2023 年 7 月 13 日（木）実施

構造体

レコードと構造体

複数の項目に渡るデータを一まとめにしたものを**レコード**という。例えば，次の様に学籍番号，氏名，履修科目コード，点数，評価といった項目による 1 人分のデータを一まとめにしたものは 1 件分のレコードである。

学籍番号	氏名	履修科目コード	点数	評価
22x0123	上田奈緒子	z1021234	100	S

C#言語では，この様なレコードのデータ構造を**構造体**（structure）によって表現する。構造体を用いれば，個々の項目を別々の変数で表す場合に比べてデータ間の関係が把握しやすく，プログラムの可読性が増す。また，C#言語のメソッドでは通常の変数を return 文に書いた場合，1 つのデータしか戻せないが，構造体を return 文に記述することが可能であるので，複数のデータを一まとめにして戻すことが出来る。なお，構造体は参照型ではなく，値型である。

構造体の利用

C#言語で構造体を用いる際の一連の流れは次の様になる。

1）レコード設計を行い，**構造体の定義**を行う。
2）**構造体変数の宣言**を行う。
3）構造体変数の**メンバ**への代入，メンバの参照。　（レコードの各項目をメンバと呼ぶ）

上述の学生データを例にとって，この一連の流れを表すと，次の様になる。

1）構造体の定義
```
struct Student
{
    public String id;
    public String name;
    public String code;
    public int point;
    public char eval;
}
```

2）構造体変数の宣言
```
Student st;
```

3）メンバへの代入
```
st.id = "22x0123";
st.name = "上田奈緒子";
st.code = "z1021234";
st.point = 100;
st.eval  = 'S';
```

＊ C#言語では構造体にコンストラクタやメソッドを含めることが出来るが，その様な使い方をするのであれば，クラスを定義した方が適している場合が多いので，この授業では扱わない。

本日の課題

今回の授業では，アプリの作成を通じて，構造体の基本について学ぶ。

手順

1) プロジェクトの作成

第 1 回の教材の pp.3-4 と同様にして，新規のプロジェクトを作成する。

今回は，『プロジェクト名』を「JimboStructure」（Jimbo の箇所は自分の名前に置き換える）とする。また，次回も今回作成したプロジェクトに機能を追加する。

2) コントロールの配置・プロパティの設定

先ず，フォームをクリックすることでフォーム（Form1）を選択し，『プロパティウィンドウ』で，次のプロパティを設定する。

Size：(350, 420)　← (X, Y)　　＊括弧は入力しない。

Font：12pt

Text："JimboStructure"　← 文字列リテラルの中身を入力　　＊二重引用符は入力しない。

次に，フォーム上に 5 個のラベル，4 個のテキストボックス，2 個のボタンを配置する。それぞれのコントロールのプロパティは次の様に設定する。

【ラベルのプロパティ】　Font：12pt

label1		label2		label3	
Location	(35, 25)	Location	(35, 65)	Location	(35, 105)
		Text	"学籍番号："	Text	"氏名："

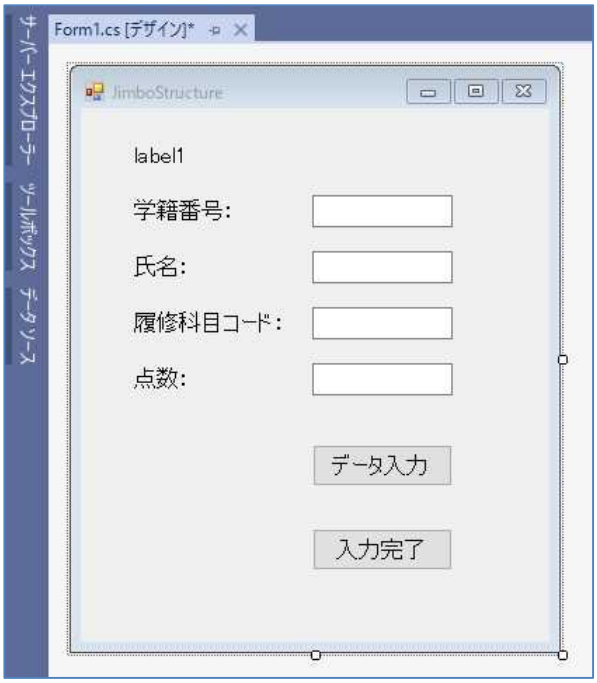
label4		label5	
Location	(35, 145)	Location	(35, 185)
Text	"履修科目コード："	Text	"点数："

【テキストボックスのプロパティ】　Font：12pt

textBox1		textBox2		textBox3		textBox4	
Location	(165, 62)	Location	(165, 102)	Location	(165, 142)	Location	(165, 182)

