



Chapter 03. 제어문

목차

1. 제어문의 이해
2. 선택문
3. 반복문
4. 보조 제어문
5. 무한 루프

학습목표

- 프로그램의 흐름을 변경하기 위한 제어문을 학습한다.
- 조건에 따라 한 문장만 선택적으로 수행하기 위한 if문, else if문, switch문의 사용법을 학습한다.
- 특정 문장을 반복적으로 수행하는 for문, while문, do~while문의 사용법을 학습한다.
- 보조 제어문인 break문의 사용법을 학습한다.

01 제어문의 이해

[표 3-1] 제어문의 종류

구문	제어 명령
선택문	if문
	if~else문
	다중 if~else문
	switch문
반복문	for문
	while문
	do~while문
보조 제어문	break문
	continue문

02 선택문

- 선택문은 조건에 따라 판단해서 원하는 문장만 수행하고자 할 때 사용한다. 선택문에는 if문, if~else문, 다중 if~else문, switch문이 있다.

■ if문

- if문은 주어진 조건을 만족하는 경우에만 특정 문장을 수행하도록 하는 제어문이다. 조건의 결과가 참이면 if문 바로 다음에 나오는 문장을 수행하고, 결과가 거짓이면 이 문장을 수행하지 않고 바로 다음 문장을 수행하게 된다.

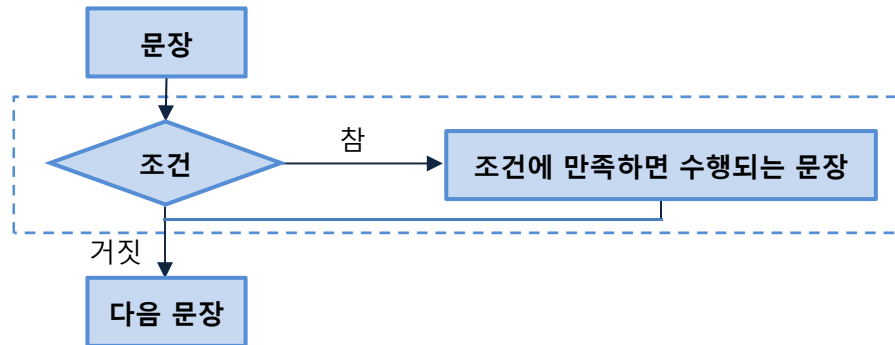
if문 기본 형식

[문장]

```
if(조건){  
    조건에 만족하면 수행되는 문장;  
}
```

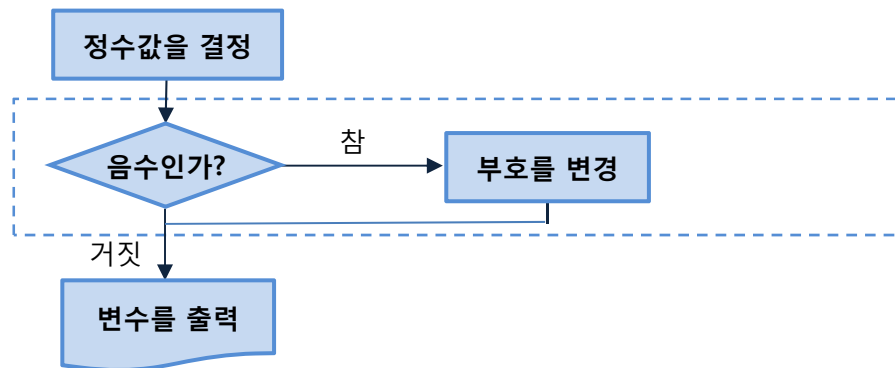
[다음 문장]

02 선택문



[그림 3-1] if문의 순서도

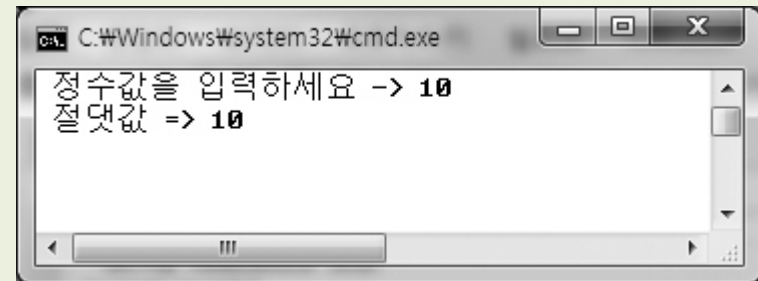
- if문을 사용해서 절댓값을 구하는 프로그램을 예제로 작성해 보자.



