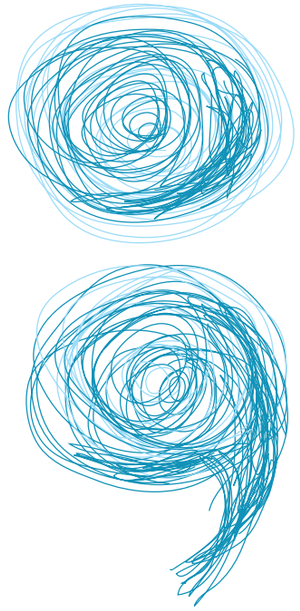


chapter

5

제어문

5.0 개요

프로그램의 각 문장은 원칙적으로는 위에서 아래로 순차적으로 실행된다. 그러나 성적이 70점 이상이면 '합격'을, 70점 미만이면 '불합격'을 출력하는 문제처럼 조건에 따라 실행할 문장이 달라지거나, 학생 1,000명에 대해 다섯 과목의 평균과 순위를 구하는 문제처럼 특정 내용을 반복 실행하는 경우에는 제어문을 사용해 실행 흐름을 제어할 수 있다. 프로그래밍의 핵심이라 할 수 있는 제어문은 조건문, 반복문, 그리고 분기문과 같은 기타 제어문으로 나눌 수 있다.

점수가 70점 이상이면 합격자수를 1 증가시키고 아니면 불합격자수를 1 증가시키기, 윤년이면 2월 마지막 일을 29로 저장하고 아니면 28로 저장하기, x가 음수가 아니라면 절대치를 x로 저장하고 음수면 −x로 저장하기, 게임 계속 진행 여부를 물어 입력받은 답에 따라 게임을 종료하거나 게임 다시하기와 같이 특정 조건의 만족 여부에 따라 실행할 내용이 달라질 경우 조건문을 사용한다. C 언어에서 제공하는 조건문으로는 if문과 switch문이 있다.

학생 1,000명의 성적 처리, 500번의 실험 데이터에 대한 통계, 직원 100명의 급여 처리처럼 동일한 처리를 여러 번 반복할 경우에는 반복문을 사용한다. C 언어에서 제공하는 반복문으로는 for문, while문, do~while문이 있다.

5.1 조건문(if문, switch문)

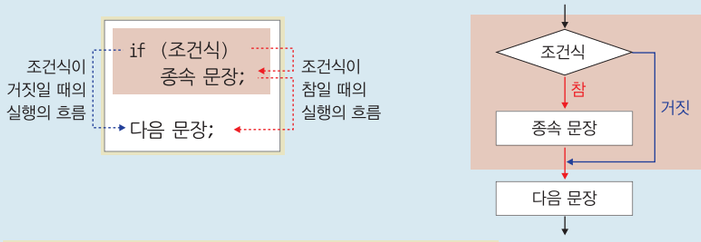
5.1.1 if문

if문은 주어진 조건을 만족하는지에 따라 실행할 문장이 다를 때 사용한다. if문은 else가 없는 단순 if문과 else를 포함하는 if~else문으로 나뉜다.

단순 if문

단순 if문은 주어진 조건을 만족할 경우에만 추가로 처리할 내용이 있을 때 사용한다. 예를 들면 성적이 90점 이상일 때만 '우수' 출력하기, 점수가 평균보다 작을 때만 평균 미만자의 수를 1 증가시키기, 자격증이 있을 때만 가산점 주기 등이 있다. 단순 if문의 형식과 흐름도는 [그림 5-1]과 같다.

형식 단순 if문




```
if (조건식)
{
    종속 문장 1;
    종속 문장 2;
    …
    종속 문장 n;
}
```

조건식이 참일 때 실행할 내용이 여러 문장인 경우에는 반드시 {}로 묶어줘야 한다.

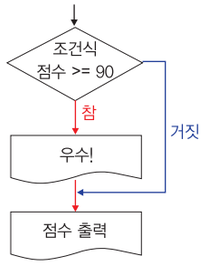
- 조건식: 조건식의 결과 논리값이 참(true)이면 종속 문장을 실행하고, 거짓(false)이면 종속 문장을 실행하지 않은 채 if문을 끝낸다. 즉 조건을 만족하는 경우에만 종속 문장을 실행한다.
- 복합문(compound statement): 조건식이 참일 때 실행할 문장이 두개 이상일 경우에는 {}를 사용해 문장들을 하나로 묶어줘야 한다. 이와 같이 {}로 묶인 문장을 복합문이라고 한다. if문에서 처리할 문장이 하나일 경우에도 가능하면 {}를 써주는 것이 좋다. 프로그램을 이해하기 쉽고, 종속 문장을 추가했을 때 {}를 빠뜨려 에러가 나는 것을 방지할 수 있으며, 종속 문장의 의미가 정확해지기 때문이다.

예
```
// score가 70 이상일 때만 통과자 수 pass를 1 증가시키기
if (score >= 70)
   pass++;
```

예
```
// score가 70 이상일 때만 통과자 수 pass를 1 증가시키고 sum에 점수 score를 누적하기
if (score >=70)
{
   pass++;
   sum = sum + score;
}
```

점수가 90점 이상인 경우에만 '우수!'를 출력하는 경우를 살펴보자. 이 상황을 처리하기 위해서는 if문을 사용해 조건(90점 이상)에 따라 종속 문장('우수!' 출력)을 실행하거나 실행하지 않도록 조절해야 하므로 다음과 같이 작성한다.

```
// 점수 score가 90이상일 때만 '우수!' 출력하기
if (score >= 90)
   printf("우수!\n");
printf("점수 : %d \n", score);
```



조건식이 논리식이라면 결과가 참 또는 거짓이지만 다음 예처럼 수치나 대입문이라면 4장에서 소개한 바와 같이 0만 거짓이고 0을 제외한 모든 수치가 참으로 해석된다.

예
```
if (n % 2)
   printf("%d는 홀수입니다.\n", n);
```

위의 예에서 n을 2로 나눈 나머지가 1이면 홀수다. 즉 식의 결과 값이 1이면 수치 1은 참을 의미하므로 다음 문장이 실행되어 홀수라고 출력한다. 식의 결과 값이 0이면 거짓을 의미하므로 if문을 끝내고 다음 문장을 실행한다.

다음은 비교 연산자 '==' 대신 대입 연산자 '='를 잘못 사용한 예다.

예
```
if (n = 0)
   printf("%d는 0입니다.\n", n);
```

n에 0이 아닌 값이 저장되어 있더라도 이 조건식의 대입문에 의해 n 값은 무조건 0이 된다. 따라서 조건식의 최종 결과는 대입문의 왼쪽 변수 값인 0이 되어 거짓을 의미하게 되므로 메시지를 출력하지 못한다.

if문의 조건식에는 관계 연산자와 논리 연산자가 많이 사용된다. 현재 x 값이 5와 10 사이인지를 확인하기 위해 if문을 다음과 같이 작성했다고 가정하자.

예
```
if (5 <= x <= 10)   ← 잘못된 조건식
   printf("5와 10 사이의 수입니다.\n");
```


위의 if문을 실행하면 x 값과 관계없이 언제나 "5와 10 사이의 수입니다."가 출력된다. 그 이유는 첫 번째 연산자가 우선 적용되므로 조건식 (5 <= x <= 10)가 ((5 <= x) <= 10)로 해석되기 때문이다. 논리식 (5 <= x)의 결과는 논리값 참 또는 거짓이 된다. 논리값과 점수 10에 대해 관계 연산자 <=를 적용하면 논리값이 정수로 형 변환된다. 결국 (5 <= x)가 참이면 (1 <= 10)이 되고, (5 <= x)가 거짓이면 (0 <= 10)이 되므로 둘 다 참이 된다. 이렇게 전체 논리식의 결과는 항상 참이 되어 출력문을 매번 실행하게 된다. 그러므로 올바른 출력을 위해서는 다음과 같이 논리 연산자 &&를 이용해 조건식을 만들어야 한다.

```
if ((5 <= x) && (x <= 10))
   printf("5와 10 사이의 수입니다.\n");
```

[프로그램 5-1]은 현재 온도를 입력받아 출력하는 프로그램이다. 단 입력받은 온도가 0도 미만이면 영하라는 메시지를 함께 출력하도록 한다.

프로그램 5-1 단순 if문을 이용해 오늘의 온도 출력하기 (ch5-1.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int temp;
6
7       printf("현재 온도는? ");
8       scanf("%d", &temp);
9
10      if (temp < 0)                    ← 현재 온도가 영하인 경우에만 실행
11          printf("오늘은 영하이며 ");
12
13      printf("현재 온도는 %d도입니다.\n", temp);
14
15      return 0;
16  }
```

실행결과
```
현재 온도는? -2
오늘은 영하이며 현재 온도는 -2도입니다.
```

실행결과
```
현재 온도는? 5
현재 온도는 5도입니다.
```

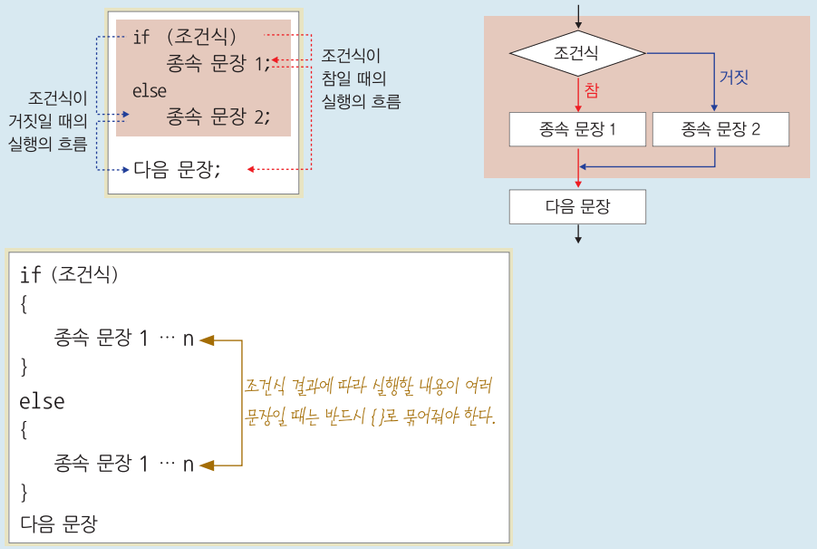
설명

10: 입력 값이 −2이면 조건식의 결과 값이 참이므로 11행을 실행하고, 입력 값이 5이면 조건식의 결과 값이 거짓이므로 11행을 실행하지 않은 채 13행을 실행한다.

if~else문

주어진 조건을 만족할 때와 만족하지 않을 때의 처리할 내용이 다른 경우, 즉 조건의 참과 거짓에 따라 둘 중 하나를 선택해 처리할 경우에 if~else문을 이용한다.

형식 if~else문



- if문의 조건식이 참이면 종속 문장 1을 실행하고 거짓이면 else로 이동하여 종속 문장2를 실행한다. 결과적으로 조건식의 결과 값에 따라 종속 문장 1이나 종속 문장 2 중 하나를 반드시 실행한다. 단순 if문과 마찬가지로 종속 문장 1과 종속 문장 2가 여러 문장이라면 { }로 묶어서 복합문으로 표현해야 한다.

예
```
// 점수가 70 이상이면 '합격!', 70 미만이면 '불합격!' 출력하기
if (score >= 70)
   printf("합격!\n");
else
   printf("불합격!\n");
```

예
```
// 합격 여부를 출력하고 합격자 수 pass와 불합격자 수 fail을 1 증가시키기
if (score >= 70)
{
   printf("합격!\n");
   pass++;
}
else
{
   printf("불합격!\n");
   fail++;
}
```

주의
- 쓸데없이 ;을 붙이지 않아야 한다.

```
1  // 20세 이상이면 adult를 1 증가, 20세 미만이면 child를 1 증가
2  if (age >= 20);
3    adult++;
4  else
5    child++;
```

조건식 뒤에 ;을 붙이면 컴파일러는 조건식이 참일 때 처리할 문장이 없는 것으로 해석한다. 즉 3행에 else가 없어 2행에서 단순 if문만 있는 것으로 판단한다. 그런데 4행에서 갑자기 else가 나타나면 새로운 if 없이 바로 else가 있다고 판단하여 에러를 발생시킨다.

- else 뒤에 조건식을 단독으로 사용할 수 없다.

```
1  if (age >= 20)
2    adult++;
3  else (age < 20)
4    child++;
```

1행의 조건문이 거짓이라면 3행으로 실행 순서가 변경된다. 즉 else에 오게 되는 경우는 age가 20보다 작을 때 뿐이다. 그러므로 굳이 조건식을 적을 필요가 없을뿐더러 if 없이 조건식이 올 수 없으므로 3행은 에러를 일으킨다. else if (age < 20)으로 작성하면 에러가 나지 않지만 굳이 if문을 사용할 필요는 없다.

간단한 if~else문은 4장에서 소개한 조건 연산자를 이용해 다음과 같이 표현할 수도 있다.

```
(score >= 70)?printf("합격!\n"):printf("불합격!\n");
```

[프로그램 5-2]는 키보드로 입력받은 정수가 짝수인지 홀수인지 출력하는 프로그램을 if~else문을 이용해 작성한 것이다. 이때 0은 짝수에 포함시키기로 한다.

프로그램 5-2 입력받은 정수가 짝수인지 홀수인지 출력하기: if~else문 (ch5-2.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int num;
```

```
6
7       printf("정수를 입력하시오 : ");
8       scanf("%d", &num);
9
10      printf("입력한 수 %d는(은) ", num);
11
12      if (num % 2 == 0)
13          printf("짝수입니다.\n");
14      else
15          printf("홀수입니다.\n");
16
17      return 0;
18  }
```

짝수는 2의 배수이므로 num을 2로 나눈 나머지가 0이다.

```
실행결과
정수를 입력하시오 : 7
입력한 수 7는(은) 홀수입니다.
```

설명

12: 입력 값이 7이면 2로 나누었을 때 나머지가 1이 되어 조건식의 결과 값이 거짓이 되므로 15행을 실행한다.

중첩된 if문

특정 조건을 만족할 때만 추가로 어떤 일을 할 경우에는 단순 if문을, 두 가지 중 한 가지를 선택해 처리할 때는 if~else문을 사용한다. 그리고 여러 선택 사항 중 한 가지를 골라 처리하거나 여러 가지 복잡한 조건에 따라 선택적으로 처리할 내용이 있을 때는 if문 안에 다시 if문을 사용하여 해결할 수 있다.

if문 안에 또 다른 if문을 포함시킨 것을 중첩된(nested) if문이라고 한다. [프로그램 5-3]은 입력받은 정수가 양의 짝수인지, 음의 짝수인지, 양의 홀수인지, 음의 홀수인지를 구분하여 출력하도록 [프로그램 5-2]를 수정한 것이다. 이때 0은 별도로 분류하지 말고 양의 짝수에 포함시키기로 하자.

