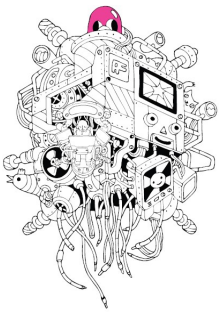


Chapter 07. 반복실행을 명령하는 반복문

반복문

- 그래서 반복문이라는 것을 만들었는데
  - "while 문" : while의 의미는 "...를 만족하는 한"
  - "while 문" : while 문은 if 문의 반복

```
int main(void) {

        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");
        printf("Hello, Guys!!₩n");

        return 0;
}
```

```
int main(void) {


        while (10번인지 확인)
        {
            printf("Hello, Guys!!₩n");
        }

        return 0;
}
```

2

반복문

- 반복문을 만드는 방법
  - (1) while 문, (2) do ~ while 문, (3) for 문

```
int main(void) {

    while (1)
    {
        printf("Hello, Guys!!₩n");
    }

    return 0;
}
```

항상 참

영원히 Hello!만 하고 있게 됨

```
int main(void) {

    int k = 1;

    while (k <= 10)
    {
        printf("Hello, Guys!!₩n");
        k++;   // k 값을 1 증가

    }

     return 0;
}
```

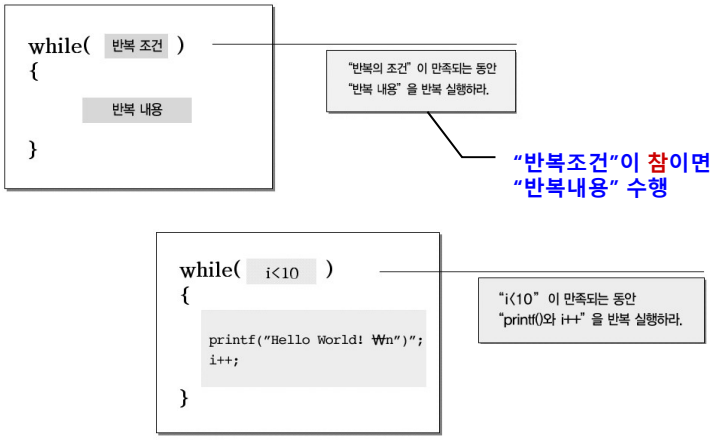
SimpleWhile.c

3

while문에 의한 문장의 반복

반복문: while 문

- **while문의 기본 원리와 의미**



5

반복문의 이해와 while문

- 반복문 : 하나 이상의 문장을 두 번 이상 반복 실행하기 위해서 구성하는 문장
- 반복문의 종류 : while, do~while, for

반복의 대상이 한 문장이면 중괄호 생략 가능

```
while(num<5)
    printf("Hello world! %d \n", num++);
```

```
while(num<5)
    printf("Hello world! %d \n", num), num++;
```

```
int main(void)
{
    int num=0;                          while 반복문
    while(num<5)
    {
        printf("Hello world! %d \n", num);
        num++;                          중괄호 내부 반복영역
    }
    return 0;
}
```

반복의 목적이 되는 대상

변수 num은 반복의 횟수를 조절하기 위한 것!

```
Hello world! 0
Hello world! 1
Hello world! 2
Hello world! 3
Hello world! 4
```

실행결과

6

while 문

- while 문의 중괄호
  - 반복하고자 하는 영역이 둘 이상의 문장으로 구성되는 경우에 필수

```
int i = 0;
while (i < 10)
{
        printf("Hello World! ₩n");
}
i++;
```

같은 의미

```
int i = 0;
while (i < 10)
        printf("Hello World! ₩n");
i++;
```

영원히 Hello!만 하고 있게 됨
WHY?

7

반복문 안에서도 들여쓰기 합니다.

들여쓰기를 하지 않은 것

```
int main(void)
{
int num=0;
while(num<5)
{
printf("Hello world! %d \n", num);
num++;
}
return 0;
}
```

들여쓰기를 한 것

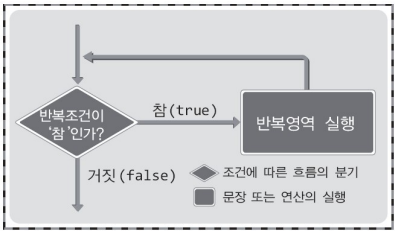
```
int main(void)
{
    int num=0;
    while(num<5)
    {
        printf("Hello world! %d \n", num);
        num++;
    }
    return 0;
}
```

들여쓰기를 한 것과 하지 않은 것의 차이가 쉽게 눈에 들어온다!

