

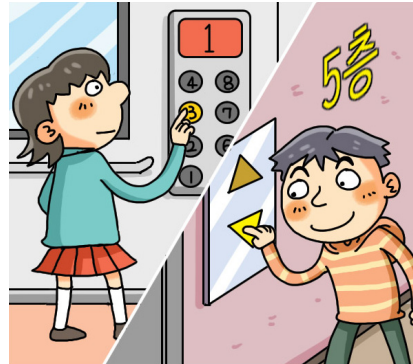
4

제어문

- 학습목표**
- 순차, 선택, 반복의 제어 구조를 이해하고 이를 활용하여 프로그램을 작성할 수 있다.
 - 중첩 제어 구조를 이해하고 중첩 제어 구조가 적용된 프로그램을 작성할 수 있다.

생각 펼치기

제어란 원하는 방향으로 작동하게 만드는 것을 말한다. 엘리베이터 시스템은 올라가고 내려가는 동작이 반복되는 제어를 통해 운행된다. 내가 원하는 층의 버튼을 누르면, 내가 선택한 층으로 엘리베이터는 이동한다. 여러 명이 엘리베이터를 이용하면서 각각 원하는 층의 버튼을 누르면, 순차적으로 엘리베이터는 작동한다.

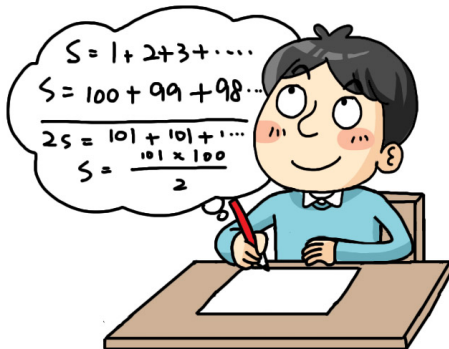


핵심 질문

▶ C 언어에서 순차, 선택, 반복 구조를 구현하는 방법은 무엇일까?

미션

이 단원을 학습하면서 해결해 보자.



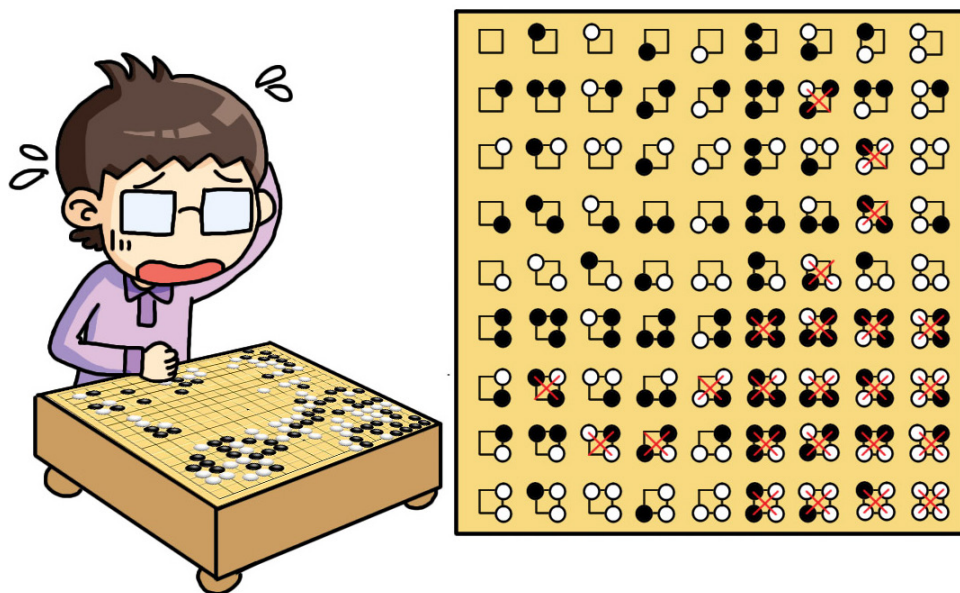
두 수를 입력받아 처음 수부터 마지막 수까지의 누적합을 구하는 프로그램을 작성해 보자.

```
C:\Windows\system32\cmd.exe
두 수를 입력하세요 : 1 100
1부터 100까지의 합은 5050입니다..
```

1 제어 구조에 대해 알아보자

우리는 하루에도 몇 번씩 여러 가지 조건 중에 하나를 선택해야 하고 순차적으로 작업을 진행하며 일을 처리한다. 이처럼 프로그램에서도 조건에 따라 프로그램의 흐름을 제어할 수 있는데, 이때 사용하는 것이 순차 구조, 선택 구조, 반복 구조이다.

