19.12.15 일 C#공부

문자 출력: Console.Write();

문자열 출력: Console.WriteLine();

3.1 데이터에도 종류가 있다.

3.2 변수?!

코드관점-> 값을 대입시켜 변화시킬수 있는 요소

\*메모리 관점 -> 데이터를 담는 일정 크리(데이터 형식)의 공간

\*변수를 “선언”한다. (선언대상)->컴파일러

Int x ;

변수x에게 메모리공간을 할당한다.

\*선언과 동시에 할당시킨다.

\*다수의 변수 동시 선언과 할당

3.3 값 형식과 참조 형식

\*값 형식: 변수에서 값을 담는 형식

\*참조 형식:값이 있는 곳의 위치를 담는 형식

값형식-스택(stack)형식: 책 더미와 같은 구조로 생긴 메모리.

3.3.2 힙과 참조형식

\*메모리 제거 메커니즘 필요: ‘가비지 컬렉터’를 이용해서 쓰지않는 것 제거

\*힙을 사용하는 이유: 코드 블록이 끝나도 데이터 유지

\*참조 형식의 변수

힙: ‘데이터’를 저장

스택: -힙의 ‘메모리 주소’를 스택구조로 저장함.

* 스택 구조는 사용후 바로 삭제

3.4 기본 데이터 형식

- 세부적으로 15가지

- 참조형식 : 문자열 형식, 오브젝트 형식

- 값 형식: 숫자 형식, 논리 형식

3.4.1 숫자 데이터 형식

-가장 많이 다루는 데이터 형식이 숫자

- 사실 텍스트 데이터도 숫자임. Ex) ascii code

-12가지 숫자 데이터 형식 ( 정수 계열, 부동소수 계열, 소수계열)

3.4.2 정수 계열 형식

3.4.3 정수 형식 예제 프로그램

\* 9가지의 정수 계열 데이터 형식 처리

크기와 데이터 범위에 따라서, 효율적인 메모리 사용.

Byte, sbyte, short, ushort, int, uint, long, ulong, char(=총 9가지)

3.4.4 2진수,10진수,16진수 리터럴

-2진수 리터럴 접두사 0b

- 16진수 리터럴 접두사 0x

- 10진수 리터럴 접두사는 x 필요없음

3.4.5 부호 있는 정수와 부호 없는 정수

- 부호 있는 정수: 양(+) 과 음(-), 0

(sbyte, short, int, long), 1개의 부호비트

-부호 없는 정수

3.4.6 데이터가 넘쳐 흘러요

\*오버플로우 : 데이터 형식의 최대값을 넘어가는 데이터를 저장할 때 발생

\*언더플로우 : 데이터 형식의 최소값을 넘어가는 데이터를 저장할 때

3.4.8 부동 소수점 형식

\* 소수점이 고정되어 있지 않고 움직이면서 수를 표현한다.: 제한된 비트를 이용해 넓은 범위의 값을 표현한다.

\*정수 형식을 대체하지 못하는 이유: 소수점 표현과 부호 표현에 일부 비트를 사용함으로 정수 계열 형식과 같은 크기를 사용하지 못함.

\*산술 연산 과정이 정수 계열 형식보다 복잡해 느림

\*float, double (double형식이 더 높은 정밀도를 가짐)

3.4.9 Decimal형식

\*실수를 다루는 데이터 형식

\*Decimal형식의 크기와 범위(29자리 데이터를 표현할 수 있음. 16바이트 128비트

3.4.10 문자 형식과 문자열 형식

\*문자형식 char ‘’(작은따옴표)로 문자를 감싼다. Char a=’r’;

\*문자열 형식 string “”(큰따옴표)로 감싼다. String b=”안녕하세요”;

3.4.11 논리 형식

\* 참(True)과 거짓(False)을 다루는 데이터형식

\* Bool : 논리 형식 : 1바이트(8비트) : true, false

\* c언어는 0과 0이아닌 값으로 참과 거짓을 표현했음.

