

3주차: 반복문과 배열

강사 : 우철

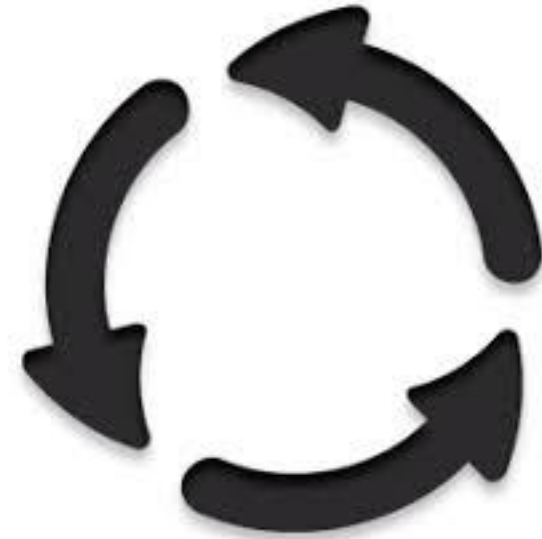
목차

1. 반복문

1. 반복문이 필요한 이유
2. While 문
3. For 문

2. 배열

1. 배열이 필요한 이유
2. 배열의 선언과 정의
3. 배열의 입출력(I/O)
4. 문자열



챕터 1: 반복문

반복문이 무엇인지 그리고 반복문의 대표, for문과 while문에 대해서 알아보자!

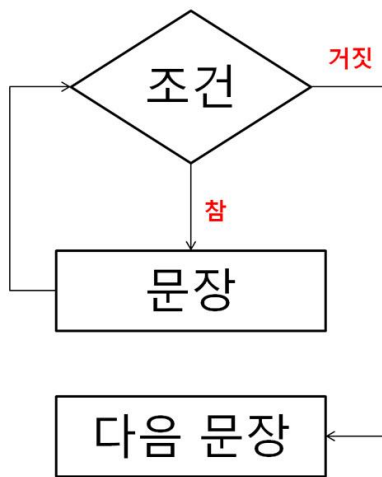
반복문이란 무엇일까?

컴퓨터 세계에서 **반복문**이란 도대체 무엇일까요?

그리고 **왜 필요**할까요?

반복문 : 명시된 조건이 만족될 때까지 반복실행 하는 명령문

while 문



<while문 순서도>

```
while(조건식)
{
    문장;
}
다음 문장;
```

for 문



```
for (①초기화식; ②조건식; ④증감식) {
    ③실행 문장;
}
```

반복문이 필요한 이유

- 코드의 간결성

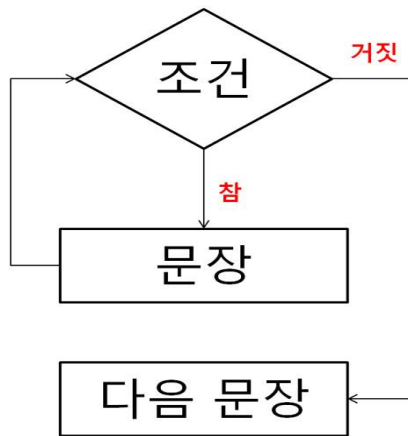
1부터 100까지 더하는 프로그램을 만든다고 해봅시다.

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

반복문은 코드를 **간결**하게 해주어 프로그래머의 손가락을 보호해 준답니다~!

자 이제 본격적으로 **반복문**을 공부해 봅시다!

While문



<while문 순서도>

```
while(조건식)
{
    문장;
}
다음 문장;
```

<문법>

<http://codingrun.com>

While 문은 다음과 같은 방식으로 동작합니다.

1. **조건** 부분의 내용이 **참**이라면 내용을 **실행**해 줍니다.
2. 만약 조건이 **거짓**이라면 내용을 건너뛰고 **다음 문장**으로 넘어가게 됩니다.
3. 조건이 거짓이 될 때까지 계속 반복합니다.

