
C 프로그래밍 및 실습

6. 반복문

p, 16 6235

p. 36 예제 6-15a 번호 추가

p. 36 예제 6-15a , 입력 추가

예제6.12 수정할 것

p. 39 cnt=1

세종대학교

목차

- 1) 반복문 개요
- 2) **while 문**
- 3) do-while 문
- 4) for 문
- 5) 중첩 반복
- 6) 반복문 기타

제어문 (Review)

▪ 제어문

- C 언어는 순차처리언어로, 특별한 지정이 없으면, 소스 코드 첫 줄부터 차례대로 처리
- 그러나 문제 해결 위해 처리 흐름 제어 필요 → 제어문
- C언어에서는 조건문(5장), 반복문(6장)과 같은 제어문 제공

▪ 제어문 종류

분류	종류
조건문	if 문, if~else 문, switch 문
반복문	for 문, while 문, do~while 문
기타	break 문, continue 문, goto 문, return 문

1) 반복문 개요

■ 반복문

- 특정 조건을 만족하는 동안 계속 반복하여 실행하는 문장
- 루프(loop)라고도 불림
- while 문, do while 문, for 문
- 반복문이 필요한 예
 - ✓ "Hello World"를 100번 출력하려면?
 - ✓ 1부터 100까지의 합을 구하려면?

```
// 동일한 내용을 반복할 경우
printf("Hello World\n");
printf("Hello World\n");
printf("Hello World\n");
```

```
// 일정한 규칙으로 반복하는 작업을 수행하는 경우
int sum = 1 + 2 + 3 + 4 + 5;
```

2) while 문

- while 문의 대략적인 형태
 - "Hello World" 3번 출력하기

```
int i = 1;           ⇒ 반복문 진입 전에 i=1 로 초기 설정
while( i <= 3 ) {    ⇒ i<=3 인 동안(while) 아래 문장 반복
    printf("Hello World\n");    ⇒ 반복할 문장
    i++;              ⇒ i 값 1 증가
}                    ⇒ while 문의 끝
```

- ✓ 키워드 **while** : 조건을 만족하는 **동안**~
- ✓ 변수 **i** 는 **반복 횟수를 제어**하는 역할

2) while 문

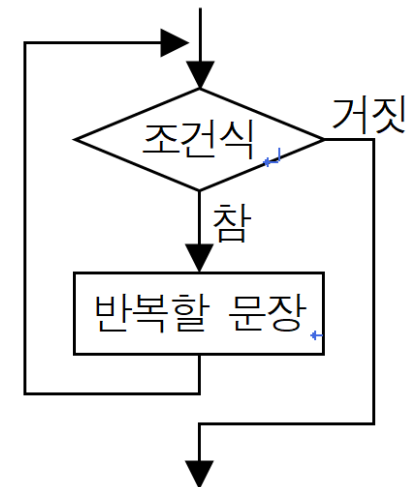
- while 문 대략적으로 살펴보기

- 조건식이 참인 동안 중괄호로 둘러 쌓인 블록 안의 문장 반복
조건식이 거짓이 되면 while문을 빠져 나옴
 - ✓ 반복할 문장이 하나이면 중괄호 생략 가능

- 구문

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

```
while( i <= 3 ) {  
    printf("Hello World\n");  
    i++;  
}
```



2) while 문

- **[프로그램 1]** 반복 횟수가 정해져 있는 예시
 - while 문을 이용하여 1부터 5까지의 합 구하기

```
int i, sum;

sum = 0;           // 합을 저장하는 변수
i = 1;             // 더하는 수를 저장하는 변수

while ( i <= 5 ) {  // i가 5보다 같거나 작은 동안
    sum += i;       // i를 sum에 더하고
    i++;            // i값 1 증가
}

printf("1부터 5까지의 합은 %d 입니다.\n", sum);
```

- ✓ 프로그램의 실행됨에 따라 제어 흐름과 변수가 어떻게 변하는 지 따져보자. (교재 p.148의 수행과정 표 참조)

2) while 문

- while문 수행 과정 확인하기

- while문의 끝에 **출력문**을 추가하여 수행과정을 눈으로 확인해보자.

