2.4　構造体

(1)構造体の考え方

あるグループの身長と体重が得られたものとして，例えば，統計分析等を行うときのデータ構造を考えてみましょう。

氏名(文字列)	体重(浮動小数点)	身長(浮動小数点)
佐 多 憲 二	65	171
村 瀬 真 吾	57	169
大 田 和 夫	60	168
北 田 洋 一	73	172
志 村 康 司	62	170
別 府 拓 哉	59	167
相 馬 修 一	68	176
木 田 裕 也	61	167

それぞれ次のように別々の配列で表現することができます。

```
private string[ ] 氏名 = new string[8];
private double[ ] 体重 = new double[8];
private double[ ] 身長 = new double[8];
```

プログラム作成者は，配列添え字が個人を識別していることを分かっていますが，プログラム上では，このことが陽に示されていません。

一方，プログラム内で次のようなカード形式に似た形で表現できれば，氏名，体重，身長がひとまとまりのデータであることが直接分かるような表現になります。

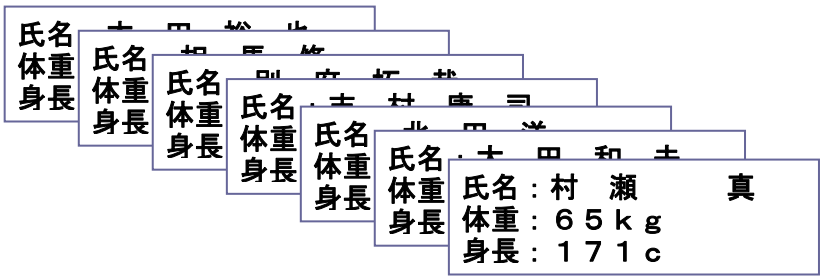


図 2-11　体格データの集まり

異なるデータの型をひとまとまりのデータとして扱うには，**構造体**を使います。このためには，まず構造体の宣言を行います。

```
public struct 体格データ
{    public string   氏名;
     public double  体重;
     public double  身長;
}
```

上記構造体宣言における識別子は，以下の意味を持ちます。

(a) **体格データ**

「体格データ」という型名。構造体タグ(structure tag)といいます。C や C++では,「struct 体格データ」が型名になりますが，C#では，型名は「体格データ」となるので注意してください。

(b) **氏名，体重，身長**

構造体の要素。構造体メンバ(structure member)といいます。なお，構造体メンバの型はすべて同じでもかまいません。

データを使うためには，まず，通常の変数の型宣言と同じように，以下のように宣言して使います。

```
体格データ X;
```

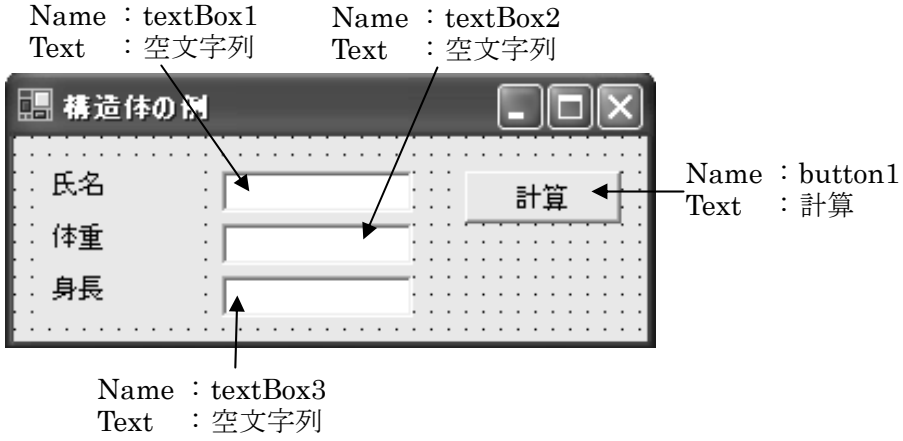
参照したり，設定するときは，次のようにドット(.)をつけて，構造体メンバ名を指定します。

```
X.氏名 = textBox1.Text;
X.体重 = double.Parse(textBox2.Text);
X.身長 = double.Parse(textBox3.Text);
```

