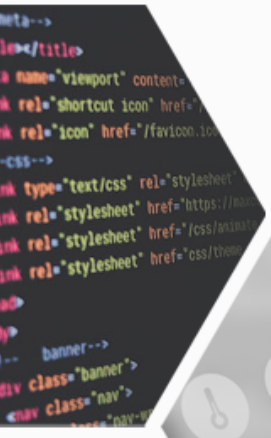
한국기술교육대학교
온라인평생교육원

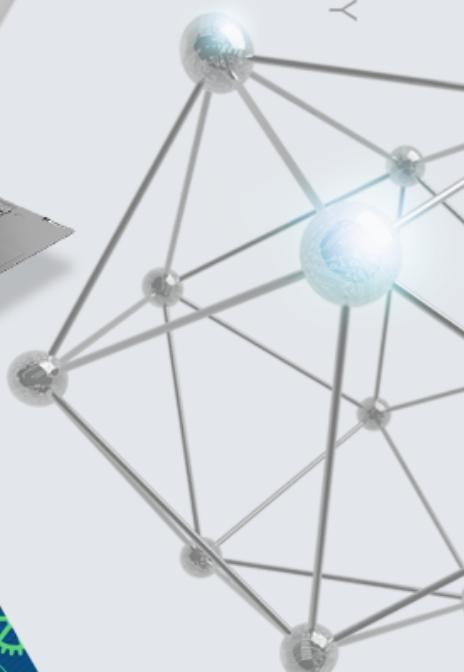
The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

TECHNOLOGY

C# 프로그래밍

C#의 개요 및 언어 구조





The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

C#의 개요 및 언어 구조

학/습/목/표

1. C# 언어의 개요와 특징을 이해하고 설명할 수 있다.
2. C# 언어 구조를 이해하고 설명할 수있다.

학/습/내/용

1. C#의 개요
2. C#의 언어 구조

1. C#의 개요

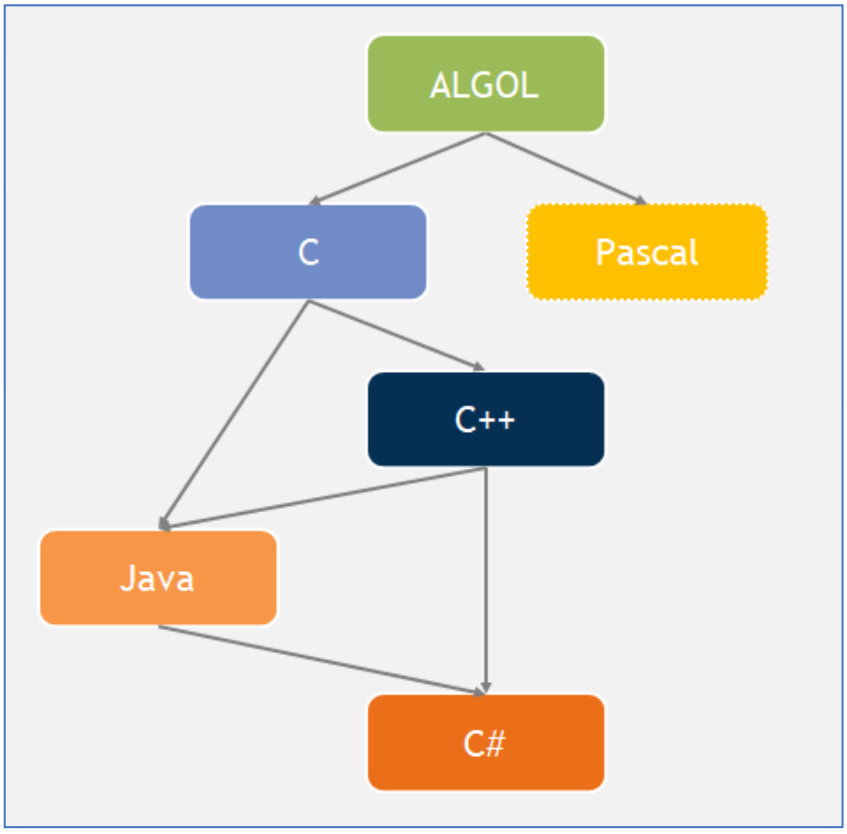
1) 소개

(1) C 계열의 언어

- C++, 자바로부터 영향을 받았음
- 연산자와 문장, 객체지향 속성, 예외처리와 스레드

(2) C# 언어의 계통도

- 현대적인 의미를 갖는 범용 프로그래밍 언어의 뿌리는 ALGOL로 간주
- C 언어 계열과 파스칼 언어 계열로 분파되어 각각 독자적으로 발전



[C# 언어의 계통도]

