モデル記述のためのプログラミング 8

オブジェクト指向プログラミング （1）

07186　井原 智彦*

平成 14 年 7 月 19 日

1　はじめに

研究室では，エネルギーシステム工学や地球環境問題対応策の評価などの研究を取り扱っているが，その際，研究対象とするシステムをプログラミング言語を用いてモデル化し，そしてそのモデルを用いて評価をおこなっている場合が多い．

その際，プログラミングの要素として必要なのは，アルゴリズムに関する能力（どのようなアルゴリズムを知っているか，如何にうまく既存のアルゴリズムを用いるか，あるいは，自らアルゴリズムを考案できるか）ではなく，システム設計の能力（如何にうまく現実の現象をプログラミング言語によるモデルに投影できるか）である．

プログラミング言語としては，Fortran, Pascal, Object Pascal, C, C++, C#, Java, Visual Basic などいろいろとあるが，最近主流となっている Object Pascal, C++, C#, Java, Visual Basic はいずれも多かれ少なかれオブジェクト指向の要素を持った言語であり，我々が現実の現象をオブジェクト化する手助けをしてくれる．

ただし，Visual Basic はオブジェクト指向言語としてはやや不完全な言語であり[*1]，オブジェクトをうまく表現できない場合がある．そこで，今回からは，一部についてのみ，Visual Basic のコードを載せるにとどめ，基本的には C++の例を用いて解説していく．ただ，これまで見てきたように，Visual Basic も C++も，基本は同じ手続き型言語であり，Visual Basic プログラマも容易に C++のコードを読めるはずである．

2　サンプルソースについて

2.1　システム設計

オブジェクト指向言語は，複雑なモデルを表現する（プログラミングする）ときに威力を発揮する．逆に，簡単な計算（例えば方程式の解を求めたい，など）をする場合には不向きであり，その場合は，手続き型言語を用いた方がよい．C++にしろ Visual Basic にしろ，手続き型言語にオブジェクト指向言語の要素を取り入れた言語である[*2]ので，そのような場合でも不自由しない．

* 東京大学大学院工学系研究科地球システム工学専攻博士課程，　E-mail ihara@globalenv.t.u-tokyo.ac.jp

[*1] Microsoft 社の製品である Visual Basic 6.0 や Visual Basic for Applications 6.0 および同 6.3．次世代の Visual Basic .NET は本格的なオブジェクト指向言語となっている．

[*2] これに対し，Java は純粋にオブジェクト指向言語であると言える．

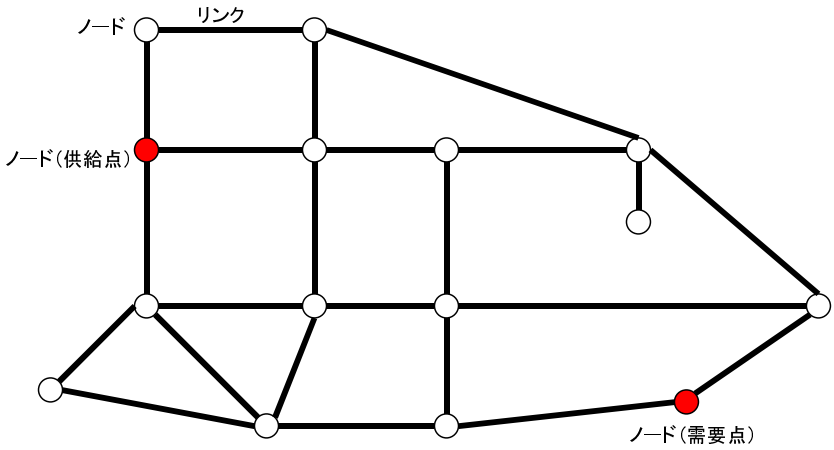

図 2.1　地域熱供給の配管

しかし，ここではオブジェクト指向プログラミングを解説したいので，何らかのシステムを例として採り上げたい．

2.2　地域熱供給の配管決定モデル

そこで，本演習では，斗成 [1] の研究テーマであった地域熱供給の配管決定モデルの一部（Dijkstra 法など）を採り上げる．

そして，全体を通して，地域熱供給の配管決定モデル（Dijkstra 法を使用）が学べるように，以降のサンプルソースでは，すべて地域熱供給配管の各要素を例として採り上げていく．

