



# ALOHA

## #1주차 (2)

변수, 사칙 연산, 연산자,  
조건문, if문

# #CH.1

변수와 연산자



# 변수란?

---

## 변수6 變數

1. 어떤 상황의 가변적 요인.
2. [수학] 어떤 관계나 범위 안에서 여러 가지 값으로 변할 수 있는 수.

# 변수란?

컴퓨터인터넷IT용어대사전

## 변수

[ variable ↕ ]

- (1) 일반적으로는 미리 정해진 범위(range) 내에서 값(value)이 변할 수 있는 수를 대표하는 문자이다.
  - (2) 프로그래밍 언어인 COBOL에서는 프로그램 실행중에 값이 바뀌는 경우가 있는 데이터 항목이다. 또 FORTRAN에서는 배열도 배열 요소도 아닌 데이터 항목이며, 영자명(symbolic name)으로 식별된다. 인용하거나 정의하는 것이 가능하다.
  - (3) 형용사로서, 목적이나 용도(application) 등에 따라서 변경이 가능한 것, 또는 변화하는 것 등을 표시하는 의미로 사용된다. 예를 들면 가변 길이 레코드(variable length record) 등.
  - (4) 어떤 주어진 적용에 있어서 실제의 값이 할당되기까지, 값이 정해지지 않은 또는 기지(既知)의 범위에 값이 정해지지 않은 것.
  - (5) (프로그래밍에 있어서) 문자 또는 문자의 집합이며, 값을 창조하여 컴퓨터 프로그램의 실행중에는 어드레스에 대응하고 있는 것.
  - (6) 다른 값을 취할 수 있으나 그런 시점에서는 한 가지 값만을 취하는 언어 대상물.
- [주] 한 기지의 변수를 취할 수 있는 값은, 보통 어떤 정해진 데이터형으로 한정된다.

```
1  #include <stdio>
2
3  int main(void)
4  {
5      int x;
6      x = 1;
7
8      int y = 2;
9
10     return 0;
11 }
```

보통은 이 방법을 많이 써요

## 변수(variable)의 선언과 정의

main 함수 내에 int형 변수 x를 **선언과 정의**를 해주고, 다음 줄에서 x의 값을 대입해 주었어요.

y처럼 한번에 **대입까지 동시에** 해줄 수도 있어요. 이렇게 선언과 정의, 대입까지 동시에 하는 것을 **초기화**라고 합니다.

# 지역변수와 전역변수

---

## 전역변수(Global Variable)

- 함수 밖에서 선언되는 변수
- 프로그램 시작부터 종료까지 존재
- 어디서나 접근 가능
- 초기화 하지 않을 경우 '0' 으로 초기화

## 지역변수(Local Variable)

- 중괄호 내에서 선언되는 변수
- 선언된 지역에서 벗어나면 소멸
- 선언된 지역 내에서만 접근 가능
- 초기화 하지 않을 경우 "쓰레기 값" 으로 초기화

```
1  #include <stdio>
2
3  int main(void)
4  {
5      int a;
6      printf("%d", a);
7      return 0;
8  }
```



Debug Error!

Program: C:\Users\W...  
Module: C:\Users\W...  
File:

Run-Time Check Failure #3 - The variable 'a' is being used  
without being initialized.

(Press Retry to debug the application)

## 참고) 쓰레기 값

지역변수 `a`를 선언만 해주고 정의해주지 않았더니 에러가 떠요.

자바처럼 초기화 하지 않았을 때 0으로 지정해주는 언어도 있지만, C 언어에서는 지역변수의 값이 초기화되지 않았을 때 그 변수에 값을 대입하지 않아요.

