Ch03.

데이터를 담는 변수와 상수

권 봄

2025-08-05

3.1 다양한 데이터의 종류

- CPU의 연산 장치는 주로 정수를 처리하는 **ALU**(Arithmetic Logic Unit)와, 부동 소수점 연산을 처리하는 **FPU**(Floating Point Unit)으로 구성된다.
- 모든 데이터의 형식의 근간이 되는 **기본 데이터 형식**(Primitive Type)과 **상수**(Constants), **열거형**(Enumeration)에 대해 배운다.
- 복합 데이터 형식에는 **구조체**와 **클래스**, **배열** 등이 있다.
- 데이터 형식은 **기본 데이터 형식**과 **복합 데이터 형식**, **값 형식**과 **참조 형식**으로 분류할 수 있다

3.1.1 ALU와 FPU

- dd

3.2 변수

- 제일 먼저 데이터의 형식을 명시하고 변수 식별자(이름)을 명시한 후 ;로 문장을 종결하여 변수를 선언한다. 예) int x;
  - 이렇게 하면 컴파일러는 int 형식을 위해 메모리 공간을 할당하고 x라는 식별자가 사용할 수 있도록 준비한다.
- 선언된 변수 x는 대입연산자를 통해 데이터를 입력할 수 있다. 예) x = 100;
  - 이를 통해 x에 할당된 메모리 공간에 데이터 100이 기록된다
  - 초기화란 변수에 최초의 데이터를 할당하는 것을 의미한다. -> 잘못된 값이 들어감을 방지하기 위해

3.2.1 변수 실행코드와 결과

- 실행코드 + 결과 ㄱㄱ

3.4 기본 데이터 형식

- C#에서 제공하는 기본 데이터 형식에는 모두 **15가지**가 있으며, 이들은 .NET의 System 네임스페이스에 정의된 구조체(struct)들이다.
- **숫자, 논리, 문자 및 문자열, 객체**(object) 형식들로 나누어진다.
- 숫자 데이터를 저장하는 형식은 **정수형**과 **부동 소수점형**으로 나뉜다.
- 객체와 문자 및 문자열도 참조형이지만 자주 쓰이므로 기본형에 포함된다.
- 대부분의 기본형은 값 형식(value type)이다.

3.4 데이터 형식 모아보기

C# 키워드	.NET 형식명	설명 / 저장 범위
bool	System.Boolean	true / false
byte	System.Byte	0 ~ 255 (8비트 부호 없음)
sbyte	System.SByte	-128 ~ 127 (8비트 부호 있음)
char	System.Char	유니코드 문자 (16비트)
decimal	System.Decimal	고정 소수점 (금융 계산용, 매우 높은 정밀도)
double	System.Double	배정도 부동 소수점 (64비트)
float	System.Single	단정도 부동 소수점 (32비트)
int	System.Int32	-2,147,483,648 ~ 2,147,483,647
uint	System.UInt32	0 ~ 4,294,967,295
long	System.Int64	-9경 ~ 9경 (64비트 정수)
ulong	System.UInt64	0 ~ 18경 (64비트 부호 없는 정수)
short	System.Int16	-32,768 ~ 32,767
ushort	System.UInt16	0 ~ 65,535
object	System.Object	모든 형식의 최상위 타입
string	System.String	유니코드 문자열

3.4.1 정수 형식

데이터 형식	설명	크기(바이트)	범위
byte	아주 작은 양수만 저장 가능	1(8비트)	0 ~ 255
sbyte	s=signed 부호 있는 byte	1(8비트)	-128 ~ 127
short	작은 범위 정수	2(16비트)	-32,768 ~ 32,767
ushort	u=unsigned 부호 없는 short	2(16비트)	0 ~ 65,535
int	기본 정수형, 가장 많이 사용됨	4(32비트)	-21억 ~ 21억 (-2,147,483,648 ~ 2,147,483,647)
uint	부호 없는 int	4(32비트)	0 ~ 42억 (0 ~ 4,294,967,295)
long	큰 범위 정수, 고정밀 계산용	8(64비트)	매우 큰 수 (약 -9경 ~ 9경)
ulong	부호 없는 long	8(64비트)	매우 큰 양수 (0 ~ 약 18경)