예를 들어, 바둑을 둘 때 수많은 경우의 수 중에서 최적의 조건을 선택한다. 두 수는 순차적으로 바둑을 진행하며 바둑알을 놓는 작업을 반복하여 수행한다.



순차 구조

순차 구조는 정해진 순서에 따라 차례대로 처리 내용을 실행하는 구조이다.

선택 구조

선택 구조는 조건에 따라 서로 다른 작업을 처리하는 구조이다.

이처럼 일상생활 속에서 나타나는 일들을 프로그래밍으로 구현하는 데 제어 구조가 많이 활용된다.



[그림 II-2] 순차 구조와 선택 구조

2 선택 구조의 종류에 대해 알아보자

C 언어의 선택 구조에는 if 문, if-else 문, 다중 if 문, switch-case 문과 같은 조건문이 있으며, 조건에 따른 제어가 필요할 때 사용한다. 그중 switch-case 문은 비교해야 할 조건이 많을 경우 다중 if 문을 대신해서 많이 사용된다.

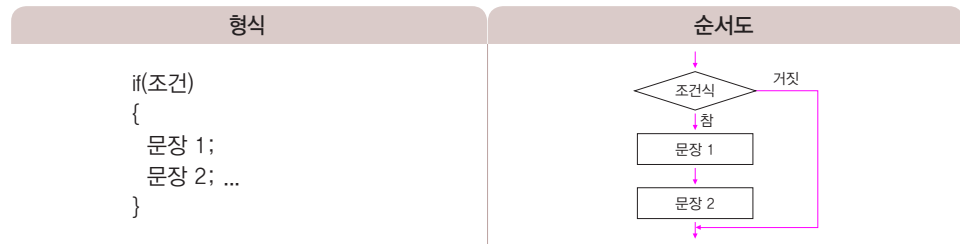
1 if 문

if 문의 조건

- if 문의 조건식에는 주로 관계 연산자를 이용한다.
- 종괄호가 없는 경우 조건이 거짓 일 때, 문장 1만 영향을 받고 문장 2는 무조건 실행된다.

단순 if 문은 조건식에 따라 특정 문장의 수행 여부를 결정한다.

- if 문의 조건이 참이면 문장 1과 문장 2를 실행하고, 거짓이면 종괄호 안의 문장을 실행하지 않는다.
- 조건식에 따라 실행할 문장이 여러 개일 때는 반드시 종괄호({ })로 묶어 준다.



예제 if 문 조건의 참, 거짓 상황에 따른 결과를 출력하는 프로그램을 작성해 보자.

실행 결과

ABCFH

C 언어에서 조건식의 결과가 0이 아니면 모두 참으로 판단해야 하는구나!



프로그램

```
01 #include <stdio.h>
02 int main( )
03 {
04     int i = 13, j = 5;
05     if (-1) putchar('A');
06     if (1) putchar('B');
07     if (13) putchar('C');
08     if (0) putchar('D');
09     if (i>j)
10     {
11         putchar('E');
12         putchar('F');
13     }
14     if (i<j)
15         putchar('G');
16     putchar('H');
17     return 0;
18 }
```

14번~16번 라인과 같이 종괄호로 묶지 않았을 때 15번 라인은 if 문의 몸체이고, 16번 라인은 무조건 실행돼.



2 if-else 문

if-else 문은 조건이 참일 때 실행되는 문장과 거짓일 때 실행되는 문장이 서로 다를 경우 사용하는 선택문이다.