1. C#의 개요

2) 개발 환경

(1) C# 프로그램을 작성하고 실행하기 위한 개발 환경

① 콘솔/윈폼 애플리케이션
- 문자기반 명령어 프롬프트 환경에서 실행
- 키보드를 통해 입력, 화면에 문자로 출력

② C# 개발 환경
- SDK를 이용 : 편집기, 컴파일러, 실행엔진, 클래스 라이브러리
- 통합개발 환경(IDE)

3) C# 애플리케이션

(1)일반적인 응용 프로그램 작성 : HelloWorld.cs

예제 프로그램

```
using System;
class HelloWorld {
  public static void Main() {
    Console.WriteLine("Hello World!");
  }
}
```

네임스페이스

출력 메소드

실행 결과

```
Hello World!
```

실행 과정

```
C:\temp>csc HelloWorld.cs
C:\temp>HelloWorld
Hello World!
```

1. C#의 개요

4) 실행 과정

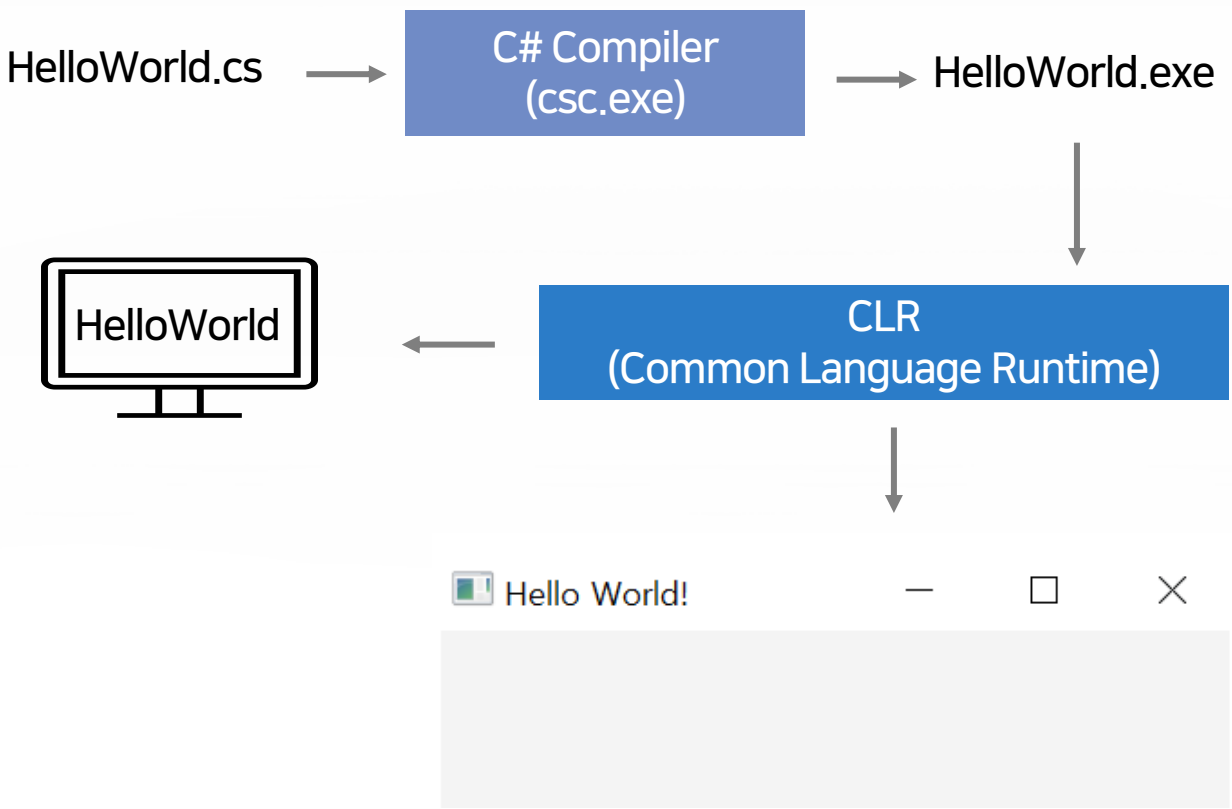
(1) 컴파일 과정

- csc : C# Compiler

(2) 실행 시스템

- CLR – Common Language Runtime

(3) 실행 과정



1. C#의 개요

5) 기본 특징

(1) 클래스

- 클래스와 객체
- 객체 자료형(클래스)
- 클래스 멤버
- 개체의 속성과 행위 결정

1. C#의 개요

5) 기본 특징

(1) 클래스

- 개체의 속성과 행위 결정

```
class CoffeeMaker {
  public bool  onState;
  public void StartCoffeeMaker() {
    if (onState == true)
      Console.WriteLine("The CoffeMaker is already on");
    else
      onState = true;
    Console.WriteLine("The CoffeMaker is now on");
  }
}
```

