4강. C# 기본 문법

프로그래밍 강사 : 이태성

목차

- 연산자
- 제어문
- 반복문
- 점프문
- 예외처리문

01 연산자

단항 연산자

- +, -, !, ~, ++, -- 등
- !은 bool형에만 사용

코드 보기

산술 연산자

- *, /, *, %, -, +
- string에서 +는 문자열 연결

  정수/부동 + “문자열” = “문자열”

  코드 보기

시프트(shift) 연산자와 관계 연산자

- <<, >>, >=, <=, >, >, ==, !=
- 관계 연산자의 결과는 ?
  true, false

is 연산자

- 형식 호환을 조사하는 연산자
- 형식

```
'변수' is '클래스형 or 데이터형'
```

결과는 true, false A is B

- 박싱/언박싱 변환, 참조 변환에서 사용

코드 보기

as 연산자

- 역할
  - 형변환과 변환 조사
  - 캐스트 연산자의 역할과
    불변환은 null 리턴
- 참조, 박싱, 언박싱, null형에 사용
- 형식

  결과형 = 참조형, 언박싱, 박싱 as 변환형

코드 보기

비트 연산자와 논리 연산자

- &, ^, |, &&, ||, ?:

null 병합 연산자

- ?? (null 조사)

C = A ?? B

A가 null이 아니면 A를 C에 대입
A가 null이면 B를 C에 대입

코드 보기

02 제어문

선택문

- if ~ else

코드 보기

- switch, case
  - 정수, 문자상수, *문자열*
  - 모든 case와 default에는 break가 반드시 있어야 함

코드 보기

반복문

- for
  for(;;) ➔ 무한반복

- while, do~ while
  while(true)

반복문

- foreach
  처음부터 끝까지 순차적을 값을
  반복하여 읽는 역할 ➔ 읽기 전용

  foreach(데이터형 *변수* in 배열명(컬렉션명))
  {

  }

  코드 보기

03 점프문

점프문

- goto, continue, return, break

04 예외 처리문

예외 처리

- 예외란?
  런타임 시에 발생할 수 있는 오류
- 예외 처리 방법
  if ~ else
  try ~ catch 문 사용

try ~ catch

- try
  {
  // 예외가 발생할 수 있는 코드
  }catch(예외처리객체 e)
  {
  // 예외 처리
  }

- System.Exception 파생 객체만 사용

try ~ catch

- System.Exception 파생 객체
  OverFlowException,
  FormatException,
  DivideByZeroException,
  FileNotFoundException,

- IndexOutOfRangeException

코드 보기

try ~ catch

- try 문 안에서 초기화한 변수를 try문 밖에서 사용할 수 없다

코드 보기

try ~ finally

- finally

  예외 발생과 상관없이 항상 실행되는 구문

- 예외적인 상황이 발생했을 때 finally 처리

  코드 보기

- 예외상황이 발생하지 않았을 때 finally 처리

  코드 보기

다중 예외 처리

- 형식

```
try {
        .....
} catch (OverFlowException e)
{    .....
} catch( FormatException e)
{     .....
 } catch( DivideByZeroException e)
{     .....
}
```

throw

- throw (던지다)
  예외 상황을 임의로 발생시키는 역할
- System.Exception 파생된 객체만 사용
- try문과 그 외에서 사용가능

코드 보기

정리

정리

- 대부분의 연산자는 C, C++언어와 같음
- C#에서 새롭게 등장한 연산자
  is 연산자, as 연산자, null 병합 연산자(??)
- for, while, do ~ while
  foreach
- 예외처리문
  try~ catch~finally
  throw