ALOHA
#2주차

반복문과 배열

#CH.1

반복문 (while문과 for문)

반복문이란 무엇일까?

컴퓨터 세계에서의 반복문이란 도대체 무엇일까요?

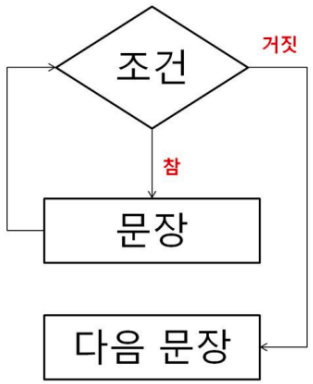
그리고 왜 필요할까요?

반복문이란 무엇일까?

“반복문”

명시된 조건이 만족될 때까지 반복실행 하는 명령문

while문



<while문 순서도>

```
while(조건식)
{
   문장;
}
다음 문장;
```

for문

반복문이 왜 필요할까?

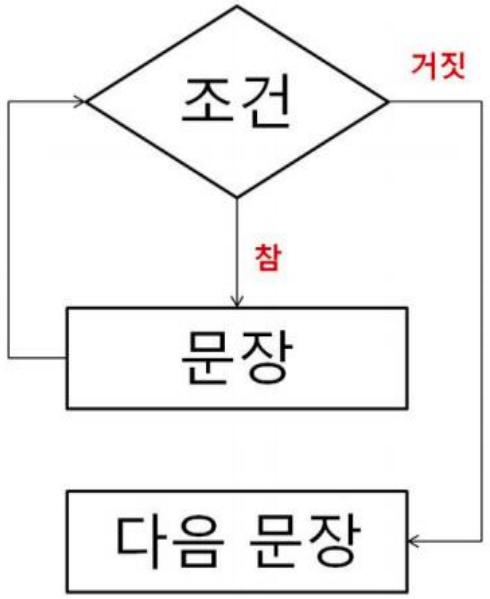
코드의 간결성

1부터 100까지 더하는 프로그램을 만든다고 해봅시다.

만약 1부터 100까지의 수를 직접 입력해서 더해주면 손가락이 정말 아프겠죠?

반복문은 코드를 간결하게 해주어 프로그래머의 손가락을 보호해 준답니다~!

while문



<while문 순서도>

```
while(조건식)
{
    문장;
}
다음 문장;
```

<문법>

http://codingrun.com

While문은 다음과 같은 방식으로 동작합니다.

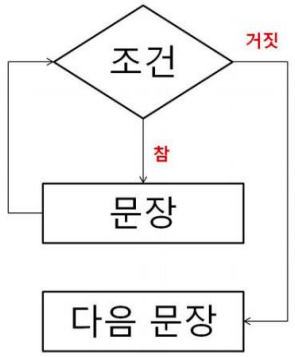
1. **조건** 부분의 내용이 **참**이라면 내용을 **실행**

2. 만약 **조건**이 **거짓**이라면 내용을 건너뛰고 **다음 문장**으로 넘어 감

3. 조건이 거짓이 될 때까지 즉, **조건이 참인 동안(while) 계속 반복**

```c
#include<stdio.h>

int main() {
    int num = 1;
    while (num < 6) {
        printf("%d\n", num);
        num++;
    }
    return 0;
}
```

이 조건에 따른 결과는??



<while문 순서도>

```
while(조건식)
{
   문장;
}
다음 문장;
```

<문법>

http://codingrun.com

while문을 직접 작성해 봅시다!

컴파일을 하기 전에 **while문의 작동 원리**를 생각해보면서 이 코드가 어떻게 **출력**될지 생각해보세요!

생각했던 것과 같은 내용이 출력 되었나요?
잘했습니다!

만약 틀리셨더라도 실망하지 마세요!
천천히 수업을 따라 오면서 이해하시면 됩니다!

```c
#include<stdio.h>

int main() {
    int num = 1;
    while (num < 6) {
        printf("%d\n", num);
        num++;
    }
    return 0;
}
```