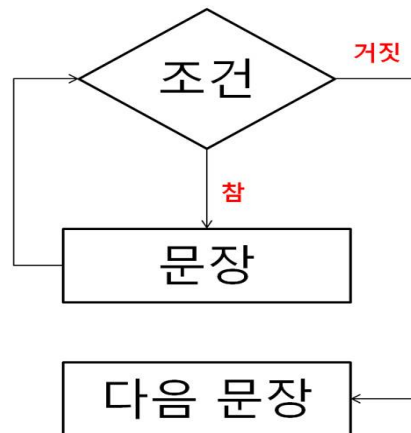
While문(직접 작성해보기)

```
1  #include<stdio.h>
2
3  int main() {
4      int num = 1;
5      while (num < 6) {
6          printf("%d\n", num);
7          num++;
8      }
9      return 0;
10 }
```

옆에 코드를 한번 따라서 작성해보세요!

그리고 컴파일을 하기 전 **while문의 작동 원리**를 생각해 보면서 이 코드가 어떻게 **출력**될지 생각해 보세요!

생각했던 것과 같은 내용이 출력 되었나요? 잘했습니다!



<while문 순서도>

```
while(조건식)
{
    문장;
}
다음 문장;
```

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

While문(직접 작성해보기)

```
1  #include<stdio.h>
2
3  int main() {
4      int num = 1;
5      while (num < 6) {
6          printf("%d\n", num);
7          num++;
8      }
9      return 0;
10 }
```

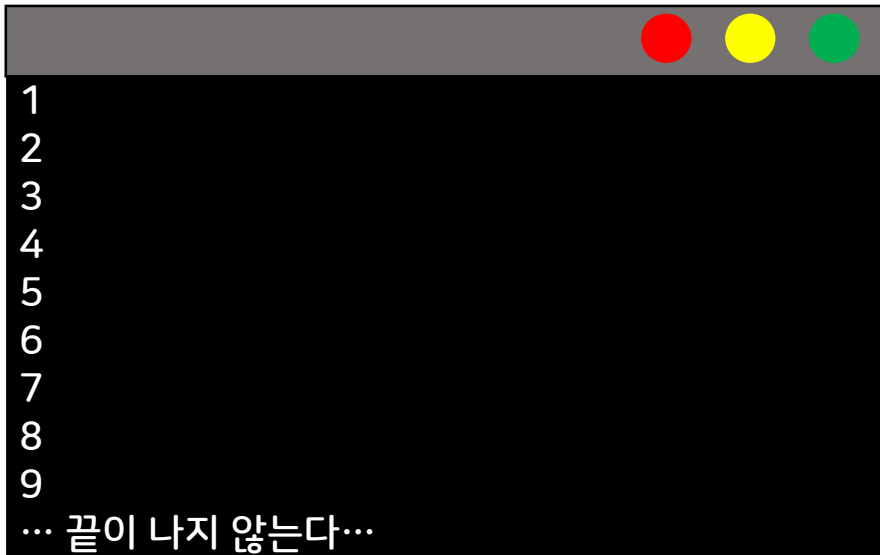
1
2
3
4
5
계속하려면 아무 키나 누르세요...

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

처음 1로 들어갔던 num이 커지면서 6이 되는 순간 탈출!

While문(break에 대해 알아보자)

```
1  #include<stdio.h>
2
3  int main() {
4      int num = 1;
5      while (1) { // 1은 true
6          printf("%d\n", num);
7          num++;
8      }
9      return 0;
10 }
```



A terminal window with a black background and white text. The window has a title bar with red, yellow, and green buttons. The text shows a list of numbers from 1 to 9, followed by an ellipsis and the text "... 끝이 나지 않는다..." (It doesn't end...), indicating an infinite loop.

```
1
2
3
4
5
6
7
8
9
... 끝이 나지 않는다...
```

옆에 코드를 작성해서 실행시키면 어떻게 될까요?

조건 부분에 들어간 1은 true를 뜻하기 때문에 while문은 끝나지 않고 반복되게 됩니다.

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

그러면 이 상황을 탈출하려면 어떻게 해야 할까요?
(무한루프)

바로 **break** 를 이용하면 됩니다!!!

While문(break에 대해 알아보자)

```
1  #include<stdio.h>
2
3  int main() {
4      int num = 1;
5      while (1) { // 1은 true
6          printf("%d\n", num);
7          num++;
8          if (num == 6) {
9              break;
10         }
11     }
12     return 0;
13 }
```

옆의 코드를 보면 이전 코드에 **if문**이 추가 되었죠?

무한 루프를 **방지**하려면 특정 조건이 되면 반복문을 빠져나갈 수 있도록 **break**를 해주어야 합니다.

```
1
2
3
4
5
계속하려면 아무 키나 누르세요...
```

While문

while문, 어땠나요?