3.4.2 부동(不動) 소수점, 고정 소수점(decimal) 형식

데이터 형식	설명	크기(바이트)	범위	정밀도 요약
float	단일 정밀도	4(32비트)	$-3.4028235 \times 10^{38}$ ~ 3.4028235×10^{38}	약 7자리 정확도
double	복수 정밀도	8(64비트)	$-1.7976931348623157 \times 10^{308}$ ~ $1.7976931348623157 \times 10^{308}$	약 15~16자리 정확도
decimal	아주 높은 정밀도, 금융용	16(128비트)	$\pm 1.0 \times 10^{-28}$ ~ $\pm 7.9 \times 10^{28}$	약 28~29자리 정확도

- C#의 float 와 double은 IEEE754라는 표준 알고리즘에 기반한 데이터 형식이다.
- float, double은 **부동소수점 (이진)** → 속도 빠름, 약간 부정확할 수 있음
- decimal은 **고정소수점 (10진 기반)** → 속도 느림, **정확함** (특히 돈 계산에 필수)
- 정밀한 과학 계산에는 double, 돈 계산에는 decimal, 일반 계산엔 float 사용

3.4.3 문자 형식과 문자열 형식

- char 형식의 변수에 문자 데이터를 담는다. ‘’로 표현한다.
- string 형식은 여러 개의 문자를 묶어서 처리한다. “”로 표현한다.

데이터 형식	설명	크기(바이트)	범위
char	한 글자(문자) 예) ‘A’, ‘b’, ‘1’ 등	2(16비트)	U+0000 ~ U+FFFF (유니코드 문자)
string	여러 글자(문자열) 예) “hello”, “hi”, “C#” 등	문자 수 x 2 + 오버헤드	이론상 20억 문자까지 가능 (메모리 제한)

3.4.4 논리 형식

- 논리 형식이 다루는 데이터는 참(True)과 거짓(False) 두가지이다.

데이터 형식	설명	크기(바이트)	범위
bool	논리 형식 (참, 거짓)	1(8비트)	True, false

3.4.5 object(객체) 형식

- object는 .NET의 System.Object 형식으로, 모든 데이터의 최상위 타입이다.
- 어떤 데이터든 다룰 수 있는 데이터, 상속의 효과를 활용할 수 있다.
- 시스템 아키텍쳐에 따라 object의 포인터 크기는 다르다.

데이터 형식	설명	크기(아키텍쳐)	범위
object	모든 C# 자료형의 최상위 기반 형식 값/참조형 모두 저장 가능	4바이트 (32비트) 8바이트 (64비트)	모든 형식 저장 가능

3.4.7 데이터 형식 바꾸기

- 변수를 다른 데이터 형식의 변수에 옮겨 담는 것을 말한다.
- 다음과 같은 5가지 형식 변환을 공부한다.

a. 크기(표현범위)가 서로 다른 정수 형식 사이의 변환
   - 예) uint 최대값 -> int로 변환 시 오버플로우 발생
   - Int 최소값 -> uint 변환 시 언더플로우 발생

b. 크기가 서로 다른 부동 소수점 형식 사이의 변환
   - 부동 소수점 형식의 특성상 오버플로우가 존재하지 않는다.
   - 정밀성에 손상이 이루어진다.
   - 2진수로는 소수를 완전히 다루지 않는다. 따라서 손상이 발생할 수 있다.

c. 부호 있는 정수형식과 부호 없는 정수 형식 사이의 변환

d. 부동 소수점 형식과 정수 형식 사이의 변환
   - 소수점에서 정수로 변환 될 때 반올림은 존재하지 않는다. 예) 0.9 -> 0

e. 문자열과 숫자 사이의 변환
   - 숫자 -> 문자는 변수.ToString()을 통해 이루어진다.
   - 문자 -> 숫자는 형식.Prase(문자)를 통해 이루어진다.