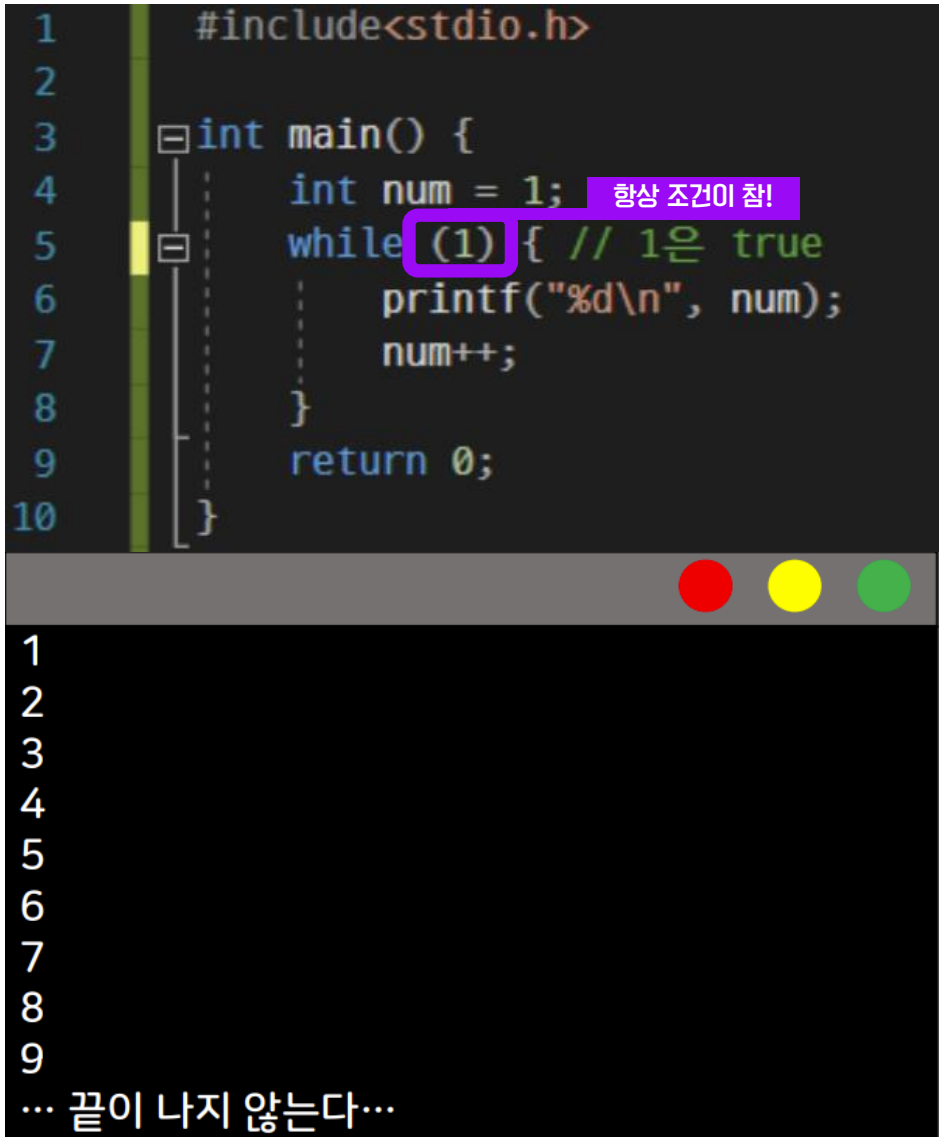
```
1
2
3
4
5
계속하려면 아무 키나 누르세요…
```

회차	초기값 (num)	변화값 (num++)
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6

먼저 지역 변수 num을 1로 초기화

반복 {
    각 회 차의 초기 num값 출력;
    num의 값 1 증가;
}

num이 6이 되면 조건이 거짓이므로
while문 탈출



break에 대해 알아 봅시다

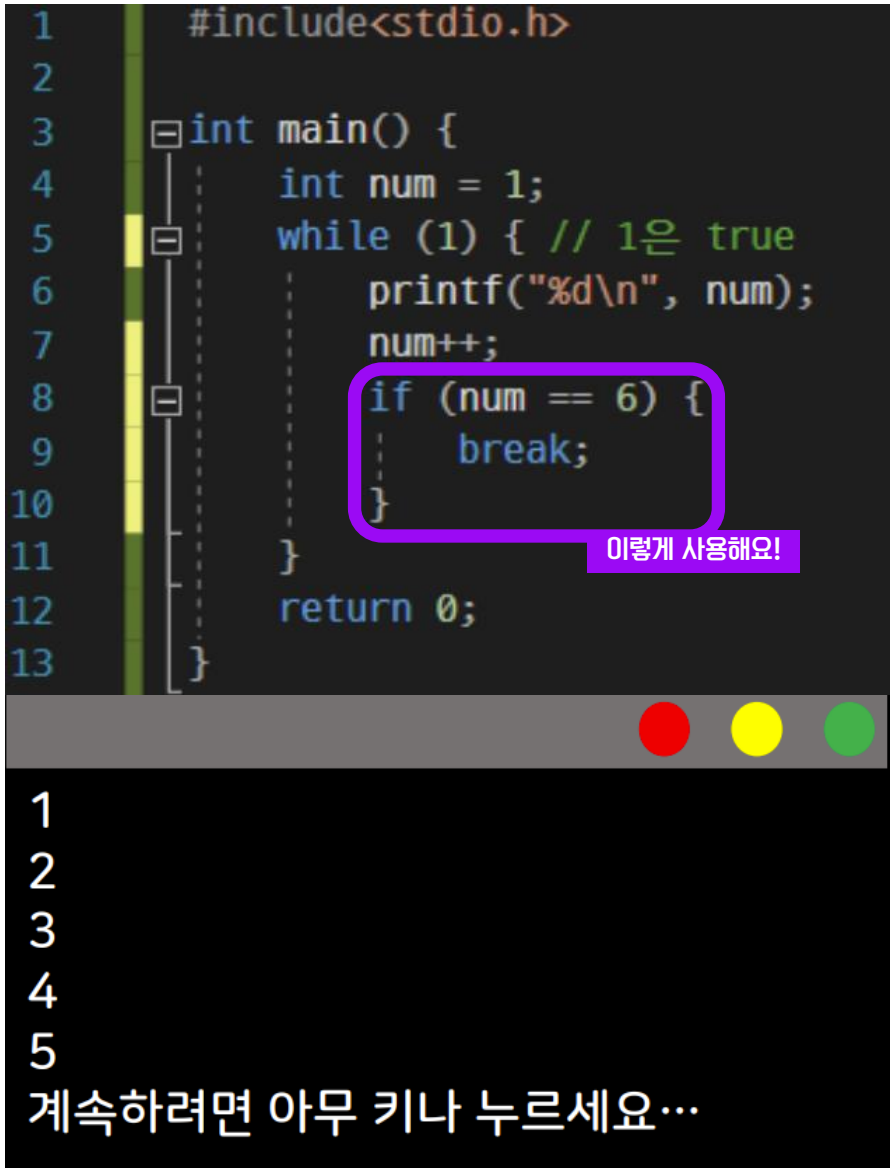
옆의 코드를 작성해서 실행하면 어떻게 될까요?

조건 부분에 들어간 1은 true를 뜻하기 때문에 while문은 **끝나지 않고 계속 반복**됩니다.

이렇게 while문을 **탈출하지 못하는 무한 루프** 상황은 우리가 코딩을 하면서 자주 겪게 될 오류 중 하나입니다.

그렇다면 이 무한 루프를 탈출하려면 어떻게 해야 할까요?

바로 **break**를 이용하면 됩니다!!!



break에 대해 알아 봅시다

옆의 코드를 보면 while문 안에 **if문**이 추가되었죠?

무한 루프를 **방지**하려면 특정 조건을 만족 할 때, 반복문을 빠져나갈 수 있도록 **break**를 해주어야 합니다.

다음 페이지에서는 문제를 풀어봅시다!

풀어볼까유

#10952
A + B - 5

덧셈, but 여러 개

while문

While문, 어땠나요?

이해가 안 간다면 앞에서 작성했던 코드들을 한 줄 씩 디버깅 (F11) (디버깅 자료 참고) 하면서 while문을 이해하도록 노력해 주세요!

다 이해하셨다면 for문으로 넘어가 봅시다!

for문

우선 for문은 초기식, 조건식, 증감식을 작성해 주어야 합니다.