【ボタンのプロパティ】　Font：12pt　Size (Width：100, Height：30)

button1		button2	
Location	(165, 240)	Location	(165, 300)
Text	"データ入力"	Text	"入力完了"



3) コーディング

先ず,『表示』→『コード』と選択してコードエディタを開き，クラスのフィールドで変数，構造体及びリストを宣言する。ここで，List<T>クラス（T はリスト内の要素の型）はインデックスを使用してアクセス出来る，厳密に型指定されたオブジェクトのリストを表す。

```
13      public partial class Form1 : Form
14      {
15          int dnum = 0;
16
            3 個の参照
17          struct Student
18          {
19              public String id;
20              public String name;
21              public String code;
22              public int point;
23              public char eval;
24          }
25
26          Student st;
27
28          List<Student> listSt = new List<Student>();
29
```

フォームデザイナー上でフォームのコントロールが置かれていない箇所をダブルクリックして，イベントハンドラ Form1_Load の枠組みを作成し，その処理内容を次の様に記述する。

```
            1 個の参照
35          private void Form1_Load(object sender, EventArgs e)
36          {
37              label1.Text = (dnum + 1) + "番目のデータ";
38          }
```

続けて，点数を受け取って評価を返すメソッド eva を作成する。

```
40          private char eva(int p)
41          {
42              int rank;
43              char c;
44
45              if (p < 0 || p > 100)
46                  c = 'X';
47              else
48              {
49                  rank = p / 10;
50
51                  switch (rank)
52                  {
53                      case 6: c = 'C'; break;
54                      case 7: c = 'B'; break;
55                      case 8: c = 'A'; break;
56                      case 9:
57                      case 10: c = 'S'; break;
58                      default: c = 'D'; break;
59                  }
60              }
61
62              return c;
63          }
```

整数割る整数の結果を利用

フォームデザイナー上で button1 をダブルクリックして，イベントハンドラ button1_Click の処理内容を次の様に記述する。FormatException クラスは引数の形式が無効である場合，または複合書式指定文字列の形式が整えられていない例外を扱う。

```
            1 個の参照
65          private void button1_Click(object sender, EventArgs e)
66          {
67              st.id = textBox1.Text;
68              st.name = textBox2.Text;
69              st.code = textBox3.Text;
70
71              try
72              {
73                  st.point = Int32.Parse(textBox4.Text);
74              }
75              catch (FormatException fex)
76              {
77                  MessageBox.Show(fex.Message
78                  + "¥n整数値をテキストボックスに入力してから"
79                  + "¥n入力ボタンを押してください。"
80                  + "¥n今回はエラーが記録されます。");
81                  st.point = -1;
82              }
83              finally
84              {
85                  st.eval = eva(st.point);
86                  listSt.Add(st);
87
88                  dnum++;
89                  textBox1.Text = "";
90                  textBox2.Text = "";
91                  textBox3.Text = "";
92                  textBox4.Text = "";
93                  label1.Text = (dnum + 1) + "番目のデータ";
94              }
95          }
```

フォームデザイナー上で button2 をダブルクリックして，イベントハンドラ button2_Click の処理内容を次の様に記述する。ここで，¥t はタブコードを表す。

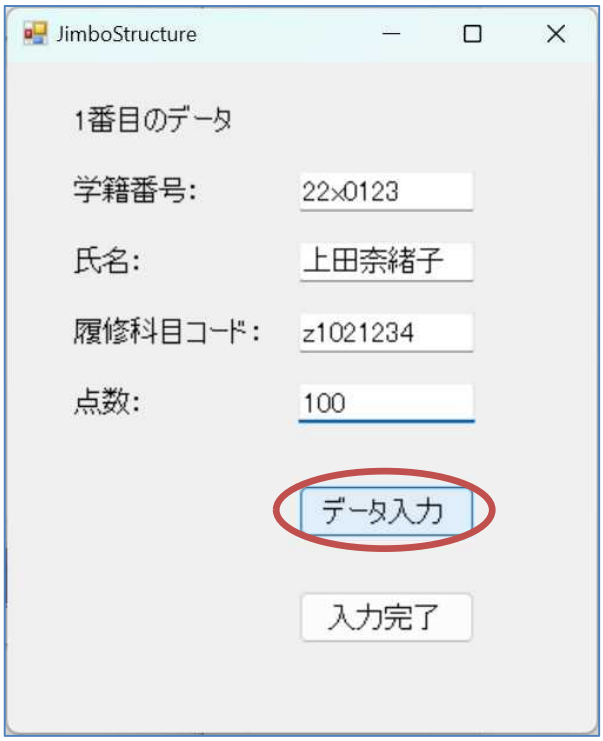
```
1 個の参照
97  private void button2_Click(object sender, EventArgs e)
98  {
99      int i = 1;
100     string mstr = "入力されたデータ";
101
102     foreach(Student list in listSt)
103     {
104         mstr += "¥n" + i + ") " + "学籍番号： " + list.id
105         + "¥t氏名： " + list.name + "¥t履修科目コード： "
106         + list.code + "¥t点数: " + list.point
107         + "¥t評価： " + list.eval;
108         i++;
109     }
110
111     MessageBox.Show(mstr);
112 }
```

