



C 프로그래밍 및 실습

C Programming

이지민

# CHAP 05 제어문

# 학습목표

제어문인 조건문, 반복문, 분기문에 대해 알아본다.

조건문인 if와 switch의 사용방법과 사용시 주의사항에 대해 알아본다.

반복문인 for, while, do while의 사용방법과 사용시 주의사항에 대해 알아본다.

분기문인 break, continue, goto, return에 대해 알아본다.

# 목차



## 조건문

- if
- switch



## 반복문

- for
- while
- Do while



## 분기문

- break
- continue
- goto
- return

# 제어문

❖ 제어문 : 특정 문장을 수행하거나 수행하지 않도록 선택하거나, 특정 문장을 여러 번 반복 수행하게 만드는 것

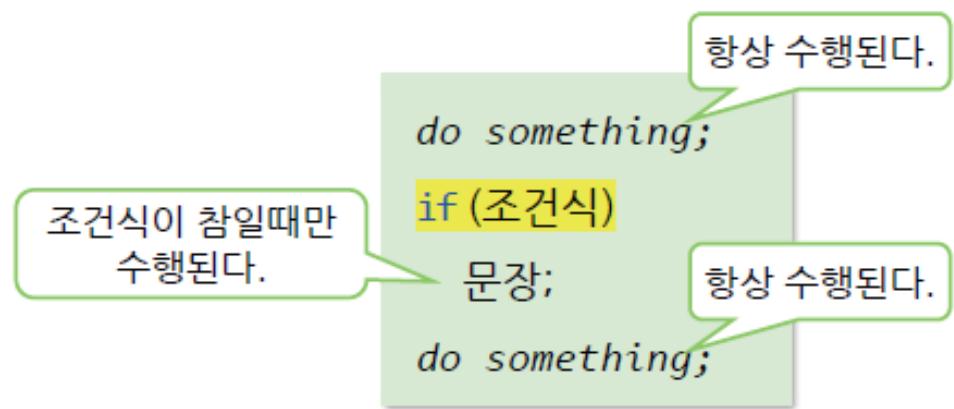
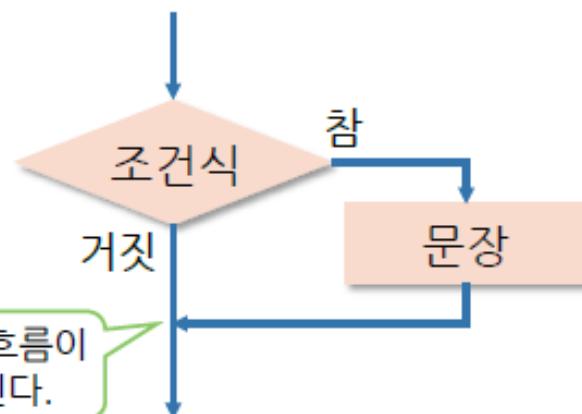
종류	C 구문	설명
조건문	if	조건식이 참이면 문장을 수행한다.
	switch	정수식의 값에 따라 수행할 문장을 선택한다.
반복문	for	조건식이 참인 동안 문장을 반복 수행한다.
	while	
	do while	
분기문	break	switch나 반복문을 빠져나간다.
	continue	반복문의 처음이나 끝으로 이동한다.
	goto	지정된 레이블의 문장으로 이동한다.
	return	함수를 호출한 곳으로 돌아간다.

# 조건문\_if

## ❖ 기본적인 if

- if문은 () 안에 있는 조건식이 참이면 주어진 문장을 수행하고, 거짓이면 수행하지 않는다.

형식	<code>if (조건식) 문장;</code>
사용예	<code>if (num &lt; 0) printf("음수입니다.");</code>



# 조건문\_if

## ❖ if의 사용 예(1/2)

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main(void)
05 {
06     int score;
07
08     printf("점수? ");
09     scanf("%d", &score);
10
11     if (score < 70)
12         printf("재시!!!\n"); ----- score < 70인 경우에만 수행
13
14     // if문의 다음 문장에서 실행의 흐름이 다시
15     printf("다음 수업은 일주일 후입니다.\n"); ----- if문의 다음 문장에서
16 }
```

### 실행 결과

점수? 67  
재시!!!

if의 조건식이  
참인 경우

다음 수업은 일주일 후입니다.

### 실행 결과

점수? 99

if의 조건식이  
거짓인 경우

다음 수업은 일주일 후입니다.

score < 70인 경우에만 수행

if문의 다음 문장에서  
실행의 흐름이 다시  
합쳐진다.

# 조건문\_if

## ❖ 복합문(compound statement)

- if의 조건식이 참일 때 수행할 문장이 여러 개면, **수행할 문장들을 { }로 묶어 주어야 한다.**
- if에서 수행할 문장이 하나일 때도 { }를 써주는 것이 좋다.

```
if (score < 70)      // 수행할 문장이 2개 이상이면 { }로 묶어준다.  
{  
    printf("재시!!!\n");  
    printf("재시는 90점 이상이어야 통과입니다\n");  
}
```

다음 줄에  
써준다.

```
if (num < 0)  
{  
    printf("음수입니다.");  
    num = -num;  
}
```

{와 열을  
맞춰준다.

공간 절약을 위해  
제어문 옆에 써준다.

```
if (num < 0) {  
    printf("음수입니다.");  
    num = -num;  
}
```

들여쓰기

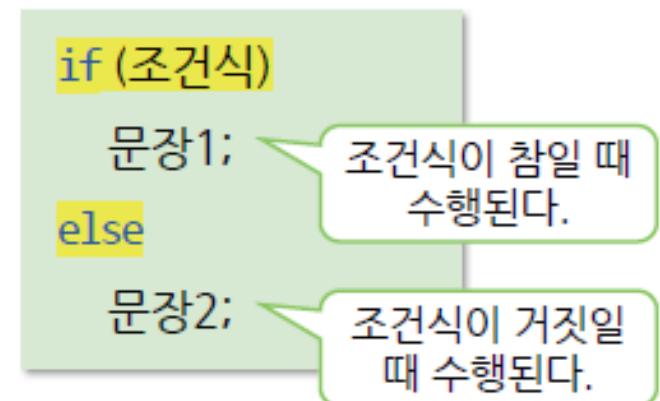
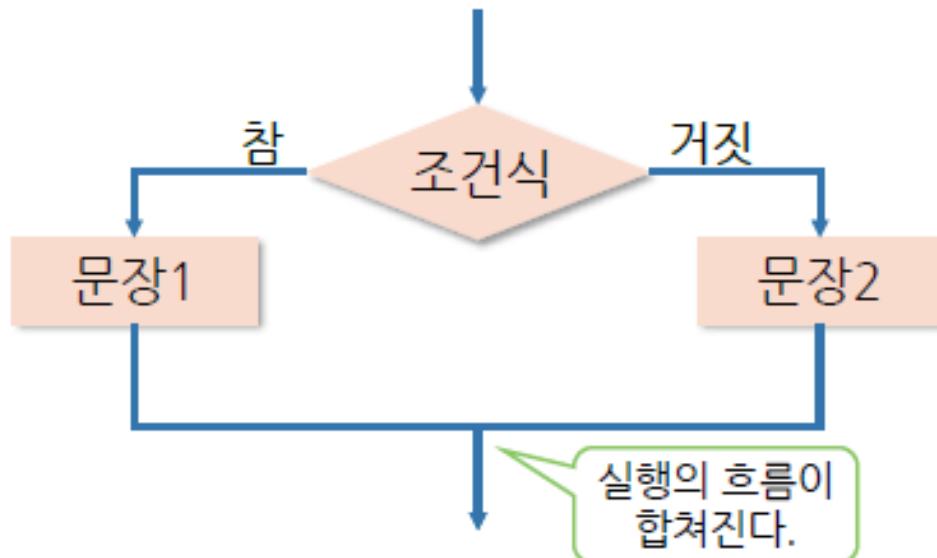
제어문과 열  
을 맞춰준다.

# 조건문\_if

## ❖ if else

- if의 조건식이 참이면 if 다음의 문장을 수행하고, 거짓이면 else 다음의 문장을 수행한다.

형식	if (조건식) 문장1; else 문장2;	사용예
		if (num < 0) printf("음수입니다."); else printf("양수입니다.");



# 조건문\_if

## ❖ if else

- else는 if의 조건식에 대하여 '그렇지 않으면'이라는 의미로 사용됨

```
if (num % 2 == 0)  
    printf("짝수입니다.");  
else  
    printf("홀수입니다.");
```

if else를 사용하는 경우

```
if (num % 2 == 0)  
    printf("짝수입니다.");  
if (num % 2 == 1)  
    printf("홀수입니다.");
```

2개의 if를 사용하는 경우

조건문을 2번 수행하므로  
비효율적이다.

# 조건문\_if

## ❖ if else의 사용 예(1/2)

```
if (score < 70)
    printf("재시!!!\n");
if (score >= 70)
    printf("통과!!!\n");
```

score < 70이  
거짓인  
경우에 해당

```
if (score < 70)
    printf("재시!!!\n");
else // 그렇지 않으면
    printf("통과!!!\n");
```

- if else 대신 조건 연산자를 이용해서 같은 코드를 작성할 수 있다.

if else를 이용하는 경우

```
if (num < 0)
    abs = -num;
else
    abs = num;
```

=

조건 연산자를 이용하는 경우

if의 조건식      else  
abs = num < 0 ? -num : num;

참일때  
수식의 값

거짓일때  
수식의 값

# 조건문\_if

## ❖ if else의 사용 예(2/2)

```
03 int main(void)
04 {
05     int score;
06
07     printf("퀴즈 점수를 입력하세요: ");
08     scanf("%d", &score);
09
10    if (score < 70)
11        printf("재시!!!\n"); // score < 70인 경우에 수행된다.
12    else
13        printf("통과!!!\n"); // score >= 70인 경우에 수행된다.
14
15    return 0;
16 }
```

### 실행결과

퀴즈 점수를 입력하세요: 90  
통과!!!

점수1과 점수2가 모두 60 이상일 경우에만 "시험 합격!"을 출력

# 조건문\_if

## ❖ 중첩된 if

- if문 안에 포함된 if문

8세 이상이면 유료,  
8세 미만은 무료

```
int fee;
if (age >= 8)
{
    if (age >= 65)
        fee = 5000;
    else
        fee = 10000;
}
else
{
    fee = 0;
}
```

65세 이상은 경로 우대,  
아니면 정상 요금

// 중첩된 if

- 이 코드에서 {}를 생략하면 코드의 의미가 혼동될 수 있음. else는 들여쓰기와 상관없이 항상 가장 가까운 if와 결합

# 조건문\_if

## ❖ 중첩된 if의 사용 예

