5장 제어의 흐름

김명호

내용

- 관계, 등가, 논리 연산자
- if
- 조건부 연산자
- while
- for
- 콤마 연산자
- do-while
- break, continue, goto
- switch

제어의 흐름

- 제어의 순차적 흐름 - 프로그램의 문장은 일반적으로 순차적으로 실행됨
- 동작의 선택, 반복을 위해 순차적 흐름을 변경할 필요가 있음
  - 선택 : if, if-else, switch
  - 반복 : while, for, do
  - 이러한 구문에 관계, 등가, 논리 연산자가 사용됨

관계, 등가, 논리 연산자

- 관계 연산자
  - ~보다 작다(<) : <
  - ~보다 크다(>) : >
  - ~보다 작거나 같다(≤) : <=
  - ~보다 크거나 같다(≥) : >=

관계, 등가, 논리 연산자

- 등가 연산자
  - 같다(=) : ==
  - 같지 않다(≠) : !=
- 논리 연산자
  - (단항)부정(¬) : !
  - 논리곱(∧) : &&
  - 논리합(∨) : ||

참과 거짓

- C에서 거짓은 0 값으로 나타내고, 참은 0 아닌 다른 값으로 나타냄
- 거짓의 값은 임의의 0 값이 될 수 있음
  - 0, 0.0, 널 문자 '\0', NULL 포인터 값
- 참 값은 임의의 0이 아닌 값
- a < b와 같은 수식은 참이나 거짓의 값을 가짐
  - 이 수식은 참이면 정수 값 1을, 거짓이면 정수 값 0을 생성

관계 연산자

- 관계 수식의 값

a - b	a < b	a > b	a <= b	a >= b
양수	0	1	0	1
영	0	0	1	1
음수	1	0	1	0

등가 연산자

- 등가 수식의 값

a - b	a == b	a != b
0	1	0
양수나 음수	0	1

- a != b와 !(a == b)는 동일한 수식

등가 연산자

- a == b와 a = b는 유사하지만, 완전히 다른 수식임

```
if (a = 1)
    . . . . . .
if (a == 1)
    . . . . . .
```

논리 연산자

- 논리 연산자
  - ! : 단항 연산자
  - &&, || : 이항 연산자
- ! 연산자

a	!a
0	1
양수나 음수	0

&&, || 연산자

- 수식의 값

a	b	a && b	a || b
0	0	0	0
0	양수나 음수	0	1
양수나 음수	0	0	1
양수나 음수	양수나 음수	1	1

단축 평가

- 결과가 참인지 거짓인지 판명되면 더 이상 다음 수식을 평가하지 않음
- expr1 && expr2
  - expr1이 거짓일 때,
    이 수식은 이미 거짓이라는 것이 판명되고
    따라서 expr2는 평가되지 않음
- expr1 || expr2
  - expr1이 참일 때,
    이 수식은 이미 참이라는 것이 판명되고
    따라서 expr2는 평가되지 않음

예제 프로그램

프로그램 5.1

```
#include <stdio.h>
int main(void){
    int i = 4, j = 5;
    (i == 4) || (j = 10);
    printf("|| 단축 평가 : j = %d\n", j);
    (i == 3) && (j = 6);
    printf("&& 단축 평가 : j = %d\n", j);
    return 0;
}
```

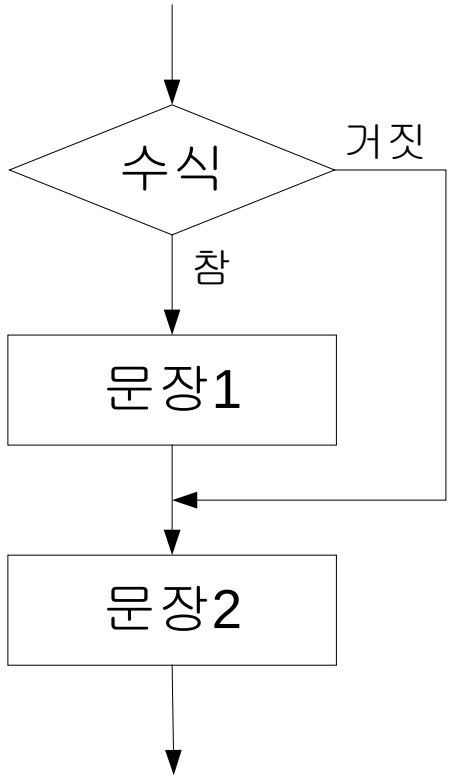
프로그램 결과

```
|| 단축 평가 : j = 5
&& 단축 평가 : j = 5
```

if

- **if 문의 형식**

```
if (수식)
    문장1
문장2
```



if

- **예제**

```
if (grade < 60)
    printf("불합격입니다.\n");
printf("%d점 입니다.\n", grade);
```

if

- **예제**

```
if (grade < 60)
    printf("불합격입니다.\n");
    printf("%d점 입니다.\n", grade);

if (grade < 60)
printf("불합격입니다.\n");
printf("%d점 입니다.\n", grade);
```

복합문

- 하나의 조건에 여러 문장을 실행해야 할 때

```
if (grade < 60)
    printf("불합격입니다.\n");
if (grade < 60)
    count++;
printf("%d점 입니다.\n", grade);
```

  - 이러한 식의 코딩은 비효율적임

복합문

- 중괄호 {}로 묶여진 선언문과 문장
- 문장들을 실행 가능한 하나의 단위로 그룹화함
- 블록이라고도 함
- 복합문 자체도 하나의 문장임
  - C에서 문법적으로 한 문장이 들어가는 자리에 복합문을 사용할 수 있음
- 선택문과 반복문에서 보통 사용됨

