講義メモ
・p.122「while文とdo while文」から

補足：p.121 sin01.csの実行方法・改

・現在のバージョンのVisual Studioでは1ページ分以上の表示を一気に行うと欠けが発生してしまうことがある
・そこで「ビルド」「ソリューションのビルド」を行ってから、「ツール」「コマンドライン」「開発者用コマンドプロンプト」を用いると良い
・ここで、プログラムのあるプロジェクトのフォルダの中のbin/debugフォルダにある.exeファイルを指定して
「プロジェクト名\bin\Debug\プロジェクト.exe」と入力すると動作する
例：「E:\ha234_C#_akiba\chap4>」に対して「chap4\bin\Debug\chap4.exe」を入力しEnter
・なお、実行確認後「X」で閉じること

提出フォローアレンジ演習：p.121 sin01.cs

・サイン値の表示がすべて「.00000」から「.99999」の5桁になるようにしよう
・「{0,7:#.#####}:」を「{0,7:#.00000}:」とすれば良い

作成例

```
//アレンジ演習：p.121 sin01.cs
using System;
class sin01
{
    public static void Main()
    {
        double s; //カウンタとして用いる実数
        //0.0度から180度(Math.PI)まで4度(Math.PI / 45.0)ずつ増やしながら繰返す
        for (double a = 0.0; a <= Math.PI; a += Math.PI / 45.0) {
            s = Math.Sin(a); //サイン値を得る
            Console.Write("{0,7:#.00000}:", s); //7桁の小数点以下5桁で表示
            //サイン値0.2につき"*"を1個表示することでグラフを描く
            for (int i = 1; i <= Math.Round(s * 50); i++) { //サイン値の50倍(小数点以下四捨五入)だけ繰返す
                Console.Write("*"); //"*"を1個表示(改行しない)
            }
            Console.WriteLine(); //1行分終わったので改行
        }
    }
}
```

アレンジ演習：p.121 sin01.cs・続き

・SinをCos(コサイン)にしよう
・Sinは0度から180度まで全て正の数だが、Cosは90度から180度までは負の数(-1まで)になる
・そこで、Cos値に+1して正の数0から2までにしよう
・そして「サイン値0.2につき"*"を1個表示」を「コサイン値0.4につき"*"を1個表示」とすれば良い

作成例

```
//アレンジ演習：p.121 sin01.cs・続き
using System;
class sin01
```

```
{
    public static void Main()
    {
        double s; //カウンタとして用いる実数
        //0.0度から180度(Math.PI)まで4度(Math.PI / 45.0)ずつ増やしながら繰返す
        for (double a = 0.0; a <= Math.PI; a += Math.PI / 45.0) {
            s = Math.Cos(a); //コサイン値を得る
            Console.Write("{0,7:#.00000}:", s); //7桁の小数点以下5桁で表示
            //コサイン値0.4につき"*"を1個表示することでグラフを描く
            for (int i = 1; i <= Math.Round((s + 1) * 25); i++) { //コサイン値+1の
25倍(小数点以下四捨五入)だけ繰返す
                Console.Write("*"); //"*"を1個表示(改行しない)
            }
            Console.WriteLine(); //1行分終わったので改行
        }
    }
}
```

p.122 while文

・for文を継続条件のみのシンプルな形式にしたものがwhile文
・よって、for文で書ける繰り返しは、全てwhile文に置き換え可能
・主に、回数指定の繰返しはfor文で、回数指定ではなく条件によっては1度も繰返さない可能性がある場合はwhile文で書くと良い
・例：活きているモンスターがいる間繰返す(いなければ繰り返しゼロ＝なにもしない)
・書式： while(継続条件) { 繰り返し内容 }
・よって、for(初期処理①, 継続条件②, 毎回最後の処理③) {繰返し内容④} は、whileにすると、
  初期処理①; while(継続条件②) { 繰返し内容④; 毎回最後の処理③ } となる
・for文もwhile文も繰返し内容が1文のみの場合「{}」は省略可能だが、チームルールによっては禁止。

p.124 while01.cs

```
//p.124 while01.cs
using System;
class while01 {
    public static void Main() {
        int i = 0;
        while (i < 100) { //iの値が100未満である間、繰返す
            Console.WriteLine("i = {0,3}", i);
            i++;
        }
    }
}
```

