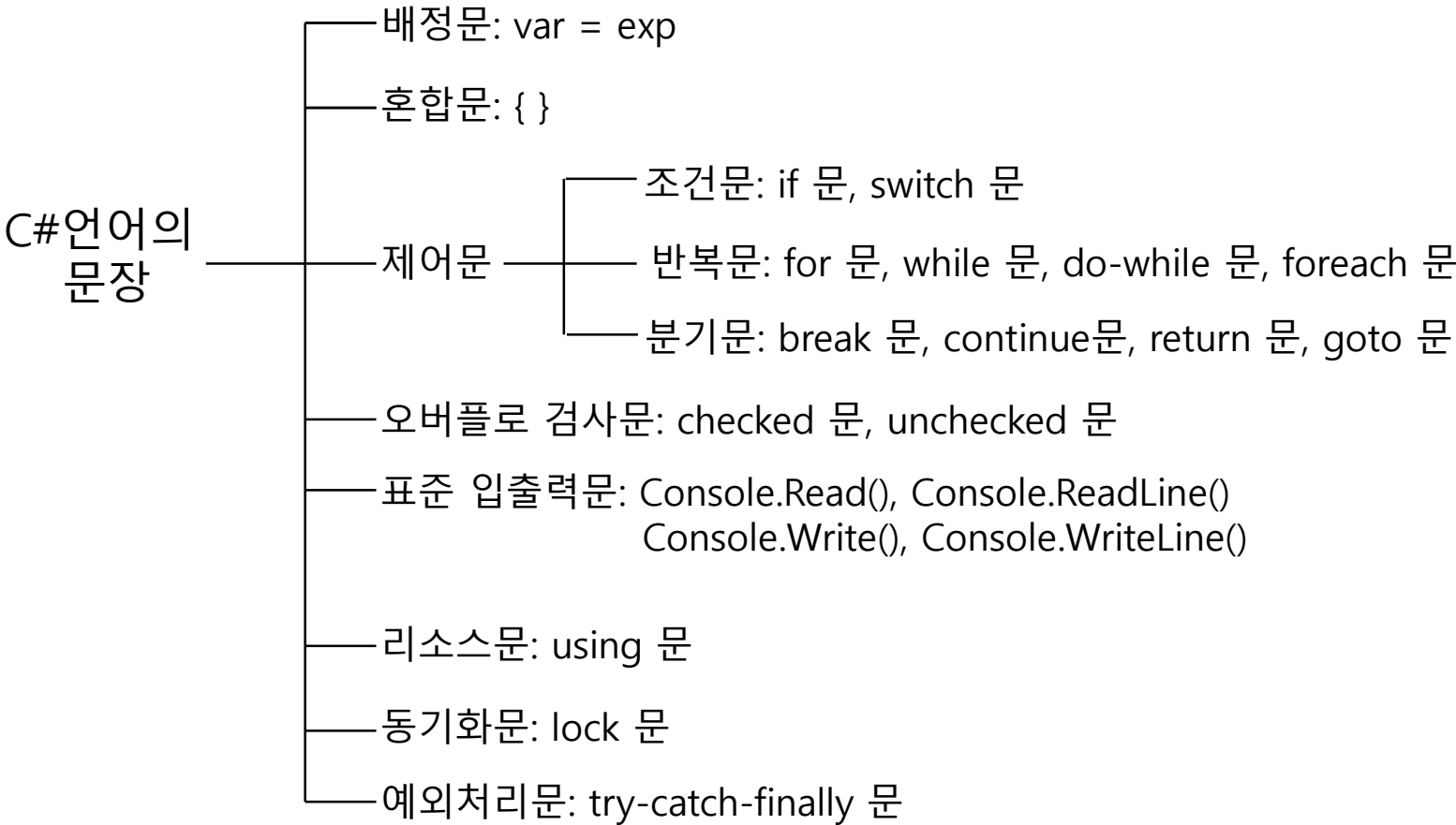


객체지향 프로그래밍

C# - 문장



문장의 종류



- 값을 변수에 저장하는데 사용

- 형태 : <변수> = <식>;

```
remainder = dividend % divisor;  
i = j = k = 0;  
var op= exp;
```