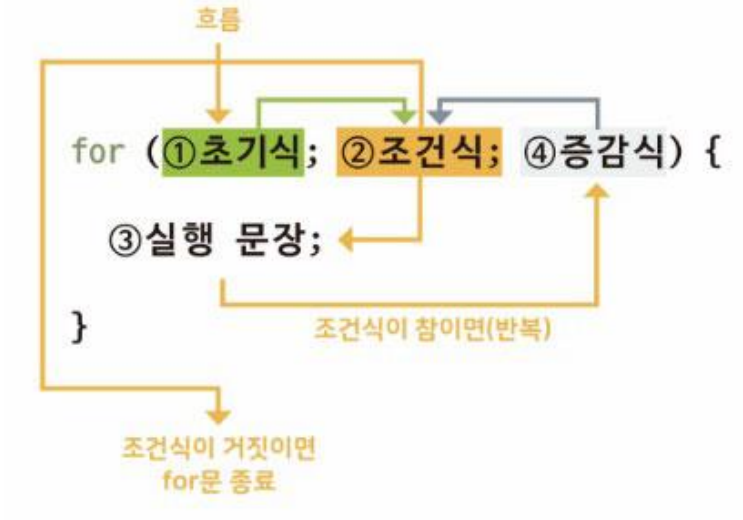
For문은 반복제어변수를 기준으로 작동합니다.

초기식: 반복제어변수에 초기값 대입

조건식: 참일 동안 실행문이 반복될 반복 조건 (while문의 조건과 동일)

증감식: 반복제어변수의 값을 증감

```c
#include<stdio.h>

int main() {
    for (int i = 1; i < 6; i++) {
        printf("%d\n", i);
    }
    return 0;
}
```

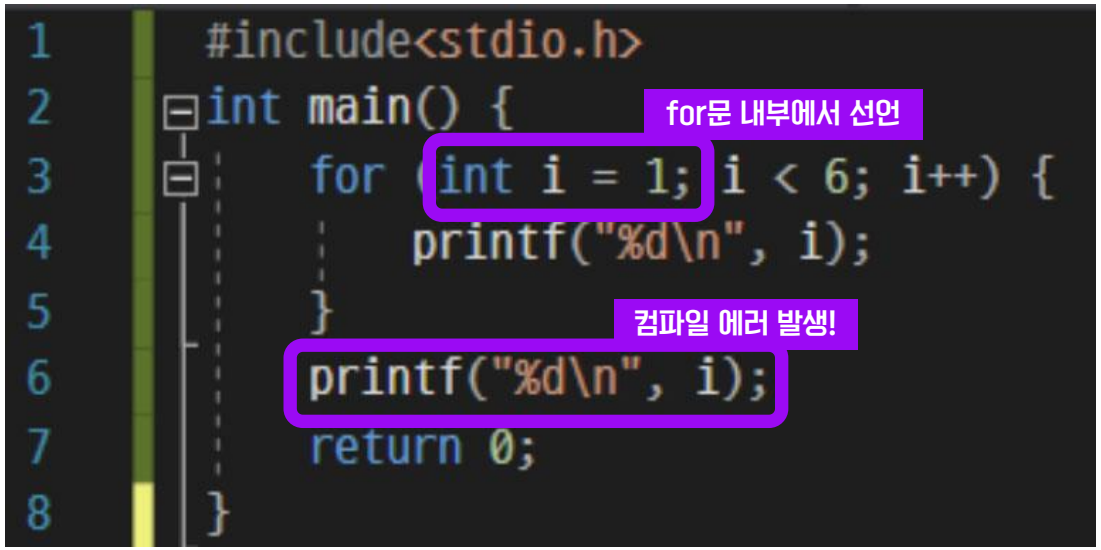


for문을 직접 작성해 봅시다!

컴파일을 하기 전에 **for문의 작동 원리**를 생각해보면서 이 코드가 어떻게 **출력**될지 생각해보세요!

while문을 이미 공부하셨으니 쉽게 맞출 수 있을 겁니다!

답은 간단하게 코드를 실행해보면 나오니 넘어가도록 하겠습니다.



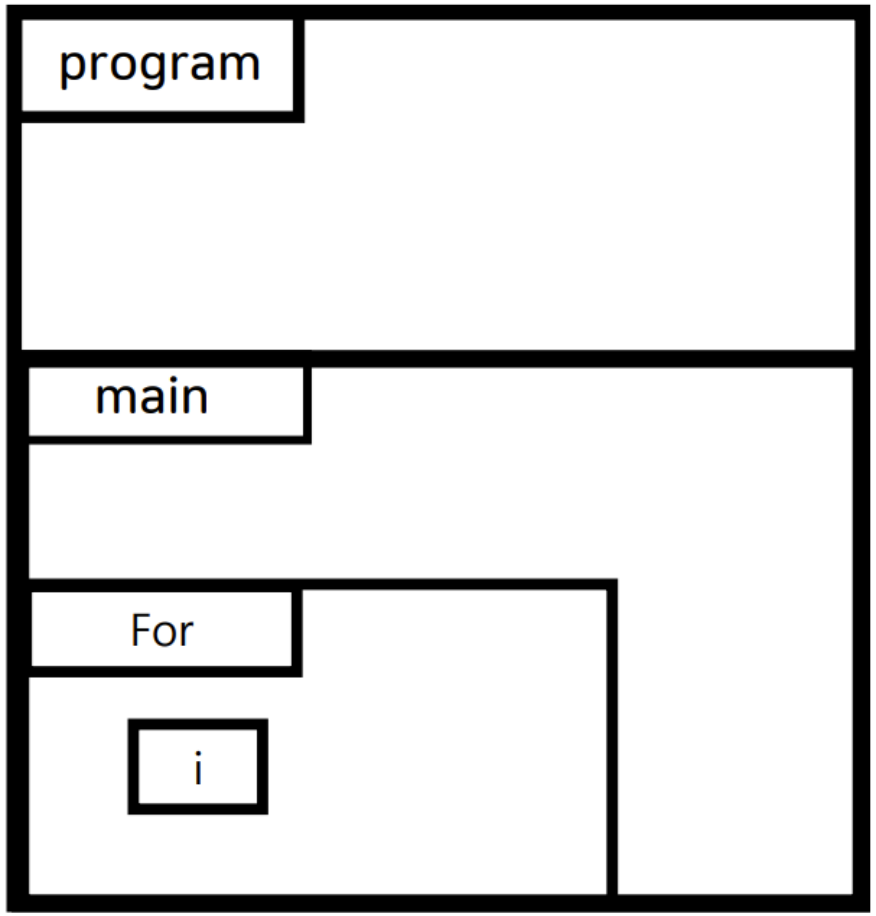
주의하세요!