この表現は，オブジェクトとプロパティの関係と同じです。

プログラム例として，体格データを入力して，肥満度を計算するプログラムを示します。標準体重の計算は BMI で行います。

[Program 2-14] 肥満度の計算




```
public struct 体格データ
{  public string 氏名;
   public double 体重,身長;
}
public class Form1 : System.Windows.Forms.Form
{
     .
     .
     .
private void button1_Click(object sender, System.EventArgs e)
{  体格データ X; string S;
   X.氏名 = textBox1.Text;
   X.体重 = double.Parse(textBox2.Text);
   X.身長 = double.Parse(textBox3.Text);
   double 標準体重 = X.身長 * X.身長 * 22 / 10000;
   double 肥満度   = (X.体重 - 標準体重) / 標準体重;
   if(肥満度>=0.1)       S = "太りすぎ";
   else if(肥満度<=-0.1) S = "やせすぎ";
   else                  S = "標準";
   MessageBox.Show(X.氏名 +"さんは, " + S + "です。肥満度 = "
                          + 肥満度.ToString("#0.00"));
}
```

(2)構造体の配列

構造体も配列として扱うことができます。 たとえば，以下のように宣言します。

```
private 体格データ[ ] Z=new 体格データ[8]
```

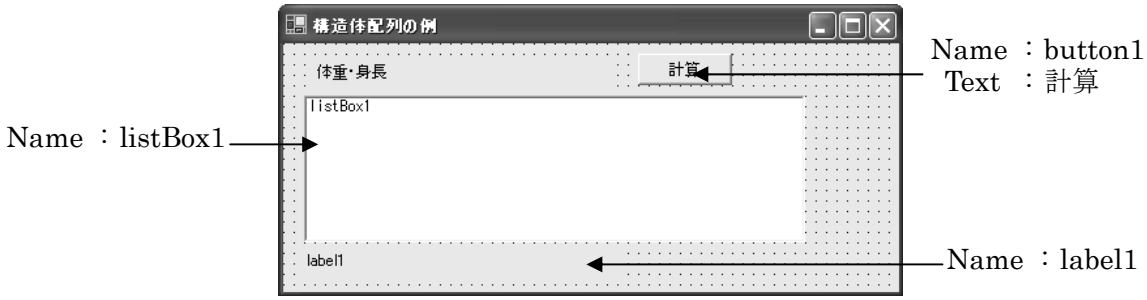
代入や参照では，以下のように指定します。

```
構造体の配列名[インデックス].構造体メンバ名
```

[例]　Z[2].氏名=“大　田　和　夫”;
　　Z[2].体重=60;
　　Z[2].身長=168;

例として，身長や体重の平均値を求めるプログラムを示します。

[Program 2-15] 身長や体重の平均値



プログラムリスト１

体格データの構造体宣言は，[Program 2-14]と同じです。

```
private 体格データ[] Z=new 体格データ[8];
private void button1_Click(object sender, System.EventArgs e)
{
  double T1=0;double T2=0;
  for(int i=0;i<Z.Length;i++) { T1 += Z[i].体重;T2 += Z[i].身長; }
  T1=T1/Z.Length; T2=T2/Z.Length;
  label1.Text=" 平均体重 = "+T1.ToString() +" 平均身長 = "
            +T2.ToString();
}
```

プログラムリスト２

```
private void Form1_Load(object sender, System.EventArgs e)
{ Z[0].氏名="佐　多　憲　二"; Z[0].体重=65; Z[0].身長=171;
  Z[1].氏名="村　瀬　　真"; Z[1].体重=57; Z[1].身長=169;
  Z[2].氏名="大　田　和　夫"; Z[2].体重=60; Z[2].身長=168;
  Z[3].氏名="北　田　洋　一"; Z[3].体重=73; Z[3].身長=172;
  Z[4].氏名="志　村　康　司"; Z[4].体重=62; Z[4].身長=170;
  Z[5].氏名="別　府　拓　哉"; Z[5].体重=59; Z[5].身長=167;
  Z[6].氏名="相　馬　修　一"; Z[6].体重=68; Z[6].身長=176;
  Z[7].氏名="木　田　裕　也"; Z[7].体重=61; Z[7].身長=167;
  for(int i=0;i<Z.Length;i++)
     listBox1.Items.Add(Z[i].氏名 + ": 体重 = "  + Z[i].体重
                                    + "kg 身長 = " + Z[i].身長);
  label1.Text="";
}
```