[그림 3-2] 절댓값을 구하는 프로그램의 순서도

예제 3-1. if문을 사용해서 절댓값 구하기(03_01.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int x;
06
07     cout << " 정수값을 입력하세요 -> ";
08     cin >> x;
09
10     if(x < 0) // 음수일 경우에만
11         x = -x; // 부호 변경
12
13     cout << " 절댓값 => " << x << "\n";
14 }
```



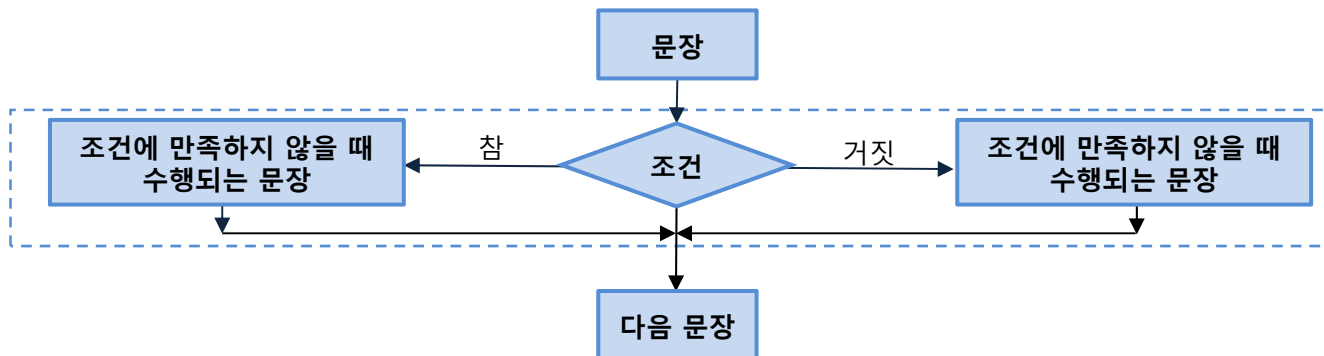
02 선택문

■ if-else문

- if문 계열 중 가장 많이 사용하는 형식으로, 2가지 경우 중 한 가지만 선택할 때 사용한다.

if~else문 기본 형식

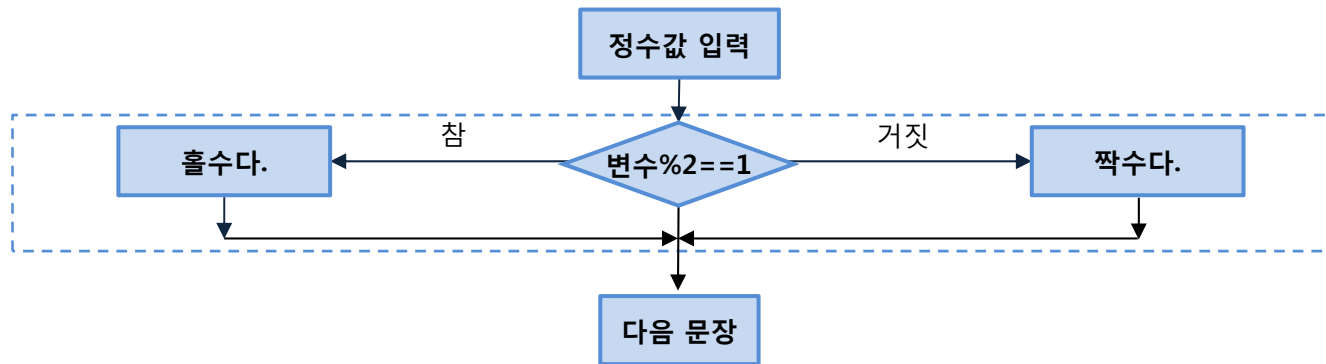
```
[문장]
if(조건){
    조건에 만족할 때 수행되는 문장;
}
else{
    조건에 만족하지 않을 때 수행되는 문장;
}
[다음 문장]
```



[그림 3-1] if~else문의 순서도

02 선택문

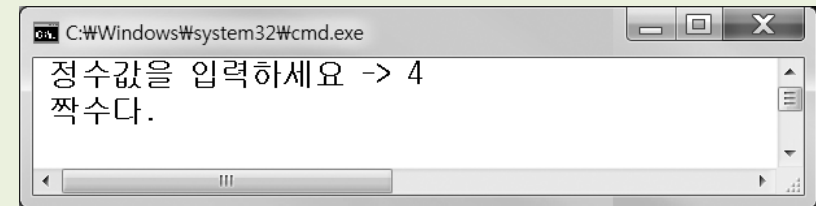
- if~else문을 사용해서 주어진 정수형 데이터가 짝수인지 홀수인지 판별하는 프로그램을 작성해 보자.



[그림 3-2] 짝댓값을 구하는 프로그램의 순서도

예제 3-2. if~else문을 사용해서 짝수 홀수 판별하기(03_02.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int x;
06
07     cout << " 정수값을 입력하세요 -> ";
08     cin >> x;
09
10     if(x % 2 == 1) // 2로 나누어 나머지가 1이면
11         cout << " 홀수다. \n";
12     else
13         cout << " 짝수다. \n";
14 }
```