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

다 이해하셨다면 For문으로 넘어가 봅시다!

For문



우선 For문은 초기화식, 조건식, 증감식을 작성해 주어야 합니다.

초기화는 무슨 뜻 일까요?

이전 시간의 변수의 선언과 정의 중에서
정의와 같은 의미라고 생각하시면 됩니다!

쉽게 말하자면 처음에 한번 값을 저장해 준다는 뜻이죠

For문



For문의 실행순서는 다음과 같습니다.

① 초기화식이 실행 된다.

(반복문이 제일 처음 시작할 때 **한번만** 초기화 해줍니다.)

② 조건식이 **true**이면 ③ 으로 간다. **false** 면 for문을 종료한다.

③ 실행문을 실행 시킨다.

④ 증감식을 실행 시킨 뒤 ②의 순서로 간다.

For문

```
1  #include<stdio.h>
2
3  int main() {
4      for (int i = 1; i < 6; i++) {
5          printf("%d\n", i);
6      }
7      return 0;
8  }
```

옆의 코드를 따라서 작성해보세요!

그리고 컴파일을 하기 전 **for문의 작동 원리**를 생각해 보면서 코드가 어떻게 **출력**될지 생각해 보세요!

While문을 이미 공부하셨으니 쉽게 맞출 수 있을 겁니다!

답은 간단하게 코드를 실행해보면 나오니 넘어가도록 하겠습니다.

For문

```
1  #include<stdio.h>
2  int main() {
3      for (int i = 1; i < 6; i++) {
4          printf("%d\n", i);
5      }
6      printf("%d\n", i); (에러!)
7      return 0;
8  }
```

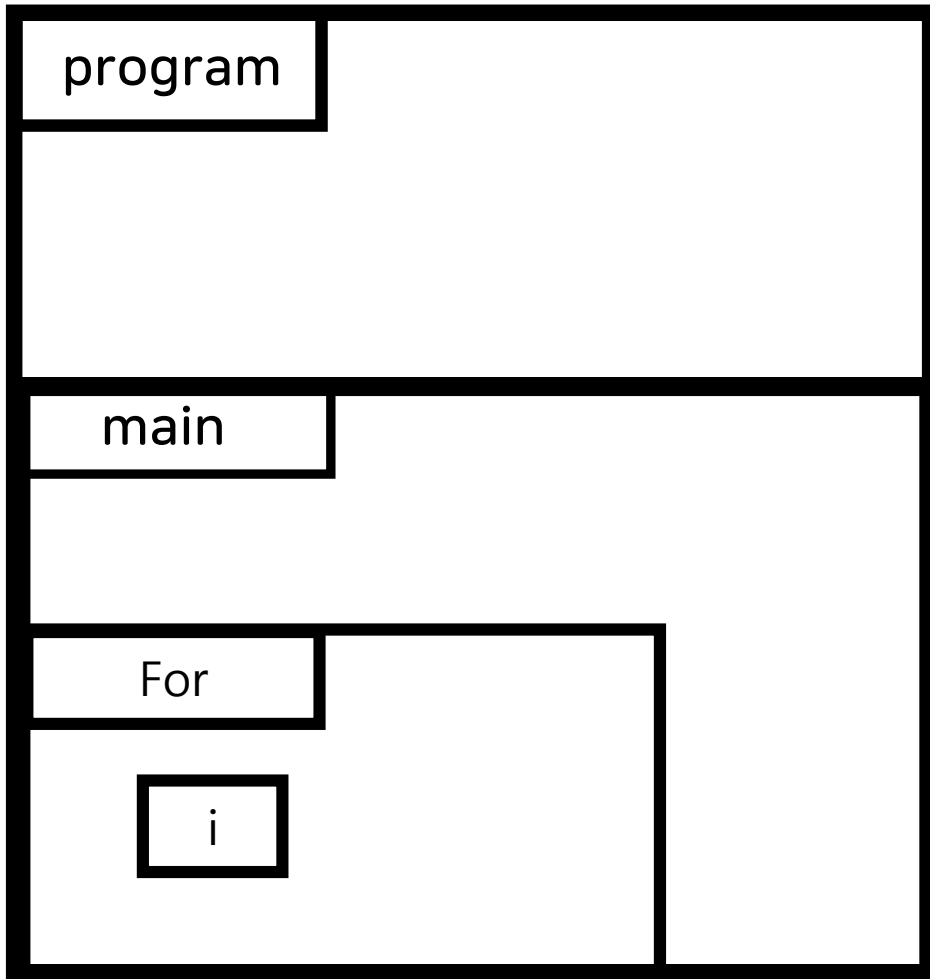
이전 코드에서 한 줄이 추가됐습니다.