- if 문의 조건이 참이면 문장 1, 문장 2를 실행하고, 조건이 거짓이면 문장 3, 문장 4를 실행한다.
- 실행할 문장이 2개 이상일 경우 중괄호({ })로 묶어 준다.

형식	순서도
<pre> if(조건) { 문장 1; 문장 2; ... } else { 문장 3; 문장 4; ... } </pre>	<pre> graph TD Start(()) --> Cond{조건식} Cond -- 참 --> S1[문장 1] Cond -- 거짓 --> S3[문장 3] S1 --> S2[문장 2] S3 --> S4[문장 4] S2 --> End(()) S4 --> End </pre>

예제 if-else 문을 사용하여 if 문의 조건이 참이면 'P'를 출력하고, 거짓이면 'F'를 출력하는 프로그램을 작성해 보자.

프로그램

```

01  #include <stdio.h>
02  int main( )
03  {
04      int standard = 80, score = 76;
05      if (standard < score)
06      {
07          putchar('P');
08      }
09      else
10      {
11          putchar('F');
12      }
13      return 0;
14  }
        
```

5번 라인의 조건이 거짓이기 때문에 else 문의 몸체가 실행되어 F를 출력해.



실행 결과

F

3 다중 if 문

다중 if 문의 조건

- if-else if 문은 비교 · 판단할 조건식이 여러 개인 경우 else if 문을 사용하여 여러 개의 조건을 비교할 수 있다.
- else if 문의 사용 개수에는 제한이 없다.
- 마지막의 else는 사용하지 않아도 된다.

if 문 안에 또 다른 if 문을 포함한 것으로 비교 · 판단 조건을 복합적으로 사용할 수 있다. 조건 1이 참이면 문장 1을 수행하고, 거짓이면 조건식 2를 비교하여 참이면 문장 2를 수행하고, 거짓이면 조건식 3을 비교하여 문장 3을 수행한다.

형식	순서도
<pre> if(조건식 1) 문장 1; else if(조건식 2) 문장 2; else if(조건식 3) 문장 3; : else 문장 n; </pre>	

예제 다중 if 문을 사용하여 조건의 참과 거짓을 판단하는 프로그램을 작성해 보자.

프로그램

```

01  #include <stdio.h>
02  int main( )
03  {
04      int standard = 173, height = 168;
05      if (standard > height)
06      {
07          printf("평균 이하");
08      }
09      else if (standard==height)
10      {
11          printf("평균");
12      }
13      else
14      {
15          printf("평균 이상");
16      }
17      return 0;
18  }
        
```

5번 라인의 조건이 참이기 때문에 7번 라인이 실행돼!

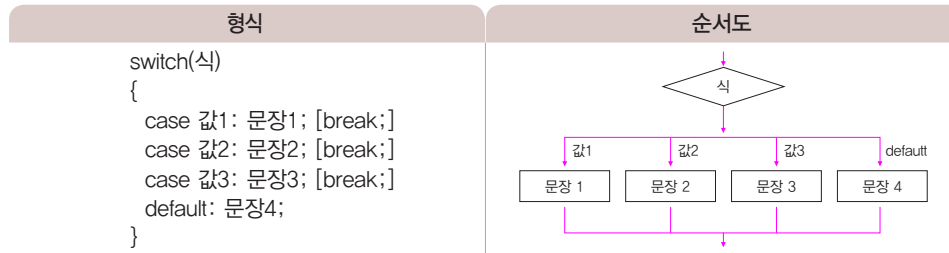


실행 결과

평균 이하

4 switch - case 문

if 문에서 비교해야 할 조건이 많으면 다중 if 문을 사용하여 이를 해결할 수 있지만, 이 경우 프로그램의 가독성이 떨어질 수 있다. 이때 **switch-case 문**을 사용하면 하나의 조건으로 여러 개의 case 문장 중 하나를 선택하여 실행할 수 있기 때문에 프로그램이 직관적이고 간결해진다.



switch-case 문의 조건

- switch 문 뒤의 식에는 정수값, 정수형 변수, 정수형 수식, 문자형 상수 등이 올 수 있다.
- 다수의 조건 중 만족하는 case 문 이후의 문장을 모두 수행한다.
- 조건에 만족하는 case 문의 문장만 수행하려면 반드시 break 문을 추가해야 하며, 그렇지 않으면 이후의 문장이 모두 수행된다.

