

- ▶ 반복문 – for
- ▶ 반복문 – while
- ▶ 반복문 – do while
- ▶ 분기문



- 특정 문장을 수행하거나 수행하지 않도록 선택하거나, 특정 문장을 여러 번 반복 수행하게 만드는 것

- 제어문의 종류

- ▶ 조건문 : if, switch
- ▶ 반복문 : for, while, do while
- ▶ 분기문 : break, continue, goto, return

■ 파일명: 학번\_이름\_7주차/max.c

■ 키보드에서 세 개의 정수를 입력 받아서 가장 큰 수와 가장 작은 수의 합을 구하시오.

## ▶ 변수선언

- 입력변수 : x, y, z
- 최대값변수 : maxVal, 최소값 변수: minVal

## ▶ 알고리즘

- 최대값 변수 maxVal와 최소값 변수 minVal에 x로 초기화한다.
- maxVal과 y, z를 비교하여 최대값을 구한다.
- minVal과 y, z를 비교하여 최소값을 구한다.

# 반복문이란?

## I 같은 코드를 여러 번 반복할 수 있도록 하는 제어문

- C언어에는 for, while, do while 이 있다

## I 반복문이 없다면 아래와 같은 끔찍한 일을 해야만 한다

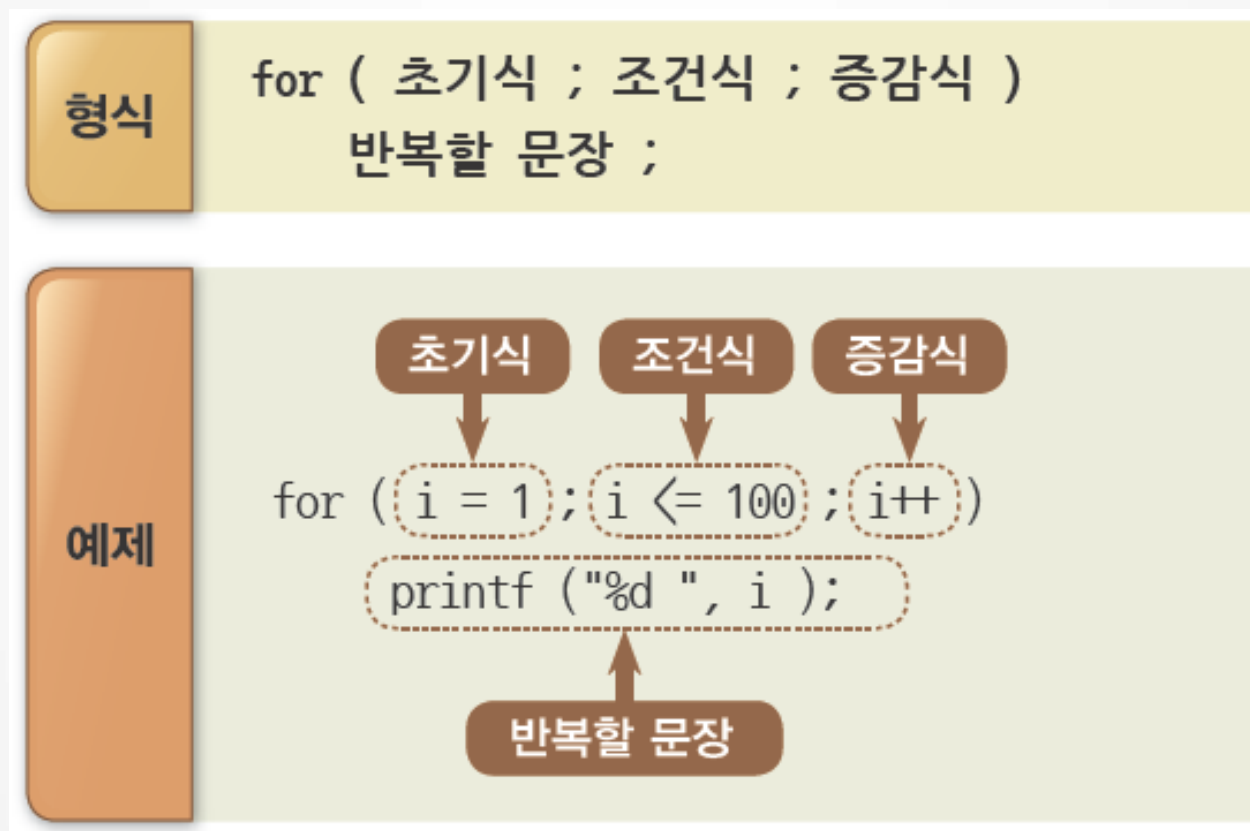
```
int data, sum = 0;
scanf("%d", &data);
sum += data;
scanf("%d", &data);
sum += data;
scanf("%d", &data);
sum += data;
scanf("%d", &data);
sum += data;
scanf("%d", &data);
sum += data;
printf("합계 : %d\n", sum);
```