이 코드를 실행해보면 어떻게 될까요?

컴파일 에러가 뜨게 됩니다! (왜?)

i 는 for문 바깥에서 사라지기 때문입니다!

For문



C언어 에서 변수는 선언 된 위치 내에서만 존재합니다.
(그림 참조)

전역변수는 main 함수 내에서 사용이 가능합니다.

하지만 main내의 변수는 그 바깥에서 사용이 불가능 하
죠

For문도 이와 같이 범위가 있습니다.

따라서 i가 for문 안에서 선언 되면,
for문 안에서만 존재합니다.

For문

```
1  #include<stdio.h>
2  int main() {
3      int i;
4      for ( i = 1; i < 6; i++) {
5          printf("%d\n", i);
6      }
7      printf("%d\n", i);
8      return 0;
9  }
```

에러가 뜨지 않게 하려면 다음과 같이 작성하면 되겠죠?

변수 `i`를 **for문 바깥**에서 **선언**하여 for문이 종료되어도 `i`는 남아있게 하였습니다.

For문 활용하기(주의 사항)

다음 페이지부터는 for문을 활용하여 여러 문제들을 풀어볼 것 입니다.

문제를 보고 코드를 작성할 때 **아래의 내용을 잘 숙지**하시고 코드를 작성해 주세요!

1. 변수 이름을 잘 써야 합니다(**누가 봐도 알 수 있게**).

Ex) 나무의 개수를 저장할 변수 선언 (**int a - X, int tree - 0**)

2. 머릿속으로만 생각하지 말고 **종이에다가 쓰면서** 합시다.(문제가 어려워 질수록 머리만으로 풀 수 있는 문제가 줄어듭니다. 미리 습관을 길러 둡시다.)

준비 됐으면 다음 페이지로!

For문 활용하기(팩토리얼)

for문을 활용하여 정수 n 을 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.)

For문 활용하기(팩토리얼)

```
1  #include <stdio.h>
2  int main() {
3
4      int n;
5      scanf("%d", &n);
6
7      int fac = 1;
8      for (int i = 1; i <= n; i++) {
9          fac *= i;
10     }
11     printf("%d! = %d\n", n, fac);
12 }
```

```
5
5! = 120
계속하려면 아무 키나 누르세요...
```

n 팩토리얼을 출력하는 프로그램 코드입니다.

여러분들도 이것과 비슷하게 코드를 짜셨나요? 잘하셨습니다.

코드를 작성하지 못하셨던 분들은 [옆의 코드](#)와 [아래 표](#)를 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)

```
1  #include <stdio.h>
2  int main() {
3
4      for (int i = 1; i <= 3; i++) {
5          for (int j = 1; j <= 3; j++) {
6              printf("%d ", j);
7          }
8          printf("\n");
9      }
10     return 0;
11 }
```

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

계속하려면 아무 키나 누르세요...

For문안에는 또 for문을 쓸 수 있습니다.

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

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

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 **배열**을 배울 것입니다.

배열은 반복문이 공부다 되었다는 가정 하에 진행되기 때문에 혹시 **아직 공부가 안되었다**고 생각하는 **복습**을 하고 배열을 공부하는 것을 추천합니다.

챕터 2: 배열

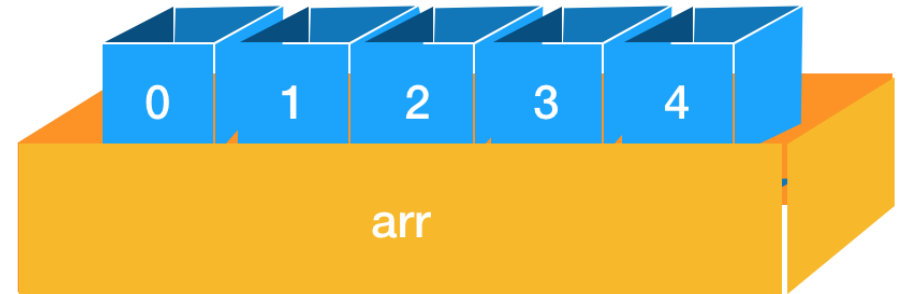
배열이 왜 필요한지 알아보고 직접 배열을 선언하고 정의 해보자!

