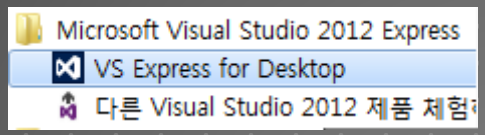
C#프로그래밍의 이해

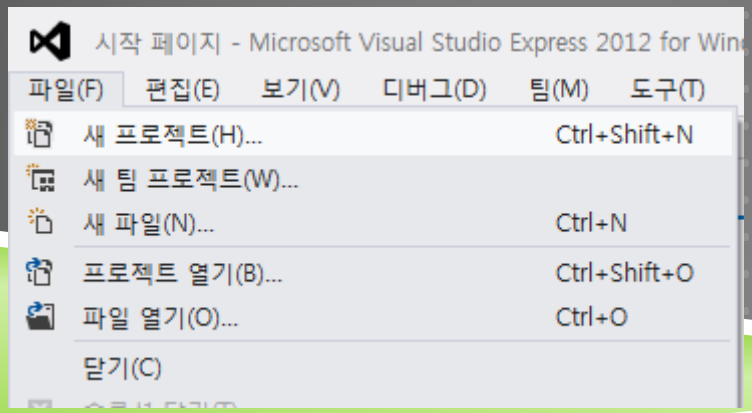
- C# 프로젝트 생성
- 데이터 구조와 표현법
- 데이터를 가공하는 연산자