이전 코드에서 한 줄이 추가됐습니다.
이 코드를 실행해보면 어떻게 될까요?

컴파일 에러가 뜨게 됩니다! (**왜??**)

반복제어변수인 **i**가 for문의 내부인 초기식에서 선언된 변수이므로, **for문 바깥**에서 **사라지기** 때문입니다.

for문

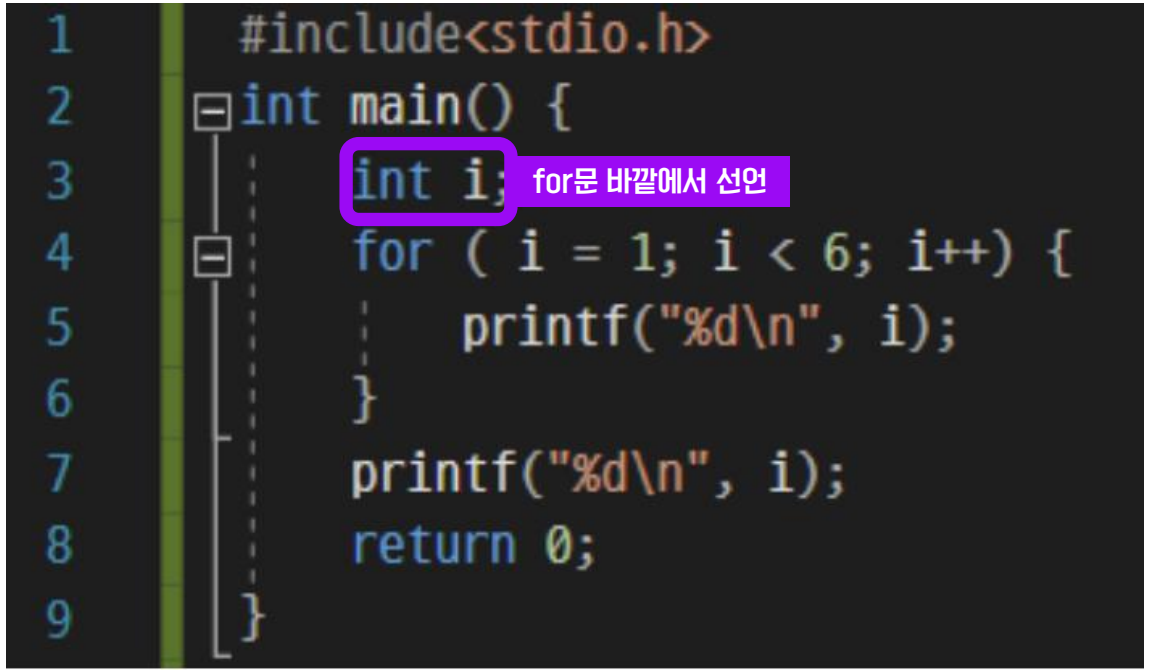


C언어에서 변수는 **선언된 위치 내에서만 존재**합니다. (**2주차 – 지역변수와 전역변수 참조**)

중괄호 내에서 선언된 지역변수는 해당 중괄호에서 **벗어나면 소멸**되어 접근할 수 없습니다.

여기에서 중괄호는 **함수, 조건문, 반복문 등**도 포함됩니다.

따라서 변수 i가 for문 안(초기식 부분 포함)에서 선언되면, **변수 i는 for문 안에서만 존재하며 접근 가능**합니다.



에러가 뜨지 않게 하려면 다음과 같이 작성하면 되겠죠?

변수 i를 **for문 바깥**에서 **선언**하여 for문이 종료되어도 i는 남아있게 하였습니다.

for문 활용하기 (주의사항)

다음 페이지부터는 for문을 활용하여 여러 문제들을 풀어볼 것입니다.
문제를 보고 코드를 작성할 때 **아래의 내용을 잘 숙지**하시고 코드를 작성해 주세요!

1. **변수 이름**을 잘 써야 합니다. (**누가 봐도 알 수 있게**)
   ex) 나무의 개수를 저장할 변수 선언 : int a (x) / int tree (o)

2. 머릿속으로만 생각하지 말고 **종이에 쓰면서** 생각합니다.
   (문제가 어려워질 수록 머리만으로 풀 수 있는 문제가 줄어듭니다. 미리 습관을 길러 둡시다.)

준비 됐다면 다음 페이지로!

풀어볼까유

#10872 팩토리얼

반복문을 이용한 팩토리얼의 계산

for문 활용하기 (팩토리얼)

for문을 활용하여 정수 n을 입력 받아 n 팩토리얼의 값을 출력하는 프로그램을 만들어 봅시다.

다음 페이지로 넘어가기 전 **직접** 만들어보고 넘어가도록 합시다!
잘 되지 않는다면 다음 페이지를 보면서 천천히 이해하도록 노력합시다.
(빠른 포기는 **절대** 안됩니다.)

```c
#include <stdio.h>

int main()
{
    int n;
    scanf("%d", &n);

    int fac = 1;
    for (int i = 1; 1 <= n; i++) {
        fac *= i;
    }

    printf("%d\n",fac);

    return 0;
}
```

n 팩토리얼을 출력하는 프로그램 코드 입니다.

여러분들도 이것과 비슷하게 코드를 짜셨나요?
잘하셨습니다.

코드를 작성하지 못하셨던 분들은 **옆의 코드**와
아래 표를 보고 천천히 **이해**하도록 노력해주세요!

For문이 진행되는 과정
(5를 입력 받았을 때)

i	1	2	3	4	5
fac	1	2	6	24	120

for문 활용하기 (2중 for문)

for문안에는 또 for문을 쓸 수 있습니다.

for문 안에 for문을 한번 더 쓰면 **2중 for문**이라고 합니다.

```c
#include <stdio.h>

int main() {

    for (int i = 1; i <= 3; i++) {
        for (int j = 1; j <= 3; j++) {
            printf("%d ",j);
        }
    }
    printf("\n");

    return 0;
}
```

```
1 2 3
1 2 3
1 2 3
```

옆의 코드를 따라서 작성해보세요! 그리고 컴파일을 하기 전 **for문의 작동 원리**를 생각해 보면서 이 코드가 어떻게 **출력**될지 생각해보세요!