복합문

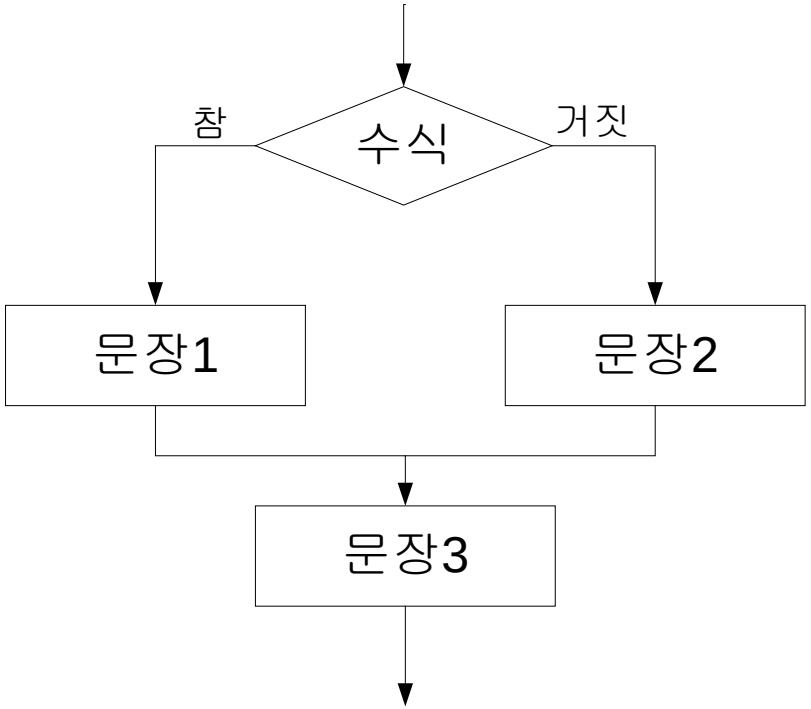
- **복합문을 사용한 코드**

```
if (grade < 60)
{
    printf("불합격입니다.\n");
    count++;
}
printf("%d점 입니다.\n", grade);
```

if-else 문

- **if-else 문의 형식**

```
if (수식)
  문장1
else
  문장2
문장3
```



if-else 문

- **예제**

```
if (grade < 60)
    printf("불합격입니다.\n");
else
    printf("합격입니다.\n");
printf("%d점 입니다.\n", grade);
```

예제 프로그램

프로그램 5.2

```
#include <stdio.h>
int main(void){
    int n, m;
    printf("n/m 프로그램\n");
    printf("n과 m을 입력하세요 : ");
    scanf("%d%d", &n, &m);
    if (m == 0)
        printf("m이 0입니다. 나누기를 할 수 없습니다.\n");
    else
        printf("%d/%d : 몫은 %d이고 나머지는 %d입니다.\n",
                n, m, n / m, n % m);
    return 0;
}
```

프로그램 결과

```
$ prog_4_2
n/m 프로그램
n과 m을 입력하세요 : 11 0
m이 0입니다. 나누기를 할 수 없습니다.
$ prog_4_2
n/m 프로그램
n과 m을 입력하세요 : 11 2
11/2 : 몫은 5이고 나머지는 1입니다.
```

중첩 if, if-else

- **예제 1**

```
if (0 <= grade && grade <= 100)
    if (grade < 60)
        printf("불합격입니다.\n");
```

- **예제 2**

```
if (0 <= grade && grade <= 100)
    if (grade < 60)
        printf("불합격입니다.\n");
    else
        printf("합격입니다.\n");
```

중첩 if, if-else

- **예제 3**

```
if (0 <= grade && grade <= 100)
   if (grade < 60)
      printf("불합격입니다.\n");
else
   printf("잘못된 점수입니다.\n");
```

- **예제 4**

```
if (0 <= grade && grade <= 100){
   if (grade < 60)
      printf("불합격입니다.\n");
}
else
   printf("잘못된 점수입니다.\n");
```

예제 프로그램

프로그램 5.3

```
#include <stdio.h>
int main(void){
    unsigned year;
    printf("년도를 입력 하세요 : ");
    scanf("%u", &year);
    if (year % 4)
        printf("%4u년은 평년입니다.\n", year);
    else
        if (year % 100)
            printf("%4u년은 윤년입니다.\n", year);
        else
            if (year % 400)
                printf("%4u년은 평년입니다.\n", year);
            else
                printf("%4u년은 윤년입니다.\n", year);
    return 0;
}
```

예제 프로그램

프로그램 5.3

```
#include <stdio.h>
int main(void){
    unsigned year;
    printf("년도를 입력 하세요 : ");
    scanf("%u", &year);
    if (year % 4)
        printf("%4u년은 평년입니다.\n", year);
    else if (year % 100)
        printf("%4u년은 윤년입니다.\n", year);
    else if (year % 400)
        printf("%4u년은 평년입니다.\n", year);
    else
        printf("%4u년은 윤년입니다.\n", year);
    return 0;
}
```

프로그램 결과

```
$ isleapyear
년도를 입력 하세요 : 2003
2003년은 평년입니다.
$ isleapyear
년도를 입력 하세요 : 2004
2004년은 윤년입니다.
$ isleapyear
년도를 입력 하세요 : 2000
2000년은 윤년입니다.
$ isleapyear
년도를 입력 하세요 : 1900
1900년은 평년입니다.
```

? : 조건부 연산자

- 삼항 연산자임
- 일반적인 형태

  수식1 ? 수식2 : 수식3
- 평가 방법

  1. 수식1 평가
  2. 참이면, 수식2를 평가하고 그 결과가 이 수식의 값이 됨
  3. 거짓이면, 수식3을 평가하고 그 결과가 이 수식의 값이 됨

? : 조건부 연산자

- 예

```
x = (y < z) ? y : z;
```

```
if (y < z)
    x = y;
else
    x = z;
```

? : 조건부 연산자

- 조건부 수식의 값과 형
  - 형 : 수식2와 수식3 중 큰 형
    - 일반적 산술 변환 규칙이 적용됨
  - 값 : 수식2와 수식3 중 평가되는 수식

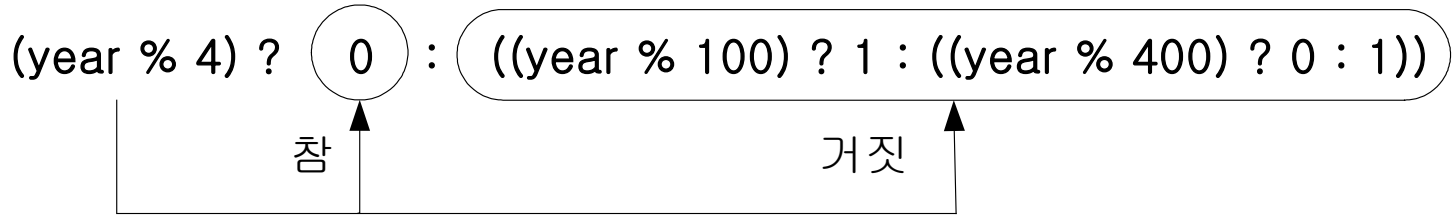
예제 프로그램

프로그램 5.4

```
#include <stdio.h>
int main(void){
    unsigned year;
    _Bool isleap; // 0: 평년, 1 : 윤년
    printf("년도를 입력 하세요 : ");
    scanf("%u", &year);
    isleap = (year % 4) ? 0 :
             ((year % 100) ? 1 : ((year % 400) ? 0 : 1));
    if (isleap)
       printf("%4u년은 윤년입니다.\n", year);
    else
       printf("%4u년은 평년입니다.\n", year);
    return 0;
}
```

? : 조건부 연산자

- 중복된 조건부 연산자



- 중복된 조건부 수식은 문장을 단수화 시킬 수는 있지만 가독성이 떨어짐

반복문

프로그램 5.5

```
#include <stdio.h>
int main(void){
    unsigned f;
    f = 5 * 4 * 3 * 2 * 1;      // f = 5!
    printf("5! = %u\n", f);
    return 0;
}
```

- 100! ?