예제 switch-case 문을 이용하여 점수(1~100점)를 입력하면, 점수에 해당하는 학점을 출력하는 프로그램을 작성해 보자.

프로그램

```
01 #include <stdio.h>
02 int main( )
03 {
04     int score;
05     char grade;
06     puts("점수를 입력하세요(1~100)");
07     putchar('>');
08     scanf("%d", &score);
09     switch (score / 10)
10     {
11         case 10:
12         case 9: grade = 'A'; break;
13         case 8: grade = 'B'; break;
14         case 7: grade = 'C'; break;
15         case 6: grade = 'D'; break;
16         default: grade = 'F';
17     }
18     printf("학점은 %c 입니다.", grade);
19     return 0;
20 }
```

실행 결과

```
점수를 입력하세요(1~100)
> 87 Enter
학점은 B 입니다.
```

3 반복 구조에 대해 알아보자

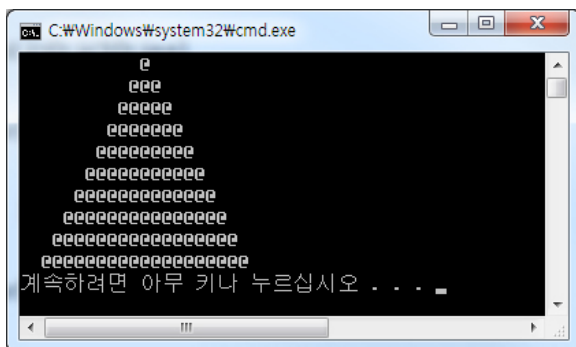
컴퓨터는 반복되는 많은 작업을 빠르고 정확하게 처리할 수 있다. C 언어는 조건에 따라 일련의 처리를 반복적으로 처리할 수 있는 반복문을 제공한다.



만약 10줄로 구성된 피라미드 모양을 나타내는 프로그램을 작성하려면 어떻게 해야 할까? 이때 사용해야 하는 것이 **반복문**이다. 효율적으로 반복문을 이용하면 프로그램의 길이도 줄일 수 있고, 잘 설계된 반복 알고리즘을 통해 프로그램의 실행 속도 또한 개선할 수 있다.

다음은 '@' 문자로 피라미드 모양을 출력하는 프로그램이다.

실행 결과



프로그램

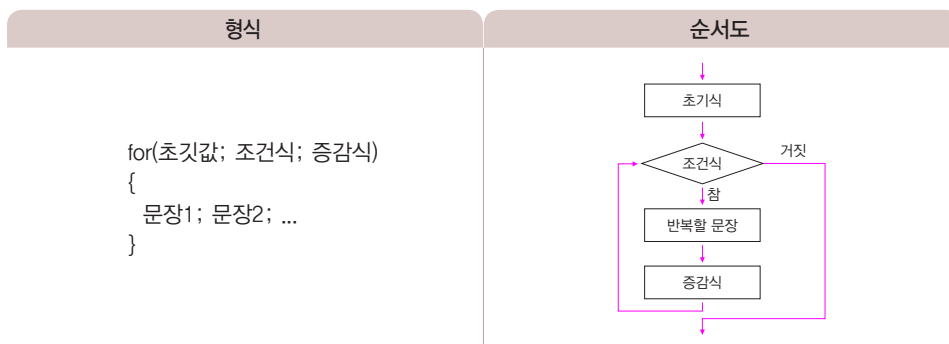
```
01 #include <stdio.h>
02 int main( )
03 {
04     int i, j, k;
05     for (i = 0; i < 10; i++)
06     {
07         for (j = 0; j <= 10 - i; j++)
08         {
09             putchar(' ');
10         }
11         for (k = 0; k <= 2*i; k++)
12         {
13             putchar('@');
14         }
15         puts("");
16     }
17     return 0;
18 }
```

4 반복 구조의 종류에 대해 알아보기

C 언어의 반복 구조에는 for 문, while 문, do-while 문과 같은 반복문이 있다. 일반적으로 반복을 예측할 수 있을 때에는 for 문을 사용하고, 반복 횟수를 정확히 예측하기 어려울 때에는 while 문 또는 do-while 문을 사용한다.

