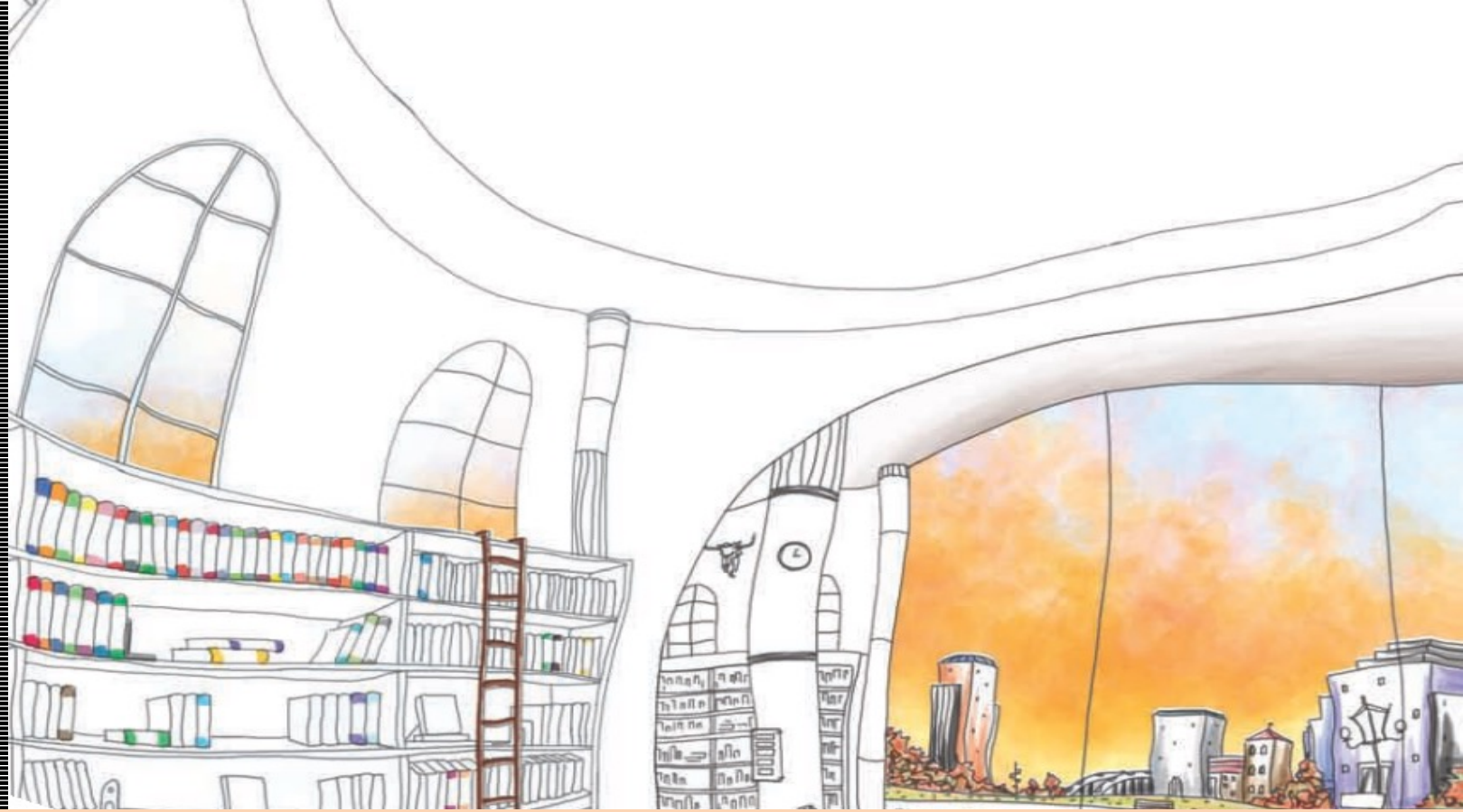


데이터 보관하기 -2

이것이 C#이다

Contents

3.4.14 데이터 형식 바꾸기

- 변환(Type Conversion)
  - 변수를 다른 데이터 형식의 변수에 옮겨 담는 것
- 형식 변환의 종류
  - 크기(표현 범위)가 서로 다른 정수 형식 사이의 변환
  - 크기(표현 범위)가 서로 다른 부동 소수점 형식 사이의 변환
  - 부호 있는 정수 형식과 부호 없는 정수 형식 사이의 변환
  - 부동 소수점 형식과 정수 형식 사이의 변환
  - 문자열과 숫자 사이의 변환

