CHAPTER 04

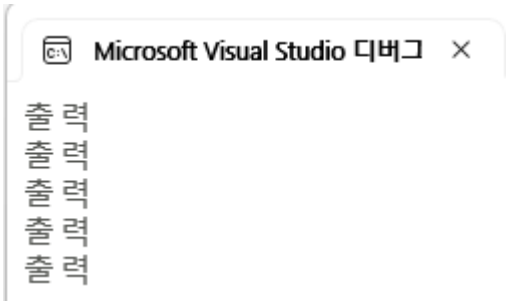
반복문

■ 반복문과 배열

- 복사해서 붙여넣기 기술을 사용하면 컴퓨터에게 반복을 명령할 수 있음

코드 4-1 복사해서 붙여넣기를 사용한 반복 /4장/LoopSummary

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine("출력");
04     Console.WriteLine("출력");
05     Console.WriteLine("출력");
06     Console.WriteLine("출력");
07     Console.WriteLine("출력");
08 }
```



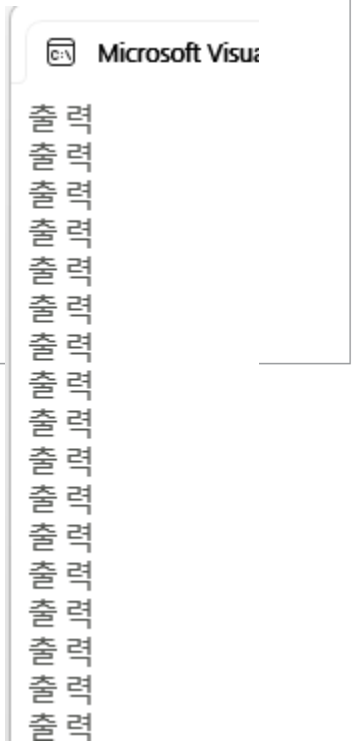
■ 반복문과 배열

- 반복문을 사용하면 훨씬 쉬움

코드 4-2 반복문을 사용한 반복 /4장/LoopSummary

```
01  static void Main(string[] args)
02  {
03      for (int i = 0; i < 1000; i++)
04      {
05          Console.WriteLine("출력");
06      }
07  }
```

1000번 반복



■ 기본적인 배열 생성 방법

▪배열은 다음과 같은 형식으로 생성

자료형[] 이름 = {자료, 자료}

그림 4-1 배열 선언 형식

코드 4-3 기본적인 배열 생성 방법 /4장/Arrays

```
01 static void Main(string[] args)
02 {
03     int[] intArray = { 52, 273, 32, 65, 103 };
04     long[] longArray = { 52, 273, 32, 65, 103 };
05     float[] floatArray = { 1.0F, 2.0F, 3.0F, 4.0F, 5.0F };
06     double[] doubleArray = { 1.0, 2.0, 3.0, 4.0, 5.0 };
07     char[] charArray = { '가', '나', '다', '라' };
08     string[] stringArray = { "윤인성", "연하진", "윤아린" };
09 }
```

■ 기본적인 배열 생성 방법

- 요소(Element): 배열 안에 들어있는 각각의 자료
- 배열의 요소에 접근할 때는 다음과 같이 대괄호[ ]를 사용
- 인덱스(Index): 대괄호 안에 넣는 숫자

그림 4-2 배열의 요소

■ 기본적인 배열 생성 방법

▪ 기본예제 3-2 홀수 짝수 구분(1)

/4장/ArrayBasic

- int 자료형의 배열을 만들고 배열의 요소에 접근하는 코드

코드 4-4 배열 생성하고 요소에 접근

```
01  static void Main(string[] args)
02  {
03      // 배열을 생성합니다.
04      int[] intArray = { 52, 273, 32, 65, 103 };
05
06      // 배열의 요소를 변경합니다.
07      intArray[0] = 0;
08
09      // 요소를 출력합니다.
10      Console.WriteLine(intArray[0]);
11      Console.WriteLine(intArray[1]);
12      Console.WriteLine(intArray[2]);
13      Console.WriteLine(intArray[3]);
14  }
```

실행 결과

```
0
273
32
65
```

여기서 잠깐!

■ Length 속성

- 배열의 Length 속성을 사용하면 배열에 몇 개의 요소가 있는지 확인할 수 있음

코드 4-5 배열의 Length 속성 /4장/Arrays

