프로그래밍 기초 with C#

02. 데이터 다루기

학습목표

- 정수와 실수를 다룰 수 있다.
- 십진수와 이진수 간의 변환을 할 수 있다.
- 컴퓨터가 문자를 어떻게 처리하는지 이해할 수 있다.
- 타입 변환에 대해서 이해할 수 있다.

들어가며

프로그램을 개발하는 것은 문제를 해결하기 위해 어떤 데이터를 정의하고, 그 데이터를 어떻게 수정할지 정하는 것이다. 따라서 데이터의 중요성은 아무리 강조해도 지나치지 않다. 이번 강의에서는 프로그램에서 데이터 저장과 조작 방법에 대해서 알아보도록 하자.

객체

메모리에 데이터를 저장하려면 주소와 크기를 알아야 한다. 하지만 우리가 어떤 데이터를 사용할 때마다 데이터가 저장된 주소와 크기를 외우는 것이 쉽지는 않다. 그래서 프로그래밍 언어에서는 이를 쉽게 사용하기 위해 **객체**(Object)를 지원한다. 객체란 데이터가 저장된 영역을 의미한다. 객체를 생성하는 문법은 아래와 같다.

```
타입 이름;
```

타입(Type)은 데이터를 저장하기 위해 메모리를 얼마나 사용할 것인지, 데이터의 종류는 무엇인지 지정하는 것이다. 타입은 프로그래밍 언어마다 종류가 다르며, C#의 경우 여기서 살펴볼 수 있다.

객체를 사용하면 컴파일러가 알아서 주소를 관리하고, 크기를 명시할 수 있기 때문에 프로그래머는 이름*으로 데이터를 관리할 수 있다. 따라서 이 객체에 어떤 데이터를 저장했는지 잘 드러나도록 명명해야 한다.** 이름은 프로그래밍 언어에서 특수한 의미를 가지는 **키워드**(Keyword)를 제외하고 일련의 규칙에 맞춰 작성할 수 있다. (1) 숫자로 시작할 수 없으며, (2) 대소문자를 구별한다.

* 엄밀히는 식별자(Identifier)라고 한다.

** 저장할 데이터에 단위가 존재한다면 단위도 접미사로 붙이도록 하자. 가령, 초 단위로 어떤 시간을 기록한다면 timeInSeconds가 되어야 한다. 그리고 용어의 사용을 일관되게 쓰자. 가령 무언가를 가져오는 것에 이름을 붙일 때, 누구는 get을 사용하고 누구는 fetch를 사용하고 누구는 retrieve를 사용하면 읽는 입장에서 불편할 것이다.

```
// 이름이 데이터를 나타내기에 불충분하다.
t = s + l - b;

// 이제는 이름만 봐도 어떤 데이터인지 한눈에 알 수 있다.
TotalLeaves = SmallLeaves + LargeLeaves - SmallAndLargeLeaves;
```

정수

일단 정수부터 다뤄보도록 하자. 정수를 표현하는 타입으로는 byte, short, int, long이 있는데, 각각 크기*가 다르다. byte의 경우 1바이트, short의 경우 2바이트, int의 경우 4바이트, long의 경우 8바이트다. 이중 많이 쓰이는 것은 int다. 아래의 프로그램을 작성해 int 타입의 객체를 생성해보자.

* 참고로 크기의 경우 sizeof 연산자를 통해 구할 수 있다.

```
using System;

class Program
{
  static void Main()
  {
    int num;
```

```
    Console.WriteLine(num);
  }
}
```

하지만 오류가 뜬다는 것을 알 수 있다. 메모리를 사용한다고 컴퓨터에게 알려줬지만, 정작 어떤 데이터를 저장할 건지는 알려주지 않았기 때문이다. 객체를 생성할 때 초기값을 지정해주는 걸 **초기화**(Initialization)라고 한다. 초기화는 아래처럼 하면 된다. 초기화를 하지 않아 오류를 발생하는 경우도 종종 있으므로 꼭 객체를 만들 때는 초기화 하는 것을 잊지 말자.*