2중 for문은 다음 시간에 제대로 다룰 예정입니다. 이번 시간에는 그냥 **2중 for문이 있다는 것을 알았다~** 정도만 공부하시면 됩니다.

반복문을 마치며

자 이제 반복문 파트는 끝났습니다!

이제 배열을 배울 것입니다.

배열을 반복문이 공부가 다 되었다는 가정 하에 진행되기 때문에 혹시 아직 공부가 안되었다고 생각하고 있다면, 복습을 하고 배열을 공부하는 것을 추천합니다.

#CH.2

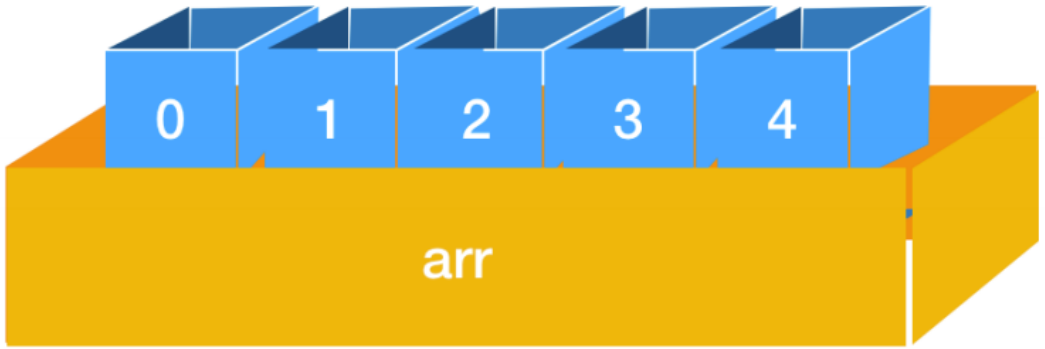
배열

배열이란 무엇일까?

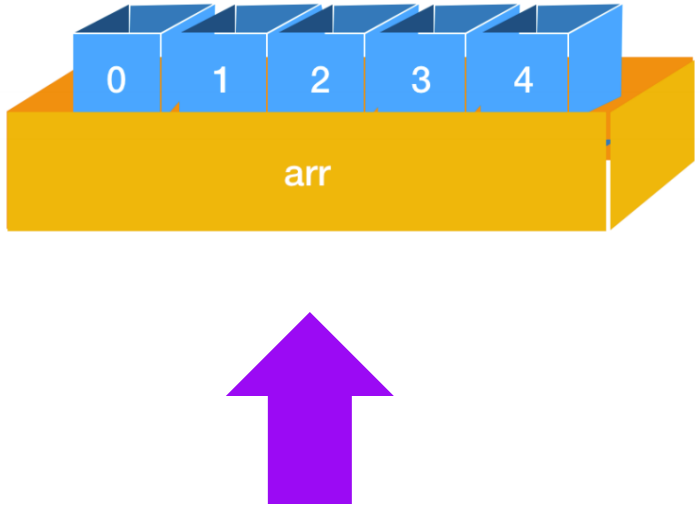
배열은 “같은 자료형을 가진 연속된 메모리 공간으로 이루어진 자료구조”입니다.

엥? 이게 무슨 말이냐구요?

사물에 빗대어 표현하자면 같은 유형의 물건을 담을 수 있는 연속된 상자들의 모임이라고 생각하면 이해하기 쉬울 것입니다.



배열이란 무엇일까?



같은 자료형으로만 구성!!!

자료형이란 이전에 배운 int, double 등
c언어에 존재하는 변수의 유형을 말합니다.

상자들이 연속되어 있으면 순서대로 번호를 매길 수 있듯이, 배열도 각자의 순서가 있어 번호를 이용해서 배열 하나하나에 접근할 수 있습니다.

여기서 주의할 점!)
c언어에서 배열의 시작번호는 0 입니다. (1아님!)
배열의 크기가 n 이면 번호는 n-1 까지 있겠죠?(0~n-1)

배열의 필요성 (간결성)

학생 100명의 소입설 점수가 주어진다고 합시다.

이를 다 저장해야 한다고 하면 지금까지 배운 내용으로는 왼쪽코드처럼 해야 합니다.

하지만 배열을 사용한다면? 오른쪽 코드처럼 아주 간결해 집니다.

```
int score0, score1, score2, score3, score4,
    score5, score6, score7, score8, score9, score10,
    score11, score12, score13, score14, score15,
    score16, score17, score18, score19, score20,
    score21, score22, score23 ……;

scanf("%d%d%d ……", &score0, &score1, &score2 ……);
```

```
1   #include <stdio.h>
2   int main() {
3
4       int scores[101];
5       for (int i = 0; i < 100; i++) {
6           scanf("%d", &scores[i]);
7       }
8
9       return 0;
10  }
```

배열의 필요성 (효율성)

학생의 점수를 저장한 후 원하는 학생의 점수를 출력한다고 합시다.

배열이 없을 때는 직접 코드를 입력해야 하지만,

배열을 사용하면 아래 두 경우처럼 **효율적**으로 학생들의 점수를 참조할 수 있습니다.

```
printf("%d", scores[원하는 사람]);
printf("%d", scores[원하는 사람2]);
```

```c
#include <stdio.h>
int main() {

    int scores[101];
    int num;
    for (int i = 1; i <= 2; i++) {
        scanf("%d", &num);
        printf("%d", scores[num]);
    }

    return 0;
}
```

배열의 선언과 초기화

배열의 선언과 초기화는 이전 시간에 배웠던 변수의 선언과 초기화와 매우 비슷합니다.
배열을 선언과 동시에 초기화 할 수 있고, 선언만 한 뒤 나중에 값을 대입할 수도 있습니다.

```
data_type name[SIZE];
```
: 배열의 선언 방식

```
7    int x = 5; // 변수의 선언과 초기화
8    int y; // 변수의 선언
9    y = 5; // 변수의 대입
```

변수의 선언과 초기화

```
11   int arr[3] = { 1,3,5 } // 배열의 선언과 초기화
12   int arr2[3]; // 배열의 선언
13   arr2[0] = -1; // 배열의 0번 원소에 대입
14   arr2[1] = 2; // 배열의 1번 원소에 대입
```

배열의 선언과 초기화

```c
#include <stdio.h>

int arr_global[3]; // 배열의 선언 (전역 변수)

int main() {

    int arr[3] = { 1,3,5 } // 배열의 선언과 초기화

    int arr2[3]; // 배열의 선언
    arr2[0] = -1; // 배열의 0번 원소에 대입
    arr2[1] = 2; // 배열의 1번 원소에 대입

    return 0;
}
```

이전 시간에 **전역 변수와 지역 변수의 차이점**을 배웠습니다. 옆의 코드를 보고 다음 질문에 대해 생각해보세요

(배열이 변수와 유사점이 많다는 것을 생각하면 쉽게 알 수 있을 거예요)

1. arr_global의 각 원소에는 무슨 값이 들어가 있을까?