(3)構造体を使った演算子の定義

構造体は，ユーザが定義したデータ型と考えることができます。新しいデータ型を定義したら，そのデータ型に対する演算を考えることができます。この演算で，＋やーなど通常の演算子と同じ記号を使うことができれば，プログラムは非常に分かりやすくなります。

例として，複素数を考えてみましょう。複素数は，実部と虚部に分けることができますので，例えば，次のような構造体として表現できます。

```
    public struct Complex
    {    public double R;   // 実部
         public double I;   // 虚部
     }
```

ここで，たとえば，

```
Complex A, B, C, D ;
D = (A + B) *D ;
```

等と表現できると非常に便利です。

■基本的な複素数演算

演算子の定義を示す前に，複素数演算について整理しておきます。

【加算】 $(a_1+b_1j)+(a_2+b_2j)=(a_1+a_2)+(b_1+b_2)j$

【減算】 $(a_1+b_1j)-(a_2+b_2j)=(a_1-a_2)+(b_1-b_2)j$

【絶対値】 $|a+b\,j|=\sqrt{a^2+b^2}$

【乗算】 $(a_1+b_1j)(a_2+b_2j)=(a_1a_2-b_1b_2)+(a_1b_2+b_1a_2)j$

【逆数】 $\dfrac{1}{a+bj}=\dfrac{a}{a^2+b^2}-j\cdot\dfrac{b}{a^2+b^2}$

【除算】 $\dfrac{a_1+b_1j}{a_2+b_2j}=(a_1+b_1j)\cdot\dfrac{1}{a_2+b_2j}$

【虚数の指数】 $e^{\theta j}=\cos\theta+j\sin\theta$

【複素数の指数】 $e^{R+\theta j}=e^R\cdot e^{\theta j}=e^R(\cos\theta+j\sin\theta)$

■複素数の関数値の求め方

一般に複素数の関数値を求めるには，

$$f(a+bj)=A+Bj$$

とおき，以下のように表現することで求めることができます。

① A，B を a，b で表現する。
② A を a，b で表現して，B を A，a，b で表現する。
③ B を a，b で表現し て，A を B，a，b で表現する。

以下に，実務上よく用いられる関数計算を示しておきます。ただし，式展開は本書の枠を超えますので省略します。興味ある方は自分で計算してみましょう。

[参考]複素数の関数

①平方根　：$\sqrt{a+bj}=\sqrt{\frac{\sqrt{a^2+b^2}+a}{2}}+j\cdot\sqrt{\frac{\sqrt{a^2+b^2}-a}{2}}$

②対数　：$\log(a+bj)=\log\sqrt{a^2+b^2}+j\cdot\mathrm{Sin}^{-1}\frac{b}{\sqrt{a^2+b^2}}$

③虚数の三角関数　：$\cos(bj)=\frac{e^b+e^{-b}}{2}\quad(=\cosh b)$

$$\sin(bj)=\frac{e^{-b}-e^b}{2j}=j\cdot\frac{e^b-e^{-b}}{2}\quad(=j\cdot\sinh b)$$

$$\tan(bj)=\frac{\sin(bj)}{\cos(bj)}=j\cdot\frac{e^b-e^{-b}}{e^b+e^{-b}}\quad(=j\cdot\tanh b)$$

④複素数の三角関数　：$\sin(a+bj)=\sin a\cosh b+j\cdot\cos a\sinh b$

$$\cos(a+bj)=\cos a\cosh b-j\cdot\sin a\sinh b$$

$$\tan(a+bj)=\frac{\tan a\left(1-\tanh^2 b\right)}{1+\tan^2 a\tanh^2 b}+j\cdot\frac{\tanh b\left(\tan^2 a+1\right)}{1+\tan^2 a\tanh^2 b}$$