3.4.7 데이터 형식 바꾸기

- 실행

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace _3_4_7_Data_Type_Conversion
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 128;
            sbyte b = (sbyte)a; // 오버플로우 발생
            Console.WriteLine(a); Console.WriteLine(b); Console.WriteLine();

            float c = 0.1f;
            double d = (double)c;
            Console.WriteLine(c); Console.WriteLine(d); Console.WriteLine();

            int e = -30;
            uint f = (uint)e; // 언더플로우 발생
            Console.WriteLine(e); Console.WriteLine(f); Console.WriteLine();

            int g = (int)c;
            Console.WriteLine(g); Console.WriteLine();

            string h = a.ToString();
            string i = c.ToString();
            int j = Convert.ToInt32(h); // int.Parse(h); 도 가능
            float k = float.Parse(i);
            Console.WriteLine(h); Console.WriteLine(i); Console.WriteLine(j); Console.WriteLine(k); Console.WriteLine();

        }
    }
}
```

- 결과

```
C:₩Windows₩system32₩cmd

128
-128

0.1
0.100000001490116

-30
4294967266

0

128
0.1
128
0.1
```

3.5 상수와 열거 형식

- 변수는 담고 있는 데이터를 얼마든지 변경할 수 있는 메모리 공간이다.
- 상수(Constants)와 열거형식(Enumerator)은 변수와 달리 안에 담긴 데이터를 절대 바꿀 수 없는 메모리 공간이다.
- 이는 값을 바꾸지 말아야 할 변수를 실수로 건드리는 것을 방지해준다.

3.5.1 상수

- 데이터 형식 앞에 const 키워드가 위치하고, 반드시 상수가 가져야하는 데이터를 반드시 대입해줘야한다.
- 한번 선언한 뒤로는 바꿀 수 없다. 바꾸려고 하면 에러 출력

