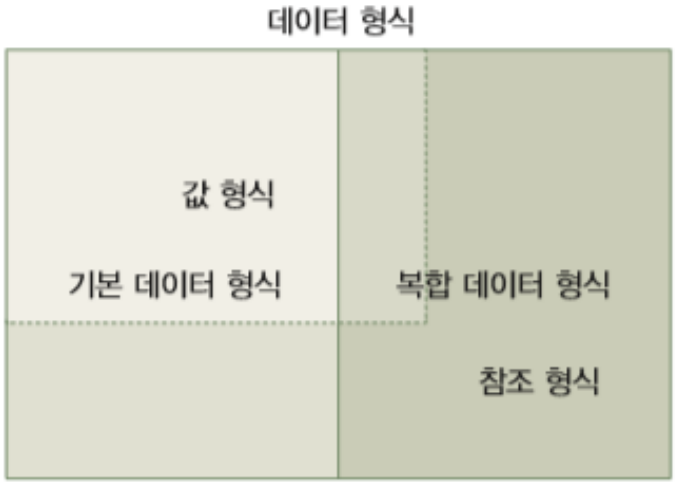
데이터 보관하기 -1

이것이 C#이다

Contents

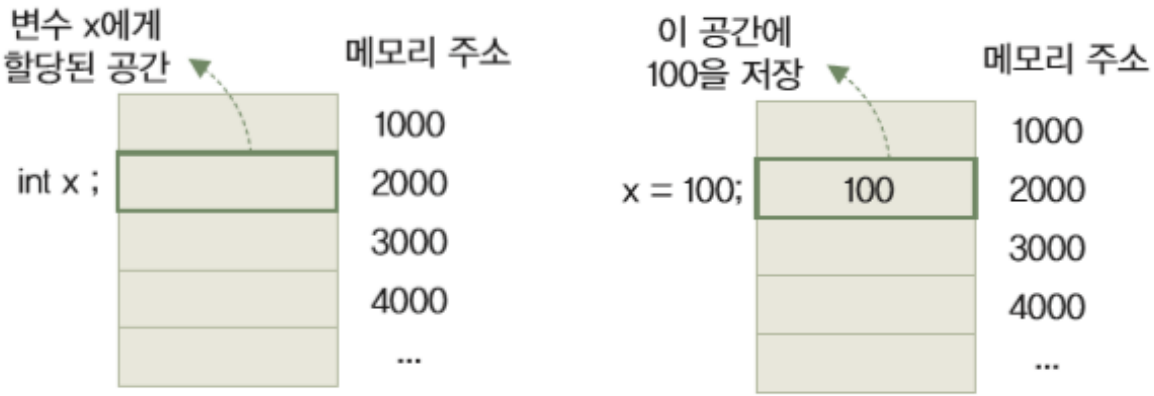
3.1 데이터에도 종류가 있다.

- 계산 머신에서 범용 목적의 머신으로 발전
  - 다양한 종류의 데이터 처리 필요.
- C#의 데이터 형식
  - 기본 데이터 형식
  - 상수, 열거형
  - 복합 데이터 형식
    - 구조체, 클래스, 배열
  - 값 형식
  - 참조 형식



3.2 변수

- **코드 관점**
  - 값을 대입시켜 변화시킬 수 있는 요소
- **메모리 관점**
  - 데이터를 담는 일정 크기(데이터 형식)의 공간
- **변수를 "선언한다" (선언 대상) → 컴파일러**