- 형 변환

- 묵시적 형 변환 : 컴파일러에 의해 자동
- 명시적 형 변환 : 프로그래머가 캐스트(cast) 연산자



```
namespace AssingmentStApp
{
    class Program
    {
        static void Main(string[] args)
        {
            short s; int i;
            float f; double d;
            s = 526;
            d = f = i = s;
            Console.WriteLine("s = " + s + " i = " + i);
            Console.WriteLine("f = " + f + " d = " + d);
        }
    }
}
```



- 여러 문장을 한데 묶어 하나의 문장으로 나타냄
 - 주로 문장의 범위를 표시

- 형태: { <선언> 또는 <문장> }

```
if (a > b) a--; b++;
```

```
if (a > b) { a--; b++; }
```

- 지역변수(Local Variable)
 - 블록의 내부에서 선언된 변수
 - 선언된 블록 안에서만 참조 가능



```
namespace CompoundStApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.Write("Enter one digit = ");
            n = Console.Read() - '0';
            if (n < 0)
                Console.WriteLine("Negative number !");
            else
            {
                Console.WriteLine(n + " squared is " + (n * n));
                Console.WriteLine(n + " cubed is " + (n * n * n));
            }
        }
    }
}
```



```
namespace AnotherBlockApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 1;
            { // int x;      // error
                int y = 2;
                Console.WriteLine("Block 1: x = " + x + ", y = " + y);
            }
            {
                int y = 3;
                Console.WriteLine("Block 2: x = " + x + ", y = " + y);
            }
        }
    }
}
```



[예제 – LocalVariableApp.cs]

```
using System;
class LocalVariableApp {
    static int x;
    public static void Main() {
        int x = (LocalVariableApp.x=2) * 2;
        Console.WriteLine("static x = " + LocalVariableApp.x);
        Console.WriteLine("local  x = " + x);
    }
}
```

실행 결과 :

static x = 2

local x = 4



제어문

- 프로그램의 실행 순서를 바꾸는 데 사용
- 실행 순서를 제어하는 방법에 따라
 - 조건문 : if 문, switch 문
 - 반복문 : for 문, while 문, do-while 문, foreach 문
 - 분기문 : break 문, continue 문, return 문, goto 문



조건문 - if문

- 조건에 따라 실행되는 부분이 다를 때 사용
- if 문 형태

```
if ( <조건식> ) <문장>  
if ( <조건식> ) <문장1> else <문장2>
```

- 조건식의 연산결과 : 논리형 (true or false)

- 예

```
if (a < 0) a = -a;           // 절대값  
if (a > b) m = a; else m = b; // 큰값
```



```
namespace IfStApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.Write("Enter a number = ");
            n = Console.Read() - '0';
            if (n % 2 == 0)
                Console.WriteLine(n + " is an even number.");
            if (n % 2 != 0)
                Console.WriteLine(n + " is an odd number.");
        }
    }
}
```



```
namespace IfElseStApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.Write("Enter a number = ");
            n = Console.Read() - '0';
            if (n % 2 == 0)
                Console.WriteLine(n + " is an even number.");
            else
                Console.WriteLine(n + " is an odd number.");
        }
    }
}
```



조건문 - if문

■ 내포된 if 문

■ 참 부분에서 if 문이 반복

```
if (<조건식>
    if (<조건식>
        // ...
        <문장>
```

■ else 부분에서 if 문이 반복

```
if (<조건식1>)    <문장1>
else if (<조건식2>) <문장2>
    ...
else if (<조건식n>) <문장n>
else <문장>
```



```
namespace NestedIfApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int day;
            Console.Write("Enter the day number 1 ~ 7 : ");
            day = (int)Console.Read() - '0';
            if (day == 1) Console.WriteLine("Sunday");
            else if (day == 2) Console.WriteLine("Monday");
            else if (day == 3) Console.WriteLine("Tuesday");
            else if (day == 4) Console.WriteLine("Wednesday");
            else if (day == 5) Console.WriteLine("Thursday");
            else if (day == 6) Console.WriteLine("Friday");
            else if (day == 7) Console.WriteLine("Saturday");
            else Console.WriteLine("Illegal day");
        }
    }
}
```



