モデル記述のためのプログラミング 6

# プログラミングの基礎（4）

07186　井原 智彦*

平成 14 年 6 月 21 日

## 1　手続き型プログラミングひとめぐり（5）

### 1.1　ループ構造

どの言語においても採用されている基本的なループ構造を簡単に解説する．

#### 1.1.1　for 文

指定された回数だけ，一連のステートメントを繰り返すフロー制御構造である．これを用いれば，たとえば階乗の計算ができる．

以下に，10 の階乗を求めるプログラムを掲載する．**［プログラム sample601 参照］**

```
 1  #include <iostream>
 2
 3  using namespace std;
 4
 5  int main() {
 6
 7      long ret = 1;
 8
 9      for ( int i = 1; i <= 10; i++ ) {
10          ret *= i;
11      }
12
13      cout << "10 の階乗は " << ret << " です。" << endl;
14
15      return 0;
16  }
```

上記は 1 から計算をおこなっているが，10 から計算してもよい．**［プログラム sample601b 参照］**

* 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      for ( int i = 10; i > 0; i-- ) {
2          ret *= i;
3      }
```

C++の場合，カウンタを 0 にセットし，それから 1 ずつ増加させる（インクリメント）のが普通である[*1]．また，ANSI C++[*2]では，for 文の括弧の中でカウンタの宣言が可能である．同じカウンタを他の箇所で使う必要がないのならば，変数のスコープを短くするという観点からも，for 文の括弧内でカウンタを宣言するのが望ましい．

**■Visual Basic の場合**　For...Next ステートメントを用いれば，ほぼ同様のことができる．以下にサンプルプログラムを掲載する．

```
 1  Option Explicit
 2
 3  Sub Main()
 4
 5      Dim lngRet As Long
 6      lngRet = 1
 7
 8      Dim i As Integer
 9      For i = 1 To 10
10          lngRet = lngRet * i
11      Next
12
13      Call MsgBox("10 の階乗は " & CStr(lngRet) & " です。")
14
15  End Sub
```

C++の場合と同じく，以下のように書ける．Step キーワードの後ろにカウンタの刻み幅を書くが，下記では-1 を指定している．なお，上記のように省略すると，自動的に+1 が指定されたものと見なされる．

```
1      Dim i As Integer
2      For i = 10 To 1 Step -1
3          lngRet = lngRet * i
4      Next
```

**■余談 1**　上記の 2 プログラムでは，それだけを作成すれば，とりあえず動くプログラムを掲載した．

たとえば，C++のコードでは，iostream ライブラリを組み込み，std 名前空間を使用する旨を述べた後，ソース全体を int 型の main() 関数で括っている．

同様に，Visual Basic のコードでは，Option Explicit から始め，ソース全体を，サブプロシージャであ

---

[*1] ただし，ここでは階乗を計算するため，インクリメントの例の場合は，1 から開始した．0 から開始すると，いくら積算を繰り返しても 0 となってしまう．

[*2] Visual C++では，Microsoft 独自拡張のチェックを外さない限り，for 文の括弧内でカウンタを宣言しても，for 文スコープとともには変数のスコープが終わらない．独自拡張の仕様を用い，かつ，ANSI C++に則ったスコープにしたい場合は，始めに，#define for if (false) ; else for と定義してやればよい．

る Sub Main() で括っている．

以下のプログラムでは，紙面を節約するために，上記の記述を前提として，説明箇所のみ掲載していく．

**■余談 2** C++のコードで用いた++演算子は，その変数を 1 つだけインクリメントさせるという演算子である（整数型でのみ使用可能[*3]）．逆に--演算子は，その変数を 1 つだけデクリメントさせる．そして，i++とすると（後置演算子），その式が解釈されてから i がインクリメントされる．一方，++i とすれば（前置演算子），その式が解釈される前に i がインクリメントされる．つまり，前置演算子の場合は，インクリメントされた i がその式で用いられる．

Visual Basic で見られない算術演算子として，他に，+=演算子などがある．これは，演算子の後ろの変数を前の変数に加算するという命令で，たとえば，i += 5; とすれば，i = i + 5; としたのと同じ結果が得られる．そのため，+=演算子は不要のように見えるが，=演算子と+演算子を使って書く後者より，記述が短く，かつ，高速であるため，よく用いられる[*4]．これは，さきほど述べた i++; が実は i = i + 1; で表現できてしまうのにもかかわらず用いられる理由と似ている．

### 1.1.2 do-while 文

for 文が回数を指定してブロックの実行を繰り返していたのに対し，この do-while 文は回数を指定せずにブロックの実行が可能である．その代わり，ブロックに入る直前もしくはブロックを出た直後に，条件式を置いておき，これを用いてブロックを繰り返すか，それとも脱出するかを決定する．

たとえば，100 を初めて超える階乗を算出してみる． **［プログラム sample602 参照］**

```
1  long ret = 1;
2  int i = 1;
3
4  do {
5      ret *= i++;
6  } while ( ret <= 100 );
7
8  cout << "100を初めて超える階乗は " << i << " の階乗で、¥n"
9      << "そのときの値は " << ret << " です。" << endl;
```

**■Visual Basic の場合** Do...Loop ステートメントを用いて同様のことができる．

```
1  Dim lngRet As Long
2  Dim i As Integer
3  lngRet = 1
4  i = 1
5
6  Do
7      lngRet = lngRet * i
8      i = i + 1
```

[*3] bool 型でも使用可能だが，その結果は常に true となる．また，bool 型には--演算子が定義されていない．

[*4] 一時変数を生成せずに，直接 i を操作するため．

```
 9  Loop While (lngRet <= 100)
10
11  Call MsgBox("100を初めて超える階乗は " & CStr(i) & " の階乗で、" & vbCrLf _
12      & "そのときの値は " & CStr(lngRet) & " です。")
```

### 1.1.3 while 文

さきほどの while 文を前に持ってくれば（つまり while(...) {...} の形にすれば），ブロックに入る前に条件分岐が可能である．

**■Visual Basic の場合** ブロックに入る前に条件をチェックしたい場合は，Do...Loop ステートメントを用いる．具体的には，Do(...)...Loop とすればよい．