프로그램 5-3 입력받은 정수가 짝수인지 홀수인지 출력하기: 중첩된 if문 (ch5-3.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int num;
6
7       printf("정수를 입력하시오 : ");
8       scanf("%d", &num);
9
10      printf("입력한 수 %d는(은) ", num);
11
12      if (num >= 0)   // num이 양수면
13      {
14          if (num % 2 == 0)
15          printf("양의 짝수입니다.\n");
16          else
17          printf("양의 홀수입니다.\n");
18      }
19      else    // num이 음수면
20      {
21          if (num % 2 == 0)
22          printf("음의 짝수입니다.\n");
23          else
24          printf("음의 홀수입니다.\n");
25      }
26
27      return 0;
28  }
```

```
실행결과
정수를 입력하시오 : -3
입력한 수 -3는(은) 음의 홀수입니다.
```

```
실행결과
정수를 입력하시오 : 4
입력한 수 4는(은) 양의 짝수입니다.
```

설명

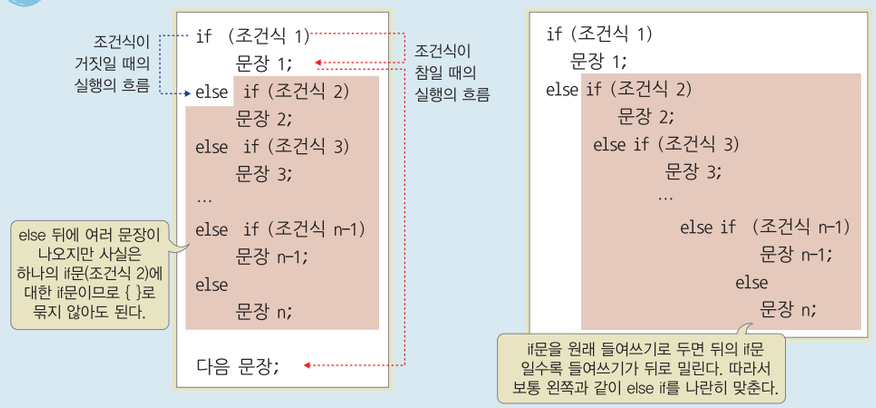
14~17: 12행의 if문 안에 하나의 if문이 더 포함되어 있다. 이곳은 num이 양수일 때만 실행되므로 14행에서는 짝수 여부만 판단하여 양의 짝수 또는 양의 홀수를 출력한다.

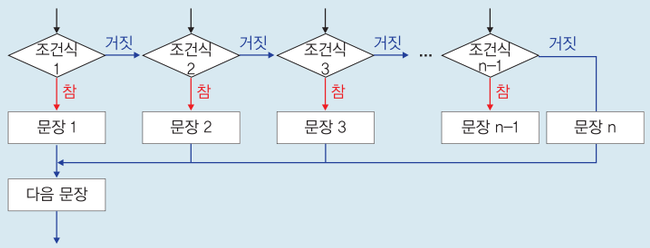
21~24: 12행의 if문 조건식이 거짓일 때 즉 num이 음수일 때만 실행된다.

13,18,20,25: 14~17행과 21~24행의 if문은 4행으로 구성되어 있지만 실질적으로는 if문 한 개이므로 {}로 묶지 않아도 된다. 여기서는 중첩된 if문의 구조와 범위를 쉽게 볼 수 있도록 하기 위해 {}를 사용하였다.

중첩된 if문 중 많이 사용되는 모양은 아래 형식과 같이 else 뒤에 곧바로 if문이 추가된 형태다. 이 중첩된 if문은 주로 나열된 if의 여러 조건 중 한 개만 선택되어 해당 문장을 실행하는 데 사용된다.

형식 중첩된 if문





- if문이 시작되면 먼저 조건식 1을 검사한다. 조건식 1이 참이면 문장 1을 실행한 후 else 뒤의 내용은 실행하지 않은 채 다음 문장을 실행한다. 그러나 조건식 1이 거짓이면 else로 이동한다. else 뒤에 if문이 중첩되어 있으므로 조건식 2를 검사하여 참이면 문장 2를 실행한 후 if문을 끝내고 다음 문장을 실행한다. 같은 방식으로 else if 뒤의 조건식을 계속 처리하다가 조건식 (n−1)이 거짓이면 문장 n을 실행한 후 다음 문장을 실행한다. 즉, 마지막 else 뒤의 문장 n은 조건식 1~조건식 (n−1)이 모두 거짓인 경우에 실행된다.
- else 뒤에 바로 if문이 올 때 else와 if 사이에는 반드시 공백이 있어야 한다.

예
```
// 입력된 x가 양수인지, 0인지, 음수인지 셋 중 한 가지를 출력하기
if (x > 0)
   printf("양수입니다.\n");
else if (x == 0)
   printf("0입니다.\n");
else
   printf("음수입니다.\n");
```


if와 else의 매칭

if문이 중첩되는 경우에는 나타난 else문이 어떤 if문과 매칭되는지 주의해서 구분해야 한다. 우선 나타난 if문이 else가 없는 단순 if문인지 구별해야 하고, 단순 if문이 아니라면 else가 어떤 if문과 짝을 이루는지를 정확히 구분한다.

다음은 중첩된 if문에서 else가 한 개인데 반해 if문이 두 개다. 그러면 5행의 else문은 어느 if문과 짝이 될까?

예
```
1  scanf("%d", &x);
2  if (x >= 0)
3     if (x % 2 == 0)
4        printf("양의 짝수입니다.\n");
5  else
6     printf("음수입니다.\n");
```
3행의 if와 짝

5행의 else와 2행의 if의 들여쓰기 위치가 나란하므로 2행의 조건식이 거짓일 때 6행을 수행할 것처럼 보인다. 그러나 실제로는 x 값으로 3을 입력하면 '음수입니다.'를 출력한다. 이는 컴파일러의 관점에서 해석상의 문제가 발생하기 때문이다.

컴파일러는 else와 짝을 이루는 if를 찾을 때 들여쓰기를 무시하고 else와 짝을 이루지 않는 상위의 if들 중 가장 가까운 if와 짝이 된다고 해석한다. 그러므로 2행과 3행의 두 if 중 위로 가장 가까운 3행의 if를 짝으로 해석한다. x에 입력된 값이 3이면 2행의 조건식이 참이므로 3행으로 실행 순서가 바뀌며, 3행의 조건식이 거짓이므로 5행으로 실행 순서가 바뀌고 6행을 실행하여 '음수입니다.'를 출력하게 된다. 이와 같이 프로그래머의 의도와 컴파일러의 해석 차이로 인한 혼란을 방지하려면 다음과 같이 { }를 사용하여 범위를 명확하게 표시해야 한다.

```
1  scanf("%d", &x);
2  if (x > 0)
3  {
4     if (x % 2 == 0)
5        printf("양의 짝수입니다.\n");
6  }
7  else
8     printf("음수입니다.\n");
```

위의 코드에서는 { }를 사용해 2행의 조건식이 참일 때 수행할 내용이 3~6행의 블록임을 명시했다. 그러므로 4행은 단순 if문으로 해석되고 7행의 else는 2행의 if와 짝을 이루는 것으로 해석한다.

[프로그램 5-4]는 입력받은 점수에 따른 학점을 출력한다. 입력한 점수가 90점 이상이면 A, 80점 이상 90점 미만이면 B, 70점 이상 80점 미만이면 C, 60점 이상 70점 미만이면 D, 60점 미만이면 F로 학점을 부여하여 출력한다.

프로그램 5-4 입력받은 점수의 학점 출력하기 (ch5-4.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int score;      // 점수를 저장할 변수
6       char grade;     // 학점에 해당하는 문자 1개를 저장할 변수
7
8       printf("점수를 입력하시오 : ");
9       scanf("%d", &score);
10
11      if (score >= 90)
12          grade = 'A';
13      else if (score >= 80)
14          grade = 'B';
15      else if (score >= 70)
16          grade = 'C';
17      else if (score >= 60)
18          grade = 'D';
19      else
20          grade = 'F';
21
22      printf("학점 : %c\n", grade);
23
24      return 0;
25  }
```

점수에 따라 A~F 중 한 문자를 grade에 저장

실행결과

```
점수를 입력하시오 : 78
학점 : C
```

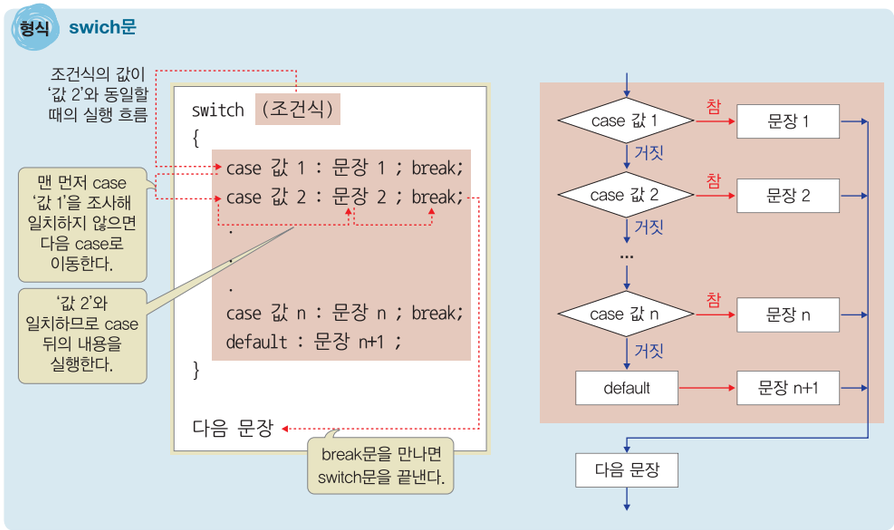
설명

11, 13, 15: 입력 값이 78이므로 11행의 조건을 만족하지 않아 13행으로 이동한다. 13행 조건 또한 만족하지 않으므로 15행으로 이동한다. 15행 조건을 만족하므로 종속 문장인 16행을 실행한 후 다음 문장인 22행을 실행한다.

5.1.2 switch문

if문은 조건식이 참이냐 거짓이냐에 따라 두 가지 경우 중 하나를 선택해 실행하므로 여러 조건 중 하나만 선택하는 경우에는 중첩된 if문을 사용하게 된다. 그러나 중첩 if문을 사용하면 프로그램이 복잡해져 이해하기 힘들고 실수로 문장 제어를 잘못하는 경우가 발생하기 쉽다. 이런 경우 switch문을 사용하면 중첩된 if문보다 단순 명료하게 작성해 조건에 따른 문장 제어를 보다 쉽게 처리할 수 있다.

switch문은 조건식 값에 따라 여러 경우 중 한 가지만 선택해 실행할 때 많이 사용되므로 다중 분기문이라고 부른다. switch문은 1개 이상의 case문과 default문으로 구성된다. 식의 결과 값이 무엇인가에 따라 각기 다른 case문으로 분기되며, 그것이 어떤 case에도 속하지 않으면 default로 분기된다. switch문의 형식과 실행의 흐름도는 다음과 같다.



조건식

조건식의 결과 값이 반드시 정수형이어야 한다. 논리식은 결과가 참일 때는 정수 1, 거짓일 때는 정수 0으로 변환되므로 조건식으로 사용할 수 있다. 그리고 문자는 문자에 해당하는 ASCII 코드 값이 정수형이므로 결과 값이 문자인 식도 사용할 수 있다.

case 뒤의 값

case문이 갖는 값은 반드시 정수형 상수 한 개만 가능하다. 그렇지 않으면 에러가 발생한다. (2+3)과 같은 식은 올 수 없다. 그러나 A, +와 같은 문자는 가능하다. 또한 1이거나 2이거

나 3인 경우를 표현한다고 case 1, 2, 3으로 작성하면 에러가 난다. 각 case문은 일반적으로 break문을 갖는다. 이는 switch문의 실행을 마치라는 의미다.

실행의 흐름

switch문은 조건식을 먼저 평가해 결과 값과 같은 값에 해당하는 case문을 찾아 그 뒤의 문장을 실행한다. 조건식 결과 값이 '값 2'와 같으면 'case 값 1' 뒤의 내용은 실행하지 않는다. 곧바로 'case 값 2'를 확인해 결과 값과 동일하면 뒤의 문장을 실행한다. 그리고 break문을 만나면 switch문에서 빠져나온다. break문이 다음 case와의 경계를 명확히 해주므로 해당 case에서 처리할 문장이 둘 이상이더라도 {}를 사용할 필요가 없다.

default

조건식을 계산한 후 case 뒤의 값과 비교해 일치하는 값을 못 찾으면 default 뒤의 내용을 실행한다. default는 switch문 안의 모든 위치에 사용할 수 있지만 보통 switch문의 맨 마지막에 작성하며, 필요 없으면 생략할 수 있다. 만약 default가 switch문의 case 레이블 구문들 중간에 위치하고 있다면 default 문을 수행한 다음 뒤의 case 값과 다시 비교를 진행한다.

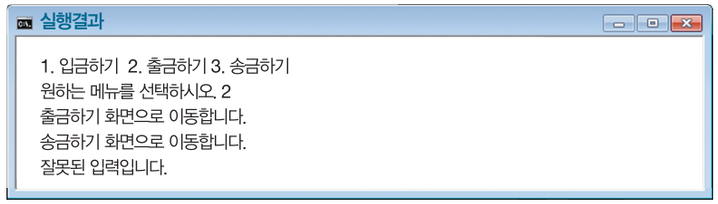
예
```
// x가 짝수인지 홀수인지 출력하기
switch (x % 2)
{
   case 0 : printf("짝수입니다.\n"); break;
   case 1 : printf("홀수입니다. \n");
}
```

예
```
// 결혼 여부를 묻고 기혼자 또는 미혼자 수를 1증가하기
switch (married)
{
   case 1 : printf("기혼자 \n"); married++; break;
   case 2 : printf("미혼자 \n"); single++; break;
   default : printf("잘못된 입력입니다.\n");
}
```

이제 switch문에서 break를 생략했을 때의 실행 순서를 살펴보자. switch문에서는 조건식의 결과 값과 case를 차례로 확인하되, case 뒤의 값과 결과 값이 일치하지 않으면 곧바로 다음 case로 이동한다. 그러나 결과 값과 일치하는 case를 찾은 후에는 break를 만나지 않는 한 그 뒤의 case에 있는 문장을 계속 실행한다. 그러므로 다음 switch문에서 break를 적절히 사용하지 않으면 원하지 않는 결과를 얻게 된다. 다음의 예를 실행하여 결과를 확인해보자.

예
```
1  printf("1. 입금하기  2. 출금하기 3. 송금하기 \n");
2  printf("원하는 메뉴를 선택하시오. \n");
3  scanf("%d", &answer);
4
5  switch (answer)
6  {
7     case 1 : printf("입금하기 화면으로 이동합니다.\n");
8     case 2 : printf("출금하기 화면으로 이동합니다.\n");
9     case 3 : printf("송금하기 화면으로 이동합니다.\n");
10    default : printf("잘못된 입력입니다.\n");
11 }
```

위의 예 3행에서 사용자가 2를 입력하면 8행이 실행된다. 이때 printf 함수 뒤에 break문이 없으므로 그 뒤의 case 값은 무시하고 다음 문장을 계속 실행한다. 결국 switch문을 빠져나오지 못하고 9행과 10행을 모두 실행한 후 switch문을 벗어나게 되어 다음과 같은 결과가 출력된다.



따라서 필요 없는 내용을 출력하지 않도록 7, 8, 9행에 break;를 추가해야 한다. switch문을 종료하려면 break문이 꼭 필요하지만, [프로그램 5-9]와 같이 break문을 사용하지 않음으로써 둘 이상의 조건식 값에 대해 동일한 처리를 할 수도 있다. 그러므로 해결할 문제에 따라 break의 필요성을 잘 분석하고 사용하도록 한다.

5.2 프로그래밍 실습

프로그래밍 실습

5.2.1 if~else문을 이용한 간단한 성적 처리 프로그램

문제 학생의 국어, 음악, 미술 점수를 입력받아 평균을 구한 후 평균이 70점 이상이면 '통과'를 70점 미만이면 '탈락'을 출력하시오.