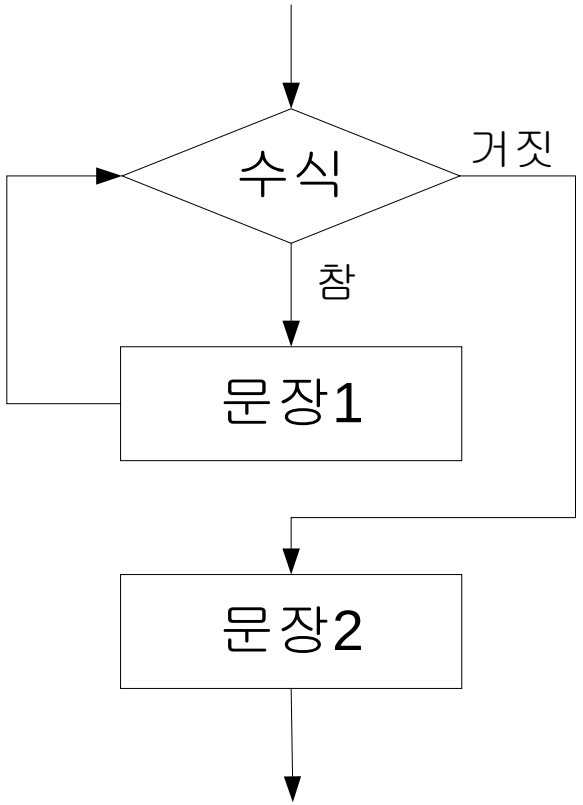
반복문

- 작은 n에 대해서는 반복문 없이 n! 프로그램을 작성할 수 있지만, 큰 n에 대해서는 반복문을 사용해야 함
- 반복문
  - 코드를 반복 실행할 수 있게 함
  - while, for, do-while

while

- **일반적인 형태**

```
while (수식)
    문장1
문장2
```



while

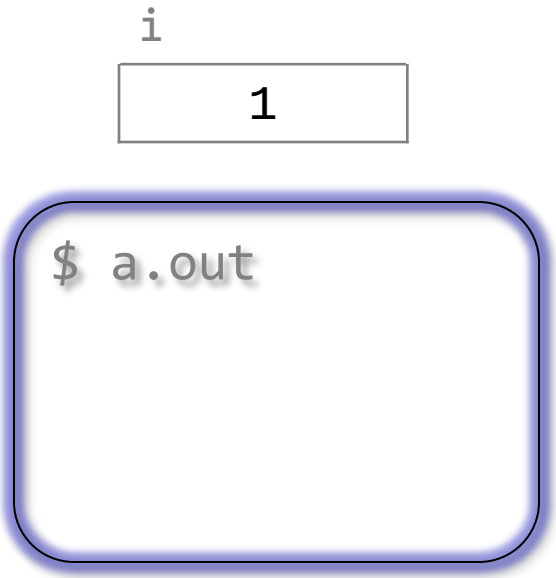
프로그램 5.6

```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```

while

프로그램 5.6

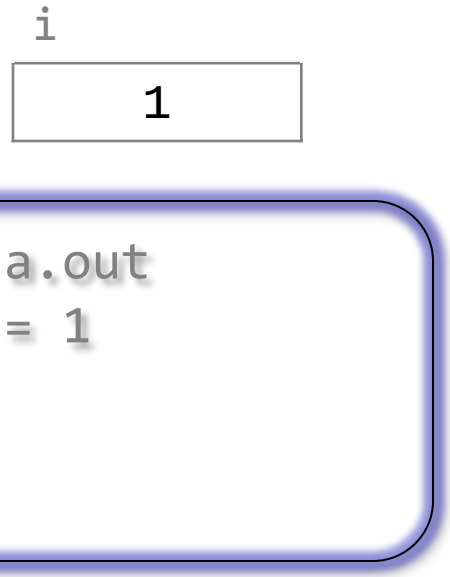
```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```



while

프로그램 5.6

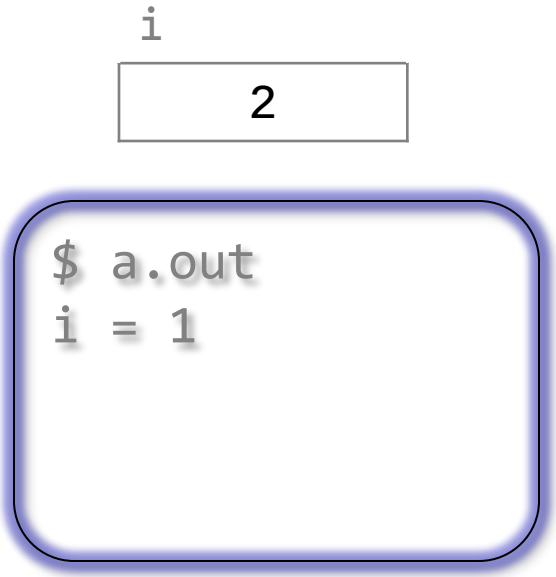
```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```



while

프로그램 5.6

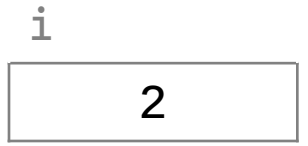
```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```



while

프로그램 5.6

```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```

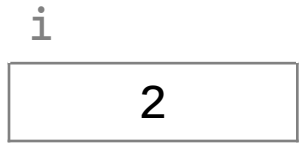