말 그대로 아무 의미 없는 “**쓰레기 값**” 이 들어가게 돼요.

```

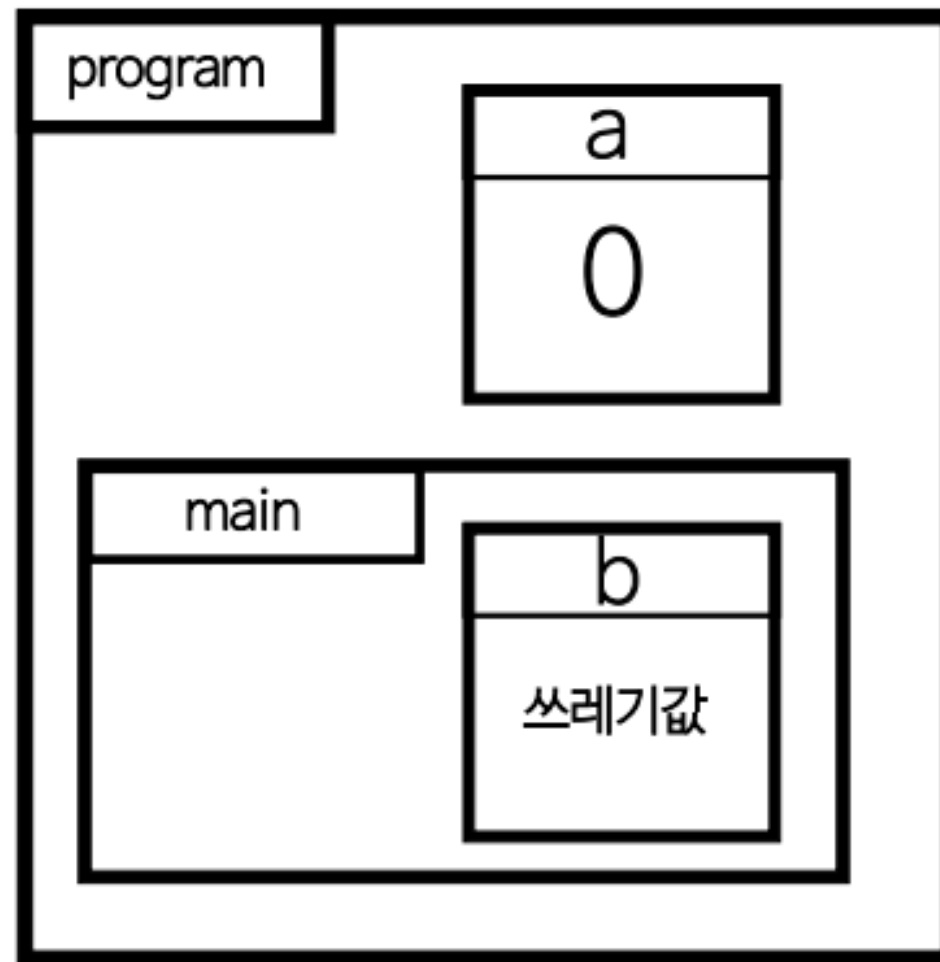
1  #include <stdio>
2
3  int a;
4
5  int main(void)
6  {
7      int b;
8
9      return 0;
10 }
```