실행 결과

```
while( i <= 5 ) {  
    sum += i;  
    i++;  
    printf("i: %d, sum: %d\n", i, sum);  
}
```

```
i: 2, sum: 1  
i: 3, sum: 3  
i: 4, sum: 6  
i: 5, sum: 10  
i: 6, sum: 15
```

- ✓ **디버깅 팁:** 반복문의 처음이나 마지막에 출력문을 넣는 것은 반복문이 의도한대로 동작하는 지 확인하는 기본적인 방법 중 하나

2) while 문

- 다음 각 문장에 해당하는 while 문을 작성하시오.
 - 변수 i가 10보다 작은 동안,
"Hello World" 를 출력하고 i를 1만큼 증가
 - 변수 i가 0보다 크고 10보다 작은 동안,
변수 x에 0을 대입하고 i를 1만큼 감소
 - 변수 num이 50보다 크기 전까지, (즉, 50보다 크면 반복 종료)
정수를 읽어 들여 num에 저장

31
50
4
51

2) while 문

- [프로그램 2] 반복횟수가 정해져 있지 않은 예시
 - 0 또는 음수가 입력되기 전까지 정수를 입력 받아, 입력 받은 정수의 합(종료 조건 0또는 음수는 합에서 제외) 구하기

```
int x, sum = 0;

printf("정수를 입력하시오: ");    // 입력 안내문
scanf("%d", &x);                  // 첫 정수 입력

while ( x > 0 ) {                 // x가 양수이면 반복
    sum += x;
    printf("정수를 입력하시오: ");
    scanf("%d", &x);
}

printf("입력된 정수의 합은 %d 입니다.\n", sum);
```

2) while 문

- [예제 6.1] 정수를 입력 받아, 그 정수에 해당하는 구구단 출력하기

입력	출력
5	5 * 1 = 5 5 * 2 = 10 ... 5 * 9 = 45

- [예제 6.2] 영어 소문자 순서대로 출력하기
 - ✓ hint) 영어 소문자의 아스키 코드 값이 연속한다는 성질 이용
 - ✓ 영문자의 아스키 코드 값을 알 필요는 없음 (2장 문자 자료형)

실행 결과

abcdefghijklmnopqrstuvwxyz

2) while 문

- [예제 6.3] 1부터 시작하여 값을 1씩 증가시키면서 차례로 더한 합이 최초로 100 이상이 되는 때, 마지막에 더한 값 구하기
 - 즉, $1+2+\dots+n \geq 100$ 인 가장 작은 n 구하기

실행 결과

14

- [예제 6.4] 0이 입력되기 전까지 정수를 입력 받아(즉, 종료 조건은 정수 0 입력), 이 중에서 가장 작은 값 구하기
 - 처음 입력 정수는 0이 아니라고 가정하고, 정수 0은 비교 대상에서 제외
 - hint) 반복문 안에서 조건문 사용

입력 예시 1

10 2 15 7 0

출력 예시 1

2

입력 예시 2

10 2 -15 7 0

출력 예시 2

-15

목차

- 1) 반복문 개요
- 2) while 문
- 3) **do-while 문**
- 4) for 문
- 5) 중첩 반복
- 6) 반복문 기타

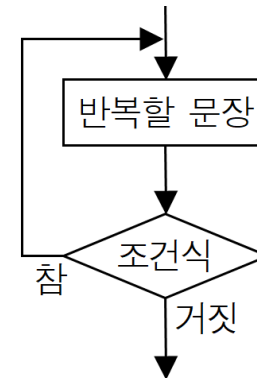
3) do-while 문

- do-while문

- 반복할 문장을 실행한 후에 조건식 검사
- 반복문 내에 있는 문장을 최소한 한 번 실행하고자 할 때 유용
- 주의) 마지막에 세미콜론(;)을 반드시 써야 함

- 구문

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



```
do {  
    sum += i;  
    i++;  
} while( i <= 5 );
```

```
while( i <= 5 ) {  
    sum += i;  
    i++;  
}
```

while 문과 비교

3) do-while 문

- [프로그램 3] 프로그램 1을 do-while 문으로 작성하기
 - do-while 문을 이용하여 1부터 5까지의 합 구하기

```
int i, sum;

sum = 0;          // 합
i = 1;            // 더하는 수

do {
    sum += i;      // i를 sum에 더하고
    i++;           // i값 1 증가
} while ( i <= 5 ) { // 반복 조건 검사

printf("1부터 5까지의 합은 %d 입니다.\n", sum);
```