해결 과정

1 점수 세 개를 kor, music, art에 입력받기
2 세 점수를 합한 후 3으로 나누어 avg에 평균을 구하기
3 학생의 점수 세 개와 평균을 출력하기
4 if~else문을 이용해 avg가 70점 이상인지 아닌지를 검사하여 통과 또는 탈락 출력하기

프로그램 5-5 세 과목의 평균을 구한 후 합격 여부를 출력하기 (ch5-5.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int kor, music, art;
6       double avg;
7
8       printf("국어 음악 미술 점수는?");
9       scanf("%d %d %d", &kor, &music, &art);
10
11      avg = (double)(kor + music + art)/3;
12
13      printf("\n \n======================== \n");
14      printf("\n \n국어 \t음악 \t미술 \t평균 \t결과\n");
15      printf("-------------------------- \n");
16      printf("%3d \t%3d \t%3d \t%6.2lf  ", kor, music, art, avg);
17      if (avg >= 70)
18          printf("통과\n");
19      else
20          printf("탈락\n");
21      printf("======================== \n");
22
23      return 0;
24  }
```

실행결과

```
국어 음악 미술 점수는?89 92 95
================================
국어   음악   미술   평균   결과
--------------------------------
 89     92     95   92.00  통과
================================
```

(평균이 70 이상이므로 '통과' 출력)

설명

16: \n을 사용하지 않음으로서 뒤의 통과 또는 탈락을 한 행에 출력할 수 있다.

17~20: 평균이 70 이상인지 아닌지에 따라 통과 또는 탈락 중 한 개를 선택해서 출력한다.

5.2.2 중첩 if문을 이용한 간단한 산술 계산기

문제 사용자가 입력한 수식 '3 * 25'를 계산해주는 간단한 산술 계산기를 if~else문을 이용해 작성하시오. 키보드로 int형 피연산자 두 개와 char형 연산자 한 개를 입력받는다. 연산자의 종류는 8가지(+, -, *, /, %, &, |, ^)로 한정하고, 그 외의 연산자는 잘못된 입력으로 간주하여 간단한 안내문을 출력한다.

분석 • 8가지 연산자에 따라 계산 방법이 달라진다. 즉 8가지 계산 방법 중 하나를 선택해야 하므로 중첩된 if문을 사용한다.

프로그램 5-6 if~else문을 이용한 산술 계산기 (ch5-6.cpp)

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main()
5   {
6       int x, y, result;
7       char op;
8
9       printf("수식을 입력하시오.\n");
10      printf("연산자의 종류 : + - * / %% & ¦ ^\n");
11      printf("입력 예 : 2 + 3\n\n입력 >> ");
12      scanf("%d %c %d", &x, &op, &y);
13
14      // op에 저장된 연산자에 따라 해당 연산을 적용한 결과를 result에 저장하기
15      if (op == '+')
16          result = x + y;
17      else if (op == '-')
18          result = x - y;
19      else if (op == '*')
20          result = x * y;
21      else if (op == '/')
22          result = x / y;
23      else if (op == '%')
24          result = x % y;
25      else if (op == '&')
26          result = x & y;
27      else if (op == '¦')
28          result = x ¦ y;
29      else if (op == '^')
30          result = x ^ y;
31      else
32      {
33          printf("잘못된 연산자입니다.\n");
```

```
34          exit(0); // 프로그램의 실행을 끝낸다.
35      }
36
37      printf("\n결과 >> %d %c %d = %d\n", x, op, y, result);
38
39      return 0;
40  }
```

실행결과

```
수식을 입력하시오.
연산자의 종류 : + - * / % & | ^
입력 예 : 2 + 3

입력 >> 10 % 3
결과 >> 10 % 3 = 1
```

12행 scanf 변환명세 %d와 %c 사이에 빈 칸이 있으므로 입력할 때 10과 % 사이에 반드시 빈 칸을 입력해야 한다.

설명

34: exit 함수를 호출하면 프로그램이 끝난다. 이 함수를 이용하려면 #include 〈stdlib.h〉가 필요하다.

5.2.3 중첩 if문을 이용한 가산점 계산

문제 자신의 가산점을 확인할 수 있는 프로그램을 작성하시오. 가산점을 부여하는 규칙은 다음과 같다.

- **여자의 가산점 부여 규칙**
  기혼자라면 +1, 기혼자로서 자녀수가 1이면 +1, 기혼자로서 자녀수가 2 이상이면 +2
- **남자의 가산점 부여 규칙**
  군 제대자라면 +1, 군 제대자로서 기혼자라면 +1

분석

- 성별에 따라 가산점 부여 규칙이 다르므로 크게 남자와 여자로 구분하여 가산점을 계산하도록 if~else문을 사용한다.
- 성별이 같아도 미혼인지 기혼인지에 따라, 기혼자는 자녀 수가 몇 명인지에 따라 가산점 여부가 달라진다. 따라서 하나의 질문에 대한 답에 따라 다른 질문을 하도록 중첩 if문을 사용한다.

해결 과정

1 남자인지 여자인지 묻기
2 if (남자)

```
{  // 군 제대자인지 묻기
   if (군 제대자)
      가산점 1 추가, 기혼인지 묻기, 기혼 여부에 따라 가산점 추가하기
}
else
{  // 기혼인지 묻기
   if (기혼)
      가산점을 1 추가, 자녀 수 묻기, 자녀 수에 따라 가산점 추가
}
```

프로그램 5-7 질문의 답에 따라 해당 가산점 계산하기 (ch5-7.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int gender, married, army, plus, children;
6
7       plus = 0; // 가산점을 먼저 0으로 초기화
8
9       printf("성별 (남:1, 여:2) ? "); // 성별을 묻기
10      scanf("%d", &gender);
11
12      if (gender == 1) // 남자인 경우의 처리 부분
13      {
14          printf("군 제대 (예:1, 아니오:2) ? ");      // 군 제대 여부를 묻기
15          scanf("%d", &army);
16
17          if (army == 1)
18          {
19              plus++; // 군 제대자라면 +1
20
21              printf("결혼 (예:1, 아니오:2) ? ");     // 제대자라면 결혼 여부를 묻기
22              scanf("%d", &married);
23              if (married == 1)
24              {
25                  plus++;     // 군 제대자면서 기혼자라면 +1
26              }
27          }
28      }
29      else // 여자인 경우의 처리 부분
```

```
{
    printf("결혼 (예:1, 아니오:2) ? ");          // 결혼 여부를 묻기
    scanf("%d", &married);

    if (married == 1)
    {
        plus++;         // 기혼자라면 +1

        printf("자녀수는? ");
        scanf("%d", &children);
        if (children == 1)
        {
            plus++;     // 자녀가 1명이라면 +1
        }
        else if (children >= 2)
        {
            plus += 2; // 자녀가 둘 이상이라면 +2
        }
    }
}

    printf("\n>> 총 가산점은 %d점입니다.", plus);

    return 0;
}
```

실행결과

```
성별 (남:1, 여:2) ? 2
결혼 (예:1, 아니오:2) ? 1   <-- +1
자녀수는? 3                 <-- +2

>> 총 가산점은 3점입니다.
```

5.2.4 간단한 산술 계산기인 [프로그램 5-6]을 switch문으로 수정하기

프로그램 5-8 switch문을 이용한 산술 계산기 (ch5-8.cpp)

```
#include <stdio.h>

int main()
{
    int x, y, result;
    char op;

    printf("수식을 입력하시오.\n");
    printf("연산자의 종류 : + - * / %% & ¦ ^\n");
    printf("입력 예 : 2 + 3\n\n입력 >> ");
    scanf("%d %c %d", &x, &op, &y);

    switch(op)
    {
        case '+' : result = x + y; break;
        case '-' : result = x - y; break;
        case '*' : result = x * y; break;
        case '/' : result = x / y; break;
        case '%' : result = x % y; break;
        case '&' : result = x & y; break;
        case '|' : result = x | y; break;
        case '^' : result = x ^ y; break;
        default : printf("잘못된 연산자입니다.\n"); exit(0);
    }
    printf("\n결과 >> %d %c %d = %d\n", x, op, y, result);

    return 0;
}
```

실행결과

```
수식을 입력하시오.
연산자의 종류 : + - * / % & | ^
입력 예 : 2 + 3

입력 >> 5 & 3        비트 AND 연산자
결과 >> 5 & 3 = 1
```

5.2.5 학점을 구하는 [프로그램 5-4]를 switch문으로 수정하기

프로그램 5-9 입력받은 점수의 학점 출력하기 : switch문 이용 (ch5-9.cpp)

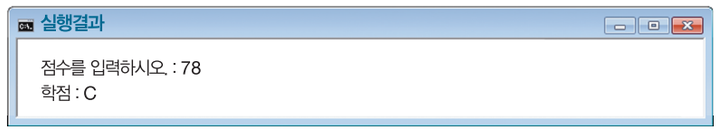
```
#include <stdio.h>

int main()
{
    int score;
    char grade;

    printf("점수를 입력하시오. : ");
    scanf("%d", &score);

```

```
11      switch (score / 10)
12      {
13          case 10 :
14          case 9 : grade = 'A'; break;
15          case 8 : grade = 'B'; break;
16          case 7 : grade = 'C'; break;
17          case 6 : grade = 'D'; break;
18          default : grade = 'F';
19      }
26      printf("학점 : %c\n", grade);
20
21      return 0;
22  }
```



설명

11: 입력받은 점수 score는 0~100 사이의 값이다. '정수/정수'의 결과는 정수이므로 나눗셈의 몫이 된다. 그러므로 조건식으로 (score / 10)을 사용하면 score가 0~9일 때 0, 10~19일 때 1, 20~29일 때 2, …, 90~99일 때 9, 100일 때 10이 되므로 점수대별로 그룹화할 수 있다.

13~14: 입력받은 점수가 100점인 경우 조건식 (score / 10)의 결과가 10이므로 90점대의 점수와는 별도로 case를 지정해줘야 한다.

```
13      case 10 : grade = 'A'; break;
14      case 9 : grade = 'A'; break;
```

그러나 [프로그램 5-9]와 같이 13행에서 'case 10:' 뒤에 아무것도 적지 않으면 조건식 값이 10일 때 13행으로 이동한 후 실행할 문장이 없으므로 다음 14행의 내용을 실행한다. 즉 조건식 결과 값이 10이거나 9일 때 동일하게 처리한 것이다. 예를 들어 조건식 결과 값이 1이거나 2이거나 3일 때 동일한 처리를 하고 싶다면 다음과 같이 작성한다.

```
case 1 :
case 2 :
case 3 :  // 1 또는 2 또는 3일 때 처리할 내용을 작성
      break;
```

18: 0~59까지는 (score / 10)이 0~5 중 하나이므로 case 값을 만족하는 것이 없어 default에서 처리된다.

5.3 반복문(for문)

반복문은 동일한 내용을 특정 횟수만큼 반복 처리할 때 사용되며, 루프(loop)라고도 한다. 일반적으로 프로그램은 방대한 양의 데이터로부터 원하는 정보를 얻기 위해 사용되므로 많은 데이터에 동일한 처리를 반복하는 경우가 많다. 데이터 하나를 처리하는 코드가 10행일 때 데이터 1,000개를 1,000*10행의 코드로 처리한다고 상상해보자. 프로그램 길이도 문제지만 프로그램을 수정하는 것도 문제일 것이다.

주어진 문제를 분석하고 해결할 때 프로그램을 반복 구조로 작성하면, 일정 코드를 지정한 횟수만큼 반복 실행하므로 프로그램이 단순하고 명확해지며 유지 관리가 쉬워진다. 우선 반복문의 필요성과 컴퓨터 프로그램에서 주어진 문제를 반복문 형태의 구조로 변환하여 해결하는 방법부터 살펴보자.

5.3.1 반복분의 필요성

for문의 실제 사용법을 익히기에 앞서 for문을 사용하면 얼마나 편리한지부터 알아보자. 'C Language'를 5행 출력하고 싶다면 왼쪽 코드처럼 printf 함수를 5번 반복해야 한다. 하지만 오른쪽 코드처럼 for문을 이용하면 코드 길이를 늘이지 않고 쉽게 수정할 수 있다.

예
```
printf("C Language \n");
printf("C Language \n");
printf("C Language \n");
printf("C Language \n");
printf("C Language \n");
```

```
for (i=1; i<=5; i++)
   printf("C Language \n");
```

예를 하나 더 살펴보자. 1번에서 5번까지의 번호를 한 행씩 출력하고 싶다면 왼쪽 코드처럼 printf 함수를 5번 반복하면 된다. 그런데 1번에서 100번까지 출력하고 싶다면? for문은 5를 100으로 고치기만하면 쉽게 수정할 수 있다. 하지만 왼쪽과 같이 같은 문장을 차례대로 직접 적은 경우에는 직접 '번' 앞의 숫자가 1~100이 되도록 100행의 문장을 작성해야 한다.

예
```
printf("1번 \n");
printf("2번 \n");
printf("3번 \n");
printf("4번 \n");
printf("5번 \n");
```

```
for (i=1; i<=5; i++)
   printf("%d번 \n", i);
```

1에서 5까지의 합 sum을 구하고 싶으면 왼쪽 코드처럼 대입문을 이용할 수 있다. 그런데 1에서 100까지의 합을 구하고 싶다면? sum에 대입할 값을 구하는 식에서 일일이 1에서 100까지 더하도록 수정해야 한다. 반면 for문은 5를 100으로 수정하기만 하면 된다.

예
```
sum = 1 + 2 + 3 + 4 + 5;
printf("1~5까지의 합은 %d \n", sum);
```

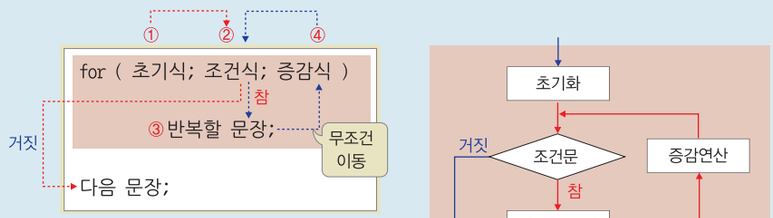
```
sum = 0;
for (i=1; i<=5; i++)
   sum = sum + i;
printf("1~5까지의 합은 %d \n", sum);
```

컴퓨터 프로그램을 효율적으로 작성하기 위해서는 주어진 문제를 프로그램으로 해결할 수 있는 형태로 변환할 수 있어야 한다. 특히 반복문 형태로 해결할 수 있는지를 살펴보고, 어떤 내용을 반복해야 문제를 해결할 수 있는지를 찾아내는 것이 중요하다.

5.3.2 for문 형식

반복문을 작성할 때는 먼저 어떤 내용을 반복해야 하는지부터 분석해야 한다. for문 형식과 실행의 흐름도는 다음과 같다.

형식 for문 형식



for문의 실행 흐름

①초기식을 실행한다.

②조건식을 평가한다.

③조건식이 참이면 반복할 문장을 실행하고, 거짓이면 for문을 벗어난다.

④반복할 문장을 실행한 후에는 증감식으로 돌아가서 실행한 후 다시 ②를 실행한다.

예
```
for (초기식; 조건식; 증감식)
{
   반복할 문장들;
}
```
← for문 헤더
for문 본체
반복할 내용이 두 문장 이상이라면 반드시 { }로 묶어야 한다.

for문 헤더(header)

for문의 첫 행을 헤더라고 하는데 이 헤더는 세 부분으로 나뉘며 반드시 ;으로 구분해야 한다.

- 초기식: 반복 루프가 시작할 때 한번만 실행되는 곳으로 주로 루프의 반복을 제어하는 제어변수를 초기화하는 용도로 사용된다.
- 조건식: 루프의 반복 내용을 실행할지 아니면 반복을 중단하고 for문을 끝낼 지를 결정하는데 사용된다. 조건식의 결과 값이 참이라면 for문의 본체(body)를 실행하고, 거짓이면 본체를 실행하지 않는다.
- 증감식: 주로 제어 변수를 1 증가하거나 1 감소하는 데 사용되며 복합 대입 연산자를 사용하여 특정 값만큼 증감 연산을 수행한다. for문의 본체를 실행하면 실행 순서가 증감식으로 이동한다. 증감식을 실행한 후에는 언제나 조건식 부분으로 이동한다. 사실 조건식은 이 증감식의 실행으로 제어 변수의 값이 변경되어 조건식의 결과 값이 거짓이 되면서 반복이 끝난다.

for문 본체(body)

반복할 내용을 적는 곳으로 한 문장이라면 { }를 사용할 필요가 없으나 두 문장 이상이라면 반드시 { }로 묶어야 한다. 본체의 마지막 문장을 실행하면 언제나 헤더의 증감식으로 돌아간다는 것에 주의한다.

다음의 네 가지 for문의 예에 대한 자세한 내용은 다음 절에서 소개한다.

예
```
// 'C Language'를 5행 출력하기
for (i=1; i<=5; i++)
printf("C Language \n");
```

예
```
// 1번~5번까지 출력하기
for (i=1; i<=5; i++)
printf("%d번 \n", i);
```

예
```
// 1에서 5까지의 합 출력하기
sum = 0;
for (i=1; i<=5; i++)
sum += i;
printf("1~5까지의 합은 %d \n", sum);
```

예
```
// 5! 출력하기
factorial = 1;
for (i=1; i<=5; i++)
   factorial *= i;
printf("5!은 %d \n", factorial);
```

• for문 헤더 바로 뒤에 ;을 쓰면 잘못된 결과를 얻을 수 있으므로 주의해야 한다. 이때 처리 문장 없이 세미콜론(;)으로만 구성된 문장을 널 문장(null statement)이라 한다.