アレンジ演習：p.124 while01.cs

・for文で書き直そう

作成例

```
//アレンジ演習：p.124 while01.cs
using System;
class while01 {
    public static void Main() {
```

```
        for (int i = 0; i < 100; i++) { //iの値が0から100未満である間、+1しつつ繰返す
            Console.WriteLine("i = {0,3}", i);
        }
    }
}
```

p.125 while文を使った無限ループ

・for文を用いて「for(;;)」としても無限ループになるが「while(true)」としても良い
※ どちらを使うかチームルールで決めることが多い
・なお、C/C++では「while(1)」という文法でも無限ループに出来るがC#では禁止

p.125 menu01.cs

```
//p.125 menu01.cs
using System;
class menu01 {
    public static void Main() {
        bool bEnd = false; //終了フラグをオフにしておく
        while (true) { //無限ループ
            Console.WriteLine("***** Menu *****");
            Console.WriteLine("0:終了");
            Console.WriteLine("1:ファイル");
            Console.WriteLine("2:編集");
            Console.WriteLine("****************");
            Console.Write("選択---- ");
            string strAns = Console.ReadLine();
            switch (strAns) {
                case "0":
                    bEnd = true; //終了フラグをオンにする
                    break; //switch構造を抜ける(必須)
                case "1":
                    Console.WriteLine("ファイルが選択されました");
                    break; //switch構造を抜ける(必須)
                case "2":
                    Console.WriteLine("編集が選択されました");
                    break; //switch構造を抜ける(必須)
                default:
                    Console.WriteLine("入力に間違いがあります");
                    break; //switch構造を抜ける(必須)
            }
            Console.WriteLine(); //改行
            if (bEnd) { //終了フラグがオン？
                Console.WriteLine("それでは、このプログラムを終了します");
                break; //whileループから脱出する
            }
        }
    }
}
```

p.127 do-while文

・if文とwhile文はどちらも１回目の繰り返しの前に継続するか判断するので、１度も行わない場合がある処理では便利

・これらを前判定繰返しという
・対して1回目の繰り返しの後に継続するか判断するのが後判定繰り返しで、do-while文がある
・よって「やってみないと繰返すかどうか決まらない」場合に便利
・また「適切な値が入力されるまで繰返す」場合にも便利
・書式:
```
 do { //繰り返し開始
   繰り返し内容
 } while(継続条件); ←セミコロンが必須(while文と区別するためにも必要)
```

p.128 do_while01.cs

```
//p.128 do_while01.cs
using System;
class do_while01 {
    public static void Main() {
        int i = 10;
        do { //繰り返し開始
            Console.WriteLine("i = {0}", i);
        } while (i > 20); //継続条件(20超ならば繰返す)
    }
}
```

アレンジ演習:p.128 do_while01.cs

・コンソールから整数を入力することを正の数(0超)が得られるまで繰返すプログラムに書き換えよう

作成例

```
//アレンジ演習:p.128 do_while01.cs
using System;
class do_while01 {
    public static void Main() {
        int i;
        do { //繰り返し開始
            Console.Write("i = "); i = int.Parse(Console.ReadLine());
        } while (i <= 0); //継続条件(0以下ならば繰返す)
    }
}
```

