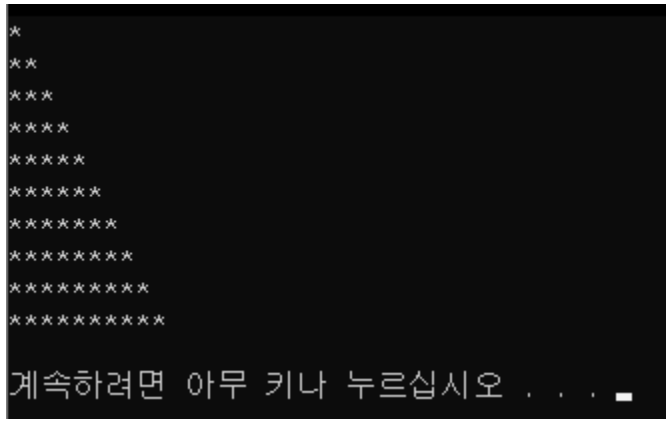
4주차 : 반복문 응용 및 기본 DP

강사: 장형준

오늘의 목표

이거(↘) 만들면서 반복문 연습하기
배열 써보기



별찍기

반복문 응용

이론 - 개념

반복문으로 변수의 크기를 1씩 증가시키며 실행할 수 있었다.

지금까지 반복문은 범위가 상수였다.

범위를 변수로 줄 수도 있다.

이론 - 개념

i 를 1부터 10까지 반복시킨다.

반복문 안에서

j 를 1 부터 i 까지 반복시킨다.

이론 – 코드

- for (int i = a; i <= b; i++)
  i 를 [a, b]에서 반복시키는 코드

- 별찍기 코드는 이렇게 쓸 수 있다.

```
for( int i = 1; i <= 10; i++ ) {
    for( int j = 1; j <= i; j++ ) {
        printf("*");
    }
    printf("\n");
}
```

이해 되신분들은 한번 짜보세요.

아닌 분들은 잠시 쉬었다 부연설명.

이론 – 부연설명 1

- 이렇게 동작합니다.
  - i = 1
  - 별 1개 출력하기
  - 엔터
  - i = 2
  - 별 2개 출력하기
  - 엔터
  - i = 3
  - 별 3개 출력하기
  - …

```
*
**
***
****
*****
******
*******
```

↑ 직접 따라가며 해보세요!

이론 - 부연설명2

- 겁 먹지 마시고 아래 코드를 찬찬히 이해해보세요.
  - for문을 배우셨으니 충분히 이해할 수 있어요!
  - 처음 하는 분들이 많이 어려워하는 부분이니 걱정 말고 파이팅!

```
for( int i = 1; i <= 10; i++ ) {
    별_i개_출력하기
    한_줄_띄우기
}
```

이론 – 부연설명 3

- 별 n개를 가로로 찍는 코드만 생각해봅시다.

```
for( int j = 1; j <= n; j++ ) {
    printf("*");
}
```

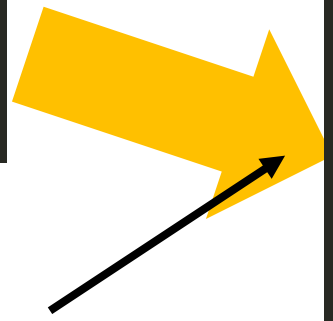
이론 – 부연설명 4

- 좋습니다! 이제 두 설명을 합쳐볼 거에요.

```
for( int i = 1; i <= 10; i++ ) {
    별_i개_출력하기
    한_줄_띄우기
}
```

```
for( int j = 1; j <= n; j++ ) {
    printf("*");
}
```

↑: 별을 n개 출력하는 방법



```
for( int i = 1; i <= 10; i++ ) {
    for( int j = 1; j <= i; j++ ) {
        printf("*");
    }
    printf("\n");
}
```

다음 챕터 전에
잠시 휴식

배열 써보기

반복문 응용

복습 - 배열

- 변수를 여러 개 만들기 귀찮을 때 쓴다.

- 배열이름[몇번째 칸] 으로 변수처럼 쓸 수 있다.
  - 예) arr[2]는 arr 배열의 2번째 칸
  - 컴소 특) 숫자는 0부터 센다.