```
for (i=1; i<=5; i++) ;
    printf("C Language \n");
```

for문에서 반복할 문장을 생략한 경우로 간주하므로 for문 헤더만 5번 반복한다.

• for문의 헤더에서 초기식, 조건식, 증감식을 구분하기 위해 ;을 써야 하는데 ,를 쓰면 에러가 된다.

```
for (i=1, i<=5, i++)
```

for문 실행 과정

이제 for문이 어떻게 동일한 내용을 원하는 횟수만큼 반복하는지 살펴보자. [표 5-1]은 for문이 실행되는 과정을 자세히 보여준다.

for문의 반복 횟수를 조절하는 변수로, 제어 변수라고 한다.

```
for (i=1; i<=5; i++)
  printf("1번 \n");
```

표 5-1 for문 실행 과정

i의 값	조건식 i<=5	반복할 문장
1	참	printf("1번 \n");
2	참	printf("1번 \n");
3	참	printf("1번 \n");
4	참	printf("1번 \n");
5	참	printf("1번 \n");
6	거짓	for문을 끝냄

[표 5-1]에서 알 수 있듯이 for문은 헤더에 포함된 변수 i를 활용해 본체의 반복 횟수를 지정한다. for문의 본체 반복 횟수는 for문 헤더에 있는 i를 1부터 5까지 1씩 증가한 5번이라고 생각하면 된다. 그러므로 100번 반복하고 싶으면 조건문을 i<=5 대신 i<=100으로 수정한다. 이와 같이 헤더에 포함된 변수 i는 for문의 반복을 제어하는 데 사용된다고 하여 제어 변수라고 한다.

특히 for문과 다음에 소개하는 while문, do~while문 같은 반복문은 대부분 본체 안에서 제어 변수를 활용해 문제를 해결한다. 다음 for문은 2행을 5번 반복하므로 단순히 '1번'을 5행 출력한다. 그런데 '1번'을 5행 출력하는 것이 아니라 '1번'에서 '5번'까지 차례대로 출력하려면 어떻게 해야 할까?


```
for (i=1; i<=5; i++)
  printf("1번 \n");
```

실행결과

```
1번   i=1일 때
1번   i=2일 때
1번   i=3일 때
1번   i=4일 때
1번   i=5일 때
      i=6일 때 끝
```

이럴 때는 다음과 같이 제어 변수를 활용해야 한다. 아래 코드는 for문의 제어 변수를 활용해 '1번'에서 '5번'까지 차례대로 출력하였다. 제어 변수 i가 1, 2, 3 순으로 변하며, 출력 결과도 '1번', '2번', '3번'처럼 '번' 앞의 숫자가 1, 2, 3 순으로 변한다는 것을 알 수 있다.

```
for (i=1; i<=5; i++)
  printf("%d번 \n", i);
```

제어 변수 i를 본체에 활용함으로써 2행의 문장은 동일하지만 내용은 다르게 출력할 수 있다.

실행결과

```
1번   i=1일 때
2번   i=2일 때
3번   i=3일 때
4번   i=4일 때
5번   i=5일 때
      i=6일 때 끝
```

이제 마지막으로 1에서 5까지의 합을 구하는 for문을 분석해보자. 합을 구하는 과정을 어떻게 반복문으로 표현할 수 있을까? 컴퓨터 프로그램에서는 1~5의 합을 sum에 저장하는 대입문 sum = 1+2+3+4+5;를 아래 코드의 1행~6행처럼 결과가 저장될 변수를 초기화한 후 변수 값을 계속 변경하는 과정을 반복하는 것으로 해석한다. 그렇지만 2행~6행의 대입문은 형태는 같아도 sum에 더하는 값이 1에서 5까지 바뀌므로 for문 본체로 곧바로 표현할 수 없다.

결과를 저장할 변수를 0으로 초기화

```
1 sum = 0;          ← 0
2 sum += 1;         ← 0+1
3 sum += 2;         ← 0+1+2
4 sum += 3;         ← 0+1+2+3
5 sum += 4;         ← 0+1+2+3+4
6 sum += 5;         ← 0+1+2+3+4+5
7
8 printf("결과: %d", sum);
```

실행결과

```
결과: 15
```

위 코드를 아래의 왼쪽 코드와 같이 변경하면 'sum += i;' 문장을 5번 반복한다. 따라서 for문을 오른쪽 코드와 같이 작성할 수 있다. 위에서 for문의 제어 변수를 활용해 계속 변하는 내용을 동일한 문장으로 만든 것처럼 오른쪽 코드 3행도 위 코드의 2행~6행의 대입문을 'sum += i;'와 같이 동일한 대입문으로 만들 수 있다.

```
sum = 0;
i = 1;
sum += i;
i++;
sum += i;
i++;
sum += i;
i++;
sum += i;
i++;
sum += i;

printf("결과: %d", sum);
```

동일한 문장을 5번 반복한다.

```
1 sum = 0 ;
2 for (i=1; i<=10; i++)
3   sum += i;
4
5 printf("결과: %d", sum);
```

한 번만

다양한 for문의 헤더

for문의 헤더는 다양한 형태로 사용할 수 있다. 다음과 같이 초기식에서 콤마 연산자를 이용해 여러 문장을 사용할 수 있다.

```
// 1에서 100까지의 합 구하기
for (sum=0, i=1; i<=100; i++)
   sum += i;

// 10명의 점수를 입력받아 평균 구하기
for (printf("데이터 입력하기 \n"), sum=0, i=1; i<=10 ; i++)
{
   for (sum=0, i=1; i<=100; i++)
   {
      printf("%d번의 점수는?", i);
      scanf("%d", &score);
      sum += score;
   }
}
```

for문의 조건식에서는 논리 연산자를 이용해 여러 조건을 결합할 수도 있다.

```
// 정수를 최대 10개 입력받아서 합을 구하되 합이 500보다 커지면 중지하기
for (sum=0, i=1; i<=10 && sum<=500; i++)
{
   printf("%d번째 정수는?", i);
   scanf("%d", &n);
   sum += n;
}
```


for문의 헤더에 포함된 초기식, 조건식, 증감식은 모두 생략할 수 있다. 그러나 세 부분을 구분하는 ;은 생략하면 안 된다.

```
for ( ; ; ) { … }
```

조건문을 생략하면 무조건 for문의 본체를 실행한다.

특히 조건식이 생략된 경우에는 무조건 본체를 실행하게 되므로 무한 반복이 되는데 이러한 반복문을 무한 루프라고 한다. 실제로는 무한 루프가 되면 프로그램이 끝나지 않으므로 본체 안에서 for문을 탈출하는 방법이 반드시 포함되어야 한다. 그래서 일반적으로 다음과 같이 특정 조건을 만족하면 for문을 벗어나도록 하는 break를 많이 사용한다.

```
// 1 + 3 + 5 + … + n이 처음으로 1000을 넘는 n 찾기
sum = 0;
for (i=1; ; i=i+2)
{
   sum += i;
   if (sum > 1000)
      break;
}
printf("1부터 %d까지의 홀수의 합은 %d", i, sum);
```

다음에 3을 더해야 하므로 i 값을 2씩 증가시킨다.

현재 sum의 값이 1000보다 커지면 반복을 끝내야 한다.

break문을 만나 for문을 빠져 나온다.

5.3.3 중첩된 for문

중첩된 for문의 실행

반복문 안에 또 다른 반복문이 포함된 것을 중첩 반복문이라 한다. 중첩 반복문에서 주의할 점은 각 반복문의 제어 변수가 달라야 한다는 것이다. 다음은 이중으로 중첩된 for문의 예와 실행결과다.

예

```
1 for (i=1; i<=2; i++)
2 {
3    printf("i=%d일 때 : ", i);
4
5    for (j=1; j<=3; j++)
6       printf("j=%d ", j);
7
8    printf("\n");
9 }
```

바깥 for문 본체의 마지막 문장을 실행한 후 바깥 for문의 헤더의 증감식으로 돌아간다. 조건문이 거짓이면 바깥 for문을 끝내므로 실행 순서가 9행으로 바뀐다.

안쪽 for문 본체의 마지막 문장을 실행한 후에 안쪽 for문의 헤더의 증감식으로 돌아간다. 조건문이 거짓이면 안쪽 for문을 끝내므로 실행 순서가 7행으로 바뀐다.

```
실행결과
i=1일 때 : j=1 j=2 j=3 ↵
i=2일 때 : j=1 j=2 j=3
```


실행 창을 보면 알 수 있듯이 바깥 for문의 본체가 한 번 실행되는 동안 안쪽 for문이 완전 실행되므로 안쪽 for문의 본체가 3번 반복 실행된다. 그래서 안쪽에 있는 for문일수록 훨씬 더 많이 반복된다. 이와 같이 중첩 for문을 이용하면 맨 안쪽의 본체, 즉 6행을 총 2*3번 반복한다.

중첩된 for문에서 제어 변수의 활용

중첩된 for문에서 안쪽으로 중첩되는 for문들은 자신보다 밖에 있는 for문의 제어 변수들을 많이 활용한다. 다양한 예를 프로그램을 통해 살펴보자.

[프로그램 5-10]은 '*****'을 5행 출력하기 위해 '*'을 5개 출력하는 for문을 5번 반복하도록 중첩 for문을 사용한다. 만일 1행에 '*'을 한 개, 2행에 '*'을 두 개, n행에는 '*'을 n개 출력하고 싶으면 어떻게 해야 할까? [프로그램 5-11]과 같이 바깥 for문의 제어 변수를 안쪽 for문에서 활용하면 된다.

프로그램 5-10 '*****'를 5행 출력하는 중첩 for문 (ch5-10.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int line, star
6
7       for (line=1; line<=5; line++)
8       {
9           for (star=1; star<=5; star++)
10              printf("*");
11          printf("\n");   // '*****' 출력 후 행 바꾸기
12      }
13
14      return 0;
15  }
```



```
실행결과
*****
*****
*****
*****
*****
```

[프로그램 5-10]의 실행 결과에서 알 수 있듯이 바깥 for문의 제어 변수 line의 값이 1, 2, 3 순으로 바뀌므로 line 값을 출력할 '*' 개수로 지정하면 n행에 n개의 '*'를 n개 출력할 수 있다. [프로그램 5-11]은 이렇게 바깥 for문의 제어 변수를 안쪽 for문의 반복 횟수로 활용해 완성한 프로그램이다. 앞으로 소개할 정렬 프로그램 등 많은 프로그램에서 중첩 반복문으로 해결할 때 각 반복문의 제어 변수를 적절히 활용하는 것을 볼 수 있을 것이다.

프로그램 5-11 중첩 for문과 제어 변수를 활용해 n행에 *을 n개 출력하기 (ch5-11.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int line, star;
6
7       for (line=1; line<=5; line++)
8       {
9           for (star=1; star<=line; star++)
10              printf("*");
11          printf("\n");
12      }
13
14      return 0;
15  }
```

```
실행결과
*
**
***
****
*****
```

설명

9: for문의 반복 횟수를 5로 고정하지 않고 바깥 for문의 제어 변수 line으로 지정함으로써 바깥 for문의 본체가 실행될 때마다 출력하는 '*'의 개수가 한 개씩 늘어난다.

[프로그램 5-12]는 1행에는 1에서 1까지의 합을, 2행에는 1에서 2까지의 합을, n행에는 1에서 n까지의 합을 차례대로 출력한다. 1에서 n까지의 합은 for문으로 작성할 수 있으며, 이러한 합을 n이 1일 때부터 n까지 반복하므로 중첩 for문으로 작성한다.

프로그램 5-12 1~n의 합을 n이 1일 때부터 10일 때까지 구하기 (ch5-12.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int n, i, sum;
6
7       for (n=1; n<=10; n++)
8       {
9           sum=0;
10          for (i=1; i<=n; i++)
11              sum = sum + i;
12
13          printf("1~%d까지의 합 : %d \n", n, sum);
14      }
15
16      return 0;
17  }
```

실행결과

```
1~1까지의 합 : 1     <-- sum=0+1
1~2까지의 합 : 3     <-- sum=0+1+2
1~3까지의 합 : 6     <-- sum=0+1+2+3
1~4까지의 합 : 10    <-- sum=0+1+2+3+4
⋮                    ⋮
1~10까지의 합 : 55   <-- sum=0+1+2+3+4+ … + 10
```

설명

9: 9행을 6행으로 이동한다면 10행의 for문을 실행함으로써 저장된 sum 값에 다시 i 값을 더하게 된다. 그러므로 안쪽 for문을 시작하기 전에 반드시 sum을 다시 0으로 초기화하는 과정이 필요하다.

5.4 프로그래밍 실습

5.4.1 1~n의 합과 곱 구하기

문제 키보드로부터 정수 n을 입력받고 정수 1부터 n까지의 합과 곱을 구하시오.

분석

- 1부터 n까지의 합 sum은 1부터 10까지 구한 for에서 조건식만 i<=n으로 수정하면 된다.

```
for (i=1; i<=n; i++)
    sum += i;
```

- 1부터 n까지의 곱은 대입문 factorial = 1*2*3* … *n;으로 구할 수 있다. 그러므로 이 대입문은 factorial을 초기화한 후 1부터 n까지의 정수를 factorial에 곱하는 것(factorial = factorial*i;)을 반복하면 된다. 여기서 주의할 것은 factorial의 초깃값이다. 'factorial=0;'으로 초기화하면 이후에 factorial에 어떤 수를 곱해도 결과는 0이 된다. 그렇다면 factorial의 초깃값은 얼마여야 할까?

  덧셈을 할 때 sum을 0으로 초기화하는 것은 덧셈에 영향을 미치지 않는 수가 0이기 때문이다. sum의 초깃값이 0이므로 for문에서 처음으로 'sum += i;'를 실행할 때 sum에 (0 + 1)이 대입될 수 있는 것이다. 그러므로 특정 변수에 어떤 수를 계속 더해갈 때는 초깃값이 0이지만, 곱셈을 해야 한다면 곱셈에 영향을 미치지 않는 값인 1이 되어야 한다. 초깃값은 항상 0이나 1인 것은 아니며 문제에 따라 다양한 값이 될 수 있다.

프로그램 5-13 정수 1~n의 합과 곱 구하기 : for문 이용 (ch5-13.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int i, n, sum, factorial;
6
7       printf("1부터 n까지의 합과 곱을 구합니다. 정수 n은? ");
8       scanf("%d", &n);
9
10      // 1에서 n까지의 합을 sum에 구하기
11      sum = 0;
12      for (i=1; i<=n; i++)
13          sum = sum + i;
14
15      // 1에서 n까지의 곱 n!을 factorial에 구하기
```

```
16      factorial = 1;
17      for (i=1; i<=n; i++)
18          factorial = factorial * i;
19
20      printf("1부터 %d까지의 합은 %d입니다.\n", n, sum);
21      printf("1부터 %d까지의 곱은 %d입니다.\n", n, factorial);
22
23      return 0;
24  }
```

실행결과

```
1부터 n까지의 합과 곱을 구합니다. 정수 n은? 10
1부터 10까지의 합은 55입니다.
1부터 10까지의 곱은 3628800입니다.
```

설명

11: 누적용 변수 초기화는 가능하면 해당 반복문 직전에 두는 것이 좋다. for문의 sum에 대한 초기화가 제대로 되었는지 확인하기가 편리하기 때문이다. 또한 이 for문을 다른 곳에 가져갈 때 초기화와 for문이 함께 있으면 11행~13행을 한번에 복사할 수 있어 코드 재사용 면에서도 편리하다.

16: factorial은 19행에서 현재 자신의 값과 i를 곱한 값으로 수정된다. 그러므로 0으로 초기화하면 언제나 결과가 0이 된다.

5.4.2 수열의 합 구하기

문제 $sum = \sum_{i=2}^{n}\left(\frac{i-1}{i}\right)$을 구하시오.

분석 • 분모가 i라고 할 때 $\frac{i-1}{i}$이 sum에 계속 더해진다. 그러므로 분모 i가 2부터 n이 될 때까지 $sum = \frac{1}{2}+\frac{2}{3}+\frac{3}{4}+\cdots+\frac{n-1}{n}$을 반복하면 최종 결과를 sum에 구할 수 있다.

프로그램 5-14 수열의 합 구하기 (ch5-14.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int i, n;
6       double sum;
7
8       printf("임의의 정수 n을 입력하시오. : ");
9       scanf("%d", &n);
10
11      sum = 0;
12      for (i=2; i<=n; i++)    // 분모는 2부터 n까지 변한다.
13          sum = sum + ((double)(i-1) / i);
14
15      // 수식 출력하기
16      for (i=2; i<=n; i++)
17          printf("%d/%d + ", i-1, i);
18      printf("\b\b= %.1lf \n", n, sum);
19
20      return 0;
25  }
```

실행결과

```
임의의 정수 n을 입력하시오. : 10
1/2 + 2/3 + 3/4 + 4/5 + 5/6 + 6/7 + 7/8 + 8/9 + 9/10 = 7.1
```

설명

13: 정수/정수의 결과는 정수형이므로 0이 되지 않게 (i-1)을 double형으로 형 넓힘 변환(캐스팅)한다.