8

while문의 구성과 실행흐름의 세세한 관찰

```
int main(void)
{
    int num=0;
    while(num<3)   // 3회 반복
    {
        printf("Hello world! %d \n", num);
        num++;
    }
    . . . .
}
```



flow chart 기준에서의 while문

<반복문에서 꼭 명심할 내용>

- (1) 반복조건에 포함된 변수의 초기화
- (2) 반복조건은 논리연산과 관계연산 사용
- (3) 반복조건에 사용된 변수 값 갱신

9

구구단의 출력

```
int main(void)
{
    int dan=0, num=1;
    printf("몇 단? ");
    scanf("%d", &dan);

    while(num<10)
    {
        printf("%d×%d=%d \n", dan, num, dan*num);
        num++;
    }
    return 0;
}
```

```
몇 단? 7
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
```

NineNineDan.c

실행결과

구구단은 반복문을 이해하는데 사용되는 대표적인 예제이다.
이후에 반복문의 중첩에서는 구구단 전체를 출력하는 예제를 접한다.

10

무한루프의 구성

```
while( 1 )
{
    printf("%d×%d=%d \n", dan, num, dan*num);
    num++;
}
```

숫자 1은 '참'을 의미하므로 반복문의 조건은 계속해서 '참'이 된다.
이렇듯 반복문의 탈출조건이 성립하지 않는 경우 무한루프를 형성한다고 한다.

이러한 무한루프는 실수로 만들어지는 경우도 있지만, break문과 함께 유용하게 사용되기도 한다.

11

while문의 중첩

while문 안에 while문이 존재하는 상태를 의미한다. 아래의 예제에서는 while문을 중첩시켜서 구구단 전체를 출력한다. 이 예제를 통해서 중첩된 while문의 코드 흐름을 이해하자.

```
int main(void)
{
    int cur=2;
    int is=0;

    while(cur<10)   // 2단부터 9단까지 반복
    {
        is=1;   // 새로운 단의 시작을 위해서
        while(is<10)    // 각 단의 1부터 9의 곱을 표현
        {
            printf("%d×%d=%d \n", cur, is, cur*is);
            is++;
        }
        cur++;  // 다음 단으로 넘어가기 위한 증가
    }
    return 0;
}
```

바깥쪽 while문

안쪽 while문

TwoToNine.c

12

실습 문제 (Lab1)

- 한 줄에 별표 (*) 3개씩 5줄을 출력하는 프로그램을 작성 (while문을 이용)
  - 프로그램에서 printf는 한 번만 보이도록 함

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

13

실습 문제 (Lab2)

- 표준입력으로 받은 정수 수만큼의 행을 한 행에 ***를 출력하는 프로그램을 작성하시오. (while문을 이용)
  - keyboard에서 입력 받은 정수는 n에 저장한다.
  - i라는 변수에 0을 넣고 그 값이 n보다 작으면 ***를 출력한다.
  - i라는 변수에 1을 더하고 그 값이 n보다 작으면 ***를 출력한다.
  - i가 n보다 커지면 프로그램을 끝낸다.

```
양의 정수를 입력하세요 : 4
***
***
***
***
```

14

실습 문제 (Lab3)

- 표준입력으로 받은 정수를 출력하는 프로그램을 작성하시오. (while 문을 이용)
  - 입력된 정수 값이 0이면 해당 값을 출력한 후 종료하시오.

```
정수 값을 입력하세요 : 4
입력된 정수 값은 4입니다.
정수 값을 입력하세요 : -5
입력된 정수 값은 -5입니다.
정수 값을 입력하세요 : 0
입력된 정수 값은 0입니다.
종료합니다.
```

15

실습 문제 (Lab4)

- 키보드로부터 짝수를 입력 받으면 그 수를 출력한 후 다시 값을 입력 받고 홀수를 입력 받으면 프로그램을 종료하시오.
  - 짝수가 입력되면 계속 반복하고 홀수가 입력되면 종료
  - 아래와 같은 출력 예제를 참고한다.

```
정수 값을 입력하세요 : 4
짝수인 4(을)를 입력하셨습니다.
정수 값을 입력하세요 : 6
짝수인 6(을)를 입력하셨습니다.
정수 값을 입력하세요 : 3
수고하셨습니다.
```

16

1부터 10까지의 합

- 1에서 10까지의 합을 구하는 프로그램

```
sum = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;
```

```
sum = 0;
sum += 1;  // sum은 0 + 1이므로 1
sum += 2;  // sum은 1 + 2이므로 3
sum += 3;  // sum은 3 + 3이므로 6
sum += 4;  // sum은 6 + 4이므로 10
sum += 5;  // sum은 10 + 5이므로 15
sum += 6;  // sum은 15 + 6이므로 21
sum += 7;  // sum은 21 + 7이므로 28
sum += 8;  // sum은 28 + 8이므로 36
sum += 9;  // sum은 36 + 9이므로 45
sum += 10;  // sum은 45 + 10이므로 55
```

17

1부터 10까지의 합

- 1에서 10까지의 합을 구하는 프로그램

```
int sum = 0;
int k = 1;

while ( k <= 10 )
{
    sum = sum + k;  // sum += k;와 동일
    k++;  // k 값을 1 증가 (k = k + 1; 또는 k += 1;)
}
```