배열이란?

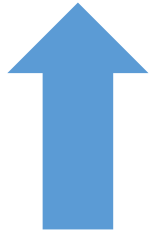
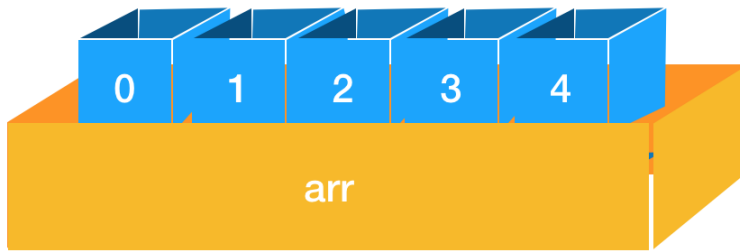
배열은 **같은 자료형**을 가진 **연속된 메모리 공간**으로 이루어진 자료구조 입니다..

엥? 이게 무슨 말이냐구요?

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



배열이란?



같은 자료형으로만 구성!!!

자료형이란 이전에 배웠던 `int`, `double`, 등을 말합니다.

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

여기서 주의할 점!)

c언어에서 배열의 시작번호는 0 입니다. (1아님!)

배열의 크기가 n 이면 번호는 $n-1$ 까지 있겠죠?($0 \sim n-1$)

배열의 필요성(간결성)

학생 100명의 소입설 점수가 주어진다고 합시다. 이를 다 저장해야 한다고 하면 지금까지 배운 내용으로는 **왼쪽 코드**처럼 해야 합니다.

하지만 배열을 사용한다면? **오른쪽 코드**처럼 아주 **간결**해 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      |  
5      | int scores[101];  
6      | for (int i = 0; i < 100; i++) {  
7      |     scanf("%d", &scores[i]);  
8      |     }  
9      |  
10     | return 0;  
11     | }  
12     }
```

배열의 필요성(효율성)

학생들의 점수를 저장한 후 원하는 학생의 점수를 출력한다고 합시다.

배열이 없을 때는 직접 코드를 입력해야 하지만, 배열을 사용하면 **효율적**으로 학생들의 점수를 참조할 수 있습니다.

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

```
1  #include <stdio.h>  
2  int main() {  
3        
4      int scores[101];  
5      int num;  
6      for (int i = 1; i <= 2; i++) {  
7          scanf("%d", &num);  
8          printf("%d", scores[num]);  
9      }  
10   
11     return 0;  
12 }
```

배열의 선언과 정의

```
1  #include<stdio.h>
2  int arr_global[3]; // 배열의 선언(전역 변수)
3
4  int main(void) {
5      int x = 5; // 변수의 선언과 정의
6      int y; // 변수의 선언
7      y = 5; // 변수의 정의
8
9      int arr[3] = { 1,2,3 }; // 배열의 선언과 정의
10     int arr2[3]; // 배열의 선언
11     arr2[0] = 1; // 배열의 정의
12     arr2[1] = 2; // 배열의 정의
13
14
15     return 0;
16 }
17
```

`data_type name[SIZE];` : 배열의 선언 방식

배열의 선언과 정의는 이전 시간에 배웠던 변수의 선언과 정의와 매우 비슷합니다.

배열을 선언과 동시에 초기화 할 수 있고 선언만 한 뒤 나중에 정의할 수 도 있습니다.

10번째 줄을 보면 배열의 선언과 정의가 동시에 된 것을 볼 수 있습니다.

11~13 줄을 보면 선언과 정의가 따로 행해진 것을 볼 수 있습니다.

배열의 선언과 정의