```
$ a.out
i = 1
```

while

프로그램 5.6

```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```

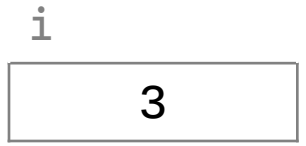



```
$ a.out
i = 1
i = 2
```

while

프로그램 5.6

```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```

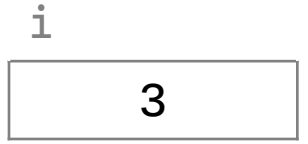



```
$ a.out
i = 1
i = 2
```

while

프로그램 5.6

```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```

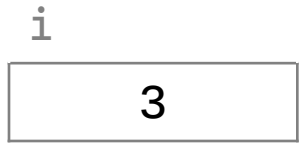



```
$ a.out
i = 1
i = 2
```

while

프로그램 5.6

```
#include <stdio.h>
int main(void){
    int i;
    i = 1;
    while (i < 3){
        printf("i = %d\n", i);
        i++;
    }
    printf("while 문 종료\n");
    return 0;
}
```




```
$ a.out
i = 1
i = 2
while 문 종료
```

while

- 반복문은 같은(일정한 패턴이 있는) 일을 반복해서 해야 하는 경우 사용
  - n! = n × (n-1) × (n-2) × (n-3) × … × 2 × 1
  - n! = n × (n-1) × (n-2) × (n-3) × … × (n-(n-2)) × (n-(n-1))
  - n! = 1 × 2 × 3 × … × (n-1) × n
  - n! = ((…((1 × 2) × 3) × … × (n-1)) × n)

while

프로그램 5.7

```
#include <stdio.h>
int main(void){
    unsigned f;
    int i;
    i = f = 1;
    while (i <= 5){
        f *= i;
        i++;
    }
    printf("5! = %u\n", f);
    return 0;
}
```

while

프로그램 5.7

```
#include <stdio.h>
int main(void){
    unsigned f;
    int i;
    i = f = 1;
    while (i <= 5){
        f *= i;
        i++;
    }
    printf("5! = %u\n", f);
    return 0;
}
```

while

프로그램 5.8

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    f = n;
    printf("%d! = ", n);
    while (--n)
        f *= n;
    printf("%llu\n", f);
    return 0;
}
```

주의 사항

1. 무한 루프

- 종료되지 않는 반복문
- 실수로 인한 무한 루프에 빠지는 경우가 많음
- 예

```
while (--n)          // 주의
    f *= n;
```

- 모든 경우를 대비하여 프로그래밍 해야 함
- 수정된 코드

```
while (--n > 0)
    f *= n;
```

주의 사항

2. 공백 문장

- 수식이 없는 문장(세미콜론만 있는 문장)
- 공백 문장도 독립적인 줄에 명시하는 것이 좋음
- 예

```
while  ((c = getchar()) == ' ')  ;
```

  - 공백 문자를 무시하는 코드
- 수정된 코드

```
while  ((c = getchar()) == ' ')
    ;
```

주의 사항

3. 등가 실수 수식

- 실수 연산은 정확하지 않음
- 실수 계산 오류를 항상 고려해야 함
- 예

```
float x = sum = 0.0;
while (x != 9.9){
    sum += x;
    x += 0.1;
}
```

- 수정 코드

```
while (x < 9.9){
```

- 등가 수식보다는 관계 수식이 안전함

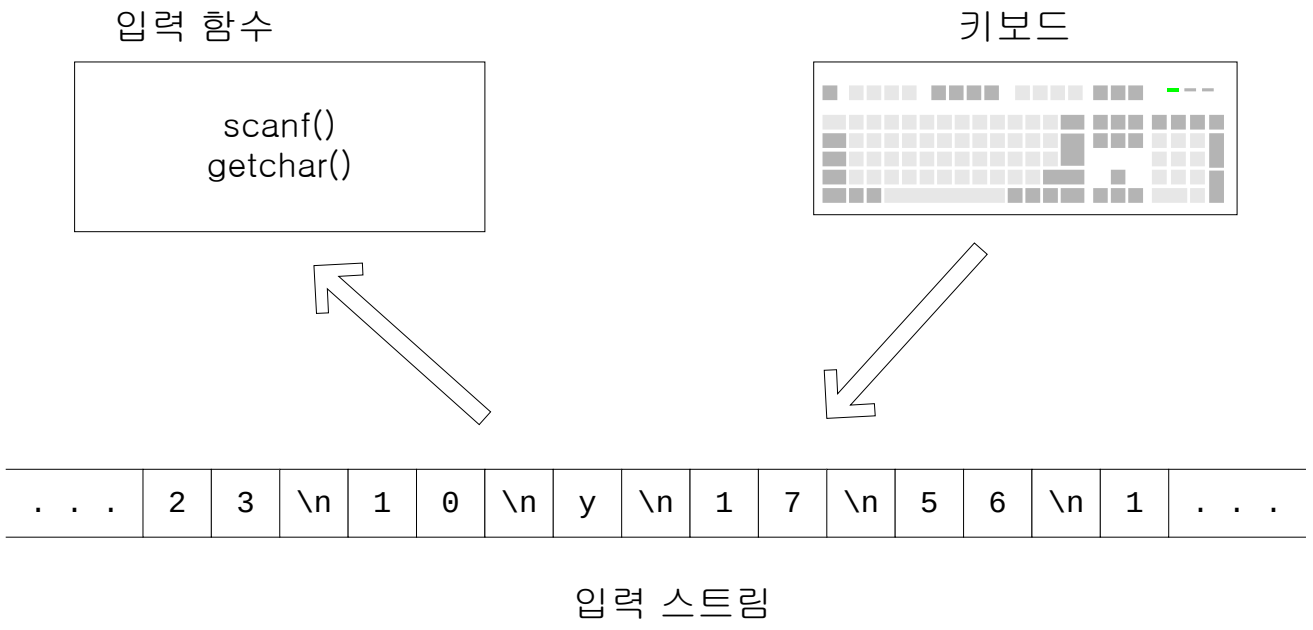
중첩 while

프로그램 5.9