3) do-while 문

- [프로그램 4] 프로그램 2를 do-while 문으로 작성하기
 - 0 또는 음수가 입력되기 전까지 정수를 입력받아, 입력 받은 정수의 합(종료 조건 0또는 음수는 합에서 제외) 구하기
 - 기본 예제 2의 코드와 무엇이 다른 지 비교해보자.

```
int x = 0, sum = 0;

do {
    sum += x;
    printf("정수를 입력하시오: ");
    scanf("%d", &x);
} while ( x > 0 );    // x가 양수이면 반복

printf("입력된 정수의 합은 %d 입니다.\n", sum);
```


3) do-while 문

- **[예제 6.5]** 정수를 입력 받아, do-while 문을 이용하여 그 정수에 해당하는 구구단 출력하기

입력

5

출력

5 * 1 = 5
5 * 2 = 10
...
5 * 9 = 45

-
- **[예제 6.6]** do-while 문을 이용하여 영어 소문자 순서대로 출력하기

실행 결과

abcdefghijklmnopqrstuvwxyz

3) do-while 문

- [예제 6.7] 1부터 시작하여 값을 1씩 증가시키면서 차례로 더한 합이 최초로 100 이상이 되는 때, 마지막에 더한 값 구하기
 - 즉, $1+2+\dots+n \geq 100$ 인 가장 작은 n 구하기

실행 결과

14

- [예제 6.8] 10 이상의 정수를 입력 받아, 각 자릿수의 합 출력하기
 - 예) $6235 \rightarrow 6+2+3+5 = 16$
 - hint) 나머지 연산자(%)와 나누기 연산자(/) 활용

입력 예시

6235

출력 예시

16

목차

- 1) 반복문 개요
- 2) while 문
- 3) do-while 문
- 4) **for 문**
- 5) 중첩 반복
- 6) 반복문 기타

4) for 문

- for 문 예제 훑어보기 (while 문과 비교하여)
 - 1부터 5까지의 합 계산

```
sum = 0;
for( i = 1 ; i <= 5 ; i++ ) {
    sum += i;
}
```

```
i = 1, sum = 0;
while( i <= 5 ) {
    sum += i;
    i++;
}
```

✓ for(i = 1 ; i <= 5 ; i++) 의미

① i의 값을 1부터 시작해서 (초기식)

② i가 5보다 작거나 같은 동안 (조건식)

③ i의 값을 1씩 증가시키면서 (증감식), 반복

✓ 반복과 관련된 수식을 하나로 모아 코드의 가독성을 높인 형태

4) for 문

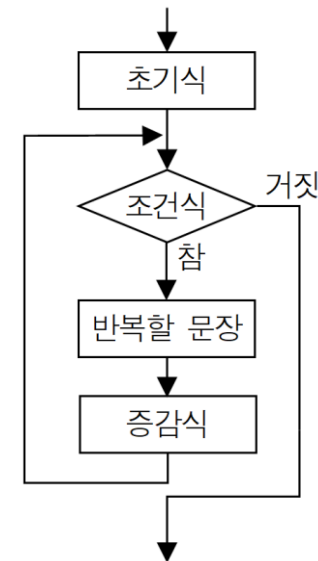
▪ for 문

- 일반적으로 반복하는 횟수가 정해진 경우에 사용
- 초기식, 조건식, 증감식으로 구성되고, **세미콜론**으로 구분
 - ✓ 초기식은 처음 한 번만 수행
 - ✓ 이후, 조건식 → 반복할 문장 → 증감식 이 반복적으로 수행됨