1 for 문

for 문은 가장 많이 사용하는 반복 명령문 중 하나로, 반복 횟수를 정확히 알고 있을 때 사용하면 편리하다.



<for 문 실행 순서>

- ① 초기식으로 변수값을 초기화한다.
- ② 변수값을 조건식과 비교하여 참이면 for 문의 문장을 수행하고, 거짓이면 반복문을 벗어난다.
- ③ 증감식에서 변수값을 증가(또는 감소)한 후 다시 ②로 이동하여 수행한다.

예제 for 문을 이용하여 구구단 2단을 출력하는 프로그램을 작성해 보자.

프로그램

```

01 #include <stdio.h>
02 int main( )
03 {
04     int i;
05     for (i = 1; i < 10; i++)
06     {
07         printf("2 x %d = %d\n", i, i * 2);
08     }
09     return 0;
10 }
```

변수 i의 값이 1부터 9까지(10보다 작을 때까지) for 문의 몸체 부분이 9번 실행됨.

초기식, 조건식, 증감식은 알고리즘에 따라 생략할 수 있어요.



반복 횟수의 예측: for 문

프로그래머가 반복 횟수를 예측(예: 구구단)할 수 있을 때, 반복문 중에서 for 문을 사용한다.

실행 결과

```

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

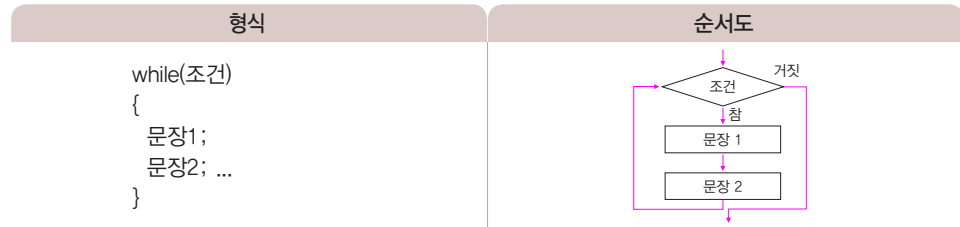
```


2 while 문

while 문의 조건

- while 문의 조건을 먼저 검사한 후 반복 여부를 결정한다.
- while 문 안에는 실행 조건의 결과를 바꿀 수 있는 문장이 있어야 한다.

while 문은 정확히 몇 번 반복되는지 알 수 없을 때 사용하는 반복문의 하나로, 주어진 조건이 참이면 반복을 수행하고, 거짓이면 while 문을 벗어난다.



예제 while 문을 이용하여 정수를 입력받아 화면에 출력하는 과정을 반복하는 프로그램을 작성해 보자(단, 숫자 0을 입력하면 종료한다.).

실행 결과 예

```
2 Enter
2 입력
0 Enter
0 입력
```

프로그램

```
01 #include <stdio.h>
02 int main( )
03 {
04     int answer = 1;
05     while (answer != 0)
06     {
07         scanf("%d", &answer);
08         printf("%d 입력\n", answer);
09     }
10     return 0;
11 }
```

사용자가 숫자 0을 입력하였을 때, 5번 라인의 조건이 거짓이 되어 while 문의 몸체를 실행하지 않고 10번 라인을 실행해.



^^로 해결하기

1부터 키보드로 입력한 숫자까지 차례로 출력하는 프로그램을 작성할 때, **실행 결과**를 보고 빈칸에 들어갈 코드를 완성해 보자.