01. HELLO, WORLD! (1/6)

- HelloWorld 프로젝트 생성
  - 1. 비주얼 스튜디오 실행

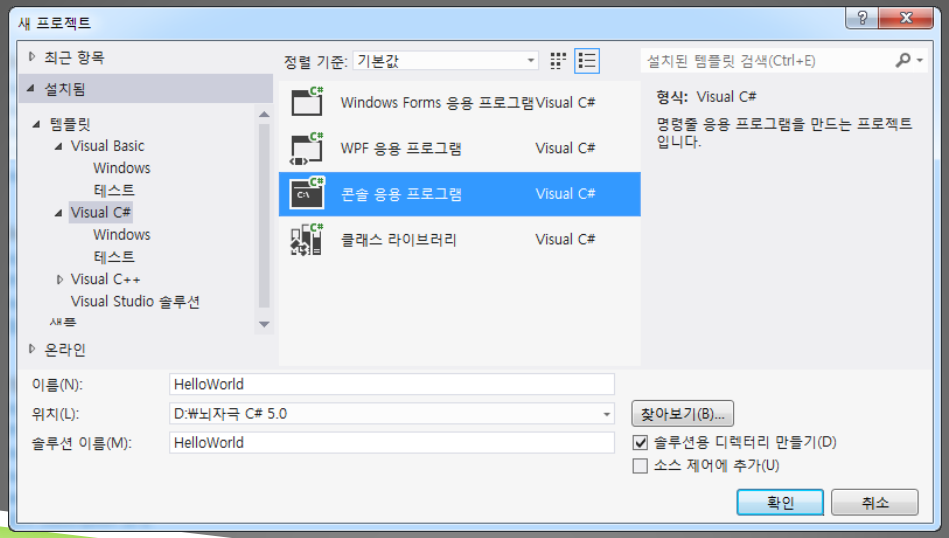


  - IDE의 [파일] → [새 프로젝트] 메뉴 클릭



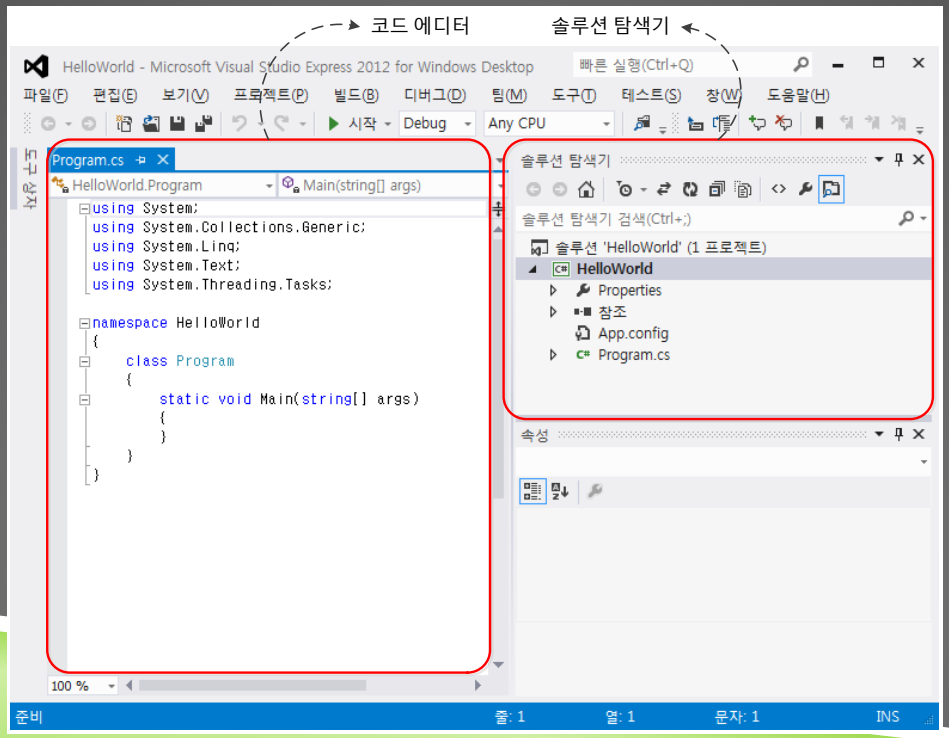
01. HELLO, WORLD! (2/6)

- HelloWorld 프로젝트 생성
  - 3. [새 프로젝트] 대화 상자를 띄웠으면 템플릿에서 [콘솔 응용 프로그램]을 선택하고 프로젝트의 '이름' 항목에는 "HelloWorld"를 입력한 후 <확인> 버튼을 클릭



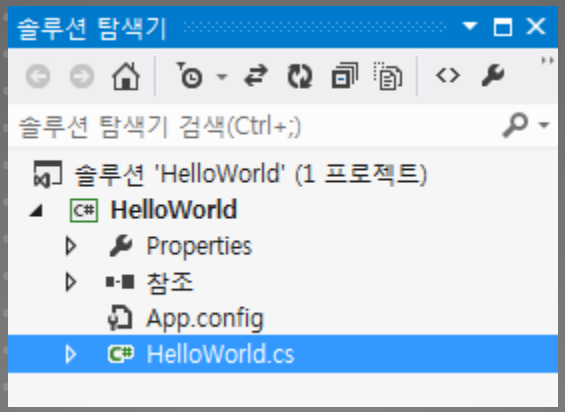
01. HELLO, WORLD! (3/6)

- HelloWorld 프로젝트 생성
  - 4. 프로젝트가 생성되면 아래와 같은 화면이 표시됨



01. HELLO, WORLD! (4/6)

- HelloWorld 프로젝트 생성
  - 5. 솔루션 탐색기에서 “Program.cs” 항목을 선택한 후, F2 키를 클릭. 파일 이름을 변경할 수 있도록 커서가 표시되면 “Program.cs”를 “HelloWorld.cs”로 이름을 변경



01. HELLO, WORLD! (5/6)

- HelloWorld 프로젝트 생성
  - 6. 다음의 코드를 HelloWorld.cs에 입력

```
using System;

namespace BrainCSharp
{
    class HelloWorld
    {
        // 프로그램 실행이 시작되는 곳
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

01. HELLO, WORLD! (6/6)

- HelloWorld 프로젝트 생성
  - 8. IDE에서 빌드를 눌러 빌드 실행(HelloWorld.exe가 생성되었는지 확인)
  - 9 키보드에서 [Windows]+[R] 키를 눌러 [실행] 창을 띄우고, 이 창에서 실행할 이름으로 "cmd"를 입력하여 명령 프롬프트 창 실행
    - 명령 프롬프트에서 "HelloWorld.exe" 파일이 생성된 디렉토리로 이동
    - 프로그램 실행 결과 확인

02. 첫 번째 프로그램 뜯어보기 (1/6)

- using System;

```
01 using System;
02
03 namespace BrainCSharp
04 {
```

  - using System;은 한 덩어리 같지만 실은 세 가지 요소로 구성.
    - using : 네임스페이스를 사용하겠다고 선언하는 키워드
    - System : 숫자, 텍스트와 같은 데이터를 다룰 수 있는 기본적인 데이터 처리 클래스를 비롯하여 C# 코드가 기본적으로 필요로 하는 클래스를 담고 있는 네임스페이스
    - 세미콜론(;) : 컴파일러에게 문장의 끝을 알리는 기호

02. 첫 번째 프로그램 뜯어보기 (2/6)

- namespace BrainCSharp{ }

