講義メモ
・p.23「変数を使ってみる（代入とデータ型）」から再開します

提出フォロー：アレンジ演習：p.17 myname01.cs

・Console.WriteLineメソッドはMainの中にいくつでも記述できる
・２つ記述すると２行表示される
・そこで、「私の名前は●●」の下に「またの名を●●」と表示するようにしよう

作成例

```
//アレンジ演習： myname01.cs
using System;
class MyName01 {
    public static void Main() {
        Console.WriteLine("私の名前はシャア");
        Console.WriteLine("またの名を赤い彗星");
    }
}
```

p.23 変数を使ってみる（代入とデータ型）」

・C#では、整数用であるint型の変数に実数(小数点のある値)は代入できない(エラーになる
)
・なお、宣言と同時に値を代入する「初期化」が可能で、誤って何も入っていない変数を使ってしまわないにように、初期化が推奨される場合がある
・初期化の書式： データ型 変数名 = 初期値; //宣言と同時に値を代入
  例： int hp = 100; //int型の変数hpを値100で初期化
  ※ 他の変数の値や式の結果で初期化することもできる
・複数の同型の変数を同時に宣言し、その全部または一部を初期化することもできる
  例： int hp = 100, mp; //int型の変数hpを値100で初期化、変数mpは宣言のみ
  例： int hp = 100, mp = hp, level = hp / 10; //int型の変数hpを値100で、mpはhpの値で、levelはその1/10で初期化
・変数名のルール：英大文字小文字、数字、「_」、２バイト文字(漢字)が利用可だが、先頭は数字不可。また、２バイト文字(漢字)や先頭の「_」はチームルールによっては推奨されない。
  ※ 他の言語では「-」「@」「$」などが利用可能な場合があり、コンバートでは注意。
・プログラムの文法上、変数名などに使えない単語がキーワード(予約語)で、C#のキーワードはすべて小文字。
・なお、予約語と同じ綴りで大文字にしたものは利用可能だが、チームルールによっては推奨されない。
・また、予約語の前に「@」を付けると変数名に使えるとあるが、推奨されない。

p.25 変数のデータを表示する

・Console.WriteLine(変数名) とすると、変数に格納されている値が表示される
  例： int hp = 100; Console.WriteLine(hp); //「100」と表示
・Console.WriteLine(式) とすると、式の計算結果が表示される
  例： int hp = 100; Console.WriteLine(hp + 20); //「120」と表示
・加算が可能な式で「+」を指定すると加算になるが、どちらかまたは両方が"文字列"だと連結になる
  例： Console.WriteLine(3 + "倍速い"); //「3倍速い」と表示
  例： Console.WriteLine("HPは" + hp + "です"); //「HPは100です」と表示

【演習ガイド】前回USBメモリに格納したC#プロジェクトをVisual Studioで開くには

① Visual Studio起動
②「プロジェクトやソリューションを開く」
③ 保存済のプロジェクトのフォルダ(例：Project1)にある「プロジェクト名.sln」をクリックして「開く」

※画面上に「0個の参照」というCodeLensメッセージが表示されて邪魔に感じたら、下記の手順で非表示にできる
  ①「ツール」「オプション」
  ②「テキストエディタ」「すべての言語」「CodeLens」
  ③「CodeLensを有効にする」のチェックを外して「OK」
※Visual Studioの画面構成を初期状態に戻したい場合①「ウィンドウ」「ウィンドウレイアウトのリセット」「はい」

p.25 text01.cs

```
//p.25 text01.cs
using System;
class Text01 {
    public static void Main() {
        int a = 10, b = 20, total; //int型の変数a,bを初期化し、cを宣言
        total = a + b; //変数aと変数bの値の和を変数totalに代入(この"+"は加算)
        Console.WriteLine(a + " + " + b + " = " + total); //3変数値と文字列を連結し表示
    }
}
```

p.26 変数のデータを表示する（式の値を表示）

・計算結果の値を連結することもできる
・この場合は先に計算する必要があるので、式をカッコで囲むこと
・例： 10 + " - " + 3 + " = " + (10 - 3) ⇒ "10 - 3 = 7"となる("+"はすべて連結)

p.26 text02.cs

```
//p.16 text02.cs
using System;
class Text02 {
    public static void Main() {
        int a = 10, b = 20; //int型の変数a,bを初期化
        Console.WriteLine(a + " + " + b + " = " + (a + b)); //2変数値と式の値を連結し表示
    }
}
```

アレンジ演習：p.26 text02.cs

・p.26にある通り「a + " + " + b + " = " + a + b」とすると誤った結果になることを確認しよう
・理由は：
  ①「a + " + "」で連結されて「"10 + "」となる
  ②「a + " + " + b」で連結されて「"10 + 20"」となる
  ③「a + " + " + b + " = "」で連結されて「"10 + 20 = "」となる
  ④「a + " + " + b + " = " + a」で連結されて「"10 + 20 = 10"」となる
  ⑤「a + " + " + b + " = " + a + b」で連結されて「"10 + 20 = 1020"」となる