전역 변수

지역 변수

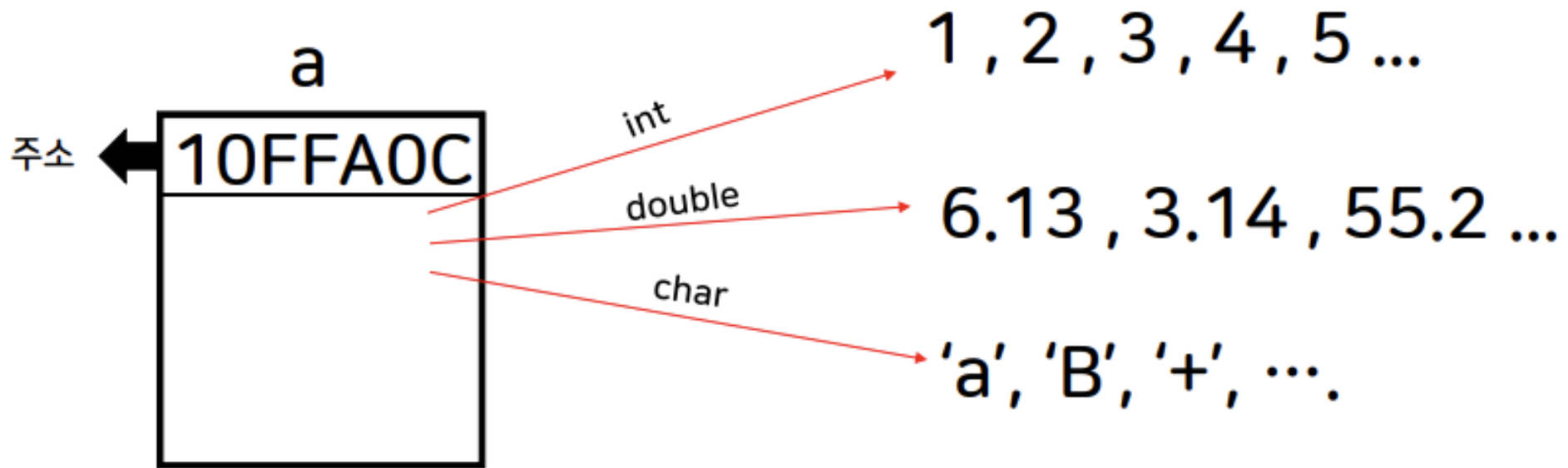
전역 변수는 정의하지 않았을 때, 메모리에 0이 들어가네요.

지역 변수는 쓰레기 값이 들어갔어요.





# 변수의 형식



메모리 자체에는 어떤 값도 들어갈 수 있어요!  
=> **변수의 형식**이 중요한 이유

# 변수의 형식

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 형식의 선언

```
1  #include <stdio>
2
3  int main(void)
4  {
5      int i = 6; // 정수형 변수(int)
6      char c = 'b'; // 문자 변수(char)
7      double d = 3.14; // 실수형 변수(double)
8
9      return 0;
10 }
```

# 산술 연산자

연산자	의미
=	대입
+	덧셈
-	뺄셈
*	곱셈
/	몫
%	나머지

→ 뒤에 나오는 비교연산자(==)와 헷갈리면 안되요.

연산자를 사용할 때는 같은 자료형끼리 사용해요.

같은 연산자라도 자료형에 따라 결과가 달라질 수 있어요.

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

## 산술 연산자

5  
계속하려면 아무 키나 누르십시오...

```

1  #include <stdio>
2
3  int main(void)
4  {
5      int a = 5;
6      int b = 2;
7      double c = 5;
8      double d = 7;
9
10     printf("%lf\n", (double)(a / b));
11     printf("%lf\n", c / d);
12     printf("%d", a % b);
13
14     return 0;
15 }

```

형 변환 (다음 슬라이드에 나와요)

## 산술 연산자

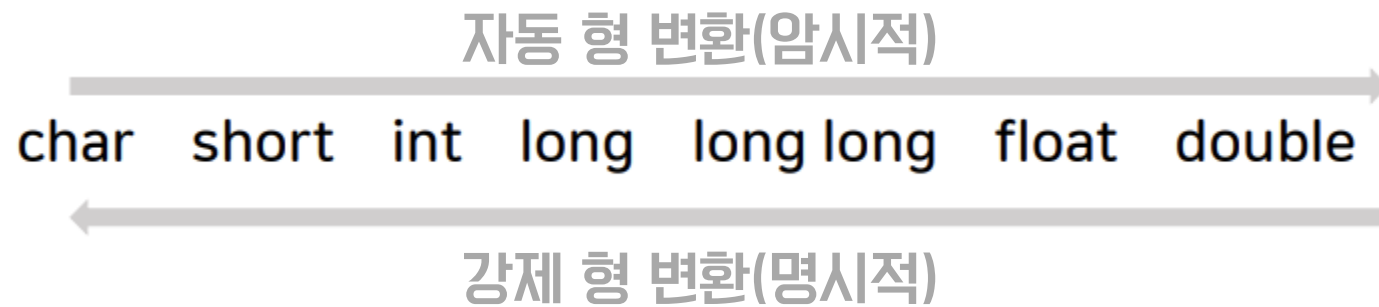
```

2.000000
0.714286
1
계속하려면 아무 키나 누르십시오...