속성

행위

1. C#의 개요

5) 기본 특징

(2) 프로퍼티

① 프로퍼티의 개념

- 클래스의 private 필드를 형식적으로 다루는 일종의 메소드
- 값을 지정하는 셋-접근자와 값을 참조하는 겟-접근자로 구성

셋-접근자
배정문의 왼쪽에서 사용되면 호출됨

겟-접근자
배정문의 오른쪽에서 사용되면 호출됨

② 프로퍼티의 동작

```
[property-modifiers] returnType PropertyName {
  get {
    // get-accessor body
  }
  set {
    // set-accessor body
  }
}
```

1. C#의 개요

5) 기본 특징

(3) 연산자 중복

① 연산자 중복의 의미

- 시스템에서 제공한 연산자를 재정의하는 것
- 클래스만을 위한 연산자로써 자료 추상화가 가능
  (시스템에서 제공한 연산자처럼 사용 가능)
- 문법적인 규칙은 변경 불가(연산 순위나 결합 법칙 등)

② 연산자 중복 정의 형태

```
public static [extern] returnType operator op
(parameter1 [, parameter2]) {
  // ... operator overloading body ...
}
```

(4) 델리게이트

① 델리게이트(Delegate)는 메소드 참조 기법

- 객체지향적 특징이 반영된 메소드 포인터
- 클래스만을 위한 연산자로써 자료 추상화가 가능
  (시스템에서 제공한 연산자처럼 사용 가능)
- 문법적인 규칙은 변경 불가(연산 순위나 결합 법칙 등)

② 이벤트와 스레드를 처리하기 위한 방법론

선언형태

```
[modifiers] delegate returnType
DelegateName(parameterList);
```

1. C#의 개요

5) 기본 특징

(5) 이벤트

① 이벤트의 정의

- 사용자 행동에 의해 발생하는 사건
- 어떤 사건이 발생한 것을 알리기 위해 보내는 메시지로 간주
- C#에서는 델리케이트를 이용하여 이벤트를 처리

② 이벤트 정의 형태

선언형태 `[event-modifier] event DelegateType EventName;`

③ 이벤트 주도 프로그래밍

- 이벤트와 이벤트 처리기를 통하여 객체에 발생한 사건을 다른
- 객체에 통지하고 그에 대한 행위를 처리하도록 시키는 구조를 가짐
- 각 이벤트에 따른 작업을 독립적으로 기술
- 프로그램의 구조가 체계적/구조적이며 복잡도를 줄일 수 있음

(6) 스레드(Thread)

① 기본 개념

- 순차 프로그램과 유사하게 시작, 실행, 종료의 순서를 가짐
- 실행되는 동안에 한 시점에서 단일 실행점을 가짐
- 프로그램 내에서만 실행 가능

② 멀티스레드 시스템

- 스레드가 하나의 프로그램 내에 여러 개 존재
- 응용 프로그램의 병행 처리를 위해 스레드 개념을 지원
- 스레드를 생성하고 실행시키고 제어하는 방법을 제공
- 델리케이트를 이용하여 처리

1. C#의 개요

5) 기본 특징

(7) 제네릭(Generic)

① 제네릭의 의미
- 자료형을 매개변수로 가질 수 있는 개념
- C++의 템플릿과 유사한 개념

② 제네릭 단위
- 클래스, 구조체, 인터페이스, 메소드

제네릭 클래스

- 범용 클래스 또는 포괄 클래스
- 형 매개변수(Type Parameter)
- <> 안에 기술

예

```
class Stack<StackType> {
  private StackType[] stack = new StackType[100];
  // ...
  public void Push(StackType element) { /* ... */ }
  public StackType Pop() { /* ... */ }
}

Stack<int> stk1 = new Stack<int>();       //정수형 스택
Stack<double> stk2 = new Stack<double>();//실수형 스택
```

