世界はオブジェクトの海に浮かぶ

.NET Frameworkで楽しむオブジェクト指向

第11回 定番アルゴリズム演習“ソート”

επιστημη
えぴすてーめー

ついにマシン増強

そんなこんなでVisual Studio 2005への乗り換えを少しずつ進めております。

Level

1 2 3 4 5

Technology Tools

- ◯ Visual Basic
- ✓ Visual C#
- ✓ Visual C++
- ◯ SQL Server
- ◯ Oracle
- ◯ Access
- ◯ ASP.NET
- ✓ Other:
  ↓
  Visual Studio 2005

Samples

この記事で取り上げたソースコードおよびサンプルプログラムは、
http://www.shoeisha.com/mag/windev/
からダウンロード可能です。

今回最大の売りはεπι的には.NETのgenerics対応とC++/CLIかな。genericsはホントに欲しかった。C++に首までどっぷり浸かってて普段からtemplate（C++版generics）使いまくりなもので、C#での実装に少なからずイライラしてたものですから。

VS2005をいそいそとインストールし、しばらくは慣らし運転のつもりでIDEで遊んでいます……が今使ってるポンコツマシンでは遅くて仕方ない。IDEが立ち上がるまでにえらく待たされるし、フォームにボタンをポトリと落としただけでマウスポインタが砂時計になっちまう。遊んでんだか焦らされてんだかわかりません。なので辛抱たまらず買っちゃいました（えへへ）。

AMD Sempron 2600+に256MB-mem、40GB-HD、DVD-RWで3万ちょい。さらにメモリ256MBと無線LANカードを買い足し、余りモノのHDを押し込みディスプレイとOS（WinXP Pro）は流用。まずまずのセットが組みあがりました。今までDVD観てるとCPUメーターが100％近くまで跳ね上がってたのが20％程度、余裕しゃくしゃくです。フォアグランドプロセスの切り替えにモタつきがあるのでメモリを倍増したいところですが、メモリスロットが二本きりなので買い足すと今挿ささってるのが浮いてしまうんですよね。それとディスプレイが今時1024×768ではちと手狭。欲を言えばキリがないのは百も承知二百も合点してますけどね。ともあれ従来よりはるかに軽快な環境にひとまず満足しています。

IDE立ち上げるのが億劫でもっぱらテキストエディタとコマンドラインを愛用していたのですが、まだまだC#／.NETは不慣れなものだからMSDN首っ引き。IDEが軽快に動いてくれるのでインテリセンスに助けてもらえるのが何よりありがたく、コードを書くのがより楽しくなりました。今回はコードを書いて遊びます。

アルゴリズム演習の定番！

どんな言語であれ、入門書に決まって現われるお題のひとつが“ソート”です。要するにプログラム中に現われる要素の大小関係に基づき、要素の列を小さい順（昇順）／大きい順（降順）に並べなさい、というもの。たとえば、次の問題。

int[] array;を昇順にソートする関数、
static void sort(int[] array)
を作りなさい。

いろんなやり方（アルゴリズム）が知られています。あなたはいくつ知ってますか？

◎バブルソート

アルゴリズムが簡単なのでアルゴリズムや言語の入門書／教科書ではお約束のように紹介されるのがこのバブルソート。“昇順に並んでいる”ということは要素数nの列の中で隣接する2つの要素array[i]とarray[i+1]（i=0...N-2）について、

array[i] <= array[i+1]

が成り立っている、ってことです。だからもし、

array[i] > array[i+1]

となったところがあれば、その2つの要素を交換します。その作業を何度も繰り返し、一度も交換しなくなったらソート完了ってスンポーです。

書いてみましょう。int limitを用意し、初期値をarray.Length - 1にしておいて、

```
for ( int i = 0; i < limit; ++i ) {
  // 隣同士で大小が逆転していたら交換
  if ( array[i] > array[i+1] ) {
    int temp = array[i];
    array[i] = array[i+1];
    array[i+1] = temp;
  }
}
```

これを実行するとまずarray[0]とarray[1]の大きいほうがarray[1]となります。次にarray[1]とarray[2]の大きいほうがarray[2]に……そしてforループを抜けたとき、最大要素がarrayの末尾に浮かび上がります。これを何度も繰り返せば大きいものから順に配列の末尾側に寄っていきます（大きな要素から順に上に浮かび上がることから“バブルソート：泡立ち法”の名がつきました）。このとき配列末端の要素はすでにソートされていますから、比較／交換の対象となる要素数は繰り返しのたびにひとつずつ減らすことができます。どれだけ繰り返せばいいかというと、一度も交換せずに済むまで。なので最終的にこんなコードになります。

```
static void bubblesort(int[] array) {
  bool swapped; // 一度でも交換したらtrue
  int limit = array.Length - 1;
  do {
    swapped = false; // まだ未交換
    for ( int i = 0; i < limit; ++i ) {
      // 隣同士で大小が逆転していたら交換
      if ( array[i] > array[i+1] ) {
        int temp = array[i];
        array[i] = array[i+1];
        array[i+1] = temp;
        swapped = true; // 交換したぞ!
      }
    }
    --limit; // コレ以降はソート済
  } while ( swapped ); // 交換してたらもう一度
}
```

これで満足してちゃ“怠惰”を是とするプログラマ失格です。楽をすることに全力を注ぐのがプログラマ。たしかにこれで与えられた課題を解くことができました。だけど今後long[]やstring[]やMyClass[]をソートしたくなったら書き直しになるやないですか。もったいないったらありゃしない。なのでobject[]をソートできるよう、ちょっとだけヒネリを加えます。object[]の要素objectは比較ができませんからarray[i] > array[i+1]がコンパイルエラーとなります。なので比較のためのdelegateを外部