```
05     int age, fee;  
06  
07     printf("나이? ");  
08     scanf("%d", &age);  
09  
10    if (age >= 8) {           중첩된 if  
11        if (age >= 65) {      // if문 안에 다른 if문을 포함할 수 있다.  
12            fee = 5000;  
13        }  
14        else {  
15            fee = 10000;  
16        }  
17    }  
18    else {  
19        fee = 0;  
20    }  
21    printf("입장료: %d원\n", fee);
```

실행결과

나이? 20

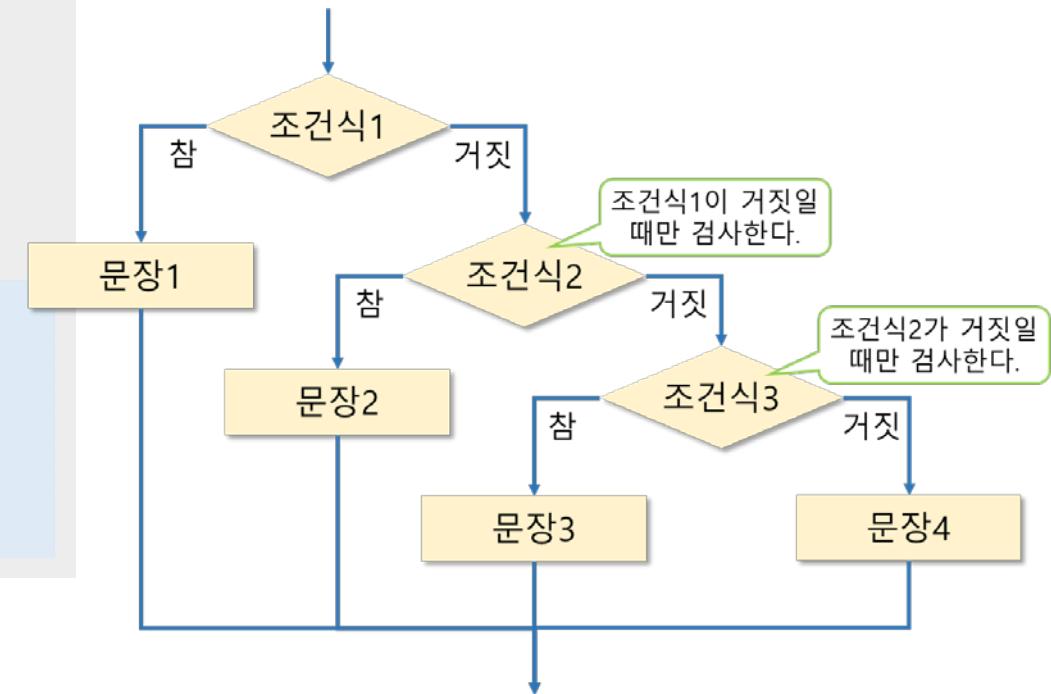
입장료: 10000원

# 조건문\_if

## ❖ if~else if(1/2)

- **if~else if문은 선택적으로 코드를 수행하게 만들 때 유용하게 사용**
    - else 블록 안에 다른 문장 없이 또 다른 if문만 들어있을 때, else if로 작성할 수 있음-여러가지 조건을 순서대로 검사
    - 텍스트 기반의 메뉴 처리

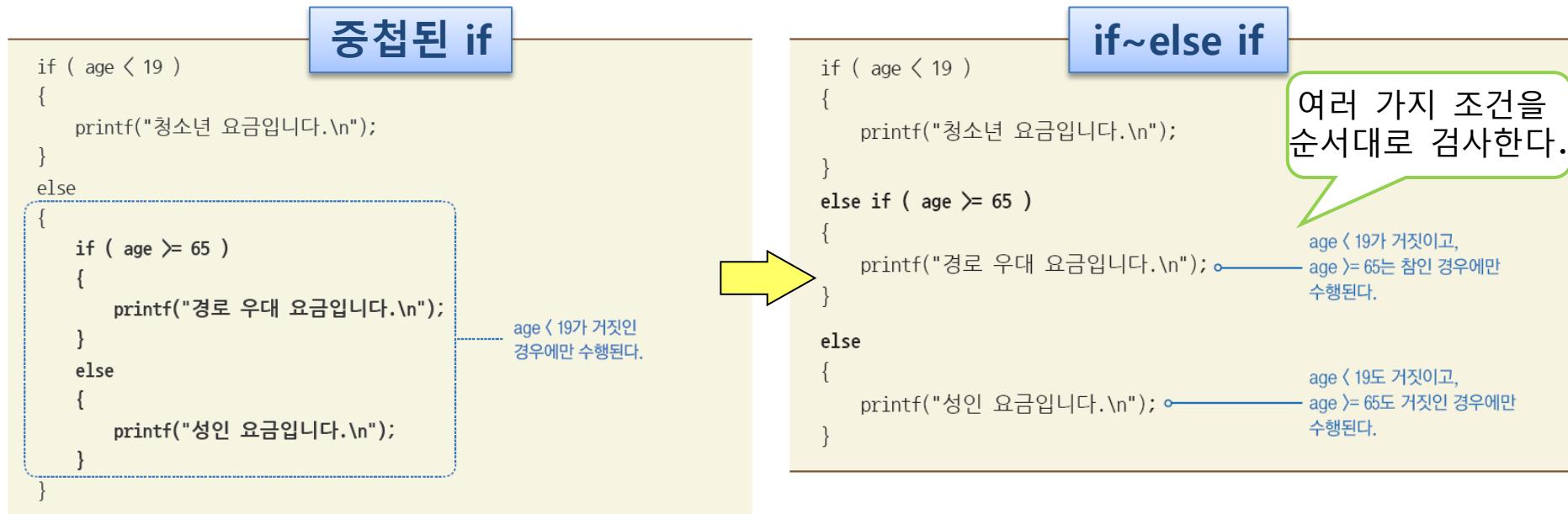
형식	if (조건식1) 문장1; else if (조건식2) 문장2; else 문장3;
사용예	if (num < 0) printf("음수입니다."); else if (num > 0) printf("양수입니다."); else printf("0입니다.");



# 조건문\_if

## ❖ if~else if(2/2)

- 중첩된 if : if 안에 다른 if문을 사용
- if~else if : 중첩된 if 중에서 if else if의 형태로 작성할 수 있는 if문
  - 여러 가지 조건을 차례차례 검사할 때 유용



# 조건문\_if

## ❖ if~else if (슬라이드 15p 중첩된 if 프로그램 수정)

```
05     int age, fee;  
06  
07     printf("나이? ");  
08     scanf("%d", &age);  
09  
10    if (age < 8) {  
11        fee = 0;  
12    }  
13    else if (age >= 65) { // age >= 8 && age >= 65라는 의미  
14        fee = 5000;  
15    }  
16    else {  
17        fee = 10000;  
18    }  
19    printf("입장료: %d원\n", fee);
```

실행결과

나이? 20  
입장료: 10000원

# 조건문\_if

## ❖ if~else if 사용 예\_텍스트기반 메뉴처리(1/2)

```
03 int main(void)
04 {
05     int menu;                      // 선택된 메뉴 번호
06     char filename[32] = "test.avi"; // 디폴트로 재생할 파일 이름
07
08     printf("1.파일 열기\n");        // 메뉴를 출력한다.
09     printf("2.재생\n");
10     printf("3.재생 옵션\n");
11     printf("선택: ");
12
13     scanf("%d", &menu);           // 메뉴 번호를 입력받는다.
14     if (menu == 1) {              // 파일 열기 메뉴
15         printf("재생할 파일 이름? ");
16         scanf("%s", filename);
17     }
```

# 조건문\_if

## ❖ if~else if 사용 예\_텍스트기반 메뉴처리(2/2)

```
18     else if (menu == 2) {           // 재생 메뉴
19         printf("%s를 재생합니다.\n", filename);
20     }
21     else if (menu == 3) {           // 재생 옵션 메뉴
22         printf("재생 옵션을 선택합니다.\n");
23     }
24     else {                         // 1~30이 아닌 메뉴 번호를 선택하는 경우
25         printf("잘못 선택하셨습니다.\n");
26     }
27
28     return 0;
29 }
```

### 실행결과

```
1.파일 열기
2.재생
3.재생 옵션
선택: 1
재생할 파일 이름? movie.avi
```

# 조건문\_if

## ❖ if~else if 사용 예\_사칙연산 계산기(1/2)

```
03 int main(void)
04 {
05     int a, b;          // 피연산자
06     char op;           // 연산자 기호를 문자로 저장할 변수
07
08     printf("수식? ");
09     scanf("%d %c %d", &a, &op, &b);      // 10 + 30 형태로 입력받는다.
10
11     if (op == '+') {
12         printf("%d + %d = %d\n", a, b, a + b);
13     }
14     else if (op == '-') {
15         printf("%d - %d = %d\n", a, b, a - b);
16     }
```

# 조건문\_if

## ❖ if~else if 사용 예\_사칙연산 계산기(1/2)

```
17     else if (op == '*') {  
18         printf("%d * %d = %d\n", a, b, a * b);  
19     }  
20     else if (op == '/') {  
21         if (b != 0) // 중첩된 if  
22             printf("%d / %d = %.2f\n", a, b, (double)a / b);  
23         printf("0으로 나눌 수 없습니다.\n");  
24     }  
25     else {           // +, -, *, /가 아닌 경우  
26         printf("잘못된 수식입니다.\n");  
27     }  
28  
29     return 0;  
30 }
```

a/b를 실수로 구하려면  
double로 형 변환

### 실행결과

수식? 78 + 21  
78 + 21 = 99

피연산자와 연산자 사이에  
빈칸을 넣어도 되고 안 넣어도 된다.

# 조건문\_if

## ❖ 다중 if

- 서로 독립적인 조건을 여러 개 비교하는 경우
- 각각의 if문은 else if로 연결되지 않음

```
fee = 10000;  
  
if (age >= 65)  
    fee -= 5000;  
  
if (is_local == 1)  
    fee -= -1000;
```

65세 이상이면  
5000원 할인

지역 주민이면  
1000원 할인

65세 이상과 지역 주민은  
서로 독립적인 조건

65세 이상이면서  
지역주민일 수도 있  
고  
아닐 수도 있다.

# 조건문\_if

## ❖ if문 비교

### 중첩된 if

```
if (age >= 8) {  
    if (age >= 65)  
        fee = 5000;  
    else  
        fee = 10000;  
}  
else  
{  
    fee = 0;  
}
```

### else if

```
if (age < 8)  
{  
    fee = 0;  
}  
else if (age >= 65)  
{  
    fee = 5000;  
}  
else  
{  
    fee = 10000;  
}
```

### 다중 if

```
fee = 10000;  
  
if (age >= 65)  
{  
    fee -= 5000;  
}  
  
if (is_local == 1)  
{  
    fee -= 1000;  
}
```

바깥쪽 if가 참일 때  
중첩된 if를 수행한다.

첫 번째 if가 거짓일 때만  
두 번째 if를 검사한다.

```
01 #include <stdio.h>
02 void main()
03 {
04     int value;
05     printf("정수를 입력하고 Enter>");
06     scanf("%d", &value);
07     if (value>0)
08         printf("양수입니다.\n");
09     else if (value<0)
10         printf("음수입니다.\n");
11     else
12         printf("0입니다.\n");
13 }
```

입력 받은 값이  
양수인가?  
음수인가?  
0 인가를 구분

두 개의 정수(a, b)를 입력 받아  $a-b$ 의 값이 0보다 크면 "변수 a가 큽니다."를,  $a-b$ 의 값이 0보다 작으면 "변수 b가 큽니다."를, 그렇지 않으면 "같은 값을 입력했습니다."를 출력하는 프로그램을 작성하시오.

# 조건문\_switch

## ❖ 기본적인 switch(1/2)

형식

```
switch (정수식) {  
    case 정수값1:  
        문장1;  
        break;  
    case 정수값2:  
        문장2;  
        break;  
        :  
    default:  
        문장n;  
        break;  
}
```

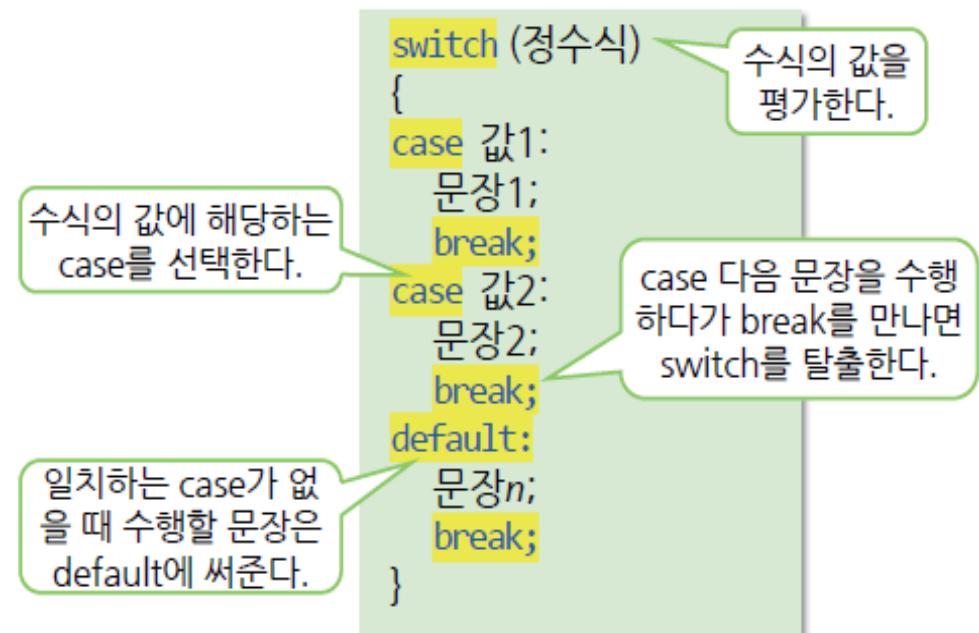
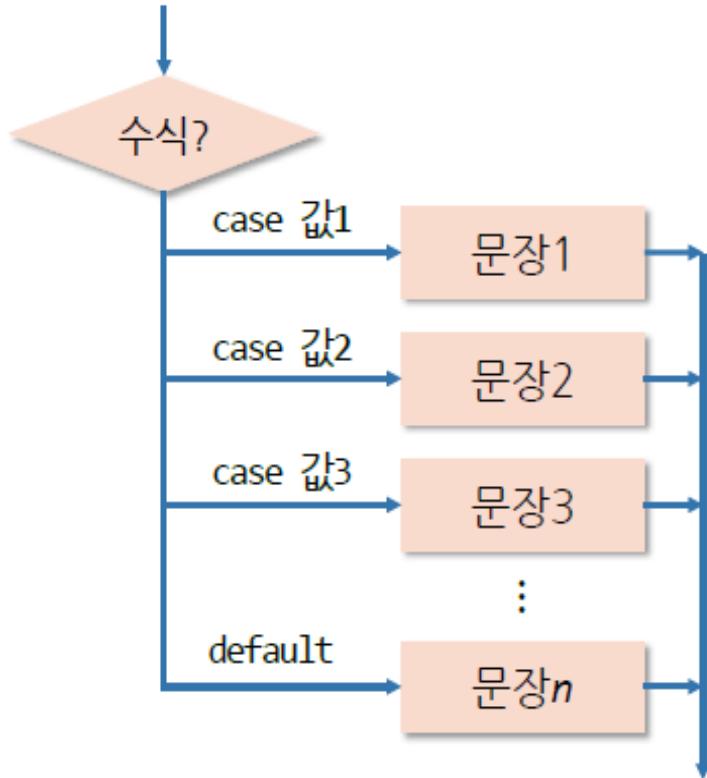
사용예