```
01 static void Main(string[] args)
02 {
03     // 배열을 생성합니다.
04     int[] intArray = { 52, 273, 32, 65, 103 };
05
06     // 배열의 길이를 출력합니다.
07     Console.WriteLine(intArray.Length);
08 }
```

실행 결과

```
5
```

여기서 잠깐!

■ IndexOutOfRangeException

- 만약intArray[5]를 사용하면 어떤 일이 일어날까?

코드 4-6 배열의 범위를 벗어나는 인덱스에 접근 /4장/Arrays

```
01 static void Main(string[] args)
02 {
03     // 배열을 생성합니다.
04     int[] intArray = { 52, 273, 32, 65, 103 };
05
06     // 요소를 출력합니다.
07     Console.WriteLine(intArray[5]);
08 }
```

실행 결과

```
처리되지 않은 예외: System.IndexOutOfRangeException: 인덱스가 배열 범위를 벗어났습니다.
…생략…
```

■ 원하는 크기의 배열 생성 방법

- 요소를 넣지 않고 일단 100개의 공간을 만들어두고 싶을 때는?

코드 4-7 원하는 크기의 배열 생성 방법 /4장/Arrays

```
01  int[] array = new int[100];
```

- 배열을 초기화하면 자료형의 기본 값으로 공간이 채워짐
- 일반적인 숫자 자료형은 0, 문자열 자료형은 빈 문자열, 이후에 알아보는 객체는 null로 초기화됨

■ 원하는 크기의 배열 생성 방법

▪기본예제 4-2 원하는 크기의 배열 생성 /4장/VariousArray

- 100개의 공간을 가지는 int 자료형의 배열을 생성

코드 4-8 특정한 크기의 배열 생성

```
01 static void Main(string[] args)
02 {
03     // 배열을 생성합니다.
04     int[] intArray = new int[100];
05
06     // 요소를 출력합니다.
07     Console.WriteLine(intArray[0]);
08     Console.WriteLine(intArray[99]);
09 }
```

실행 결과

```
0
0
```

while 반복문

- 가장 기본적인 반복문
- if 조건문과 달리 불_표현식이 참인 동안 중괄호 안의 문장을 계속 실행

```
while (불_표현식)
{
    // 불_표현식이 참인 동안 실행할 문장
}
```

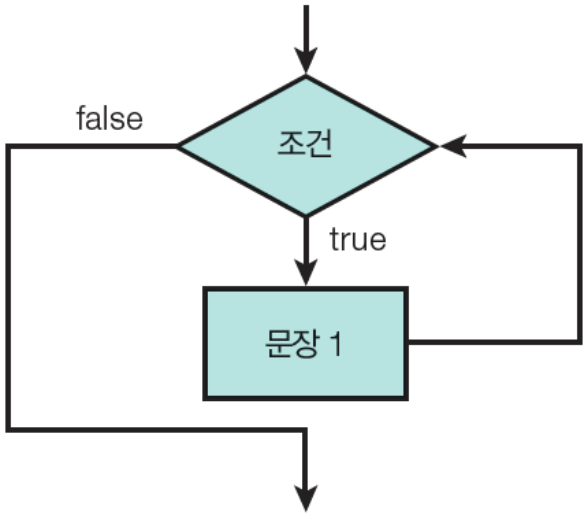


그림 4-3 while 반복문 형식과 순서도

02 while 반복문

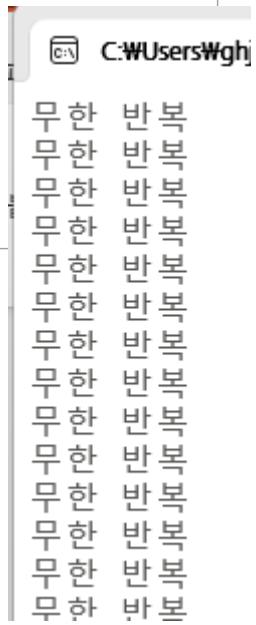
■ while 반복문

- 조건이 변하지 않으면 무한히 반복되므로 반드시 조건을 거짓으로 만들 수 있는 문장이 포함되어야 함

코드 4-9 무한 반복 /4장/Loops