$\mathrm{Tan}^{-1}(a+bj)=A+Bj$ として

$$A=\mathrm{Tan}^{-1}\frac{\left(a^2+b^2-1\right)+\sqrt{\left(a^2+b^2-1\right)^2+4a^2}}{2a},\quad B=\frac{1}{2}\log\frac{a\tan A+b+1}{a\tan A-b+1}$$

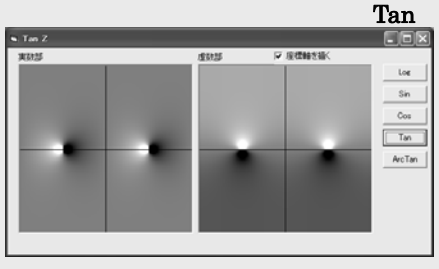
ただし $a\tan A-b+1\to 0$ のとき $B\to\infty$, $a\tan A+b+1\to 0$ のとき $B\to-\infty$

[ちょっと一息]

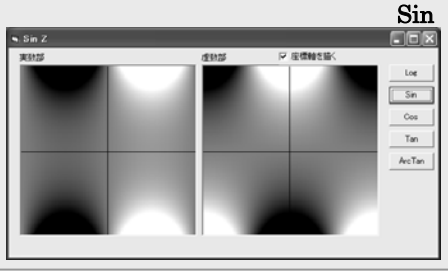
複素数関数の実数部と虚数部の値を色マップで表示すると，興味深い画像になります。以下の図は，入力の複素数の実数部を X 軸に，虚数部を Y 軸にして，結果の実数部と虚数部を色マップで表示したものです。

Log

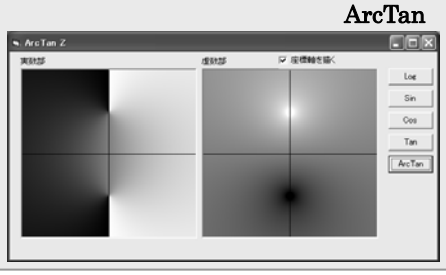
Tan



Sin



ArcTan



■その他

複素数の等値(==)は,差の絶対値が微小な値以下であることで判定します。なお，等値を定義しても

「Equals および GetHashCode はオーバーライドされません」

とのワーニングエラー(軽度のエラー)が表示されます。実行に影響はありませんが，目障りですので，これらの関数のオーバーライドを定義しておきます。

[Program 2-16] 複素数演算子定義の例

基本的演算，比較の定義例を示します。

プログラムリスト１

```
public struct Complex //複素数の構造体
{   public double R, I, E;
    // R:実部, I：虚部, E:等値比較のための誤差
    public Complex(double P1,double P2) // 誤差指定なし
    {   this.R=P1; this.I=P2;this.E=0.0000001;
    }
    public Complex(double P1,double P2,double Err)//誤差指定
    {   this.R=P1; this.I=P2;this.E=Err;
    }
    // 加算
   public static Complex operator +(Complex a,Complex b)
   {    return new Complex(a.R+b.R,a.I+b.I);
    }
    public static Complex operator +(double a,Complex b)
   {    return new Complex(a+b.R, b.I);
    }
   public static Complex operator +(Complex a,double b)
   {    return new Complex(a.R+b, a.I);
   }
   public static Complex operator +(Complex b)
   {    return new Complex(b.R,b.I);
    }
    //減算
    public static Complex operator -(Complex b)
   {  return new Complex(- b.R, - b.I);
    }
```