```

a 와 b는 **정수형 변수**이기 때문에 a/b의 결과인 2.5에서 소수점 이하가 먼저 버려져요.

## cf ) 형 변환(타입 캐스팅)

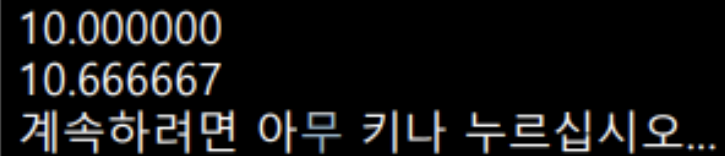


자동 형 변환 : 자료형의 크기가 큰 쪽으로 자동 변환 되는 것

강제 형 변환 : 프로그래머가 강제로 자료형을 변환하는 것

```
1  #include <stdio>
2
3  int main(void)
4  {
5      int a = 32;
6      int b = 3;
7      float c;
8
9      c = a/b; // 경고가 발생함
10     printf("%f\n", c);
11
12     c = (float)a / b;
13     printf("%f\n", c);
14
15     return 0;
16 }
```


## cf ) 형 변환(타입 캐스팅)



```
10.000000
10.666667
계속하려면 아무 키나 누르십시오...
```



## cf ) 형 변환(타입 캐스팅)

	코드	설명
	C4244	'=': 'int'에서 'float'(으)로 변환하면서 데이터가 손실될 수 있습니다.

앞의 코드 9번째 줄에서 경고가 발생하게 됩니다.  
int/int 의 결과는 int 인데 float에 대입을 했기 때문이에요.

12번째 줄에서 처럼 (float)a 타입 캐스팅을 해주면 해결되겠죠?

# 대입 연산자

연산자	의미
<code>+=</code>	<code>a=a+b</code>
<code>-=</code>	<code>a=a-b</code>
<code>*=</code>	<code>a=a*b</code>
<code>/=</code>	<code>a=a/b</code>
<code>%=</code>	<code>a=a%b</code>

`a = a + b;`  
이렇게 쓰는 것 보다는  
`a += b;`  
이렇게 쓰는게 코딩하기 더 편하겠죠?

```
1  #include <stdio>
2
3  int main(void)
4  {
5      int a = 5;
6      int b = 8;
7      a -= 2;
8      b %= 3;
9      printf("%d %d\n", a, b);
10
11     return 0;
12 }
```

## 대입 연산자

```
3 2
계속하려면 아무 키나 누르십시오...
```

# 증감 연산자

연산자	의미
a++	연산 후 증가 (후위 연산자)
a--	연산 후 감소
++a	증가 후 연산 (전위 연산자)
--a	감소 후 연산

# 증감 연산자

n1 = a ++

2 증가연산

← 1 할당

후위 연산자(postfix)

n2 = ++ b

1 증가연산

← 2 할당

전위 연산자(prefix)