2. arr의 각 원소에는 무슨 값이 들어가 있을까?

3. arr_local의 각 원소에는 무슨 값이 들어가 있을까?

다 생각해 보셨다면 다음페이지로 넘어가셔서 자신의 생각과 같은 지 비교해 보세요!

```c
#include <stdio.h>

int arr_global[3]; // 배열의 선언 (전역 변수)

int main() {

    int arr[3] = { 1,3,5 } // 배열의 선언과 초기화

    int arr_local[3]; // 배열의 선언
    arr_local[0] = -1; // 배열의 0번 원소에 대입
    arr_local[1] = 2; // 배열의 1번 원소에 대입

    printf("전역 배열: ");
    for (int i = 0; i < 3; i++) {
        printf("%d ",arr_global[i]);
    }
    printf("\n지역 배열 (arr): ");
    for (int i = 0; i < 3; i++) {
        printf("%d ",arr[i]);
    }
    printf("\n지역 배열 (arr_local): ");
    for (int i = 0; i < 3; i++) {
        printf("%d ",arr_local[i]);
    }

    return 0;
}
```

옆의 코드를 따라서 작성하고 실행해 보세요!

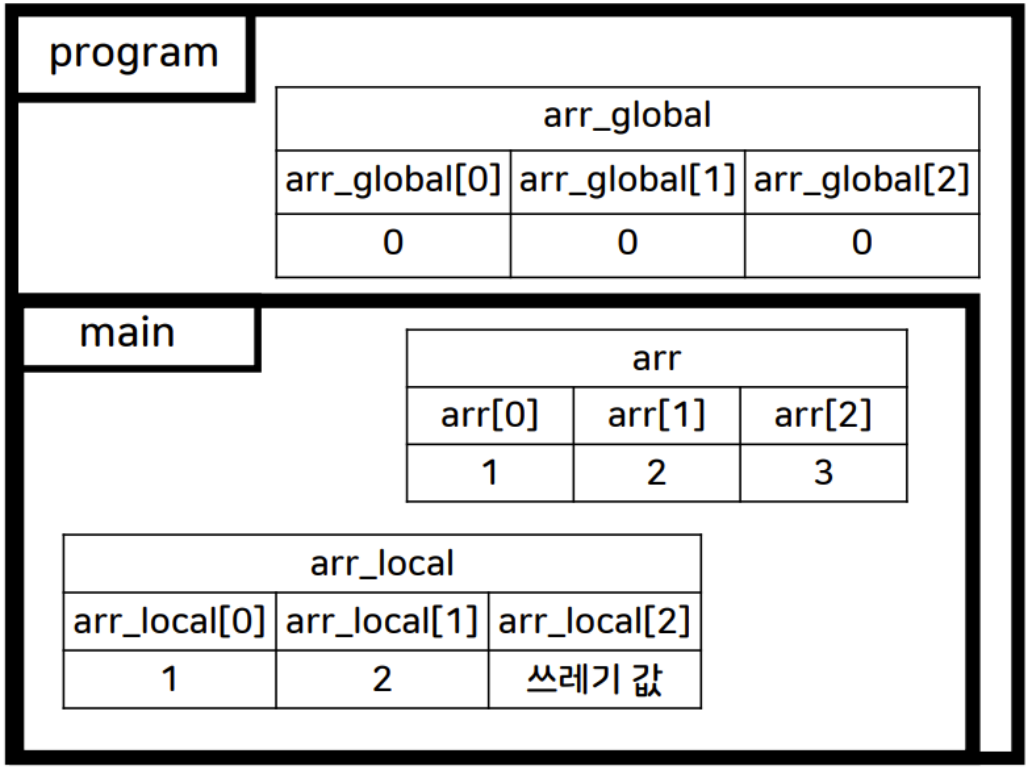
아마 arr_local 배열의 인덱스 2번 값에서 **이상한 값**이 출력될 것입니다.

지역 배열도 지역 변수처럼 **값을 정의하지 않으면 쓰레기 값**이 들어가기 때문입니다!

(배열의 출력 방법은 조금 이따가 배우니 배열 안의 값에 집중!)

배열의 선언과 초기화 (생각했던 것과 같나요?)

```c
#include <stdio.h>

int arr_global[3]; // 배열의 선언 (전역 변수)

int main() {

    int arr[3] = { 1,3,5 } // 배열의 선언과 초기화

    int arr_local[3]; // 배열의 선언
    arr_local[0] = -1; // 배열의 0번 원소에 대입
    arr_local[1] = 2; // 배열의 1번 원소에 대입

    return 0;
}
```