18: 16행에서 i가 n일 때 출력된 '+'를 지우기 위해 백스페이스 이스케이프 문자를 출력한 뒤 다음 내용을 출력한다.

5.4.3 for문을 이용해 퀴즈 결과 출력하기

문제 학생 10명의 퀴즈 점수를 입력받은 후, 전체 평균과 통과자수, 탈락자수를 구해 출력하시오. 점수가 70점 이상이면 합격이다.

분석 • 학생 1명의 퀴즈 점수마다 처리할 일은 다음과 같다.

① 퀴즈 점수 입력받기

② 입력된 점수를 sum에 더하기

③ 점수가 70점 이상이면 통과자수를 1 증가시키고 그렇지 않으면 탈락자수를 1 증가시키기

• 학생 10명에 대해 동일 작업을 반복해야 하므로 for문으로 작성한다. for문에서 학생 10명의 퀴즈 점수 합계가 sum에 구해지므로 for문을 끝낸 후 평균을 구한다.

프로그램 5-15 퀴즈의 평균과 통과자수, 탈락자수 구하기 (ch5-15.cpp)

```
#include <stdio.h>

int main()
{
    int i, quiz, sum, pass, fail;
    double avg;

    // 누적용 변수를 0으로 초기화
    sum=0;
    pass=0;
    fail=0;

    for (i=1; i<=10; i++)
    {
        printf("%d번의 퀴즈 점수는? ", i);
        scanf("%d", &quiz);     // i번 학생의 퀴즈 점수 입력

        // 평균을 구하기 위해 점수를 sum에 누적하기
        sum += quiz;

        if (quiz >= 70)
            pass++;    // 70점 이상이면 통과
        else
            fail++;    // 70점 미만이면 탈락
    }

    avg = (double)sum / 10;

    printf("========================\n");
    printf("평균:%.2lf점\n", avg);
    printf("통과자:%2d명\n", pass);
    printf("탈락자:%2d명\n", fail);

    return 0;
}
```

실행결과

```
1번의 퀴즈 점수는? 90
2번의 퀴즈 점수는? 78
3번의 퀴즈 점수는? 86
4번의 퀴즈 점수는? 98
5번의 퀴즈 점수는? 76
6번의 퀴즈 점수는? 62
7번의 퀴즈 점수는? 88
8번의 퀴즈 점수는? 95
9번의 퀴즈 점수는? 80
10번의 퀴즈 점수는? 58
==================
평균:81.10점
통과자: 8명
탈락자: 2명
```

설명

18~24: for문에서 i가 증가되어 실행이 반복되면 quiz 값이 다음 학생의 점수로 바뀌게 되므로 다시 반복하기 전에 필요한 처리를 해둔다.

5.4.4 사각형을 반으로 접을 때마다 변하는 넓이를 for문을 이용해 출력하기

문제 넓이가 100cm^{2}인 사각형 종이를 반으로 접을 때 최초로 1cm^{2}보다 작아지는 횟수를 확인할 수 있도록 넓이 변화를 출력하시오.

분석

- 사각형 종이의 현재 넓이가 area이면 종이를 반으로 접으면 넓이가 1/2로 줄어든다. 그러므로 반 접었을 때의 넓이는 'area = area / 2;'다.
- 반으로 접은 후의 넓이가 1보다 작아졌을 때 그만 접어야 하므로 for문을 몇 번 반복해야 하는지 몰라 조건식을 미리 결정할 수 없다. 사실은 이 반복 횟수를 찾는 것이 이 프로그램의 문제다. 그러므로 for문의 헤더에서 조건식을 생략한다. 대신 문제 해결 조건을 만족하면 반복을 그만하도록 지정한다.

프로그램 5-16 사각형을 반으로 접을 때마다 변하는 넓이 구하기 (ch5-16.cpp)

```
#include <stdio.h>

int main()
{
    int i;
    double area = 100, target = 1;    // 현재 사각형의 면적, 최저 면적

    for (i=1; ;i++)
    {
```

```
10          area = area / 2;
11          printf("%2d번 접었을 때의 넓이 : %6.2lf \n", i, area);
12          if (area < target)
13              break;
14      }
15      return 0;
16  }
```

실행결과

```
1번 접었을 때의 넓이 : 50.00
2번 접었을 때의 넓이 : 25.00
3번 접었을 때의 넓이 : 12.50
4번 접었을 때의 넓이 :  6.25
5번 접었을 때의 넓이 :  3.13
6번 접었을 때의 넓이 :  1.56
7번 접었을 때의 넓이 :  0.78
```

설명

8: 조건식이 없으므로 무조건 본체를 실행한다.

12: 10행에서 반으로 접었을 때 넓이가 target보다 작으면 그만 접는다.

5.4.5 특정 단의 구구단 출력하기

문제 키보드로 1~9 사이의 숫자를 입력받아 그 숫자에 해당되는 구구단을 출력하는 프로그램을 for문을 이용해 작성하시오.

프로그램 5-17 구구단 출력하기 : for문 이용 (ch5-17.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int i, n;
6
7       printf("출력을 원하는 단은?");
8       scanf("%d", &n);
9
10      for (i=1; i<=9; i++)
11          printf("%d * %d = %2d\n", n, i, n*i);
12
13      return 0;
14  }
```

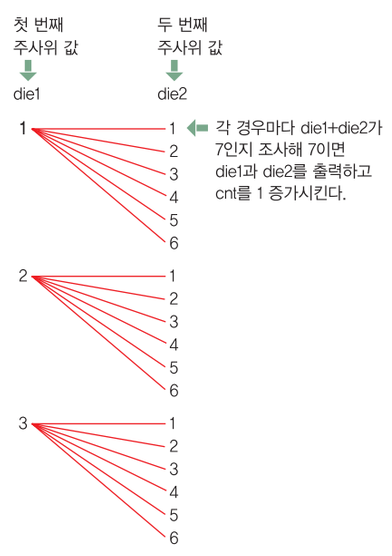
실행결과

```
출력을 원하는 단은? 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
```

5.4.6 중첩된 for문을 이용해 주사위 경우의 수 구하기

문제 주사위를 두 개 던졌을 때 합이 7이 되는 경우를 출력하시오.

분석 • 주사위를 던졌을 때 나올 수 있는 값은 1~6 중 하나다. 첫 번째 주사위가 1이 나왔을 때 두 번째 주사위 값도 1~6 중 하나가 나올 수 있다. 그러므로 다음 그림과 같이 두 주사위에 대해 모든 경우를 조사하려면 첫 번째 주사위 값을 바깥 for문의 제어 변수로 사용하고 두 번째 주사위 값을 안쪽 for문의 제어 변수로 사용해 중첩 for문으로 작성한다.



프로그램 5-18 두 주사위의 합이 7이 되는 경우의 수 구하기 (ch5-18.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int die1, die2;
6       int cnt;
7
```

```
8       cnt = 0;
9
10      printf("   ==================\n");
11      printf("    주사위1   주사위2 \n");
12      printf("   ==================\n");
13
14      for (die1=1; die1<=6; die1++)
15      {
16          for (die2=1; die2<=6; die2++)
17          {
18              if (die1 + die2 == 7)
19              {
20                  printf("\t%d \t%d \n", die1, die2);
21                  cnt++;
22              }
23          }
24      }
25
26      printf("   ==================\n");
27      printf("\t총 %d가지\n", cnt);
28
29      return 0;
30  }
```

실행결과

```
   ============
   주사위1   주사위2
   ============
      1         6
      2         5
      3         4
      4         3
      5         2
      6         1
   ============
      총 6가지
```

설명

18: 두 주사위의 합이 7이 될 때만 경우의 수를 1 증가시키고 die1과 die2의 현재 내용을 출력한다.

5.4.7 중첩된 for문을 이용해 알파벳 출력하기

문제 첫 번째 줄에는 z만, 다음 줄에는 y와 z만 출력하는 식으로 알파벳을 증가시켜 최종 a…z까지 출력하는 프로그램을 작성하시오. 단 출력되지 않은 앞의 알파벳 자리에는 공백으로 넣는다.

분석

- 각 행에 출력하는 내용은 빈 칸 즉 ' '를 출력하는 부분과 뒤에 알파벳을 z까지 출력하는 부분으로 나뉜다.
- 앞에 출력하는 빈 칸의 개수는 a부터 뒤에 출력되는 알파벳 이전까지의 알파벳 개수와 같다. 실제 출력을 시작하는 알파벳이 start 변수에 저장되어 있다고 가정하면 for문 두 개로 표현할 수 있다.

```
for (ch='a'; ch<start; ch++)      // 공백 출력
   printf(" ");

for (ch=start; ch<='z'; ch++)     // 알파벳 출력
   printf("%c", ch);
```

- 전체 출력 행 수가 알파벳 개수와 같으므로 for(start='a'; start<='z'; start++)처럼 각 행의 출력을 반복하면 될 것 같다. 그러나 start는 a부터 z까지 변하는데 다음 행으로 갈수록 출력을 시작하는 문자는 오히려 z에서 a까지로 변한다. 그러므로 for문의 헤더를 for(start='z'; start>='a'; start--)로 수정해야 한다.

프로그램 5-19 알파벳 출력하기 (ch5-19.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       char start, ch;
6
7       for(start='z'; start>='a'; start--)
8       {
9           // 각 행 시작  부분의 연속된 빈 칸 출력하기
10          for(ch='a'; ch<start; ch++)
11              printf(" ");
12
13          // 빈 칸 뒤에 알파벳을 start부터 z까지 출력하기
14          for(ch=start; ch<='z'; ch++)
15              printf("%c", ch);
16
17          printf("\n");   // 알파벳 출력 후 행 바꾸기
18      }
19
20      return 0;
21  }
```

실행결과

```
z
yz
xyz
wxyz
vwxyz
uvwxyz
tuvwxyz
stuvwxyz
rstuvwxyz
qrstuvwxyz
pqrstuvwxyz
opqrstuvwxyz
nopqrstuvwxyz
mnopqrstuvwxyz
lmnopqrstuvwxyz
klmnopqrstuvwxyz
jklmnopqrstuvwxyz
ijklmnopqrstuvwxyz
hijklmnopqrstuvwxyz
ghijklmnopqrstuvwxyz
fghijklmnopqrstuvwxyz
efghijklmnopqrstuvwxyz
defghijklmnopqrstuvwxyz
cdefghijklmnopqrstuvwxyz
bcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

행 수는 알파벳 개수와 같다.

설명

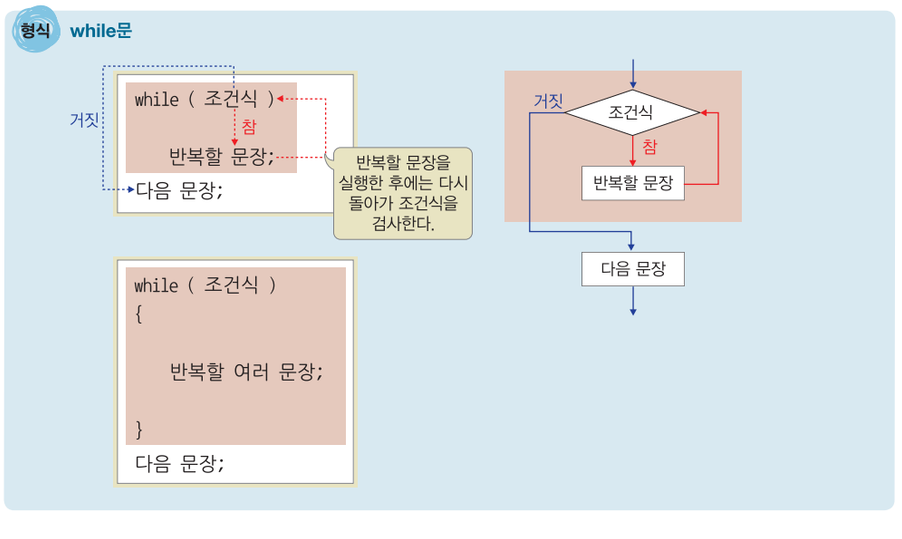
7: start는 char형 변수이므로 start--를 하면 ASCII 코드 값이 1 감소하므로 결국 현재 알파벳 이전의 문자가 된다.

5.5 반복문(while문, do~while문)

5.5.1 while문

특정 작업을 반복하기는 하지만 반복 횟수가 얼마나 되어야 하는지 정확히 모르는 경우가 있다. 비밀번호를 잊어버린 경우 몇 번을 입력해야 찾을 수 있는지 알지 못한다. 확실한 사실은 비밀번호를 틀리는 한 계속 반복해야 한다는 것이다. 비슷한 예로 반복 횟수는 정확히 모르지만 음수가 입력된다면 계속 입력받기, 합이 1000을 넘지 않는 한 계속 더하기, 게임을 계속 하고 싶어하는 한 다시 게임하기와 같은 문제를 들 수 있다. 이처럼 특정 조건을 만족하는 한 계속 반복하는 문제는 while문을 사용하는 것이 좋다.

사실 반복 문제는 모두 for문으로 작성할 수 있다. for문은 헤더인 for(i=1; i<=10; i++)에 i가 1부터 1씩 증가하며 10이 될 때까지 10번 반복하는 것을 정확히 표현할 수 있다. 따라서 주어진 문제의 반복 횟수가 명확할 때 이용하면 편리하다. 반면 while문은 반복 횟수는 모르지만 어떤 조건을 만족할 때까지 반복할지를 명확히 아는 경우 편리하다. 그러므로 문제에 따라 for문과 while문을 선택해서 사용하면 된다. 다음은 while문의 형식과 실행의 흐름을 나타낸다.



- 조건식은 while문 본체의 반복 여부를 결정한다. while문을 실행할 때는 먼저 조건식을 검사한다. 조건식이 참이면 다음에 나오는 반복 문장을 실행하고 거짓이면 while문을 끝낸다.
- while문 본체의 마지막 문장을 실행한 후에는 다시 조건식이 있는 곳으로 돌아와 조건식을 검사한다. 조건식이 참이면 해당 문장을 다시 반복하고 거짓이면 while문을 끝낸다. 즉 조건식이 참인 동안은 본체를 계속 반복 실행한다.
- while문을 시작할 때 처음부터 조건식의 값이 거짓일 경우에는 본체가 한번도 실행되지 않을 수 있다.

예

```
1 // 1~n의 합이 처음으로 100을 넘게 되는 n 구하기
2 sum = 0, i = 0;
3 while (sum <= 100)
4 {
5    i++;
6    sum = sum + i;
7 }
8 printf("1~%d까지의 합: %d \n", i, sum);
```

while문 본체의 마지막 문장을 실행한 후에는 다시 while문의 헤더로 돌아가 조건을 검사한다.

while문과 for문의 비교

다음은 1~10의 합을 구하는 for문과 while문을 비교한 것이다. for문의 헤더는 초기식, 조건식, 증감식을 포함하지만 while문의 헤더는 조건식만 포함한다. 그러므로 일반적으로 while문 이전에 제어 변수의 초기화(2행), while문의 본체가 끝나기 전에 제어 변수의 증감 연산 부분(6행)이 포함된다.

```
1  sum = 0;
2  i = 1;
3  while (i <= 10)
4  {
5     sum = sum + i;
6     i++;
7  }
```

```
1  sum = 0;
2  for (i=1; i<=10; i++)
3     sum = sum + i;
```

반복 횟수를 정확히 알면서도 while문을 사용하게 되면 코드 길이도 길어지고 반복 횟수도 한눈에 알 수 없어 좋지 않다. [프로그램 5-20]은 1~n의 합과 곱을 구하는 [프로그램 5-13]에서 1~n의 곱을 구하는 부분만 while문으로 수정한 것이다.