다음 예제를 통해 이해해봐요

```

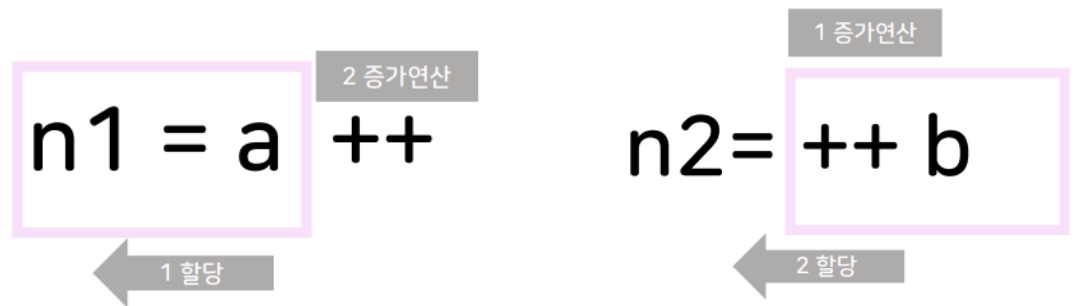
1  #include <stdio>
2
3  int main(void)
4  {
5      int a = 5;
6
7      printf("%d %d\n", a++, a);
8      printf("%d %d\n", --a, a);
9
10     return 0;
11 }

```

## 증감 연산자

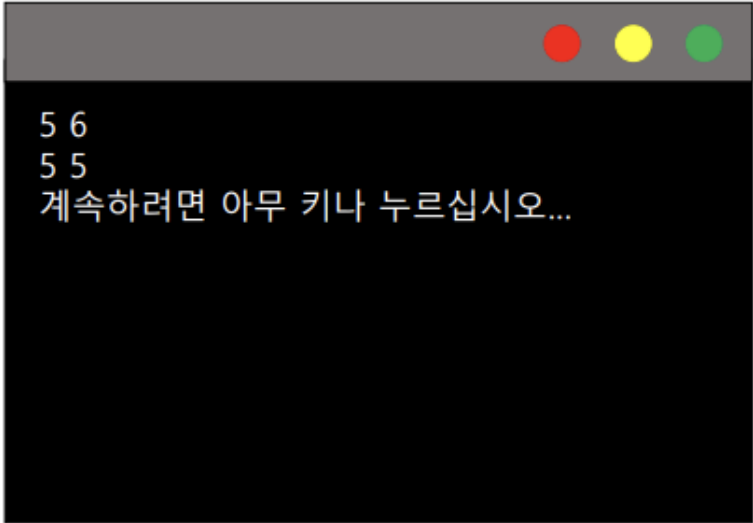
결과가 어떻게 나올까요??

실행해보기 전에 먼저 생각해 봅시다.



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

## 증감 연산자



```
5 6
5 5
계속하려면 아무 키나 누르십시오...
```

**연산자의 위치**를 보면 결과를 알 수 있어요.

교수님에 따라 다르지만 증감 연산자 부분은 소입설 시험에서 잘 나오는 부분이니 스스로 다른 예제들을 만들어서 확인해봐요.

# 비교 연산자

비교 연산자는 true/false 를 리턴해요.

앞에서 나온 대입연산자(=)와 헷갈리면 안되요.  
(코딩할 때 자주 하는 실수!)

연산자	의미
a==b	a와 b가 같은가
a!=b	a와 b가 다른가
a>b	a가 b보다 큰가
a<b	a가 b보다 작은가
a>=b	a가 b 이상인가
a<=b	a가 b 이하인가



bool 자료형이 정의된 헤더파일을 선언해 주세요

```
1  #include<stdio.h>
2  #include<stdbool.h>
3
4  int main(void) {
5
6      int a = 5, b = 5, c = 2;
7      bool b1 = (a == b);
8      bool b2 = (a == c);
9      bool b3 = (a != b);
10
11      printf("%d %d %d\n", b1, b2, b3);
12  }
```

## 비교 연산자

```
1 0 0
계속하려면 아무 키나 누르십시오...
```

C 언어에서는

0을 제외한 모든 값 == true  
0 == false  
입니다.