* 초기화를 어떤 값으로 해야할지 모르겠다면 기본값을 사용하자.

```
int num = 100; // int 타입의 객체 num을 만들고, 이를 100으로 초기화했다.
```

이진수

여기서 의문점이 하나 있다. 우리가 사용하는 수 체계는 십진법이다. 즉, 각 자리수를 0에서 9까지 사용하고, 자리수가 올라갈 수록 10배씩 커지게 된다. 하지만 비트는 2가지의 의미만 담을 수 있기에 이진법을 사용한다. 따라서 십진수를 이진수로, 이진수를 십진수로 바꿀 줄 알아야 한다. 이진법에 대한 설명은 여기로 대체한다.

이진수에 대해서 알았다면 이번엔 음수에 대해서 생각해보자. -12 같이 음수를 어떻게 이진수로 나타낼 수 있을까? 여기엔 2의 보수법을 사용하며, **MSB**(Most Significant Bit)*를 부호 비트로 사용해 0이면 양수, 1이면 음수로 취급한다. 2의 보수에 대한 설명 또한 여기로 대체한다.

* 최상위 비트를 말한다. 반대는 LSB(Least Sigificant Bit)다.

위 내용을 바탕으로 정수 타입의 표현 가능 범위는 여기서 확인할 수 있다.

실수

실수를 표현할 수 있는 타입에는 float, double이 있다. 실수의 경우 정수와는 다르게 IEEE 754*라는 표준을 사용해 표현한다. 핵심은 실수는 언제나 부정확하다는 것이다. 아래의 프로그램을 작성해 결과를 확인해보자.

* 지수부와 가수부로 표현하기 때문에 **부동소수점**(Floating-Point) 타입이라고도 한다.

```
using System;

class Program
{
  static void Main()
  {
    double num = 0.1 + 0.2;
    Console.WriteLine(num);
  }
}
```

문자

컴퓨터는 문자를 어떻게 처리할까? 문자를 표현하기 위해선 문자 집합, 인코딩, 폰트에 대해서 알아야 한다. **문자 집합**(Character Set)은 컴퓨터가 인식하고 표현할 수 있는 문자의 모음*으로 각 문자에는 이진수**가 할당되어 있다. 그래서 문자를 이진수로 변환하는 과정을 **문자 인코딩**(Character Encoding), 반대로 이진수를 문자로 변환하는 과정을 **문자 디코딩**(Character Decoding)이라 한다. 그리고 문자의 모양을 정의한 것이 **폰트**(Font)라고 할 수 있다. 대표적인 문자 집합에 대해서 살펴보자.

* **코드 페이지**(Code Page)라고도 한다.

** **코드 포인트**(Code Point)라고 한다.

아스키 코드

아스키(ASCII; American Standard Code for Information Interchange) 코드는 알파벳과, 아라비아 숫자, 일부 특수 문자를 포함하는 문자 집합이다. 7비트를 사용해 총 128개의 문자를 표현한다. 여기서 확인할 수 있다.

EUC-KR

아스키 코드는 인코딩이 매우 간단하지만, 영어밖에 표현할 수 없다. 따라서 각 나라에서는 그들의 언어를 표현하기 위한 인코딩이 필요했다. 한국도 마찬가지였다. 한글은 각 음절이 초성, 중성, 종성의 조합으로 구성되는데, 이를 표현하는 방법에는 완성된 하나의 글자에 코드 포인트를 부여하는 완성형과 초성, 중성, 종성을 위한 각각의 비트열을 할당하여 하나의 글자를 만드는 조합형이 있다.

EUC-KR은 KS X 1001, KS X 1003의 문자 집합을 기반으로 하는 완성형 인코딩으로, 2바이트를 사용하여 2,350개 정도의 한글 단어를 표현할 수 있었다. 하지만, 이 역시도 11,172자나 되는 모든 한글 조합을 표현하기엔 부족했기에 때때로 문제가 발생하기도 했다.*

* 실제 청원이 올라온 적이 있다. 마이크로소프트도 이런 문제점을 알고 있었기에 남은 8,000여자를 지원하는 EUC-KR의 확장형 격인 CP949 인코딩을 내놓았다.