なお，地域熱供給の配管とは，図 2.1 のようになっている．

配管は道路下にしかおこなえないので，地図のうち，交差点をノード (node)，道路をリンク (link) に見立てるわけである[*3]．そして，本来は建物が供給点や需要点となるのであるが，計算の都合上，建物近くの交差点のノードを供給点や需要点とする[*4]．以下，この図を念頭において解説していく．

3　配列，構造体，STL，……（1）

オブジェクト指向の核である「クラス (class)」を解説する前に，まず，「構造体 (struct)」（Visual Basic では「ユーザー定義型 (user defined datatype)」と呼ばれる）について説明する．合わせて，配列や STL といった構造体以外のデータ構造についても紹介する．

3.1　データの集合

まず，1 つのノードについて考えてみる．すると，ノードはどのような属性を持っているであろうか？
考慮するのならば，地図であることを考えると，以下のような感じかもしれない．

[*3] “node” は「結節」「接点」，“link” は「連結」「連鎖」という意．グラフ理論の基本用語．
[*4] 本来は，建物の前の道路にノードを新しく設定するのが望ましい．

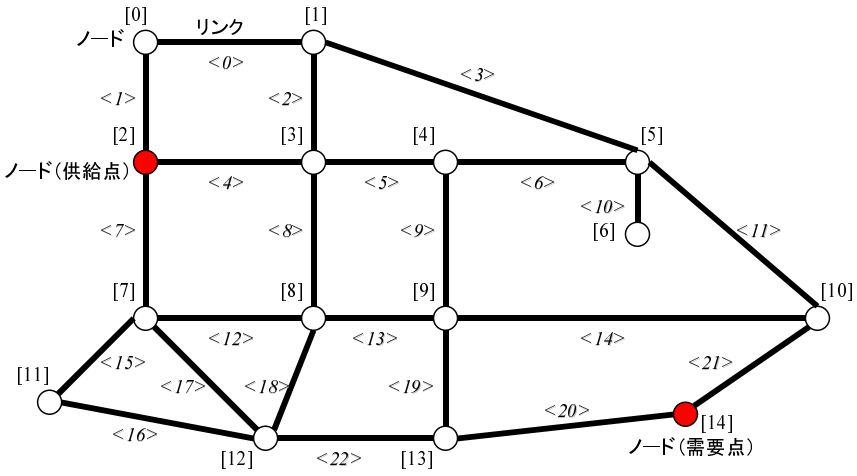

図 3.1　地域熱供給の配管（No. 付き）

```
1  #include <string>
2
3  using namespace std;
4
5  long int nNo;                          // ノード No.
6  string chartName;                      // 図名
7  double x;                              // x 座標
8  double y;                              // y 座標
```

なお，プログラムで扱う場合，それぞれのノードやリンクに No. もしくは ID を持たせてやると管理しやすくて，いろいろと好都合である．図 2.1 のノードやリンクに対して，0 から順に適当に番号（No.）を振ったものが，図 3.1 である[*5]．以下のサンプルソースは，この図 3.1 を対象にする．

ちなみに，`string` というのは文字列型（正確に言うと STL に含まれる文字列クラス）であり，旧来の文字型配列での文字列表現（`char[]`）と異なって，Visual Basic の `String` 型と同じように文字数を気にすることなく使える．ただし，`string` ヘッダをインクルードする必要がある．