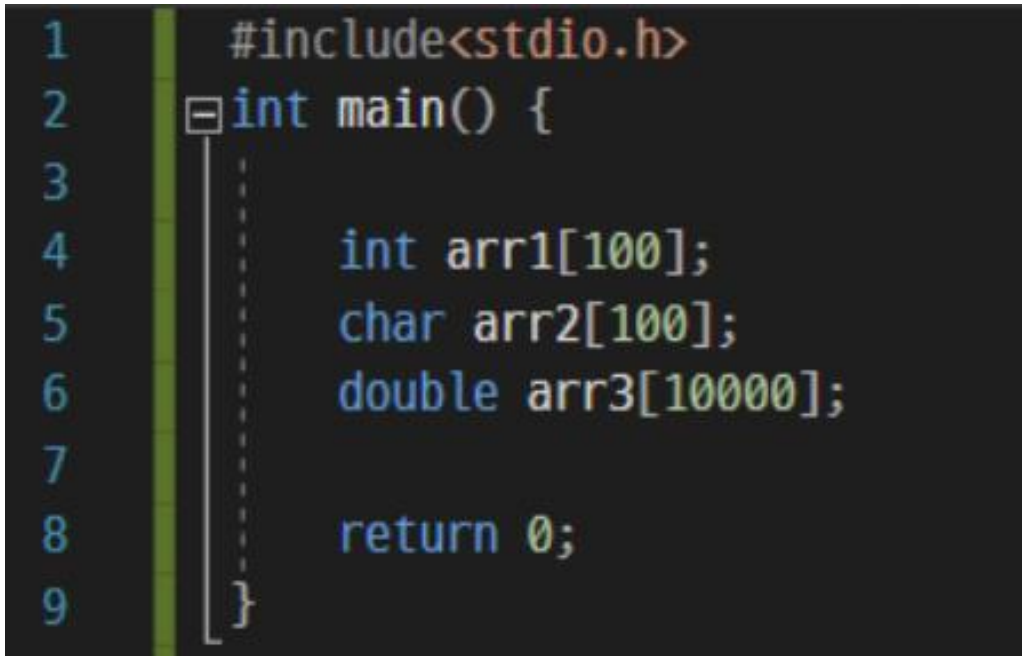
전역배열은 값을 정의 하지 않으면 **0으로 초기화**!

지역배열은 값을 정의 하지 않으면 **쓰레기 값**이 들어갑니다!

변수의 자료형 Review

변수의 자료형으로 쓸 수 있는 모든 자료형들은 배열을 선언할 때 자료형으로 쓸 수 있습니다.

유형	자료형	크기(byte)	포맷 형태	범위
정수	int	4	%d	–2,147,483,648 ~ 2,147,483,647
	unsigned int	4	%u	0 ~ 4,294,967,295
	long long	8	%lld	–9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
	short	2	%d	-32,768 ~ 32,767
실수	float	4	%f	3.4E+/-38(7개의 자릿수)
	double	8	%lf	1.7E+/-38(7개의 자릿수)
문자(문자열)	char	1	%c(%s)	-128~127(15개의 자릿수)
논리(T/F)	bool	1	-	-



다양한 종류의 배열 만들기

특정한 자료형의 변수를 원소로 가지는 **여러 종류의 배열**을 만들어 볼 거예요

옆의 코드를 보고 **여러 자료형의 배열**을 **선언**해 봅시다!

그리고 다른 배열들도 직접 만들어 보세요!!

배열 입출력 (I/O)

배열 입출력은 변수 입출력 방식과 동일합니다!

scanf(“%d”, &arr[번호]); -입력 받은 정수를 arr[번호]의 주소에 저장

printf(“%d”,arr[번호]); - arr[번호] 안에 있는 값을 출력

배열 입출력 (I/O)

질문) arr[0] 부터 arr[99] 까지 입력을 받으려면 어떻게 해야 할까요?

scanf(“%d”,&arr[0]), scanf(“%d”,&arr[1]) … scanf(“%d”,&arr[99])
이렇게 해야 할까요?

물론 그럴 수도 있지만 for문을 쓰면 간편하게 입력 받을 수 있습니다.

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

배열을 입력 받아봅시다

옆의 코드처럼 배열을 입력 받으면 0 부터 N-1 까지의 배열을 입력 받을 수 있습니다.

참 쉽죠?

문자열

문자열이란 문자의 배열입니다.

즉, char형 변수들이 나열되어 있는 구조이죠.

문자열의 선언과 초기화

다음과 같이 다른 배열들처럼 문자열을 선언하고 초기화할 수 있습니다.

```c
#include<stdio.h>
int main() {
    char str[5] = { 'H','i' };

    char str1[5] = { 'H','e','l','l','o' };

    return 0;
}
```

Str[]	[0]	[1]	[2]	[3]	[4]
Char	'H'	'I'	0	0	0

Str1[]	[0]	[1]	[2]	[3]	[4]
Char	'H'	'e'	'l'	'l'	'o'

문자열의 선언과 초기화

문자열은 다른 배열들과 다르게 “”(쌍 따옴표)를 이용해 초기화할 수 있습니다.
이렇게 할 때에는 마지막 문자 다음에 ‘\0‘ (NULL) 값이 들어가게 됩니다.

```c
#include <stdio.h>

int main() {

    char str1[6] = { 'H', 'e', 'l', 'l', 'o', '\0' };
    char str2[6] = "Hello";
    char str3[5] = "Hello";

    return 0;
}
```

‘\0’을 넣을 공간이 없어 에러!

str1[]	[0]	[1]	[2]	[3]	[4]	[5]
char	‘H’	‘e’	‘l’	‘l’	‘o’	‘₩0’

str2[]	[0]	[1]	[2]	[3]	[4]	[5]
char	‘H’	‘e’	‘l’	‘l’	‘o’	‘₩0’

str2[]	[0]	[1]	[2]	[3]	[4]	
char	‘H’	‘e’	‘l’	‘l’	‘o’	‘₩0’

문자열의 선언과 초기화 (NULL == '\0' == 0)

NULL ('\0')은 값이 0인 제어 문자입니다.

문자열에서는 문자열의 끝을 알리기 위해 사용됩니다.
문자열을 선언하고 초기화할 때는 항상 NULL을 위한 공간을 잊지 마세요!

0	1	2	3	4	5
'A'	'L'	'O'	'H'	'A'	'₩0'

"ALOHA"는 길이가 5인 문자열이고,
배열에 저장하기 위해서는 6칸이 필요하죠

참고) ‘’ (홑 따옴표)와 “” (쌍 따옴표) 의 차이

‘’ (홑 따옴표)

- 문자 하나를 의미
- 그 문자의 아스키코드 값을 의미
  ex) ‘\0’ == 0, ‘A’ == 65
- 주로 char형 변수를 초기화할 때 사용
  ex) char chr = ‘A’;
- 1 byte

“” (쌍 따옴표)

- 문자열을 의미
- 그 문자열의 메모리 주소 값을 의미
  (소입설 시간 ‘포인터’에서 자세히 배움)
- char형 배열을 초기화할 때 사용
  ex) char str[10] = “ALOHA”;
- 문자열의 길이+1 byte (+1은 NULL)

문자열의 입출력 (I/O)

문자열은 여러 입출력 방식이 있습니다.