유니코드

각 나라마다 다른 인코딩 방법을 사용하는 것은 국제화에 장애물이었다. 따라서 대부분의 언어를 아우르는 문자 집합과 통일된 표준 인코딩 방식이 필요했는데 이것이 **유니코드**(Unicode)이다. 유니코드 문자에는 각 나라의 문자 외에도 이모티콘이나 여러 특수 기호도 포함되어 있다. 인코딩 방법에는 UTF-8, UTF-16, UTF-32 등이 있는데, 이중 UTF-8을 가장 많이 사용하고 있다. C#에서 문자 타입은 char이며, UTF-16을 사용하고 있다.

리터럴

한편, 우리가 작성한 소스 코드 중 값을 나타내는 문자를 **리터럴**(Literal)이라고 한다. 리터럴도 객체와 마찬가지로 타입이 있다. 아래의 프로그램을 작성해보자.

```
using System;

class Program
{
  static void Main()
  {
    float num = 0.1 + 0.2;
    Console.WriteLine(num);
  }
}
```

이처럼 객체의 타입과 리터럴의 타입이 맞지 않으면 오류가 발생하게 된다. 리터럴의 타입은 접두사 혹은 접미사를 붙여 타입을 바꿔줄 수 있다. 정수 리터럴에 관한 내용은 여기, 실수 리터럴에 관한 내용은 여기를 참고하자.

연산

프로그래밍 언어에는 산술, 비교 등등 여러 가지 연산자가 있다. 수식을 계산할 때 괄호로 닫혀 있는 걸 먼저 계산하는 것처럼 연산자끼리는 우선순위가 있고, 결합 방법도 있다. 각 연산자마다 필요한 피연산자 개수와 타입이 다름에 유의하도록 하자. 이를 모르고 있다면 의도치 않은 결과가 일어날 수 있다. 지금 모든 연산을 배우진 않을 것이며, 필요할 때 하나씩 배워가도록 하겠다. 우선은 산술, 대입, 비트 연산부터 알아보도록 하자.

변환

타입을 정확하게 사용하는 것은 안전하고 강건한 프로그램을 작성하는 기본이 된다. 때에 따라서는 데이터의 타입을 바꿔줘야 할 때가 있는데, 이를 **타입 변환**(Type Conversion)이라고 한다. 변환에는 4가지 종류가 있는데 우선 암시적 변환과 명시적 변환에 대해서만 알아보자.

암시적 변환

암시적 변환(Implicit Conversion)은 변환이 항상 가능하고, 데이터의 손실이 일어나지 않는 경우로 특별한 문법이 없다. 보통 작은 숫자 타입의 데이터를 큰 숫자 타입의 객체에 저장하는 경우라고 할 수 있다.

```
// byte는 1바이트의 크기를 가지기 때문에
// 상위의 모든 정수 타입에 저장할 수 있다.
byte smallNumber = 100;
short shortObject = smallNumber;
int intObject = smallNumber;
long longObject = smallNumber;

// float는 4바이트의 크기를 가지기 때문에
// double에 저장할 수 있다.
float floatNumber = 12.34f;
double doubleNumber = floatNumber;
```

암시적 변환이 불가능한 경우 컴파일 오류가 발생한다.

```
// 오류!
// double 타입의 리터럴을 float 타입으로 변환할 수 없다.
float floatNumber = 12.34;
```

암시적 변환이 가능한 경우는 여기서 참고하라.

