講義メモ

・「p.50 type05.cs」の解説とアレンジ演習から

p.50 type05.csの解説

・double bl = double.Parse(Console.ReadLine()); は分解すると、下記のようになる
① double bl; //変換用の実数型変数の宣言
② string work; //入力用の文字列型変数の宣言
③ work = Console.ReadLine(); //コンソールから文字列（実数になる数字列）を入力
④ bl = double.Parse(work); //文字列を実数に変換して①に代入
・Console.WriteLine("BMI = {0:##.#}", bw / Math.Pow(bl, 2.0)); は分解すると、下記のようになる
① double work; //身長の2乗用の実数型変数の宣言
② work = Math.Pow(bl, 2.0); //変数blの値(身長)の2乗を得る
③ double bmi; //BMI用の実数型変数の宣言
④ bmi = bw / work; //BMIを得る
⑤ Console.WriteLine("BMI = {0:##.#}", bmi); //変数bmiからBMI値を得てカスタム書式指定##.#（整数部2桁以上、小数部1桁）で表示

アレンジ演習：**p.50 type05.cs**

・身長の入力をメートル単位ではなくセンチメートル単位にしよう

作成例

```
//アレンジ演習：p.50 type05.cs
using System;
class type05
{
   public static void Main()
   {
      Console.Write("身長(cm)--- "); //【変更】
      double bl = double.Parse(Console.ReadLine()); //入力し実数変換
      Console.Write("体重(kg)--- ");
      double bw = double.Parse(Console.ReadLine()); //入力し実数変換
      Console.WriteLine("BMI = {0:##.#}", //カスタム書式指定で表示
         bw / Math.Pow(bl / 100, 2.0)); //【変更】BMI値の計算を表示時に行う
   }
}
```

p.50 decimal型

・C#以外の言語にはあまり見られない（C/C++/Javaにはない）実数型/10進データ型がdecimal（10進数の意味）
・本来、コンピュータ内部での処理は2進数であり、10進数からの変換が行われるため、誤差が避けられない
・例えば、10進数の0.5は2進数0.1にできるが、10進数0.1は2進数にすると（1/7や1/3のような）循環小数になる
・そこで、1変数あたりの容量を大きくする（128ビット）ことで誤差を出来る限り出さないようにするのが、decimal型で金額計算など精密計算に利用できる
・decimal型変数への実数の代入や初期化では（p.59で後述する）サフィックスMまたはmを末尾に加えること。
例：decimal a = 3.14M, b; b = 2.21m; //Mでもmでも良いが統一することが推奨されることが多い
※チームルールによるが、mはメートルに見えるのでMを推奨する場合や、Mはメガに見えるのでmを推奨する場合がある

アレンジ演習：**p.51 type06.cs**

・結果の桁数が非常に多いので、指数表示(p.31)にできるかどうか確認しよう
・decimal:-7.922816E+028～7.922816E+028 となり、±約7.9×10^{28}の範囲だと分かる
・また、円周率 3.14159265358979323826464643502884197 を用いて、float型、double型、decimal型で何桁まで扱えるかを試そう。

作成例

```
//アレンジ演習：p.51 type06.cs
using System;
class type06
{
   public static void Main()
   {
      Console.WriteLine("decimal:{0,0:E}～{1,0:E}", //【変更】指数表記に
         decimal.MinValue, decimal.MaxValue);
      //【以下追加】
      float   f = 3.14159265358979323826464643502884197F;
      double  d = 3.14159265358979323826464643502884197;
      decimal m = 3.14159265358979323826464643502884197M;
      Console.WriteLine("float   : {0}", f);
      Console.WriteLine("double  : {0}", d);
      Console.WriteLine("decimal : {0}", m);
   }
}
```

実行結果

```
decimal:-7.922816E+028～7.922816E+028
float   : 3.141593
double  : 3.14159265358979
decimal : 3.14159265358979323826464435029
```

アレンジ演習：**p.52 type07.cs**

・021～022行目、024～025行目、027～028行目の各2行は同じ内容で、各1行に短縮できることを試そう
・すると、変数aは変数totalで代用できてしまうことがわかる。無駄な変数aを取り除こう