- **선언과 동시 할당**

```
int x = 100;
```

- **다수의 변수 동시 선언과 할당**

```
int a, b, c;
int x = 30, y = 40, z = 50;
```

3.3 값 형식과 참조 형식

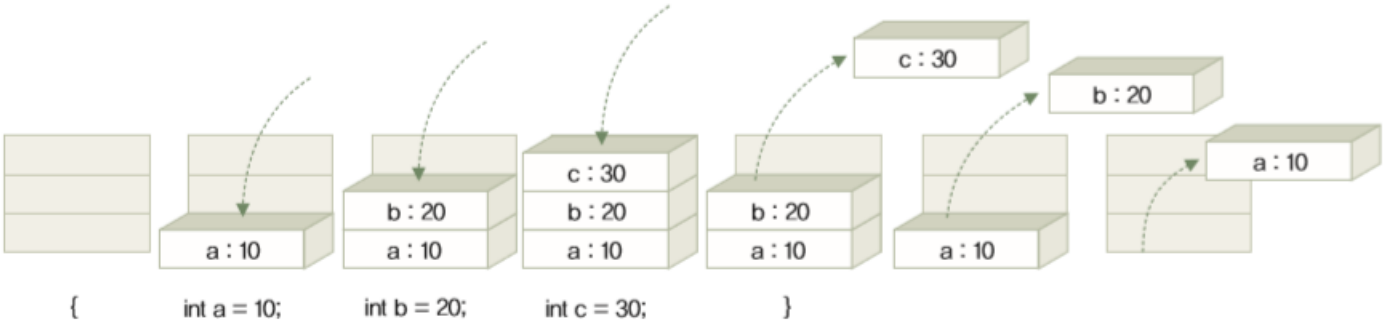
- 값 형식
  - 변수에서 값을 담는 형식
- 참조 형식
  - 값이 있는 곳의 위치(참조)를 담는 형식
- 두 가지 메모리 영역
  - 스택(Stack) ←→ 값 형식
  - 힙(Heap) ←→ 참조 형식

3.3.1 스택과 값 형식

❖ 스택

- 책 더미와 같은 구조로 생긴 메모리
- 값 형식의 변수 저장
- 저장된 데이터 자체 제거

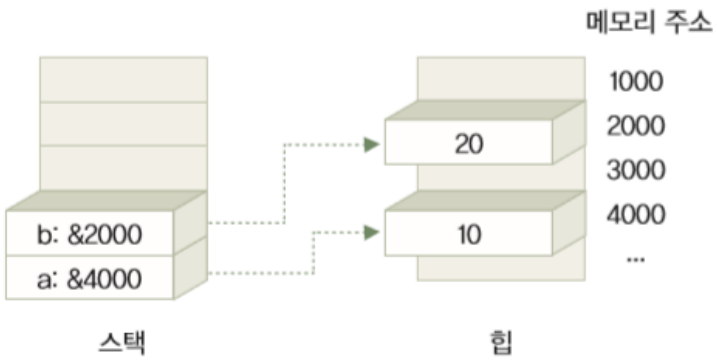
```
{ // 코드 블록 시작
    int a = 100;
    int b = 200;
    int c = 300;
} // 코드 블록 끝
```

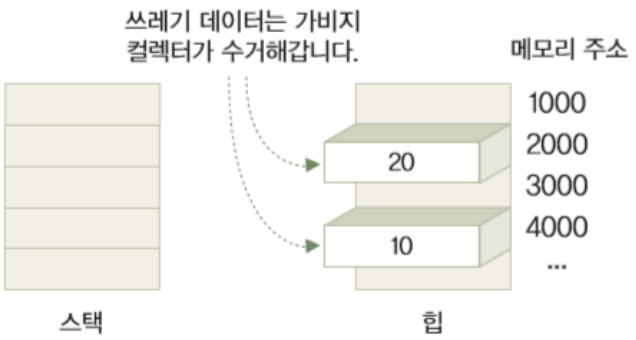


3.3.2 힙과 참조 형식

- 메모리 제거 메커니즘 필요
  - 가비지 컬렉터
- 힙을 사용하는 이유
  - 코드 블록이 끝나도 데이터 유지
- 참조 형식의 변수
  - 힙 – 데이터 저장
  - 스택 – 힙의 메모리 주소

```
{
    object a = 10;
    object b = 20;
}
```





3.4 기본 데이터 형식

- 세부적으로 15가지
- 참조 형식
  - 문자열 형식
  - 오브젝트 형식
- 값 형식
  - 숫자 형식
  - 논리 형식

3.4.1 숫자 데이터 형식

- **가장 많이 다루는 데이터 형식**
  - **텍스트 데이터도 숫자다.**
  - A → 63, b → 64
- **12가지 숫자 데이터 형식**
  - **정수 계열**
  - **부동 소수 계열**
  - **소수 계열**

3.4.2 정수 계열 형식 / 3.4.3 정수 형식 예제 프로그램

❖ 9가지의 정수 계열 데이터 형식 처리

- 크기와 데이터 범위에 따라.
- 효율적인 메모리 사용

데이터 형식	설명	크기(바이트)	담을 수 있는 값의 범위
byte	부호 없는 정수	1(8비트)	0~255
sbyte	signed byte 정수	1(8비트)	–128~127
short	정수	2(16비트)	–32,768~32,767
ushort	unsigned short 부호 없는 정수	2(16비트)	0~65,535
int	정수	4(32비트)	–2,147,483,648~2,147,483,647
uint	unsigned int 부호 없는 정수	4(32비트)	0~4,294,967,295
long	정수	8(64비트)	–922,337,203,685,477,508~922,337,203,685,477,507
ulong	unsigned long 부호 없는 정수	8(64비트)	0~18,446,744,073,709,551,615
char	유니코드 문자	2(16비트)	