```
03 namespace BrainCSharp
04 {
05     class HelloWorld
06     {
…
13     }
14 }
```

  - 네임스페이스는 성격이나 하는 일이 비슷한 클래스, 구조체, 인터페이스, 델리게이트, 열거 형식 등을 하나의 이름 아래 묶는 역할을 수행
    - 예) System.IO 네임스페이스에는 파일 입출력을 다루는 각종 클래스, 구조체, 델리게이트, 열거 형식들이 있고, System.Printing 네임 스페이스에는 인쇄에 관련된 일을 하는 클래스 등등이 소속
    - .NET 프레임워크 라이브러리에 1만 개가 훨씬 넘는 클래스들이 있어도 프로그래머들이 전혀 혼란을 느끼지 않고 이들 클래스를 사용할 수 있는 비결임

02. 첫 번째 프로그램 뜯어보기 (3/6)

- class HelloWorld { }
  - HelloWorld라는 클래스를 선언
  - 클래스는 C# 프로그램을 구성하는 기본 단위
  - 클래스는 "데이터 + 메소드"로 이루어짐
  - C# 프로그램은 최소 하나 이상의 클래스로 이루어지며 수백,수천의 클래스로 구성되기도 함

```
05 class HelloWorld
06 {
07     // 프로그램 실행이 시작되는 곳
08     static void Main(string[] args)
09     {
10         Console.WriteLine("Hello, World!");
11         Console.WriteLine("Hello, {0}!", args[0]);
12     }
13 }
```

02. 첫 번째 프로그램 뜯어보기 (4/6)

- // 프로그램 실행이 시작되는 곳
  - //로 시작되는 코드는 주석(Comment)
  - 주석이란, 소스 코드 안에 기록하는 메모
    - 컴파일에는 아무 영향을 끼치지 않음
    - 컴퓨터가 아닌 사람이 참고하기 위한 기록
  - C#은 아래와 같이 두 가지 스타일의 주석을 지원

```
// 프로그램
// 실행이
// 시작되는 곳
static void Main(string[] args)
{
    Console.WriteLine("Hello, World!");
}
```

```
/* 프로그램
실행이
시작되는 곳 */
static void Main(string[] args)
{
    Console.WriteLine("Hello, World!");
}
```

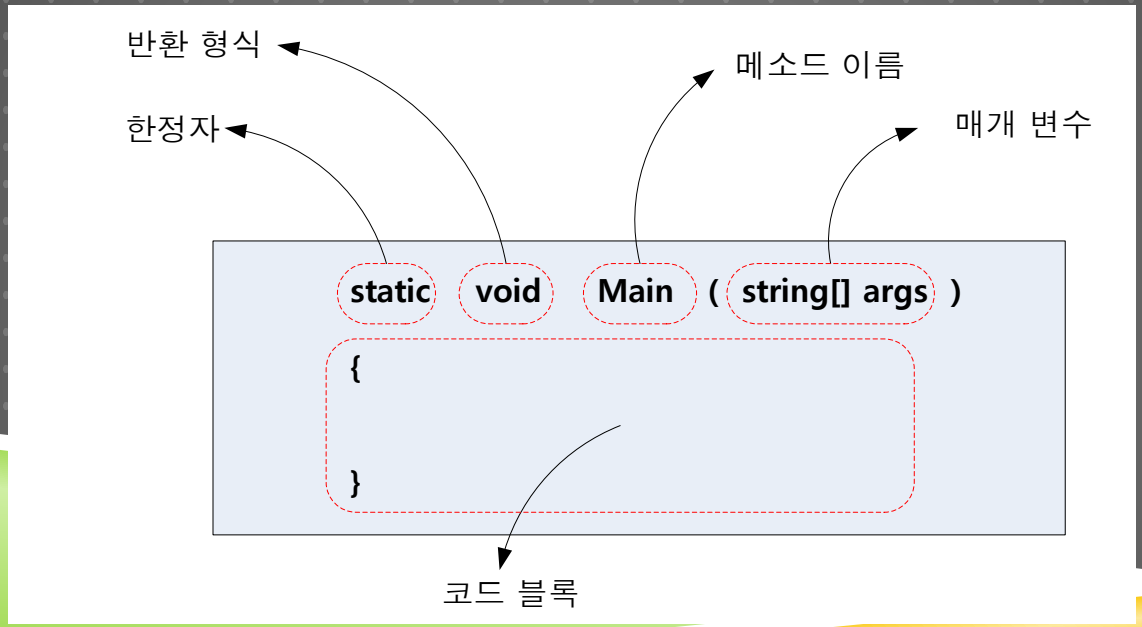
02. 첫 번째 프로그램 뜯어보기 (5/6)

- static void Main(string[] args) { } (1/2)
  - 프로그램의 진입점(Entry Point)으로써, 프로그램을 시작하면 실행되고, 이 메소드가 종료되면 프로그램도 역시 종료
  - 모든 프로그램은 반드시 Main이라는 이름을 가진 메소드를 가지고 있어야 함

