  
}
```



최대값 구하기 - 배열

- 최대값과 최대값 위치

```
int findMax(int a[], int len) { //최대값
    int i, maxVal;
    maxVal = a[0];
    for (i = 0; i < len; i++) {
        if (a[i] > maxVal)
            maxVal = a[i];
    }
    return maxVal;
}

int findMaxIdx(int a[], int len) { //최대값의 위치
    int i, maxIdx;
    maxIdx = 0;
    for (i = 0; i < len; i++) {
        if (a[i] > a[maxIdx])
            maxIdx = i;
    }
    return maxIdx;
}
```



최대값 구하기 - 배열

- 최대값과 최대값 위치

```
int arr[] = {53, 11, 65, 36, 29};  
int max, maxIndex;  
  
max = findMax(arr, 5);  
maxIndex = findMaxIdx(arr, 5);  
  
printf("최대값: %d, 최대값의 위치: %d\n", max, maxIndex);
```

```
max3(3, 2, 1) = 3  
max3(3, 1, 2) = 3  
max3(2, 1, 3) = 3  
max3(2, 3, 1) = 3
```



최대값 구하기 – 동적 할당

- 사람의 키를 입력받아 최대값 구하기

```
int number;
int* height; //동적 할당할 포인터배열

printf("사람 수: ");
scanf("%d", &number);
height = (int*)malloc(sizeof(int) * number);

printf("%d명의 키를 입력하세요\n", number);
for (int i = 0; i < number; i++) {
    printf("height[%d]: ", i);
    scanf("%d", &height[i]);
}
printf("최대값은 %d입니다.\n", findMax(height, number));

free(height); //메모리 해제
```

```
사람 수: 3
3명의 키를 입력하세요
height[0]: 172
height[1]: 165
height[2]: 180
최대값은 180입니다.
```



최대값 구하기 – 동적 할당

- 사람의 키를 랜덤하게 생성하고 최대값 구하기

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int findMax(int a[], int len) { //최대값
    int i, maxVal;
    maxVal = a[0];
    for (i = 0; i < len; i++)
        if (a[i] > maxVal) maxVal = a[i];
    return maxVal;
}
```

```
사람 수 : 9
height[0] = 113
height[1] = 153
height[2] = 153
height[3] = 152
height[4] = 160
height[5] = 120
height[6] = 174
height[7] = 170
height[8] = 147
최대값은 174입니다.
```



최대값 구하기 – 동적 할당

- 사람의 키를 랜덤하게 생성하고 최대값 구하기

```
int number;
int* height; //동적 할당할 포인터배열

printf("사람 수: ");
scanf("%d", &number);
height = (int*)malloc(sizeof(int) * number);
srand(time(NULL)); //시간으로 난수 seed 설정

for (int i = 0; i < number; i++) {
    height[i] = 100 + rand() % 91; //100 ~ 190의 난수 생성
    printf("height[%d] = %d\n", i, height[i]);
}
printf("최대값은 %d입니다.\n", findMax(height, number));

free(height); //메모리 해제
```



재귀 알고리즘

➤ 재귀 호출

어떤 함수 안에서 자기 자신을 부르는 것을 말한다.

재귀 호출은 무한 반복하므로 종료 조건이 필요하다.

```
func(입력값):  
    if 입력값이 충분히 작으면: //종료 조건  
        return 결과값  
    else  
        func(더 작은 입력값) //자기 자신 호출
```



재귀 알고리즘

➤ SOS 구현

```
void func(int n)
{
    //방법 2
    if (n <= 0) { //종료 조건
        return;
    }
    else {
        printf("Help Me!\n");
        func(n - 1);
    }

    //방법 1
    /*printf("Help Me!\n");
    n--;
    if (n <= 0)
        return; //종료 조건
    else
        func(n);
    */
}
```



재귀 알고리즘

➤ SOS 구현

```
int main()
{
    int count = 4;

    func(count);

    return 0;
}
```



재귀 알고리즘

➤ 팩토리얼 계산하기 – 일반적인 계산

```
#include <stdio.h>
/*
    - 1부터 5까지 곱하기
      1x2x3x4x5 -> 5!
*/
int factorial(int n) {
    int i, facto = 1;
    for (i = 1; i <= n; i++) {
        facto *= i;
    }
    return facto;
}
```



재귀 알고리즘

➤ 팩토리얼 계산하기 – 일반적인 계산

```
int main()
{
    int a, b, c;

    a = factorial(1); // 1 * factorial(0), 1 * 1 = 1
    b = factorial(2); // 2 * factorial(1), 2 * 1 = 2
    c = factorial(3); // 3 * factorial(2), 3 * 2 = 6

    printf("1!=%d, 2!=%d, 3!=%d\n", a, b, c);

    return 0;
}
```