18

실습 문제 (Lab5)

- 1에서 n까지의 곱을 구하는 while 문을 작성
  - 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10...*n의 계산하는 프로그램 작성
  - n은 키보드로부터 입력 받도록 한다.
  - 아래와 같은 출력 예제를 참고한다.

```
n의 값을 입력하세요 : 3
1부터 3까지의 곱은 6입니다.
```

19

1부터 n까지의 곱

- 축약연산자를 사용해서 프로그램 축소
  - while 문 몸체에 곱할 축약 대입연산자를 이용하고 조건검사에 이용할 i 값을 하나 증가 (mult *= i;는 mult = mult*i;와 동일)

```
mult = 1;
i = 1;
while (i <= n)
{
      mult = mult * i;
      i++;
}
```

```
mult = 1;
i = 1;
while (i <= n)
{
      mult *= i;
      i++;
}
```

```
mult = 1;
i = 1;
while (i <= n)
{
      mult *= i++;
}
```

20

실습 문제 (Lab6)

- 키보드로부터 정수 값을 하나 입력 받아 그 값이 3의 배수이면 그 값을 출력하고, 아니면 종료하는 프로그램을 작성하시오. 단, 종료 전까지는 계속 반복하도록 하시오.
  - 즉, 3의 배수를 입력할 경우에는 프로그램을 종료하지 않고 다시 정수 값을 받도록 해야 함
  - 출력 형태는 다음과 같다.

```
정수 값을 입력하세요. > 3
입력한 값은 3의 배수입니다.
정수 값을 입력하세요. > 6
입력한 값은 3의 배수입니다.
정수 값을 입력하세요. > 5
수고하셨습니다.
```

21

do~while문에 의한 문장의 반복

반복문: do-while

- do~while문과 while문의 차이점
  - do~while문은 일단 무조건 한번은 실행하고 나서 조건 검사 시행

```
do
{
        반복 내용
} while( 반복의 조건 );
```

일단 루프를 **한번 실행** 후, **"반복의 조건"** 이 만족되는 동안 **"반복 내용"** 을 반복 실행하라.

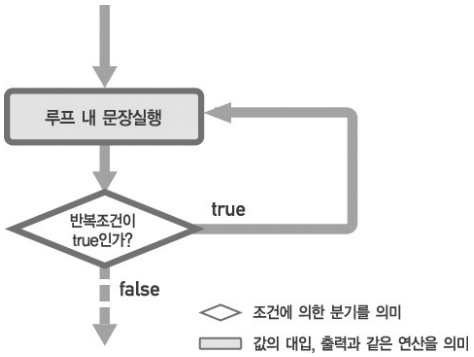
```
do
{
    printf("%d*%d=%d  \n", val, i, val*i);
    i++;
} while( i<10 );
```

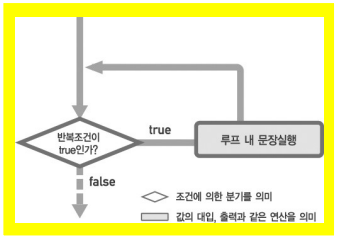
일단 한번 "printf()와 i++" 를 실행후 "i<10" 이 만족되는 동안 "printf()와 i++" 를 반복 실행하라.

23

do-while

- do~while문의 순서도





<참고 : while문>

24

do~while문의 기본구성

```
do
{
    printf("Hello world! \n");
    num++;
} while(num<3);
```

반복의 과정은?

반복조건을 반복문의 마지막에 진행하는 형태이기 때문에
최소한 1회는 반복영역을 실행하게 된다. 이것이 while문과의 가장 큰 차이점이다.

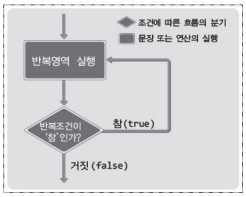
25

do~while문이 자연스러운 상황

```
while(num<10)
{
    printf("%d×%d=%d \n", dan, num, dan*num);
    num++;
}
```

동일한 횟수를 반복하는 반복문들

```
do
{
    printf("%d×%d=%d \n", dan, num, dan*num);
    num++;
} while(num<10);
```



do~while문의 순서도

```
int main(void)
{
    int total=0, num=0;
    do
    {
        printf("정수 입력(0 to quit): ");
        scanf("%d", &num);
        total += num;
    }while(num!=0);
    printf("합계: %d \n", total);
    return 0;
}
```

```
정수 입력(0 to quit): 1
정수 입력(0 to quit): 2
정수 입력(0 to quit): 3
정수 입력(0 to quit): 4
정수 입력(0 to quit): 5
정수 입력(0 to quit): 0
합계: 15
```

실행결과

UsefulDoWhile.c

최소한 1회 이상 실행되어야 하는
반복문은 do~while문으로 구성하는 것이
자연스럽다.