3.4.15 크기가 서로 다른 정수 형식 사이의 변환

- 작은 정수 형식 변수 데이터 → 큰 정수 형식 변수
- 큰 정수 형식 변수 데이터 → 작은 정수 형식 변수
  - → 오버플로우 발생 가능성
  - 작은 정수 형식의 변수 범위인 경우는 예외
- 데모 예제 - IntegerConversion

3.4.16 크기가 서로 다른 부동 소수점 형식 사이의 변환

- 부동 소수점 형식의 특성상 오버플로우 존재 안함
- 정밀성에 손상을 입음.
  - float과 double은 소수를 이진수로 메모리에 보관
  - 다른 형식 변환: 10진수 복원 → 이진수로 변환
  - → 이진수로 표현하는 소수는 완전하지 않음(예, 1/3)
- 데모 예제 – FloatConversion

3.4.17 부호 있는 정수 형식과 부호 없는 정수 형식 사이의 변환

- 최저 값 보다 작은 값을 저장해 언더플로우 발생 가능성
- 데모 예제 - SignedUnsignedConversion

3.4.18 부동 소수점 형식과 정수 형식 사이의 변환

- 부동 소수점 -> 정수 형식 변환
  - → 소수점 이하 버림
  - 반올림 없음.
- 데모 예제 - FloatToInteger

3.4.19 문자열을 숫자로, 숫자를 문자열로

- “12345” vs. 12345

```
string a = "12345";
int b = (int)a; // b는 이제 12345?
```

```
int c = 12345;
string d = (string)c;
```

- 문자 → 숫자 해결책: Parse(), Convert.ToInt32()
- 숫자 → 문자 해결책: ToString()
- 데모 예제 – StringNumberConverion

3.5 상수와 열거 형식

- 데이터를 절대 바꿀 수 없는 메모리 공간
- 상용 소프트웨어 제작의 복잡성에서 변수의 일관성
  - 값을 바꿔도 되는 변수
  - 값을 바뀌지 않아야 하는 변수의 구분 → 미리 정해 놓자.
- 상수와 열거 형식은 프로그래머를 편하게 한다.
  - 컴파일러가 미리 알아챈다.
  - → 버그 최소화

3.5.1 상수 – 언제나 변하지 않을 거예요

❖ 상수 선언 형식

- const **자료형 상수명 = 값**;

```
const int a = 3;
const double b = 3.14;
const string c = "abcdef";
```

❖ 데모 예제 - Constant

3.5.2 열거 형식

- **종류는 같지만 다른 값을 갖는 상수**
  - **같은 범주에 속하는 여러 개의 상수를 선언할 때** .
  - **사용자의 응답** → (Yes, No, Confirm, Cancel, OK)
- **열거형 선언 형식**
  - enum **열거 형식명** : **기반자료형** { **상수**1, **상수**2, **상수**3, … }
  - enum **열거형식명** { **상수**1 = **값**1 , **상수**2 = **값**2 , **상수**3 = **값**3 , … }
  - enum **열거형식명** { **상수**1 = **값**1 , **상수**2 , **상수**3 = **값**3 , … }
  - **기반자료형은 정수 계열**(byte, sbyte, short, ushort, int, uint, long, ulong, char)**만 사용**
  - **기본값은** int
- **데모 예제 – Enum1, Enum2, Enum3**

3.6 Nullable 형식

- **어떤 값도 가지지 않는 변수가 필요할 때**
  - 0이 아닌 비어 있는 변수, 즉 null 상태
- **Nullable 선언 형식**
  - **데이터형식? 변수이름;**

```
int? a = null;
float? b = null;
double? c = null;
```