같은 코드를 복사해서 작성한다.

# 반복문 - for

for은 일정 횟수를 반복하고 싶을 때 쓴다

- 10회 덧셈 등



# 반복문 - for

## 초기식

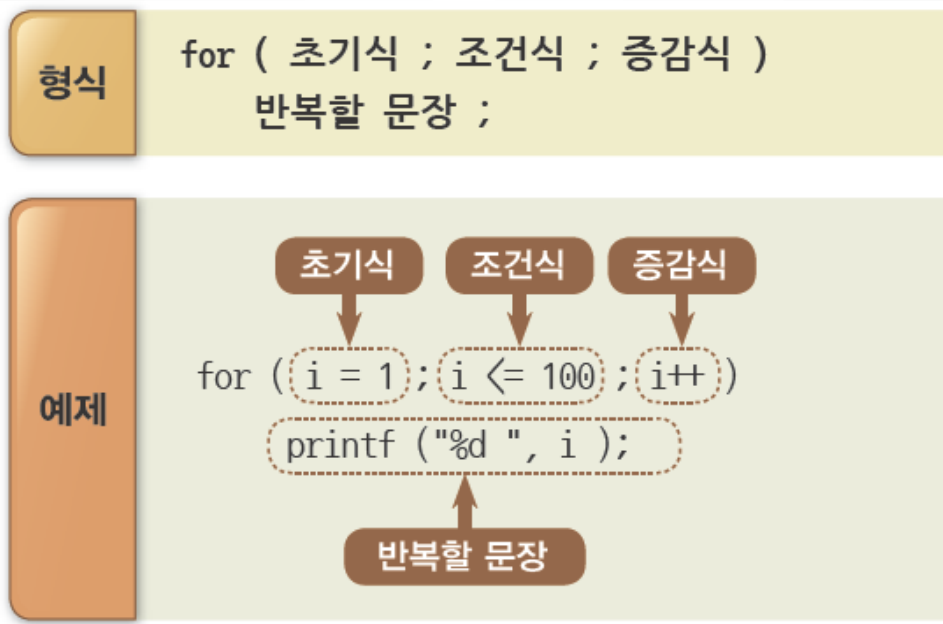
- 최초 한 번만 실행될 코드

## 조건식

- 이 조건을 만족하는 동안만 반복됨

## 증감식

- 한 사이클이 끝날 때마다 실행할 코드



# 반복문 - for의 사용 예

```
01: /* Ex05_08.c */
02: #include <stdio.h>
03:
04: int main(void)
05: {
06:     int i;
07:
08:     for ( i = 1 ; i <= 10 ; i++)
09:         printf("%d ", i);
10:
11:     printf("\n");
12:
13:     return 0;
14: }
```

실행 결과

1 2 3 4 5 6 7 8 9 10

# 중첩된 for의 사용

```
01: /* Ex05_10.c */
02: #include <stdio.h>
03:
04: int main(void)
05: {
06:     int i, j;
07:
08:     for ( i = 1 ; i < 10 ; i++ )
09:     {
10:         for ( j = 1 ; j < 10 ; j++ )
11:         {
12:             printf("%d*%d=%2d ", i, j, i*j);
13:         }
14:         printf("\n");
15:     }
16:
17:     return 0;
18: }
```