실습 – 10개의 숫자 입력받기

- 아래처럼 할 수 있다.
  - 대괄호 안에 들어가는 숫자는 [0, 9]임에 유의한다.
  - arr[10]은 존재하지 않는다.

```
int arr[10];
for( int i = 0; i < 10; i++ ) {
    scanf( "%d", &arr[i] );
}
```

실습 – 누적 합 구하기 - 개괄

- “누적 합 구하기"라는 테크닉을 배울 것이다.
- 이 테크닉은 고급반에서도 써먹는다.
- 집중해서 보자.

그 전에 잠시 휴식

누적 합 구하기

반복문 응용

내용 이해 안 돼도 괜찮아요.
배열 써보는게 중요한 거니까.

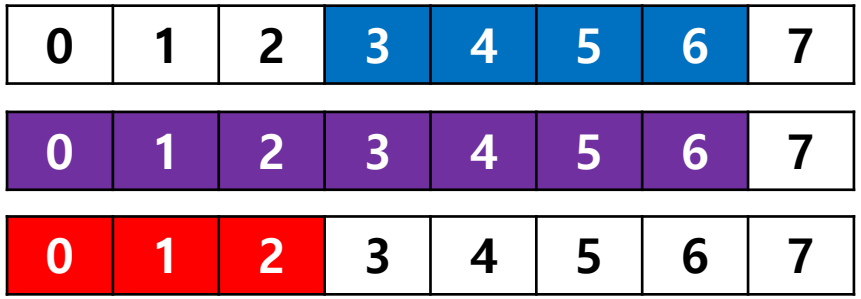
누적 합 구하기 - 문제상황

- n개의 숫자가 주어진다.
- a 번째 숫자부터 b 번째 숫자까지 총 합을 구하고 싶다.
- [a, b]를 모두 확인하는 것보다 빠른 방법은 없을까?

누적 합 구하기 - 아이디어

- $\sum_{i=a}^{b} i = \sum_{i=0}^{b} i - \sum_{i=0}^{a-1} i$

- [a, b]의 합은 [0, b]의 합에서 [0, a-1]의 합을 뺀 것과 같다.
  - 3,4,5,6번째 숫자의 합은
  - 0,1,2,3,4,5,6번째 숫자의 합에서
  - 0,1,2번째 숫자의 합을 뺀 것과 같다.



잠시 휴식

누적 합 구하기 – 앞으로 할 내용

- 새로운 배열 sum[N]을 만든다.
- sum[i] = arr[0 ~ i]의 합이다.
- 결과) 두 칸만으로 구간의 합을 구할 수 있게 된다.

누적 합 구하기 - 해보기

- 따라 써봅시다.
  - 어떤 일이 벌어지는지를 이해하는게 목표입니다.
  - 이 테크닉은 언젠가 다시 배울 테니까요.

```
int arr[15];
int sum[15];
for( int i = 1; i <= 10; i++ ) {
    scanf( "%d", &arr[i] );
}
for( int i = 1; i <= 10; i++ ) {
    sum[i] = sum[i-1] + arr[i];
}
```

누적 합 구하기 – 어떤 일이 벌어지는가?

- arr에는 입력된 숫자가 [1~10]까지 들어간다.
- sum[i]는 arr[0~i]까지의 누적합이다.

```
int arr[15];
int sum[15];
for( int i = 1; i <= 10; i++ ) {
    scanf( "%d", &arr[i] );
}
for( int i = 1; i <= 10; i++ ) {
    sum[i] = sum[i-1] + arr[i];
}
```

기본 Dynamic Programming

DP에 대해서 알아보자!

DP란?

영어 그대로 해석하면, Dynamic programming (동적 계획법) 이에요!

다양한 알고리즘을 통해서,
시간이나 메모리 제한이 있는 복잡한 문제를,
작은 하위문제 들로 나누어서,
여러가지 기법을 더하여 문제를 해결해 나가는 과정 또는 방식을 의미한답니다.

수학문제를 풀때처럼, 일종의 설계를 하는 과정으로 이해하면 좋을 거 같아요!

DP란?

DP의 방식은 매우 매우 다양해요.

그런데 대표적으로 두가지로 나뉘는데,

Top-Down 방식 : 큰 문제를 점점 작은 단위문제로 나누어서 푸는 방식
Bottom-up 방식: 작은 문제들을 해결하면서 점점 큰 문제에 접근하는 방식

이렇게 두가지 방식이 있는데, 우리는 이번시간에 Bottom-up 방식만 확인해 볼 거예요!