```
switch (menu) {  
    case 1:  
        printf("1번 메뉴 선택");  
        break;  
    case 2:  
        printf("2번 메뉴 선택");  
        break;  
    default:  
        printf("잘못 선택하셨습니다.\n");  
        break;  
}
```

- 정수식의 값을 평가하여, 해당 값을 가진 case문을 찾아서 case문 다음에 나열된 문장들을 수행
- switch문의 () 안에 써주는 수식의 값은 반드시 정수여야 함.
- case에서 수행할 문장의 끝에는 break를 써준다.
  - switch 안에서 break를 만나면 switch를 탈출
- 일치하는 case가 없을 때 수행할 문장은 default 다음에 써준다.

# 조건문\_switch

## ❖ 기본적인 switch(2/2)



# 조건문\_switch

## ❖ switch를 이용한 텍스트 기반의 메뉴 처리(1/2)

```
03 int main(void)
04 {
05     int menu;                      // 선택된 메뉴 번호
06     char filename[32] = "test.avi"; // 디폴트로 재생할 파일 이름
07
08     printf("1.파일 열기\n");
09     printf("2.재생\n");
10     printf("3.재생 옵션\n");
11     printf("선택: ");
12
13     scanf("%d", &menu);
14     switch (menu) {                선택된 메뉴 번호를
15         case 1:                   정수로 입력받는다.
16             printf("재생할 파일 이름? ");
17             scanf("%s", filename);
18             break;
19     }
20 }
```

선택된 메뉴 번호를  
정수로 입력받는다.

menu의 값에 따라  
case를 선택한다.

# 조건문\_switch

## ❖ switch를 이용한 텍스트 기반의 메뉴 처리(2/2)

```
19     case 2:  
20         printf("%s를 재생합니다.\n", filename);  
21         break;  
22     case 3:  
23         printf("재생 옵션을 선택합니다.\n");  
24         break;  
25     default:  
26         printf("잘못 선택하셨습니다.\n");  
27         break;  
28     }  
29  
30     return 0;  
31 }
```

case, default 다음에  
나열된 문장의 끝에  
break가 필요하다.

// 1~30이외의 메뉴 번호 선택 시

1~3이외의 메뉴  
번호 입력 시

### 실행결과

```
1.파일 열기  
2.재생  
3.재생 옵션  
선택: 1  
재생할 파일 이름? movie.avi
```

# 조건문\_switch

## ❖ switch를 이용한 사칙연산 계산기(1/2)

```
03 int main(void)
04 {
05     int a, b;                      // 피연산자
06     char op;                      // 연산자 기호를 문자로 저장할 변수
07
08     printf("수식? ");
09     scanf("%d %c %d", &a, &op, &b);    // 10 + 30 형태로 입력받는다.
10
11     switch (op) {
12         case '+':
13             printf("%d + %d = %d\n", a, b, a + b);
14             break;
15         case '-':
16             printf("%d - %d = %d\n", a, b, a - b);
17             break;
```

# 조건문\_switch

## ❖ switch를 이용한 사칙연산 계산기(2/2)

```
18     case '*':
19         printf("%d * %d = %d\n", a, b, a * b);
20         break;
21     case '/':
22         if (b != 0) // 중첩된 if
23             printf("%d / %d = %.2f\n", a, b, (double)a / b);
24         else
25             printf("0으로 나눌 수 없습니다.\n");
26         break;
27     default: // +, -, *, /가 아닌 경우
28         printf("잘못된 수식입니다.\n");
29         break;
30     }
31
32     return 0;
33 }
```

실행결과

수식? 78 + 21  
78 + 21 = 99

# 조건문\_switch

## ❖ switch문의 수행 순서

menu에 2가 입력된 경우 switch의 수행 순서

```
switch (menu) {  
    case 1:  
        printf("재생할 파일 이름? ");  
        scanf("%s", filename);  
        break;  
    case 2:  
        printf("%s를 재생합니다.\n", filename);  
        break;  
    case 3:  
        printf("switch를 선택합니다.\n");  
        break; // 탈출한다.  
    default:  
        printf("잘못 선택하셨습니다.\n");  
        break;  
}
```

① 정수식의 값을 평가한다.

② 값이 일치하는 case를 찾는다.

③ case 다음 문장을 수행한다.

④ switch를 탈출한다.

10\*20 입력

10\*20 입력

switch문의 수행 순서

```
switch( op ) {  
    case '+':  
        printf("%d + %d = %d\n", a, b, a + b);  
        break;  
    case '-':  
        printf("%d - %d = %d\n", a, b, a - b);  
        break;  
    case '*':  
        printf("%d * %d = %d\n", a, b, a * b);  
        break; // 탈출합니다  
    case '/':  
        printf("%d / %d = %d\n", a, b, a / b);  
        break;  
    default:  
        printf("계산할 수 없습니다.\n");  
        break;  
}
```

① 정수식의 값을 계산합니다.

② case를 찾습니다.

③ case 다음의 문장을 수행합니다.

# 조건문\_switch문을 else if로 바꾸기

## ❖ else if를 이용한 텍스트 기반의 메뉴 처리(1/2)

```
03 int main(void)
04 {
05     int menu;                      // 선택된 메뉴 번호
06     char filename[32] = "test.avi"; // 디폴트로 재생할 파일 이름
07
08     printf("1.파일 열기\n");        // 메뉴를 출력한다.
09     printf("2.재생\n");
10     printf("3.재생 옵션\n");
11     printf("선택: ");
12
13     scanf("%d", &menu);           // 메뉴 번호를 입력받는다.
14     if (menu == 1) {              // 파일 열기 메뉴
15         printf("재생할 파일 이름? ");
16         scanf("%s", filename);
17     }
```

# 조건문\_switch문을 else if로 바꾸기

## ❖ else if를 이용한 텍스트 기반의 메뉴 처리(2/2)

```
18     else if (menu == 2) {           // 재생 메뉴  
19         printf("%s를 재생합니다.\n", filename);  
20     }  
21     else if (menu == 3) {           // 재생 옵션 메뉴  
22         printf("재생 옵션을 선택합니다.\n");  
23     }  
24     else {                         // 1~3이 아닌 메뉴 번호를 선택하는 경우  
25         printf("잘못 선택하셨습니다.\n");  
26     }  
27  
28     return 0;  
29 }
```

### 실행결과

1.파일 열기  
2.재생  
3.재생 옵션  
선택: 1  
재생할 파일 이름? movie.avi

# 조건문\_switch문을 else if로 바꾸기

## ❖ else if를 이용한 사칙연산 계산기(1/2)

```
03 int main(void)
04 {
05     int a, b;          // 피연산자
06     char op;           // 연산자 기호를 문자로 저장할 변수
07
08     printf("수식? ");
09     scanf("%d %c %d", &a, &op, &b);      // 10 + 30 형태로 입력받는다.
10
11     if (op == '+') {
12         printf("%d + %d = %d\n", a, b, a + b);
13     }
14     else if (op == '-') {
15         printf("%d - %d = %d\n", a, b, a - b);
16     }
```

# 조건문\_switch문을 else if로 바꾸기

## ❖ else if를 이용한 사칙연산 계산기(2/2)

```
17     else if (op == '*') {  
18         printf("%d * %d = %d\n", a, b, a * b);  
19     }  
20     else if (op == '/') {  
21         if (b != 0) // 중첩된 if  
22             printf("%d / %d = %.2f\n", a, b, (double)a / b);  
23         printf("0으로 나눌 수 없습니다.\n");  
24     }  
25     else {          // +, -, *, /가 아닌 경우  
26         printf("잘못된 수식입니다.\n");  
27     }  
28  
29     return 0;  
30 }
```

a/b를 실수로 구하려면  
double로 형 변환

실행결과

수식? 78 + 21  
78 + 21 = 99

피연산자와 연산자 사이에  
빈칸을 넣어도 되고 안 넣어도 된다.

# 조건문\_switch

## ❖ Switch VS. if-else if(1/3)

- switch와 else if 중 어떤 것을 사용할지는 프로그래머가 결정할 수 있다.

비교할 값이 2개 미만일 때는 if를 사용한다.

```
switch (num % 2) {  
    case 0:  
        printf("even\n");  
        break;  
    case 1:  
        printf("odd\n");  
        break;  
}
```

```
if (num % 2 == 0) {  
    printf("even\n");  
}  
else {  
    printf("odd\n");  
}
```

==로 비교할 값이  
2개 미만인 경우

# 조건문\_switch

## ❖ Switch VS. if-else if(2/3)

비교할 값이 2개 이상일 때는 switch를 사용한다.

else-if를 사용하면 menu값을 여러 번 비교해야 함

```
if (menu == 1) {  
    printf("메뉴1");  
}  
else if (menu == 2) {  
    printf("메뉴2");  
}  
else if (menu == 3) {  
    printf("메뉴3");  
}  
else {  
    printf("에러");  
}
```

```
switch (menu) {  
case 1:  
    printf("메뉴1");  
    break;  
case 2:  
    printf("메뉴2");  
    break;  
case 3:  
    printf("메뉴3");  
    break;  
default:  
    printf("에러");  
    break;  
}
```

switch를 사용하면  
menu값은 한번만  
평가하면 됨

==로 비교할 값이 2  
개 이상인 경우

특정 정수값을 여러 번 비교할 때는 switch문이 더 유리

# 조건문\_switch

## ❖ Switch VS. if-else if(3/3)

범위를 비교할 때는 else if를 사용한다.