2. C#의 언어 구조

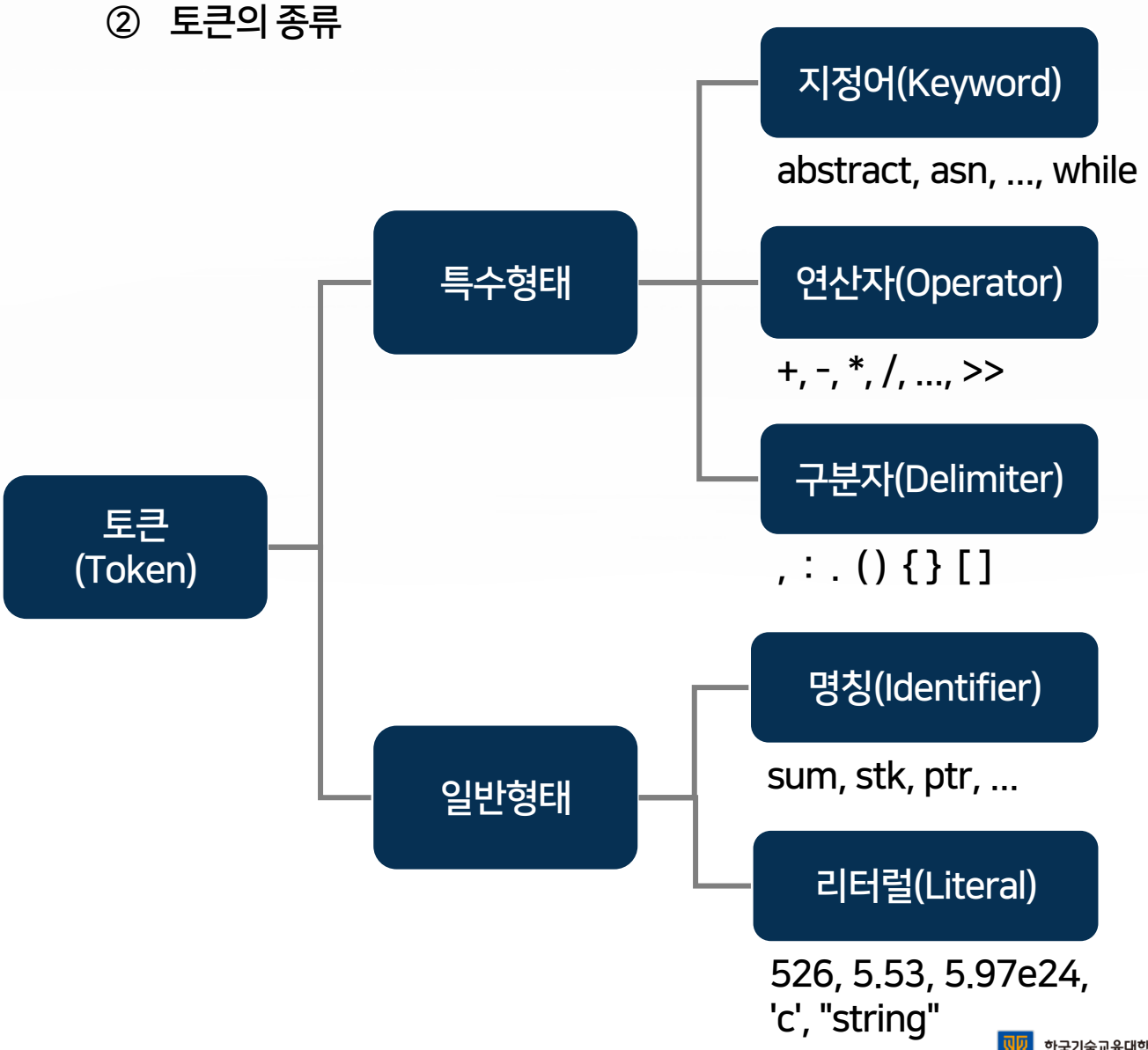
1) 어휘 구조

(1) 개요

① 어휘

- 프로그램을 구성하고 있는 기본 소자
- 토큰(Token)이라 불림
- 문법적으로 의미 있는 최소 단위

② 토큰의 종류



2. C#의 언어 구조

1) 어휘 구조

(1) 개요

③ 지정어
- 프로그래밍 언어 설계시에 그 기능과 용도가 이미 정의되어 있는 단어
- 사용자가 임의로 결정하는 변수 이름이나 메소드 이름 등으로 사용할 수 없음

④ C# 지정어(77개)
- C# language specification(ECMA TC39/TG2)