(이유를 말하자면, Top-Down의 재귀함수가 다음 주차에 다뤄지기 때문이에요!)

Fibonacci Sequence

Bottom-up을 통해 구현해 보자!

Bottom-up이란?

작은 문제들을 해결하면서 점점 큰 문제에 접근하는 방식이에요.

예를 들어서, 7번째 피보나치 수를 구하기 위해서
F(1), F(2) ··· F(5), F(6) 를 순서대로 구해 나가는 과정이 있겠네요.

그리고, **한번 구해낸 결과값**은 (F(5)라던가, F(6) 등등 말이죠.)
매번 다시 계산해서 구하지 않고, **배열에 저장해서 다시 꺼내 쓰면 빠르겠네요!**

(보통 for문을 통해 반복적으로 배열에 쌓아서 올라가요!)

Bottom-up이란?

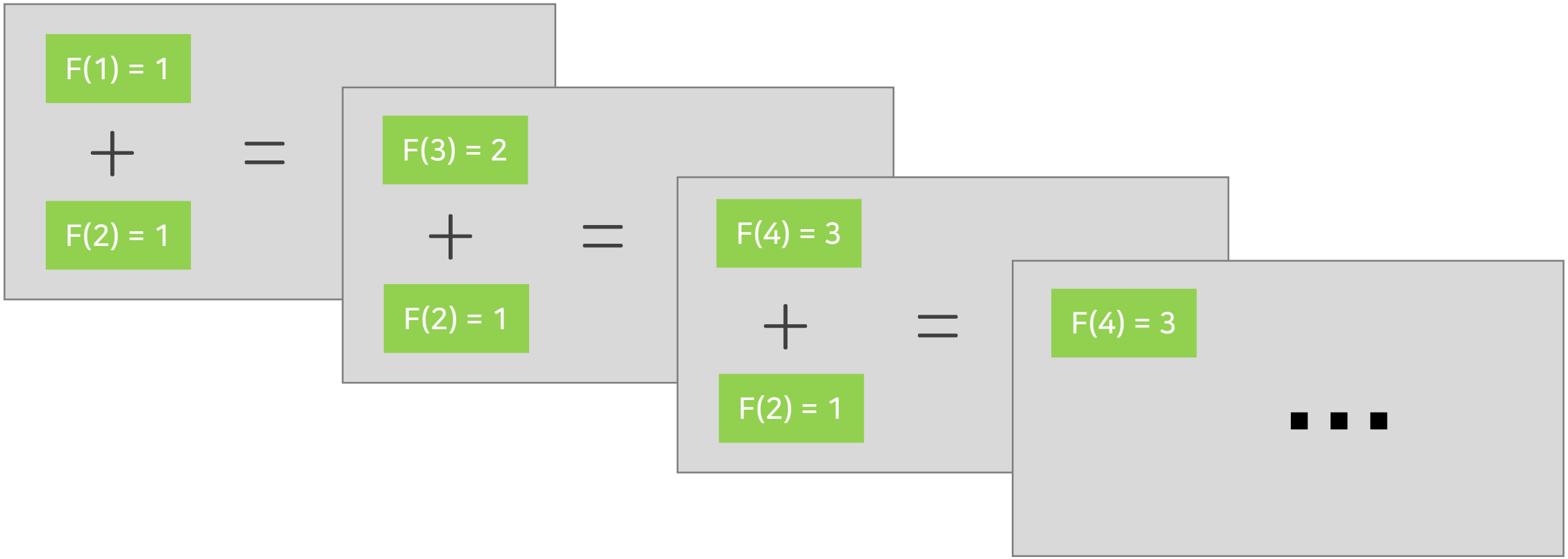
그렇게 N번째 값에 도달하기 위해, 우리는 수열의 항과 항의 관계를 알아야해요.
그전에 기초적으로 Fibo(1) = 1, Fibo(2) = 1 인 것은 누구나 알고있죠.

그럼 다음 항과의 관계는?