3.4.12 Object 형식

\* 어떤 물건이든지 다룰 수 있는 데이터 형식

- 상속의 효과

- 참조 형식-> 힙에 데이터 할당

\*C#의 특별 조치

- Object는 모든 데이터 형식의 조상

Ex) object a=123;

Objcet b=3.142145234231;

Object c = true;

Object d = “안녕하세요”;

3.4.13 박싱과 언박싱

\* 박싱(Boxing)

- 값 형식의 데이터를 Object형식에 담는다면? -> 스택? 힙?

\* 언박싱(Unboxing)

- object 했던 데이터를 언박싱해서 값형식으로 바꾸는..?? 그런….

3.4.14 데이터 형식 바꾸기

\*변환: 변수를 다른 데이터 형식으로 바꾸는거

3.4.15 크기가 서로 다른 정수 형식 사이의 변환

3.4.16 크기가 서로 다른 부동 소수점 형식 사이의 변환

\* 부동 소수점 형식의 특성상 오버플로우 존재하지 않음

\*정밀성에 손상.

- float과 double은 소수를 이진수로 메모리에 보관하는데 다른형식으로 변환하면 즉

10진수 복원 -> 이진수로 변환 -> 이진수로 표현하는 소수는 완전하지 않음. Ex) 1/3

3.4.17 부호있는 정수 형식과 부호 없는 정수 형식 사이의 변환

\* 최저 값 보다 작은 값을 저장해 언더플로우 발생 가능성 有

3.4.18 부동 소수점 형식과 정수 형식 사이의 변환

\*부동 소수점 -> 정수 형식 변환 :소수점이하는 버림. 반올림은 없음…

3.4.19 문자열을 숫자로, 숫자를 문자열로

\* “12345” vs 12345

\* 문자 -> 숫자 해결책: Parse(), Convet.Tolnt32()

\* 숫자 -> 문자 해결책: ToString()

3.5 상수와 열거 형식

\*데이터를 절대 바꿀 수 없는 메모리 공간

\*상용 소프트웨어 제작의 복잡성에서 변수의 일관성

-값을 바꿔도 되는 변수

-값을 바뀌지 않아야 하는 변수의 구분->미리 정해놓자…

\*상수와 열거 형식은 프로그래머를 편하게 한다.

-컴파일러가 미리 알아챈다. -> 버그를 최소화함.

3.5.1 상수 – 언제나 변하지 않을거에요

\* 상수 선언 형식

-const 자료형 상수명 = 값;

Ex) const int a = 3;

Const double b = 3.14;

3.5.2 열거 형식

\*종류는 같지만 다른 값을 갖는 상수

- 같은 범주에 속하는 여러 개의 상수를 선언할 때 사용자의 응답 -> yes, no, confirm, cancle, ok

\*열거형 선언 형식

Enum 열거 형식명: 기반 자료형 {상수1,상수2,상수3,…}

Enum 열거 형식명 {상수1=값1, 상수2=값2}

기본값은 int

3.6 Nullable 형식

\* 어떤 값도 가지지 않는 변수가 필요할 때: 0이 아닌 비어있는 변수, 즉 null상태

\* Nullable 선언 형식 – 데이터형식? 변수이름; (= int? a = null;)

\* Nullable 형식의 2가지 속성.

- HasValue: 변수에서 값을 갖고 있는지 여부

- Value: 변수에 담겨 있는 값

3.7 var 형식

\*C#은 강력한 형식 언어: 의도치 않은 형식의 데이터를 읽거나 할당하는 일 차단

\*약한 형식 검사: 코드작성단계에서 편리. 컴파일러에서 변수에 담긴 데이터에 따라 자동으로 형식 지정

\*C#의 약한 형식 검사 방법-> Var

@선언과 동시 초기화 필수/ 지역 변수로만 사용.

Ex) var a = 30;

Var b = ‘Hi’;

3.8 공용 형식 시스템

\*C#의 모든 데이터 형식 체계

-공용 형식 시스템이라는 .NET프레임워크의 형식 체계의 표준 준수 -> .NET 언어들 간의 호환성

\* CTS를 따르는 C# vs Visual Basic vs C++

\* GetType() 메소드를 사용하면 공용형식시스템을 알수있음.