조건문 - switch문

- 조건에 따라 여러 경우로 처리해야 되는 경우
- switch 문의 형태

```
switch (<식>) {  
    case <상수식1> : <문장1> break;  
    case <상수식2> : <문장2> break;  
    .  
    .  
    case <상수식n> : <문장n> break;  
    default : <문장> break;  
}
```

- 여기서, default의 의미는 otherwise
- break 문을 사용하여 탈출



```
namespace SwitchStApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter the day number 1 ~ 7 : ");
            int day = Console.Read() - '0';
            switch (day)
            {
                case 1: Console.WriteLine("Sunday"); break;
                case 2: Console.WriteLine("Monday"); break;
                case 3: Console.WriteLine("Tuesday"); break;
                case 4: Console.WriteLine("Wednesday"); break;
                case 5: Console.WriteLine("Thursday"); break;
                case 6: Console.WriteLine("Friday"); break;
                case 7: Console.WriteLine("Saturday"); break;
                default: Console.WriteLine("Illegal day"); break;
            }
        }
    }
}
```




```
namespace SwitchStWithString
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter the weekday (Sunday-Saturday) : ");
            string day = Console.ReadLine();
            switch (day)
            {
                case "Sunday": Console.WriteLine(1); break;
                case "Monday": Console.WriteLine(2); break;
                case "Tuesday": Console.WriteLine(3); break;
                case "Wednesday": Console.WriteLine(4); break;
                case "Thursday": Console.WriteLine(5); break;
                case "Friday": Console.WriteLine(6); break;
                case "Saturday": Console.WriteLine(7); break;
                default: Console.WriteLine("Illegal day"); break;
            }
        }
    }
}
```



```
namespace CalculatorApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y, r = 0;
            char opr;
            Console.WriteLine("Enter an operator & two numbers = ");
            opr = (char)Console.Read();
            x = Console.Read() - '0';
            y = Console.Read() - '0';
            switch (opr)
            {
                case '+': r = x + y;
                    Console.WriteLine(x + " + " + y + " = " + r); break;
                case '-': r = x - y;
                    Console.WriteLine(x + " - " + y + " = " + r); break;
                case '*': r = x * y;
                    Console.WriteLine(x + " * " + y + " = " + r); break;
                case '/': r = x / y;
                    Console.WriteLine(x + " / " + y + " = " + r); break;
                default: Console.WriteLine("Illegal operator "); break;
            }
        }
    }
}
```



반복문 - for문

- 정해진 횟수만큼 일련의 문장을 반복
- for 문의 형태

```
for ( <식1> ; <식2> ; <식3> )  
    <문장>
```

- <식1> : 제어 변수 초기화
- <식2> : 제어 변수를 검사하는 조건식
- <식3> : 제어 변수의 값을 수정

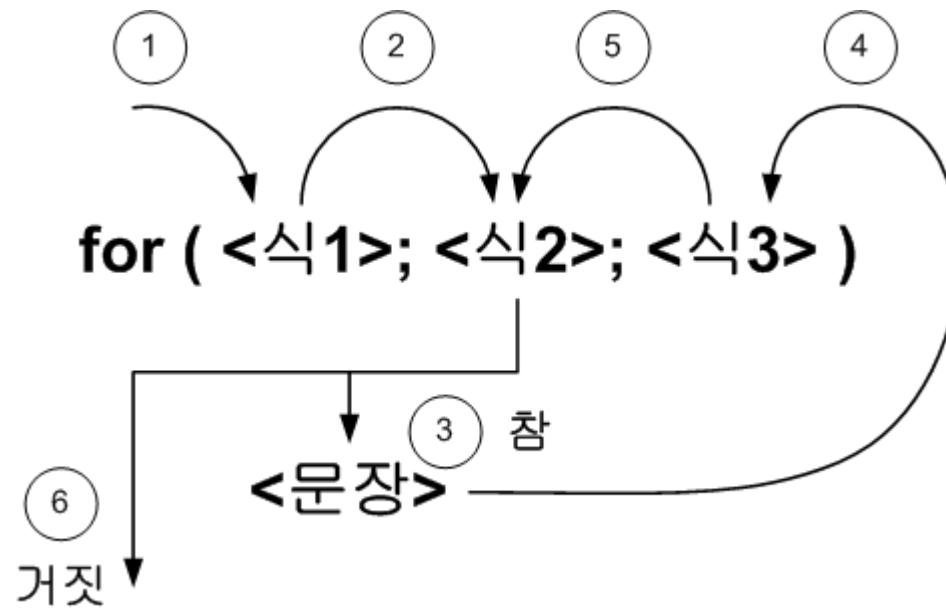
■ 예

```
s = 0;  
for (i = 1; i <= N; ++i) // 1부터 N까지의 합 : i 증가  
    s += i;
```