作成例

```
//アレンジ演習：p.53 type07.cs
using System;
class type07
{
   public static void Main()
   {
      //decimal total; //【削除】元利合計用
      Console.Write("借入金額---");
      decimal total = decimal.Parse(Console.ReadLine()); //【変更】実数変換
      Console.Write("利息(%)---");
```

```
        decimal p = decimal.Parse(Console.ReadLine()); //実数変換
        decimal r = p / 100M; //パーセントから小数値を得る
        total = total * (1m + r); //【変更】元利合計にする
        Console.WriteLine("1期間後の元利合計は{0:c}です", total);
        //a = total; //【削除】
        total = total * (1m + r); //【変更】元利合計にする
        Console.WriteLine("2期間後の元利合計は{0:c}です", total);
        //a = total; //【削除】
        total = total * (1m + r); //【変更】元利合計にする
        Console.WriteLine("3期間後の元利合計は{0:c}です", total);
        //a = total; //【削除】
        total = total * (1m + r); //【変更】元利合計にする
        Console.WriteLine("4期間後の元利合計は{0:c}です", total);
    }
}
```

p.53 文字型

・C#では文字を現わす番号の体系（文字コード）としてUnicodeを用いている
　⇒「スタート」「すべて」「Windowsツール」「文字コード表」で表示される「U+●●●●」の「●●●●」が文字コード（16進数）
　※フォントがArialなどでは、ひらがなや漢字は表示されないが、MSゴシックなどに切り替えると表示される
・C#の文字型は半角文字（A,1など）全角文字（あ,亜,ア,A,1など）の両方に対応し、char型（.NETではSystem.Char型）として1文字ずつ扱える
・文字型変数への文字の代入や初期化では’(シングルコーテーション)で文字を囲むこと（p.58で後述する文字リテラル）。
　例：char a = ’あ’, b; b = ’1’; //文字型変数aを文字’あ’で初期化、bを宣言し文字’1’を代入
・なお、文字’1’が示すのは文字コード0x31であり、数値1とは一致しないので注意

アレンジ演習：**p.54 type08.cs**

・文字型変数g、h、iを追加し、すべてに文字’1’を代入し、表示しよう

作成例

```
///アレンジ演習：p.54 type08.cs
using System;
class type08
{
    public static void Main()
    {
        char a = '猫', b = 'で', c = 'も', d = 'わ', e = 'か', f = 'る'; //文字型変数6個の初期化
        Console.Write(a); //文字型変数の値(格納されている文字)を表示※改行しない
        Console.Write(b); //同上
        Console.Write(c); //同上
        Console.Write(d); //同上
        Console.Write(e); //同上
        Console.Write(f); //同上
        Console.WriteLine(); //改行のみ行う
        //【以下追加】
        char g, h, i; //文字型変数の宣言
        g = h = i = '1'; //3つの文字型変数全てに文字'1'を代入
        Console.WriteLine("gは{0} hは{1} iは{2}", g, h, i);
```

```
    }
}
```

p.53 文字型（つづき）

・Uncodeの文字コード●●●●を '\u●●●●' 形式で文字の代わりに指定することも可能
　例：char g = '\u4FA1'; //文字型変数gを文字'価'で初期化
・16進数を示す「\x」で代用可能
　※チームルールによっては文字の場合は「\u」の指定を義務付けている場合もある
・Uncodeの文字コードを10進数に変換して用いることも可能で、この値を文字コードとして文字型変数に格納する場合は、文字型に型キャスト（p.65で後述）すれば良い
　例：char h = (char)20385; //文字'価'の文字コードは16進数4FA1で、10進数に変換すると20385

アレンジ演習：**p.55 type09.cs**

・上記の2つの例をプログラムに追記して、表示させて確認しよう

作成例

```
//アレンジ演習：p.55 type09.cs
using System;
class type09
{
    public static void Main()
    {
        char a = '\u732B'; //Unicodeで'猫'
        char b = '\x3067'; //16進数で'で'
        char c = 'も';
        char d = (char)12431; //10進数で'わ'
        char e = '\u304B'; //Unicodeで'か'
        char f = '\x308B'; //16進数で'る'
        Console.WriteLine("{0}{1}{2}{3}{4}{5}", a, b, c, d, e, f);
        //【以下追加】
        char g = '\u4FA1'; //文字型変数gを文字'価'で初期化
        char h = (char)20385; //文字'価'の文字コードは16進数4FA1で、10進数に変換すると20385
        Console.WriteLine("{0}{1}", g, h);
    }
}
```