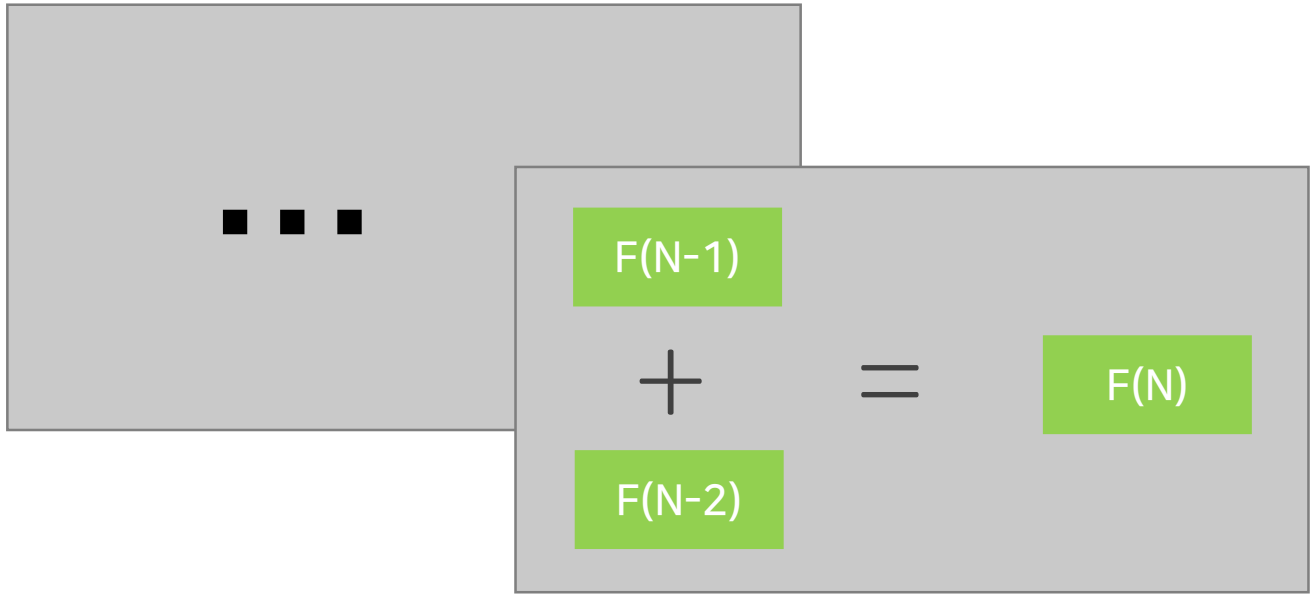
바로 점화식이라고 해요! (피보나치 수열에서는, F(N) = F(N-2) + F(N-1))

이 점화식을 통해 값을 차근차근 구해 나가서 N까지 가볼 거에요.

Bottom-up이란?



Bottom-up이란?



대략적으로 보았을 때 이러한 과정으로 진행됩니다!

Fibonacci Sequence 코드

```c
#include <stdio.h>

#define MM 10000
// MM이라는 문구는 10000으로 정의하겠다는 말이에요.

int Fibo[MM];
// int Fibo[10000]으로 치환된답니다.
// 또한 전역배열은 모든 칸이 0으로 초기화돼요!

int main()
{
    int N;
    scanf("%d", &N);
    // N번째 피보나치 값을 구하기 위해, N을 입력받아요.

    Fibo[1] = 1;
    Fibo[2] = 1;
    // 이 둘이 1이기 때문에, 다른 값을 구할 수 있었죠?
    // 기저조건이라고 불리는, 이러한 초기값들은 미리 저장해두는게 좋습니다!

    for (int i = 3; i <= N; i++)
    {
        Fibo[i] = Fibo[i - 1] + Fibo[i - 2];
        // i번째 피보나치 수는, 이전에 미리 구해둔 i-1, i-2번째 값을 가져와서 만들 수 있어요.
        // 매번 i-1, i-2번째 값을 새로 구하지 않고 배열에 담아둬서 가져오면 빠른 시간안에 구할 수 있겠죠?
        // i=3, i=4, ... , i=N까지 돌면서 우리가 원했던 N번째 피보나치 값을 구할 수 있겠네요!
    }

    printf("%d", Fibo[N]);
    // 지금은 작은 값을 보기 때문에, %d이지만
    // 더 큰 피보나치 수를 위해서는 long long 형식을 이용하여 %lld로 써야할 수도 있답니다.

}
```