2. C#의 언어 구조

1) 어휘 구조

(2) 명칭

① 명칭의 의미
- 자료의 항목(변수, 상수, 배열, 클래스, 메소드, 레이블)을 식별하기 위하여 붙이는 이름

② 명칭의 형태
- 문자로 시작
- 길이에는 제한이 없음
- 대소문자 구분
- @기호를 붙이면 명칭으로 사용할 수 있음

③ 옳은 명칭들
- `sum, sum1, money_sum, moneySum, @int, 변수`

④ 틀린 명칭들
- `1sum, sum!, $sum, #sum, Money Sum, virtual`

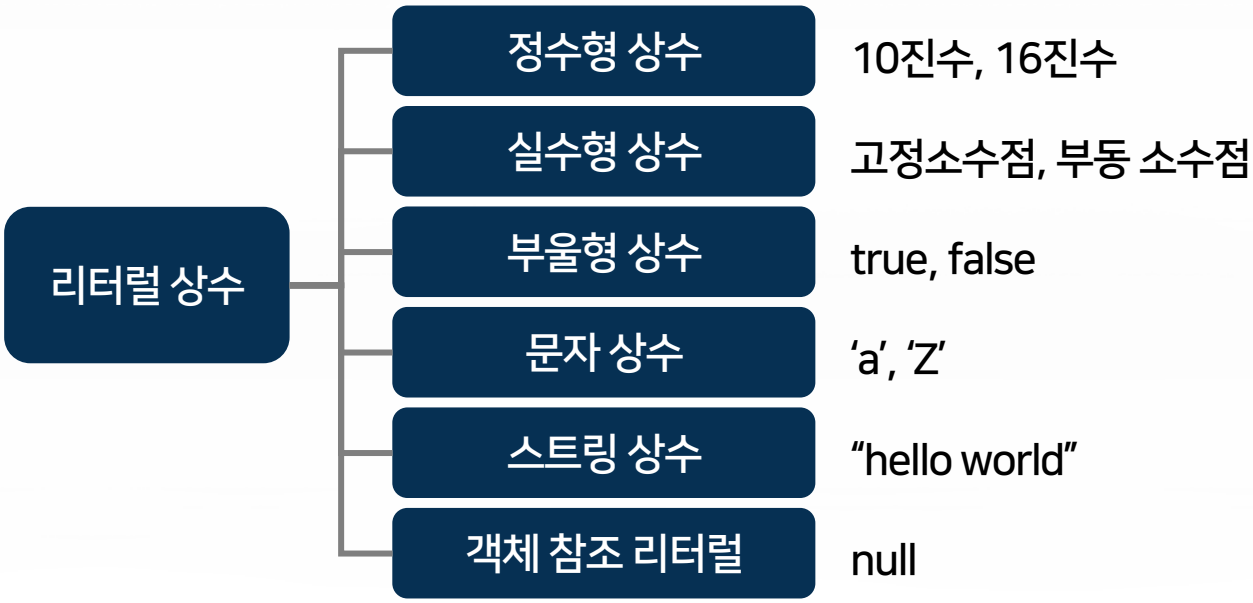
2. C#의 언어 구조

1) 어휘 구조

(2) 리터럴

① 리터럴의 의미
- 자신의 표기법이 곧 자신의 값이 되는 상수
- "12"라고 표기하면 값이 12인 정수를 나타내는 경우

② 리터럴의 종류



③ 객체참조 리터럴(Object Reference)
- 널(null) : 아무 객체도 가리키지 않는 상태
- 부적당하거나 객체를 생성할 수 없는 경우 초기화에 사용

2. C#의 언어 구조

1) 어휘 구조

(3) 주석

① 주석의 의미

- 프로그램을 설명하기 위한 문장
  - 프로그램의 실행에는 무관
  - 프로그램 유지보수에 중요

② 주석의 종류

// comment

- //부터 새로운 줄 전까지 주석으로 간주
- 예) int size = 100; //size는 100으로 초기화

/* comment */

- /*와 다음 */ 사이의 모든 문자들은 주석으로 간주
- 주석문 안에서 또 다른 주석이 포함될 수 없음
- 예) /* C# 언어에서는 여러 줄의 주석을 위해
- 지금 사용하고 있는 주석의 형태를 지원하고 있음

/// comment

- /// 다음의 문자들은 주석으로 간주
- C# 프로그램에 대한 웹 보고서를 작성하는데 사용하는 방법
- XML 태그를 이용하여 기술
- 컴파일 시에 /doc 옵션을 사용하여 XML 문서 생성