```
printf("=== 게임 시작 ===\n");
while (game){
    printf("\n= %u 번째 게임 =\n", num++);
    . . .
    while (i++ <= 7) {
        printf("0부터 99 사이의 수를 입력 하세요 : ");
        scanf("%u", &guess);
        . . .
    }
    printf("\n다시 하겠습니까? (y/n) ");
    getchar();     // 개행 문자 삭제
    again = getchar();
    if (!(again == 'y' || again == 'Y'))
          game = 0;
    . . .
}
printf("=== 게임 종료 ===\n");
```

입력

- scanf()나 getchar()와 같은 입력 함수 키보드로부터 입력된 내용을 입력 버퍼(입력 스트림)에서 읽음



입력

- scanf()
  - 여백문자들은 무시하고 변환명세에 지정된 형식으로 읽을 수 있는 곳까지만 읽음
- getchar()
  - 입력 버퍼에 있는 여백 문자 여부와 상관 없이 첫 번째 문자를 하나 읽음
  - 키보드로 입력 받을 경우 원하는 문자인지 확인할 필요가 있음

입력 프로그램

프로그램 5.10

```
printf("=== 게임 시작 ===\n");
while (game){
    printf("\n= %u 번째 게임 =\n", num++);
    . . .
    while (i++ <= 7) {
        printf("0부터 99 사이의 수를 입력 하세요 : ");
        scanf("%u", &guess);
        . . .
    }
    printf("\n다시 하겠습니까? (y/n) ");
    scanf("%s", again);
    if (!(again[0] == 'y' || again[0] == 'Y'))
        game = 0;

    . . .
}
printf("=== 게임 종료 ===\n");
```

입출력 재지정

- 입출력 재지정을 통해 파일 입출력을 할 수 있음
- 입력 재지정 : <
- 출력 재지정 : >
- 예

```
$ a.out < input_file > output_file
```

  - a.out은 input_file 파일을 표준 입력으로 output_file 파일을 표준 출력으로 사용

프로그램 예제

프로그램 5.11

```
#include <stdio.h>
int main(void)
{
   int  c, total_byte = 0, line_count = 0;

   while ((c = getchar()) != EOF){
      total_byte++;
      if (c == '\n')
         ++line_count;
   }
   printf("파일 크기 : %d 바이트\n", total_byte);
   printf("파일 줄 수 : %d 행\n", line_count);

   return 0;
}
```

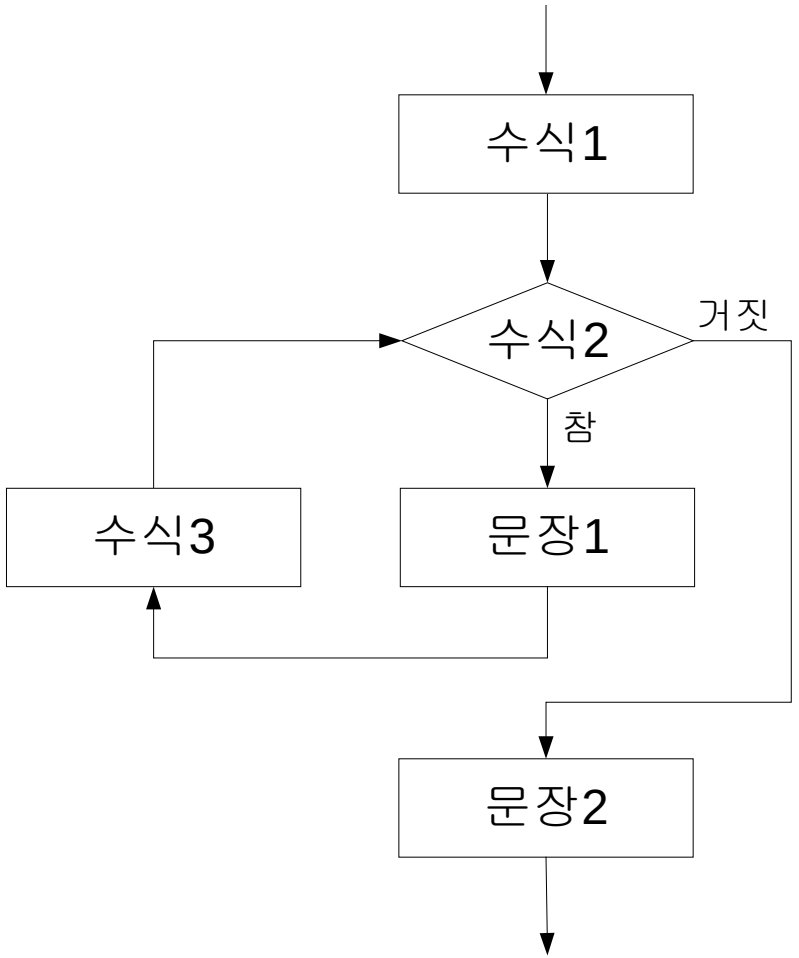
프로그램 결과

```
$ file_info  <  file_info.c
파일 크기 : 297 바이트
파일 줄 수 : 15 행
$ file_info  <  file_info.c  >  output
$ cat output
파일 크기 : 297 바이트
파일 줄 수 : 15 행
$ file_info  <  file_info.c  >>  output
$ cat output
파일 크기 : 297 바이트
파일 줄 수 : 15 행
파일 크기 : 297 바이트
파일 줄 수 : 15 행
$
```

for

- **일반적인 형태**