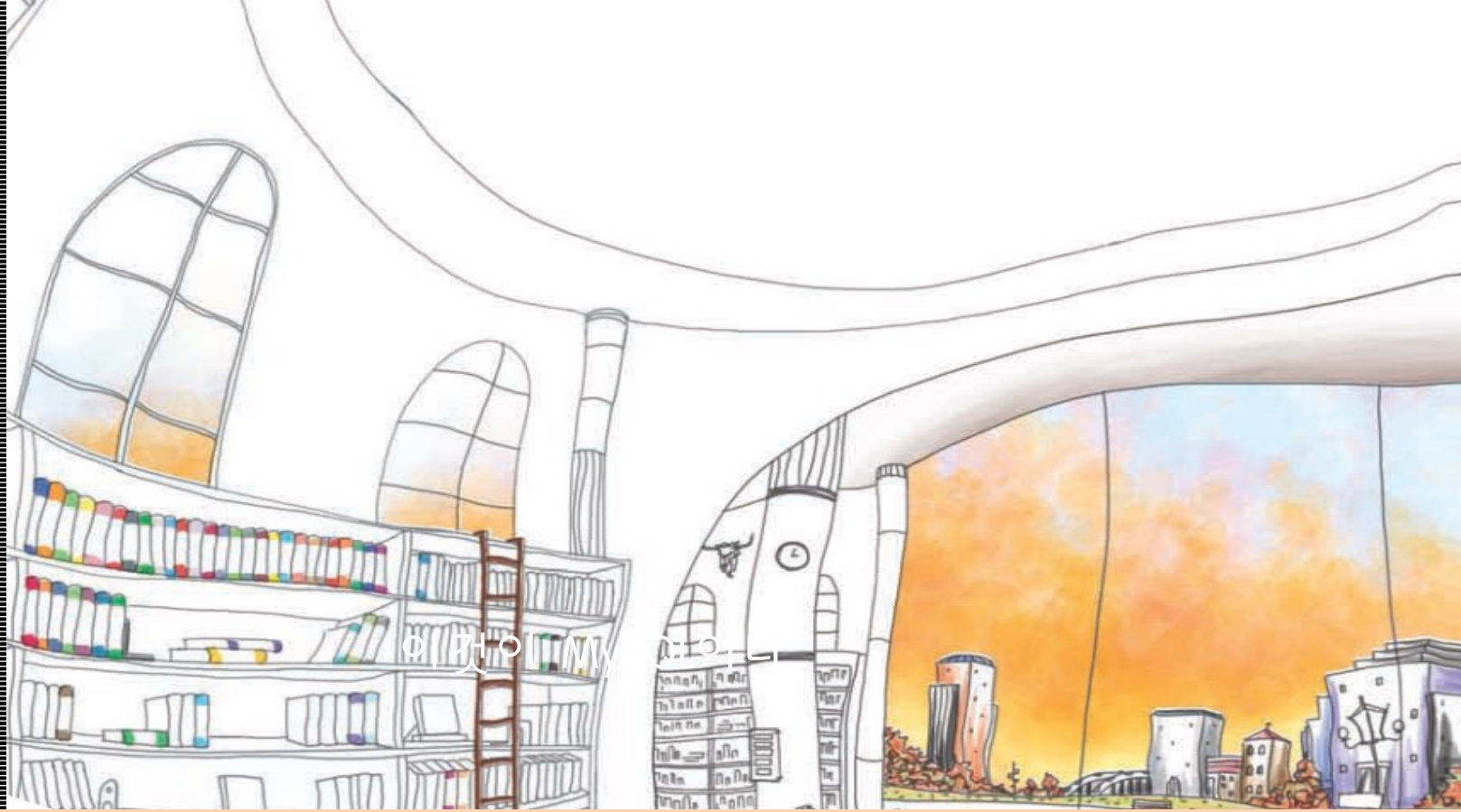
- **Nullable 형식의 2가지 속성.**
  - HasValue: **변수에서 값을 갖고 있는지 여부**
  - Value: **변수에 담겨 있는 값**
- **데모 예제 – Nullable**

3.7 var 형식

- C#은 강력한 형식 언어
  - 의도치 않은 형식의 데이터를 읽거나 할당하는 일 차단
- 약한 형식 검사
  - 코드 작성 단계에서 편리
  - 컴파일러에서 변수에 담긴 데이터에 따라 자동으로 형식 지정
- C#의 약한 형식 검사 방법 → Var
  - 선언과 동시 초기화 필수
  - 지역 변수로만 사용
- 데모 예제 - UsingVar

3.8 공용 형식 시스템

- C#의 모든 데이터 형식 체계
  - 공용 형식 시스템이라는 .NET 프레임워크의 형식 체계의 표준 준수
  - → .NET 언어들 간의 호환성
- CTS를 따르는 C# vs. Visual Basic vs. C++
- 데모 예제 – CTS
  - GetType()



Thank You !

이것이 C#이다