プログラムリスト2

```
public static Complex operator -(Complex a,Complex b)
{  return new Complex(a.R - b.R, a.I - b.I);
}
public static Complex operator -(double a,Complex b)
{  return new Complex(a - b.R, -b.I);
}
public static Complex operator -(Complex a,double b)
{  return new Complex(a.R - b, a.I);
}
// 乗算
public static Complex operator *(Complex a,Complex b)
{  double X, Y;
   X = a.R * b.R - a.I * b.I; Y = a.R * b.I + a.I * b.R;
   return new Complex(X,Y);
}
public static Complex operator *(double a,Complex b)
{  double X, Y;
   X = a * b.R;   Y = a * b.I;  return new Complex(X,Y);
}
public static Complex operator *(Complex a,double b)
{  double X, Y;
   X = a.R * b;   Y = a.I * b;  return new Complex(X,Y);
}
public static Complex operator /(Complex a,Complex b)
{  double X, Y, S;
   S = b.R * b.R + b.I * b.I; X = b.R / S; Y = - b.I / S;
   Complex c=new Complex(X,Y); return ( a * c );
}
public static Complex operator /(double a,Complex b)
{  double X, Y, S;
   S = b.R * b.R + b.I * b.I; X = b.R / S; Y = - b.I / S;
   Complex c=new Complex(X,Y);return ( a * c );
}
public static Complex operator /(Complex a,double b)
{  return new Complex( a.R/b, a.I/b );
}
// 等値比較
public static bool operator ==(Complex a,Complex b)
{   Complex c;double S1,S2; S1 = Abs(a); S2 = Abs(b);
    if(S1 < S2) S1 = S2; if (S1 < a.E)          return(true);
    c = a - b;           if (Abs(c) < S1 * a.E) return(true);
    return(false);
}
```

プログラムリスト3

```
public static bool operator ==(double a,Complex b)
{   if(Math.Abs(b.I) >b.E) return(false);
    double c = a - b.R; if (Math.Abs(c) < b.E) return(true);
    return(false);
}
public static bool operator ==(Complex a,double b)
{   if(Math.Abs(a.I) >a.E) return(false);
    double c = a.R - b; if (Math.Abs(c) < a.E) return(true);
    return(false);
}
// 非等値比較
public static bool operator !=(Complex a,Complex b)
{   return ( ! (a==b));
}
public static bool operator !=(double a,Complex b)
{   return ( ! (a==b));
}
public static bool operator !=(Complex a,double b)
{   return ( ! (a==b));
}
public static double Abs(Complex a)
{   return (a.R * a.R + a.I * a.I);
}
public static Complex Sqrt(Complex a)
{   double SS=Math.Sqrt(a.R*a.R+a.I*a.I);
    return new Complex(Math.Sqrt((SS+a.R)/2),
    Math.Sqrt((SS-a.R)/2) );
}
// 誤差値変更
public static void ErrorValue(Complex a,double Err)
{   a.E=Err;
}
 // 絶対値
public static Complex Exp(Complex a)
{  double EP=Math.Exp(a.R);double TH=a.I;
   return new Complex( Math.Cos(TH)*EP, Math.Sin(TH)*EP );
}
//指数
public static Complex Exp(Complex a)
{  double EP=Math.Exp(a.R);double TH=a.I;
   return new Complex( Math.Cos(TH)*EP, Math.Sin(TH)*EP );
}
```

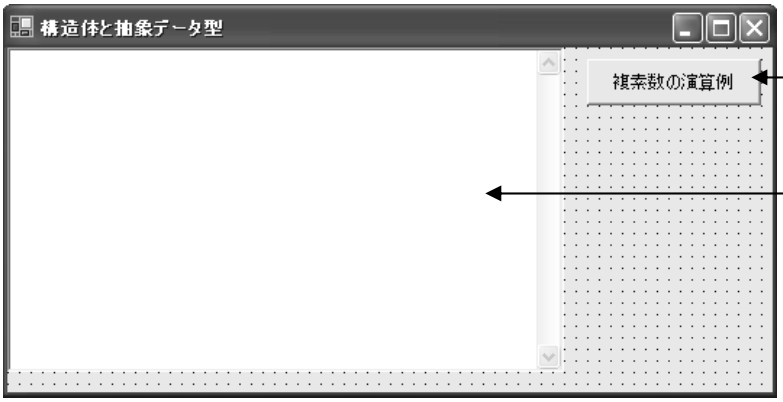
プログラムリスト４

```
    //文字列への変換
    public override string ToString( )
    {  if (R==0) return(I.ToString()+" j");
       if (I<0)  return("("+R.ToString()+" - " + (-I).ToString() +" j )");
       return("( "+R.ToString()+" + " + I.ToString() +" j )");
    }
    // 以下の定義がなければ，ワーニングエラー（軽度のエラー）が
    //表示される。なお，ワーニングエラーなので実行に影響はない。
    public override bool Equals(Object obj)
    {   //Check for null and compare run-time types.
        if (obj == null || GetType() != obj.GetType()) return false;
        Complex p = (Complex)obj;
        return (R == p.R) && (I == p.I);
    }
    public override int GetHashCode()
    {   return (int)R ^ (int) I;
    }
}
```