3.2　配列

図 3.1 を見ると，このノードは，全部で 15 個存在する．ではこれをどうやってまとめればよいのだろうか？配列を用いてまとめてやると次のようになる．

```
1  #include <string>
2
3  using namespace std;
4
5  const long int NODE_COUNT = 15;
```

[*5] 実際には，数値地図データの方にそれぞれユニークな ID が振られている．ここではそのような ID ではなくて，単に番号付けのために数字を振りたかったので，番号（No.）とした．

```
 6
 7  long int nNo[ NODE_COUNT ];                  // ノード No. が 15 個
 8  string chartName[ NODE_COUNT ];              // 図名が 15 個
 9  double x[ NODE_COUNT ];                      // x 座標が 15 個
10  double y[ NODE_COUNT ];                      // y 座標が 15 個
```

確かにこういったやり方も方法の 1 つではある．しかし，全ての属性に対して，いちいち 15 個と指定しているのは煩わしいし，また，ノードの属性を増やした場合，例えば，z 座標も加えたくなった場合は，プログラムの管理がやっかいになる．何より，このようなやり方では，これらの変数がノードの属性ということが分かりにくい．

3.3 構造体

3.3.1 構造体の定義

そこで，構造体を導入する．すると，以下のように書き表せる． **［プログラム sample802 参照］**

```
 1  #include <string>
 2
 3  using namespace std;
 4
 5  const long int NODE_COUNT = 15;
 6
 7  struct NODE {
 8      long int nNo;                            // ノード No.
 9      string chartName;                        // 図名
10      double x;                                // x 座標
11      double y;                                // y 座標
12  };
13
14  NODE nodeVec[ NODE_COUNT ];                  // ノードが 15 個
```

この方が見た目にも分かりやすく，一目瞭然である[*6]．

■Visual Basic の場合 Type ステートメント（要するにユーザー定義型）を用いて同様のことができる．なお，Visual Basic の Long 型は，Visual C++など 32bit の C++での int 型でしかないが，long int に相当する型が存在しないため，とりあえず，以下は Long 型で代用する．

```
 1  Const NODE_COUNT As Long = 15
 2
 3  Type NODE
 4      lngNo As Long                            ' ノード No.
 5      strChartName As String                   ' 図名
 6      dblX As Double                           ' x 座標
```

[*6] ただし，各要素の特定の属性に対して繰り返してアクセスする場合は，構造体を導入すると低速になってしまう．少しでも速度を稼ぐようなプログラム（リアルタイムの画像処理など）では構造体を導入しない場合もあり得る．

```
 7      dblY As Double                          ' y 座標
 8  End Type
 9
10  Dim nodNodeVec(NODE_COUNT) As NODE          ' ノードが 15 個
```

3.3.2 構造体への値の代入

そして，構造体への値の代入は以下のような手順でおこなう． **[プログラム sample802 参照]**

```
1  nodeVec[ 0 ].nNo = 0;
2  nodeVec[ 0 ].chartName = "地図 1 交差点";
3  nodeVec[ 0 ].x = 321.1;
4  nodeVec[ 0 ].y = 15.9;
```

ちなみに，次のようにおこなうこともできる．これは配列が中括弧で括ることによって代入できるのと同じである．

```
1  node[ 0 ] = { 0, "地図 1 交差点", 321.1, 15.9 };
```

■Visual Basic の場合　ユーザー定義型も同じような手順で値を代入できる．なお，C++のような略記法は用意されていない．

```
1  nodNodeVec[ 0 ].lngNo = 0
2  nodNodeVec[ 0 ].strChartName = "地図 1 交差点"
3  nodNodeVec[ 0 ].dblX = 321.1
4  nodNodeVec[ 0 ].dblY = 15.9
```

以上でノードを構造体化し，それを配列にした（構造体配列）．

参考文献

[1] 斗成聡一. 道路配管ネットワークを考慮した地域熱供給システムに関する研究. 平成 11 年度修士論文, 東京大学, 2000.