반복문 - for문

■ for 문의 실행순서



반복문 - for문

- 무한 루프를 나타내는 for 문

```
for ( ;; )  
    <문장>
```

- 루프 종료 : break 문, return 문

- 내포된 for 문

- for 문 안에 for 문이 있을 때
- 다차원 배열을 다룰 때

```
for (i = 0; i < N; ++i)  
    for (j=0; j<M; ++j)  
        matrix[i, j] = 0;
```



```
namespace ForStApp
{
    class Program
    { //  $h(n) = 1 + 1/2 + 1/3 + \dots + 1/n$ 
        static void Main(string[] args)
        {
            int i, n;
            double h = 0.0;
            Console.Write("Enter a number = ");
            n = Console.Read() - '0';
            for (i = 1; i <= n; ++i)
                h = h + 1 / (double)i;
            Console.WriteLine("n = {0}, h = {1}", n, h);
        }
    }
}
```



```
namespace PrintMatrixApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int[,] m = { { 1, 2, 3},
                        { 4, 5, 6},
                        { 7, 8, 9} };
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                    Console.Write("m[" + i + "," + j + "]=" + m[i, j] + " ");
                Console.WriteLine();
            }
        }
    }
}
```



반복문 - while문

■ while 문의 형태

```
while ( 조건식 )  
    <문장>
```

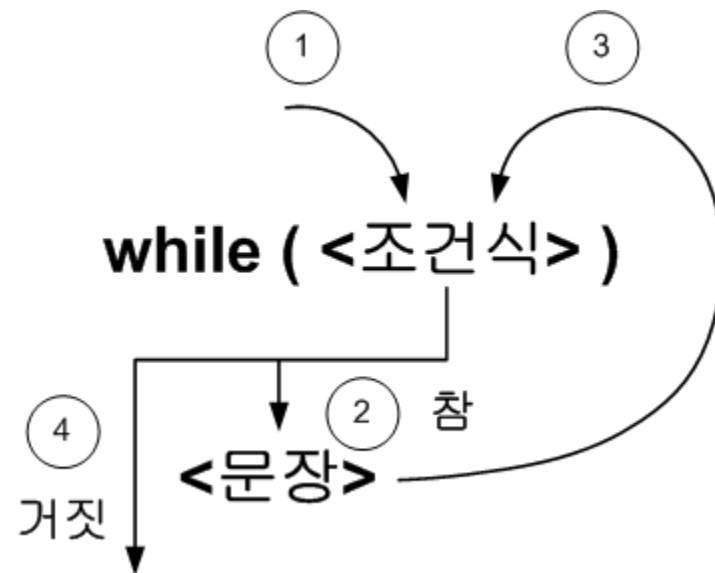
■ 예

```
i = 1; s = 0;  
while (i <= N) { // 1부터 N까지의 합  
    s += i;  
    ++i;  
}
```



반복문 - while문

■ while 문의 실행순서



반복문 - while문

■ for 문과 while 문의 비교

```
for (i = 0; i < N; ++i)
    s += i;
```

```
i = 0;
while (i < N) {
    s += i;
    ++i;
}
```

- for --- 주어진 횟수
- while --- 주어진 조건



```
namespace WhileStApp
{
    class Program
    { // h(n) = 1 + 1/2 + 1/3 + ... + 1/n
        static void Main(string[] args)
        {
            int i, n;
            double h = 0.0;
            Console.Write("Enter a number = ");
            n = Console.Read() - '0';
            i = 1;
            while(i <= n)
            {
                h = h + 1/(double) i;
                i++;
            }
            Console.WriteLine("n = {0}, h = {1}", n, h);
        }
    }
}
```