```
1  #include<stdio.h>
2  int arr_global[3]; // 배열의 선언(전역 변수)
3
4  int main(void) {
5      int arr[3] = { 1,2,3 }; // 배열의 선언과 정의
6
7      int arr_local[3];
8      arr_local[0] = 1;
9      arr_local[1] = 2;
10
11     return 0;
12
13 }
```

이전 시간에 **전역 변수와 지역 변수의 차이점**을 배웠습니다.

옆의 코드를 보고 다음 질문에 대해 생각해 보세요

- (배열이 변수와 유사점이 많다는 것을 생각하면 쉽게 알 수 있을거예요)

1. arr_global[3] 에는 무슨 값이 들어가 있을까?
2. arr[3] 에는 무슨 값이 들어가 있을까?
3. arr_local[3] 에는 무슨 값이 들어가 있을까?

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

배열의 선언과 정의(코드 따라해보기)

```
1  #include<stdio.h>
2  int arr_global[3]; // 배열의 선언 (전역 변수)
3
4  int main(void) {
5      int arr[3] = { 1,2,3 }; // 배열의 선언과 정의
6
7      int arr_local[3];
8      arr_local[0] = 1;
9      arr_local[1] = 2;
10
11     printf("전역 배열 : ");
12     for (int i = 0; i < 3; i++) {
13         printf("%d ", arr_global[i]);
14     }
15     printf("\n지역 배열 arr : ");