프로그램 5-20 정수 1~n의 합과 곱 구하기 : while문 이용 (ch5-20.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int i, n, factorial;
6
7       printf("1부터 n까지의 곱을 구합니다. 정수 n은? ");
8       scanf("%d", &n);
9
10      // 1~n의 곱인 n!을 factorial에 구하기
11      factorial = 1;
12      i = 1;
13      while (i <= n)
14      {
15          factorial = factorial * i;
16          i++;
17      }
18
19      printf("1부터 %d까지의 곱은 %d입니다.\n", n, factorial);
20
21      return 0;
22  }
```

실행결과

```
1부터 n까지의 곱을 구합니다. 정수 n은? 10
1부터 10까지의 곱은 3628800입니다.
```

설명

12: while문이 시작되기 전에 반복의 초기화를 담당하는 제어 변수를 반드시 초기화해야 한다.

13: (i < n)으로 지정하면 1에서 (n-1)까지의 곱만 구해진다. while문은 for문보다 조건식을 작성할 때 더욱 주의해야 한다.

16: 다시 13행으로 돌아가기 전에 i 값을 증가시키지 않으면 13행의 조건식이 언제나 참이 되므로 무한 루프가 발생한다.

while문과 무한 루프

for문에서 조건식을 생략하면 무한 루프를 만들 수 있었다. while문에서 무한 루프를 만들려면 조건식을 항상 참이 되게 한다. 0이 아닌 값은 모두 참이지만 보통 아래 프로그램의 4행처럼 조건식으로 1을 이용한다. 무한 루프 형태로 반복문을 표현한 경우에는 반복을 끝내고 while문을 탈출할 수 있도록 8~9행처럼 break를 이용하는 경우가 많다.

```
1   // 1~n의 합이 처음으로 100을 넘게 되는 n 구하기
2   sum = 0, i = 0;
3
4   while (1)          <- 정수 1은 참이므로 무조건 본체를 실행한다.
5   {
6      i++;
7      sum = sum + i;
8      if (sum > 100)  <- while문 탈출 조건
9         break;
10  }
11  printf("1~%d까지의 합: %d \n", i, sum);
```

왼쪽 코드는 사용자가 양수를 입력할 때까지 값을 계속 입력받도록 while문을 무한 루프 형태로 표현한 것이다. 오른쪽은 [프로그램 5-20]의 11행~17행을 무한 루프 형태로 표현한 것이다.

```c
while (1)
{
   printf("양수를 입력하세요.");
   scanf("%d", &n);
   if (n > 0)
      break;
}
```

while문 탈출 조건

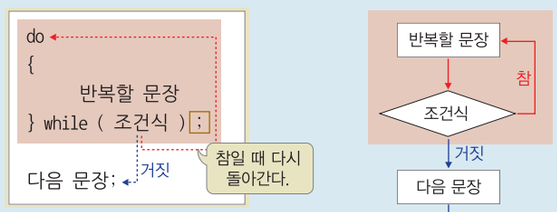
```c
factorial = 1;
i = 1;
while (1)
{
   if (i > n)
      break;
   factorial = factorial * i;
   i++;
}
```

while문 탈출 조건

5.5.2 do~while문

이공 계열에서 다루는 문제는 수치 데이터를 다루거나 특정한 일을 정해진 횟수만큼 반복하는 경우가 많다. 그러나 일반 응용 프로그램에서는 일단 한 번 실행한 후에 다시 반복 실행할지 그만둘지를 결정해야 하는 경우가 많다. 그 예가 메뉴 방식 프로그램이다. 프로그램을 통해 처리할 수 있는 기능을 메뉴로 표시하고 사용자가 원하는 메뉴를 선택하면 해당 메뉴에 대한 기능을 처리해주며, 끝내기 전까지는 다시 메뉴를 처리하도록 반복하는 형태다. 프로그래머 입장에서 반복 횟수를 미리 알 수 없으므로 while문으로 작성하는 것이 좋다. 그런데 이러한 메뉴 방식의 응용 프로그램은 사용자에게 일단 한 번은 메뉴를 표시한다는 특징이 있다. 이처럼 반복 내용을 반드시 한 번 실행한 후 계속할지 그만둘지 결정하는 경우에는 while문보다 do~while문을 이용하는 것이 훨씬 편리하다. 다음은 do~while문의 형식과 실행의 흐름을 보여준다.

형식 do~while문



do~while문의 실행 순서

do문 뒤에 조건식이 없으므로 무조건 본체를 실행한다. 본체를 실행한 후에는 while 뒤의 조건식을 검사해 참이면 다시 본체를 실행하고 거짓이면 do~while문을 끝낸다.

do~while의 본체

- for문이나 while문과 달리 반복할 문장이 하나뿐이더라도 반드시 {}로 묶어야 한다.
- 본체가 끝난 후 } 뒤에 'while (조건식)'을 붙여서 쓰며 마지막에는 ;을 꼭 붙여야 한다.

예
```c
// 양의 정수가 입력될 때까지 age에 나이 입력받기
do
{
   printf("나이는? \n");
   scanf("%d", &age);
} while (age <= 0);          //입력된 나이가 0 이하면 다시 입력받도록 돌아가기
```

조건식이 참일 때 되돌아가서 4행부터 다시 실행한다.

예
```c
// 'C Language'를 100행 출력하기
i = 0;                       // for문의 초기식에 해당
do {
   printf("C Language\n"); // for문의 반복할 문장에 해당
   i++;                      // for문의 증감식에 해당
} while (i <= 100);          // for문의 조건식에 해당
```

[프로그램 5-21]은 정수 1~n의 합과 곱을 구하는 프로그램을 do~while문으로 작성한 것이다.

프로그램 5-21 정수 1~n의 합과 곱 구하기 : do~while문 이용 (ch5-21.cpp)

```c
#include <stdio.h>

int main()
{
    int i, n, sum = 0, mult = 1;

    printf("정수 n : "); scanf("%d", &n);

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

    printf("1부터 %d까지의 합은 %d입니다.\n", n, sum);
    printf("1부터 %d까지의 곱은 %d입니다.\n", n, mult);

    return 0;

}
```

```
실행결과
정수 n : 10
1부터 10까지의 합은 55입니다.
1부터 10까지의 곱은 3628800입니다.
```

5.6 기타 제어문

프로그램이 실행되면 제일 먼저 main 함수가 호출되고 main 함수에 있는 일련의 문장들이 나열된 순서대로 실행된다. C 언어는 앞에서 설명한 조건문과 반복문 외에도 문장들의 실행 순서를 변경할 수 있는 다양한 제어문을 제공한다.

5.6.1 break문

break문은 앞에서 보았듯이 switch문, for문, while문, do~while문을 실행하는 중간에 완전히 탈출할 때 사용되는 유용한 제어문이다. 특히 while문과 같이 조건식이 항상 참이라 반복문을 빠져나올 수 없는 무한 루프에서 매우 유용하다.

```
1  // n 값이 0 이하면 다시 입력받기
2  while (1)
3  {
4     printf("필요한 물건의 개수는? \n");
5     scanf("%d", &n);
6    if (n >= 0)
7        break;
8     printf("잘못된 개수를 입력했습니다. \n");
9  }
10 ◀
```

루프 안에서 break문이 실행되면 루프를 빠져나와 루프 다음에 있는 문장이 실행된다. break문 개수는 제한이 없으나 너무 많이 사용하면 코드가 복잡해지므로 특수한 경우를 빼고는 사용을 자제하는 것이 좋다. 주의할 점은 중첩된 루프인 경우 안쪽 루프에서 break문을 사용하면 break문이 속한 안쪽 루프만 빠져나온다는 것이다. 그러므로 다음과 같이 맨 안쪽 루프에서 가장 바깥 루프까지 빠져나오려면 break문이 루프 개수만큼 필요하다.

```
1  for (i=1; i<=n; i++)
2  {
3     for (j=1; j<=n; j++)
4     {
5        for (k=1; k<=n; k++)
6        {
7        …
8           if (조건식)
9              break;
10             …
11          } // for (k)의 끝
12          …
13          if (조건식)
14             break;
15          …
16          } // for (j)의 끝
17          …
18          if (조건식)
19             break;
20             …
21          } // for (i)의 끝
```

또 다른 주의점으로는 break문은 반복문과 switch문에만 해당하므로 다음과 같이 if문을 탈출하는 데는 사용할 수 없다는 것이다.

```
1  if (answer == 1)
2  {
3     printf("1번을 선택했습니다.\n");
4     printf("필요한 물건의 개수를 입력하세요.");
5     scanf("%d", &n);
6     if (n <= 0)
7        break;
8     printf("총 금액은 %d입니다.", n * 1500);
9  }
```

위의 예에서 프로그래머의 의도는 1행의 if문을 탈출해 10행으로 이동하기 위한 것으로 보인다. 그러나 반복문과 switch문이 없이는 break가 나타날 수 없으므로 'illegal break'라는 에러 메시지가 나타난다.

[프로그램 5-22]는 키보드로 정수 n을 입력받아 1~n의 합을 구해 출력하는 프로그램을 약간 변경한 것이다. n 값으로 -1이 입력될 때까지 반복 실행하다가 -1이 입력되면 무한 루프를 탈출해 프로그램을 종료한다.

프로그램 5-22 1~n의 합 구하기 : 무한 루프 이용 (ch5-22.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int i, n, sum;
6
7       while (1)   // 무한 루프
8       {
9           printf("정수 n을 입력(종료:-1) :  ");
10          scanf("%d", &n);
11
12          if (n == -1)
13              break;
14
15          sum = 0;
16          for (i=1; i<=n; i++)
17              sum += i;
18          printf("정수 1에서 %d까지의 합은 %d입니다.\n\n", n, sum);
19      }
20      printf("프로그램을 끝냅니다. \n");
21
22      return 0;
23  }
```

입력된 n 값이 -1이라면 루프를 탈출

실행결과

```
정수 n을 입력(종료:-1) : 5
정수 1에서 5까지의 합은 15입니다.

정수 n을 입력(종료:-1) : 10
정수 1에서 10까지의 합은 55입니다.

정수 n을 입력(종료:-1) : -1
프로그램을 끝냅니다.
```

설명

12: 7행 while문의 조건식이 항상 참(1)이므로 무한 루프가 된다. 그러므로 무한 루프를 탈출하기 위해 탈출 조건을 명시해야 한다. n의 입력으로 −1이 들어오면 break문이 실행되므로 루프를 빠져나와 루프 다음의 20행이 실행된 후 프로그램이 종료된다.

5.6.2 continue문

반복문에서 사용되는 또 다른 제어문으로 continue문이 있다. break문은 반복문의 실행을 완전히 끝내는 데 반해 continue문은 단순히 continue문 다음 내용만 실행하지 않은 채 다음 반복으로 진행하기 위해 루프의 시작이나 끝으로 이동한다. for문은 무조건 헤더의 증감식으로 이동하고, while문은 헤더의 조건식으로 이동하며, do~while은 맨 끝 while 뒤의 조건식으로 이동한다.

[프로그램 5-23]은 1~n의 짝수 합을 구하기 위해 continue문을 이용한 프로그램이다.

프로그램 5-23 1~n의 짝수 합 구하기 (ch5-23.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int i, n, sum = 0;
6
7       printf("정수 n을 입력 : ");
8       scanf("%d", &n);
9
10      i = 0;
11      while (i <= n)
12      {
13          i++;
14          if (i % 2 == 1)
15              continue;
16          sum += i;
17      }
18
19      printf("정수 1에서 %d 이하 짝수들의 합은 %d입니다.\n", n, sum);
20
21      return 0;
22
23  }
```

실행결과

```
정수 n을 입력 : 50
정수 1에서 50 이하 짝수들의 합은 650입니다.
```

설명

15: i 값이 홀수(14행)인 경우 continue문을 만나면 while문 내의 종속 문장을 더 이상 실행하지 않고 11행으로 이동해 조건식을 다시 검사한다.

프로그램 5-24 양의 정수 5개의 합 구하기 (ch5-24.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int i, n, cnt, sum;
6
7       sum = 0;    // 총 합계
8       cnt = 1;    // 입력된 양의 정수 개수
9       for (i=1; cnt<=5; i++)      // 양의 정수 5개를 입력받을 때까지 반복
10      {
11          printf("\n양의 정수 n을 입력하세요. ");
12          scanf("%d", &n);
13
14          if (n <= 0)             // 양수만 더하기 위해 0 이하면 다음 번 반복으로 진행하기
15              continue;
16
17          cnt++;
18          printf("지금까지의 합 %2d + %2d = ", sum , n);
19          sum += n;
20          printf("%2d \n", sum);
21      }
22
23      return 0;
24  }
```

실행결과

```
양의 정수 n을 입력하세요. 10
지금까지의 합  0 + 10 = 10

양의 정수 n을 입력하세요. -6

양의 정수 n을 입력하세요. 5
지금까지의 합 10 +  5 = 15

양의 정수 n을 입력하세요. 9
지금까지의 합 15 +  9 = 24

양의 정수 n을 입력하세요. -1

양의 정수 n을 입력하세요. 23
지금까지의 합 24 + 23 = 47

양의 정수 n을 입력하세요. 42
지금까지의 합 47 + 42 = 89
```

음수가 입력되었으므로 곧바로 9행으로 돌아가 증감식인 i++을 실행한 후 조건식을 검사한다.

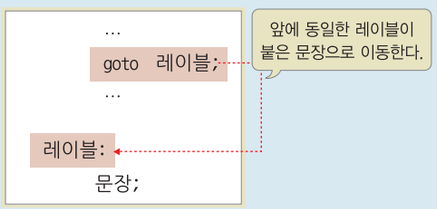
설명

15: 0 또는 음수가 입력되면 뒤의 합을 구하는 과정을 생략하고 증감식으로 돌아간다.

5.6.3 goto문

goto문은 프로그램 실행 중 레이블(label)이 붙은 위치로 제어를 무조건 이동시킬 때 사용한다. goto문을 사용하려면 이동할 문장을 가리키는 레이블(label)이 필요하며 레이블을 정의할 때는 레이블을 구별하기 위한 이름과 콜론(:)이 필요하다.

형식 goto문



예

```
// 1~n의 합이 처음으로 1000을 넘게 되는 n 구하기
i = 0;
addition:
i++;
sum += i;
if (sum <= 1000)
   goto addition;
printf("1에서 %d까지의 합 : %d \n", i, sum);
```

잠깐

C 언어에는 세 가지의 레이블이 있다. switch~case문의 case, default 레이블과 goto문의 레이블이다. 레이블 이름은 변수명 작명 규칙에 따라 만들고, 레이블 이름 뒤에는 반드시 콜론(:)을 붙여야 한다. goto문은 case와 default 레이블이 붙은 곳으로는 이동할 수 없고, 해당 레이블 이름이 있는 곳으로만 이동할 수 있다.

goto문을 사용한 프로그램은 복잡하고 이해하기 어려우므로 가능한 한 사용하지 말아야 한다. 하지만 [프로그램 5-25]와 같은 중첩된 반복 루프에서 문제가 발생하였을 때 중첩된 루프를 단번에 빠져나올 수 있어 유용하다. 앞에서 소개한 break문은 루프 하나만을 빠져나올 수 있다.