26

실습 문제 (Lab5)

- 1에서 n까지의 곱을 구하는 while 문을 작성
  - 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10...*n의 계산하는 프로그램 작성
  - n은 키보드로부터 입력 받도록 한다.
  - 아래와 같은 출력 예제를 참고한다.

```
n의 값을 입력하세요 : 3
1부터 3까지의 곱은 6입니다.
```

27

실습 문제 (Lab6)

- 키보드로부터 정수 값을 하나 입력 받아 그 값이 3의 배수이면 그 값을 출력하고, 아니면 종료하는 프로그램을 작성하시오. 단, 종료 전까지는 계속 반복하도록 하시오.
  - 즉, 3의 배수를 입력할 경우에는 프로그램을 종료하지 않고 다시 정수 값을 받도록 해야 함
  - 출력 형태는 다음과 같다.

```
정수 값을 입력하세요. > 3
입력한 값은 3의 배수입니다.
정수 값을 입력하세요. > 6
입력한 값은 3의 배수입니다.
정수 값을 입력하세요. > 5
수고하셨습니다.
```

28

실습 문제 (Lab7)

- 표준입력으로 받은 정수를 출력하는 과정을 반복하는 프로그램을 작성하시오. 다만 0을 입력하면 프로그램을 종료하도록 한다. (do-while문을 이용)
  - keyboard에서 입력 받은 정수는 value에 저장한다.
  - 입력 받은 정수를 출력한다.
  - 입력 받은 정수가 0이 아니면 다음 정수를 입력 받는다.
  - 입력 받은 정수가 0이면 입력을 그만 받고 종료한다.

```
정수를 입력하세요 : 10
입력한 수는 10입니다.
정수를 입력하세요 : 20
입력한 수는 20입니다.
정수를 입력하세요 : 2
입력한 수는 2입니다.
정수를 입력하세요 : 0
입력한 수는 0입니다.
```

29

실습 문제 (Lab8)

- 키보드로부터 숫자를 입력 받아 계속 더한 후 매번 그 결과를 출력하는 프로그램을 작성하시오. 단, 숫자 0을 입력하면 프로그램을 종료하도록 하시오.
  - while 문을 사용하는 경우
  - do ~ while 문을 사용하는 경우

```
숫자를 입력하세요 : 3
현재까지의 합계는 3입니다.
숫자를 입력하세요 : 5
현재까지의 합계는 8입니다.
숫자를 입력하세요 : 0
수고하셨습니다.
```

30

Homework: 10진수 자릿수 반대로 (1/2)

- 예를 들어, 정수 125에서 각각의 자릿수 1, 2, 5를 그 순서를 반대로 출력하는 방법 (5, 2, 1 출력)
  - 나머지 연산자 %를 이용하면 항상 1의 자릿수에 있는 숫자를 알 수 있음

PrintDigitReverse.c

실행순서	1		2	
정수	정수 % 10	결과	정수 / 10	결과
125	125 % 10	5	125 / 10	12
12	12 % 10	2	12 / 10	1
1	1 % 10	1	1 / 10	0
0				

```
int r_digit, value;

do {
        r_digit = value % 10;  // r_digit은 value이 가장 우측 숫자 1개
        printf("%d", r_digit);  // 가장 우측 숫자 출력
        value = value / 10;  // 그 다음 우측 숫자 구하기 위한 값 조정
} while ( value != 0 );  // value가 0이면 모든 자릿수를 구한 것이므로 종료
```

31

Homework: 10진수 자릿수 반대로 (2/2)

- 키보드로부터 세 자리 수 이상의 정수를 입력 받음
- 입력 받은 정수를 꺼꾸로 출력함
- do ~ while 반복문을 사용할 것
- 파일 제출형태: PrintDigitReverse_20191234.c
- 동작 예~~~

```
정수를 입력 (세 자리 수 이상): 12345
입력한 숫자 (12345)의 역순: 54321
```

32

실습 시간 (2019년 5월 8일)

- 예제 문제
  - while 예제: SimpleWhile.c, NineNineDan.c, TwoToNine.c
  - do while 예제: UsefulDoWhile.c
- 실습 문제
  - 실습문제(Lab5), 실습문제(Lab6),
  - 실습문제(Lab7), 실습문제(Lab8-1), 실습문제(Lab8-2),
- Homework (~ 5/12)
  - PrintDigitReverse.c 포함 미완성 실습문제
  - 제출처: yah2123@naver.com
  - 마감일자: 5월 12일 자정까지

33

for문에 의한 문장의 반복

반복문의 필수3요소

```
int main(void)
{
    int num=0;   // 필수요소 1. 반복을 위한 변수의 선언
    while(num<3)    // 필수요소 2. 반복의 조건검사
    {
        printf("Hi~");
        num++;    // 필수요소 3. 반복의 조건을 '거짓'으로 만들기 위한 연산
    }
    . . . .
}
```

정해진 횟수의 반복을 위해서는 하나의 변수가 필요하다.
그 변수를 기반으로 하는 조건검사가 필요하다.
조건검사가 false가 되게 하기 위한 연산이 필요하다.

