ポインター

山本昌志*

2007 年 5 月 29 日

概 要

昨年の講義で簡単にポインターの仕組みを説明したが，ここではもう一歩踏み入れていろいろなオブジェクトのポインターについて学習する．

1 前回の復習と本日の学習内容

1.1 復習

前回の授業では，再帰関数について学習した．関数の定義において，自分自身を呼び出す関数を再帰関数という．具体的には，リスト 1 のような階乗を計算する関数である．階乗は，

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 2 \times 1 \tag{1}$$

であるが，漸化式を使って

$$n! = n \times (n-1)! \qquad (\text{ただし}, 1 \leqq n) \tag{2}$$

$$0! = 1 \tag{3}$$

のように定義することもできる．この漸化式そのものをプログラムにすると，リスト 1 のように再帰関数を使うことになる．

リスト 1: 繰り返し文を使った階乗の計算.

```
1  #include <stdio.h>
2
3  int kaijyo(int n);                  // プロトタイプ宣言
4
5  //========== メイン関数 ================================
6  int main(void)
7  {
8    int nx, result;
9
10   scanf("%d", &nx);                 // 整数入力
11   result=kaijyo(nx);                // 関数呼出し
12   printf("%d!=%d\n", nx, result);    // 計算結果表示
13
```

*独立行政法人　秋田工業高等専門学校　電気情報工学科

```
14    return 0;
15  }
16
17  //========== 階乗を計算する関数(再帰呼出し)===============
18  int kaijyo(int n)
19  {
20
21    if(n==0){
22      return 1;
23    }else{
24      return n*kaijyo(n-1);
25    }
26
27  }
```

実行結果

```
12
12!=479001600
```

次の二つのことをおさえれば，再帰関数を書くことができる．

- 再帰関数の終了条件を書く．
- 漸化式のように問題を分割し，その通りにプログラムを書く．ただし，問題の分割はつぎのようにする．
  - 分割したものを合わせることにより，元の問題となる．
  - 分割したひとつの問題は，元の問題よりも小さい．
  - 分割したひとつの問題は，元の問題と同じ方法で解ける．

  このように分割することにより，ある自明な解—再帰関数の終了条件—まで分割できる．そうすると問題が解けるのである．

1.2 本日の学習内容

本日は，ポインターについて学習する．ポインターについては，1 年生のときにそのメカニズムについて学習した．ここでは，後のデータ構造の学習のために，ポインターの使い方の基礎を学ぶ．本日の学習のゴールは，以下のとおりである．

- ポインターとオブジェクトの関係が分かる．
- 構造体へのポインターの使い方が分かる．
- 関数へのポインターの使い方が分かる．
- ポインターの配列を使って，文字列定数の並びを処理できる．

教科書 [2] の pp.114–133 が本日の範囲である．

2　ポインターとは何か?

ポインターとは何か?—と問われると，最も適切な答えは「ポインターはオブジェクトのアドレスである」[1] であろう．そうするとオブジェクトとは何か?—ということになる．オブジェクトとはメモリー上のデータ領域のことでアドレスがあり，型を持つのでサイズもある．ポインターはオブジェクトの先頭アドレスを示すことにより，オブジェクトを指し示す．実際には，ポインター変数にオブジェクトの先頭アドレスを格納しているのである．先頭アドレスのみならず，オブジェクトの大きさもどこかに情報として持っていることを忘れてはならない．

このことを具体例をつかって，説明しよう．リスト 2 のプログラムでは，p がポインター変数で hoge がオブジェクトである．この関係は図 1 のように表すことができる．

リスト 2: ポインターを使った簡単なプログラム．

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    int hoge=3;
6    int *p;
7
8    p=&hoge;
9
10   printf("hoge=%d\n",*p);
11
12   return 0;
13 }
```

実行結果