2. C#의 언어 구조

1) 어휘 구조

(3) 주석

③ 주석의 종류 예

/// comment 에서 /doc 옵션을 사용한 예

csc CommentApp.cs /doc:CommentApp.xml

XML 문서

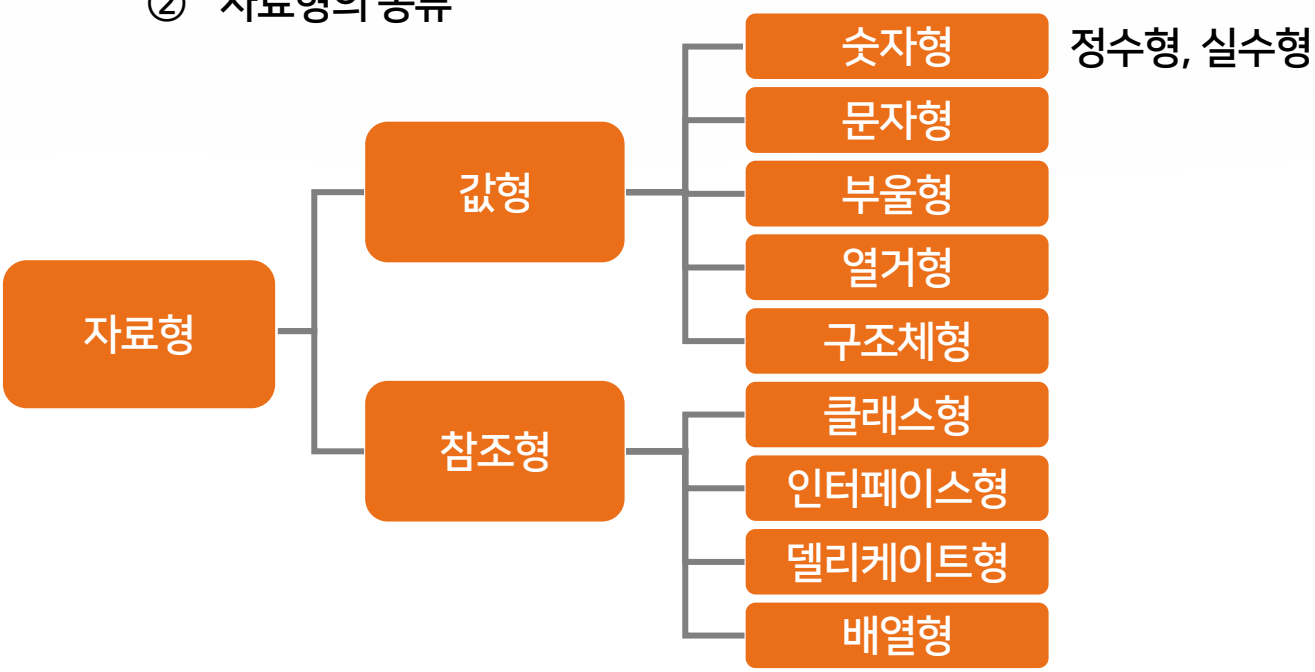
T:CommentApp - T is Type
M:CommentApp.Main - M is Method

2) 자료형

① 자료형의 의미

- 자료 객체가 갖는 형으로 구조 및 개념, 값의 범위, 연산 등을 정의

② 자료형의 종류



2. C#의 언어 구조

2) 자료형

(1) 값형

① 숫자형
- 값을 표현하는 방법과 연산하는 방식에 따라 정수형과 실수형으로 구분됨

② 문자형과 부울형
- 문자형 : 16비트 유니코드를 사용
- 부울형 : true, false, 숫자값을 가질 수 없음, 다른 자료형으로 변환 불가

③ 열거형
- 서로 관련 있는 상수들의 모음을 심볼릭한 명칭의 집합으로 정의한 것
- 기호 상수 : 집합의 원소로 기술된 명칭
- 순서값 : 집합에 명시된 순서에 따라 0부터 부여된 값