} 이 세 가지를 한 줄에 표시하도록 돕는 것이 for문이다.

위의 while문에서 보이듯이 반복문에 필요한 세 가지 요소가 여러 행에 걸쳐서 분산되어 있다.
따라서 반복의 횟수가 바로 인식 불가능하다.

35

반복문: for 문

- for문의 기본 원리와 의미
  - 가장 많이 사용되는 반복문
  - 초기문, 조건문, 증감문 모두를 기본적으로 포함!
  - while문과 서로 교환 사용 가능

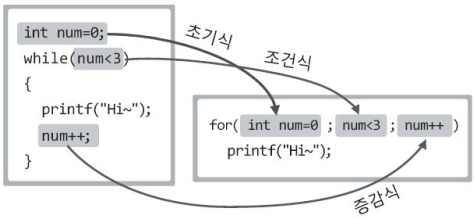
<반복문에서 꼭 명심할 내용>
- (1) 반복조건에 포함된 변수의 초기화
- (2) 반복조건은 논리연산과 관계연산 사용
- (3) 반복조건에 사용된 변수 값 갱신

```
int main (void)
{
   int i;
   i=0;                          → 초기문
   while( i<10 )                 → 조건문
   {
        printf("Hello World! ₩n");
        i++;                     → 증감문
   }
   . . . . .
```

```
for( 초기문 ; 조건문 ; 증감문 )
{
        반복하고자 하는 내용
}
```

36

for문의 구조와 이해




```
for( 초기식 ; 조건식 ; 증감식 )
{
    // 반복의 대상이 되는 문장들
}
```

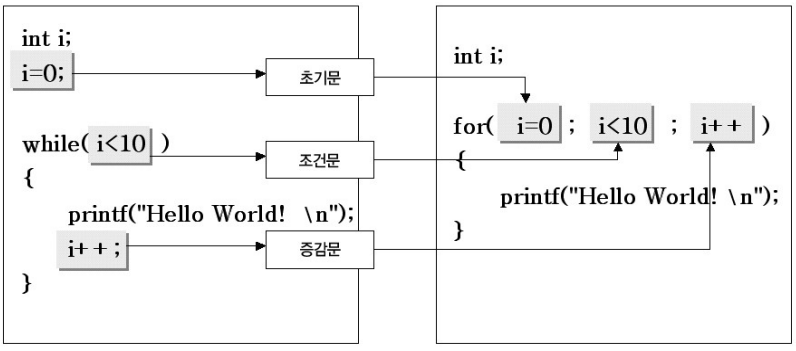
```
int main(void)
{
    int num;
    for(num=0; num<3; num++)
        printf("Hi~");
    . . . .
}
```

일부 컴파일러는 여전히 초기식에서의 변수 선언을 허용하지 않음
for문의 반복영역도 한 줄이면 중괄호 생략 가능!

37

반복문 for 문

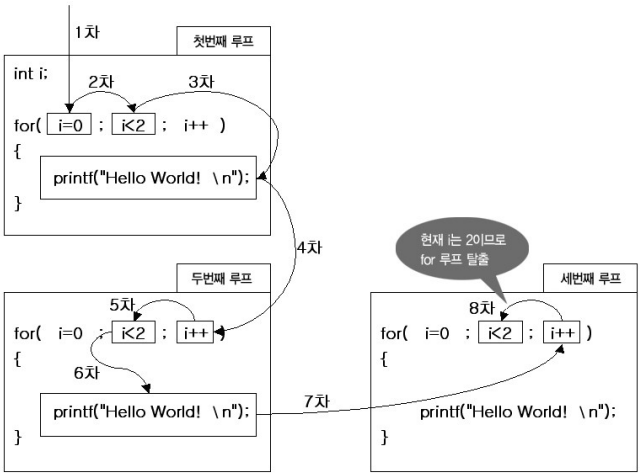
- for문과 while문의 비교



38

반복문 for 문

- 반복 과정의 이해



39

for문의 흐름 이해

for문의 구성요소

√ 초기식 본격적으로 반복을 시작하기에 앞서 딱 한번 실행된다.
√ 조건식 매 반복의 시작에 앞서 실행되며, 그 결과를 기반으로 반복유무를 결정!
√ 증감식 매 반복실행 후 마지막에 연산이 이뤄진다.

for문 흐름의 핵심

int num=0에 해당하는 초기화는 반복문의 시작에 앞서 딱 1회 진행!
num<3에 해당하는 조건의 검사는 매 반복문의 시작에 앞서 진행!
num++에 해당하는 증감연산은 반복영역을 실행한 후에 진행!