02 선택문

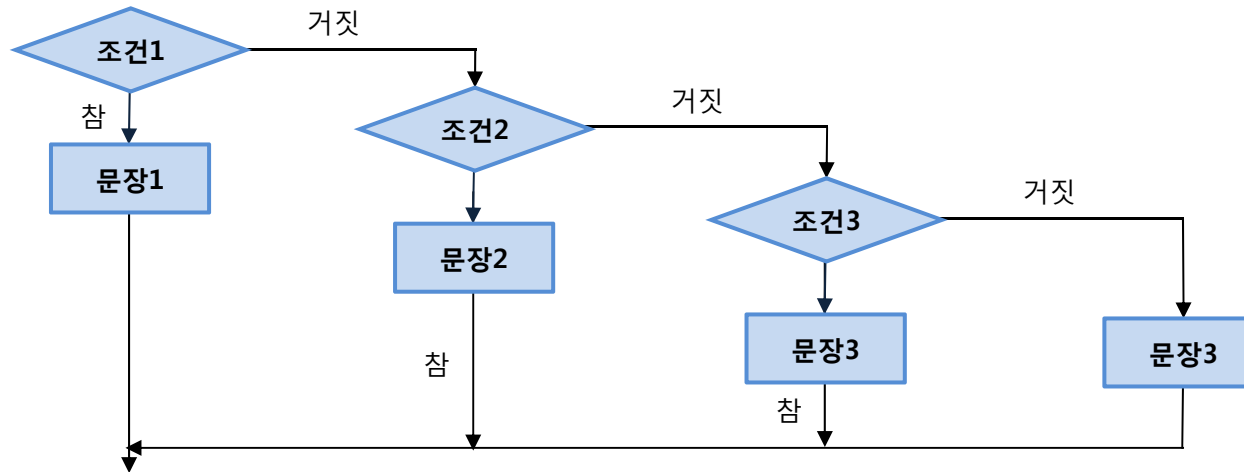
■ 다중 if-else문

- if~else문은 참, 거짓을 선택하는 과정에서 한 번만 사용되었지만, 만일 그 경우의 수가 둘이 아닌 셋 이상에서 하나를 선택해야 할 경우에는 if~else문을 중첩해서 사용해야 한다.

다중 if~else문 기본 형식

```
[문장]
if(조건1){
    조건1에 만족할 때 처리할 문장;
}
else if(조건2){
    조건1에 만족하지 않지만 조건2에 만족할 때 처리할 문장;
}
...
else if(조건n){
    조건1부터 조건n-1에 만족하지 않지만 조건n에 만족할 때 처리할 문장;
}
else{
    위에서 언급한 모든 조건에 대해서 만족하지 않을 때 처리할 문장;
}
[다음 문장]
```