반복문 - do-while문

- 반복되는 문장을 먼저 실행 한 후에 조건식을 검사
- do-while 문의 형태

```
do  
    <문장>  
while ( <조건식> );
```

조건식이 거짓이라도 <문장>
부분이 적어도 한번은 실행

- precondition check --- for, while
- postcondition check --- do-while



```
namespace DoWhileStApp
{
    class Program
    { // h(n) = 1 + 1/2 + 1/3 + ... + 1/n
        static void Main(string[] args)
        {
            int n, i;
            double h = 0.0;
            Console.Write("Enter a number = ");
            n = Console.Read() - '0';
            i = 1;
            do
            {
                h = h + 1/(double)i;
                i++;
            } while (i <= n);
            Console.WriteLine("n = " + n + ", h = " + h);
        }
    }
}
```



반복문 – foreach문

- 데이터의 집합에 대한 반복을 수행
- foreach 문의 형태

```
foreach ( 자료형 변수명 in 데이터의 집합 )  
    <문장>
```

- 예

```
foreach ( string s in color )  
    Console.WriteLine(s);
```



```
namespace ForeachStApp
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] color = { "red", "green", "blue" };
            foreach (string s in color)
                Console.WriteLine(s);
        }
    }
}
```



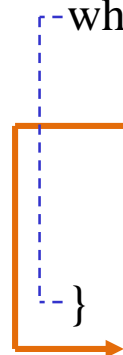
분기문 - break문

- 블록 밖으로 제어를 옮기는 역할
- break 문의 형태

```
break;
```

- 예

```
int i = 1;
while (true) {
    if (i == 3)
        break;
    Console.WriteLine("This is a " + i + " iteration");
    ++i;
}
```




```
namespace BreakStApp
{
    class Program
    { // h(n) = 1 + 1/2 + 1/3 + ... + 1/n
        static void Main(string[] args)
        {
            int n, i;
            double h = 0.0;
            Console.Write("Enter a number = ");
            n = Console.Read() - '0';
            i = 1;
            while (true)
            {
                h = h + 1/(double) i;
                if (++i > n) break;
            }
            Console.WriteLine(" n = " + n + ", h = " + h);
        }
    }
}
```