アレンジ演習:p.125 menu01.cs

・whileの代わりにdo whileにしてみよう

作成例

```
//アレンジ演習:p.125 menu01.cs
using System;
class menu01 {
    public static void Main() {
        string strAns; //do-whileの継続条件に用いるので外で定義
        do { //ループ開始
            Console.WriteLine("***** Menu *****");
            Console.WriteLine("0:終了");
            Console.WriteLine("1:ファイル");
```

```
            Console.WriteLine("2:編集");
            Console.WriteLine("****************");
            Console.Write("選択---- ");
            strAns = Console.ReadLine();
            switch (strAns) {
                case "0":
                    Console.WriteLine("それでは、このプログラムを終了します");
                    break; //switch構造を抜ける(必須)
                case "1":
                    Console.WriteLine("ファイルが選択されました");
                    break; //switch構造を抜ける(必須)
                case "2":
                    Console.WriteLine("編集が選択されました");
                    break; //switch構造を抜ける(必須)
                default:
                    Console.WriteLine("入力に間違いがあります");
                    break; //switch構造を抜ける(必須)
            }
            Console.WriteLine(); //改行
        } while(strAns != "0"); //終了が選ばれていない間、繰返す
    }
}
```

p.128 goto文

・通常、業務においては利用が禁止されているか、推奨されないので割愛

p.130 continue文

・break文を繰り返しの中（ただしcaseの中を除く）に記述すると、繰り返しを中止できる。
・対して、今回の繰り返しの残り部分をスキップし、次の繰り返しに進むのがcontinue(続行)文
・主に、繰り返しの中にif文などを用いた複雑な分岐がある場合に用いる
・単純な内容であれば、if文で書いた方がわかりやすいことが多い

p.130 continue01.cs

```
//p.130 continue01.cs --- 0から100未満の2の倍数の合計を求める
using System;
class continue01 {
    public static void Main() {
        int sum = 0; //合計用
        for (int i = 0; i < 100; i++) { //0から100未満について繰返す
            if (i % 2 == 0) { //iを2で割って余りが0かどうか(偶数かどうか)
                sum += i;
            } else { //iが奇数の場合
                continue; //後続の処理をスキップして次の繰返しへ
            }
            //iが奇数の場合はcontinueされるので次の行は実行されない
            Console.WriteLine("i = {0, 2}, sum = {1, 4}", i, sum);
        }
        Console.WriteLine("合計は {0}です", sum);
    }
}
```

アレンジ演習：p.130 continue01.cs

・「continue;」を用いずに同じ処理になるようにしよう

作成例

```
//アレンジ演習：p.130 continue01.cs --- 0から100未満の2の倍数の合計を求める
using System;
class continue01 {
    public static void Main() {
        int sum = 0; //合計用
        for (int i = 0; i < 100; i++) { //0から100未満について繰返す
            if (i % 2 == 0) { //iを2で割って余りが0かどうか(偶数かどうか)
                sum += i;
                Console.WriteLine("i = {0, 2}, sum = {1, 4}", i, sum);
            }
        }
        Console.WriteLine("合計は {0}です", sum);
    }
}
```

p.132 練習問題1 ヒント

・BMI＝体重÷(身長×身長)なので、これが22となる体重は：
 22＝体重÷(身長×身長) ⇒ 体重＝22×身長×身長 で得られる
・ただし、この身長はメートル単位なので、センチメートル単位にする必要がある
・よって、体重＝22×(身長cm÷100)×(身長cm÷100)
・この式に身長を160cmから180cmについてインクリメントしながら繰り返し、身長と体重を表示すれば良い
・for文にして、初期処理を「身長←160」、継続条件を「身長≦180」、繰返し後処理を「身長++」、処理内容を「身長と体重＝22×(身長cm÷100)×(身長cm÷100)を表示」とすれば良い
・身長と体重はdouble型が適切