```
01 static void Main(string[] args)
02 {
03     while (true)
04     {
05         Console.WriteLine("무한 반복");
06     }
07 }
```

무한 반복



02 while 반복문

■기본예제 4-3 while 반복문 /4장/WhileBasic

- 특정한 숫자를 증가시켜 불_표현식을 거짓으로 만들고 반복문을 벗어남

코드 4-10 while 반복문

```
01 static void Main(string[] args)
02 {
03     // 변수를 선언합니다.
04     int i = 0;
05     int[] intArray = { 52, 273, 32, 65, 103 };
06
07     // 반복을 수행합니다.
08     while (i < intArray.Length)
09     {
10         // 출력합니다.
11         Console.WriteLine(i + "번째 출력:" + intArray[i]);
12
13         // 탈출을 위해 변수를 더합니다.
14         i++;
15     }
16 }
```

실행 결과

```
0번째 출력:52
1번째 출력:273
2번째 출력:32
3번째 출력:65
4번째 출력:103
```

03 do while 반복문

■ do while 반복문

- 조건의 참 거짓 여부와 상관없이 내부의 문장을 최소한 한 번은 실행해야 하는 경우에 사용
- while 반복문과 달리 조건비교보다 문장의 실행이 먼저 일어나므로 문장을 적어도 한 번 실행

```
do
{
    // 불_표현식이 참인 동안 실행할 문장
} while (불_표현식);
```

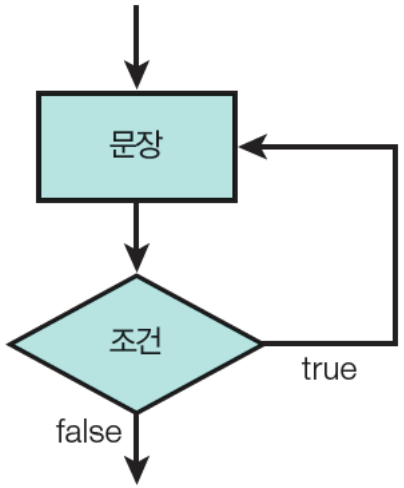


그림 4-4 do while 반복문 형식과 순서도

03 do while 반복문

■ 기본예제 4-4 do while 반복문 활용 /4장/DoWhileBasic

- 사용자에게 입력을 지속적으로 받고, 입력 값이 exit일 때 종료

코드 4-11 do while 반복문 활용

```
01 static void Main(string[] args)
02 {
03     string input;
04     do
05     {
06         Console.Write("입력(exit을 입력하면 종료): ");
07         input = Console.ReadLine();
08     } while (input != "exit");
09 }
```

실행 결과

```
입력(exit을 입력하면 종료): ㅇㅂㅇ
입력(exit을 입력하면 종료): ㅇㅅㅇ
입력(exit을 입력하면 종료): exit
```

for 반복문

- 원하는횟수만큼 반복하고 싶은 경우에 사용
- for 반복문은 맨 먼저 초기식을 실행하고 조건식을 확인
- 조건식이 거짓이면 반복문을 빠져나가고, 참이면 문장과 종결식을 차례로 실행하고 다시 조건식이 참인지 확인

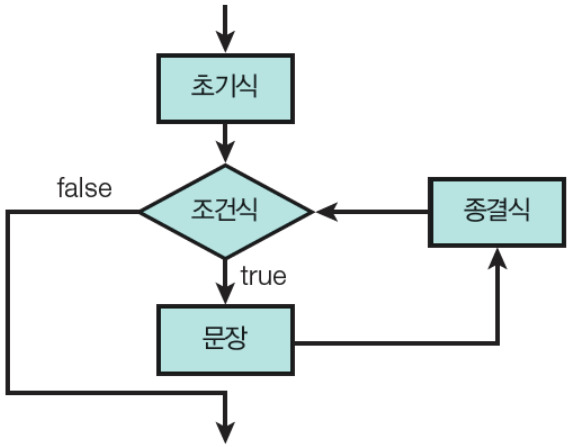


그림 4-5 for 반복문 순서도

for 반복문

- 원하는횟수만큼 반복하고 싶은 경우에 사용
- for 반복문은 맨 먼저 초기식을 실행하고 조건식을 확인
- 조건식이 거짓이면 반복문을 빠져나가고, 참이면 문장과 종결식을 차례로 실행하고 다시 조건식이 참인지 확인

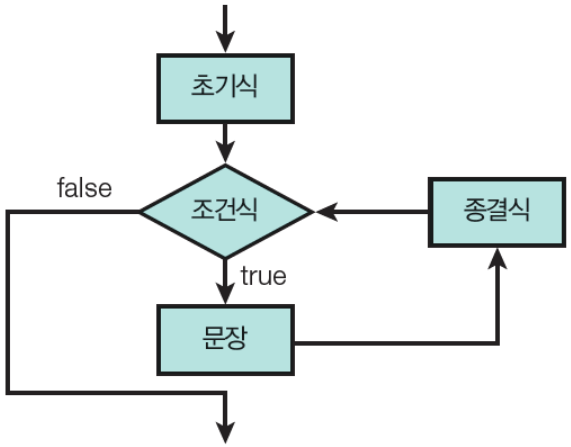


그림 4-5 for 반복문 순서도

■ for 반복문

- for 반복문의 단계

❶ 초기식을 설정한다

❷ 조건식을 비교한다. 조건이 거짓이면 반복문을 종료한다

❸ 문장을 실행한다

❹ 종결식을 실행한다

❺ 2단계로 이동한다