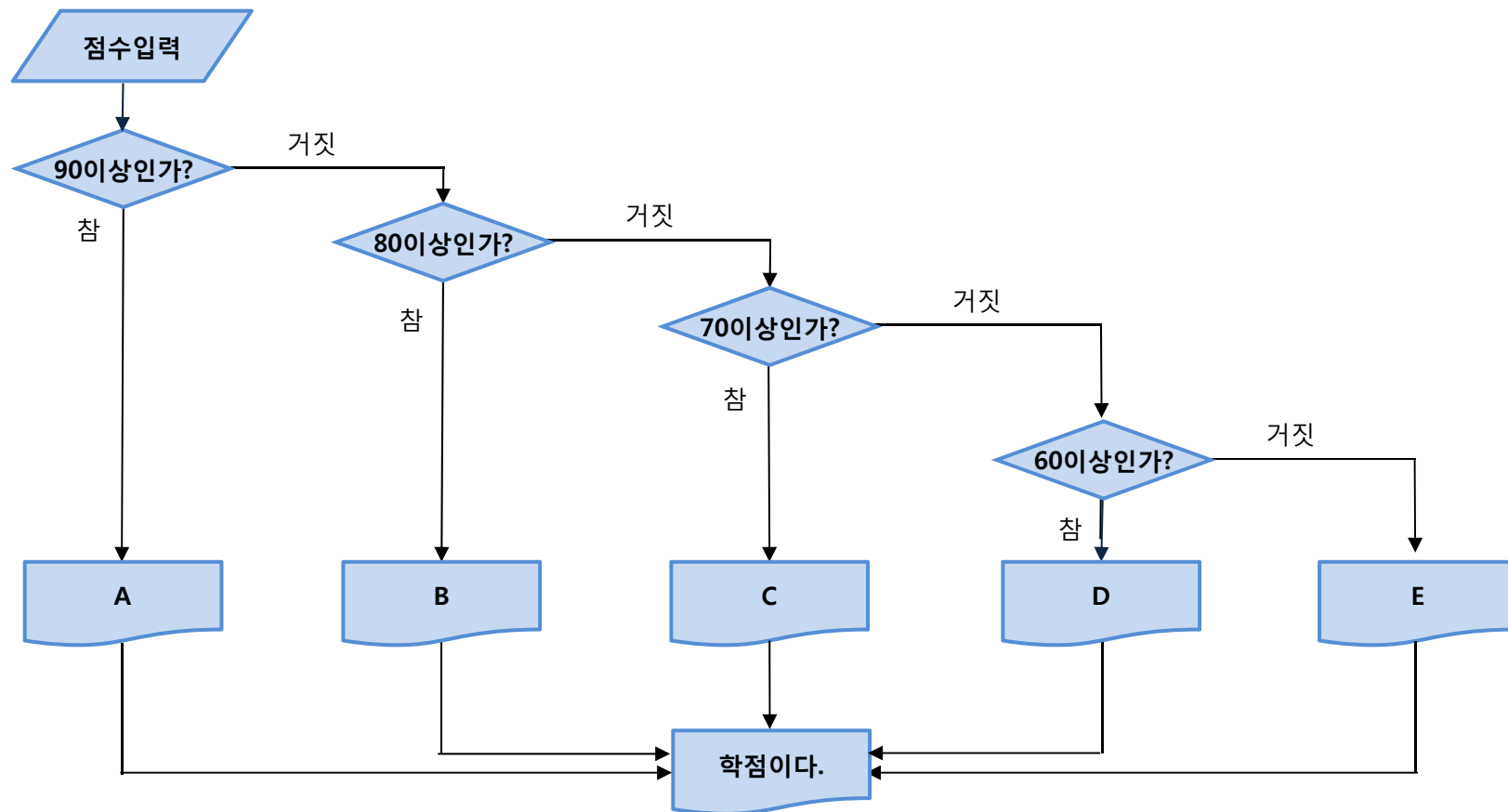
02 선택문



[그림 3-5] 다중 if~else문의 순서도

02 선택문

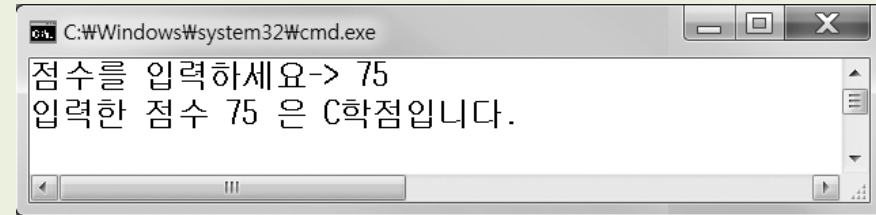
- 다중 if~else문을 사용해서 점수를 입력받아 학점을 구하는 프로그램을 작성해 보자.



[그림 3-6] 학점을 구하는 프로그램의 순서도

예제 3-3. 다중 if~else문을 이용해서 학점 계산하기(03_03.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int score; // 입력받은 점수를 저장할 변수
06     char grade; // 구한 학점을 저장할 변수
07     cout<<"점수를 입력하세요 -> ";
08     cin>>score;
09     // 조건 검사
10     if(score>=90) // score가 90이상이나?
11         grade='A'; // 만족하면 grade='A'
12     else if (score>=80) // 아니면 score가 80이상이나?
13         grade='B'; // 만족하면 grade='B'
14     else if (score>=70) // 아니면 score가 70이상이나?
15         grade='C'; // 만족하면 grade='C'
16     else if (score>=60) // 아니면 score가 60이상이나?
17         grade='D'; // 만족하면 grade='D'
18     else // 아니면
19         grade='F'; // grade='F'
20     cout<<"입력한 점수 " <<score<<" 은 "<<grade<<"학점입니다.\n";
21 }
```



02 선택문

■ Switch문

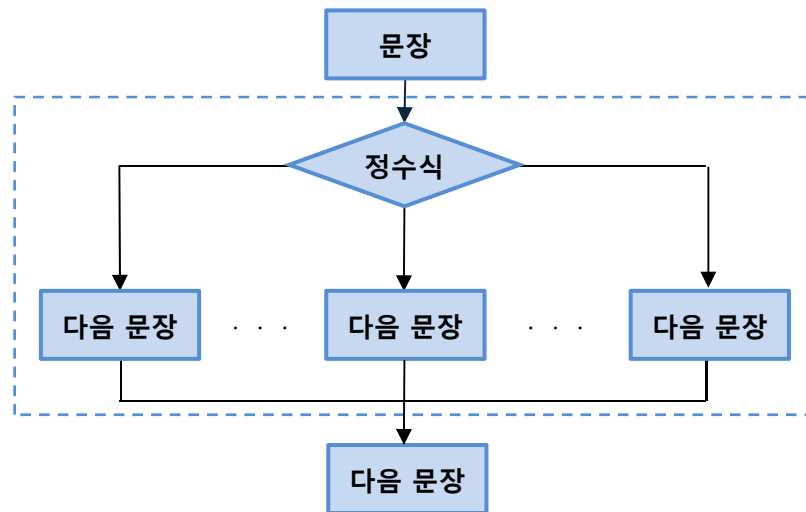
- switch문은 다중 if-else문과 같은 용도로 쓰이나 다중 if-else문이 참이나 거짓의 결과를 문장의 행태로 수행하는 반면 switch문은 정수식에 따라 분기한다.

switch문 기본 형식