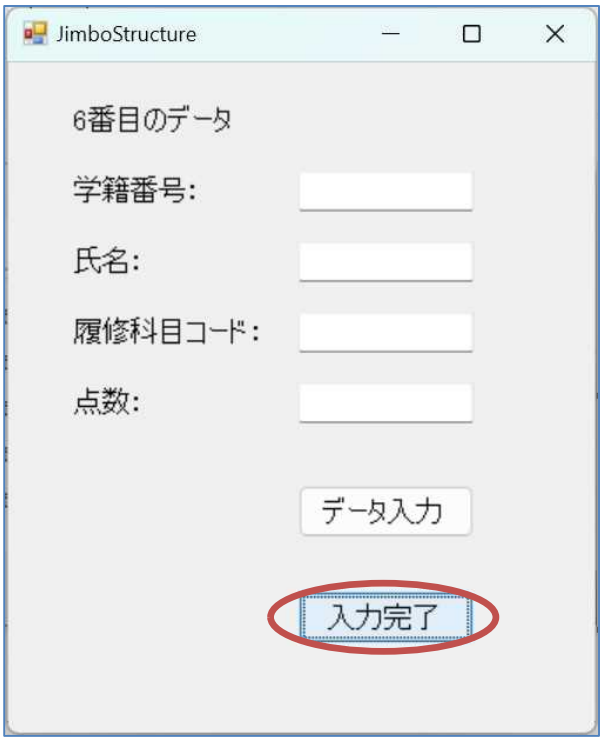
4）実行

ここで，『保存』ボタンを押してから，『開始』ボタンを押して，プログラムを実行する。

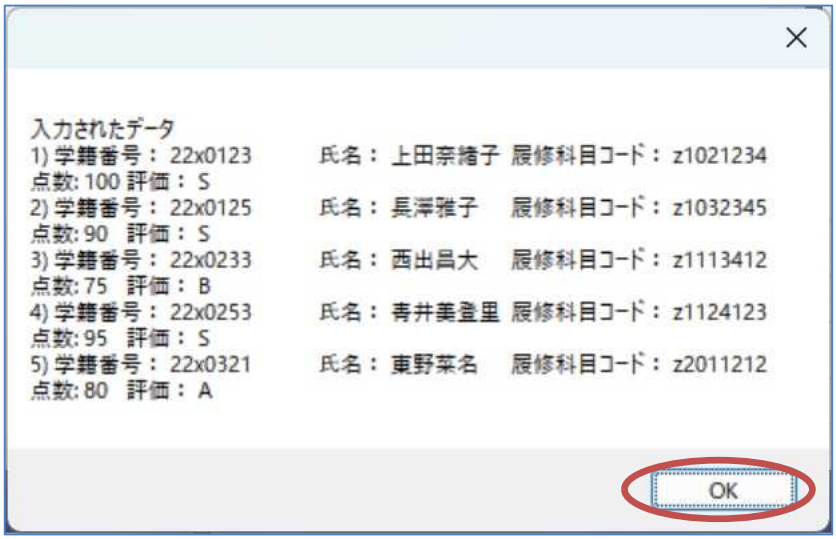
先ず，それぞれの学生のデータをテキストボックスに入力して，『データ入力』ボタンをクリックするという操作を 5 件分行う。

次に，『入力完了』ボタンをクリックする。





＊ ここでは 5 件分のデータ入力としているが，要素数が決まっている静的な配列ではなく，List<T>クラスを用いているので，入力出来るデータ数には制限は無い。



提出物：

1) フォームのデザインファイル Form1.Designer.cs をメールに添付して提出する。
2) コードエディタで編集したソースファイル Form1.cs をメールに添付して提出する。
3) 完成後に実行して，アプリが起動後，1 件目のデータをテキストボックスに文字入力した状態のスクリーンショット第 12 回実行結果 1.jpg（.png も可）をメールに添付して提出する。
4) 上の状態の後，5 件目のデータをテキストボックスに文字入力し，『データ入力』ボタンをクリックした状態のスクリーンショット第 12 回実行結果 2.jpg（.png も可）をメールに添付して提出する。
5) 上の状態の後，『入力完了』ボタンをクリックして表示されたメッセージボックスのスクリーンショット第 12 回実行結果 3.jpg（.png も可）をメールに添付して提出する。
6) 質問を記述したファイル Prog2_Questions_12th.txt に解答を書き込んで保存し，メールに添付して提出する。