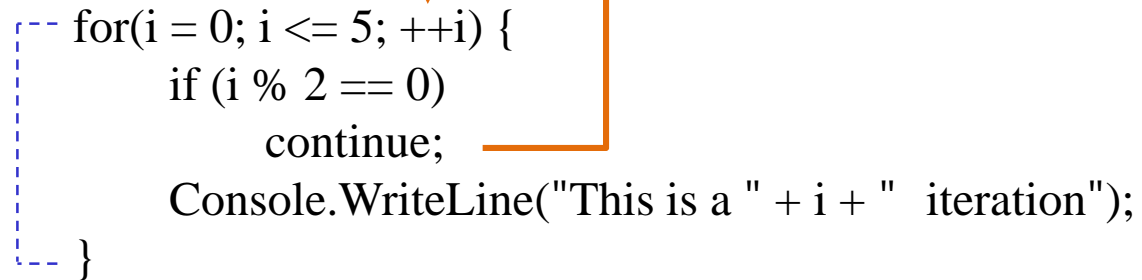
분기문 - continue문

- 다음 반복이 시작되는 곳으로 제어를 옮기는 기능
- continue 문의 형태

```
continue;
```

- for 문 안에서 사용될 때

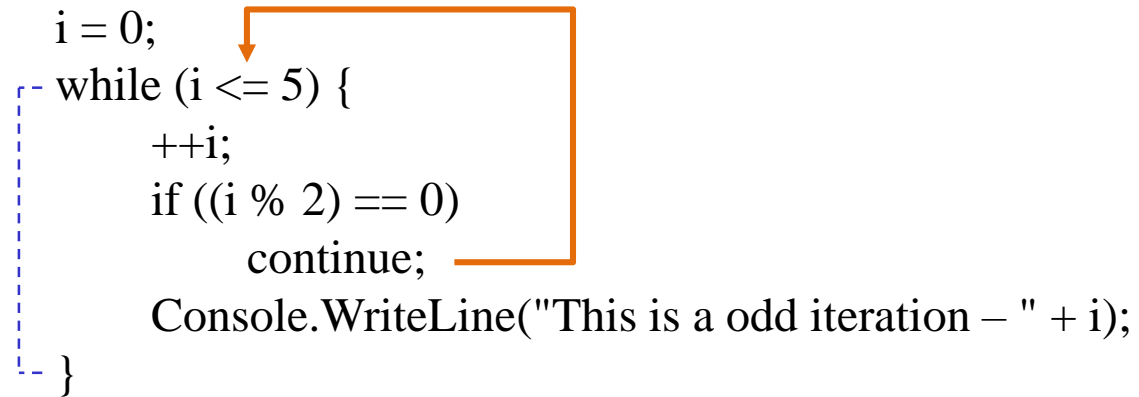
```
for(i = 0; i <= 5; ++i) {  
    if (i % 2 == 0)  
        continue;  
    Console.WriteLine("This is a " + i + " iteration");  
}
```



분기문 - continue문

- while 문 안에서 사용될 때
 - 조건식을 검사하는 부분으로 옮김

```
i = 0;
while (i <= 5) {
    ++i;
    if ((i % 2) == 0)
        continue;
    Console.WriteLine("This is a odd iteration - " + i);
}
```



```
namespace ContinueStApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int n, s, i;
            Console.Write("Enter a number = ");
            for (; ; )
            {
                n = Console.Read() - '0';
                if (n == 0) break;
                else if (n < 0) continue;
                for (s = 0, i = 1; i <= n; ++i)
                    s += i;
                Console.WriteLine("n = " + n + ", sum = " + s);
            }
            Console.WriteLine("End of Main");
        }
    }
}
```



분기문 - goto문

- 지정된 위치로 제어 흐름을 이동
- goto 문의 형태

```
goto label;  
goto case constant-expression;  
goto default;
```

- goto 문이 분기할 수 없는 경우
 - 외부에서 복합문 안으로 분기
 - 메소드 내에서 외부로 분기
 - finally 블록에서 블록 밖으로 분기



분기문 - return문

- 메소드의 실행을 종료하고 호출한 메소드(caller)에게 제어를 넘겨주는 문장
- return 문의 형태

```
return;  
return <식>;
```



```
namespace JumpStApp
{
    class Program
    {
        static int sum(int s, int i)
        {
            return s + i;
        }
        static void Main(string[] args)
        {
            int n, s, i;
            for (;;)
            {
                Console.WriteLine("Enter a number : ");
                n = Int32.Parse(Console.ReadLine());
                if (n == 0) break;
                else if(n < 0) continue;
                for (s = 0, i = 1 ; i <= n ; ++i)
                    s = sum(s, i);
                Console.WriteLine("n = {0}, sum = {1}", n, s);
            }
            Console.WriteLine("End of Main");
        }
    }
}
```



오버플로우 검사문 – checked문

- 오버플로를 명시적으로 검사하는 문장
 - System 네임스페이스에 있는 OverflowException 예외가 발생
- checked 문의 형태

```
checked {  
    // 오버플로가 발생하는지를 확인하려는 문장  
}
```

- 수식 checked 문의 형태

```
checked (오버플로가 발생하는지를 확인하려는 수식)
```



오버플로우 검사문 - unchecked문

- 오버플로를 의도적으로 검사하지 않을 경우
- unchecked 문의 형태

```
unchecked {  
    // 오버플로를 의도적으로 검사하지 않으려는 문장  
}
```



```
namespace OverflowApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, max = int.MaxValue;
            try
            {
                Console.WriteLine("Start of try statement");
                i = max + 1;    // default: don't check Overflow.
                Console.WriteLine("After default overflow");
                unchecked
                {
                    i = max + 1; // don't check Overflow intensionally.
                }
                Console.WriteLine("After unchecked statement");
                checked
                {
                    i = max + 1; // check Overflow
                }
                Console.WriteLine("After checked statement");
            }
            catch (OverflowException e)
            {
                Console.WriteLine("caught an OverflowException");
            }
        }
    }
}
```