❖ 데모 예제 – IntegerTypes

3.4.4 2진수, 10진수, 16진수 리터럴

- 2진수 리터럴 접두사, 0b
- 16진수 리터럴 접두사, 0X(0x)
- 10진수 리터럴 접두사는 필요 없음.
- 데모 예제 - IntegerLiterals

3.4.5 부호 있는 정수와 부호 없는 정수

❖ 부호 있는 정수: 양(+)과 음(-), 0

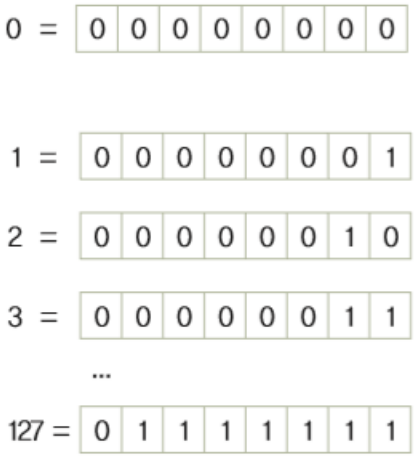
- sbyte, short, int, long
- 1개의 부호 비트

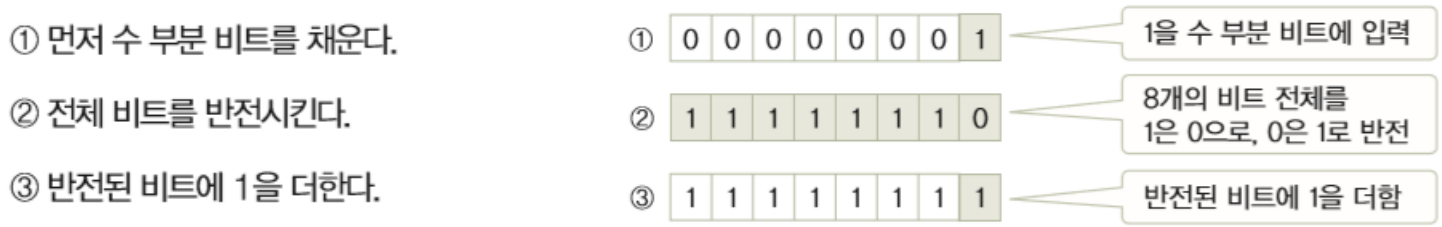
❖ 부호 없는 정수: 양(+)과 0

- byte, ushort, uint, ulong

❖ 예, sbyte와 byte의 값 표현

- 0의 표현
- 1~127까지의 표현
- Sbyte의 음수 표현-2의 보수법





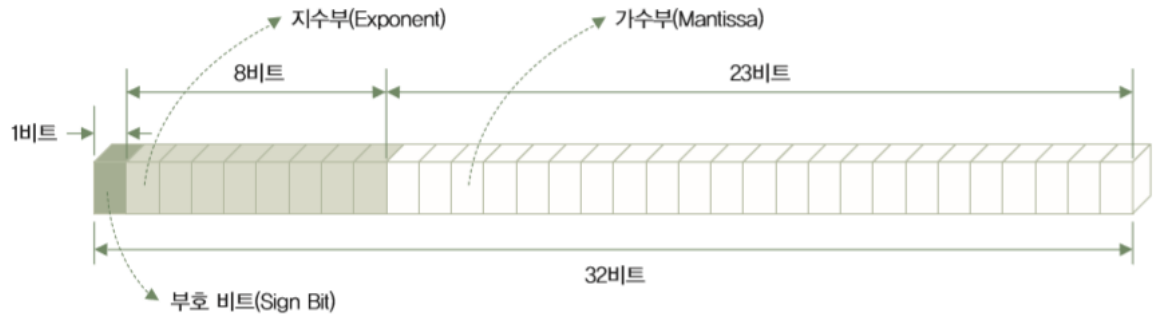
❖ 데모 예제- SignedUnsigned

3.4.6 데이터가 넘쳐 흘러요

- **오버플로우(Overflow)**
  - **각 데이터 형식의 최대값을 넘어가는 데이터를 저장할 때**
  - Byte**값** 255(1111 1111) + 1 → 0
- **언더플로우(Underflow)**
  - **최저 값 보다 작은 데이터를 저장할 때**
  - Byte**값에** -1**을 담으면** → 255
- **데모 예제 – Overflow**