```
05 class HelloWorld
06 {
07     // 프로그램 실행이 시작되는 곳
08     static void Main(string[] args)
09     {
10         Console.WriteLine("Hello, World!");
11         Console.WriteLine("Hello, {0}!", args[0]);
12     }
13 }
```

02. 첫 번째 프로그램 뜯어보기 (6/6)

- static void Main(string[] args) { } (2/2)
  - 한편, 메소드는 C 프로그래밍 언어에서는 함수(Function)라 불림
    - 함수는 입력을 받아 계산한 후, 출력함
  - 다음은 Main() 메소드를 이루는 각 구성요소의 명칭을 나타냄

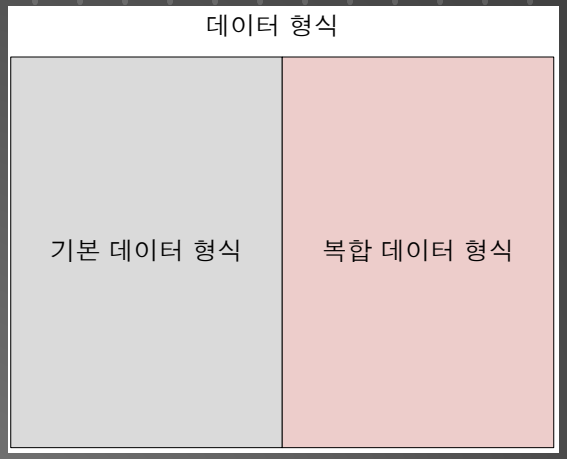


C# 프로그래밍의 이해

데이터 구조와 표현법

01. 데이터에도 종류가 있다

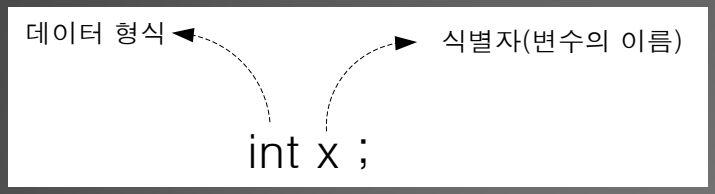
- C#의 데이터 형식 체계는 기본 데이터 형식(Primitive Data Type)과 복합 데이터 형식(Complex Data Type)
  - 기본 데이터 형식 : int, double, decimal …
  - 복합 데이터 형식 : string, Socket, Image …



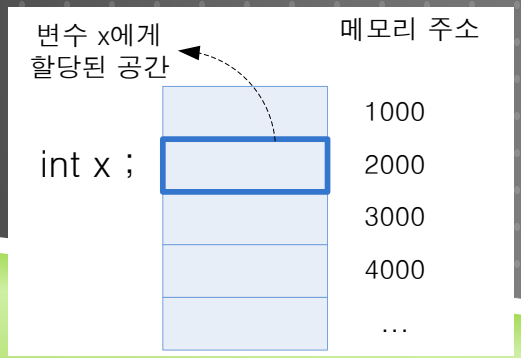
- 그리고 값 형식과 참조 형식으로 나뉨
  - 값 형식 : 변수에 데이터의 값을 담는 형식
  - 참조 형식 : 변수에 데이터의 위치를 담는 형식

02. 변수(VARIABLE) (1/3)

- 데이터를 담는 일정 크기의 공간(여기에서 "일정 크기"는 데이터 형식에 따라 결정)
- C#에서 변수는 다음의 꼴로 선언

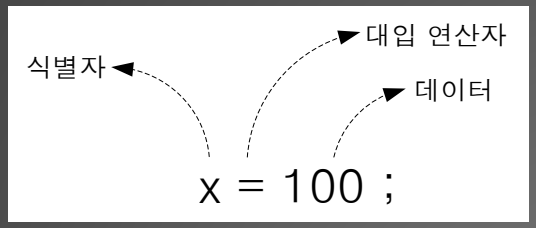


- 위와 같이 선언한 변수는 다음과 같이 메모리에 할당

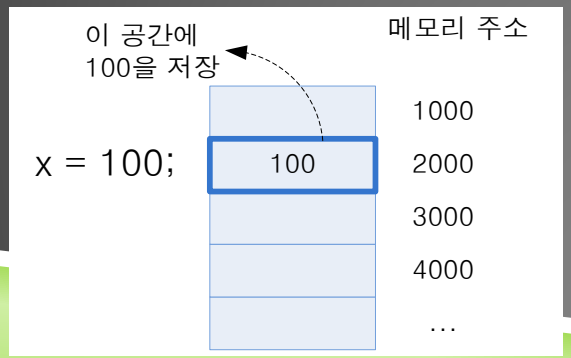


02. 변수(VARIABLE) (2/3)

- 선언된 변수에는 데이터할당이 가능해짐



- 이 코드가 실행되고 나면 x를 위해 할당된 메모리 공간에는 데이터 100이 기록됨



02. 변수(VARIABLE) (3/3)

- 초기화 : 변수에 최초의 데이터를 할당하는 것