프로그램

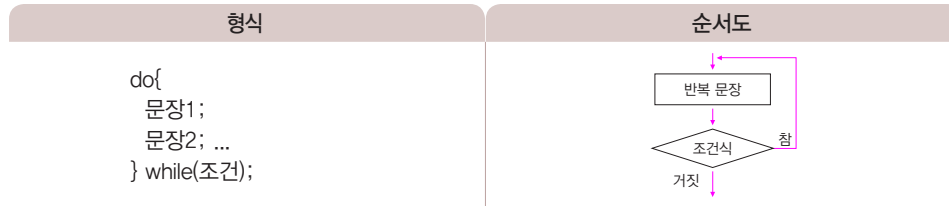
```
01 #include <stdio.h>
02 int main( )
03 {
04     int i=1, end;
05     printf("수를 입력하세요: ");
06     scanf("%d", &end);
07     _____
08     printf("%d ", i); i++;
09 }
10 return 0;
11 }
```

실행 결과

수를 입력하세요: 5 Enter
1 2 3 4 5

3 do-while 문

do-while 문은 기본적으로 while 문과 같다. 그러나 while 문과의 차이점은 참·거짓과 관계없이 무조건 한 번은 do-while 문을 실행한 후, 조건을 검사하여 참이면 반복문을 계속 수행하고, 거짓이면 do-while 문을 벗어난다는 점이다.



예제 do-while 문을 이용하여 키보드로 정수를 입력받아 출력하는 과정을 반복하는 프로그램을 작성해 보자(단, 숫자 0을 입력하면 종료한다.).

프로그램

```

01  #include <stdio.h>
02  int main( )
03  {
04      int answer;
05      do {
06          scanf("%d", &answer);
07          printf("%d 입력\n", answer);
08      } while (answer != 0);
09      return 0;
10  }
```

만약 answer 변수값이 0이면
반복문은 실행되지 않음.

실행 결과 예

```

2 [Enter]
2 입력
0 [Enter]
0 입력

```

스스로 해결하기

소문자를 입력하면 대문자로 출력하는 프로그램을 작성해 보자(단, 사용자가 q 문자를 입력하면 프로그램이 종료된다.).

프로그램

```

01  #include <stdio.h>
02  int main( )
03  {
04      int c;
05      do {
06          c=getchar( );
07          getchar( );
08          printf("입력한 소문자 %c는 대문자 %c입니다.\n", c, c - 32);
09      } while(c != 'q');
10      return 0;
11  }
```

실행 결과 예

```

a [Enter]
입력한 소문자 a는 대문자 A입니다.
c [Enter]
입력한 소문자 c는 대문자 C입니다.
q [Enter]
입력한 소문자 q는 대문자 Q입니다.


```

5 break 문, continue 문에 대해 알아보자

break 문과 continue 문은 조건문이나 반복문 수행 중에 해당 범위를 벗어나거나 이후의 문장을 무시하고 반복문의 처음으로 이동하여 실행하는 명령문이다.

1 break 문

break 문은 switch-case 문과 같은 조건문이나 반복문을 수행하다가 해당 영역을 벗어날 때 사용한다. 여러 개의 반복문이 쌓여 있을 때에는 가장 가까운 반복문을 벗어난다.

 **예제** break 문을 이용하여 정수를 입력받아 출력하는 과정을 반복하는 프로그램을 작성해 보자(단, 숫자 0을 입력하면 종료한다.).

프로그램

```
01 #include <stdio.h>
02 int main( )
03 {
04     int answer;
05     while (1)
06     {
07         scanf("%d", &answer);
08         if (answer == 0)
09         {
10             break;
11         }
12         else {
13             printf("%d 입력\n", answer);
14         }
15     }
16     return 0;
17 }
```

while 문의 조건에 1을 넣어서 무한 반복 수행됨.

실행 결과 예

```
2 Enter
2 입력 Enter
0 Enter
```

2 continue 문

반복문 안에서 **continue** 문을 만나면 반복문의 처음으로 프로그램의 제어를 이동한다. while 문에서는 다시 조건을 검사하게 되고 for 문에서는 증감문으로 제어한다.

예제 사용자가 키보드로 정수를 입력하면 1~100까지의 수 중에서 입력한 수의 배수가 몇 개인지 출력하는 프로그램을 작성해 보자.