▪ 구문

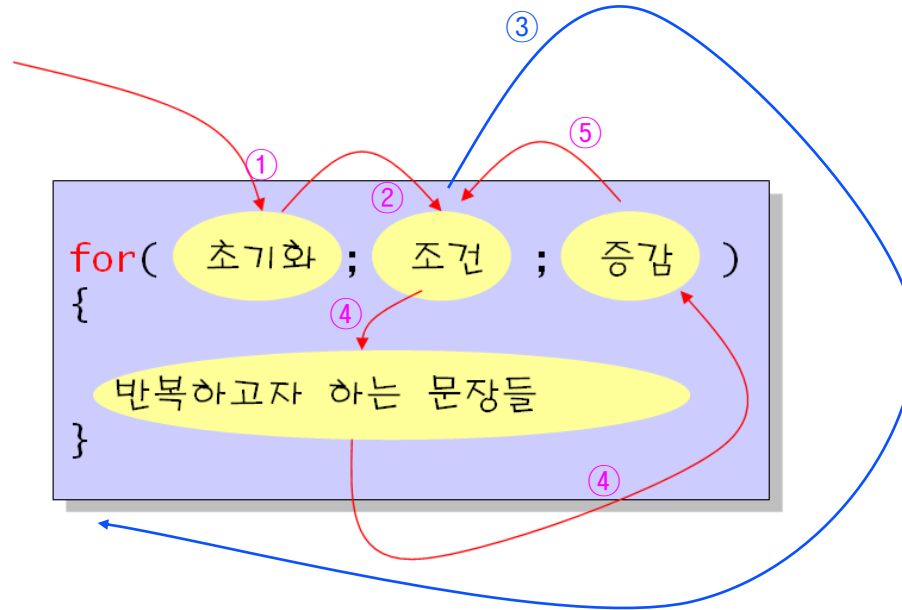
```
for( 초기식 ; 조건식 ; 증감식 ) {  
    반복할 문장;  
}
```

```
for( i = 1 ; i <= 5 ; i++ ) {  
    sum += i;  
}
```



4) for 문

- for 문의 상세 동작 방식



- ① 초기화를 실행한다.
- ② 반복 조건을 나타내는 조건식을 계산한다.
- ③ 조건식의 값이 거짓이면 for 문의 실행이 종료된다.
- ④ 조건식의 값이 참이면 문장이 실행된다.
- ⑤ 증감을 실행하고 ②로 돌아간다.

4) for 문

- while문과 for문의 형식 비교

```
초기식; ①  
while( 조건식 ) { ②  
    ③ 반복할 문장;  
    ④ 증감식;  
}
```

```
for( ① 초기식; ② 조건식; ④ 증감식 ) {  
    ③ 반복할 문장;  
}
```

- ✓ 위 비교는 for 문의 동작을 이해시키기 위함이다.
- ✓ for문을 여러 번 연습하면 자연스럽게 체득된다.

4) for 문

- [프로그램 5] 프로그램 1을 for 문으로 작성하기
 - for 문을 이용하여 1부터 5까지의 합 구하기

```
int i, sum;

sum = 0;          // 합

for( i = 1 ; i <= 5 ; i++ ) { // i를 1~5까지
                                // 1씩 증가시키면서
    sum += i;      // i를 sum에 더하기
}

printf("1부터 5까지의 합은 %d 입니다.\n", sum);
```

- 프로그램의 실행됨에 따라 제어 흐름과 변수가 어떻게 변하는 지 따져보자. (교재 p.157의 수행과정 표 참조)

4) for 문

- for문 수행 과정 확인하기

- for문의 끝에 **출력문**을 추가하여 수행과정을 눈으로 확인해보자.

실행 결과

```
for( i = 1 ; i <= 5 ; i++ ) {  
    sum += i;  
    printf("i: %d, sum: %d\n", i, sum);  
}
```

```
i: 1, sum: 1  
i: 2, sum: 3  
i: 3, sum: 6  
i: 4, sum: 10  
i: 5, sum: 15
```

- ✓ while 문의 수행과정 출력 결과(슬라이드 p.8) 와 동일한가?
- ✓ 다르다면 왜 다른 지 이해할 수 있는가?

4) for 문

- 다음 각 문장에 해당하는 for 문의 첫 줄을 작성해보자.
 1. 변수 i 를 2부터 10까지, 2씩 증가시키면서 반복
 2. 변수 i 를 10부터 시작하여, 1씩 감소시키면서 i 가 0보다 큰 동안 반복
 3. 변수 i 의 초기값은 0이고, i 값을 $i*i+2$ 로 바꾸면서, i 가 50보다 작은 동안 반복
 - ✓ 교재의 답에 오타 있음, 조건식에서 등호가 빠져야 함.