```
int x;          // 선언과
x = 100;        // 데이터 할당을 별도로 할 수도 있지만
int x = 100; // 선언과 초기화를 한번에  할 수 있음
```

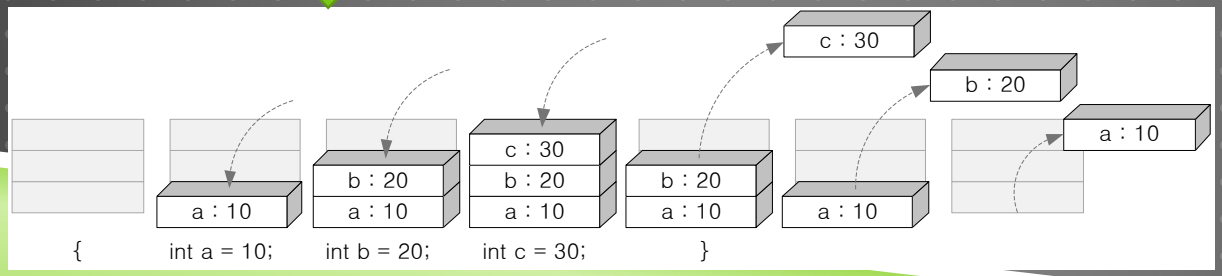
- 콤마(,)를 이용하여 여러 개의 변수 동시 선언 가능