# 논리 연산자

연산	의미
a&&b (AND)	a,b 둘다 참인가
a  b (OR)	a,b 둘 중 하나라도 참인가
!a (NOT)	a가 true 면 false , false 면 true

# 논리 연산자

논리곱(&&,AND)		
A	B	결과값
0	0	0
0	1	0
1	0	0
1	1	1

논리합(  ,OR)		
A	B	결과값
0	0	0
0	1	1
1	0	1
1	1	1

```
1 #include<stdio.h>
2 #include<stdbool.h>
3
4 int main(void) {
5
6     bool a = true;
7     bool b = true;
8     bool c = false;
9     printf("%d\n", a&&b);
10    printf("%d\n", a || c);
11    printf("%d\n", !a);
12    return 0;
13 }
```

## 논리 연산자

```
1
1
0
계속하려면 아무 키나 누르십시오...
```

C 언어에서는

0을 제외한 모든 값 == true  
0 == false  
입니다.

# 연산자 우선순위

연산자	내용
( ), [ ]	괄호
!, ++, --	부정/증감 연산자
* / %	곱셈/나눗셈 연산자
+ -	덧셈/뺄셈 연산자
< <= > >=	관계 연산자
== !=	
&&	논리곱 연산자
	논리합 연산자
?:	조건 연산자
= += -= *= /= %=	대입/할당 연산자

풀어볼까유



#1000  
A+B

덧셈

풀어볼까유



#1001  
A-B

뽕셈

풀어볼까유



#10998  
 $A \times B$

곱셈



풀어볼까유



#1008  
A/B

이번엔 나눠봅시다

풀어볼까유



#10869  
사칙연산

총 정리



잠시 휴식시간

# #CH.2

조건문



# 조건문

---

조건문에서는 프로그래머가 명시한 Boolean 자료형 조건이 '참' 인지 '거짓' 인지에 따라 계산이나 상황을 수행해요.

```
#include <stdio.h>

int main()
{
    if(조건1){
        (실행 내용1);
    }
    else if(조건2){
        (실행 내용2);
    }
    else{
        (실행 내용3);
    }
    return 0;
}
```

## if 문

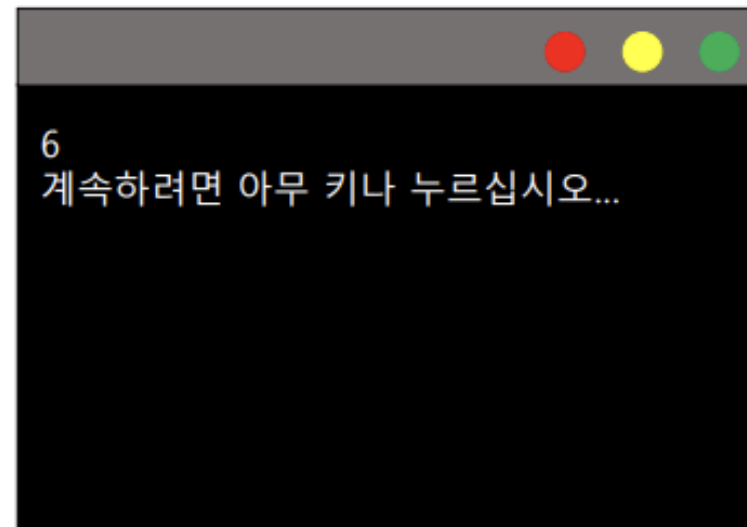


```

1  #include<stdio.h>
2  #include<stdbool.h>
3
4  int main(void) {
5
6      int a = 2;
7      if (a == 1) {
8          printf("%d\n", a * 2);
9      }
10     else if (a == 2) {
11         printf("%d\n", a * 3);
12     }
13     else {
14         printf("%d\n", a);
15     }
16     return 0;
17 }