```
for (수식1; 수식2; 수식3)
    문장1
문장2
```



for

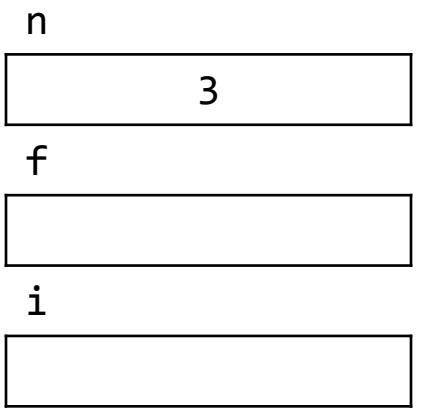
프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

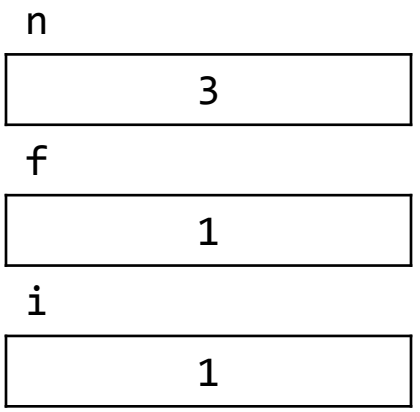



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

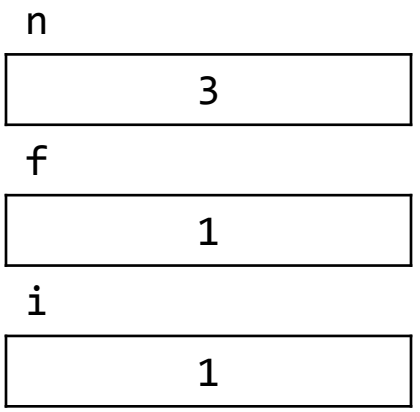



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

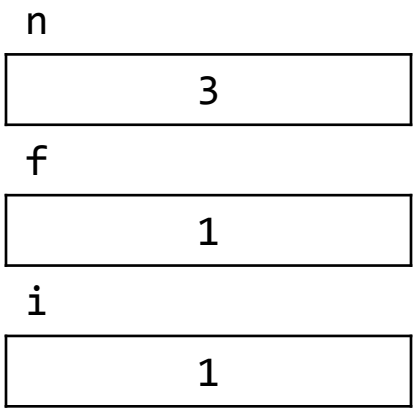



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

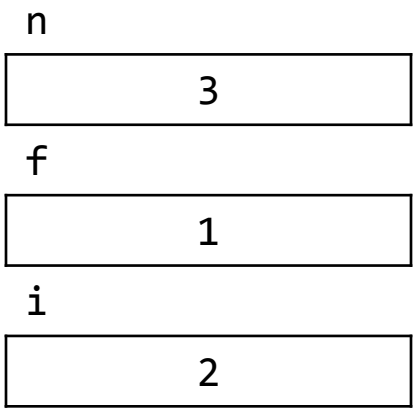



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

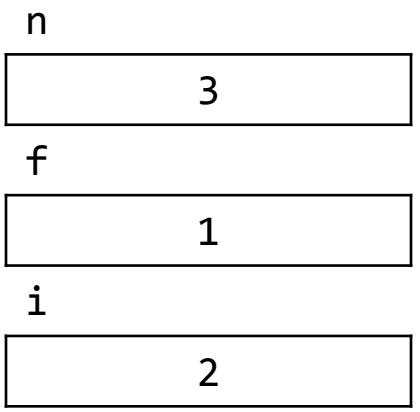



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

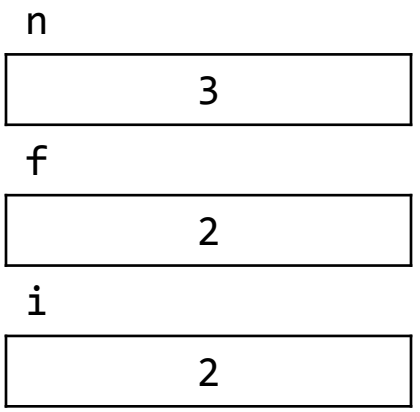



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

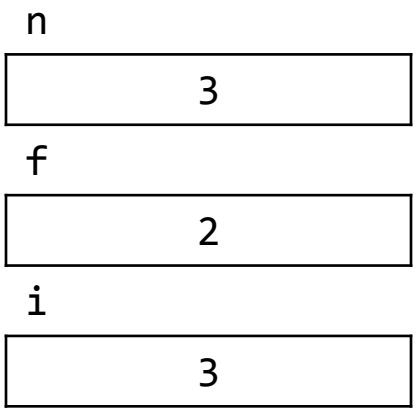



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

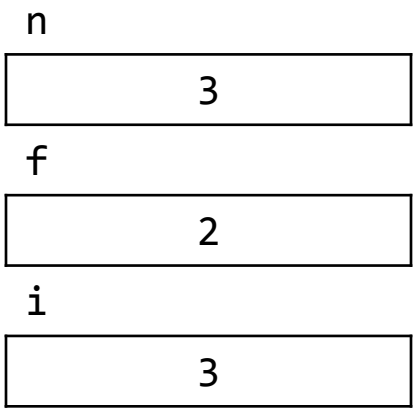



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

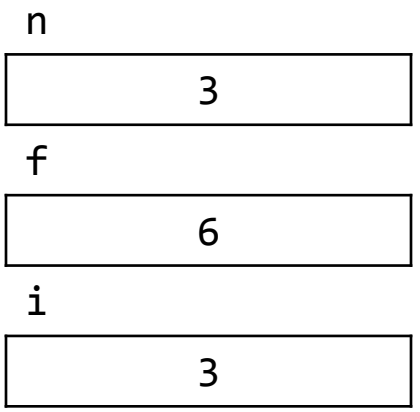



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

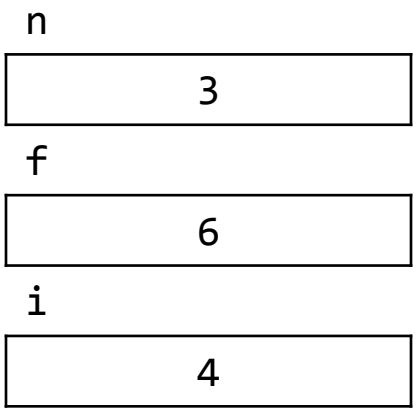



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

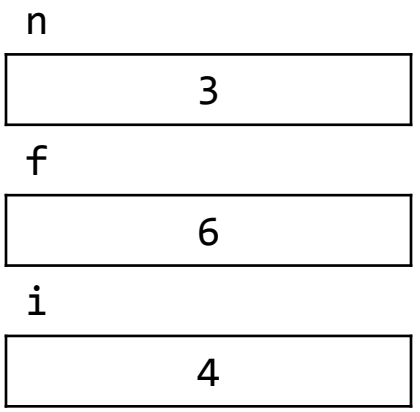



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```

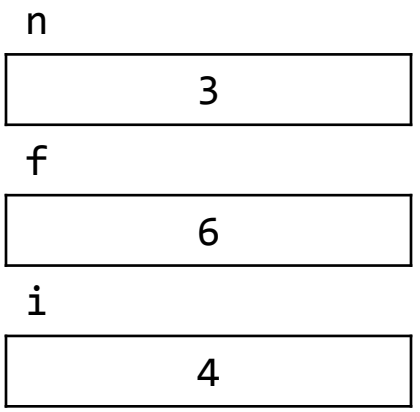