재귀 알고리즘

➤ 팩토리얼 계산하기 - 재귀 알고리즘

```
/*  
    4! = 4 x 3 x 2 x 1  
    4! = 4 x 3!  
    3! = 3 x 2 x 1  
    3! = 3 x 2!  
*/  
  
int factorial(int n)  
{  
    if (n == 0)  
        return 1;  
    else  
        return n * factorial(n - 1);  
}
```



재귀 알고리즘

➤ 팩토리얼 계산하기

```
int main()
{
    int a, b, c;

    a = factorial(1); // 1 * factorial(0), 1 * 1 = 1
    b = factorial(2); // 2 * factorial(1), 2 * 1 = 2
    c = factorial(3); // 3 * factorial(2), 3 * 2 = 6

    printf("1!=%d, 2!=%d, 3!=%d\n", a, b, c);

    return 0;
}
```



재귀 알고리즘

➤ 십진수를 이진수로 변환하기

```
int printBin(int a){  
    if (a == 0 || a == 1)  
        printf("%d", a);  
    else{  
        printBin(a/2);  
        printf("%d", a%2);  
    }  
}
```

```
/*  
    a 값      몫      나머지  
    printBin(11), printBin(11/2), 5      1  
    printBin(5), printBin(5/2), 2      1  
    printBin(2), printBin(2/2), 1      0  
    printBin(1), printBin(1/2), 0      1  
    //아래서 위로 기록 - 1011  
    //가중치 방식  
    11 -> 8 4 2 1  
           1 0 1 1  
*/
```



재귀 알고리즘

- 십진수를 이진수로 변환하기

```
int main()
{
    int x = 11;
    printBin(x); //1011

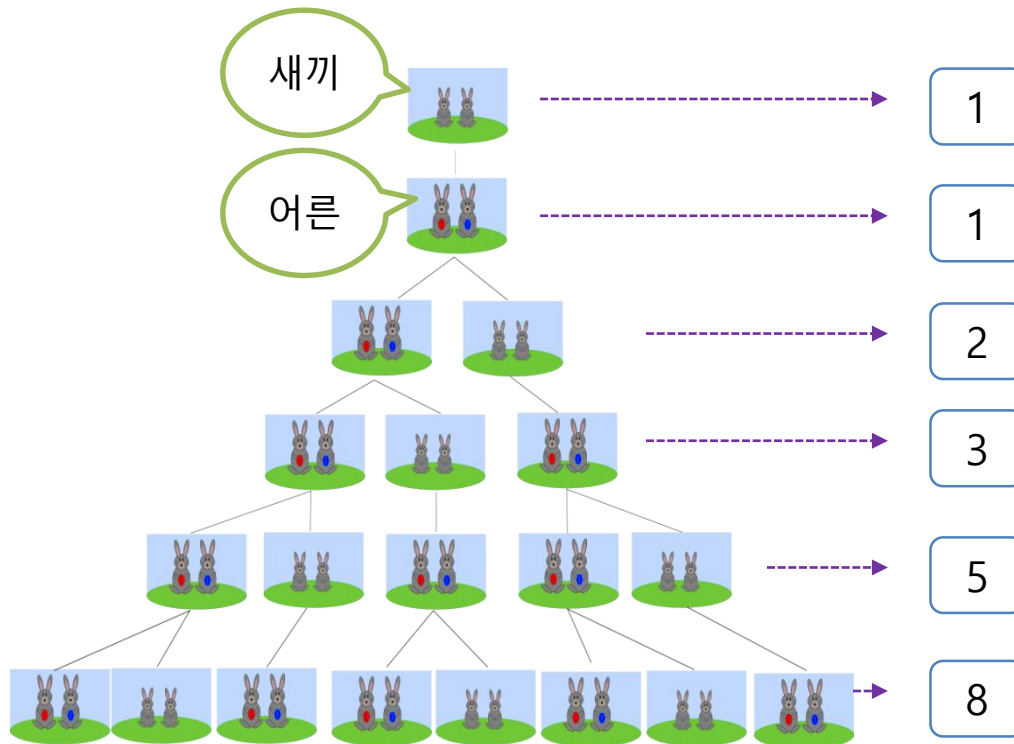
    return 0;
}
```



재귀 알고리즘

● 피보나치(Fibonacci) 수열

수학에서 피보나치 수는 첫째 및 둘째 항이 1이며, 그 뒤의 모든 항은 바로 앞 두 항의 합인 수열이다. 처음 여섯 항은 각각 1, 1, 2, 3, 5, 8이다.



첫번째 달에 새로 태어난 토끼 한쌍이 있고, 둘째달에 토끼가 커서 그대로 어른토끼 한쌍, 세째달에는 새끼를 한쌍 낳아 어른, 새끼 두쌍, 네째달에는 어른이 새끼를 낳고, 새끼는 어른이 되어 총 세쌍, 이렇게 계속 새끼를 낳고, 죽지 않는다는 가정을 세우면 피보나치의 수가 된다.