- 첫 번째 반복의 흐름
  1→2→3→4 [num=1]
- 두 번째 반복의 흐름
  2→3→4 [num=2]
- 세 번째 반복의 흐름
  2→3→4 [num=3]
- 네 번째 반복의 흐름
  2 [num=3] 따라서 탈출!

```
   1          2        4
for( int num=0 ; num<3 ; num++ )
{ 3
   printf("Hi~");
}
```

40

문자 *를 한 줄에 5개씩 5줄 출력

- 정수형 변수 i의 초기값을 0으로 지정, 5보다 작을 때까지, 증감연산 부분에 증가자(i++)를 이용하여 출력문을 출력

```
printf("*****\n");
printf("*****\n");
printf("*****\n");
printf("*****\n");
printf("*****\n");
```

```
for (i = 0; i < 5; i++)
{
    printf("*****\n");
}
```

초기화	i = 0			
실행순서	2 →		3 →	4
변수 i 값	검사 (i<5)	검사 결과	몸체실행	증감연산 (i++)
0	0 < 5	1 (참)	printf("*****\n");	0+1
1	1 < 5	1 (참)	printf("*****\n");	1+1
2	2 < 5	1 (참)	printf("*****\n");	2+1
3	3 < 5	1 (참)	printf("*****\n");	3+1
4	4 < 5	1 (참)	printf("*****\n");	4+1
5	5 < 5	0 (거짓)		

41

증감 부분의 이해

- for 문의 증간연산 부분
  - 아래의 두 프로그램은 같은 기능을 수행할까?

```
for (i = 0; i < 5; i++)
{
    printf("*****₩n");
}
```

```
for (i = 0; i < 5; i = i +1)
{
    printf("*****₩n");
}
```

- 그렇다면 아래 프로그램은 어떠한가?

```
for (i = 0; i < 5; ++i)
{
    printf("*****₩n");
}
```

42

for문 기반의 다양한 예제

```
int main(void)
{
    int total=0;
    int i, num;
    printf("0부터 num까지의 덧셈, num은? ");
    scanf("%d", &num);

    for(i=0; i<num+1; i++)
        total+=i;

    printf("0부터 %d까지 덧셈결과: %d \n", num, total);
    return 0;
}
```

```
0부터 num까지의 덧셈, num은? 10
0부터 10까지 덧셈결과: 55
```

AddToNum.c

실행결과

다양한 예제를 통해서 for문에 익숙해지자!

오른쪽 예제에서 보이듯이 불필요하다면, 초기식, 조건식, 증감식을 생략할 수 있다.

단 조건식을 생략하면 참으로 인식이 되어 무한루프를 형성하게 된다.

```
int main(void)
{
    double total=0.0;
    double input=0.0;
    int num=0;

    for( ; input>=0.0 ; )
    {
        total+=input;
        printf("실수 입력(minus to quit) : ");
        scanf("%lf", &input);
        num++;
    }
    printf("평균: %f \n", total/(num-1));
    return 0;
}
```

RealMean.c

실행결과

```
실수 입력(minus to quit) : 3.2323
실수 입력(minus to quit) : 5.1891
실수 입력(minus to quit) : 2.9297
실수 입력(minus to quit) : -1.0
평균: 3.783700
```

43

for문의 중첩

```
int main(void)
{
    int cur, is;

    for(cur=2; cur<10; cur++)
    {
        for(is=1; is<10; is++)
            printf("%d×%d=%d \n", cur, is, cur*is);
        printf("\n");
    }
    return 0;
}
```

TwoToNineForVer.c

for문의 중첩은 while, do~while문의 중첩과 다르지 않다.

구구단 전체를 출력하는 왼편의 예제를 통해서 for문의 중첩을 이해하자.

44

반복문의 생략과 탈출: continue & break

break! 이제 그만 빠져나가자!

```
int main(void)
{
    int sum=0, num=0;

    while(1)
    {
        sum+=num;
        if(sum>5000)
            break; // break문 실행! 따라서 반복문 탈출
        num++;
    }

    printf("sum: %d \n", sum);
    printf("num: %d \n", num);
    return 0;
}
```

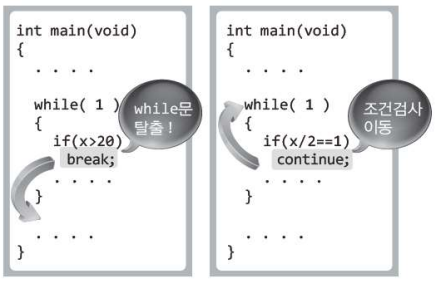
- **break문은 자신을 감싸는 반복문 하나를 빠져 나간다.**
- **if문과 함께 사용이 되어서 특정 조건만이 만족될 때 반복문을 빠져나가는 용도로 주로 사용된다.**

실행결과

```
sum: 5050
num: 100
```

46

continue! 나머지 생략하고 반복조건 확인하러



- continue문은 반복문을 빠져나가지 않는다!
- 다만 반복조건을 확인하러 올라갈 뿐이다.
- 그리고 반복조건이 여전히 '참'이라면 반복영역을 처음부터 실행하게 된다.

```
int main(void)
{
    int num;
    printf("start! ");

    for(num=1; num<20; num++)
    {
        if(num%2==0 || num%3==0)
            continue;
        printf("%d ", num);
    }
    printf("end! \n");
    return 0;
}
```

```
start! 1 5 7 11 13 17 19 end!
```

실행결과

47

실습 문제 (Lab9)

- 표준입력으로 받은 정수 수만큼의 행을 한 행에 ***를 출력하는 프로그램을 작성하시오. (for 문을 이용)
  - 예를 들어 입력이 3이라면 다음을 출력한다.