```
$ a.out
입력 하세요 : 3
```

for

프로그램 5.12

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n >= 0){
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
    }
    else
        printf("음수를 입력했습니다.\n");
    return 0;
}
```




```
$ a.out
입력 하세요 : 3
3! = 6
$
```

for

- 수식1,수식2, 수식3 은 생략 가능

```
for (f = i = 1; i <= n; i++)
    f *= i;
```

- 수식1 생략

```
f = i = 1
for ( ; i <= n; i++)
    f *= i;
```

- 수식3 생략

```
for (f = i = 1; i <= n; )
    f *= i++;
```

for

- 수식1,수식2, 수식3 은 생략 가능

```
for (f = i = 1; i <= n; i++)
    f *= i;
```

- 수식1, 수식3 생략

```
f = i = 1
for ( ; i <= n; )
    f *= i++;
```

- 수식1, 수식2, 수식3 생략

```
f = i = 1
for ( ; ; )          // 무한 루프
    f *= i++;
```

* 수식2가 없으면 조건이 항상 참임을 의미함

for 문에서 선언문

- C99
- for의 첫 번째 수식에 for에서만 사용할 변수를 선언할 수 있음

```
for (int i = 1; i <= n; ++i)
    f *= i;
```

  - for 문의 첫 번째 수식에 i가 선언됨
  - 여기서 선언된 i는 이 for 문에서만 사용할 수 있음

프로그램 예제

프로그램 5.13

```
int main(void){
    double y;
    int n, i = 0;
    printf("2^n 프로그램\n");
    printf("n : ");
    scanf("%d", &n);
    if (n < 0){
        i = 1;
        n = -1 * n;
    }
    y = 1.0;
    for (int i = 1; i <= n; i++)
        y *= 2;
    if (i)
        y = 1.0 / y;
    printf("2^%d = %f\n", i ? -n : n, y);
}
```

프로그램 결과

```
$ power
2^n 프로그램
n : 3
2^3 = 8.000000
$ power
2^n 프로그램
n : -3
2^-3 = 0.125000
```

콤마 연산자

- 일반적인 형태

  수식1, 수식2

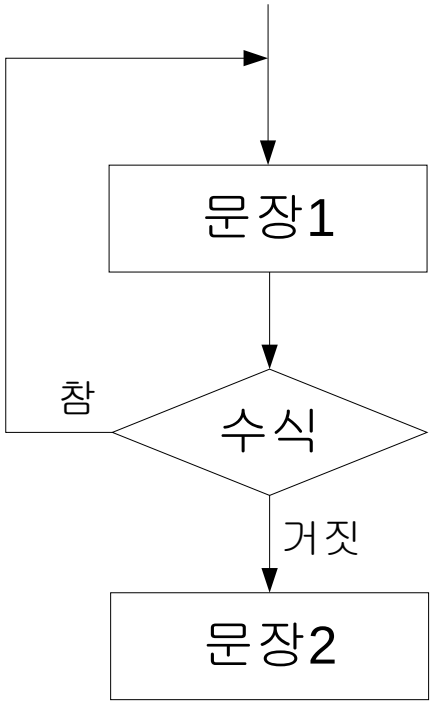
  – 수식1이 먼저 수행되고 수식2가 수행됨
- 모든 연산자들 중에서 가장 낮은 우선순위를 가짐
- 콤마 수식의 값과 형은 콤마 연산자의 오른쪽 수식의 값과 형이 됨
- for 문에서 많이 사용

```
for (f = 1, i = 2; i <= n; ++i)
    f *= i;
```

do-while

- **일반적인 형태**

```
do
    문장1
while (수식) ;
문장2
```



do-while

- do 문은 몸체 부분을 반드시 한번 이상 실행해야 할 때 유용
- 예제

```
do {
    printf("계승을 구할 수를 입력 하세요 : ");
    scanf("%d", &n);
    if (n < 0)
        printf("\n음수를 입력했습니다. 양수를 입력하세요.\n");
} while (n < 0);
```

break

- 반복문이나 switch 문에 사용되고 제어를 현재 구조물에서 벗어나게 함
- 반복문에서 조건에 따라 반복을 멈춰야 할 때 유용함
- switch 문에서 적절한 문장만을 실행하게 하기 위해 사용됨

프로그램 예제

프로그램 5.14

```
#include <stdio.h>
int main(void){
    unsigned long long f;
    int n, i;
    while (1){
        printf("계승을 구할 수를 입력 하세요 : ");
        scanf("%d", &n);
        if (n >= 0)
            break;
        printf("음수를 입력했습니다. 양수를 입력하세요.\n");
    }
    // while 문 몸체에 있는 break를 만나면 제어는 여기로 넘어옴
    for (f = i = 1; i <= n; i++)
        f *= i;
    printf("%d! = %llu\n", n, f);
    return 0;
}
```

프로그램 결과

```
계승을 구할 수를 입력 하세요 : -10
음수를 입력했습니다. 양수를 입력하세요.
계승을 구할 수를 입력 하세요 : -3
음수를 입력했습니다. 양수를 입력하세요.
계승을 구할 수를 입력 하세요 : 10
10! = 3628800
```

continue

- 반복문에서 사용되어 현재 반복을 멈추고 즉시 다음 반복을 하게 함

continue

프로그램 5.15

```
int main(void){
    unsigned long long f;
    int n, i, done = 1;
    while (done){
        printf("계승을 구할 수를 입력 하세요 : ");
        scanf("%d", &n);
        if (n < 0){
            printf("음수를 입력했습니다. 양수를 입력하세요.\n");
            continue;
        }
        for (f = i = 1; i <= n; i++)
            f *= i;
        printf("%d! = %llu\n", n, f);
        done = 0;
        // continue를 만나면 제어는 여기로 넘어옴
    }
}
```

continue

프로그램 5.16

```
#include <stdio.h>
int main(void){
    int n, i, sum;

    printf("n을 입력하세요 : ");
    scanf("%d", &n);
    for (sum = i = 0; i <= n; i++){
        if (i % 2)
            continue;
        sum += i;
        // continue를 만나면 제어는 여기로 넘어옴
        // for 문의 세 번째 수식 (i++)가 실행됨
    }
    printf("%d까지 짝수의 합 = %d\n", n, sum);
    return 0;
}
```

goto

- goto 문은 현재 함수 내의 레이블이 붙은 문장으로 무조건 분기함
- 다른 제어 흐름 메커니즘(for, while, do, if, switch)들이 제공하는 구조적 프로그래밍에 악영향을 줌
- 가급적 사용하지 않는 것이 바람직함

레이블 문장

- **일반적인 형태**

  식별자: 문장

  – 콜론 앞의 식별자를 레이블이라고 함

- **예제**

```
bye:   return 0;
add:   a = b + c;
error1: error2: printf("오류\n");   // 다중 레이블
```