프로그램

```
01 #include <stdio.h>
02 int main( )
03 {
04     int i, count = 0, answer;
05     scanf("%d", &answer);
06     for (i = 1; i < 101; i++)
07     {
08         if (i % answer != 0)
09         {
10             continue;
11         }
12         count++;
13     }
14     printf("1부터 100까지 수 중에서 %d의 배수는 %d개입니다.",
15           answer, count);
16     return 0;
17 }
```

변수 i를 answer로 나눈 나머지가 0과 같지 않으면 for 문의 중심으로 제어가 이동함.

실행 결과 예

7 Enter
1부터 100까지 수
중에서 7의 배수는
14개입니다.

continue 문을 만나면 continue 문 이후의 문장을 실행하지 않고, 반복문의 처음으로 이동해.



^^로 해결하기

다음은 숫자 1~10까지를 출력하는 프로그램이다. **실행 결과**를 보고 빈칸에 코드를 채워 프로그램을 완성해 보자(단, 숫자 7은 출력하지 않는다.).

프로그램

```
01 #include <stdio.h>
02 int main( )
03 {
04     int i=0;
05     while(i++ < 10) {
06         _____
07         printf("%d ", i);
08     }
09     return 0;
10 }
```

실행 결과

1 2 3 4 5 6 8 9 10

6 중첩 제어 구조를 사용한 프로그램을 살펴보자

중첩 제어문이란 제어문 안에 또 다른 제어문이 1개 이상 포함된 것을 말한다. 구구단 중 2단만을 출력할 때에는 2×1 에서 2×9 까지 1씩 증가하는 곱한 값을 구할 수 있도록 for 문은 1개만 필요하다. 하지만 구구단을 2단에서 3단까지 출력하기 위해서는 단을 변화시킬 for 문이 하나 더 있어야 한다.

예제 중첩 제어문을 이용하여 구구단 2, 3단을 출력하는 프로그램을 작성해 보자.

실행 결과

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
```

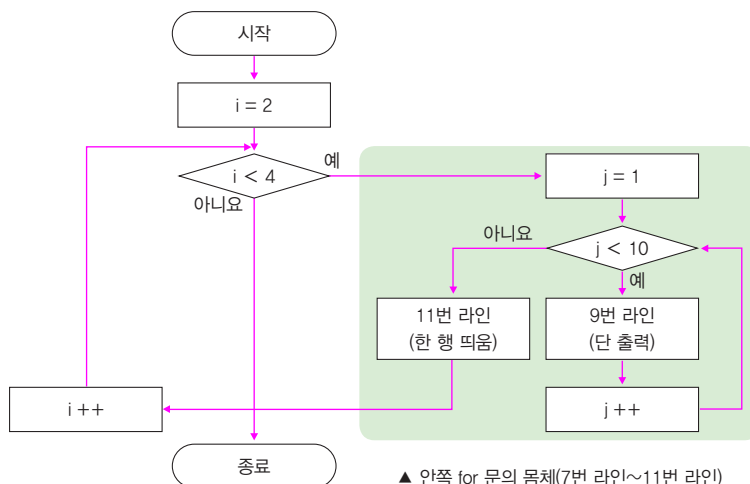
```
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
```

프로그램

```
01 #include <stdio.h>
02 int main( )
03 {
04     int i,j;
05     for (i = 2; i < 4; i++)
06     {
07         for (j = 1; j < 10; j++)
08         {
09             printf("%d x %d = %2d\n", i, j, i * j);
10         }
11         printf("\n");
12     }
13     return 0;
14 }
```

i < 10으로 바꾸면 2~9단까지 출력

단과 단 사이를 구분하기 위해 한 행을 띄움.



[그림 II-3] 구구단 프로그램의 순서도

미션

해결하기

두 수를 입력받아 처음 수부터 마지막 수까지의 누적합을 구하는 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int start, end, tempStart, sum=0;
05      printf("두 수를 입력하세요: ");
06      scanf("%d %d", &start, &end);
07      tempStart = start;
08      for (;start<end+1;start++)
09      {
10          sum += start;
11      }
12      printf("%d부터 %d까지의 합은 %d입니다.", tempStart, end, sum);
13      return 0;
14  }
```

실행 결과 예)

두 수를 입력하세요: 1 100
1부터 100까지의 합은 5050입니다.