```
int a, b, c;                        // 같은 형식의 변수들은 동시에 선언 가능
int x = 30, y = 40, z = 50;  // 선언과 초기화를 한번에 하는 것도 가능
```

03. 값 형식과 참조 형식 (1/2)

- 값 형식 : 변수가 값을 담는 데이터 형식
- 값 형식 변수는 스택에 할당되고 변수 생성 코드 범위를 벗어나면 해제됨.

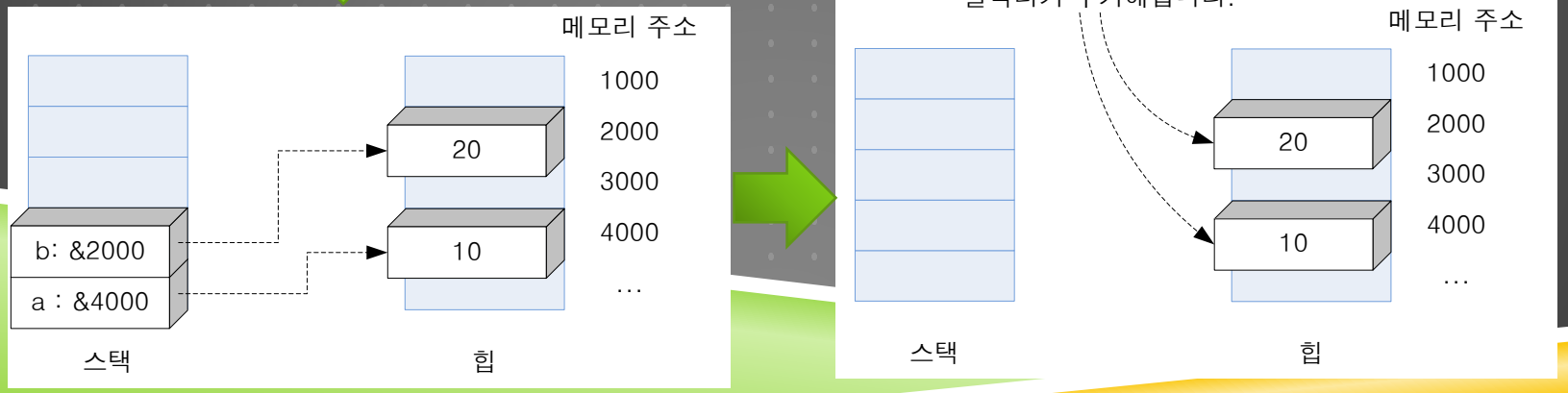
```
{ // 코드 블록 시작
    int a = 100;
    int b = 200;
    int c = 300;
} // 코드 블록 끝
```



03. 값 형식과 참조 형식 (2/2)

- 참조 형식 : 변수가 값 대신 값이 있는 곳의 위치(참조)를 담는 데이터 형식
- 참조 형식 변수는 힙에 할당되고 가비지 컬렉터에 의해 해제됨.

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



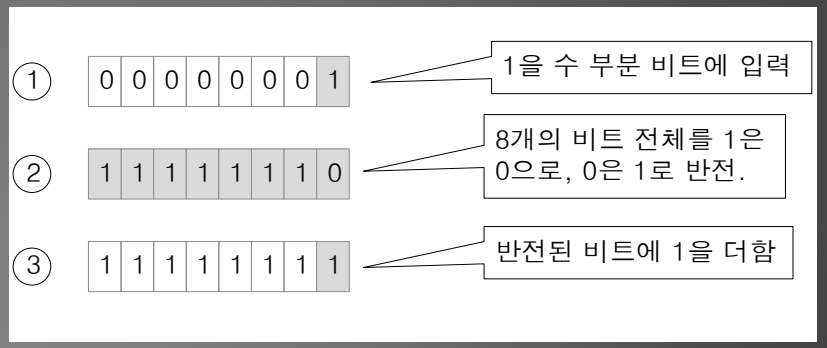
04. 기본 데이터 형식 (1/15)

- 정수 계열 형식(Integral Types) (1/3)

데이터 형식	설명	크기(Byte)	담을 수 있는 값의 범위
byte	부호 없는 정수	1 (8bit)	0 ~ 255
sbyte	signed byte 정수	1 (8bit)	-128 ~ 127
short	정수	2 (16bit)	-32,768 ~ 32,767
ushort	unsigned short 부호 없는 정수	2 (16bit)	0 ~ 65535
int	정수	4 (32bit)	-2,147,483,648 ~ 2,147,483,647
uint	unsigned int 부호 없는 정수	4 (32bit)	0 ~ 4294967295
long	정수	8 (64bit)	-922337203685477508 ~ 922337203685477507
ulong	unsigned long 부호 없는 정수	8 (64bit)	0 ~ 18446744073709551615
char	유니코드 문자	2 (16bit)	

04. 기본 데이터 형식 (2/15)

- 정수 계열 형식(Integral Types) (2/3)
  - **부호 있는 정수와 부호 없는 정수**
    - 부호 있는 정수 : sbyte, short, int, long …
    - 부호 없는 정수 : byte, ushort, uint, ulong …
  - C#에서는 "2의 보수법"을 이용하여 음수를 표현
    1) 먼저 수 부분 비트를 채운다.
    2) 전체 비트를 반전시킨다.
    3) 반전된 비트에 1을 더한다.



예) -1을 2의 보수법으로 표현

04. 기본 데이터 형식 (3/15)

- 정수 계열 형식(Integral Types) (3/3)