- **레이블은 자신의 이름 영역을 가짐**

  – 동일한 식별자가 레이블과 변수 모두로 사용될 수 있음

goto

- **goto 문의 일반적인 형태**

  goto 레이블;

  – 이 문장을 만나면 레이블이 붙은 문장으로 제어가 넘어감

goto

프로그램 5.17

```
int main(void){
   unsigned long long f;   int n, i;
   goto read;
compute:
   for (f = i = 1; i <= n; i++)
      f *= i;
   printf("%d! = %llu\n", n, f);
   goto end;
read:
   printf("계승을 구할 수를 입력 하세요 : ");
   scanf("%d", &n);
   if (n < 0){
      printf("음수를 입력했습니다. 양수를 입력하세요.\n");
      goto read;
   }
   goto compute;
end:
   return 0;
}
```

goto

프로그램 5.17

```
int main(void){
   unsigned long long f;   int n, i;
   goto read;
compute:
   for (f = i = 1; i <= n; i++)
      f *= i;
   printf("%d! = %llu\n", n, f);
   goto end;
read:
   printf("계승을 구할 수를 입력 하세요 : ");
   scanf("%d", &n);
   if (n < 0){
      printf("음수를 입력했습니다. 양수를 입력하세요.\n");
      goto read;
   }
   goto compute;
end:
   return 0;
}
```

goto

프로그램 5.17

```
int main(void){
   unsigned long long f;   int n, i;
   goto read;
compute:
   for (f = i = 1; i <= n; i++)
      f *= i;
   printf("%d! = %llu\n", n, f);
   goto end;
read:
   printf("계승을 구할 수를 입력 하세요 : ");
   scanf("%d", &n);
   if (n < 0){
      printf("음수를 입력했습니다. 양수를 입력하세요.\n");
      goto read;
   }
   goto compute;
end:
   return 0;
}
```

goto

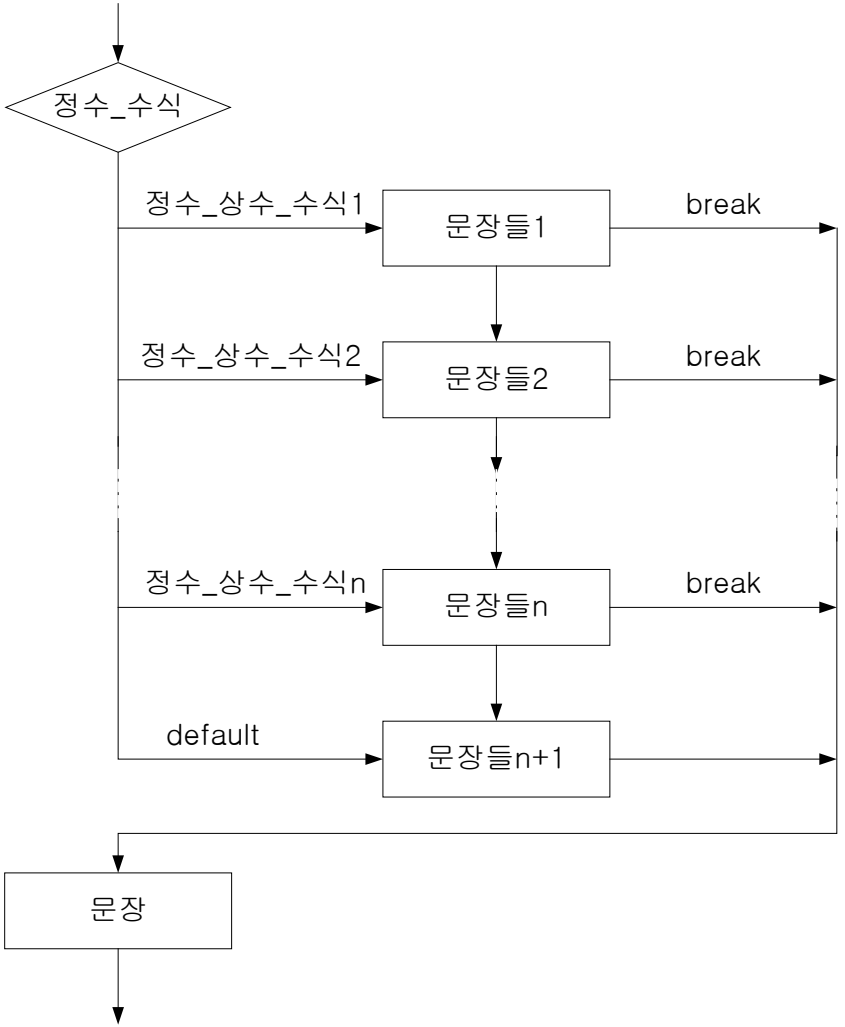
프로그램 5.17

```
int main(void){
   unsigned long long f;   int n, i;
   goto read;
compute:
   for (f = i = 1; i <= n; i++)
      f *= i;
   printf("%d! = %llu\n", n, f);
   goto end;
read:
   printf("계승을 구할 수를 입력 하세요 : ");
   scanf("%d", &n);
   if (n < 0){
      printf("음수를 입력했습니다. 양수를 입력하세요.\n");
      goto read;
   }
   goto compute;
end:
   return 0;
}
```

switch

- **일반적인 형태**

```
switch (정수_수식) {
case 정수_상수_수식1 :
  문장들1
case 정수_상수_수식2 :
  문장들2
...
case 정수_상수_수식n :
  문장들n
default :
  문장들n+1
}
문장
```



switch

- 다중 조건문
- if 문을 여러 개 중첩하여 사용하는 것보다 실행 속도가 빠름
- 각 case 문 문장들 다음에 break가 있어야 되는 경우가 보통임
  - break가 생략되면 이상한 결과를 얻을 수 있음

switch

프로그램 5.18 일부

```
int main(void){
   . . .
   switch(op) {
   case '+' :
      printf("%.2f + %.2f = %.2f\n", opd1, opd2, opd1 + opd2);
      break;
   case '-' :
      printf("%.2f - %.2f = %.2f\n", opd1, opd2, opd1 - opd2);
      break;
   case '*' :
      printf("%.2f * %.2f = %.2f\n", opd1, opd2, opd1 * opd2);
      break;
   case '/' :
      printf("%.2f / %.2f = %.2f\n", opd1, opd2, opd1 / opd2);
      break;
   default :
      printf("잘못된 연산자입니다.\n");
   }
   . . .
}
```