증감 연산자

종류		기호	특징
전위 연산자	++	++i	더하고 실행한다.
	--	--i	빼고 실행한다.
후위 연산자	++	i++	실행하고 더한다.
	--	i--	실행하고 뺀다.

```
#include<stdio.h>
int main()
{
    int i = 7;
    printf("%d\n", i++);
    printf("%d\n", i);
    printf("%d\n", ++i);
    return 0;
}
```

```
output:
7
8
9
```

for

```
for(①초기값; ②조건; ③증감값){
    ④코드
}

for문의 동작 순서는
① → ② → ④ → ③ → ② → ④ → ③ → ② → ④ → ③
으로 조건에서 거짓이 뜰 때까지 계속 반복한다.
```

이해가 어렵다면 아래처럼 이해해도 좋다.

```
초기값
if(조건){
    코드
    증감값
    if문 반복
}
else{
    for문 종료
}
```

```c
#include<stdio.h>
int main()
{
    int i;
    for (i = 0;i < 10;i++) {
        printf("%d ", i);
    }
    printf("\n");
    printf("for문이 종료된 후 i의 값은 %d이다.", i);
    return 0;
}
```

output:

```
0 1 2 3 4 5 6 7 8 9
for문이 종료된 후 i의 값은 10이다.
```

변수를 따로 선언하는 것이 아니라, 초기값을 설정할 때 선언할 수도 있다.

```
#include<stdio.h>
int main()
{
    for (int i = 0;i < 10;i++) {
        printf("%d ", i);
    }
    return 0;
}
```

output:

```
0 1 2 3 4 5 6 7 8 9
```

다만 위와 같은 방식은 for문이 끝나면 초기값에서 선언한 변수에 다시 접근할 수 없다.

```
#include<stdio.h>
int main()
{
    printf("1~10까지 홀수 출력\n");
    for (int i = 1;i <= 10;i += 2) {
        printf("%d ", i);
    }
    return 0;
}
```

output:

```
1~10까지 홀수 출력
1 3 5 7 9
```

while

```
while(조건){
    코드
}

조건이 참인동안 코드를 계속 반복한다.
```

```c
#include<stdio.h>
int main()
{
    int n = 5873;
    int cnt = 0;

    printf("%d은 ", n);

    while (n != 0) {
        n /= 10;
        cnt++;
    }

    printf("%d자리의 수입니다.", cnt);
    return 0;
}
```

output:

```
5873은 4자리의 수입니다.
```

for문과 달리 초기값과 증감값을 따로 설정해줄 수 없기 때문에 직접 설정해줘야한다.

```c
#include<stdio.h>
int main()
{
    int n = 5873;
    int cnt = 1;
    while (n != 0) {
        printf("%d의 자리수는 %d\n", cnt, n % 10);
        n /= 10;
        cnt *= 10;
    }
    return 0;
}
```

output:

```
1의 자리수는 3
10의 자리수는 7
100의 자리수는 8
1000의 자리수는 5
```

do-while

```
do{
    코드
}while(조건);

조건이 참인동안 반복한다.
```

```c
#include<stdio.h>
int main()
{
    int n = 10;
    do {
        printf("%d\n", n *= 10);
    } while (n != 10 && n < 1000);

    n = 10;
    while (n != 10 && n < 1000) {
        printf("%d\n", n *= 10);
    }
    return 0;
}
```

output:

```
100
1000
```

do-while문은 처음에 조건이 거짓이더라도 코드를 실행하지만,
while문은 처음에 조건이 거짓이면 바로 종료하는 것을 알 수 있다.

for문과 while문은 조건 확인 → 코드 실행
do-while문은 코드 실행 → 조건 확인
을 거친다.

break와 continue

```c
#include<stdio.h>
int main()
{
	printf("break 사용\n");
	for (int i = 0;i < 10;i++) {
		if (i == 5) break;
		printf("%d ", i);
	}

	printf("\n\n");
	printf("continue 사용\n");
	for (int i = 0;i < 10;i++) {
		if (i == 5) continue;
		printf("%d ", i);
	}
	return 0;
}
```

output:

```
break 사용
0 1 2 3 4

continue 사용
0 1 2 3 4 6 7 8 9
```

break : 만나는 즉시 가장 가까운 반복문을 빠져나온다.
continue : 만나는 즉시 해당 코드를 끝내고 증감 부분으로 넘어간다.