④ 구조체형
- 클래스처럼 고유의 필드, 메서드, 생성자를 가지나 참조 형식이 아닌 형식
- 기본 생성자 선언 불가

2. C#의 언어 구조

2) 자료형

(2) 참조형

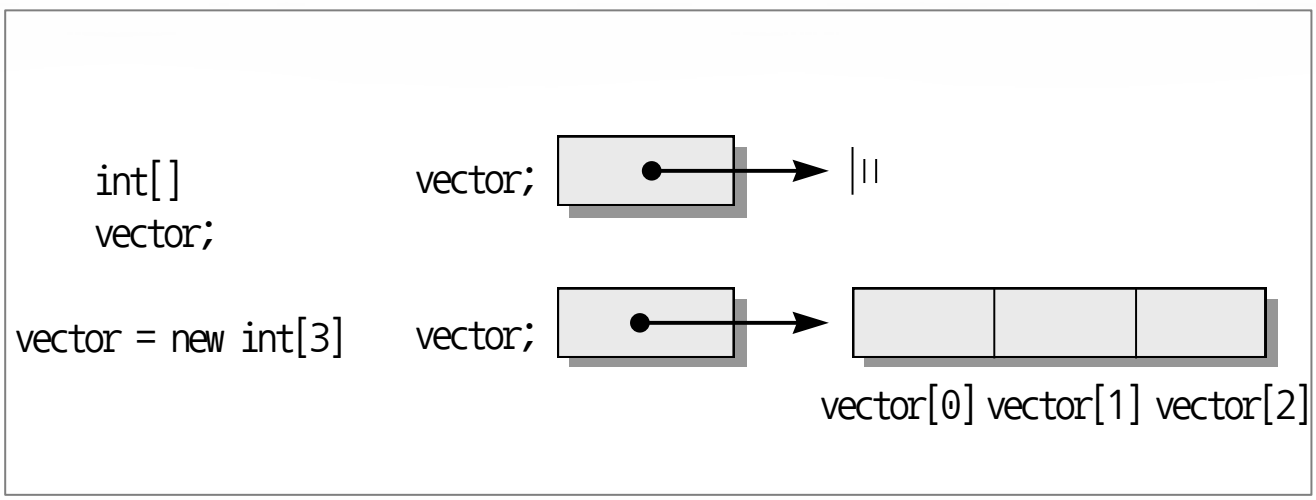
① 배열형

- 같은 형의 여러 개의 값을 저장하는데 사용하는 자료형
- 순서가 있는 원소들의 모임
- 배열을 하나의 객체로 취급하여 참조형으로 다룸

② 배열형의 예

```
int[]       vector;      // 1차원 배열
short[,]    matrix;      // 2차원 배열
object[]    myArray;
int[]       initArray = {0, 1, 2, 3, 4, 5};
// 선언과 함께 초기값 부여
```

```
vector  = new int[100];
matrix  = new short[10,100];
myArray = new Point[3];
```



2. C#의 언어 구조

2) 자료형

(2) 참조형

③ 스트링형의 의미
- 문자열을 표현하기 위해 사용하는 자료형
- System.String 클래스형과 동일한 자료형

④ StringBuilder 클래스
- 효율적으로 스트링을 다루기 위한 클래스
- 객체에 저장된 내용을 임의로 변경 가능
- 스트링 중간에 삽입, 추가시키는 다양한 메소드 제공
- 같은 형의 여러 개의 값을 저장하는데 사용하는 자료형
- 순서가 있는 원소들의 모임
- 배열을 하나의 객체로 취급하여 참조형으로 다룸

3) 연산자

① 식
- 문장에서 값을 계산하는데 사용

② 연산자
- 식의 의미를 결정
- 피연산자가 어떻게 계산될지를 나타내는 기호

2. C#의 언어 구조

3) 연산자

③ 연산자의 종류

C# 언어의 연산자

배정 연산자	= += -= *= /= %= &= |= ^= <<= >>=
산술 연산자	+ - * / % 단항+ 단항-
관계 연산자	> >= < <= == !=
캐스트 연산자	(자료형)
증감 연산자	전위++ 전위– 후위++ 후위--
형 검사 연산자	is as
논리 연산자	&& || !
배열 연산자	[]
메소드 연산자	()
비트 연산자	& | ^ ~ << >>
멤버 접근 연산자	.
조건 연산자	? :
지정어 연산자	new typeof checked unchecked

2. C#의 언어 구조

4) 형변환

(1) 연산할 때 모든 피연산자는 자료형이 서로 일치해야 함

(2) 형 변환은 피연산자의 자료형이 서로 다를 때 일정한 규칙에 따라 피연산자의 자료형을 일치시키는 것을 의미함

(3) 묵시적 형 변환
- 컴파일러에 의해 자동적으로 수행되는 형 변환
- 작은 크기 자료형 ➡ 큰 크기 자료형

(4) 명시적 형 변환
- 프로그래머가 캐스트 연산자를 사용하여 수행하는 형 변환
- 형태 : (자료형) 식
- 큰 크기 자료형에서 작은 크기 자료형으로 변환 시 정밀도 상실

예제

```
using System;
class LosePrecisionApp {
  public static void Main() {
    int big = 1234567890;
    float approx;
    approx = (float)big;
    Console.WriteLine("difference = " + (big - (int)approx));
  }
}
```