```
switch (score/10) {  
    case 9:  
        grade = 'A';  
        break;  
    case 8:  
        grade = 'B';  
        break;  
    case 7:  
        grade = 'C';  
        break;  
    case 6:  
        grade = 'D';  
        break;  
    default:  
        grade = 'F';  
        break;  
}
```

```
if (score >= 90) {  
    grade = 'A';  
}  
else if (score >= 80) {  
    grade = 'B';  
}  
else if (score >= 70) {  
    grade = 'C';  
}  
else if (score >= 60) {  
    grade = 'D';  
}  
else {  
    grade = 'F';  
}
```

값의 범위를 비교한다.

단순한 정수 값 비교가 아닌 경우

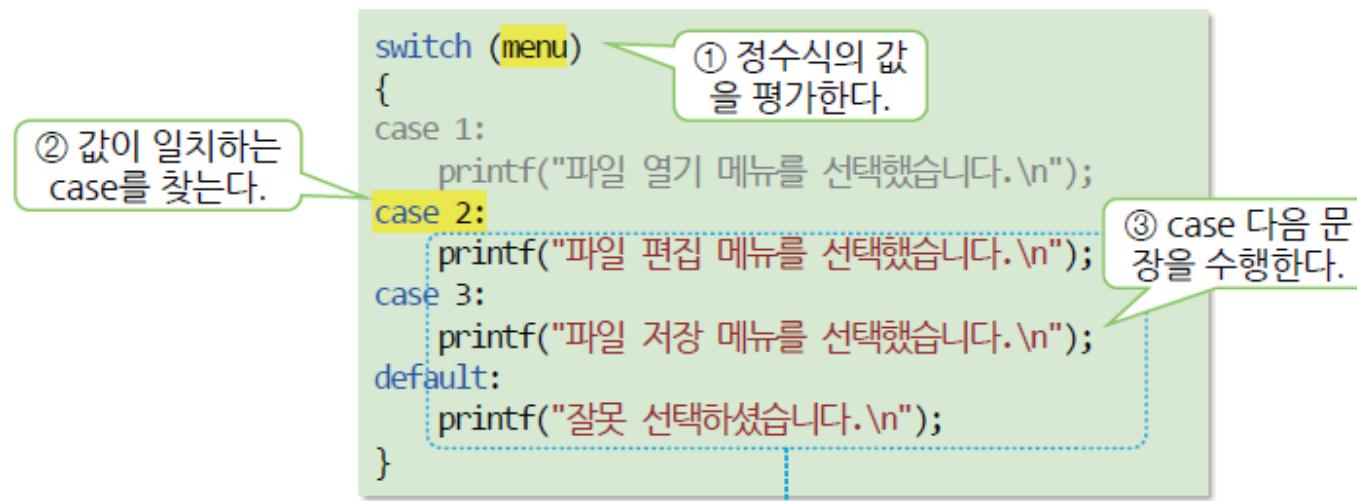
# 조건문\_switch

## ❖ switch 사용 시 주의사항(1/5)

### ▪ switch문에서 break는 생략할 수 있다.

- 실수로 break를 빠뜨리면, break를 만나거나 switch의 끝을 만날 때까지 나 타나는 모든 문장들을 수행한다.
- switch문이 올바르게 수행되도록 하려면 case마다 break를 써주어야 하며, default에도 break를 써주는 것이 좋다.

menu에 2가 입력된 경우



break나 switch의 끝을 만날 때  
까지의 문장들을 모두 수행한다.

# 조건문\_switch

## ❖ switch 사용 시 주의사항(2/5)

- switch문에서 break는 생략할 수 있다.

의도적으로 특정  
case에 대한 break를  
생략할 수도 있다.

```
switch (day_of_week) {           // 항상 0~6사이의 값
    case 1:                      // mon
        fee = 5000;               // 월요일은 할인 요금
        break;
    case 6:                      // sat
    case 0:                      // sun
        fee = 10000;              // 토,일은 주말 요금
        break;
    default:                     // 화~목은 평일 요금
        fee = 8000;
        break;
}
```

day\_of\_week가 6일 때와 0일 때  
같은 코드를 수행한다.

# 조건문\_switch

## ❖ break를 사용하지 않은 경우의 예

```
11:     switch ( op )  
12:     {  
13:         case '+':  
14:             printf("%d + %d = %d\n", a, b, a + b);  
15:         case '-':  
16:             printf("%d - %d = %d\n", a, b, a - b);  
17:         case '*':  
18:             printf("%d * %d = %d\n", a, b, a * b);  
19:         case '/':  
20:             printf("%d / %d = %d\n", a, b, a / b);  
21:         default:  
22:             printf("계산할 수 없습니다.\n");  
23:     }  
24:  
25:     return 0;  
26: }
```

Break를 생략한 경우

### 실행 결과

수식을 입력하세요 : **10 + 20**

$10 + 20 = 30$

$10 - 20 = -10$

$10 * 20 = 200$

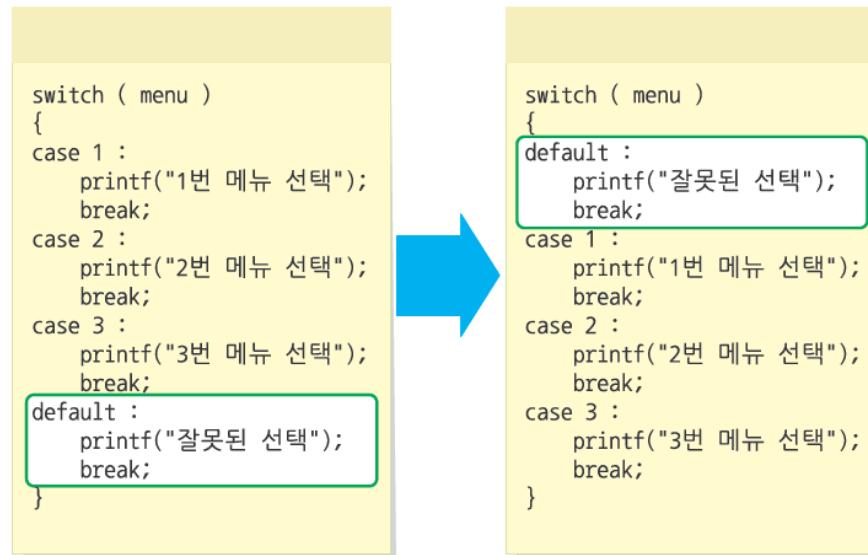
$10 / 20 = 0$

계산할 수 없습니다.

# 조건문\_switch

## ❖ switch 사용 시 주의사항(3/5)

- **switch문에서 사용되는 default도 생략할 수 있다.**
  - switch문에 일치하는 case가 없고 default도 없으면, 아무것도 수행하지 않고 switch문을 빠져나간다.
- **default는 switch문 안에 어떤 위치에도 사용할 수 있다.**
  - 보통은 switch문의 맨 마지막에 작성한다.
  - default에도 break를 써주는 것이 좋다.



default문의 위치는 관계없습니다.

# 조건문\_switch

## ❖ switch 사용 시 주의사항(4/5)

menu에 5가 입력된 경우

```
switch (menu)
{
    case 1:
        printf("파일 열기 메뉴를 선택했습니다.\n");
        break;
    case 2:
        printf("파일 편집 메뉴를 선택했습니다.\n");
        break;
    case 3:
        printf("파일\n");
        break;
    default:
        printf("잘못 선택하셨습니다.\n");
        break;
}
```

② 값이 일치하는  
case가 없으면  
default를 찾는다.

① 정수식의 값  
을 평가한다.

③ default 다음  
문장을 수행한다.  
했습니다.\n");

④ switch를  
탈출한다.

default의 위치는 상관없  
지만 일반적으로 switch  
의 끝부분에 써준다.

# 조건문\_switch

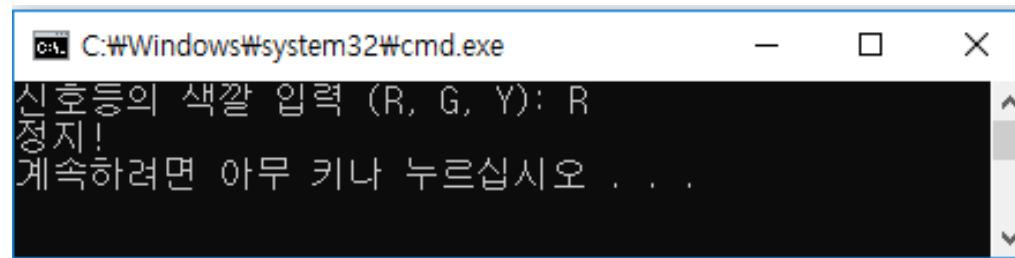
## ❖ switch 사용 시 주의사항(5/5)

- switch의 () 안에는 정수식만 사용할 수 있으며, 실수나 문자열은 사용할 수 없다.

```
float value;  
scanf("%f", &value);  
switch ( value )○————— value는 실수이므로 switch에서 사용할 수 없다.  
{  
    case 0.5 :○————— case 다음에 실수 값을 지정할 수 없다  
        value *= 0.01;  
        break;  
    case 1.5 :  
        value *= 0.02;  
        break;  
}
```

# 조건문 예제 1

- ❖ 사용자가 신호등의 색깔을 입력하면 “정지”, “주의”, “진행”과 같은 문장을 출력하는 프로그램을 작성해보자.
  - 주의 : 소문자를 입력해도 같은 결과가 나오도록 할 것!

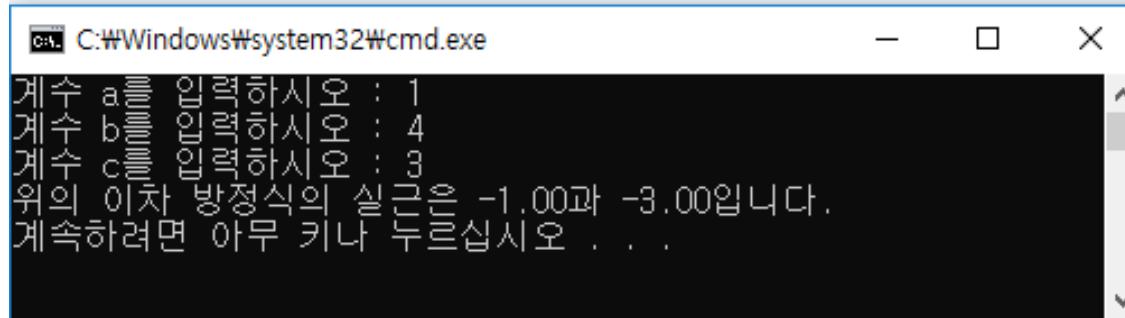


## 조건문 예제 2

❖ 이차 방정식  $ax^2 + bx + c = 0$ 의 근을 계산하는 프로그램을 작성해보자.

- 사용자에게 이차 방정식의 계수  $a$ ,  $b$ ,  $c$ 를 입력하도록 한다.
- 만약  $a$ 가 0이면 근은  $\frac{c}{b}$ 이다.
- 만약 판별식  $\sqrt{b^2 - 4ac}$ 가 음수이면 실근은 존재하지 않는다.
  - 제곱근을 구하기위해, 전처리기에 math.h를 추가하고, 제곱근 구하는 함수 sqrt를 사용한다. (ex)sqrt(b\*b-4\*a\*c)
- 위의 조건에 해당하지 않으면 다음과 같은 공식을 이용하여 실근을 구한다.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



The screenshot shows a command-line interface window titled 'cmd.exe' with the path 'C:\Windows\system32\cmd.exe'. The window displays the following text:  
계수 a를 입력하시오 : 1  
계수 b를 입력하시오 : 4  
계수 c를 입력하시오 : 3  
위의 이차 방정식의 실근은 -1.00과 -3.00입니다.  
계속하려면 아무 키나 누르십시오 . . .

# 반복문

## ❖ 반복문

- 같은 코드를 여러 번 반복할 수 있도록 하는 제어문
- 코드를 반복해서 수행해야 할 때 코드를 복사해서 작성하는 대신 반복문을 사용
- 루프(loop)라고도 부른다.

## ❖ 반복문의 종류

- **for**
- **while**
- **do while**

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

같은 코드를 복사해서 작성한다.

# 반복문\_for

## ❖ 기본적인 for(1/2)

형식

**for** (초기식; 조건식; 증감식)  
반복할문장;

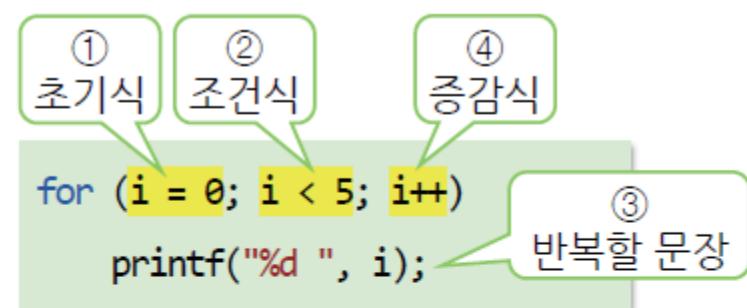
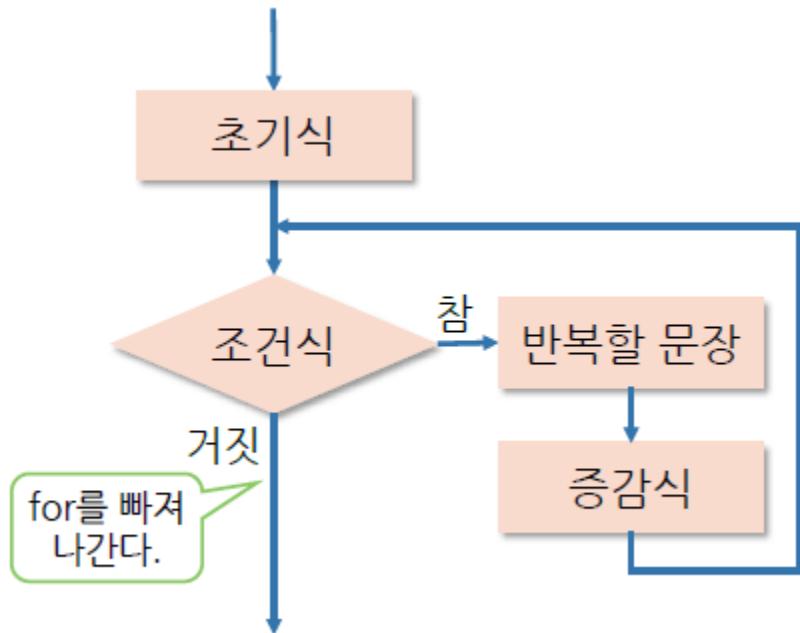
사용예

```
for (i = 0; i < 10; i++)  
    printf("%d ", i);
```

- **for**문은 정해진 횟수만큼 반복 수행할 때 주로 사용
- **for**문은 **초기식**, **조건식**, **증감식**과 **반복할 문장**으로 구성
- 반복할 문장이 여러 개일 때는 {}로 묶어줌
- **for**에서는 먼저 초기식을 수행한 다음에 조건식을 검사
  - 조건식이 참이면 반복할 문장을 수행하고 나서 증감식을 수행

# 반복문\_for

## ❖ 기본적인 for(2/2)



# 반복문\_for문의 루프제어 변수

## ❖루프 제어 변수의 선언

### ANSI C

for 앞에 i를 선언해야 한다.