4) for 문

- for 문의 자주 활용되는 형태

- ✓ for(; i <= 5 ; i++) ⇨ 빈 초기식
- ✓ for(int i = 1 ; i <= 5 ; i++) ⇨ 변수 선언 및 초기화
단, i는 for 문 안에서만 사용 가능
- ✓ for(sum=0, i=1 ; i <= 5 ; i++) ⇨ 여러 개의 초기식 (콤마로 구분)
- ✓ for(i=0 ; i <= 5 ;) ⇨ 빈 증감식
- ✓ for(i=0, j=0 ; i <= 5 ; i++, j++) ⇨ 여러 개의 초기식과 증감식
- ✓ for(i=0 ; ; i++) ⇨ 빈 조건식 (무한 루프)
이 경우 조건식의 결과는 항상 참으로 간주

(문법적으로는 초기식, 조건식, 증감식에는 어떤 수식도 가능)

4) for 문

- [예제 6.9] 정수 n 을 입력 받아, $n! = 1*2*...*n$ 계산하기

입력

5

출력

120

- [예제 6.10] 문자와 정수를 입력 받아, 문자를 정수 개수 만큼 출력하기

입력

a 6

출력

□ a a a a a a

□는 공백을 의미

- [예제 6.11] 1부터 10까지의 홀수의 합, 짝수의 합 각각 출력하기
✓ hint) 모든 홀수를 구하려면,
1부터 시작하여 몇 씩 증가하면 될까?

실행 결과

25
30

4) for 문

- **[예제 6.12]** 0~6사이의 정수를 입력 받아, 그 정수에 해당하는 요일 (0이면 일요일, 1이면 월요일, ..., 6이면 토요일)에 1일이 시작하는 달력 출력하기

- 한 달은 31일로 가정
- 각 날짜는 3개의 칸에 출력
(3장 printf의 서식 플래그 활용)

입력

5

출력

	1	2	3				
□□4	5	6	7	8	9	10	
□11	12	13	14	15	16	17	
□18	19	20	21	22	23	24	
□25	26	27	28	29	30	31	

- ✓ hint 1) 1일 앞의 빈 칸은 따로 출력
- ✓ hint 2) 각 주의 줄 바꿈.
 - ✓ 매주 토요일에 해당하는 날짜 출력 후 개행 문자 출력
 - ✓ 토요일인지는 어떻게 판단? 몇 가지 예를 통해 규칙 찾기

목차

- 1) 반복문 개요
- 2) while 문
- 3) do-while 문
- 4) for 문
- 5) 중첩 반복
- 6) 반복문 기타

5) 중첩 반복

- 중첩 반복(다중 반복)

- 반복문에서 반복 대상은 어떤 문장(예: 조건문)이든 가능
- 반복문 안에 또다른 반복문이 오는 경우를 **중첩 반복**이라 함
 - ✓ 중첩 반복은 기존 문법의 단순한 조합

- 예시) 구구단 출력을 위한 반복문

- for문 안에 for문이 사용된 형태

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

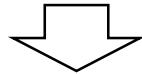
- ✓ 다른 종류의 반복문도 중첩 가능(예: for문 안에 while 문 사용)

5) 중첩 반복

- 구구단(하나의 단 출력)

- 하나의 단은 다음과 같이 반복문으로 표현 가능

```
printf("1 x 1 = %d\n", 1*1 );   ⇨ 1 x 1 = 1
printf("1 x 2 = %d\n", 1*2 );   ⇨ 1 x 2 = 2
...                               (생략)
printf("1 x 9 = %d\n", 1*9 );   ⇨ 1 x 9 = 9
```

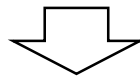


```
for( j=1 ; j < 10 ; ++j )
    printf("1 x %d = %d\n", j, 1*j );
    ⇨ 변수 j는 곱해지는 수
```


5) 중첩 반복

- 구구단(9개의 단 출력)
 - 하나의 단 출력을 9번 반복

```
for( j=1 ; j < 10 ; ++j )           ⇒ 1단
    printf("1 x %d = %d\n", j, 1*j );
for( j=1 ; j < 10 ; ++j )           ⇒ 2단
    printf("2 x %d = %d\n", j, 2*j );
...   (생략)
for( j=1 ; j < 10 ; ++j )           ⇒ 9단
    printf("9 x %d = %d\n", j, 9*j );
```



```
for( i=1 ; i < 10 ; ++i )           ⇒ i번째 단의
    for( j=1 ; j < 10 ; ++j ) ⇒ j번째 곱
        printf("%d x %d = %d\n", i, j, i*j );
```