```
const 자료형 상수명 = 값;
```

3.5.2 열거 형식

- 종류는 같지만 다른 값을 갖는 상수를 선언해야 할 때가 가끔 있다.
- 실수를 방지하기 위해 enum이라는 키워드를 사용한다. 첫번째 요소는 0부터 시작

3.6 Nullable 형식

- 비어있는 상태를 저장해야 할 때 Nullable 형식을 사용한다.
- 형식 이름 뒤에 ? 만 붙여주면 된다.

- 데이터형식? 변수이름;
- Int? a = null; float? b = null;

- 이는 int a와는 다른 의미, int?와 달리 int는 비워둘 수 없는 데이터 형식이다.
- Nullable의 형식은 HasValue와 Value 속성을 지니고 있다. 이를 통해 **변수가 있는지 없는지, 그 값이 무엇인지** 나타낼 수 있다.

3.8 공용 형식 시스템

- C#의 모든 데이터 형식 체계는 공용 형식 시스템(Common Type System)이라는 .NET의 형식 체계 표준을 그대로 따르고 있다.
- 공용 형식 시스템은 **모두가 함께 사용하는 데이터 형식 체계**라는 의미이다.
- 모두는 C#을 비롯한 .NET을 지원하는 모든 언어들끼리 서로 호환성을 갖도록 하기 위한 범위이며 C#의 데이터 형식 체계가 CTS 표준을 따르고 있다.

클래스 이름	C# 형식	C++ 형식	비주얼 베이직 형식
System.Byte	byte	unsigned char	Bute
System.Sbyte	sbyte	char	Sbyte
System.Int16	short	short	Short
System.UInt64	ulong	unsigned__int64	Ulong
System.Single	float	float	Single
System.Boolean	bool	bool	Boolean
System.Object	object	Object*	Object
System.String	string	String*	String

3.9. 문자열 다루기

- String은 단순히 문자열을 담는 것 뿐만 아니라, 문자열을 가공하기 위한 다양한 기능도 제공된다.

3.9.1 문자열 안에서 찾기

- string 형식이 제공하는 탐색 메소드의 종류와 역할은 다음과 같다.

  IndexOf() : 현재 문자열 내에서 찾고자 하는 지정된 문자 또는 문자열의 위치를 찾는다. 위치하는 문자 순번를 반환한다.

  LastIndexOf() : IndexOf()를 뒤에서 부터 찾는다. 위치하는 문자 순번를 반환한다.

  StartsWith() : 현재 문자열이 지정된 문자열로 시작하는지 평가한다. Ture, False를 반한한다.

  EndsWith() : 현재 문자열이 지정된 문자열로 끝나는지를 평가한다. (True, False)

  Contains() : 현재 문자열이 지정된 문자열을 포함하는지 평가한다. (True, False)

  Replace() : 현재 문자열에서 지전된 문자열이 다른 지정된 문자열로 모두 바뀐 새 문자열을 반환한다. 문자열.Replace("A","B"); : A를 B로 바꾼다.

3.9.1 문자열 안에서 찾기

• string 형식이 제공하는 탐색 메소드의 종류와 역할은 다음과 같다.

메소드	설명
IndexOf()	현재 문자열 내에서 찾고자 하는 지정된 문자 또는 문자열의 위치를 찾는다. 위치하는 문자 순번를 반환한다.
LastIndexOf()	IndexOf()를 뒤에서 부터 찾는다. 위치하는 문자 순번를 반환한다.
StartsWith()	현재 문자열이 지정된 문자열로 시작하는지 평가한다. Ture, False를 반한한다.
EndsWith()	현재 문자열이 지정된 문자열로 끝나는지를 평가한다. (True, False)
Contains()	현재 문자열이 지정된 문자열을 포함하는지 평가한다. (True, False)
Replace()	현재 문자열에서 지전된 문자열이 다른 지정된 문자열로 모두 바뀐 새 문자열을 반환한다. 문자열.Replace("A","B"); : A를 B로 바꾼다.

3.9.2 문자열 변형하기

- string 형식의 문자열 변형 기능 관련 메소드는 다음과 같다.

메소드	설명
ToLower()	현재 문자열의 모든 대문자를 소문자로 바꾼 새 문자열을 반환
ToUpper()	현재 문자열의 모든 소문자를 대문자로 바꾼 새 문자열을 반환
Insert()	현재 문자열의 지정된 위치에 지정된 문자열이 삽입된 새 문자열을 반환
Remove()	현재 문자열의 지정된 위치로부터 지정된 수만큼의 문자가 삭제된 새 문자열을 반환
Trim()	현재 문자열의 앞/뒤에 있는 공백을 삭제한 새 문자열을 반환
TrimStart()	현재 문자열의 앞에 있는 공백을 삭제한 새 문자열을 반환
TrimEnd()	현재 문자열의 뒤에 있는 공백을 삭제한 새 문자열을 반환

3.9.3 문자열 분할하기

- "MSFT,GOOG,AMZN,AAPL,RHT"와 같은 문자열이 있을때, 콤마(,)로 구분된 부분을 제외한 내용을 단번에 배열로 만들 수 있다.
  - Split() : 지정된 문자를 기준으로 현재 문자열을 분리한 다음, 분리한 문자열의 배열을 반환한다.

```
string[] arr = 문자열.Split( new string[] {" "}, StringSplitOptions.None);
```

  - SubString() : 현재 문자열의 지정된 위치로부터 지정된 수만큼 문자로 이루어진 새 문자열을 반환한다.

```
문자열.Substring(a, b);
```

  - 문자열의 a번재부터 b번째를 반환한다.

3.9.4 문자열 서식 맞추기

- 여기에서 서식은 글꼴, 색상, 모양새와 달리, 문자열이 일정한 틀과 모양을 갖추는 것을 의미한다.

{첨자, 맞춤 : 서식 문자열}

  - Format() : 문자열 틀에 입력하는 {0}, {1} .. 를 "서식항목(Format Item)" 이라 한다.
    - 왼쪽/오른쪽 맞춤
      string.Format("{0,-10}DEF","ABC"); // "ABC.......DEF" D가 10번째부터 시작
      string.Format("{0,-10}DEF","ABC"); // ".......ABCDEF" "ABC"가 뒤로 쌓임
    - 숫자 서식화
    - 날짜 및 시간 서식화
    - 길이 = 맞춤과 같음, 서식 = 서식 문자열과 같음

감사합니다