```

## if 문



```

6
계속하려면 아무 키나 누르십시오...

```

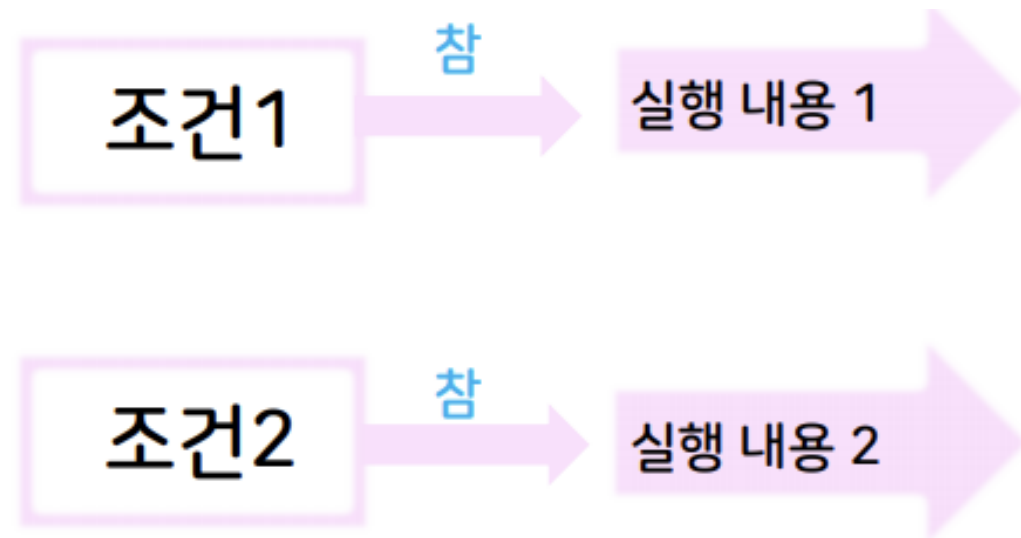
a의 값이 2이니까 11번째 줄이 실행되고  
a\*3 = 6이 출력됐네요.

```
#include <stdio.h>

int main()
{
    if(조건1){
        (실행 내용1);
    }
    if(조건2){
        (실행 내용2);
    }

    return 0;
}
```

## if 문





```
1 #include<stdio.h>
2 #include<stdbool.h>
3
4 int main(void) {
5     |
6     |
7     int a = 3;
8     if (a * 3 == 9) {
9         |
10        printf("%d\n", a * 3);
11    }
12    if (a < 5) {
13        |
14        printf("%d\n", a);
15    }
16    return 0;
17 }
```

## if 문

```
9
3
계속하려면 아무 키나 누르십시오...
```

else 없이 if 만 연결되어 있다면 두 if 문 전부 실행 될 수 있어요.

풀어볼까유



#2753  
윤년

조건문을 써봐요

풀어볼까유



#1330

두 수 비교하기

비교 연산자  
기억하시나요?

# switch 문

Switch문은 if문과 비슷하면서도 조금 달라요!

if문과 **다른 점은 <, >와 같은 부등식을 쓰지 못한다**는 겁니다.  
상황에 따라 if문과 switch문 중에 골라서 코딩할 수 있겠죠?

문법은 다음 페이지를 참고해주세요

```
#include <stdio.h>

int main()
{
    switch (변수){
        case (변수가 만족시키는 조건 1):
            (실행 내용 1);
            break;
        case (변수가 만족시키는 조건 2):
            (실행 내용 2);
            break;
        default:
            (실행 내용 3);
            break;
    }
    return 0;
}
```

## switch 문



```
#include <stdio.h>

int main()
{
    switch (변수){
        case (변수가 만족시키는 조건 1):
            (실행 내용 1);
            break;
        case (변수가 만족시키는 조건 2):
            (실행 내용 2);
            break;
        default:
            (실행 내용 3);
            break;
    }
    return 0;
}
```

## switch 문

Switch 문은 괄호안의 변수의 값과 동일한 값을 가지는 case로 가서 실행 내용을 실행해요.