중첩된 for의 사용

## 실행 결과

```
1*1= 1 1*2= 2 1*3= 3 1*4= 4 1*5= 5 1*6= 6 1*7= 7 1*8= 8 1*9= 9
2*1= 2 2*2= 4 2*3= 6 2*4= 8 2*5=10 2*6=12 2*7=14 2*8=16 2*9=18
3*1= 3 3*2= 6 3*3= 9 3*4=12 3*5=15 3*6=18 3*7=21 3*8=24 3*9=27
4*1= 4 4*2= 8 4*3=12 4*4=16 4*5=20 4*6=24 4*7=28 4*8=32 4*9=36
5*1= 5 5*2=10 5*3=15 5*4=20 5*5=25 5*6=30 5*7=35 5*8=40 5*9=45
6*1= 6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36 6*7=42 6*8=48 6*9=54
7*1= 7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49 7*8=56 7*9=63
8*1= 8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64 8*9=72
9*1= 9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```



# 반복문 - while

- while은 특정 조건을 채울 때까지 계속 반복시키고 싶을 때 쓴다
  - 돈이 없어질 때까지 무한 구매 등

형식

```
while ( 조건식 )
    반복할 문장 ;
```

예제

```
i = 1;
```

```
while ( i <= 10 )
```

조건식

```
    printf ("%d ", i++ );
```

반복할 문장

# 반복문 - while

- 조건식이 거짓이 될 때까지 무한히 반복한다

형식

```
while ( 조건식 )
    반복할 문장 ;
```

예제

```
i = 1;
```

```
while ( i <= 10 )
```

조건식

```
    printf ("%d ", i++ );
```

반복할 문장

# while의 사용 예

```
01: /* Ex05_11.c */
02: #include <stdio.h>
03:
04: int main(void)
05: {
06:     int i;
07:
08:     i = 1;
09:     while ( i <= 10 )
10:         printf("%d ", i++);
11:
12:     printf("\n");
13:
14:     return 0;
15: }
```

← while의 사용

실행 결과

1 2 3 4 5 6 7 8 9 10

# for와 while

- for를 while로 변경할 때는 while문 앞에 초기식을 쓰고 while 블록 안쪽의 맨 끝에 증감식을 사용함

## for 문

```
for ( 초기식 ; 조건식 ; 증감식 )  
{  
    문장;  
}
```

## while 문

```
초기식;  
while ( 조건식 )  
{  
    문장;  
    증감식;  
}
```

# 반복문 – do while

## 한 가지 차이를 제외하고는 while과 동일

- 제일 처음, 일단 검사를 하지 않고 반복할 문장을 실행하고 본다

### 형식

```
do{
    반복할 문장 ;
} while (조건식);
```

### 예제


```
i=1;
do {
    printf("%d ", i++ );
} while ( i <= 10 );
```

반복할 문장

조건식

# do while의 사용 예

```
01: /* Ex05_15.c */
02: #include <stdio.h>
03:
04: int main(void)
05: {
06:     int i;
07:
08:     i = 1;
09:     do
10:         printf("%d ", i++);
11:     while ( i <= 10 );
12:     printf("\n");
13:
14:     return 0;
15: }
```



do while의 사용

실행 결과

1 2 3 4 5 6 7 8 9 10

# for, while과 do while의 차이

- for와 while은 조건식을 먼저 검사해서 조건식이 참인 경우에만 문장을 수행하지만, do while은 일단 먼저 문장을 수행한 다음에 조건식을 검사함

for 문	while 문	do while 문
<pre>for ( i = 1 ; i &lt; 0 ; i++ )     printf("%d ", i);</pre>	<pre>i = 1; while( i &lt; 0 )     printf("%d ", i++);</pre>	<pre>i = 1; do {     printf("%d ", i); } while ( i &lt; 0 );</pre>
<p>수행결과 :</p> <p>조건식이 거짓이므로 반복할 문장이 한 번도 수행되지 않습니다.</p>	<p>수행결과 :</p> <p>반드시 한 번은 수행됩니다.</p>	<p>수행결과 :</p> <p>반드시 한 번은 수행됩니다.</p>

## ❑ 분기문을 이용하면 실행 순서를 변경할 수 있다.

- ❖ **break** : 반복문을 탈출한다.
- ❖ **continue** : 반복문의 시작 부분으로 이동한다.
- ❖ **return** : 함수를 호출한 곳으로 돌아간다.
- ❖ **goto** : 레이블이 지정한 위치로 이동한다.

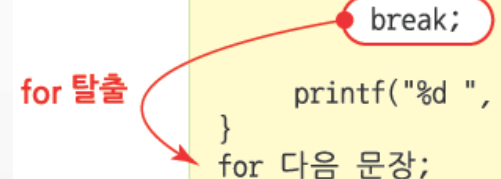
## ❑ break

- ❖ switch문 안에 사용하면 switch를 탈출해서 switch의 다음 문장으로 이동함
- ❖ for, while, do while 등의 반복문 안에서 사용하면 반복문을 빠져나감

### break를 이용한 루프 탈출