```
for (int i = 0; i < 반복_횟수; i++)
{


}
```

04 for 반복문

for 반복문

- 일반적으로 다음과 같은 형식을 사용하여 특정한 횟수만큼 반복할 때 사용

```
for (int i = 0; i < 반복_횟수; i++)
{


}
```

04 for 반복문

■ 기본예제 4-5 for 반복문으로 덧셈 /4장/SumWithFor

- 0부터 100까지 더하는 예제

코드 4-12 for 반복문으로 덧셈

```
01  static void Main(string[] args)
02  {
03      // 변수를 선언합니다.
04      int output = 0;
05
06      // 반복을 수행합니다.
07      for (int i = 0; i <= 100; i++)
08      {
09          output += i;
10      }
11
12      // 출력합니다.
13      Console.WriteLine(output);
14  }
```

실행 결과

```
5050
```

04 for 반복문

■ 기본예제 4-6 for 반복문으로 곱셈 /4장/MultiplyWithFor

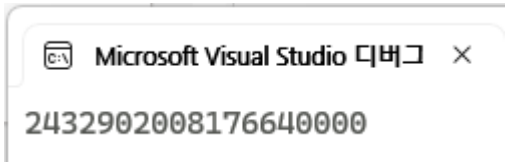
- 1부터 20까지 곱하는 예제

코드 4-13 for 반복문으로 곱셈

```
01 static void Main(string[] args)
02 {
03     // 변수를 선언합니다.
04     long output = 1;
05
06     // 반복을 수행합니다.
07     for (int i = 1; i <= 20; i++)
08     {
09         output *= i;
10     }
11
12     // 출력합니다.
13     Console.WriteLine(output);
14 }
```

초기 값을 0으로 놓으면 무엇을 곱해도 0이 됩니다.
따라서 이번에는 1로 설정합니다.

곱셈은 숫자가 급격하게 커집니다.
최종 결과가 2432902008176640000으로 int 자료형의 한계를 넘으므로 int 자료형보다 많은 값을 담을 수 있는 long 자료형을 사용합니다.



만약 변수 output이 long이 아니라 int 라면

04 for 반복문

■ 기본예제 4-7 한글 전부 출력 /4장/KoreanCharacters

- char 자료형은 숫자로 변환될 수 있음
- 한글은 컴퓨터 내부에서 가부터 힣까지 11,172자가 구현됨

코드 4-14 한글 전부 출력

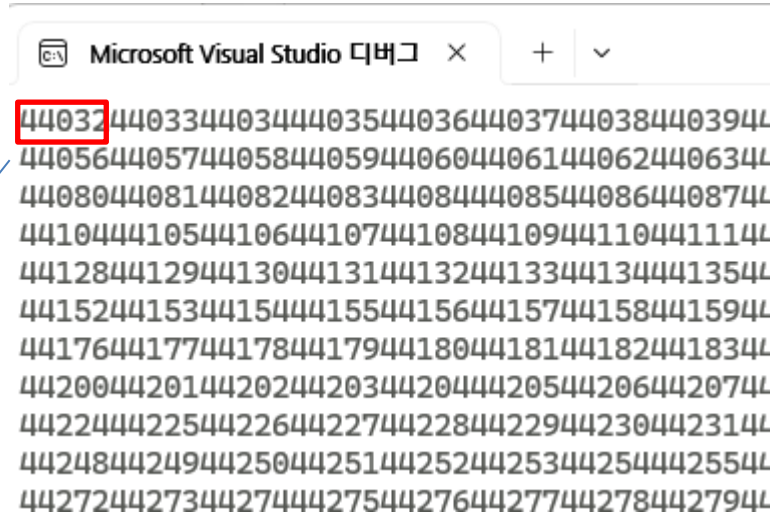
```
01 static void Main(string[] args)
02 {
03     for (int i = '가'; i <= '힣'; i++)
04     {
05         Console.Write((char)i);
06     }
07 }
```

실행 결과