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

```
양의 정수를 입력하세요 : 4
***
***
***
***
```

  - keyboard에서 입력 받은 정수는 n에 저장한다.
  - i라는 변수에 0을 넣고 그 값이 n보다 작으면 ***를 출력한다.
  - i라는 변수에 1을 더하고 그 값이 n보다 작으면 ***를 출력한다.
  - i가 n보다 커지면 프로그램을 끝낸다.

48

for 문이 반복되는 과정 (1/2)

- 1에서 10까지의 합을 구하는 for 문을 작성

```
sum = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;
```

- 이를 for 문으로 바꿔보자.
  - 처음에 합이 저장될 변수 sum에 0을 대입하고, 반복 횟수에 대한 제한에 이용될 i를 1로 초기화
  - 조건검사는 10번을 반복해야 하므로 i<=10이고 증감연산은 i 값을 하나씩 증가하도록 i++로 구성

```
sum = 0;

for (i = 1; i <= 10; i++)
{
   sum = sum + i;
}
```

```
sum += i;
```

49

for 문이 반복되는 과정 (2/2)

- 1에서 10까지의 합을 구하는 for 문 동작

실행순서	1		2	3
변수 i 값	검사 (i <= 10)	검사 결과	몸체실행 (sum+=i) (sum = sum + i)	증감연산 (i++) (i = i+1)
1(초기 값)	1 <= 10	참	sum = 0+1	1+1
2	2 <= 10	참	sum = 0+1+2	2+1
3	3 <= 10	참	sum = 0+1+2+3	3+1
4	4 <= 10	참	sum = 0+ … +4	4+1
5	5 <= 10	참	sum = 0+ … +5	5+1
6	6 <= 10	참	sum = 0+ … +6	6+1
7	7 <= 10	참	sum = 0+ … +7	7+1
8	8 <= 10	참	sum = 0+ … +8	8+1
9	9 <= 10	참	sum = 0+ … +9	9+1
10	10 <= 10	참	sum = 0+ … +10	10+1
11	11 <= 10	거짓		

50

1부터 10까지의 합 (1/2)

- 1에서 10까지의 합을 구하는 프로그램 (while 문 이용)

```
int sum = 0;
int k = 1;

while ( k <= 10 )
{
    sum = sum + k;  // k는 1부터 시작해서 1씩 증가
    k++;  // k 값을 1 증가 (k = k + 1;)
}
```

51

1부터 10까지의 합 (2/2)

- 1에서 10까지의 합을 구하는 프로그램 (for 문 이용)

```
int sum = 0;
int k = 1;

for (k=1;k<=10;k++)
{
    sum = sum + k;  // k는 1부터 시작해서 1씩 증가
}
```

52

무한 반복

- for 문에서 조건검사 부분이 없다면? (NO ERROR!!)
  - 조건검사를 무시하고 계속적으로 반복문을 수행
    - 반복문이 종료되지 않고 무한반복 (infinite loop)을 실행
- while(조건검사) 문에서의 조건검사 부분이 없다면? (ERROR!!)
  - while 구문에서 조건검사 부분은 생략할 수 없음
  - while 구문으로 무한반복문을 만드는 방법
    - 조건검사 부분을 참 (0이 아닌 값)으로 지정

조건검사부

```
for ( ;; )
{
    printf("*****₩n");
}
```

조건검사부

```
while (*)
{
    printf("*****₩n");
}
```

```
while ( 1 )
{
    printf("*****₩n");
}
```

53

for(;;) 실습 문제 (Lab10)

- 키보드로부터 숫자를 입력 받아 계속 더한 후 매번 그 결과를 출력하는 프로그램을 작성하시오. 단, 숫자 0을 입력하면 프로그램을 종료하도록 하시오.
  - for (;;) {... }을 사용
  - for문 내에서 숫자를 입력 받음
  - (1) for문만 사용할 것 : Lab10-1
  - (2) for문과 if() break 문을 사용할 것 : Lab10-2

```
숫자를 입력하세요 : 3
현재까지의 합계는 3입니다.
숫자를 입력하세요 : 5
현재까지의 합계는 8입니다.
숫자를 입력하세요 : 0
현재까지의 합계는 8입니다.
수고하셨습니다.
```

(1)

```
숫자를 입력하세요 : 3
현재까지의 합계는 3입니다.
숫자를 입력하세요 : 5
현재까지의 합계는 8입니다.
숫자를 입력하세요 : 0
수고하셨습니다.
```