作成例

```
//アレンジ演習：p.16 text02.cs
using System;
class Text02 {
    public static void Main() {
        int a = 10, b = 20; //int型の変数a,bを初期化
        Console.WriteLine(a + " + " + b + " = " + a + b); //2変数値とaとbの値を
連結し表示
    }
}
```

p.27 WriteLineメソッドとWriteメソッド

・Console.WriteLineメソッドはカッコ内に(引数として)指定された内容を表示して改行する
・改行の必要がなく、次の表示をその後ろからにしたい場合はConsole.Writeメソッドを用いると良い
例：
```
Console.Write("円周率は");
Console.WriteLine(3.141592653258979); //「円周率は3.141592653258979」と表示して
改行
```
・また、改行だけを行うにはConsole.WriteLine();を用いると良い

p.27 text03.cs

```
//p.27 text03.cs
using System;
class Text03 {
    public static void Main() {
        Console.Write("あ"); //文字列を表示(改行しない)
        Console.Write("い"); //文字列を表示(改行しない)
        Console.Write("う"); //文字列を表示(改行しない)
        Console.Write("え"); //文字列を表示(改行しない)
        Console.Write("お"); //文字列を表示(改行しない)
        Console.WriteLine(); //改行のみ
    }
}
```

p.28 フォーマット指定子を使った変数の表示

・連結によって文字列を編集して表示する手法は「+」を連結に用いるために見づらくなる
・そこで、予め値を埋め込む穴を番号付きで指定して、第２引数以降で値や式を記述できる
・これに用いる書式をフォーマット指定子といい、穴は{0},{1},{2},…といくつで指定できる
・例： Console.WriteLine("1 + 2 = {0}", 3); //「1 + 2 = 3」と表示
・例： Console.WriteLine("{0} + {1} = 3", 1, 2); //「1 + 2 = 3」と表示
・例： Console.WriteLine("{0} + {1} = {2}", 10, 20, 10 + 20); //「10 + 20 = 30」と表示
・例： Console.WriteLine("{0}倍速い！", 3); //「3倍速い！」と表示
・穴の番号は必ずゼロからにすること。なお、再利用も可能。
・例： Console.WriteLine("{0}倍速いので{0}倍先に行ける！", 3); //「3倍速いので3倍先に行ける！」と表示
・穴の番号のことを「引数の番号」ともいう

p.28 text04.cs

```
//p.28 text04.cs
using System;
class Text04 {
    public static void Main() {
        int x = 10;
        Console.WriteLine("x = {0}", x); //「x = 10」を表示
        Console.WriteLine("x = {0}, xの10倍は{1}です", x, x * 10);
        Console.WriteLine("{0}は{1}ですが、{2}は{1}ではありません", "猫", "哺乳類", "トカゲ
");
    }
}
```

p.29 桁数の指定

・フォーマット指定子で引数の番号の後に「,表示桁数」を指定できる。ただしこれは「最低桁数」であり、表示する値がこの桁数を上回ると指定は無視される（エラーにはならない）
・つまり「表示において確保してほしいスペースの大きさ」を示す
・桁数を下回る場合、前にその数のスペースが挿入される
・例： Console.WriteLine("{0, 5}", 123); //「 123」と前に空白が２個入る
・例： Console.WriteLine("{0, 5}", 123456); //「123456」となり、桁数指定は無視される
・桁数に負の数を指定すると、桁数を下回る場合、後ろにその絶対値の数のスペースが挿入される
・例： Console.WriteLine("{0, -5}", 123); //「123 」と後ろに空白が２個入る

p.30 text05.cs

```
//p.30 text05.cs
using System;
class Text05 {
    public static void Main() {
        int x = 10, y = 123456789;
        Console.WriteLine("12334567890123456789012345678901234567890"); //30桁のゲージを表示
        Console.WriteLine("{0, 10}", "abc"); //桁数超過なので桁数は無視
        Console.WriteLine("{0, 5}",  "def"); //前に空白が2個入る
        Console.WriteLine("{0, 0}",  "ghi"); //桁数ゼロなので無視
        Console.WriteLine("{0, 10}{1, 10}",  "あ", "い"); //共に前に空白が9個入る
        Console.WriteLine("{0, -10}{1, -10}",  "あ", "い"); //共に後に空白が9個入る
        Console.WriteLine("x = {0, 5}, y = {1, 3}",  x, y); //10の前に空白が3個入る
    }
}
```

p.30 標準書式指定文字を使った書式の指定

・桁数の指定において、桁数の次に「:標準書式指定文字」を付けることで、予め用意されている書式への変換や編集をしてくれる
・ただ、あまり利用されいないものや仕様が複雑なものもあるので、ここでは抜粋して説明のみとする
① 16進数に変換　：{引数の番号, 桁数:X} 例：123 ⇒ 7B
② 指数形式に変換：{引数の番号, 桁数:E} 例：123.456 ⇒ 1.23456E+002（1.23456×10の2乗の意味）

③ 通貨形式に変換：{引数の番号, 桁数:C} 例：123456 ⇒ ¥123,456
④ 百分率に変換　：{引数の番号, 桁数:P} 例：0.05 ⇒ 5.00%

p.32 カスタム書式指定文字を使った書式の指定

・桁数を「0」または「#」によるプレースホルダで指定する手法で、小数点の位置と３桁カンマ区切りも指定できる
・ここでは抜粋して説明のみとする
① ゼロサプレス 例：″{0, 7:0000.00}″,1.2 ⇒ 0001.20（前と後ろに0が入る）
② 空白　　例：″{0, 7:####.##}″,1.2 ⇒　 1.2 （前と後ろに空白が入る）
③ カンマ区切り 例：″{0, 7:#,#}, 1234546 ⇒ 123,456（３桁区切りでカンマが入る）

p.35 ユーザの入力を知る

・ここまでのプログラムは実行すると常に同じ結果を表示したが、実行してからコンソールから文字列を渡すことができる
・この処理には文字列型の変数が必要
・宣言書式： string 変数名;
・そして「変数名 = Console.ReadLine();」とすると、コンソールから文字列を渡すことができる
・しかし、これだけでは分かりづらいので、その直前にConsole.Writeで「何を入力して欲しいか」を表示すると良い
例：
```
string n;
Console.Write("好きなキャラは：");
n = Console.ReadLine();
```
・文字列型の変数はそのまま表示しても良いし、連結に用いることもできる

提出：p.35 readline01.cs

次回予告：p.36「練習問題」を済ませて、第３章に進みます