실행 결과

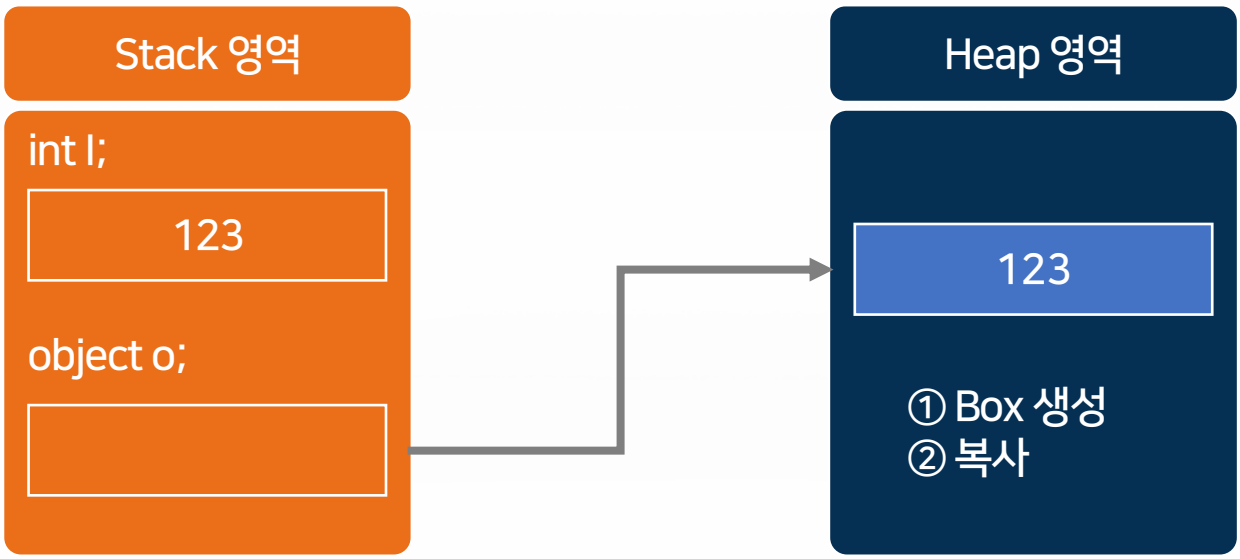
```
difference = -46
```

2. C#의 언어 구조

5) 박싱과 언박싱

① 박싱(Boxing)
- 값형의 데이터를 참조형으로 변환하는 것
- 컴파일러에 의해 묵시적으로 행해짐

② 박싱 과정



③ 언박싱(Unboxing)
- 참조형의 데이터를 값형으로 변환하는 것
- 반드시 캐스팅을 통하여 명시적으로 행해짐
- 반드시 박싱될 때 형으로 언박싱을 해주어야 함

2. C#의 언어 구조

5) 박싱과 언박싱

④ 예제 : BoxingUnboxingApp.cs

예제

```
using System;
class BoxingUnboxingApp {
  public static void Main() {
    int foo = 526;
    object bar = foo;            // foo is boxed to bar.
    Console.WriteLine(bar);
    try {
      double d = (short)bar;
      Console.WriteLine(d);
    } catch (InvalidCastException e) {
      Console.WriteLine(e + "Error");
    }
  }
}
```

실행 결과

```
526
System.InvalidCastException:
at BoxingUnboxingApp.Main() Error
```

1. C#의 개요

- C#은 마이크로소프트사의 랜더스 헬스버그에 의해 고안된 언어로서 C언어 계열에 속하는 범용 프로그래밍 언어임
- C# 프로그램을 작성하고 실행하기 위한 개발 환경은 Visual Studio와 같은 통합 개발 환경을 사용하는 방법과 .NET 프레임워크에서 지원하는 개발도구 (SDK)를 직접 이용하는 방법이 있음
- C# 언어의 주요 특징은 클래스와 프로퍼티, 연산자 중복, 델리게이트, 이벤트, 제네릭, 스레드 등이 있음

2. C#의 언어 구조

- 프로그래밍 언어의 어휘란 프로그램을 구성하고 있는 문법적으로 의미 있는 최소의 단위를 말하며 보통 이것을 토큰이라 부름
- 지정어란 프로그래밍 언어 설계 시에 그 기능과 용도가 이미 정의되어 있는 단어임
- 명칭은 자료의 항목 (변수, 상수, 배열, 클래스, 메소드, 레이블)을 식별하기 위하여 붙이는 이름임
- 주석은 프로그램이 무엇을 하는 가를 설명하기 위하여 소스 프로그램 내에 기술하는 문장임
- 자료형이란 자료 객체가 갖는 형으로 자료형의 실질적인 구조 및 개념, 그 자료형이 가질 수 있는 값 그리고 그 자료형에서 행할 수 있는 연산 등을 정의함
- 연산자는 식의 의미를 결정함
- 형 변환은 피연산자의 자료형이 서로 다를 때 일정한 규칙에 따라 피연산자의 자료형을 일치시키는 것을 의미함