16     for (int i = 0; i < 3; i++) {
17         printf("%d ", arr[i]);
18     }
19     printf("\n지역 배열 arr_local : ");
20     for (int i = 0; i < 3; i++) {
21         printf("%d ", arr_local[i]);
22     }
23     return 0;
24 }
```

옆의 코드를 따라서 작성하고 실행해보세요!

아마 arr_local 배열에 **이상한 값**이 출력될 것입니다.

지역 배열은 값을 정의하지 않으면 **쓰레기 값**이 들어가기 때문입니다!

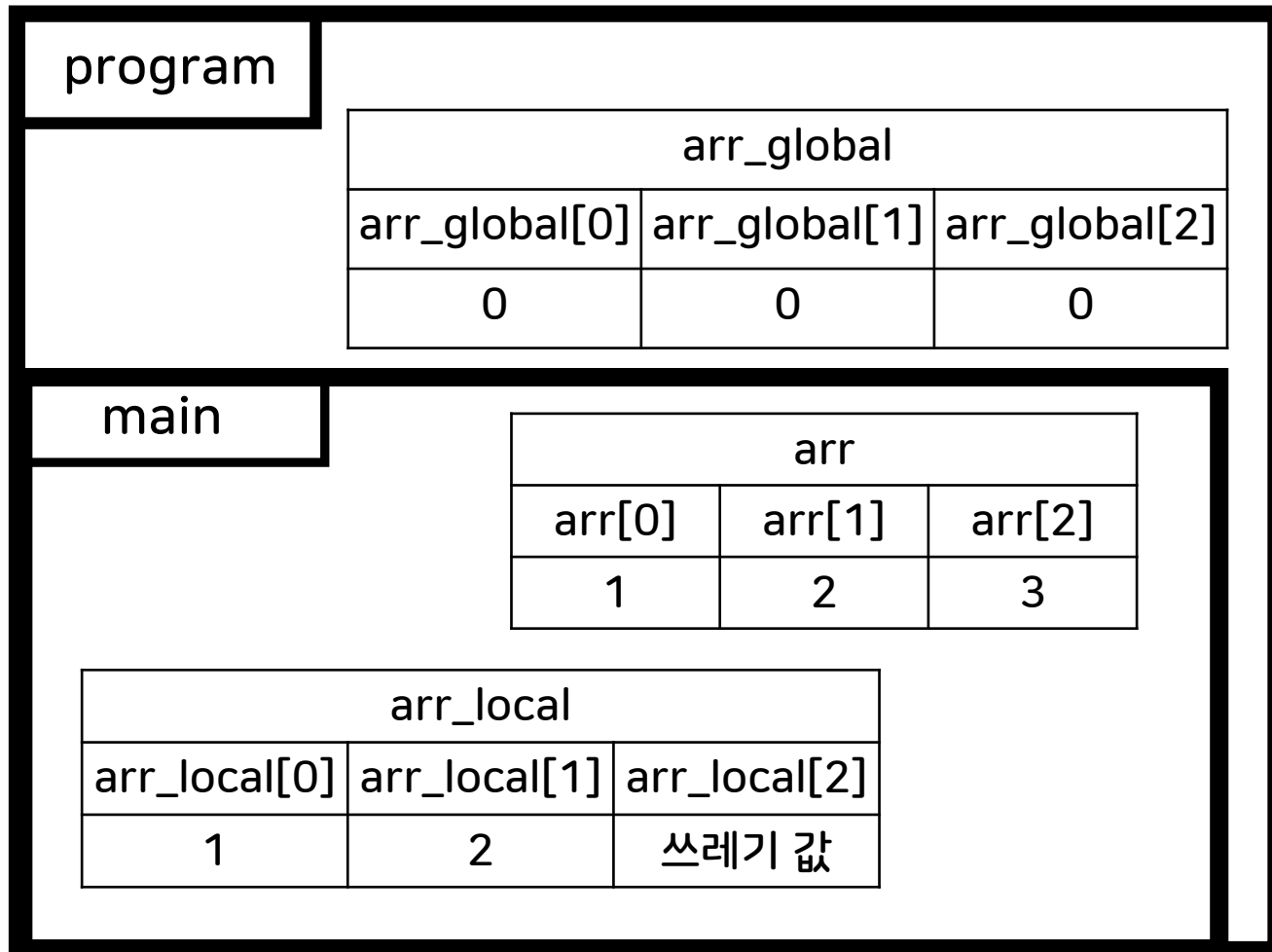
(배열의 출력 방법은 조금 이따가 배우니 배열 안의 값에 집중!)

배열의 선언과 정의(생각했던 것과 같나요?)

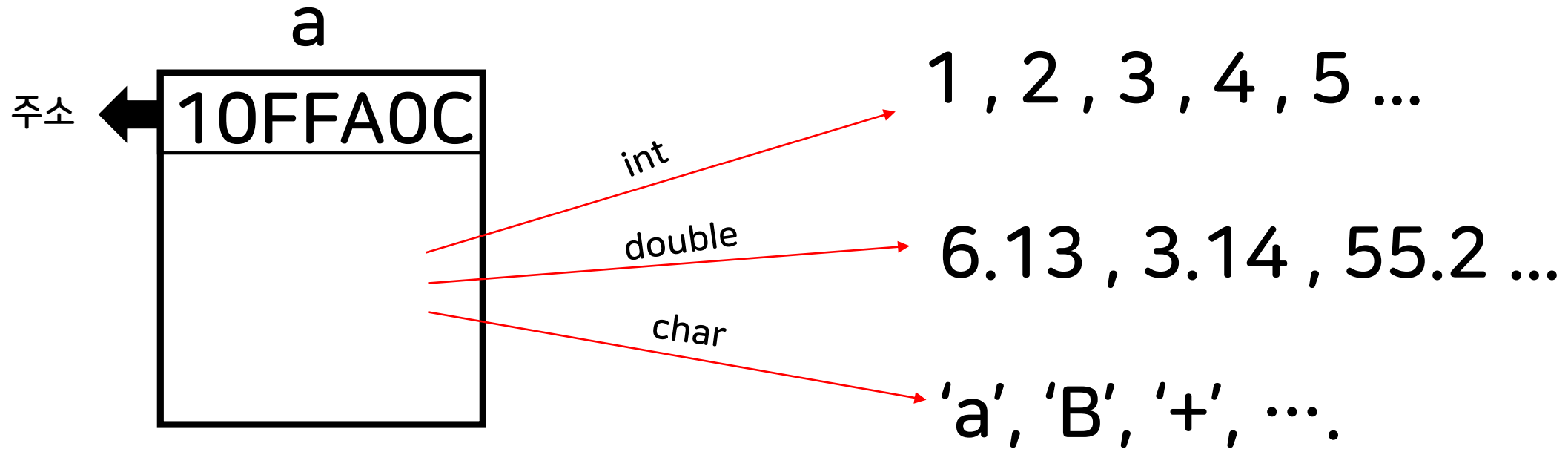
```
1  #include<stdio.h>
2  int arr_global[3]; // 배열의 선언(전역 변수)
3
4  int main(void) {
5
6      int arr[3] = { 1,2,3 }; // 배열의 선언과 정의
7
8      int arr_local[3];
9      arr_local[0] = 1;
10     arr_local[1] = 2;
11
12     return 0;
13 }
```

전역배열은 값을 정의 하지 않으면 0으로 초기화!

지역배열은 값을 정의 하지 않으면 쓰레기 값이 들어갑니다!



배열의 형식 = 변수의 형식



배열은 변수의 형식과 같아요!

배열의 형식(저번 시간에 배웠던 내용 복습)

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