p.55 エスケープ文字

・Unicodeの前方には制御文字と言われる特別な意味を持つ文字が含まれており、各種の動作の指定に用いられている
・制御文字は文字コードでも指定できるが可読性の為に、エスケープ文字（またはエスケープシーケンス）と呼ばれる文字表現があてはめられている
・p.56表3.4のうち、改行を示す「\n」などが良く用いられるが、「\r」「\t」「\b」「\v」「\f」の動作は実行環境に依存するので注意
・加えて、特殊な意味を持つ文字を単なる文字として扱いたい場合のエスケープ文字があり「\\」「\'」「\"」は円マーク、シングルクォーテーション、ダブルクォーテーションの持つ意味を消し、単なる文字として扱う為に用いる

アレンジ演習　**p.56 escape01.cs**

・エスケープ文字を使って文字列cat「名前は”猫”です」を表示する処理を追加しよう
・また、文字型変数cにシングルコーテーションを、yに円マークを代入し、この2つを用いて「Let’s get \50」と表示する処理を追加しよう

作成例

```
//アレンジ演習 p.56 escape01.cs
using System;
class escape01
{
    public static void Main()
    {
        char n = '\n'; //文字型変数nを改行文字で初期化
        string str1 = "今日は";
        string str2 = "よい天気です";
        Console.WriteLine(str1 + n + str2); //途中で改行文字を出力
        string str3 = "今日は\nよい天気です"; //改行文字を含む文字列
        Console.WriteLine(str3); //表示すると途中で改行
        //【以下追加】
        string cat = "名前は\"猫\"です"; //エスケープ文字\"を含む文字列
        Console.WriteLine(cat); //表示すると「名前は"猫"です」
        char c = '\'', y = '\\'; //シングルコーテーション、円マークで初期化
        Console.WriteLine("Let{0}s get {1}50", c, y); //表示すると「Let’s get \50」
    }
}
```

p.57 論理型

・変数が2つの状態しか持たない場合や、2進数1桁分の情報のみの場合、論理型にすると効率が良く、2種類以外の値が混入することを避けられる。
・また、後述する関係演算子（p.88）などで得られる「該当する」「該当しない」「はい」「いいえ」などの情報を変数で持つことができる
※ C言語には論理型がないので、0と0以外(通常1)で整数型で表現しているが、0、1以外の値が混入することによるトラブルが避けられない
・論理型は「該当する」「はい」などを示すキーワード true、「該当しない」「いいえ」などを示すキーワード false で現わし、bool型（.NETではSystem.Boolean型）で扱える
  例：bool a = true, b; b = false; //論理型変数aをtrue’初期化、bを宣言しfalseを代入
・なお、ゲームなどではフラグのオン・オフに用いることが多い
  例：bool life = true; //生存フラグオン

p.57 bool01.cs 補足

・013行目の「a.GetType()」は変数aの型情報を文字列として返すメソッドで、型情報は.NET型で得られるので、今回は文字列”System.Boolean”が返ってくる（詳細はp.59で後述）
・014行目の「a.ToString()」は変数aがもつ情報の文字列表現が返されるメソッドで、論理型変数の場合、値がtrueであれば”True”、falseであれば”False” が返ってくる
・なお、012行目で、変数aとbの値を表示しており、この場合も”True”、”False”になっているのは、bool型を実装しているSystem.Booleanの中にあるToString()が自動的に呼び出されているため
※ 詳細は、第7章以降で説明するクラスにおいて説明

提出：アレンジ演習：**p.57 bool01.cs**

・bool型の変数aに整数値を代入するとどうなるか確認し、コメントアウトしよう
・bool型の変数bに文字列”true”を代入するとどうなるか確認し、コメントアウトしよう
・int型の変数c、double型の変数d、float型の変数e、decimal型の変数f、uint型の変数gを定義し、それぞれの型情報をGetType()で表示する処理を追加しよう

次回予告：

・p.58「リテラル」から再開します