프로그램 5-25 키보드를 누를 때까지 * 출력하기 (ch5-25.cpp)

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4
5   int main()
6   {
```

```
7       int i, j;
8
9       for (i=1; i<10000; i++)
10      {
11          for (j=1; j<=10; j++)
12          {
13              if (kbhit()) goto end;
14              printf("*");
15          }
16          printf("\n");
17      }
18      printf("\n프로그램을 종료합니다!\n");
19      exit(0);
20
21  end :
22      printf("\n키보드를 눌렀습니다!\n");
23      getch();
24      return 1;
25
26  }
```

실행결과

```
**********
**********
**********
**********
**********
**********
**********
***
키보드를 눌렀습니다!
```

설명

13: 키보드가 눌리면 레이블 이름 end인 21행으로 이동한다. kbhit()는 키보드의 키가 눌렸는지를 판단하는 함수로 〈conio.h〉 헤더 파일을 필요로 한다. 함수 원형은 int kbhit(void)며, 키가 눌린 경우에는 1, 눌리지 않은 경우에는 0을 반환한다.

19: break, return, exit 모두 루프를 탈출할 때 사용된다. break문은 반복문이나 switch문, return문은 함수, exit는 프로그램 단위로 제어한다고 생각하면 이해가 쉽다. 이 중 exit() 함수는 프로그램에 값을 반환하고 종료하는 함수로 〈stdlib.h〉 헤더 파일이 필요하다. 보통 exit(0), exit(1)의 두 가지 형태로 쓰는데, 프로그램 실행이 정상적으로 종료되면 0을, 비정상적으로 종료되면 0이 아닌 정수로 표시한다.

5.7 프로그래밍 실습

5.7.1 while문을 이용해 1~n의 홀수 합과 짝수 합 구하기

문제 정수 n을 입력받아 n 이하의 모든 홀수의 합과 짝수의 합을 구하는 프로그램을 while문을 이용해 작성하시오.

분석 • 짝수를 저장할 변수 even과 홀수를 저장할 변수 odd의 최댓값은 n이어야 하므로 while문의 반복 조건식은 (odd<=n && even<=n)가 된다.

프로그램 5-26 1~n의 홀수 합과 짝수 합 출력하기 : while문 이용 (ch5-26.cpp)

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int n, odd, even, odd_sum = 0, even_sum = 0;
6
7       printf("정수 n을 입력하시오. : ");
8       scanf("%d", &n);
9
10      odd = 1; even = 2;
11      while (odd<=n && even<=n)
12      {
13          odd_sum += odd;
14          even_sum += even;
15          odd += 2;
16          even += 2;
17      }
18      printf("정수 1에서 %d 이하 홀수들의 합은 %d이고", n, odd_sum);
19      printf("짝수들의 합은 %d입니다.\n", even_sum);
20
21      return 0;
22  }
```

실행결과

```
정수 n을 입력하시오. : 50
정수 1에서 50 이하 홀수들의 합은 625이고 짝수들의 합은 650입니다.
```

설명

10: odd는 1로, even은 2로 초기화된 다음, 값이 2씩 증가하며(15행~16행) while문을 돈다.

11: n까지의 합을 구하므로 반복문은 변수 odd와 even이 모두 n 이하인지 판단하는 조건식이 참인 경우에만 실행된다.

5.7.2 반감기를 이용한 물질의 잔량 구하기(while문 이용)

문제 인의 반감기는 14일이다. 연구소에 들어온 실험 물질에 포함된 인의 양은 100mg이었다. 앞으로 n일 후 인의 양이 최소 몇 mg일지 구하는 프로그램을 작성하시오.

분석

- 현재 인의 양을 amount라고 하면 14일이 지날 때마다 amount는 반으로 줄어든다. 그러므로 새로운 반감기가 지나가기 전까지는 현재 양의 최소 반 이상(amount/2)이 남아 있다.
  - 현재 1일 ~ 14일: 최소 50mg (← amount = amount /2)
  - 15일 ~ 28일 : 최소 25mg (← amount = amount /2)
  - 29일 ~ 42일: 최소 12.5mg (← amount = amount /2)
  - …
- n일이 지날 때마다 반으로 줄어드는 인의 양을 반복 출력하기 위해 while문을 이용한다.

프로그램 5-27 반감기를 이용한 물질의 잔여량 예측하기 (ch5-27.cpp)

```
1   #include <stdio.h>
2   #include <math.h>
3
4   int main()
5   {
6       int half=14, days, n;
7       double amount = 100;  // 현재 인의 양
8
9       printf("현재 인의 양은 %6.1lfmg입니다.\n\n", amount);
10      printf("며칠째 되는 날의 인의 최소 잔량을 구할까요? ");
11      scanf("%d", &n);
12
13      days = 0;
14      while (1)   // 무한 루프로 처리
15      {
16          // 새로운 반감기가 지나면 양이 반으로 줄어든다.
17          days += half;          // 현재일로부터 지나간 일을 반감기만큼 증가하기
18          amount = amount / 2;   // 반감기인 half 일을 지나면 양이 반으로 줄어든다.
19          printf("%2d일까지의 인의 최소 잔량은 %6.1lfmg 이상 \n", days, amount);
20
21          if (days >= n) // 사용자가 입력한 일이 지났다면 반복을 끝내기
22              break;
23      }
24      printf("\n>>그러므로 %2d일 째에는 %6.1lfmg 이상 남아있습니다.\n", n, amount);
25
26      return 0;
27  }
```

지나간 총 일수와 새로운 반감기를 지났을 때 인의 양

• 30일째 되는 날의 인의 최소 잔량

```
실행결과
현재 인의 양은  100.0mg입니다.

며칠째 되는 날의 인의 최소 잔량을 구할까요? 30
14일까지의 인의 최소 잔량은   50.0mg 이상
28일까지의 인의 최소 잔량은   25.0mg 이상
42일까지의 인의 최소 잔량은   12.5mg 이상

>>그러므로 30일 째에는 최소   12.5mg 이상 남아있습니다.
```

• 42일째 되는 날의 인의 최소 잔량

```
실행결과
현재 인의 양은  100.0mg입니다.

며칠째 되는 날의 인의 최소 잔량을 구할까요? 42
14일까지의 인의 최소 잔량은   50.0mg 이상
28일까지의 인의 최소 잔량은   25.0mg 이상
42일까지의 인의 최소 잔량은   12.5mg 이상

>>그러므로 42일 째에는 최소   12.5mg 이상 남아있습니다.
```

• 45일째 되는 날의 인의 최소 잔량

```
실행결과
현재 인의 양은  100.0mg입니다.

며칠째 되는 날의 인의 최소 잔량을 구할까요? 45
10일까지의 인의 최소 잔량은  50.0mg 이상
28일까지의 인의 최소 잔량은  25.0mg 이상
42일까지의 인의 최소 잔량은  12.5mg 이상
56일까지의 인의 최소 잔량은   6.3mg 이상

>>그러므로 45일 째에는 최소   6.3mg 이상 남아있습니다.
```

설명

14: 이 프로그램에서는 무한 루프 형식으로 작성했지만, 21행과 22행을 지우고 14행의 조건식을 (days < n)으로 고쳐도 결과는 같다.

TIP 이 프로그램은 길이가 짧지만 쉽지 않다. 모든 n에 대해 정확한 결과를 얻는지 꼭 확인해야 한다. 특히 초보자의 경우 n 값이 반감기의 배수일 때는 정확한 답을 내지만 다른 세 가지 경우에는 틀리게 작성하는 경우가 많다. 새로운 반감기 이전 즉 42일보다 며칠 전에 대해서, 새로운 반감기가 지나고 그 다음 반감기가 되기 전 즉 42일보다 며칠 뒤에 대해서, 반감기인 14일보다 작은 경우에 대해서 결과를 꼭 확인하고 while문의 조건식이나 while문 탈출 조건을 정확하게 작성해야 한다.

5.7.3 do~while문을 이용해 메뉴 프로그램 작성하기

문제 사용자가 그만두기 원할 때까지 한 원의 둘레와 넓이, 구의 부피를 반복해서 구할 수 있는 메뉴 방식의 프로그램을 작성하시오.

분석

- 명확한 반복 횟수를 알 수 없으므로 for문보다 while이나 do~while문을 이용하는 것이 좋다. 일단 사용자에게 한 번은 메뉴를 표시한 후에 선택한 메뉴에 따라 둘레, 넓이 또는 부피를 구하거나 그만두기를 선택해야 하므로 do~while문으로 작성한다.

```
do {
    '1. 원의 둘레 구하기, 2. 원의 넓이 구하기, 3. 구의 부피 구하기, 4. 그만두기' 메뉴 표시 후 번호 입력받기
    사용자가 선택한 메뉴 번호에 따라 다른 일을 처리하도록 switch문으로 작성
} while (사용자의 선택인 4가 아니라면 계속하기);
```

프로그램 5-28 그만두기를 원할 때까지 여러 메뉴 중 하나를 선택해 실행하기 (ch5-28.cpp)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define PI 3.141592    // PI를 매크로 상수로 정의

int main()
{
    int answer, r;     // 사용자가 선택한 메뉴 번호, 반지름 저장 변수
    double circum, area, volume;

    do {
        printf("==============================\n");
        printf("==   1. 원의 둘레 구하기   ==\n");
        printf("==   2. 원의 넓이 구하기   ==\n");
        printf("==   3. 구의 부피 구하기   ==\n");
        printf("==    4. 그만두기          ==\n");
        printf("==============================\n");
        printf("◈ 원하는 내용은? ");
        scanf("%d", &answer);

        if (answer==1 || answer==2 || answer==3)
        {
            printf("\n》 반지름은? ");
            scanf("%d", &r);
            printf("\n》 반지름이 %d인 ", r);
        }

        switch (answer)
        {
            case 1: circum = 2 * PI * r;
                printf("원의 둘레는 %6.2lf \n", circum);
                break;
            case 2: area = PI * r * r;
                printf("원의 넓이는 %6.2lf \n", area);
                break;
            case 3: volume = 4./3 * PI * r * r * r;
                printf("구의 부피는 %6.2lf입니다.\n", volume);
                break;
            case 4: printf("프로그램을 끝냅니다. \n"); exit(0);
        }

        printf("\n》 결과를 확인했으면 아무키나 누르세요.");
        getch();
        system("cls"); // 현재 화면의 내용 지우기
    } while (answer != 4);

    return 0;
}
```

실행결과

```
==============================
==   1. 원의 둘레 구하기   ==
==   2. 원의 넓이 구하기   ==
==   3. 구의 부피 구하기   ==
==    4. 그만두기          ==
==============================
◈ 원하는 내용은? 2

》 반지름은? 7

》 반지름이 7인 원의 넓이는 153.94

》 결과를 확인했으면 아무키나 누르세요.
```

43행의 getch 함수 호출로 입력을 대기함

```
==============================
==   1. 원의 둘레 구하기   ==
==   2. 원의 넓이 구하기   ==
==   3. 구의 부피 구하기   ==
==    4. 그만두기          ==
==============================
◈ 원하는 내용은? 4
프로그램을 끝냅니다.
```

44행의 system 호출로 앞의 화면을 깨끗이 지우고 결과 창의 처음부터 메뉴가 다시 나타난다.

설명

21: 선택한 메뉴 번호가 1, 2, 3일 때만 반지름을 입력받는다.

28: 선택한 메뉴 번호에 따라 해당 기능을 처리한다.

45: 4가 입력되었을 때는 프로그램을 끝내고 4 이외의 값을 선택하면 다시 반복한다.

5.7.4 입력받은 알파벳 소문자를 대문자로 바꾸기

문제 알파벳 소문자를 입력받아 대문자로 바꾸는 프로그램을 작성하시오. 입력된 문자가 소문자가 아니면 다시 문자를 입력받고 [Esc] 키(키 값은 16진수 0x1b)가 눌리면 프로그램을 종료한다.

분석

- 일단 문자를 한 번은 입력 받은 후에 입력된 문자가 [Esc] 키인지를 보고 반복 여부를 판단하므로 do~while문이 적합하다.
- [Esc] 키의 ASCII 코드 값 0x1b보다는 매크로 상수 ESC를 사용하는 것이 작성하기도 해석하기도 쉽다.
- 입력된 문자가 알파벳 소문자인지 확인하려면 관계 연산자를 사용한다. b는 a보다 코드 값이 1 더 크다. 그러므로 'a' < 'b'는 참이 된다. 결국 알파벳 소문자는 a보다 크거나 같고 z보다 작거나 같다.
- 대문자도 소문자처럼 대문자 C가 대문자 A보다 코드 값이 2 더 크다. 그러므로 코드 값을 모르더라도 다음 대입문을 통해 소문자를 해당 대문자로 변경할 수 있다.

  대문자 = 'A' + (소문자 − 'a');

프로그램 5-29 알파벳 소문자를 입력받아 대문자로 출력하기 (ch5-29.cpp)

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4
5   #define ESC 0x1b  // [Esc] 키의 ASCII 코드값을 매크로 상수로 정의
6
7   int main()
8   {
9       char ch, upper;
10
11      do
12      {
13          printf("알파벳 소문자를 입력하시오. : ");
14          ch = getche();              // 키보드로 문자 한 개를 입력
15
16          if (ch>='a' && ch<='z')     // ch가 소문자라면
17          {
18              upper = 'A' + ch - 'a'; // 소문자를 대문자로 변경
19              printf("\n변환된 대문자는 %c입니다.\n", upper);
20          }
21          else
22              printf("\n");
23      } while (ch != ESC);            // [Esc] 키가 입력되지 않는 한 반복하기
24
25      return 0;
26  }
```

실행결과

```
알파벳 소문자를 입력하시오. : a
변환된 대문자는 A입니다.
알파벳 소문자를 입력하시오. : s
변환된 대문자는 S입니다.
알파벳 소문자를 입력하시오. : D
알파벳 소문자를 입력하시오. : f
변환된 대문자는 F입니다.
알파벳 소문자를 입력하시오. : g
변환된 대문자는 G입니다.
알파벳 소문자를 입력하시오. :
```

[Esc] 키를 입력해서 종료시킨다.

설명

14: getch 함수는 키보드로 문자 하나를 입력받는데, 입력받은 문자를 모니터에 출력해주지는 않는다. 따라서 입력받은 문자를 확인하려면 별로도 putch 함수를 사용해야 한다. getch 함수는 특별한 의미가 없는 문자 하나를 입력받거나 프로그램 실행을 잠시 중지시키고자 할 때 주로 사용한다. getche 함수는 하나의 문자를 입력받는 즉시 모니터에 출력해준다.

23: [Esc] 키를 누르면 do~while문을 빠져나와 프로그램이 종료된다.

핵심요약

제어문

문장을 순차적으로 실행하지 않고 필요에 따라 문장의 실행 순서를 제어해야 할 때 제어문(조건문, 반복문, 분기문)을 사용한다.

if~else문

조건식의 결과에 따라 다른 문장을 실행한다. 조건식은 관계 연산자 등을 이용해 괄호 안에 표현되며 결과가 0인 값은 거짓으로, 0이 아닌 값은 참으로 판단한다. 단순 if문은 else문이 생략된 문장으로 조건식이 참일 경우에만 문장을 실행하며, 판단할 조건식이 많은 다중 분기인 경우에는 중첩된 if문을 사용한다.

switch~case문

다중 분기일 경우 유용한 조건문이다. 조건식의 값을 평가한 다음, 그 값과 같은 값에 해당하는 case문을 찾아 해당 문장을 실행한다. 이때 break문을 만날 때까지 문장을 실행하고, switch문에서 빠져나온다. 만약 조건식의 값이 모든 case에 해당되지 않을 때는 default문 다음에 나오는 문장을 실행한다. 이때 default문이 없다면 switch문을 그냥 빠져나가 다음 문장을 실행한다.

for문

문장을 특정 횟수만큼 반복 처리할 때 사용되는 for문은 초기식, 조건식, 증감 연산의 세 부분으로 구성되며 세미콜론(;)으로 구분된다. 반복 루프가 시작할 때 초기식을 실행하고 조건식을 평가하여 참이면 반복할 문장을 실행하고 증감식을 실행한 후 다시 조건식을 평가한다. 조건식의 결과 값이 거짓이면 반복할 문장을 실행하지 않고 블록을 빠져나온다.

while문

조건식이 참인 동안 문장을 반복 실행한다. 주의할 점은 while문을 시작하는 처음부터 조건식의 값이 거짓일 경우에는 아래 문장을 한 번도 실행하지 않는다는 것이다.

do~while문

일단 문장을 실행한 다음 조건식을 검사한다. 따라서 for문이나 while문이 처음부터 조건식에 맞지 않을 경우 반복 문장을 한 번도 실행하지 않지만, do~while문의 문장은 반드시 한 번은 실행된다.

break문

for, while, do~while과 같은 반복문이나 switch~case와 같은 조건문에서 탈출할 때 사용된다. 조건식이 항상 참이라 빠져나올 수 없는 무한 루프에서 유용하게 쓰인다.

continue문

반복문을 실행하다가 continue문을 만나면 반복문의 시작이나 끝 부분으로 건너뛴다.

goto문

프로그램의 실행 중 레이블(label)이 붙은 위치로 제어를 무조건 이동시킨다.