```
1  #include<stdio.h>
2  int main() {
3
4      int arr1[100];
5      char arr2[100];
6      double arr3[10000];
7
8      return 0;
9  }
```

배열의 형식에 관해 공부해 봤으니 여러 종류의 배열을 만들어 봅시다!

옆의 코드를 보고 여러 자료형의 배열을 선언해봅시다!

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 **for문**을 쓰면 간편하게 입력 받을 수 있습니다.

배열 입출력(I/O)

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

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

참 쉽죠?

문자열

문자열이란 **문자의 배열** 입니다.

즉 char형 변수들이 나열되어 있는 구조이죠.

문자열의 선언과 정의

다음과 같이 문자열을 선언하고 정의할 수 있습니다.

```
1  #include<stdio.h>
2  int main() {
3      char str[5] = { 'H','i' };
4
5      char str1[5] = { 'H','e','l','l','o' };
6
7      return 0;
8  }
```


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 정의

또 문자열을 다르게 정의할 수 있습니다.
하지만 여기서는 끝에 '\0'(NULL) 값이 들어가게 됩니다.

```
1  #include<stdio.h>
2  int main() {
3
4      char str1[5] = { 'H','e','l','l','o' };
5      char str2[6] = "Hello";
6      char str3[5] = "Hello";
7
8      return 0;
9  }
```



'\0'을 넣을 공간 없어서 에러!

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

Str2[]	[0]	[1]	[2]	[3]	[4]	[5]
Char	'H'	'e'	'l'	'l'	'l'	'\0'

Str3[]	[0]	[1]	[2]	[3]	[4]	[5]
Char	'H'	'e'	'l'	'l'	'o'	'\0'

문자열의 선언과 정의(NULL = '\0')

NULL 은 값이 0인 제어 문자입니다.

문자열에서는 문자열의 끝을 알리기 위해 사용됩니다. ('\0')

0	1	2	3	4	5
'A'	'L'	'O'	'H'	'A'	'\0'

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

문자열의 입출력(I/O)

문자열은 여러 입출력 방식이 있습니다.

- 문자 단위로 입출력 할 수 있고
- 문자열을 한번에 입출력 할 수 도 있습니다.

문자열의 입출력(I/O)

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

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

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

문자열의 입출력(I/O)

```
scanf("%s", str);  
                                &str 아님!  
printf("%s", str);
```

"%s" 를 사용하여 문자열을 입력 받을 수 있습니다.

%s는 공백문자(' ' '\n' '\0' 등등) 까지 입력 받습니다.
(공백문자는 저장 안함.)

마지막 문자 뒤에는 '\0'을 저장해 줍니다.

출력은 NULL까지 합니다.

배열

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!

배열 문제 풀기



BOJ 5597

과제 안 내신 분..?

배열 문제 풀기(과제 안 내신 분) -답안

```
1  #include<stdio.h>
2
3  int students[31]; // 배열은 30보다 크게 잡아주어야 합니다.
4
5  int main() {
6      int i;
7      int bunho; // 학생들의 번호
8      for (i = 0; i < 28; i++) {
9          scanf("%d", &bunho);
10         students[bunho] = 1; // 출석이면 1을 저장
11     }
12     for (i = 1; i <= 30; i++) { // 1부터 30까지 보고 student[i]가 0이면 결석
13         if (students[i] == 0) {
14             printf("%d\n", i); // 결석자 번호 출력
15         }
16     }
17     return 0;
18 }
```

배열 문제 풀기



BOJ 3052

나머지

배열 문제 풀기(나머지) - 답안

```
1  #include<stdio.h>
2
3  int nanu[43]; // A를 B로 나눈 나머지를 저장합니다.
4
5  int main() {
6      int N;
7      for (int i = 1; i <= 10; i++) { // 10번 입력을 받습니다.
8          scanf("%d", &N);
9          nanu[N % 42] = 1; // 나머지의 값의 존재 여부를 배열에 저장합니다.(존재하면 1)
10     }
11     int stack = 0; // 서로 다른 나머지가 몇 개 있는지 저장합니다.
12     for (int i = 0; i <= 42; i++) {
13         if (nanu[i] == 1) {
14             stack++;
15         }
16     }
17     printf("%d", stack); // 서로 다른 값을 출력합니다.
18     return 0;
19 }
```

끝

수고하셨습니다~~