作成例

```
//p.132 練習問題1
using System;
class continue01 {
    public static void Main() {
        for (double height = 160; height <= 180; height++) { //身長160から180につい
て繰返す
            double weight = 22 * (height / 100) * (height / 100); //この身長で
BMI22な体重を得る
            Console.WriteLine("身長 = {0}, 体重 = {1}", height, weight);
        }
    }
}
```

p.132 練習問題2 ヒント

・コンソールから整数nを入力することを繰返し、1以上の値が入力されたら抜ける。（do-whileが良い）
・合計用の変数sumを0で初期化しておく
・カウント用の変数iの初期値を1にして、n以下である間、+1しながら、（forが良い）
 ・sumに変数iの初期値を足し込む
・繰り返しを終えたら、sumの値を表示しよう

作成例

```
//p.132 練習問題2
using System;
class ex0502 {
    public static void Main() {
        int n, sum = 0; //入力用と合計用の変数
        do {
            Console.Write("1以上の整数:"); n = int.Parse(Console.ReadLine());
        } while (n < 1); //1未満である間、繰返す(やりなおす)
        for (int i = 1; i <= n; i++) { //カウント用の変数iの初期値を1にして、n以下である
間、+1しながら繰返す
            sum += i; //合計に足し込む
        }
        Console.WriteLine("合計 = {0}", sum);
    }
}
```

第6章 配列

p.133 配列とは

・データの全てに変数名を付けるのではなく、同じ型のデータをグループにして名前を付け、何番目かを指定して用いる仕掛けが配列
・グループの名前を配列名、何番目かを指定する番号を添字、インデックスという
・また、配列を構成するデータ1つずつを要素という
・利用には変数と同様に、定義が必要だが、その前に配列名を宣言する必要があり、データ型も明示する
・宣言書式: 型[] 配列名;
・例: string[] monstername;
・配列の定義(生成)は変数の場合と同様に必要な領域の確保なので、要素何個分かも明示する
・定義(生成)書式: 配列名 = new 型[要素数];
・例: monstername = new string[100]; //モンスター100体分の配列を定義(生成)
・配列の宣言と定義(生成)はまとめて行うことも可能
・宣言と定義(生成)の書式: 型[] 配列名 = new 型[要素数];
・例: string[] monstername = new string[100];
・要素へのアクセスは、配列名[添字]で行う。添字は0から要素数-1までなので注意。
・例: monstername[0] = "ヴェルドラ"; monstername[1] = "リムル";
・変数の初期化と同様に、配列の宣言と初期値の設定を同時にすることが可能で、この時、初期値の数だけ要素が作られるので要素数は指定不要
・初期化の書式: 型[] 配列名 = {初期値①, 初期値②, …};
・例: string[] monstername = {"ヴェルドラ", "リムル"}; //要素数は2になる
・配列はオブジェクト(後述)として扱われるので、プロパティ(後述)の仕掛けを持っている。これにより、配列名.Length とすることで、要素数が得られる

p.136 average02.cs

```
//p.136 average02.cs
using System;
class average02 {
    public static void Main() {
        int[] point = {70, 80, 50}; //int型の配列pointを3要素で初期化
        int sum = 0, no; //合計、要素数
        no = point.Length; //プロパティで配列の要素数を得る
```

```
        for (int i = 0; i < no; i++) { //iを0から要素数未満まで＝全要素について繰返す
            sum += point[i]; //i番の要素の値をsumに足し込む
        }
        double average = (double)sum / no; //合計を件数で割って平均値を得る
        Console.WriteLine("合計 = {0}, 平均 = {1:##.#}", sum, average); //合計値と平均値(小数点以下1桁)を表示
    }
}
```

提出：アレンジ演習：p.136 average02.cs

・配列の3要素の値をコンソールから入力するようにしよう
・ヒント① 配列の初期化ではなく宣言と生成にする
　例：`int[] point = new int[3];`
・ヒント② コンソールからの入力は繰返しの中(sumに足し込む前)で行うと良い
・ヒント③ 配列の要素は変数と同様に扱えるので、コンソールからの入力を要素に代入する
　例：`Console.Write("{0}番：", i); point[i] = int.Parse(Console.ReadLine());`

次回予告：p.137「2次元配列」から