스스로 해결하기

다음은 소수를 판별하기 위한 프로그램이다. 빈칸에 코드를 채워 프로그램을 완성해 보자(단, 소수는 1과 자신의 수만으로 나누어지는 1보다 큰 수이다.).

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i, answer;
05      scanf("%d", &answer);
06      
07
08
09      if (answer == i) printf("소수입니다.");
10      else printf("소수가 아닙니다.");
11      return 0;
12  }
```

실행 결과 예)

29
소수입니다.



수행 평가

중첩 제어 구조를 이용하여 완전수 판단 프로그램

활동 목표 중첩 제어 구조를 이용하여 각 수의 약수를 더해 완전수를 판단하는 프로그램을 작성할 수 있다.

완전수란 자신을 제외한 약수의 합으로 나타낼 수 있는 수이다. 예를 들면, 6의 약수는 1, 2, 3, 6이고 자신(6)을 제외한 약수의 합(1+2+3)이 자신과 같은 수일 때, 이 수를 완전수라고 한다.

1부터 10000 사이의 완전수를 구하는 프로그램이다.

다음의 실행 결과를 보고, 빈칸에 코드를 채워 프로그램을 완성해 보자.

프로그램

```

01  #include <stdio.h>
02  int main()
03  {
04      int i, j, sum=0, count=0;
05      for (i = 1; i < 10000; i++) {
06          for (j = 1; j < i; j++) {
07              if (i%j == 0) {
08                  sum += j;
09              }
10          }
11          if (  ) printf("%d번째 완전수: %d\n", ++count, i);
12          sum = 0;
13      }
14      return 0;
15  }
```

실행 결과

1번째 완전수: 6
 2번째 완전수: 28
 3번째 완전수: 496
 4번째 완전수: 8128

스스로 평가하기

평가 항목	구분		
	그렇다	보통이다	그렇지 않다
• 선택 구조를 프로그램에 적용할 수 있다.			
• 반복 구조를 프로그램에 적용할 수 있다.			



내 실력 확인하기

내용을 이해했나요?

- 순차 구조: 한 방향으로 쭉 뻗은 직선처럼 정해진 순서에 따라 차례대로 처리 내용을 실행하는 구조이다.
- 선택 구조: 구조화되어 있는 순서도에서 어떤 조건에 따라 다른 선택의 길을 제공하는 구조이며, if 문, if-else 문, 다중 if 문, switch-case 문 등이 있다.
- 반복 구조: 프로그램이 수행될 때 제어 조건이 참이면 프로그램 일부를 반복적으로 수행하고, 거짓이면 반복문을 벗어나는 구조이며, for 문, while 문, do-while 문 등이 있다.
- 중첩 제어 구조: 제어 구조 안에 또 다른 제어 구조가 1개 이상 포함된 것을 말한다.

문제로 확인할까요?

1. 다음은 바깥 for 문의 i 값이 안쪽 for 문의 조건 부분에 영향을 주는 프로그램이다. 실행 결과를 작성해 보자.

```

01  #include <stdio.h>
02  int main( )
03  {
04      int i, j, count=1;
05      for (i = 0; i < 5; i++)
06      {
07          for (j = 0; j < i; j++)
08          {
09              printf("%d ", count++);
10          }
11          printf("\n");
12      }
13      return 0;
14  }
```

실행 결과

평가해 볼까요?

★ 다음 평가 항목에 따라 자신의 성취 척도를 스스로 점검해 보자.

영역	평가 항목	척도				
		1	2	3	4	5
이해	순차 구조, 선택 구조, 반복 구조의 개념을 이해하고 있는가?					
적용	순차 구조, 선택 구조, 반복 구조를 프로그램에 적용할 수 있는가?					