3.4.8 부동 소수점 형식

- 소수점이 고정되어 있지 않고 움직이면서 수 표현
  - 제한된 비트를 이용해 넓은 범위의 값 표현
- 정수 형식을 대체하지 못하는 이유
  - 소수점 표현과 부호 표현에 일부 비트를 사용으로 정수 계열 형식과 같은 크기 사용 못함
  - 산술 연산 과정이 정수 계열 형식 보다 복잡해 느림
- float과 double



- 데모 예제 - FloatingPoint

3.4.9 Decimal 형식

- 실수를 다루는 데이터 형식
- Decimal 형식의 크기와 범위

데이터 형식	설명	크기(바이트)	범위
decimal	29자리 데이터를 표현할 수 있는 소수 형식	16 (128비)트	±1.0 × 10e–28~±7.9 × 10e28

- 데모 예제 - Decimal

3.4.10 문자 형식과 문자열 형식

- **문자 형식, char**
  - 작은 따옴표 ‘와 ’ 로 문자를 감싼다.
  - char a = '가';
- **데모 예제 – Char**
- **문자열 형식, string**
  - 여러 개의 문자 형식을 하나의 실로 묶어 처리
  - 문자열 데이터를 큰 따옴표 “ 와 ” 로 감싼다.
  - string a = "안녕하세요?";
- **데모 예제 – String**

3.4.11 논리 형식

❖ 참(True)과 거짓(False)을 다루는 데이터형식

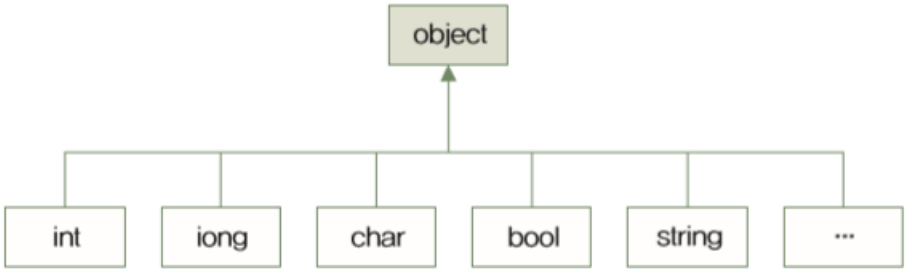
데이터 형식	설명	크기(바이트)	범위
bool	논리 형식	1(8비트)	true, false

❖ C 언어는 0과 0이 아닌 값으로 참과 거짓 표현

❖ 데모 예제 - Bool

3.4.12 Object 형식

- 어떤 물건이든지 다룰 수 있는 데이터 형식
  - 상속의 효과
  - 참조 형식 → 힙에 데이터 할당
- C#의 특별 조치
  - Object를 모든 데이터 형식의 조상
  - →Object 형식은 모든 데이터 형식을 담을 수 있다.



- 데모 예제 – Object

3.4.13 박싱과 언박싱

❖ 박싱(Boxing)

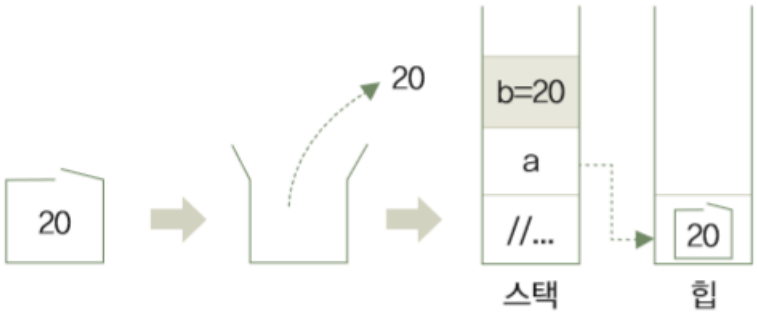
- 값 형식의 데이터를 Object 형식에 담는다면 → 스택? 힙?

```
object a = 20;
```

❖ 언박싱(Unboxing)

- 힙의 값 형식 데이터를 값 형식 객체에 할당하는 경우

```
object a = 20;
int    b = (int)a;
```



❖ 데모 예제 – BoxingUnboxing

Thank You !

이것이 C#이다