```
int i;  
for (i = 0; i < 10; i++)  
    printf("%d ", i);
```

초기식에 선언문이 올 수 없다.

```
printf("%d ", i);  
for (i = 10; i > 0; i--)  
    printf("%d ", i);
```

for 다음에서도 i를 계속 사용할 수 있다.

### C99

초기식에 선언문이 올 수 있다.

```
for (int i = 0; i < 10; i++)  
    printf("%d ", i);  
  
//printf("%d ", i);  
for (int i = 10; i > 0; i--)  
    printf("%d ", i);
```

i의 사용 범위

for 다음에서 i를 사용할 수 없다.

i를 다시 만들어서 사용할 수는 있다.

# 반복문\_for문의 루프제어 변수

## ❖ for문의 루프 제어 변수

- for의 초기식, 조건식, 증감식에서 사용되는 변수
- 일반적으로 어떤 문장을 N번 반복 수행하는 용도로 사용

*do something*을 N번  
반복 수행한다.

```
for (i = 0; i < N; i++) {  
    do something;  
}
```

- for문의 반복 회차마다 루프 제어 변수의 값이 변경되어, 특정 시점에 for의 조건식이 거짓이 되어 루프를 탈출할 수 있으면 된다.

```
for (i = 10; i > 0; --i)  
    printf("%d ", i);  
  
for (i = 0; i < 20; i += 2)  
    printf("%d ", i);  
  
for (j = 1; j <= 1000000; j *= 10)  
    printf("%7d\n", j);
```

# 반복문\_for

## ❖ 입력된 정수들의 합계 구하기

```
03 int main(void)
04 {
05     int num = 0;      // 입력받은 정수를 저장할 변수
06     int sum = 0;      // 합계를 저장할 변수
07     int i;           // 루프 제어 변수
08
09     printf("정수 5개를 입력하세요: ");
10    for (i = 0; i < 5; i++) // i가 5가 되면 루프 탈출
11    {
12        scanf("%d", &num);
13        sum += num;
14    }
15    printf("합계: %d\n", sum);
16
17    return 0;
18 }
```

sum에 합계를 구해야 하므로 0으로 초기화한다.

i가 0~4일 때 입력받은 정수를 5번 더한다.

### 실행결과

정수 5개를 입력하세요: 12 32 45 63 7

합계: 159

# 반복문\_for

## ❖ for의 여러 가지 변형

- for문에서 루프 제어 변수를 여러 개 사용할 수도 있다.

```
int i, j;  
for (i = 0, j = 100; i < 10 && j > 0; i++, j /= 2)  
    printf("i = %d, j = %d\n", i, j);
```

콤마 연산자를 이용한다.

- for 문을 구성하는 초기식, 조건식, 증감식, 반복할 문장은 모두 생략할 수 있다.

for의 초기식을  
생략할 수 있다.

```
for (; i < 5; i++)  
    printf("%d ", i);
```

for의 조건식을  
생략할 수 있다.

```
for (i = 0; ; i++)  
    printf("%d ", i);
```

for의 증감식을  
생략할 수 있다.

```
for (i = 0; i < 5;) // ;  
    printf("%d ", i++);
```

### 무한 루프

for의 반복할 문장을  
생략할 수 있다.

```
for (i = 0; i < 5; )  
;
```

for의 초기식, 조건식,  
증감식을 모두 생략할 수 있다.

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

### 널 문장

### 무한 루프

# 반복문\_for

## ❖ 중첩된 for문-for문 안에 다시 for문을 사용하는 것

- 중첩된 for에서 바깥쪽 for가 N번, 안쪽 for가 M번 반복 수행하는 경우에 전체 반복 횟수는  $M * N$ 이 된다.

바깥쪽 for는  
height번 반복

```
for (i = 0; i < height; i++) {  
    for (j = 0; j < width; j++)  
        printf("%c", ch);  
    printf("\n");  
}
```

안쪽 for는  
width번 반복

# 반복문\_for

## ❖ 중첩된 for문-구구단 출력

```
01: /* 구구단 출력*/
02: #include <stdio.
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: }
```

구구단 출력  
: i는 세로, j는 가로

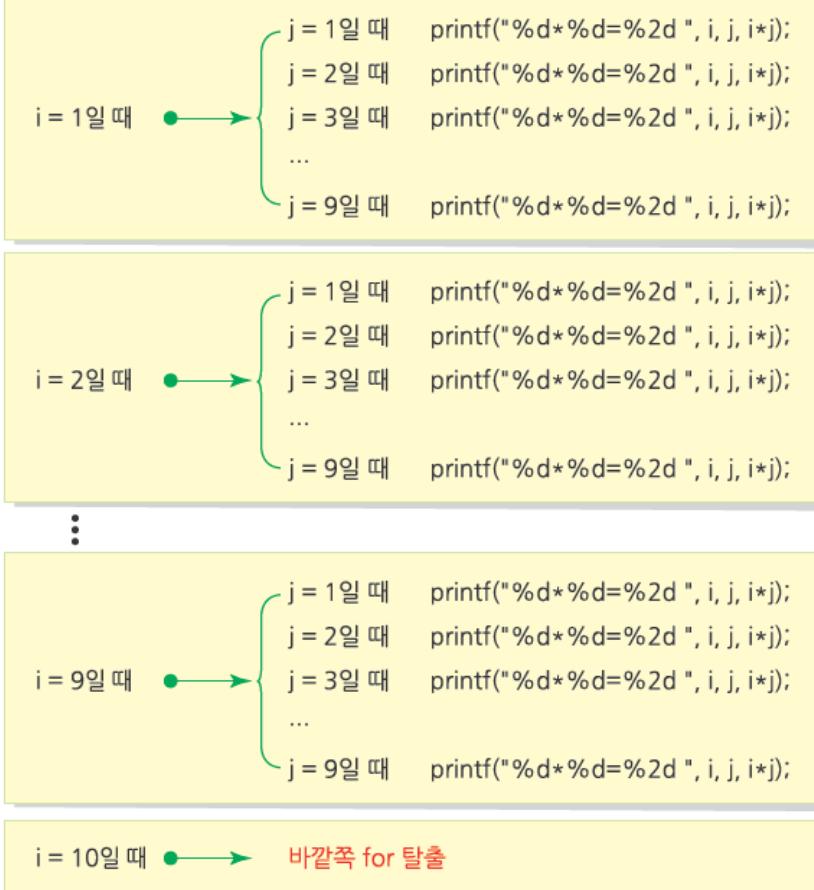
실행 결과

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\_for

## ❖ 중첩된 for의 수행 순서

```
for ( i = 1 ; i < 10 ; i++ )
{
    for ( j = 1 ; j < 10 ; j++ )
    {
        printf("%d * %d = %2d ", i, j, i*j);
    }
    printf("\n");
}
```



# 반복문\_for

## ❖ 입력된 문자로 직사각형 그리기(1/2)

```
03 int main(void)
04 {
05     int width, height;
06     char ch;
07     int i, j;
08
01     printf("직사각형의 폭과 높이? ");
02     scanf("%d %d", &width, &height);
03     printf("직사각형을 그릴 문자? ");
04     scanf(" %c", &ch);           // %c앞에 빈칸 지정
05
```

입력 버퍼에 남아 있는 공백 문자('n', ' ', 't')를 무시한다.

# 반복문\_for

## ❖ 입력된 문자로 직사각형 그리기(1/2)

```
06     for (i = 0; i < height; i++)  
07     {  
08         for (j = 0; j < width; j++)  
09             printf("%c", ch);  
10             printf("\n");  
11     }  
12  
13     return 0;  
14 }
```

// 중첩된 for

ch 문자를 가로로 width개,  
세로로 height개 출력한다.

### 실행결과

직사각형의 폭과 높이? 20 3

직사각형을 그릴 문자? \*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

# 참고-입력버퍼의 공백문자 무시

## 실행 결과

직사각형의 폭과 높이? 20 3

콘솔에서 입력한  
내용은 입력 버퍼  
로 저장된다.

입력 버퍼



scanf("%d %d", &width, &height);

공백문자를 모  
두 무시한다.

scanf("%c", &ch);

입력 버퍼에 남아 있는  
\n을 ch로 읽어온다.

# 중첩 반복문 연습

❖ 다음 반복문 안의 printf() 문장은 몇 번 반복하는가? 그리고 출력될 결과는?

```
for (i = 1; i <= 3; i++)
    for (j = 4; j > 0; j -= 2)
        printf("%d", i * j);
```

```
for (i = 1; i <= 3; i++)
    for (j = 4; j > i; j -= 2)
        printf("%d", i * j);
```

6번 : 4,2,8,4,12,6  
i=1 j=4,2  
i=2 j=4,2  
i=3 j=4,2

4번, 4,2,8,12  
i=1 j=4,2  
i=2 j=4  
i=3 j=4

# 중첩 반복문 연습

❖ 중첩된 반복문을 이용하여 아래와 같이 출력되도록 구현하시오.

```
012345  
012345  
012345  
012345  
012345  
012345
```

```
00000  
11111  
22222  
33333  
44444  
55555
```

# 중첩 반복문 연습

❖ 중첩된 반복문을 이용하여 아래와 같이 출력되도록 구현하시오.

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

# 중첩 반복문 연습

❖ 중첩된 반복문을 이용하여 아래와 같이 출력되도록 구현하시오.

```
012345  
01234  
0123  
012  
01  
0
```

```
543210  
43210  
3210  
210  
10  
0
```

# 중첩 반복문 연습

❖ 중첩된 반복문을 이용하여 아래와 같이 출력되도록 구현하시오.

```
55555  
44444  
3333  
222  
11  
0
```

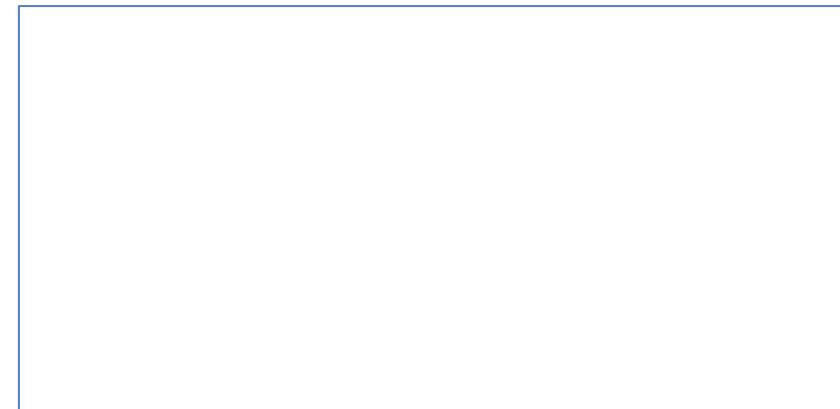
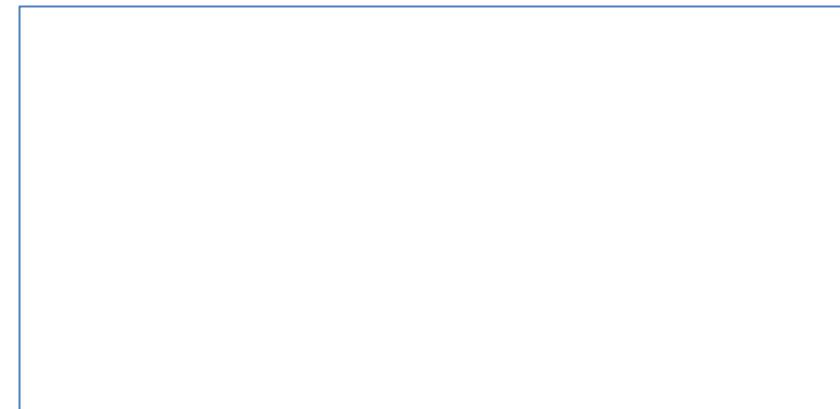
```
0  
01  
012  
0123  
01234  
012345
```

# 중첩 반복문 연습

❖ 중첩된 반복문을 이용하여 아래와 같이 출력되도록 구현하시오.