표준 입출력

- 입출력 장치가 미리 정해진 입출력을 의미
- C# 언어의 기본 네임스페이스인 System으로부터 제공
- 표준 입력 메소드
 - Console.Read()
 - 키보드로부터 한 개의 문자를 읽어 그 문자의 코드값을 정수형으로 반환하는 기능
 - Console.ReadLine()
 - 한 라인을 읽어 string형으로 반환하는 기능
 - 숫자 값으로 바꿔야 하는데 정수인 경우
 - int.Parse() 메소드 사용



■ 표준 출력 메소드

■ Console.Write()

- 화면에 매개 변수의 값을 출력

■ Console.WriteLine()

- 화면에 매개 변수의 값을 출력한 후 다음 라인으로 출력 위치를 이동



■ 형식화된 출력(formatted output)

■ 출력하려는 값에 포맷을 명시하여 원하는 형태로 출력

■ 출력 포맷의 형태

`{N[,W][:formatCharacter]}`

- N : 매개 변수를 위치적으로 지칭하는 정수 (단, 0부터 시작)
- W : 출력될 자릿수의 폭을 나타내며 선택으로 명시
 - '-' 기호를 붙이면 좌측정렬로 출력
- formatCharacter : 한 문자로 이루어진 형식 지정 문자를 의미



표준 입출력

- 형식 지정 스트링
 - 매개 변수의 개수와 일치하는 출력 포맷
- 표준 형식 지정문자

형식 지정자	설명
C 또는 c	통화 표시
D 또는 d	10진수 형태(정수형만 가능)
E 또는 e	지수 형태
F 또는 f	고정 소수점 형태
G 또는 g	고정 소수점 또는 지수 형태 중 간략한 형태를 선택한다.
N 또는 n	10진수(자릿수 구분을 위한 ',' 포함)
P 또는 p	백분율('%'도 포함)
R 또는 r	결과 스트링을 다시 읽었을 때, 원 값과 동일함을 보장 (부동소수점 수만 가능)
X 또는 x	16진수(정수형만 가능)



```
namespace SimpleIOApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int i; char c;
            Console.Write("Enter a digit and a character = ");
            i = Console.Read() - 48;
            c = (char)Console.Read();
            Console.Write("i = " + i);
            Console.WriteLine(", c = " + c);
        }
    }
}
```



```
namespace ReadLineApp
{
    class Program
    {
        static void Main(string[] args)
        {
            int time, hour, minute, second;
            Console.Write("*** Enter an integral time : ");
            time = int.Parse(Console.ReadLine());
            hour = time / 10000;
            minute = time / 100 % 100;
            second = time % 100;
            Console.WriteLine("*** Time is " + hour + ":" + minute + ":" + second);
        }
    }
}
```




```
namespace ReadIntegerApp
{
    class Program
    {
        static int ReadInt()
        {
            char ch;
            int n=0;
            while (!char.IsDigit(ch = (char)Console.Read()));
            do
            {
                n = n*10 + (ch - '0');
                ch = (char) Console.Read();
            } while (char.IsDigit(ch));
            return n;
        }
        static void Main(string[] args)
        {
            Console.Write("*** input data : ");
            Console.WriteLine("*** read data : " + ReadInt() + " " + ReadInt());
        }
    }
}
```



```
namespace FormattedOutputApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("1) {0,-5},{1,5},{2,5}", 1.2, 1.2, 123.45);
            double d = Math.PI;
            Console.WriteLine("2) {0}", d);
            Console.WriteLine("3) {0:C}", d);
            Console.WriteLine("4) {0:E}", d);
            Console.WriteLine("5) {0:F}", d);
            Console.WriteLine("6) {0:G}", d);
            Console.WriteLine("7) {0:P}", d);
            Console.WriteLine("8) {0:R}", d);
            Console.WriteLine("9) {0:X}", 255);
        }
    }
}
```



Reference

- ✓ C# 프로그래밍 입문, 오세만 외4, 생능출판
- ✓ 초보자를 위한 C# 200제, 강병익, 정보문화사
- ✓ 프랙티컬 C#, 이데이 히데유키, 김범준, 위키북스
- ✓ C#언어 프로그래밍 바이블, 김명렬 외1, 홍릉과학출판사
- ✓ C# and the .NET Platform, Andrew Troelsen, 장시혁, 사이텍미디어