```
문장1;  
Switch(정수식) {  
    case 정수값1 : 문장2; break;  
    case 정수값2 : 문장3; break;  
    ...  
    case 정수값n : 문장n; break;  
    ...  
    default: 문장m;  
}  
다음 문장;
```

- Switch문에서는 break문을 생략할 수 있다. break문이 없는 case문은 if문에서 논리합 연산자(||)와 유사한 기능을 하기 때문에 이를 이용하면 더욱 간략하게 프로그램을 만들 수 있다.

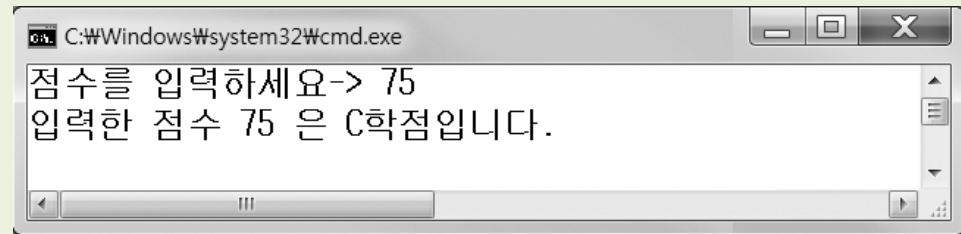
02 선택문



[그림 3-7] switch문의 순서도

예제 3-5. switch문으로 학점 판별하기(03_05.cpp)

```
01 #include<iostream>
02 using namespace std;
03 void main()
04 {
05     int score;    // 입력받은 점수를 저장할 변수
06     char grade; // 구한 학점을 저장할 변수
07     cout<<"점수를 입력하세요 -> ";
08     cin>>score;
09
10     switch(score/10) {    // 결과가 정수로 나오는 산술식
11     case 10 : grade='A'; break;
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     cout<<"입력한 점수 " <<score<<" 은 "<<grade<<"학점입니다.\n";
19 }
```



03 반복문

- 반복문이란 특정 부분의 문장을 반복해서 수행하게 하는 문장을 말한다. 반복문에는 for문, while문, do~while문이 있다.

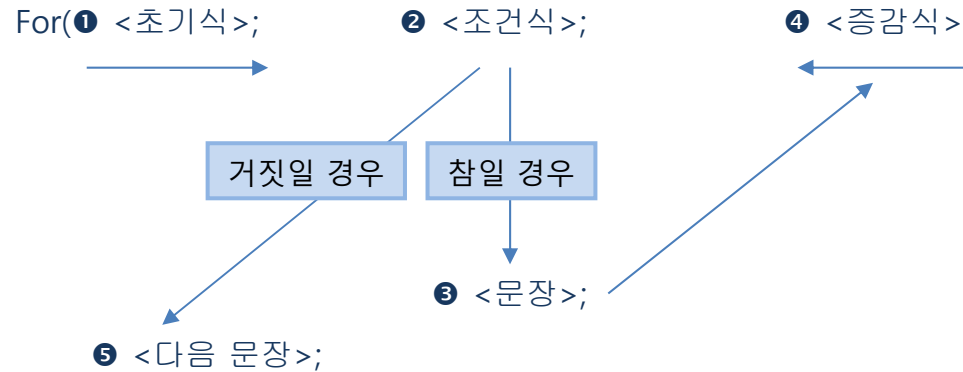
■ for문

- 지정된 횟수만큼 반복하는 for문은 다음과 같이 <초기식>, <조건식>, <증감식>, <문장>으로 구성된다.