```
0  
01  
012  
0123  
01234  
012345
```

```
*  
**  
***  
****  
*****  
*****
```



# 중첩 반복문 연습

❖ 중첩된 반복문을 이용하여 아래와 같이 출력되도록 구현하시오.

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

# 중첩 반복문 연습

- ❖ 중첩된 반복문을 이용하여 아래와 같이 출력되도록 구현하시오.

```
*  
**  
***  
****  
*****  
******
```

# 반복문\_while

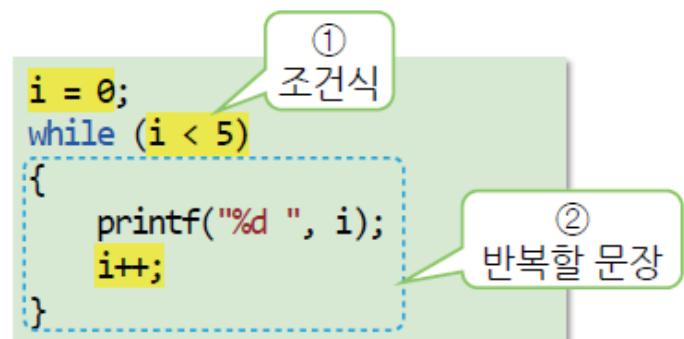
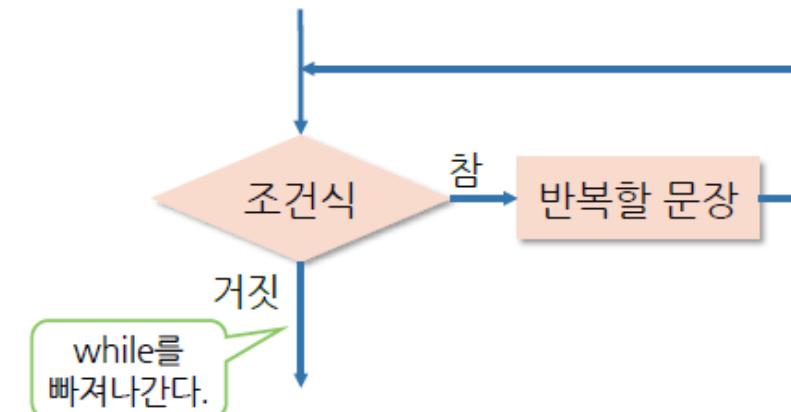
## ❖ 기본적인 while

형식

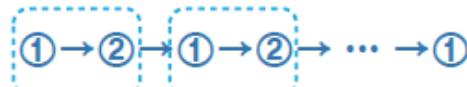
**while** (조건식)  
반복할문장;

사용예

```
i = 0;  
while (i < 10)  
    printf("%d ", i++);
```



수행 순서:

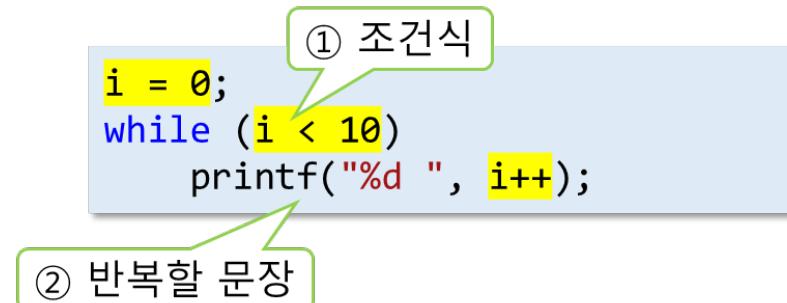
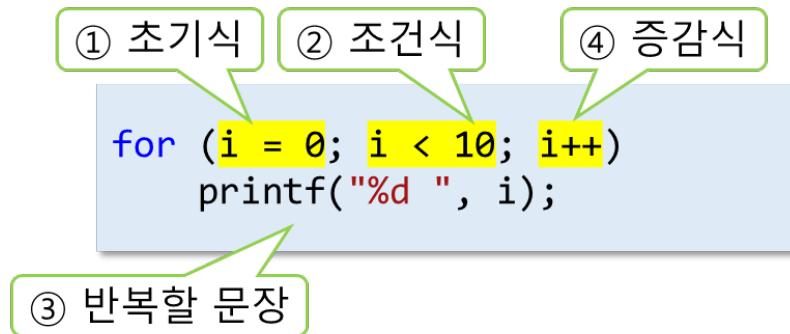


반복1회차 반복2회차

# 반복문\_while

## ❖ for와 while

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



### for의 수행 순서

①→②→③→④→②→③→④→…→②

### while의 수행 순서

①→②→①→②→…→①

# 반복문\_while

## ❖while의 사용 예

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int i = 0;
06     while (i < 10)          // i가 10이 되면 루프 탈출
07         printf("%d ", i++); //반복할 문장
08     printf("\n");
09
10     return 0;
11 }
```

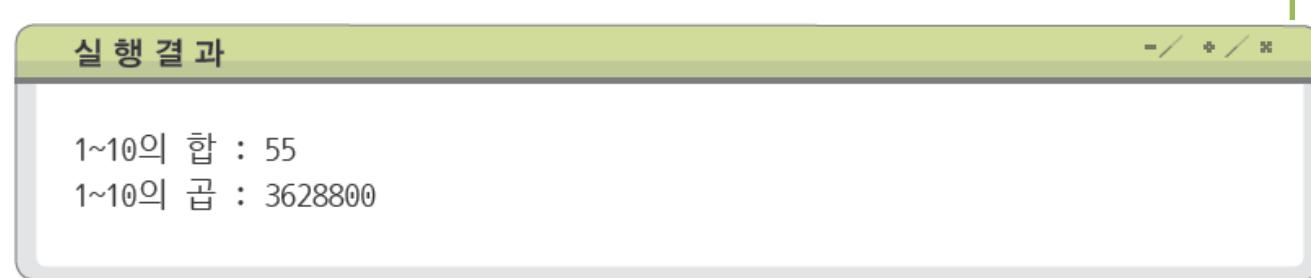
실행결과

0 1 2 3 4 5 6 7 8 9

# 반복문\_while

## ❖ while을 이용한 합계와 곱 구하기

```
01: /* Ex05_12.c */
02: #include <stdio.h>
03:
04: int main(void)
05: {
06:     int i;
07:     int sum = 0;
08:     int factorial = 1;
09:
10:     i = 1; ← for의 초기식은 while 앞에서 수행
11:     while ( i <= 10 ) ← for의 조건식을 while의 조건식으로 사용
12:     {
13:         sum += i;
14:         factorial *= i;
15:         i++; ← for의 증감식을 while의 맨 끝에서 수행
16:     }
17:
18:     printf("1~10의 합계 : %d\n", sum);
19:     printf("1~10의 곱 : %d\n", factorial);
20:
21:     return 0;
22: }
```



# 반복문\_while

## ❖ while의 사용 예\_입력된 정수들의 합계 구하기(for문과 비교)

```
03 int main(void)
04 {
05     int num = 0;
06     int sum = 0;
07     int i = 0;      // for의 초기식
08
09     printf("정수 5개를 입력하세요: ");
```

### 실행결과

```
정수 5개를 입력하세요: 12 32 45 63 7
합계: 159
```

```
10     while (i < 5)    // 조건식
11     {
12         // 반복할 문장
13         scanf("%d", &num);
14         sum += num;
15         i++;           // for의 증감식
16     }
17     printf("합계: %d\n", sum);
18
19     return 0;
20 }
```

# 반복문\_while

## ❖while의 사용예\_반복 수행되는 사칙연산 계산기(1/2)

```
03 int main(void)
04 {
05     int a, b;
06     char op;
07     char yesno = 'Y';           // 처음에 while문을 수행해야 하므로
08                               // 'Y'로 초기화 한다.
09     while (yesno == 'Y' || yesno == 'y')
10     {
11         printf("수식? ");
12         scanf("%d %c %d", &a, &op, &b);           // 10 + 30 형태로 입력받는다.
13
14         switch (op) {
15             case '+':
16                 printf("%d + %d = %d\n", a, b, a + b);
17                 break;
18             case '-':
19                 printf("%d - %d = %d\n", a, b, a - b);
20                 break;
```

// 계속 수행할지를 나타내는 변수

처음에 while문을 수행해야 하므로  
'Y'로 초기화 한다.

# 반복문\_while

## ❖ while의 사용 예\_반복 수행되는 사칙연산 계산기(2/2)

```
21     case '*':
22         printf("%d * %d = %d\n", a, b, a * b);
23         break;
24     case '/':
25         if (b != 0)
26             printf("%d / %d = %.2f\n", a, b, (double)a / b);
27         else
28             printf("0으로 나눌 수 없습니다.\n");
29         break;
30     default:           // +, -, *, /가 아닌 경우
31         printf("잘못된 수식입니다.\n");
32         break;
33     }
34     printf("계속 하시겠습니까(Y/N)? ");
35     scanf(" %c", &yesno);
36 }
37
38 return 0;
39 }
```

공백 문자를 무시하고 문자를 입력받으려면  
%c앞에 빈칸을 지정한다.

### 실행결과

```
수식? 3 + 10
3 + 10 = 13
계속 하시겠습니까(Y/N)? y
수식? 78 * 5
78 * 5 = 390
계속 하시겠습니까(Y/N)? y
수식? 56 & 3
잘못된 수식입니다.
계속 하시겠습니까(Y/N)? n
```

# 반복문\_while

## ❖ 무한 루프를 이용한 구구단 프로그램

```
01: /* Ex05_14.c */
02: #include <stdio.h>
03:
04: int main(void)
05: {
06:     int num; ← 구구단의 단수를 입력받기 위한 int형 변수 num 선언
07:     int i;
08:
09:     while ( 1 ) ← while을 이용한 무한 루프
10:    {
11:        printf("정수를 입력하세요 (Ctrl+C 입력 시 종료) : ");
12:        scanf("%d", &num);
13:
14:        for ( i = 1 ; i < 10 ; i++ )
15:            printf("%d*%d=%2d ", num, i, num * i);
16:        printf("\n");
17:    } ← 실행 결과
18:
19:    return 0;
20: }
```

실행 결과

```
정수를 입력하세요 (Ctrl+C 입력시 종료) : 3
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
정수를 입력하세요 (Ctrl+C 입력시 종료) : 5
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
정수를 입력하세요 (Ctrl+C 입력시 종료) : 9
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
정수를 입력하세요 (Ctrl+C 입력시 종료) :
```

# 반복문 \_do while

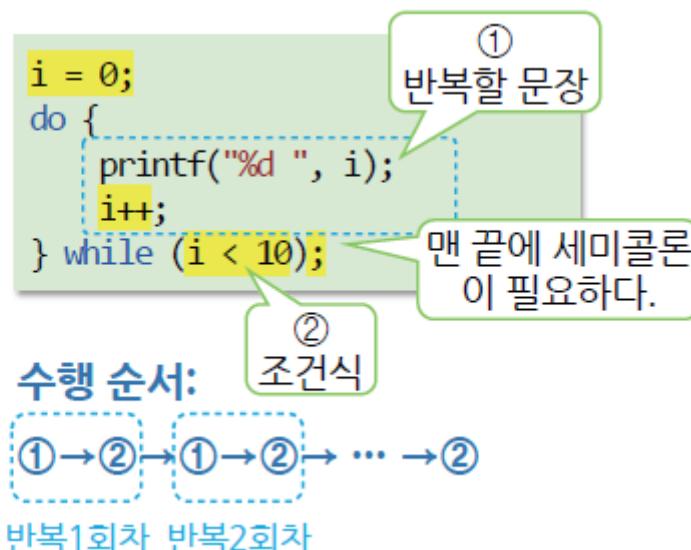
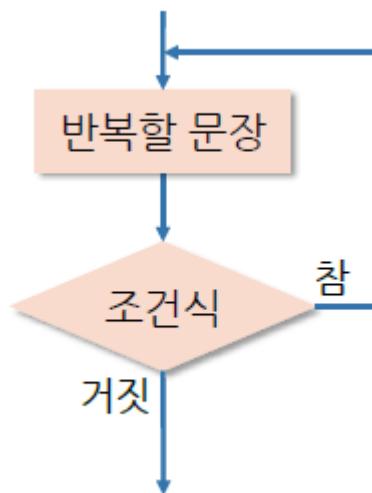
## ❖ 기본적인 do while