(4)複素数演算子定義例を用いたプログラム

演算子定義が正しいことを確認するためのプログラムを例題として示します。

[Program 2-17] 複素数演算子を使った例



Name　：button1
Text　：複素数の演算例

Name　：textBox1
Text　：True
ScrollBars：Vertical
Font　：MS ゴシック, 9pt
MulitiLine：True

```
private void button1_Click(object sender, System.EventArgs e)
{
  string S="";
  Complex A=new Complex(5,6); S += "¥r¥n A = " + A.ToString();
  Complex B=new Complex(3,4); S += "¥r¥n B = " + B.ToString();
  Complex X=new Complex(2,0); S += "¥r¥n X = " + X.ToString();
  double C=2                ; S += "¥r¥n C = " + C.ToString();

  Complex D;           S += "¥r¥n¥";
  D=A + B;             S += "¥r¥n A + B = " + D.ToString();
  D=A + C;             S += "¥r¥n A + C = " + D.ToString();
  D=C + A;             S += "¥r¥n C + A = " + D.ToString();
                       S += "¥r¥n¥";
  D=A - B;             S += "¥r¥n A - B = " + D.ToString();
  D=A - C;             S += "¥r¥n A - C = " + D.ToString();
  D=C - A;             S += "¥r¥n C - A = " + D.ToString();
                       S += "¥r¥n¥";
  D=A * B;             S += "¥r¥n A * B = " + D.ToString();
  D=A * C;             S += "¥r¥n A * C = " + D.ToString();
  D=C * A;             S += "¥r¥n C * A = " + D.ToString();
                       S += "¥r¥n¥";
  D=A / B;             S += "¥r¥n A / B = " + D.ToString();
  D=A / C;             S += "¥r¥n A / C = " + D.ToString();
  D=C / A;             S += "¥r¥n C / A = " + D.ToString();
  bool E;              S += "¥r¥n¥";
  E = A == B;          S += "¥r¥n A == B = " + E.ToString();
  E = A == C;          S += "¥r¥n A == C = " + E.ToString();
  E = C == A;          S += "¥r¥n C == A = " + E.ToString();
                       S += "¥r¥n¥";
  E = X == B;          S += "¥r¥n X == B = " + E.ToString();
  E = X == C;          S += "¥r¥n X == C = " + E.ToString();
  E = C == X;          S += "¥r¥n C == X = " + E.ToString();
                       S += "¥r¥n¥";
  E = A != B;          S += "¥r¥n A != B = " + E.ToString();
  E = A != C;          S += "¥r¥n A != C = " + E.ToString();
  E = C != A;          S += "¥r¥n C != A = " + E.ToString();
                       S += "¥r¥n¥";
  E = X != B;          S += "¥r¥n X != B = " + E.ToString();
  E = X != C;          S += "¥r¥n X != C = " + E.ToString();
  E = C != X;          S += "¥r¥n C != X = " + E.ToString();
  Complex Y;           S += "¥r¥n¥";
  Y = (A / B) * B;     S += "¥r¥n (A / B) * B = "     + Y.ToString();
  Y = (C / A) * A;     S += "¥r¥n (C / A) * A = "     + Y.ToString();
  Y = Complex.Sqrt(A); S += "¥r¥n Sqrt(A) = "         + Y.ToString();
  Y = Y * Y;           S += "¥r¥n Sqrt(A)*Sqrt(A) = " + Y.ToString();
  Y = Complex.Exp(new Complex(0,Math.PI/6));
                       S += "¥r¥n Exp(πj/6) = "       + Y.ToString();
  textBox1.Text=S;
}
```