```
...생략...
홓홖홗홚홛훠훡훢훣훤훥훦훧훨훩훪훫훬훭훮훯훰훱훲훳훴훵훶훷훸훹훺훻훼훽훾훿휀휁휂휃휄휅휆휇휈휉휊휋휌휍휎휏휐휑휒휓휔휕휖휗휘휙휚휛휜휝휞휟휠휡휢휣휤휥휦휧휨휩휪휫휬휭휮휯휰휱휲휳휴휵휶휷휸휹휺휻휼휽휾휿흀흁흂흃흄흅흆흇흈흉흊흋흌흍흎흏흐흑흒흓흔흕흖흗흘흙흚흛흜흝흞흟흠흡흢흣흤흥흦흧흨흩흪흫희흭흮흯흰흱흲흳흴흵흶흷흸흹흺흻흼흽흾흿힀힁힂힃힄힅힆힇히힉힊힋힌힍힎힏힐힑힒힓힔힕힖힗힘힙힚힛힜힝힞힟힠힡힢힣
```

04 for 반복문

```
namespace KoreanCharacters
{
    참조 0개
    internal class Program
    {
        참조 0개
        static void Main(string[] args)
        {
            for (int i = '가'; i <= '힣'; i++)
            {
                Console.Write(i);
            }
        }
    }
}
```



	0	1	2
AC0	가	각	갂
AC1	감	갑	값
AC2	갠	갡	갢

가 : 44032

AC00 = A * 16^3 + C * 16^2 + 0 * 16^1 + 0 * 16^0

= 10 * 16^3 + 12 * 16^2 + 0 * 16^1 + 0 * 16^0

= 40960 + 3072 + 0 + 0

= 44032

05 역 for 반복문

■역 for 반복문

- 반복을 뒤에서 부터 실행하는 경우에 사용

```
for (int i = length - 1; i >= 0; i--)
{


}
```

05 역 for 반복문

■기본예제 4-8 역 for 반복문 /4장/ForReverse

▪다음 코드는 배열을 생성하고 배열의 요소를 뒤쪽부터 출력

코드 4-16 역 for 반복문

```
01 static void Main(string[] args)
02 {
03     // 배열을 생성합니다.
04     int[] intArray = { 1, 2, 3, 4, 5, 6 };
05
06     // 요소의 길이를 출력합니다.
07     for (int i = intArray.Length - 1; i >= 0; i--)
08     {
09         Console.WriteLine(intArray[i]);
10     }
11 }
```

실행 결과

```
6
5
4
3
2
1
```

06 foreach 반복문

■foreach 반복문

- 컬렉션에 쉽게 반복문을 적용할 때에 사용
- 컬렉션은 여러 개체가 모여서 집합을 이룬 것

```
foreach (자료형 변수 in 컬렉션)
{

}
```

- foreach 반복문을 사용하면 다음과 같은 for 반복문을 사용하는 것과 같은 역할을 수행

```
for (int i = 0; i < 컬렉션.길이; i++)
{
    자료형 변수 = 컬렉션[i];
}
```

06 foreach 반복문

■ 기본예제 4-9 foreach 반복문과 배열 /4장/ForeachBasic

- 다음 코드는 string 배열을 만들고 foreach 반복문 내부에서 배열의 요소를 차례대로 출력

코드 4-17 foreach 반복문과 배열

```
01  static void Main(string[] args)
02  {
03      // 변수를 선언합니다.
04      string[] array = { "사과", "배", "포도", "딸기", "바나나" };
05
06     // 반복을 수행합니다.
07      foreach (string item in array)
08      {
09          // 출력합니다.
10          Console.WriteLine(item);
11      }
12  }
```

실행 결과