  - **데이터 오버플로우 (Data Overflow)**
    - 변수는 데이터를 담는 그릇
    - 그릇에 용량을 넘어서는 양의 물을 담으면 넘치는 것처럼,
      변수에도 데이터 형식의 크기를 넘어서는 값을 담게 되면 넘침
    - byte형 변수 a에 byte가 담을 수 있는 최대 값인 255를 넣어놓고
      a에 1을 더하면 a는 얼마가 될까…? ➔ 0
      - byte의 최대값은 255를 이진수로 바꾸면 1111 1111
      - 이진수 1111 1111에 1을 더하면 1 0000 0000
      - 하지만 byte는 1바이트, 즉 8개의 비트만 담을 수 있으므로 넘쳐흘러 왼쪽의 비트는 버리고 오른쪽 8개의 비트만 보관
      - 그래서 최대값을 가진 byte 형 변수가 오버플로우되면 0이 되는 것

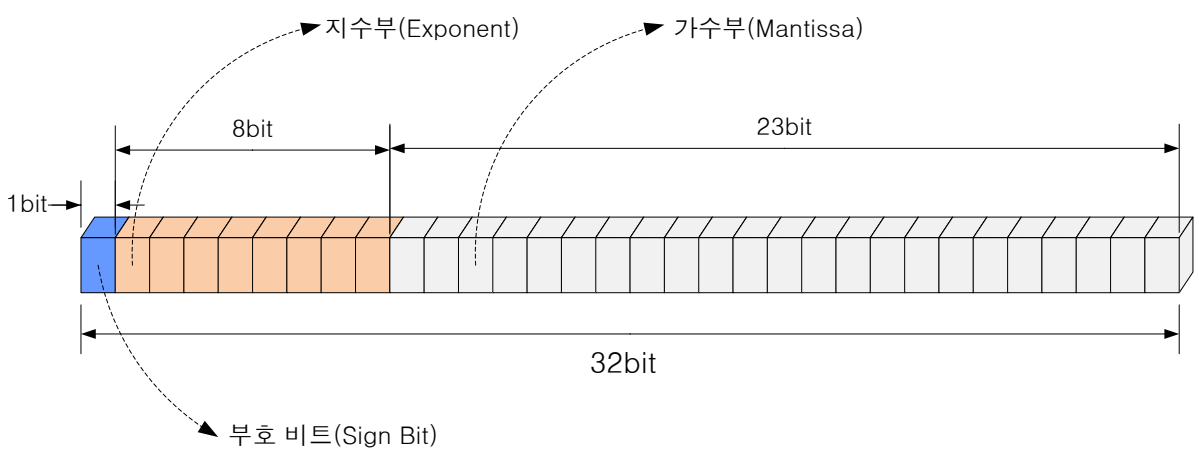
04. 기본 데이터 형식 (4/15)

- 부동 소수점 형식(Floating Point Types) (1/2)
  - "부동"이라는 말은 뜰 부(浮), 움직일 동(動), 즉 떠서 움직인다는 뜻
  - 부동 소수점이라는 이름은 소수점이 고정되어 있지 않고 움직이면서 수를 표현한다는 뜻에서 지어진 이름
    - 소수점을 이동시켜 수를 표현하면 고정시켰을 때보다 보다 제한된 비트를 이용해서 훨씬 넓은 범위의 값을 표현할 수 있음.

데이터 형식	설명	크기(byte)	범위
float	단일 정밀도 부동 소수점 형식 (7개의 자릿수만 다룰 수 있음.)	4 (32bit)	-3.402823e38 ~ 3.402823e38
double	복수 정밀도 부동 소수점 형식 (15~16개의 자릿수를 다룰 수 있음.)	8 (64bit)	-1.79769313486232e308 ~ 1.79769313486232e308

04. 기본 데이터 형식 (5/15)

- 부동 소수점 형식(Floating Point Types) (2/2)
  - IEEE754
    - 4바이트(32비트) 크기의 float 형식은 수를 표현할 때
      - 1비트를 부호 전용으로 사용
      - 가수부 23비트를 수를 표현하는 데 사용
      - 나머지 지수부 8비트를 소수점의 위치를 표현하는데 사용



04. 기본 데이터 형식 (6/15)

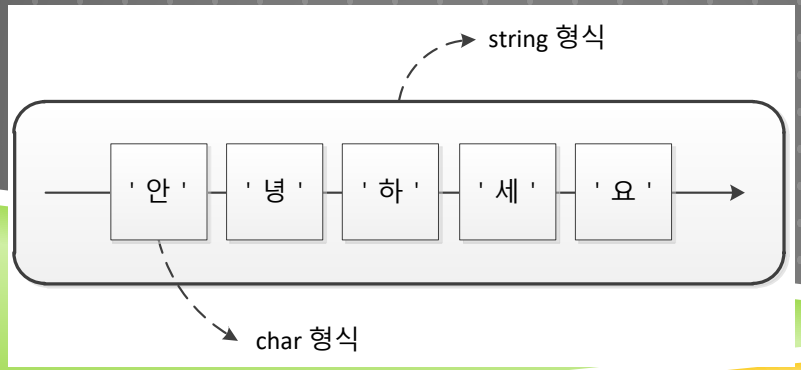
- 문자 형식과 문자열 형식
  - char 형식(문자 데이터)에 데이터를 담을 때는 작은 따옴표 '와 '로 감싸줌

    예) char a = '가';

    char b = 'a';
  - 문자열(**文字列**, string): 문자들이 연속해서 가지런히 놓여있는 줄

    예) string a = "안녕하세요?";

    string b = "박상현입니다.";



04. 기본 데이터 형식 (7/15)

