



C# -CHAPTER 1-

SOUL SEEK



목차

-
1. 기초이론
 2. HelloWorld
 3. 변수

기초 이론

1. 기초 이론

.NET

- 여러 기능을 지원하는 클래스 라이브러리를 제공
- 프레임 워크를 설치하면 다양한 플랫폼을 지원하는 동작을 하게된다.
- **C#**에 최적화 되어 있다.

CRL(Common Language Runtime)

- **Java JVM** 같은 가상 머신 기능을 한다.
- **.NET** 프레임워크와 함께 **OS**위에 설치된다.
- 네이티브 코드로 작성된 프로그램들은 운영체제가 직접 실행할 수 있다.
- **C#**은 **OS**가 바로 알아볼 수 없는 **IL**이라는 중간 언어로 작성되어 있어 **JIT** 과정이 필요하다.



JIT(Just In Time) 컴파일

- **IL**이라는 중간 언어로 작성된 실행파일을 만들어낸다. 사용자가 이 파일을 실행시키면 **CLR**이 중간 코드를 읽어 들어서 다시 **OS**가 이해할 수 있는 네이티브 코드로 컴파일한 후 실행하게 된다. 서로 다른 멀티 플랫폼을 지원하기 위한 과정이기 때문에 **C#**역시 이런 컨셉으로 만들어진 언어이기 때문이다. **C/C++**에 비해 한 단계 많은 과정을 거치기 때문에 컴파일이 느리다는 단점은 있지만 그것이 큰 의미를 가지는 건 아니다.

The image features a dark blue gradient background with faint, light blue concentric circles centered behind the text. In the four corners, there are decorative white line art elements resembling circuit traces or neural network connections, with small circles at various points.

HELLOWORLD

2. HELLOWORLD

```
//키워드 선언
using System;

namespace HelloWorld
{
    class HelloWorld
    {
        //CLR에 메모리 할당
        static void Main(string[] args)
        {
            //Hello World 출력
            Console.WriteLine("Hello World!!");

            //콘솔창 유지하려고 넣은 코드
            Console.ReadKey();
        }
    }
}
```

변수

3. 변수

- 전역변수를 지원하지 않는다.
- 초기화 필수 → 초기화 없이 쓰레기값을 가지게 되면 컴파일 에러 발생!
- 변수타입 : **Value Type, Reference Type**이 존재한다.
- **Reference Type : Heap**영역 → **string, object**
 - 힙 메모리에 할당된 데이터들은 언제 지나 살아 있다. 하지만 **C#**에서는 일정시간 사용하지 않으면 사용자가 실수로 해제하지 않은 것으로 간주하여 **GC**가 일어나면서 메모리에서 삭제한다.
- **Value Type : Stack**영역 → 숫자, 정수, 문자(문자열 아님), 부동소수, 논리형식
 - 스택 메모리에 순차적으로 쌓아놓고 코드블록이 끝나는 지점에서 모두 사라진다.

string

- 문자열 연산이 편리하다.
- **C++ string**과 비슷하지만 내부적으로 동작, 구성이 다르다. → 기능적으로는 같다.

object

- 모든 데이터 형식을 대변할 수 있다.
- 모든 데이터 형식은 **object**를 상속받고 있기 때문에 어떠한 값이 대입되어도 맞는 형식으로 적용해준다.
- **Reference Type**이기 때문에 힙메모리 영역에 저장된다.
- **object**로 실질적으로 선언된 실제 값들은 **heap**에 저장하고 **heap**에 저장된 값들의 주소 값들만 스택에 저장한다. 이런 과정에 의해서 박싱과 언박싱이 발생한다.

var 형식

- 지역변수로만 사용 가능하다.
- 컴파일러가 값이 담긴 형식을 찾아서 자동으로 맞는 형식으로 지정해준다.

3. 변수

박싱과 언박싱

- 힙 메모리 영역으로 메모리 할당하기 위해 힙메모리 형식으로 포장하는 것을 박싱 그 반대로 힙 메모리 영역에 있는 것을 스택메모리 영역으로 변환하여 사용하는 것을 언박싱이라고 한다.
- `object` 형식이 기존 데이터 형식의 값을 직접적으로 힙 메모리에 넣을 때 박싱을 하여 넣어준다.하지만 이것의 형을 우리는 모르지만 스택 영역에 할당되는 형식으로 변환 했을 때 박싱된 것을 다시 꺼내어 쓰기 때문에 언박싱이 일어납니다.

매개변수

- **ref**(참조매개변수), **out**(출력매개변수)

ref

- **C/C++**의 레퍼런스 매개변수와 같은 동작을 한다.
- 결과 값의 대입여부와 상관없이 동작한다.

out

- 포인터 매개 변수와 같은 동작을 한다.
- 결과값을 대입해주지 않는다면 에러가 난다. 이는 연산하지 않고 초기화 되지 않은 상태의 쓰레기 값을 그대로 전달해 주지 않기 위한 안전장치이다.
- **ref**보다 **out** 사용을 권장한다.

3. 변수

가변길이 매개 변수

- **Params** 키워드와 배열을 이용해서 선언.
- 오버로딩의 경우 매개변수의 타입이 서로 다를 경우에는 유용하다 하지만 매개변수의 타입의 변화가 없는데 그 수가 늘어나서 적용하고 싶은 경우가 있다 이때 가변길이 매개변수를 사용 할 수 있다.

선택적 매개 변수

- **C++**의 디폴트 매개변수 같은 기능을 하게 해줍니다.
- 이 선택적 매개변수는 매개변수를 다 선언하고 제일 뒤쪽에 와야하며 선택적 매개변수의 수는 정해져 있지 않습니다.