(2)

54

반복문 내부의 반복문 문제

- 0부터 9까지 정수 값을 출력하는데 우측 그림과 같이 출력하시오.
  - for문을 i = 0부터 i = 9까지 돌면서
  - for문 내부에 또 for문을 두어
  - 이번 내부 for문에서는
  - 0부터 i까지 돌면서 그 값을 출력하고
  - 다음 줄로 넘어가도록 함

```
0
01
012
0123
01234
012345
0123456
01234567
012345678
0123456789
Press any key to continue
```


반복문 내부의 반복문 문제 : 풀이

- 0부터 9까지 정수 값을 출력하는데 우측 그림과 같이 출력하시오.
  - for문을 i = 0부터 i = 9까지 돌면서
  - for문 내부에 또 for문을 두어
  - 이번 내부 for문에서는
  - 0부터 i까지 돌면서 그 값을 출력하고
  - 다음 줄로 넘어가도록 함

```
0
01
012
0123
01234
012345
0123456
01234567
012345678
0123456789
Press any key to continue
```

```
#include <stdio.h>

int main(void) {

    int i, k;

    for(i=0;i<10;i++) {
        for (k=0;k<= i;k++) {
            printf("%d", k);
        }
        printf("₩n");
    }

    return 0;
}
```

반복문 전체 정리

- 반복문 종류 및 특징 정리

반복문의 종류	구문	특징

for	for (초기화; 조건검사; 증감연산)
	{
	for문 몸체(body);
	}
		일정한 반복 횟수를 이용하는 반복문에 적합

while	while (조건검사)
	{
	while문 몸체(body);
	}
		구문이 간단하며, 검사부분이 처음에 있음

do while	do { do while문 몸체(body); } while (조건검사) ;	반복 몸체를 1번은 실행하며, 검사부분이 뒤에 있음

do~while문에는 반드시 필요
for문, while문에는 없음

57

콤마 연산자의 이용

- 초기화와 증감부분에 여러 개의 문장 기술

```
int sum = 0, i;
for (i = 1; i <= 10; i++)
        sum += i;
```

- 1부터 10까지의 합을 구하는 프로그램

```
int sum, i;
for (sum = 0, i = 1; i <= 10; sum += i, i++)
```

58

for(;;) 실습 문제 (Lab10)

- 키보드로부터 숫자를 입력 받아 계속 더한 후 매번 그 결과를 출력하는 프로그램을 작성하시오. 단, 숫자 0을 입력하면 프로그램을 종료하도록 하시오.
  - for (;;) {... }을 사용
  - for문 내에서 숫자를 입력 받음
  - (1) for문만 사용할 것 : Lab10-1
  - (2) for문과 if() break 문을 사용할 것 : Lab10-2

```
숫자를 입력하세요 : 3
현재까지의 합계는 3입니다.
숫자를 입력하세요 : 5
현재까지의 합계는 8입니다.
숫자를 입력하세요 : 0
현재까지의 합계는 8입니다.
수고하셨습니다.
```

(1)

```
숫자를 입력하세요 : 3
현재까지의 합계는 3입니다.
숫자를 입력하세요 : 5
현재까지의 합계는 8입니다.
숫자를 입력하세요 : 0
수고하셨습니다.
```

(2)

59

실습 문제 (Lab11)

- 10부터 50까지의 정수 중에서 2의 배수이면서 동시에 5의 배수인 숫자를 출력하는 프로그램을 작성하시오.
  - 아래와 같이 출력되도록 하시오.
  - (1) 다중 for() 문을 사용: Lab 11-1
  - (2) for() 문과 if() 문을 사용: Lab 11-2

```
10부터 50까지의 정수 중에서
2의 배수이면서 동시에 5의 배수인 숫자는
10 20 30 40 50입니다.
```

60

실습 문제 (Lab12)

- 아래와 같은 모양을 갖는 구구단 (1단 ~ 9단) 프로그램을 작성하시오.
  - 아래와 같이 출력되도록 하시오.
  - (1) 다중 for() 문을 사용

```
1 X 1 = 1        2 X 1 = 2        3 X 1 = 3
1 X 2 = 2        2 X 2 = 4        3 X 2 = 6
   …                …                …
1 X 9 = 9        2 X 9 = 18       3 X 9 = 27

4 X 1 = 4        5 X 1 = 5        6 X 1 = 6
4 X 2 = 8        5 X 2 = 10       6 X 2 = 12
   …                …                …
```

61

실습 시간 (2019년 5월 15일)

- 예제 문제
  - for 예제: AddToNum.c, RealMean.c, TwoToNineForVer.c
- 실습 문제
  - 실습문제(Lab10-1), 실습문제(Lab10-2),
  - 실습문제(Lab11-1), 실습문제(Lab11-2),
  - 실습문제(Lab12)

62

Chapter 07이 끝났습니다.