```
for(<초기식>;<조건식>;<증감식>) {  
    문장1;  
}
```

for문 기본 형식

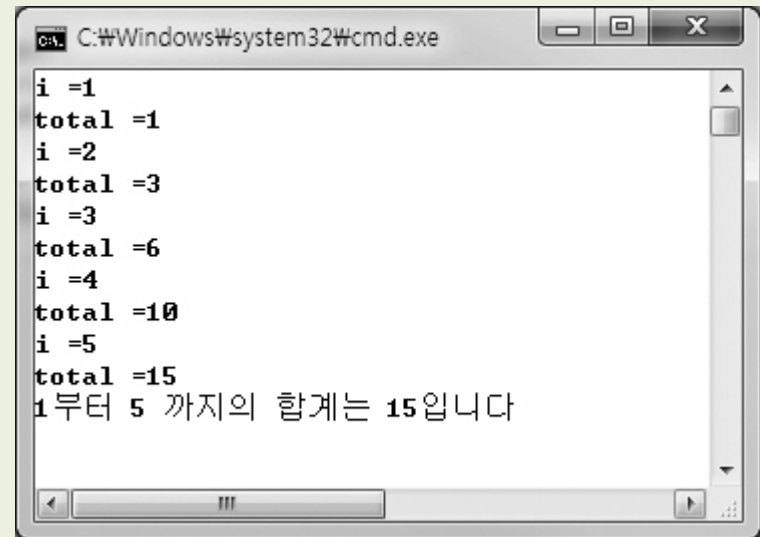
03 반복문



- ❶ 먼저 <초기식>에서 제어변수를 초기화하고
- ❷ <조건식>에서 조건이 참인지 거짓인지를 검사한다.
- ❸ <조건식>이 참이면 블록 안의 <문장>을 수행하고,
- ❹ <증감식>에 따라 제어변수를 증가시키거나 감소시킨다.
- ❷ <증감식>에 의해 변경된 값이 적용된 <조건식>이 참인지 검사한다.
- ❺ 제어변수의 값을 적용한 <조건식>이 거짓이면 <문장>을 수행하지 않고 for문을 빠져 나와서 <다음 문장>이 수행된다.

예제 3-7. for문을 이용해서 1부터 5까지의 합계 구하기(03_07.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int total=0; // 반드시 초기화해야 한다.
06     int i;
07     for(i=1;i<=5;i++) {
08         cout<<"i ="<<i<<endl;
09         total+=i; // total=total+i;
10         cout<<"total ="<<total<<endl;
11     }
12     cout<<"1부터 " << i-1 <<" 까지의 합계는 "
13     << total <<"입니다"<<endl;
14 }
```



```
C:\Windows\system32\cmd.exe
i =1
total =1
i =2
total =3
i =3
total =6
i =4
total =10
i =5
total =15
1부터 5 까지의 합계는 15입니다
```

03 반복문

■ 다중 for문

- for문 안에 for문을 포함해서 사용하는 것을 다중 for문이라고 한다.

```
int out, in;

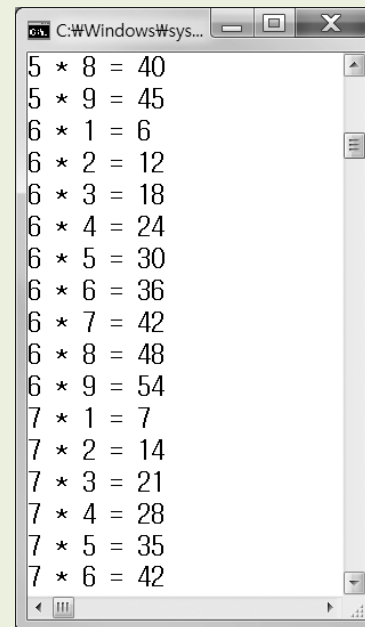
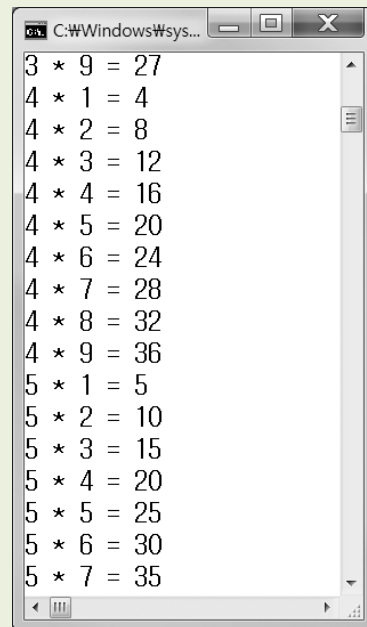
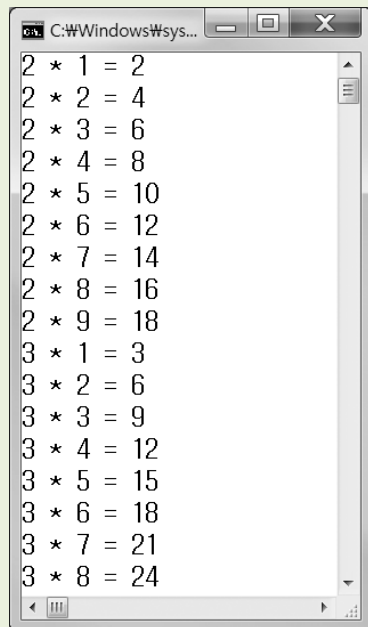
cout<< "out:시침 \t >> \t in:분침";
cout<< "바깥쪽 제어변수 \t >> \t 안쪽 제어변수";

for (out = 1; out <= 3; out++){ ----- ❶
    // 문장1;
    for (in = 1; in <= 5; in++){ ----- ❷
        cout<< out << "\t\t >> \t " << in; // 문장2;
    }
    // 문장3;
}
```

- 문장1과 문장3은 바깥쪽에 기술된 for문(❶)의 반복 횟수만큼만 반복 수행된다. 하지만 문장2는 (바깥쪽 for문의 반복 횟수) * (안쪽 for문의 반복 횟수) 만큼 반복한다.