코드로 보았을 때는 이러해요!

```
21      for (int i = 3; i <= N; i++)
22      {
23          Fibo[i] = Fibo[i - 1] + Fibo[i - 2];
24          // i번째 피보나치 수는, 이전에 미리 구해둔 i-1, i-2번째 값을 가져와서 만들 수 있어요.
25          // 매번 i-1, i-2번째 값을 새로 구하지 않고 배열에 담아둬서 가져오면 빠른 시간안에 구할 수 있겠죠?
26          // i=3, i=4, ... , i=N까지 돌면서 우리가 원했던 N번째 피보나치 값을 구할 수 있겠네요!
27      }
```

i = 3 일 때	Fibo[3] = Fibo[2] + Fibo[1]	Fibo[3] = 2의 값이 배열에 저장!
i = 4 일 때	Fibo[4] = Fibo[3] + Fibo[2]	Fibo[4] = 3의 값이 배열에 저장!
i = 5 일 때	Fibo[5] = Fibo[4] + Fibo[3]	Fibo[5] = 5의 값이 배열에 저장!
i = 6 일 때	Fibo[6] = Fibo[5] + Fibo[4]	Fibo[6] = 8의 값이 배열에 저장!
	...	
i = N 일 때	Fibo[N] = Fibo[N-1] + Fibo[N-2]	Fibo[N]의 값이 배열에 저장!

```
21      for (int i = 3; i <= N; i++)
22      {
23          Fibo[i] = Fibo[i - 1] + Fibo[i - 2];
24          // i번째 피보나치 수는, 이전에 미리 구해둔 i-1, i-2번째 값을 가져와서 만들 수 있어요.
25          // 매번 i-1, i-2번째 값을 새로 구하지 않고 배열에 담아둬서 가져오면 빠른 시간안에 구할 수 있겠죠?
26          // i=3, i=4, ... , i=N까지 돌면서 우리가 원했던 N번째 피보나치 값을 구할 수 있겠네요!
27      }
```

i = N 일 때	Fibo[N] = Fibo[N-1] + Fibo[N-2]	Fibo[N]의 값이 배열에 저장!

결론적으로, 점화식을 통해 반복문에서 모든 값을 구해냈고
우리가 원하는 Fibo[N]까지 구했어요!

DP_Bottom-up

Bottom-up의 대표적인 예시인 피보나치를 보면서 알게 된 사실.
우리가 DP문제를 Bottom-up방식으로 접근하려면?

초기 값
항과 항의 관계 (점화식)

이 두가지를 알아야 해요.

매우 중요한 두 요소이니까, 꼭 다양한 문제로 연습해서 스스로 익히길 바랍니다!

연습문제

피보나치 수

타일 장식물

피보나치 수 2

점화식

끝

부록 – while문으로 별찍기

- while도 반복문이다.

```
int i, j;
i = 1;
while ( i <= 10 ) {
    j = 1;
    while( j <= i ) {
        printf( "*" );
        j++;
    }
    i++;
    printf( "\n" );
}
```

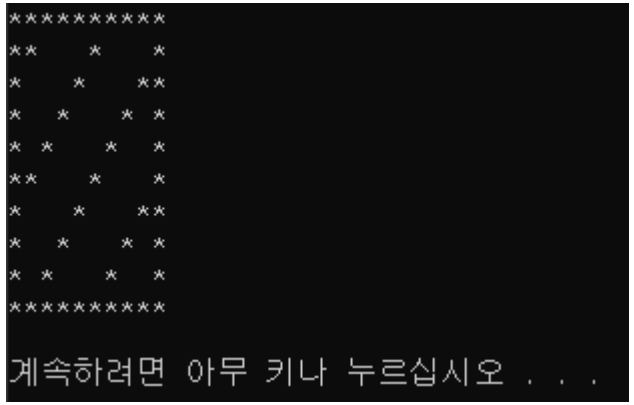
부록 – 수식으로 별찍기

- 그래프에서 부등식이 나타내는 범위로 찍는다.

```
for( int i = 1; i <= 10; i++ ) {
    for( int j = 1; j <= 10; j++ ) {
        if( i >= j ) printf("*");
    }
    printf("\n");
}
```

부록 – 수식으로 별찍기 응용

- 이런 것도 된다.




```
for( int i = 1; i <= 10; i++ ) {
    for( int j = 1; j <= 10; j++ ) {
        if( i == 1 || j == 1 || i == 10 || j == 10 || (i + j) % 4 == 0 )
            printf("*");
        else
            printf(" ");
    }
    printf("\n");
}
```