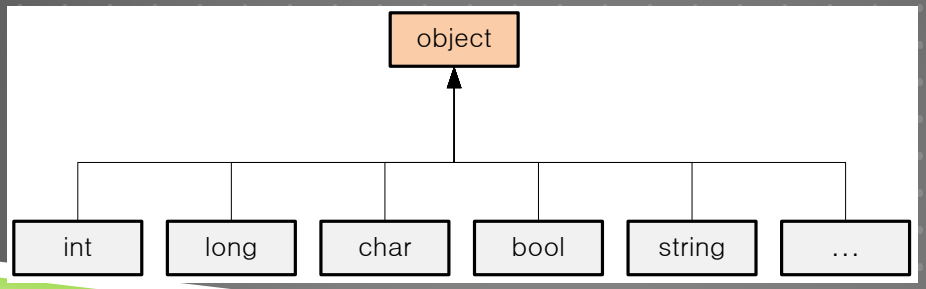
- 논리 형식
  - 논리 형식이 다루는 데이터는 참(true)과 거짓(false) 두가지 뿐

데이터 형식	설명	크기(byte)	범위
bool	논리 형식	1 (8bit)	true, false

  - 논리 형식 없는 언어(예:C)에서 거짓과 참을 0과 0이 아닌 수로 대신
    - 나중에 코드를 읽을 때 거짓을 의미하는지 또는 숫자 0을 의미하는지 혼란
    - C#은 이런 문제를 논리 형식을 별도로 도입함으로써 해결

04. 기본 데이터 형식 (8/15)

- object 형식 (1/3)
  - 어떤 형식이든 다룰 수 있는 데이터 형식
  - 참조 형식
  - 부모 클래스로부터 파생된 자식 클래스는 부모 클래스와 동일하게 취급하는 "상속"을 이용한 데이터 형식
    - C#의 모든 데이터 형식은 object 형식으로 파생받음
    - 즉, C#의 모든 데이터 형식은 object 형식으로 다룰 수 있음.



04. 기본 데이터 형식 (9/15)

- object 형식 (2/3)

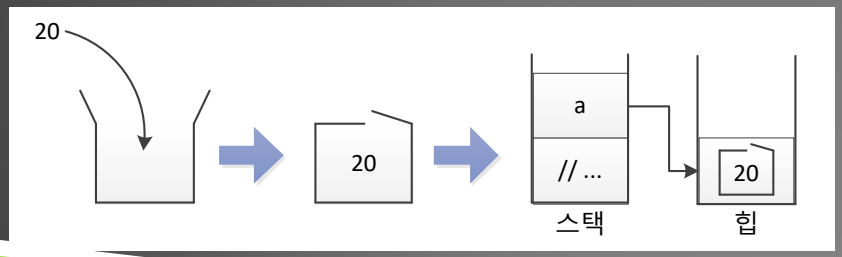
```
01  using System;
02
03  namespace Object
04  {
05      class Program
06      {
07          static void Main(string[] args)
08          {
09              object a = 123;
10              object b = 3.14159265358979323846264338327 9m;
11              object c = true;
12              object d = "안녕하세요.";
13
14              Console.WriteLine(a);
15              Console.WriteLine(b);
16              Console.WriteLine(c);
17              Console.WriteLine(d);
18          }
19      }
20  }
```

object 형식은 참조 형식이면서 값 형식 데이터와 참조 형식 데이터 모두를 담을 수 있다. 어떻게 이런 일이 가능한 걸까?

04. 기본 데이터 형식 (10/15)

- object 형식 (3/3)
  - Boxing과 Unboxing
    - Boxing : 값 형식 데이터를 상자에 담아 힙에 올려놓고, 그 힙의 위치를 object 형식 변수가 가리키도록 하는 것
    - Unboxing : 힙 안에 올라가 있는 상자를 풀어 값형식 데이터를 꺼내는 것

Boxing 예)
object a = 20;



Unboxing 예)
object a = 20;
int b = (int)a;