만약 변수의 값과 동일한 값을 가지는 case가 없다면...?

```
#include <stdio.h>

int main()
{
    switch (변수){
        case (변수가 만족시키는 조건 1):
            (실행 내용 1);
            break;
        case (변수가 만족시키는 조건 2):
            (실행 내용 2);
            break;
        default:
            (실행 내용 3);
            break;
    }
    return 0;
}
```

default 로 가게 돼요

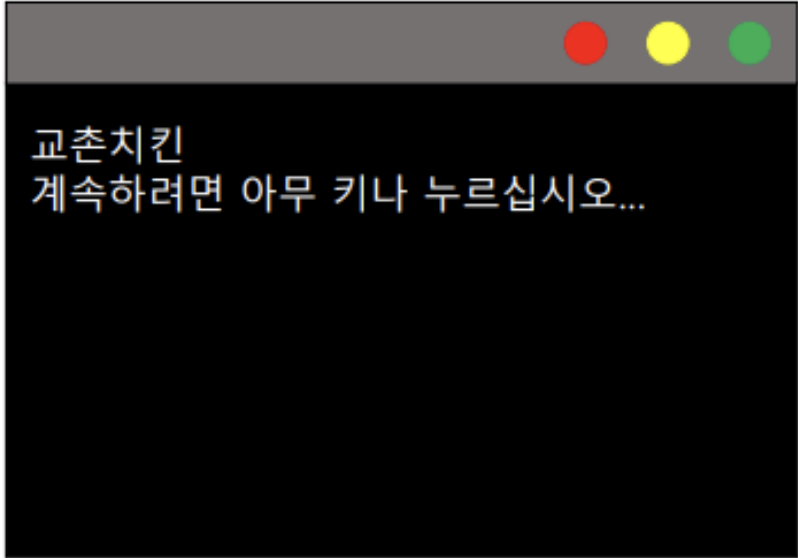
## switch 문

Switch 문은 괄호안의 변수의 값과 동일한 값을 가지는 case로 가서 실행 내용을 실행해요.

만약 변수의 값과 동일한 값을 가지는 case가 없다면...?

```
1  #include<stdio.h>
2
3  int main() {
4      int a = 1;
5      switch (a) {
6          case 0:
7              printf("네네치킨");
8              break;
9          case 1:
10             printf("교촌치킨");
11             break;
12          case 2:
13             printf("푸라닭 치킨");
14             break;
15          default:
16             printf("먹지마");
17             break;
18      }
19  }
```

## switch 문



교촌치킨  
계속하려면 아무 키나 누르십시오...

a = 1 이기때문에 1번 case로 가서 코드를 실행하겠네요



# if 문의 중첩

```
1  #include<stdio.h>
2
3  int main(void) {
4
5      int a, b, c;
6      scanf("%d %d %d", &a, &b, &c);
7      if ((a == 2) && (b % 3 == 0) && (c != 5)) {
8          printf("%d %d %d", a, b, c);
9      }
10     else {
11         printf("0");
12     }
13     return 0;
14 }
```

그림과 같이 if 조건문이 길면 코드를 짜는 것이 불편하지 않을까요?

다르게 짜는 방법은 없을까요?

```

1  #include<stdio.h>
2
3  int main(void) {
4
5      int a, b, c;
6      scanf("%d %d %d", &a, &b, &c);
7      if (a == 2) {
8          if (b % 3 == 0) {
9              if (c != 5) {
10                 printf("%d %d %d", a, b, c);
11             }
12             else {
13                 printf("0");
14             }
15         }
16         else {
17             printf("0");
18         }
19     }
20     else {
21         printf("0");
22     }
23     return 0;
24 }

```

## if 문의 중첩

If문을 여러 개 중첩해서 왼쪽과 같이 쓸 수 있어요.

앞 페이지의 코드와 같은 기능을 해요.

하지만, if 조건문이 무조건 짧다고 좋은 것도, 여러 개 중첩된다고 해서 좋은 것도 아니에요.

상황에 따라 유동적으로 코딩합시다.



다음 시간에 만나요~