연습문제

01 C 언어에서 제공하는 반복문인 while문과 do~while문의 차이점을 설명하시오.

02 다음 두 if문을 논리 연산자를 이용해 if문 하나로 작성하시오.

①
```
if (x < 10)
    printf("%d\n", x);if (x > 20)
    printf("%d\n", x);
```

②
```
if (x > 10)
    if (x < 20)
      printf("%d\n", x);
```

03 다음 프로그램에서 잘못된 부분을 찾아 수정하시오.

```
#include <stdio.h>
int main()
{
   float n;
   printf("입력 : ");
   scanf("%f", &n);

   switch(n)
   case 1.0 : printf("%f\n", n);
   case 2.0 : printf("%f\n", n+1);
   case 3.0 : printf("%f\n", n+2);
   default  : printf("실행 종료!");

   return 0;
}
```

04 다음 프로그램의 실행 결과를 추정한 뒤 직접 실행한 결과 값과 비교해보시오.

①
```
#include <stdio.h>
int main()
{
   if (10%3 && 1-0)
      printf("ABC");
   else
      printf("DEF");
   printf("G");

   return 0;
}
```

②
```
#include <stdio.h>
int main()
{
   int x, y;
      for (i=1; i<=3; i++)
         for(j=1; j<=3; j++)
            i +=j;
   printf("i = %d, j = %d\n", i, j);

   return 0;
}
```

③
```
#include <stdio.h>
int main()
   {
      int n = 1;
      while (n++ <= 10)
      {
         if (n == 7) continue;
         printf("n = %d\n", n);
      }

   return 0;
}
```

④
```
#include <stdio.h>
int main()
{
   int n = 1, sum = 0;
   do
   {
      sum = ++n + 5;
   } while (n <= 10);
   printf("sum = %d\n", sum);
}
```

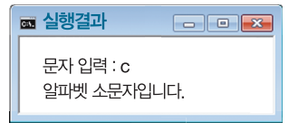
05 다음 프로그램에서 ①은 동일한 결과를 출력하는 while문으로, ②는 동일한 결과를 출력하는 for문으로 변환해 작성하시오.

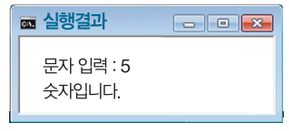
①
```
int i;
for (i=1; i<=10; i+=3)
{
   if (i % 5 == 0)
      printf("%d\n", i);
}
```

②
```
int i = 10;
while (i >= 0)
{
   printf("%d\n", i);
   i -= 3;
}
```

06 하나의 문자를 입력받고, 입력받은 문자를 알파벳 대문자, 알파벳 소문자, 숫자, 그 외의 문자들로 구별하는 프로그램을 if문을 이용해 작성하시오.

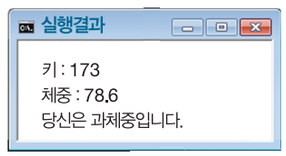
hint 입력받은 단일 문자가 '0'과 '9' 사이의 값이면 숫자, 'A'와 'Z' 사이의 값이면 알파벳 대문자, 'a'와 'z' 사이의 값이면 알파벳 소문자다.

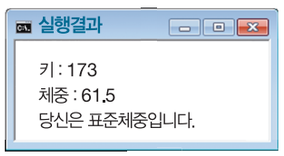




07 한 사람의 키와 체중을 입력받아 표준 체중을 계산한 후, 입력받은 사람의 체중과 비교해 저체중인지, 표준인지, 과체중인지를 판단하는 프로그램을 작성하시오.

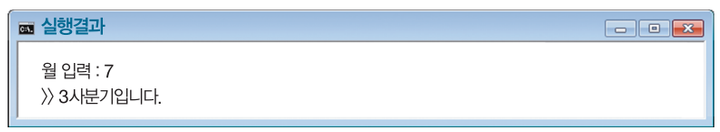
hint 표준체중 = (키 – 100) * 0.9





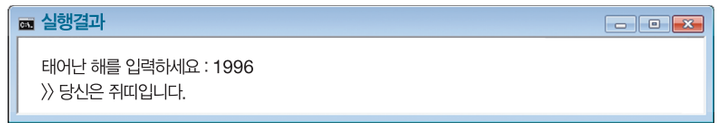
08 월을 입력받아 몇 분기인지를 출력하는 프로그램을 switch~case문을 이용해 작성하시오.

hint 1~3월(1사분기), 4~6월(2사분기), 7~9월(3사분기), 10~12월(4사분기)

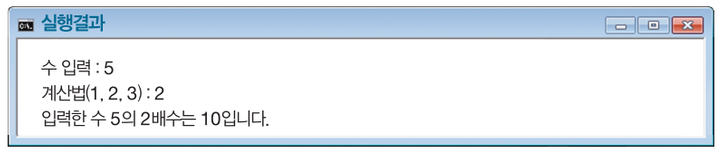


09 키보드로 태어난 년도를 입력받아 그 해의 띠를 출력하는 프로그램을 switch문으로 작성하시오.

hint 입력받은 년도를 12로 나눈 나머지를 구하여 띠를 판단하도록 한다. 나머지가 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11일 때 각각 원숭이띠, 닭띠, 개띠, 돼지띠, 쥐띠, 소띠, 범띠, 토끼띠, 용띠, 뱀띠, 말띠, 양띠다.

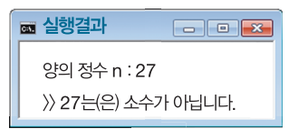


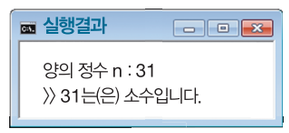
10 키보드로 정수 하나와 계산 방법을 입력받은 뒤 계산 방법이 1이면 입력된 수를, 2이면 2배수를, 3이면 3배수를 출력하는 프로그램을 switch~case문을 이용해 작성하시오. 계산 방법으로 1, 2, 3 외의 입력이 들어오는 경우에는 '잘못된 연산자입니다.'라는 메시지를 출력한다.



11 양의 정수 하나를 입력받고, 입력받은 수가 소수(prime number)인지 판단하는 프로그램을 작성하시오.

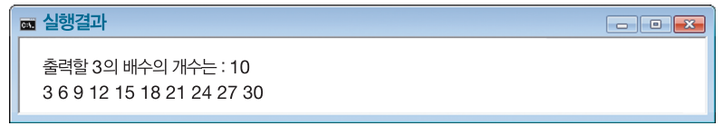
hint 소수는 1과 자기 자신만으로 나누어지는 1보다 큰 양의 정수를 말한다. 예를 들면 2, 3, 5, 7, 11, 13, …이다. 입력받은 수 n이 소수인지를 판단하려면 n을 2부터 (n–1) 사이의 정수로 나누어 어떤 수로도 나누어지지 않는지 확인한다.





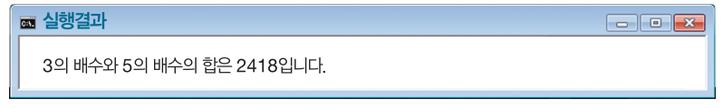
12 숫자 하나(n)를 입력받은 후, 3의 배수를 그 수만큼 출력하는 프로그램을 작성하시오.

hint
```
while(  ) // n을 1씩 감소시켜 0이 되면 while문을 탈출하도록 한다.
{
    3의 배수(3×1, 3×2, …)를 출력
}
```



13 1과 100 사이에 존재하는 모든 3의 배수와 5의 배수의 합을 계산해 출력하는 프로그램을 작성하시오.

hint 3으로 나눈 나머지가 0이거나 5로 나눈 나머지가 0이면 합한다.



14 입력 단위(cm, inch)와 값을 입력받은 후, 입력 단위가 cm이면 inch로, inch이면 cm로 변환하는 프로그램을 조건문을 이용해 작성하시오. 단, 입력 단위가 cm이면 문자 'c'나 'C'를, inch이면 'i'나 'I'를 입력한다. 1inch는 2.54cm에 해당하며 출력 값은 소수 두 자리로 나타낸다.

```
실행결과
입력 단위가 cm이면 'c'나 'C'를, inch이면 'i'나 'I'를 입력하세요.
입력 단위 : i
입력 값 : 2

입력한 2.00inch는 5.08cm입니다.
```

15 어떤 수를 입력받고 입력받은 수보다 작은 소수를 전부 출력하는 프로그램을 작성하시오.

hint 소수인지 판단해야하는 수들은 2부터 입력받은 수(max)까지며, 소수 판단은 앞의 연습문제 11의 프로그램을 이용한다.

```
while(n<max) // 2부터 max까지의 모든 수를 소수인지 판단함. n 값은 2로 초기화
{
    // 소수 판단(연습문제 11)
}
```

```
실행결과
최대 몇까지의 소수를 출력? 50
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

16 1~n의 합 중 1000을 넘지 않는 가장 큰 합과 그때의 n을 구하는 프로그램을 작성하시오.

hint 숫자를 1부터 1씩 증가하며 합계를 구하는 과정을 합계가 1000을 넘을 때까지 반복한다.

```
while(1)
{
    // 합계를 구하여 1000을 넘으면 루프를 탈출하도록 한다.(break문 이용)
}
```

```
실행결과
1부터 44까지의 합이 990입니다.
```

17 다음 수식을 계산하는 프로그램을 작성하시오. (for문 이용)

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \cdots + n^2$$

```
실행결과
제곱의 합을 구할 n을 입력 : 10
1부터 10까지의 제곱의 합은 385입니다.
```

18 구구단(2단~9단)을 출력하는 프로그램을 작성하시오. 이때 단 사이는 한 행씩 띄우며 출력하도록 한다.

hint 이중 for문을 이용한다.

```
for(i=2; i<=9; i++)
    for(j=1; j<=9; j++)
```

```
실행결과                    (i 값, j 값)
2*1= 2
2*2= 4
...
2*9=18

3*1= 3
3*2= 6
...
3*9=27

4*1= 4
4*2= 8
...
```

19 구구단(2단~9단)을 가로로 출력하는 프로그램을 작성하시오.

hint 가로 출력은 이중 for문을 돌릴 때 제어 변수를 바꿔서 지정해주면 간단히 해결할 수 있다.

실행결과

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

20 키보드로 정수 n을 입력받아 n!(factorial)을 구하는 프로그램을 작성하시오.

hint n!의 정의는 다음과 같다.

n = 0일 때 n! = 1
n >= 1일 때 n!=n × (n-1) × (n-2) × … × 1

실행결과

```
숫자를 입력하세요 : 5
5! = 120
```

21 양의 정수 a와 b를 입력받은 후, 1 이상 100 이하의 정수 중 a의 배수지만 b의 배수가 아닌 수를 모두 출력하는 프로그램을 작성하시오.

hint 어떤 수 i가 출력되기 위해서는 a로는 나누어지고 b로는 나누어지지 않아야 한다.

```
(i%a)==0 && (i%b)!=0
```

실행결과

```
두 개의 정수 입력: 5 3
5의 배수이지만 3의 배수는 아닌 수
5 10 20 25 35 40 50 55 65 70 80 85 95 100
```


22 반복문을 이용해 다음과 같은 패턴을 출력하는 프로그램을 작성하시오.

실행결과

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

실행결과

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

실행결과

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
```

실행결과

```
123456789
12345678
1234567
123456
12345
1234
123
12
1
```

23 원하는 피보나치의 개수를 입력해서 그 개수만큼 수열이 출력되는 프로그램을 작성하시오.

hint 피보나치 수는 0과 1로 시작하며, 다음 피보나치 수는 바로 앞의 두 피보나치 수의 합이 된다. 피보나치 수열의 정의는 다음과 같다.

$$f_0 = 0$$
$$f_1 = 1$$
$$f_{i+1} = f_i + f_{i-1}, i = 1, 2, \ \ldots$$

실행결과

```
피보나치 개수 입력 : 12

*** 피보나치 수열 ***
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89입니다.
```

24 자연수 두 개 입력받은 후, 최대 공약수를 구하는 프로그램을 while문을 이용해 작성하시오.

hint 최대공약수를 구하는 알고리즘은 다음과 같다.

① 두수 중 큰 수를 x, 작은 수를 y라 한다.
② y = 0이면 최대공약수는 x이며 프로그램을 종료한다.
③ r = x % y
④ x = y
⑤ y = r
⑥ 다시 ②로 되돌아간다.

실행결과

```
두 개의 정수를 입력하시오 : 12 8
최대 공약수는 4입니다.
```

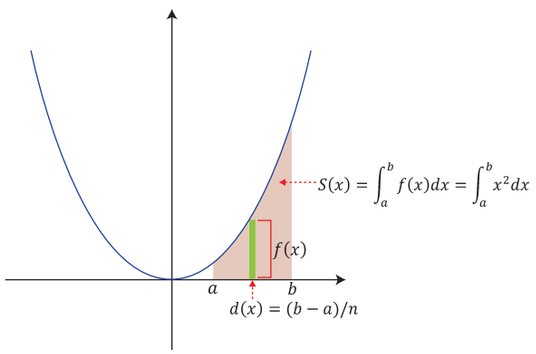
25 구간 $[a, b]$에서 연속인 함수 $f(x) = x^2 (f(x) \geq 0)$에 대해 $x = a,\ x = b,\ y = f(x)$로 둘러싸인 넓이(정적분)는 다음과 같이 정의된다.

$$S(x) = \int_a^b f(x)dx = \int_a^b x^2 dx$$

구분구적법을 이용하여 $f(x)$의 정적분 값을 계산하시오.

hint 주어진 평면도형의 넓이를 구하기 위해 주어진 도형을 충분히 작은 n개의 기본도형으로 세분하고, 이 기본 도형들의 넓이의 합을 정적분의 근삿값으로 취하는 것이 구분구적법이다. 적분 값과 근삿값의 오차를 줄이려면 n을 될 수 있는 한 큰 값으로 설정해야 한다.

– n : 정수로서 적분 구간을 몇 개로 나눌 것인지의 값
– dx : 주어진 구간[a, b]를 n개의 사각형으로 나누었을 때, 가로 길이에 해당하는 변수



실행결과

```
적분 구간 시작은? 1
적분 구간 끝은? 3
n(적분 구간의 개수)은? 10000

f(x) = x²의 [1.0, 3.0] 구간의 적분 근사치는 8.668입니다.
```

26 다음 조건을 만족시켜 369게임 프로그램을 작성하시오.(sprintf(), strchr() 함수 이용)

① 1부터 999까지의 진행 결과만 출력한다.
② 숫자가 3의 배수이거나 숫자에 3이 들어간 경우에는 박수를 친다.
③ 한 행에 열한 개를 출력한다.
④ 박수를 쳐야 할 위치에서는 대문자 P를 출력한다.

hint – sprintf()가 printf()와 다른 것은 printf()가 출력 대상을 stdout 즉, 콘솔 화면에 출력해주는 데 반해 sprintf()는 지정한 문자열에 출력해준다는 것이다.

```
a = 10;
char str[100];
printf("a : %d\n", a);          // printf는 콘솔 화면에 "a : 10" 출력
sprintf(str, "a : %d\n", a);    // sprintf는 str 문자열에 "a : 10" 문자열을 할당
```

– strchr()는 문자열 내에서 지정한 문자가 처음으로 발견되는 위치를 구하는 문자열 검색 함수다. 문자 ch가 발견될 때까지 문자열 str을 왼쪽에서 오른쪽으로 검색하고, 동일 문자를 발견하면 해당 문자의 포인터 값을, 없으면 NULL을 반환한다.

```
char *strchr(char *str, int ch);
```

실행결과

```
 1   2   P   4   5   P   7   8   P  10  11
 P   P  14   P  16  17   P  19  20   P  22
 P   P  25  26   P  28  29   P   P   P   P
 P   P   P   P   P   P  40  41   P   P  44
 P  46  47   P  49  50   P  52   P   P  55
56   P  58  59   P  61  62   P  64  65   P
67  68   P  70  71   P   P  74   P  76  77
 P  79  80   P  82   P   P  85  86   P  88
89   P  91  92   P  94  95   P  97  98   P
100 101  P   P  104  P  106 107  P  109 110
 P  112  P   P  115 116  P  118 119  P  121
                    ...
```