```
사과
배
포도
딸기
바나나
```

06 foreach 반복문

■ 기본예제 4-10 foreach 반복문과 var 키워드 /4장/ForeachWithVa

- foreach 반복문에도 다음과 같이 var 키워드를 사용할 수 있음

코드 4-18 foreach 반복문과 var 키워드

```
01  static void Main(string[] args)
02  {
03      // 변수를 선언합니다.
04      string[] array = { "사과", "배", "포도", "딸기", "바나나" };
05
06      // 반복을 수행합니다.
07      foreach (var item in array)
08      {
09          // 출력합니다.
10          Console.WriteLine(item);
11      }
12  }
```

07 중첩 반복문

- 반복문을 여러 겹 중첩해서 사용하면 중첩 반복문이라고 부름
- **기본예제 4-11** 별 피라미드(1) /4장/PyramidA
  - 1차원(한 줄)으로 무언가를 출력할 때는 반복문을 한 겹으로 사용
  - 2차원(면)으로 무언가를 출력할 때는 반복문을 두 겹 사용

코드 4-20 별 피라미드(1)

```
01 static void Main(string[] args)
02 {
03     for (int i = 0; i < 10; i++)
04     {
05         for (int j = 0; j < i + 1; j++)
06             Console.Write('*');
07         Console.Write('\n');
08     }
09 }
```

실행 결과

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

■기본예제 4-12 별 피라미드(2)

/4장/PyramidB

코드 4-21 별 피라미드(2)

```
01 static void Main(string[] args)
02 {
03     for (int i = 0; i < 10; i++)
04     {
05         for (int j = 0; j < 10 - i; j++)
06             Console.Write(' ');
07         for (int j = 0; j < i + 1; j++)
08             Console.Write('*');
09         Console.Write('\n');
10     }
11 }
```

실행 결과

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

08 break 키워드

■Break 키워드

- Switch 조건문 또는 반복문을 벗어날 때에 사용하는 키워드
- 무한 반복문은 내부에서 break 키워드를 사용해야 벗어날 수 있음

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

08 break 키워드

■ 기본예제 4-13 break 키워드 /4장/BreakBasic

- 다음 코드는 짝수를 입력하면 break 키워드로 반복문을 벗어남

코드 4-22 break 키워드

```
01 static void Main(string[] args)
02 {
03     while (true)
04     {
05         Console.Write ("숫자를 입력해주세요(짝수를 입력하면 종료): ");
06         int input = int.Parse(Console.ReadLine());
07         if (input % 2 == 0)
08         {
09             break;
10         }
11     }
12 }
```

실행 결과

```
숫자를 입력해주세요(짝수를 입력하면 종료): 31
숫자를 입력해주세요(짝수를 입력하면 종료): 51
숫자를 입력해주세요(짝수를 입력하면 종료): 61
숫자를 입력해주세요(짝수를 입력하면 종료): 52
```

09 continue 키워드

- 반복문 내부에서 현재 반복을 멈추고 다음 반복을 진행하게 만드는 키워드
- **기본예제 4-14** continue 키워드 /4장/ContinueBasic
  - 다음 코드는 변수 i가 짝수일 때 continue 키워드로 현재 반복을 멈추고 다음 반복을 진행

코드 4-24 continue 키워드

```
01 static void Main(string[] args)
02 {
03     for (int i = 1; i < 10; i++)
04     {
05         if (i % 2 == 0)
06         {
07             continue;
08         }
09         Console.WriteLine(i);
10     }
11 }
```

짝수라면 다음 반복으로 바로 넘어갑니다.
따라서 이 아래의 코드(09행)는 실행되지 않습니다.

실행 결과

```
1
3
5
7
9
```

09 continue 키워드

■ 기본예제 4-14 continue 키워드 /4장/ContinueBasic

- 조건문의 조건을 잘 사용한다면 break 키워드와 continue 키워드의 사용을 줄일 수 있음
- [코드 4-24]는 다음과 같이 간단한 코드로 바꿀 수 있음

코드 4-25 [코드 4-24]를 간단하게 변경

```
01 static void Main(string[] args)
02 {
03     for (int i = 1; i < 10; i++)
04     {
05         if (i % 2 != 0)
06         {
07             Console.WriteLine(i);
08         }
09     }
10 }
```

실행 결과

```
1
3
5
7
9
```