5) 중첩 반복

- [프로그램 6] 구구단(완성 프로그램)

```
int i, j;

for( i=1 ; i < 10 ; ++i ) {      // i 단
    printf("== %d 단 ==\n", i);
    for( j=1 ; j < 10 ; ++j )    // j번째 곱
        printf("%d x %d = %d\n", i, j, i*j );
    printf("-----\n");
}
```

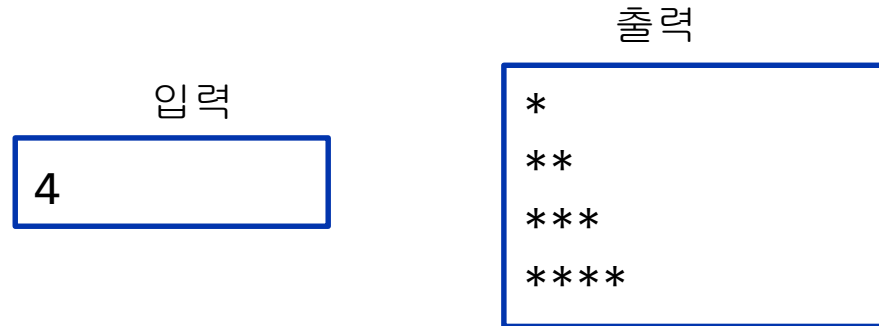
실행 결과

```
== 1 단 ==
1 x 1 = 1
...(생략)
1 x 9 = 9
-----
...
== 9 단 ==
9 x 1 = 9
...(생략)
9 x 9 = 81
-----
```

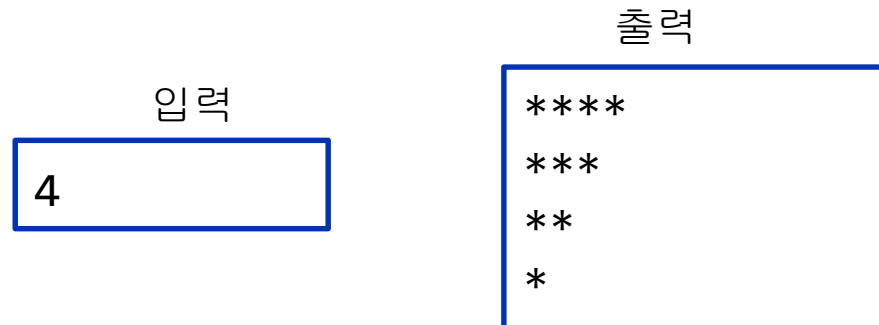
- 프로그램의 실행됨에 따라 제어 흐름과 변수가 어떻게 변하는 지 따져보자. (교재 p.163의 수행과정 표 참조)

5) 중첩 반복

- [예제 6.13] 양의 정수 N을 입력 받아, 높이가 N인 삼각형 출력하기



-
- [예제 6.14] 양의 정수 N을 입력 받아, 높이가 N인 역삼각형 출력하기



5) 중첩 반복

- **[예제 6.15]** 양의 정수 N을 입력 받아, 높이가 N이고 오른쪽으로 치우친 역삼각형 출력하기
 - ✓ 별 앞에 적절히 공백 출력

입력

4

출력

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

[예제 6.15a]

- [응용] 별 대신 수를 출력하도록 위 프로그램을 수정해보자
 - ✓ 각 수는 3칸을 차지하도록 출력

입력

5

실행 결과

```
□□1  2  3  4  5
      6  7  8  9
        10 11 12
          13 14
            15
```

5) 중첩 반복

- **[예제 6.16]** 2~50 사이의 소수를 구하여 각 줄에 5개씩 출력하기
 - ✓ 소수: 1과 자신 이외의 수로 나누어 떨어지지 않는 수
 - ✓ 소수를 검사하는 간단한 방법
 - ✓ n 을 2부터 $(n-1)$ 까지 나누어 보기