```
for ( i = 1 ; i <= 10 ; i++ )  
{  
    if ( i % 2 == 0 )  
        break;  
    printf("%d ", i);  
}  
for 다음 문장;
```

for 탈출





# break의 사용 예

```
01: /* Ex05_16.c */
```

```
02: #include <stdio.h>
```

```
03:
```

```
04: int main(void)
```

```
05: {
```

```
06:     int i;
```

```
07:
```

```
08:     for ( i = 1 ; i <= 10 ; i++ )
```

```
09:     {
```

```
10:         if ( i % 2 == 0 )
```

```
11:             break;
```

```
12:             printf("%d ", i);
```

```
13:     }
```

```
14:     printf("\n");
```

```
15:
```

```
16:     return 0;
```

```
17: }
```

break를 만나면  
for 루프 탈출

i가 1~10인 동안  
수행되는 for 문

실행 결과

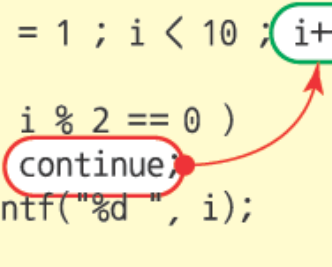
1

# 분기문 - continue

- 반복문 안에서 `continue`를 만나면 루프의 시작 부분으로 이동해서 조건문 검사부터 다시 계속함

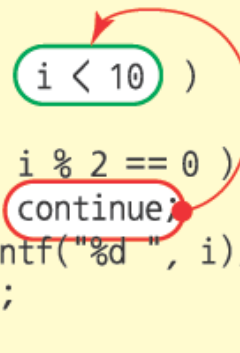
## for 문

```
for ( i = 1 ; i < 10 ; i++ )  
{  
    if( i % 2 == 0 )  
        continue;  
    printf("%d ", i);  
}
```



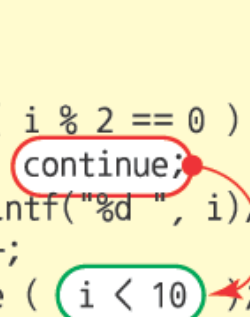
## while 문

```
i = 1;  
while ( i < 10 )  
{  
    if( i % 2 == 0 )  
        continue;  
    printf("%d ", i);  
    i++;  
}
```



## do while 문

```
i = 1;  
do {  
    if( i % 2 == 0 )  
        continue;  
    printf("%d ", i);  
    i++;  
} while ( i < 10 );
```



# 분기문 - goto

- 프로그램 수행 중 제어의 흐름을 프로그램의 특정 위치로 이동하려면 goto 문을 사용함
- Goto문을 사용하려면 먼저 이동할 문장을 가리키는 레이블(label)이 필요함
- 레이블을 정의할 때는 레이블을 구별하기 위한 이름과 콜론(:)이 필요함
- goto문으로 제어의 흐름을 갑자기 아무데로나 이동하게 되면, 프로그램이 이해하기 어려워지므로 꼭 필요한 경우가 아니면 goto문을 사용하지 않는 것이 좋음

# 분기문 - return

- 프로그램 수행 중에 return문을 만나면 함수를 호출한 곳으로 되돌아감
- Main 함수 안에서 return문을 만나면 프로그램이 종료됨

## 실습 2

- 파일명: 학번\_이름\_7주차/factorial.c
- 입력 받은 정수 N에 대하여 팩토리얼(N!)을 구하는 프로그램을 작성하시오.
  - ▶  $0! = 1$
  - ▶  $N! = N * N-1 * N-2 * \dots * 2 * 1$