명시적 변환

암시적 변환이 불가능한 경우 캐스트 연산자를 사용해 **명시적 변환**(Explicit Conversion)을 할 수 있다. **캐스팅**(Casting)이라고도 한다.

```
// 캐스팅을 통해 오류를 해결했다.
float floatNumber = (float)12.34;
```

명시적 변환에도 규칙이 있다. 다시 말해 내가 원하는 대로 바꿀 수 없다.

```
// 아래의 경우는 캐스팅을 해도 되지 않는다.
int number = (int)"123";
```

명시적 변환이 가능한 경우는 여기서 확인할 수 있다.

더 나아가기

1. 프로그래머스의 아래 문제를 풀어봅시다.
    1.1. 코딩테스트 연습 - 두 수의 합 | 프로그래머스 스쿨
    1.2. 코딩테스트 연습 - 두 수의 차 | 프로그래머스 스쿨
    1.3. 코딩테스트 연습 - 두 수의 곱 | 프로그래머스 스쿨
    1.4. 코딩테스트 연습 - 몫 구하기 | 프로그래머스 스쿨
    1.5. 코딩테스트 연습 - 두 수의 나눗셈 | 프로그래머스 스쿨
    1.6. 코딩테스트 연습 - 나머지 구하기 | 프로그래머스 스쿨
    1.7. 코딩테스트 연습 - 나이 출력 | 프로그래머스 스쿨

1.8. 코딩테스트 연습 - 양꼬치 | 프로그래머스 스쿨

1.9. 코딩테스트 연습 - 피자 나눠 먹기 (3) | 프로그래머스 스쿨

1.10. 코딩테스트 연습 - 개미 군단 | 프로그래머스 스쿨

1.11. 코딩테스트 연습 - 세균 증식 | 프로그래머스 스쿨

2. 수업 내용을 복기하며 아래 내용을 정리해봅시다.

2.1. 객체(Object)와 타입(Type)의 역할을 정의하고, 프로그래밍 언어가 이 둘을 지원하는 이유를 설명해 보세요.

2.2. C#에서 정수형 타입 중 가장 많이 사용되는 타입은 무엇이며, 실수를 표현하는 타입(float, double)이 가지는 근본적인 한계점은 무엇일까요?

2.3. IEEE 754로 실수를 어떻게 표현하나요? 예시와 함께 설명해 주세요.

2.4. 초기화하지 않은 객체를 사용하면 어떤 오류가 발생하나요? 실제 스크린샷과 함께 답변을 작성해 주세요.

2.5. 183과 -58을 2진수와 16진수로 바꿔보세요. 비트는 8개를 사용합니다.

2.6. 컴퓨터가 문자를 처리하는 과정에 필요한 핵심 개념 3가지를 정의해 보세요.

2.7. 한글 인코딩의 한 종류인 EUC-KR의 문제점은 무엇이었나요?

2.8. 리터럴이란 무엇인가요?

2.9. 콘솔에서 System.Console.ReadLine()으로 사용자에게 숫자를 입력받았을 때, 이 데이터를 정수형 변수에 저장하기 위해 필요한 타입 변환 방법과 그 이유를 설명해 주세요.

2.10. 암시적 변환과 명시적 변환의 차이점을 데이터 손실 가능성 측면에서 설명하고, 각각이 가능한 상황의 구체적인 코드 예시를 하나씩 제시해 보세요.

3. 오버플로우(Overflow)는 무엇인가요? 여기를 참고하여 조사해보세요.

4. 아래의 데이터를 저장하기 위해 어떤 타입과 이름을 지으면 좋을까요?

4.1. 재장전하는 시간

4.2. 적의 총 개체 수

4.3. 캐릭터의 최대 점프 횟수

4.4. 플레이어가 갖고 있는 금화의 양

5. 아래의 num에는 각각 어떤 값이 저장될까요?

```
sbyte num = -128;
--num;
```

num의 값은?

```
byte num = 2 << 4;
```

num의 값은?

```
byte num = 5 ^ 12;
```

num의 값은?

```
byte num = 12 / 8;
```

num의 값은?

```
byte num = 15 >> 2;
```

num의 값은?

참고자료

- C# 교과서
- 코드 작성 가이드 | 이시가와 무네토시 - 교보문고