예제 3-10. 다중 for문을 이용해서 특정 구구단 출력하기(03_10.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     for(int dan=2; dan<=9; dan++) // 바깥 for문
06         for(int j=1; j<10; j++) // 안쪽 for문
07             cout << dan << " * " << j << " = " << dan*j << "\n";
08 }
```



...

03 반복문

■ while문

- while문은 조건이 만족하는 동안 문장을 반복 수행한다.

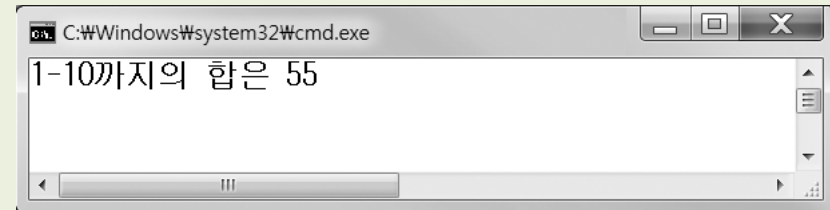
for문과 비교해보면 <초기식>과 <증감식>이 없고 <조건식>만 있는 형태다.

```
i = 1; .....①
while (i <= 10) {...②
    total += i;.....③
    i++; .....④
}
```

- 조건식(②)을 만족하면 문장(③, ④)을 수행하고, 문장 수행 후에 다시 조건식(②)을 검사해 조건이 맞으면 문장(③, ④)을 수행하는 과정을 반복하며, 조건식이 거짓이 되면 문장을 수행하지 않고 while문 밖으로 빠져 나온다.

예제 3-11. while문을 사용해서 1부터 10까지의 합 구하기(03_11.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int total=0;
06     int i=1;        // for문의 초기식에 해당
07     while(i<=10) { // for문의 조건식에 해당
08         total+=i;
09         i++;        // for문의 증감식에 해당
10     }
11     cout<<"1-10까지의 합은 "<<total<<"\n";
12 }
```



03 반복문

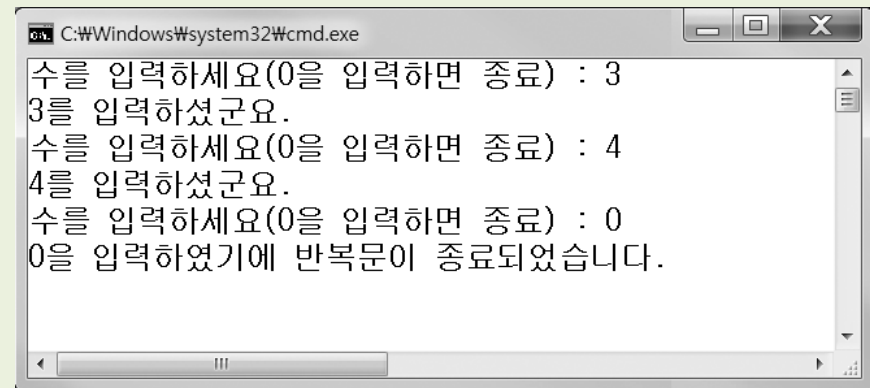
■ do~while문

- do~while문은 while문과 비슷하지만 while문 앞에 do라는 지정어가 추가되는 것이 다르다. 즉, 반복되는 문장을 일단 한번은 실행하고 그 다음에 조건을 검사해 조건이 참이면 계속 반복하고 거짓이면 while문을 빠져 나온다.

do~while문 기본 형식	do~while문 사용 예
<pre>do { 문장 } while(조건식);</pre>	<pre>do{ cout<<"수를 입력하세요(0을 입력하면 종료) : "; cin>>num; cout<< num <<"를 입력하셨습니다.Wn"; } while(num!=0);</pre>