형식

do  
반복할문장;  
while (조건식);

사용예

```
i = 0;  
do  
    printf("%d ", i++);  
while (i < 10);
```



# 반복문\_do while

## ❖ do while의 사용 예

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int i = 0;
06     do
07         printf("%d ", i++);    //반복할 문장
08     while (i < 10);        // i가 10이 되면 루프 탈출
09     printf("\n");
10
11     return 0;
12 }
```

실행결과

0 1 2 3 4 5 6 7 8 9

# 반복문 \_do while

## ❖ for, while과 do while의 차이

- for와 while은 조건식을 먼저 검사해서 조건식이 참인 경우에만 문장을 수행하지만, do while은 일단 먼저 문장을 수행한 다음에 조건식을 검사한다.
  - 반복문의 끝 부분에서 반복문의 탈출 여부를 결정해야 하는 프로그램에서는 while보다는 do while을 사용하는 것이 자연스럽다.

for	while	do while
<p>처음부터 조건식 이 거짓인 경우</p> <pre>for (i = 0; i &lt; 0; i++)     printf("%d ", i);</pre>	<pre>i = 0; while (i &lt; 0)     printf("%d ", i++);</pre>	<pre>i = 0; do     printf("%d ", i++); while (i &lt; 0);</pre>
<p>실행 결과</p> 	<p>실행 결과</p> 	<p>실행 결과</p>  <p>문장이 한번도 수행되지 않는다.</p> <p>문장이 한번도 수행되지 않는다.</p> <p>반드시 한번은 수행된다.</p>

# 반복문\_do while

## ❖ do while을 이용한 사칙연산 계산기(1/2)

```
03 int main(void)
04 {
05     int a, b;
06     char op;
07     char yesno; // yesno의 초기화를 생략할 수 있다.
08
09     do { // 일단 한번 수행한다.
10         printf("수식? ");
11         scanf("%d %c %d", &a, &op, &b); // 10 + 30 형태로 입력받는다.
12
13         switch (op) {
14             case '+':
15                 printf("%d + %d = %d\n", a, b, a + b);
16                 break;
17             case '-':
18                 printf("%d - %d = %d\n", a, b, a - b);
19                 break;
```

# 반복문\_do while

## ❖ do while을 이용한 사칙연산 계산기(2/2)

```
20     case '*':
21         printf("%d * %d = %d\n", a, b, a * b);
22         break;
23     case '/':
24         if (b != 0)
25             printf("%d / %d = %.2f\n", a, b, (double)a / b);
26         else
27             printf("0으로 나눌 수 없습니다.\n");
28         break;
29     default: // +, -, *, /가 아닌 경우
30         printf("잘못된 수식입니다.\n");
31         break;
32     }
33     printf("계속 하시겠습니까(Y/N)? ");
34     scanf(" %c", &yesno);
35 } while (yesno == 'Y' || yesno == 'y');
36
37     return 0;
38 }
```

### 실행결과

```
수식? 3 + 10
3 + 10 = 13
계속 하시겠습니까(Y/N)? y
수식? 78 * 5
78 * 5 = 390
계속 하시겠습니까(Y/N)? y
수식? 56 & 3
잘못된 수식입니다.
계속 하시겠습니까(Y/N)? n
```

# 반복문\_do while

## ❖ 루프 탈출 위치

### while

```
while (루프탈출조건)
{
    문장1; 루프 시작 부분에서
    문장2; 만 탈출할 수 있다.
    문장3;
    :
}
```

### do while

```
do {
    문장1;
    문장2;
    문장3;
    :
} while (루프탈출조건);

루프 끝 부분에서
만 탈출할 수 있다.
```

### break를 함께 사용하는 경우

```
while (1) 무한 루프
{
    문장1;
    문장2;
    if(루프탈출조건)
        break;
    문장3;
    :
}
```

원하는 위치에서 언제든지 탈출할 수 있다.

## 반복문\_무한 루프

❖ **for**에서는 무한 루프를 만들기 위해서 조건식을 생략한다.

- **for ( ; ; )**

❖ **while**에서는 조건식을 생략할 수 없는 대신에 항상 참인 값을 조건식에 써준다.

- **while (1)**

```
while (1)          // 무한 루프
{
    printf("infinite loop");
}
```

❖ 무한 루프를 수행중인 프로그램은 **Ctrl+C**로 강제 종료해야 한다.

- 무한 루프를 안전하게 탈출하려면 **break**를 이용한다.

# 반복문\_무한 루프

## ❖ 무한 루프를 이용한 메뉴 처리(1/2)

```
03 int main(void)
04 {
05     int menu;
06     char filename[32] = "test.avi";
07
08     while (1)      // 무한 루프이므로 Ctrl+C로 강제 종료해야 한다.
09     {
10         printf("1.파일 열기\n");
11         printf("2.재생\n");
12         printf("3.재생 옵션\n");
13         printf("선택: ");
14
15         scanf("%d", &menu);
16         switch (menu) {
17             case 1:
18                 printf("재생할 파일 이름? ");
19                 scanf("%s", filename);
20                 break;
```

# 반복문\_무한 루프

## ❖ 무한 루프를 이용한 메뉴 처리(2/2)

```
21     case 2:  
22         printf("%s를 재생합니다.\n", filename);  
23         break;  
24     case 3:  
25         printf("재생 옵션을 선택합니다.\n");  
26         break;  
27     default:  
28         printf("잘못 선택하셨습니다.\n");  
29         break;  
30     }  
31 }  
32  
33     return 0;  
34 }
```

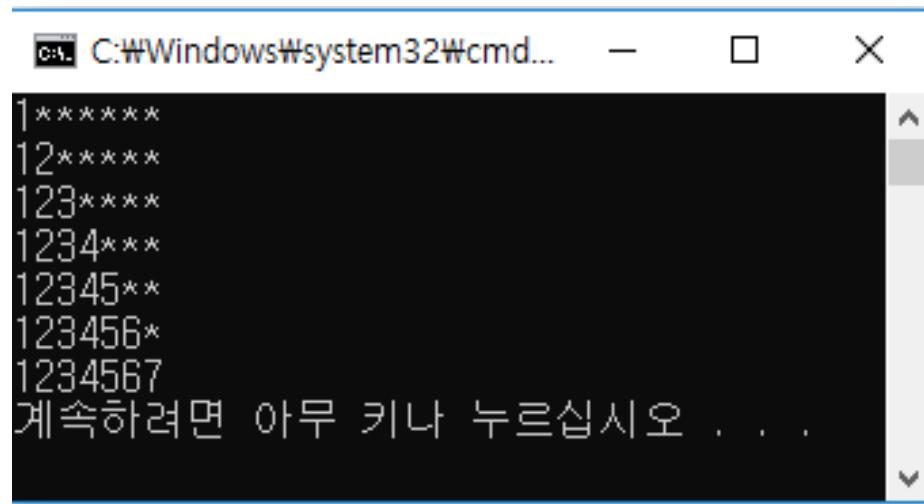
### 실행결과

```
1.파일 열기  
2.재생  
3.재생 옵션  
선택: 1  
재생할 파일 이름? test.avi  
1.파일 열기  
2.재생  
3.재생 옵션  
선택:
```

Ctrl+C로 강제 종료해야 한다.

# 반복문 예제 1

❖ 다음과 같은 출력을 생성하는 프로그램을 작성하여 보자.



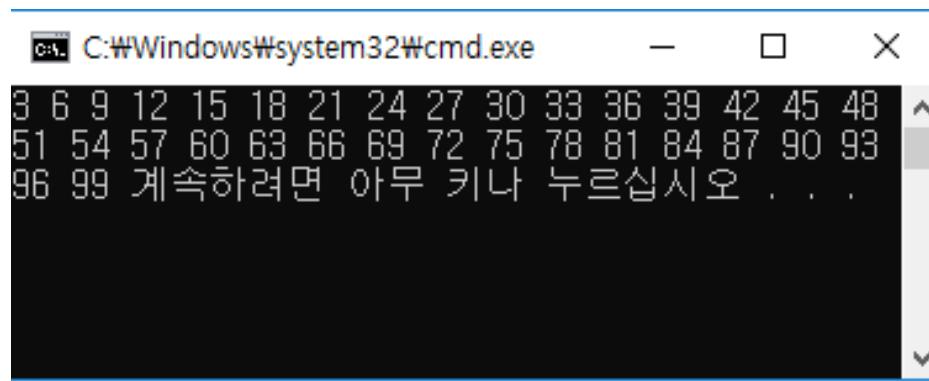
The screenshot shows a Windows Command Prompt window titled 'cmd' with the path 'C:\Windows\system32\cmd...'. The window displays the following text:

```
1*****
2*****
12*****
123****
1234***  
12345**  
123456*  
1234567  
계속하려면 아무 키나 누르십시오 . . .
```

The text consists of a series of numbers followed by asterisks (\*). The first seven lines each have a different number of digits and asterisks. The last line is a prompt for the user to press a key to continue.

# 반복문 예제 2

❖ 1부터 100까지의 자연수 중에서 3의 배수를 출력하여 보자.

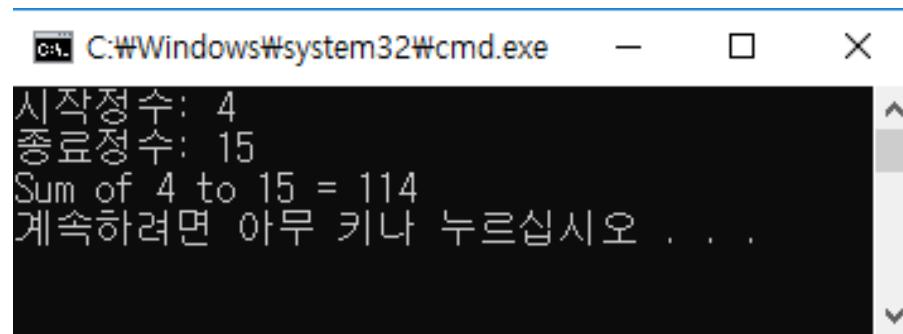


A screenshot of a Windows Command Prompt window titled "cmd C:\Windows\system32\cmd.exe". The window contains the following text:

```
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48
51 54 57 60 63 66 69 72 75 78 81 84 87 90 93
96 99 계속하려면 아무 키나 누르십시오 . . .
```

# 반복문 예제 3

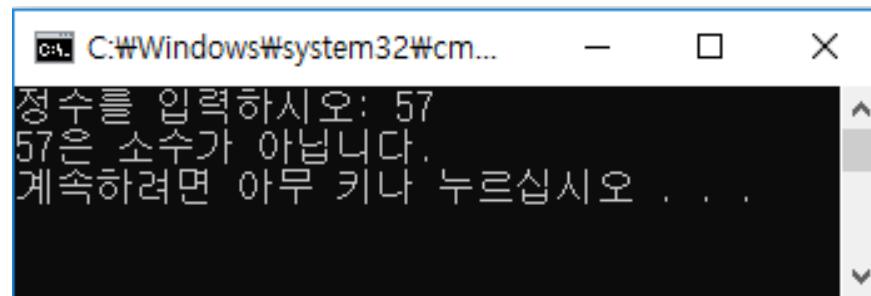
- ❖ 사용자로부터 정수 x, y를 입력 받아서 x에서 y까지의 합을 구하는 프로그램을 작성하라.