1. 문자 단위로 입출력 할 수 있고
2. 문자열을 한번에 입출력 할 수 도 있습니다.

```c
scanf("%c", &str[0]);
for ( i = 1; str[i - 1] != '\n'; i++ )
{
    scanf("%c", &str[i]);
}
```

```c
for ( i = 0; str[i] != '\n'; i++ )
{
    printf("%c", str[i]);
}
```

for문의 **조건식**을 이용하여 **문자 하나하나** 씩을 받고 출력할 수 있습니다.

```c
char str[10];

scanf("%s", str);
printf("%s", str);
```

&str이 아니어도 됨

“%s” 를 사용하여 **문자열**을 한번에 입력 받을 수 있습니다.

“%s”는 **공백문자**(‘ ’, ‘\n’, ‘\t’, ‘\0’ 등등) **전까지** 입력 받습니다.

마지막 문자 뒤에는 **‘\0‘**을 저장해 줍니다.

출력은 **NULL**까지 합니다.

```c
char str[10];

scanf("%s", str);
printf("%s", str);
```

&str이 아니어도 됨

문자열을 scanf로 한번에 입력 받을 때에는 일반적으로 &str이 아니라 str과 같이 씁니다.

'&변수명' 은 그 변수의 메모리 주소 값을 의미하는데, 문자열과 같은 배열들은 그 이름 자체가 배열의 메모리 시작 주소 값을 의미하기 때문입니다.

(소입설에서 배열과 포인터의 관계에 대해 자세히 배웁니다.)

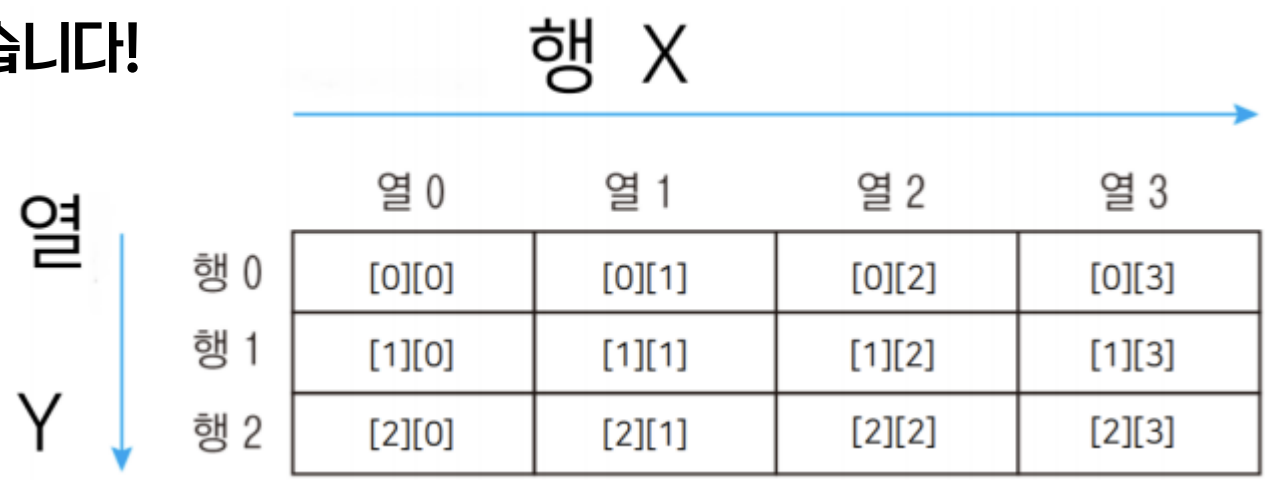
배열 활용하기 (2차원 배열)

2차원 배열이란?

배열 안에 배열이 들어간 형태를 말합니다.

(x,y)가 arr[y][x]와 같이 표현되는

2차원 좌표 평면이라고 생각할 수 있습니다!



배열 활용하기 (2차원 배열)

```
double arr[80][60]; // 2차원 배열의 선언
int arr2[3][2] = {{1, 2}, {3, 4}, {5, 6}}; // 2차원 배열의 선언과 초기화
```

위와 같이 2차원 배열을 선언하고 초기화할 수 있어요!
필요에 따라 n차원 배열을 만들어 사용할 수도 있습니다

입출력 방법은 1차원 배열과 동일하지만 인덱스([i])가 두개 필요하다는 걸 잊지 말아요

배열을 마치며

자 이제 배열에 관한 기본적인 설명은 끝이 났습니다.

다음 페이지부터는 지금까지 배운 내용을 토대로 문제를 풀어보는 시간을 가지도록 하겠습니다.

for문 활용하기에서 설명 드렸던
변수 이름 설정과 종이에 쓰기를 적절히 활용하여 문제를 풀어보세요!

풀어볼까유

#5597

과제 안 내신 분..?

배열 선언과 입출력

배열의 활용 (BOJ #5597)

```c
#include<stdio.h>

int students[31]; // 배열은 30보다 크게 잡아주어야 합니다.

int main() {
    int i;
    int bunho; // 학생들의 번호
    for (i = 0; i < 28; i++) {
        scanf("%d", &bunho);
        students[bunho] = 1; // 출석이면 1을 저장
    }
    for (i = 1; i <= 30; i++) { // 1부터 30까지 보고 student[i]가 0이면 결석
        if (students[i] == 0) {
            printf("%d\n", i); // 결석자 번호 출력
        }
    }
    return 0;
}
```

풀어볼까유

#3052 나머지

배열을 어떻게 이용할까?

배열의 활용 (BOJ #3052)

```c
#include<stdio.h>

int nanu[43]; // A를 B로 나눈 나머지를 저장합니다.

int main() {
    int N;
    for (int i = 1; i <= 10; i++) { // 10번 입력을 받습니다.
        scanf("%d", &N);
        nanu[N % 42] = 1; // 나머지의 값의 존재 유무를 배열에 저장합니다.(존재하면 1)
    }
    int stack = 0; // 서로 다른 나머지가 몇 개 있는지 저장합니다.
    for (int i = 0; i <= 42; i++) {
        if (nanu[i] == 1) {
            stack++;
        }
    }
    printf("%d", stack); // 서로 다른 값을 출력합니다.
    return 0;
}
```

다음 시간에 만나요~