예제 3-13. do~while문으로 좀 더 간결하게 프로그램 작성하기 (03_13.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int num;
06     do{
07         cout<<"수를 입력하세요(0을 입력하면 종료) : ";
08         cin>>num;
09         cout<< num <<"를 입력하셨습니다.\n";
10     } while(num!=0);
11     cout<< num << "을 입력하였기에 반복문이 종료되었습니다.\n";
12 }
```

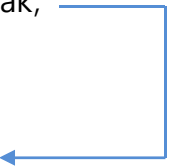


04 보조 제어문

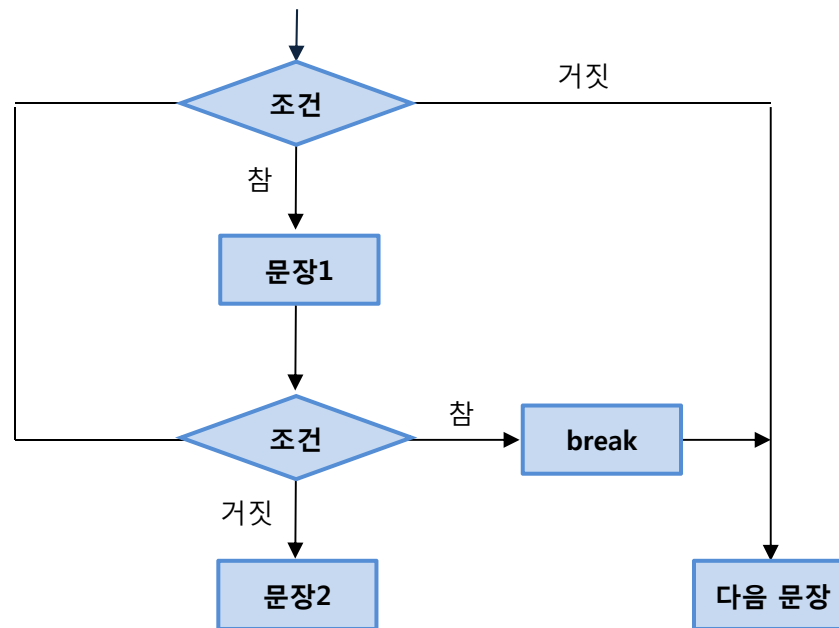
- 보조 제어문인 break문과 continue문은 반복문 등과 함께 사용해서 프로그램의 흐름을 좀 더 구체적으로 제어할 수 있다.

■ break문

- break문은 프로그램의 일부를 수행하지 않고 건너뛰게 해서 switch문, for문, while문, do~while문의 제어를 벗어나기 위해 사용할 수 있다.

break문 기본 형식	break문 사용 예
<pre>for(초기식; 조건식; 증감식) { 문장1; if(조건식) break; 문장2; } 문장3;</pre> 	<pre>for(i=1; i<=10; i++) { if(i%2==0) // i가 2로 나누어서 떨어지면 break // for문을 벗어남 total+=i; }</pre>

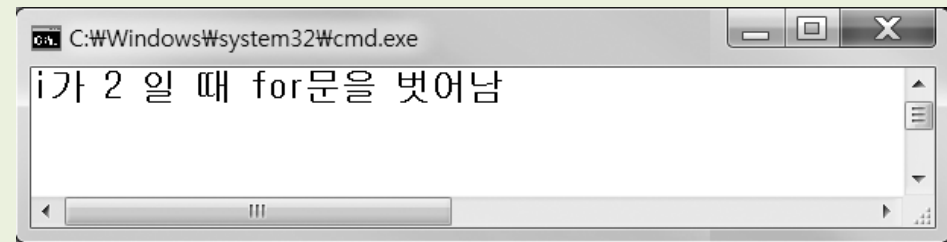
04 보조 제어문



[그림 3-8] break문의 순서도

예제 3-14. for문에서 반복 도중에 벗어나기(03_14.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int total=0;
06     int i;
07     for(i=1; i<=10; i++) {
08         if(i%2==0) // I가 2로 나누어서 떨어지면
09             break // for문을 벗어남
10         total+=i;
11     }
12     cout<<"i가"<< i <<" 일 때 for문을 벗어남\n"
13 }
```



05 무한 루프

- 무한 루프는 프로그램의 실행이 종료되지 않고 끝없이 수행되어지는 상태를 말한다.

■ for문을 이용한 무한 루프

- 다음은 가장 간단한 형태의 for문이다. for문은 세미콜론(;)만 2번 기술하면 문법적으로 문제가 없다. 하지만 반복문을 벗어날 조건을 기술하지 않았기 때문에 무한 루프에 빠지게 된다.

```
for( ; ) {  
    ...  
    ...  
}
```

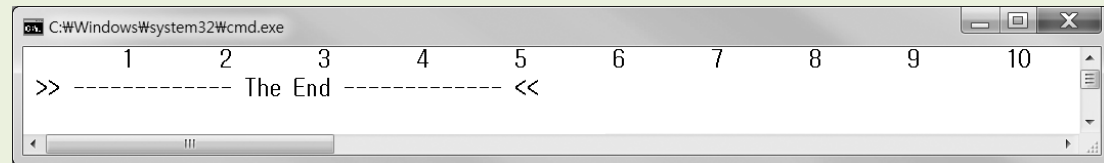
■ while문을 이용한 무한 루프

- while문은 조건식의 결과가 참이면 반복을 계속하고 거짓이면 반복문을 벗어난다. while문을 이용한 무한 루프는 일반적으로 조건식에 true를 기술해서 표현한다.

```
while( true ) {  
    ...  
    ...  
}
```

예제 3-17. 무한 루프에서 벗어나기 위한 break문(03_17.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int i=0;
06     for( ; ; ){
07         cout << "Wt" << ++i;
08         if(i%10==0)
09             break;          // 무한루프 탈출
10     }
11     cout << "Wn >> ----- The End ----- << Wn";
12 }
```



Homework

- Chapter 3 Exercise: 4, 5, 9, 10, 12, 14, 15, 17, 18, 21, 24, 27, 29