```
C:\Windows\system32\cmd.exe
시작정수: 4
종료정수: 15
Sum of 4 to 15 = 114
계속하려면 아무 키나 누르십시오 . . .
```

## 반복문 예제 4

❖ 사용자가 입력한 수가 소수인지 아닌지를 출력하는 프로그램을 작성하라. 소수는 1과 자기 자신 이외에는 약수를 가지지 않아야 한다. 약수는 %연산자를 이용하여서 검사할 수 있다. 즉, i가 5의 약수라면  $i\%5$ 가 0이 된다.



# 분기문

❖ 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.

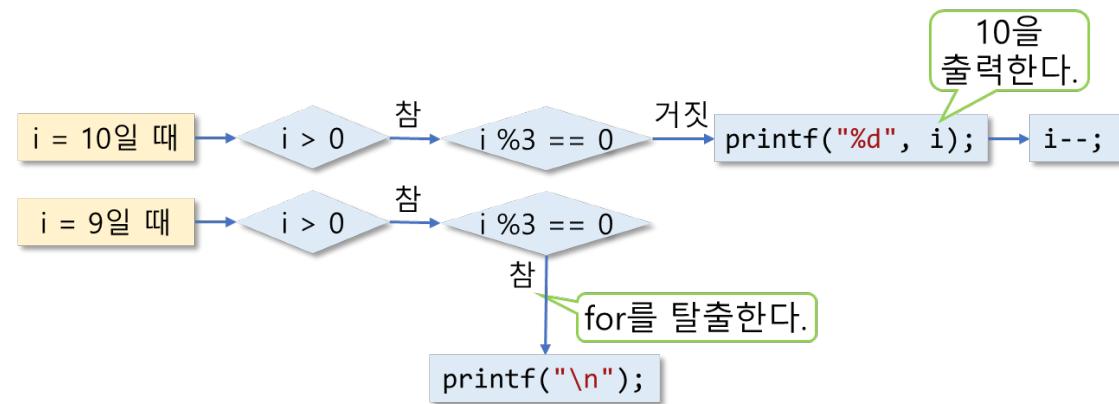
# 분기문\_break

## ❖ break

- switch문 안에 사용하면 제어의 흐름이 switch를 탈출해서 switch의 다음 문장으로 이동한다.
- for, while, do while 등의 반복문 안에서 사용하면 반복문을 빠져 나가게 된다.

```
for (i = 10; i > 0; i--)  
{  
    if (i % 3 == 0)  
        break;  
    printf("%d ", i);  
}  
printf("\n");
```

10



# 분기문\_break

## ❖break의 사용 예

```
03 int main(void)
04 {
05     int i;
06
07     for (i = 10; i > 0; i--)
08     {
09         if (i % 3 == 0)      // 루프 탈출 조건
10             break;
11         printf("%d ", i);
12     }
13     printf("\n");
14
15     return 0;
16 }
```

실행결과

10

# 분기문\_break

## ❖ break의 활용

- 무한 루프와 break를 이용하면 원하는 위치에서 루프 탈출 조건을 검사해서 루프를 빠져나올 수 있다.

```
while (1) {  
    printf("1. 파일 열기\n");  
    printf("2. 재생\n");  
    printf("3. 재생 옵션\n");  
    printf("선택: ");  
    scanf("%d", &menu);  
  
    if (menu == 0) break;  
  
    switch (menu) {  
    case 1:  
        printf("재생할 파일 이름? ");  
        scanf("%s", filename);  
        break;  
    :  
    }  
}
```

menu를 입력받은 다음 루프 탈출 조건을 검사한다.

# 분기문\_break

## ❖ 종료 메뉴를 가진 메뉴 처리 프로그램(1/2)

```
03 int main(void)
04 {
05     int menu;
06     char filename[32] = "test.avi";
07
08     while (1)
09     {
10         printf("0.종료\n");
11         printf("1.파일 열기\n");
12         printf("2.재생\n");
13         printf("3.재생 옵션\n");
14         printf("선택: ");
15
16         scanf("%d", &menu);
17         if (menu == 0)      // menu를 입력받은 다음 루프 탈출 조건을 검사한다.
18             break;
```

# 분기문\_break

## ❖ 종료 메뉴를 가진 메뉴 처리 프로그램(2/2)

```
20     switch (menu) {  
21         case 1:  
22             printf("재생할 파일 이름? ");  
23             scanf("%s", filename);  
24             break;  
25         case 2:  
26             printf("%s를 재생합니다.\n", filename);  
27             break;  
28         case 3:  
29             printf("재생 옵션을 선택합니다.\n");  
30             break;  
31         default:  
32             printf("잘못 선택하셨습니다.\n");  
33             break;  
34     }  
35 }  
36 printf("프로그램을 종료합니다.\n");
```

### 실행결과

```
1.파일 열기  
2.재생  
3.재생 옵션  
선택: 1  
재생할 파일 이름? test.avi  
1.파일 열기  
2.재생  
3.재생 옵션  
선택: 0  
프로그램을 종료합니다.
```

종료 메뉴를 선택해서  
프로그램을 종료한다.

# 분기문\_break

## ❖ 센티널 값을 이용한 루프 탈출

- **센티널** : 입력되는 데이터의 끝을 나타내는 특별한 값
  - 특정 값을 입력하면 프로그램을 종료하기로 미리 약속

```
while (1) {  
    printf("수식? ");  
    scanf("%d %c %d", &a, &op, &b);  
  
    if (a == 0 && op == '0' && b == 0)  
        break;  
  
    switch (op) {  
    case '+':  
        printf("%d + %d = %d\n", a, b, a + b);  
        break;  
    :  
    }  
}
```

"0 0 0"이 입력되면  
무한 루프를 탈출한다.

# 분기문\_break

## ❖ 센티널 값을 이용한 사칙연산 계산기

```
08     while (1)           무한 루프
09     {
10         printf("수식? ");
11         scanf("%d %c %d", &a, &op, &b);
12
13         // 0 0 0이 입력되면 루프를 빠져나간다.
14         if (a == 0 && op == '0' && b == 0)
15             break;           루프 탈출 조건
16
17         switch (op) {
18             case '+':
19                 printf("%d + %d = %d\n", a, b, a + b);
20                 break;
21             case '-':
22                 printf("%d - %d = %d\n", a, b, a - b);
23                 break;
```

### 실행결과

수식? 3 + 10

3 + 10 = 13

수식? 78 \* 5

78 \* 5 = 390

수식? 56 & 3

잘못된 수식입니다.

수식? 0 0 0

센티널 값이 입력되면 종료한다.

# 분기문\_continue

❖ continue-루프의 시작이나 끝 부분으로 이동한다.

- 반복문 안에서 continue를 만나면 루프의 시작 부분으로 이동해서 조건문 검사부터 다시 계속한다.

for

```
for (i = 10; i > 0; i--)  
{  
    if (i % 3 == 0)  
        continue;  
    printf("%d ", i);  
}
```

루프 시작 부분  
으로 이동한다.

while

```
i = 10;  
while (i > 0)  
{  
    if (i % 3 == 0)  
        continue;  
    printf("%d ", i);  
    i++;  
}
```

루프 시작 부분  
으로 이동한다.

do while

```
i = 10;  
do  
{  
    if (i % 3 == 0)  
        continue;  
    printf("%d ", i);  
    i++;  
} while (i > 0)
```

루프 끝 부분  
으로 이동한다.

# 분기문\_continue

## ❖ continue의 사용 예

```
03 int main(void)
04 {
05     int i;
06
07     for (i = 10; i > 0; i--)
08     {
09         if (i % 3 == 0)
10             continue;          // 루프의 시작 부분으로 이동한다.
11         printf("%d ", i);
12     }
13     printf("\n");
14
15     return 0;
16 }
```

### 실행결과

10 8 7 5 4 2 1 → 가 3의 배수일 때는 출력하지 않는다.

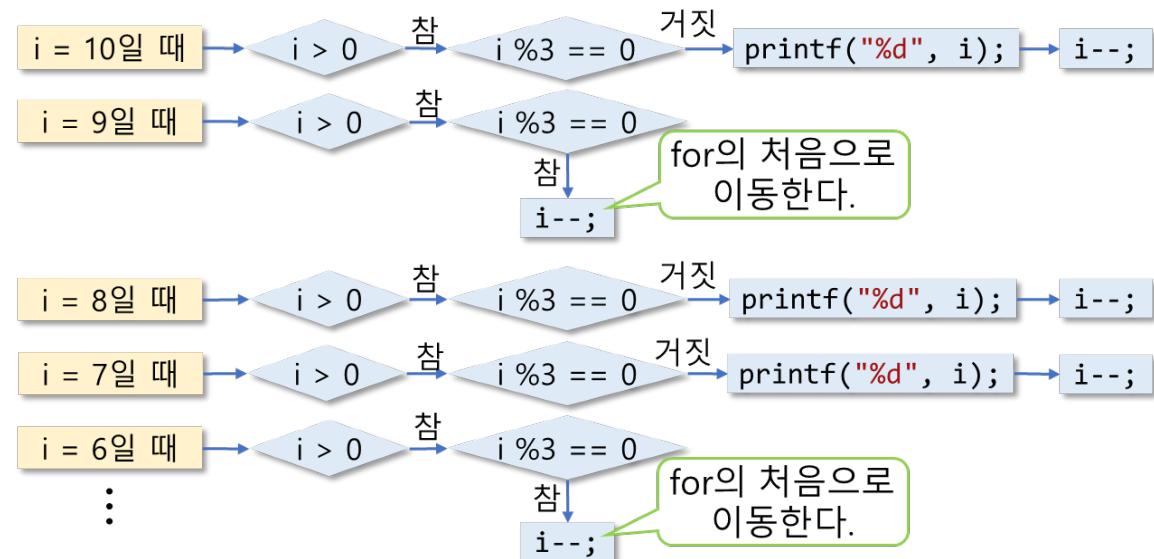
# 분기문\_continue

## ❖ continue의 수행 순서

- for문 안에서 continue를 for의 시작 부분으로 이동해서 루프를 반복한다.

```
for (i = 10; i > 0; i--)  
{  
    if (i % 3 == 0)  
        continue;  
    printf("%d ", i);  
}  
printf("\n");
```

10 8 7 5 4 2 1



# 분기문\_continue

## ❖ continue 예제

- 앞 슬라이드의 예제를 수정하여, 1~100까지 짝수의 합을 구하시오.

# 분기문\_goto

## ❖ goto

- 프로그램 수행 중 제어의 흐름을 프로그램의 특정 위치로 이동하려면 goto 문을 사용한다.
- goto문을 사용하려면 먼저 이동할 문장을 가리키는 레이블(label)이 필요하다.
- 레이블을 정의할 때는 레이블을 구별하기 위한 이름과 콜론(:)이 필요하다.
- goto문으로 제어의 흐름을 갑자기 아무데로나 이동하게 되면, 프로그램이 이해하기 어려워지므로 꼭 필요한 경우가 아니면 goto문을 사용하지 않는 것이 좋다.
- 한꺼번에 여러 개의 루프를 탈출해야 할 때 유용

```
while (1)
{
    i = 10;
    while (i > 0) {
        if (i % 3 == 0)
            goto quit;
        printf("%d ", i--);
    }
}
quit:
printf("\n");
```

# 분기문\_goto

## ❖ goto의 사용 예

```
03 int main(void)
04 {
05     int i;
06
07     for (i = 10; i > 0; i--)
08     {
09         if (i % 3 == 0)
10             goto quit;      // quit 레이블이 지정하는 문장으로 이동한다.
11         printf("%d ", i);
12     }
13 quit:
14     printf("\n");
15
16     return 0;
17 }
```

실행결과

10

# 분기문\_return

## ❖ return

- 프로그램 수행 중에 return문을 만나면 함수를 호출한 곳으로 되돌아간다.
- 리턴 값이 있는 함수에서는 return 다음에 값을 써주고, 리턴 값이 없는 함수에서는 return만 써준다.

```
void test(void)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        if (i % 3 == 0)
            return;      // 리턴 값이 없는 함수에서는 return만 써준다.
        printf("%d ", i);
    }
}
```

# 분기문\_return

## ❖ return의 사용 예

```
03 int main(void)
04 {
05     int i;
06
07     for (i = 10; i > 0; i--)
08     {
09         if (i % 3 == 0)
10             return 1;          // main 함수를 리턴시킨다. (프로그램 종료)
11         printf("%d ", i);
12     }
13     printf("\n");
14
15     return 0;
16 }
```

실행결과

10

# 학습정리

## ❖ 조건문

- if : 조건식이 참이면 문장을 수행하고, 조건식이 거짓이면 수행하지 않는다.
- switch : 정수식에 값에 따라서 다중 분기를 수행한다.

## ❖ 반복문

- for : 초기식, 조건식, 증감식으로 구성되며, 조건이 참인 동안 문장을 반복 수행한다.
- while : 조건식이 참인 동안 문장을 반복 수행한다.
- do-while : 조건식이 참인 동안 문장을 반복 수행하며, 문장을 한번 수행한 다음에 조건식을 검사한다.

## ❖ 분기문

- break : 반복문을 빠져나가게 만든다.
- continue : 반복문의 시작 부분으로 돌아가게 만든다.
- return : 함수를 호출한 곳으로 돌아가게 만든다.
- goto : 레이블이 지정한 위치로 이동하게 만든다.