실행 결과

□	2	3	5	7	11
□	13	17	19	23	29
□	31	37	41	43	47

목차

- 1) 반복문 개요
- 2) while 문
- 3) do-while 문
- 4) for 문
- 5) 중첩 반복
- 6) 반복문 기타

6) 반복문 기타

- 반복문의 실행 상태를 직접 제어하고자 **break** 문과 **continue** 문을 사용한다.
- **break** 문
 - 현재 사용 중인 반복문을 중단하고 제어를 반복문 바깥으로 이동

```
for (cnt = 1; cnt < 10; cnt++)  
    if (cnt % 4 == 0 )
```

break;

cnt 값은 4 입니다

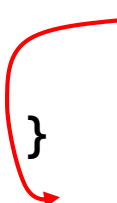
printf("cnt 값은 %d 입니다\n", cnt);

- ✓ cnt 값이 1 에서 부터 증가하다, 4가 됐을 때,
if 문이 참이 되고 break 문 수행(for문을 빠져나옴)

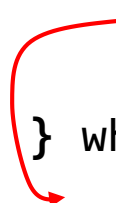
6) 반복문 기타

- break 문 동작 과정

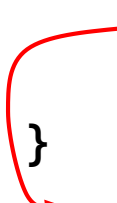
```
while( 조건식 ) {  
    ...  
    break;  
    ...  
}  
문장 1;
```



```
do {  
    ...  
    break;  
    ...  
} while( 조건식 );  
문장 1;
```



```
for( 초기식; 조건식 ; 증감식 ) {  
    ...  
    break;  
    ...  
}  
문장 1;
```



6) 반복문 기타

▪ continue 문

- 현재 수행 중인 반복문에서 현재 조건 값에 대한 처리를 중단하고, 다음 조건 값에 대한 처리를 수행
 - ✓ 결과적으로 continue 문과 반복문의 마지막 부분 사이에 있는 문장은 실행되지 않음

```
cnt = 0;
for ( i = 1; i < 10; i++) {
    if (i % 4 != 0 )
        continue;
    cnt++;
}
```

4의 배수는 2 개 입니다


```
printf("4의 배수는 %d 개 입니다\n", cnt);
```

- ✓ i 값이 4, 8 일 때 continue 문 수행(cnt++ 문장을 건너뛰)
- ✓ 4의 배수의 개수를 구하는 바람직한 코드는 아님


6) 반복문 기타

- continue 문 동작 과정


```
while( 조건식 ) {  
    ...  
    continue;  
    ...  
}  
문장 1;
```



```
do {  
    ...  
    continue;  
    ...  
} while( 조건식 );  
문장 1;
```



```
for( 초기식; 조건식 ; 증감식 ) {  
    ...  
    continue;  
    ...  
}  
문장 1;
```



6) 반복문 기타

- 중첩 반복에서 **break** 문과 **continue** 문
 - 그 문장을 둘러싸고 있는 가장 안쪽 반복문에 대해 적용됨

```
for( ... ) {
```

```
...
```

```
break; ⇒ for 문 종료, 이후 문장2 수행
```

```
while( ... ) {
```

```
...
```

```
break; ⇒ while 문 종료, 이후 문장1 수행
```

```
...
```

```
}
```

```
문장 1;
```

```
}
```

```
문장 2;
```

6) 반복문 기타

■ 무한반복

- 일반적으로 반복문에서는 조건을 지정하여 조건에 맞는 경우에만 반복을 시킴
- 경우에 따라서는 반복이 무한히 지속되는 **무한 반복**을 사용하는 경우도 있음
 - ✓ 조건식을 항상 참이 되도록 설정
 - ✓ 보통 다음 형태 사용

```
while( 1 ) {  
    반복할 문장;  
}
```

```
for( ; ; ) {  
    반복할 문장;  
}
```

6) 반복문 기타

- 무한 반복을 이용한 반복문 예

```
/* 일반적인 반복 형태 */  
while( i < 10 ) {  
    printf("%d\n", i);  
    i++;  
}
```

```
/* 무한 반복을 이용한 형태 */  
while ( 1 ) {  
    if( i >= 10 ) break;  
    printf("%d\n", i);  
    i++;  
}
```

- break문, continue문, 무한반복

- 반복문 중간에 제어를 마음대로 조정할 수 있어 편리
- But, 남용할 경우 **프로그램의 가독성에 악영향**
 - ✓ 위 두 코드 중 어느 것이 코드의 동작을 이해하기 편한가?
- 가능하면 break 문, continue 문, 무한 반복을 사용하지 않고 코딩하는 습관을 들이자..