```
hoge=3
```

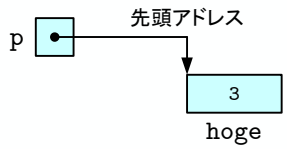

図 1: ポインター p とオブジェクト hoge の関係．

このような結果が得られるのは，リスト 2 の 10 行目でポインター p が hoge を指し示すからである．オブジェクトとポインターの関係は 8 行目で hoge で決められている; オブジェクトの先頭アドレス[1] をポインター変数に代入している．

更にコンピューターの内部まで踏み込めば，10 行目の printf() 関数では次のようなことが行われている．

[1] int は 4 バイトなので 4 つのアドレスがあるが，アドレス演算子&で先頭アドレスを取り出している．そして，それをポインター変数に代入している

- ポインター変数 p に格納されているアドレスを取り出す.
- 取り出されたアドレスは整数型のオブジェクト[2]の先頭アドレスなので，そこから 4 バイトのデータを取り出す.
- 取り出されたデータ—32 ビットの 1 と 0 の並び—を%d の書式で書き出す.

3 さまざまなポインター

3.1 単純型変数へのポインター

単純型変数—文字型や整数型，倍精度実数型など—へのポインターは，説明するまでもないだろう．1 年生の授業で学習したし，先ほどの例でも示した．そのため，ここでは単純型変数へのポインターについては，説明しない.

3.2 構造体へのポインター

構造体へのポインターはしばしば使われ，プログラムを分かりやすくするためには重要である．実際には，中間試験以降のデータ構造の学習では頻繁にこれを使うことになる.

構造体のポインターの例をリスト 3 にしめす．図 2 にポインターと構造体の関係を示す．リストを見て分かるように，構造体へのポインターは次のようにして使う.

- アドレス演算子&を使い先頭アドレスを取り出し[3]，ポインター変数へ代入する.
- ポインターが指し示す構造体のメンバーへのアクセスは，アロー演算子 (->) を使う.

リスト 3: 構造体へのポインターを使ったプログラム例.

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    typedef struct{                          // 構造体の定義
6      char name[16];
7      int math;
8      int info;
9    }student;
10
11   student yama={"yamamoto",72,83};         // 宣言と初期化
12   student *p;                              // 構造体 student 型へのポインター
13
14   p=&yama;                                 // 先頭アドレスの代入
15
16   printf("name = %s\n", p->name);          // メンバーへのアクセスは，アロー演算子
17   printf("math = %d\n", p->math);
18   printf("info = %d\n", p->info);
19
20   return 0;
21 }
```

[2]6 行目のポインターの宣言

[3]ポインターを取り出している—と表現した方がよい．しかし，混乱するだろう．ほとんどの場合，ポインターはオブジェクトの先頭アドレスである.

実行結果

```
name = yamamoto
math = 72
info = 83
```

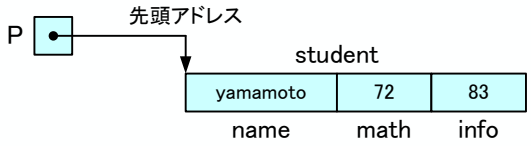

図 2: ポインター p と構造体のオブジェクト student の関係.

3.3 配列へのポインター

ここはかなり混乱しそうなので，ほとんど説明しないでおこう．もう少し C 言語の理解が深まったならば，各自，勉強すればよい．配列との関係で重用なことは，配列のコンピューターでの処理は

```
hoge[i]    ⇒    *(hoge+i)
```

となることである．

3.4 関数へのポインター

諸君にとって，最もけったいに感じるのが関数へのポインターであろう．実行すべき命令が書かれている関数もメモリーにロード—読み込んで格納—-される．したがって，先頭アドレス—関数のエントリー—をポインターに代入することができる．

リスト 4: 関数へのポインターを使ったプログラム例.

```
1  #include <stdio.h>
2
3  int add(int i, int j);              // プロトタイプ選言
4
5  //============ メイン関数================================
6  int main(void)
7  {
8    int (*fp)(int, int);              // 関数へのポインター
9
10   fp=add;                           // 関数のアドレスを代入
11
12   printf("%d\n",fp(5,9));
13
14
15   return 0;
```

```
16 }
17
18 //============ ユーザー定義関数=========================
19 int add(int i, int j)
20 {
21   return i+j;
22 }
```

実行結果

```
14
```

これは使い方によってはかなり便利である．サブルーチンへ関数を渡すことが可能となる．リスト 5 がそれを使った例である．ここで，関数定義の仮引数の `double (*f)(double)` が関数へのポインターの宣言で，

戻り値の型（*関数へのポインター変数)(引数の型)

と書く．この関数へのポインター変数に関数のポインター—先頭アドレス—を代入すると，ポインター変数が関数のように使える．関数のポインターへの代入は関数名を右辺値として，代入するだけである．

リスト 5: 関数を引数にしたプログラム例.

```
1  #include <stdio.h>
2  #include <math.h>
3
4  void print_func(double (*f)(double));  // プロトタイプ選言
5
6  //======== メイン関数 ======================================
7  int main(void)
8  {
9
10   print_func(sin);
11   print_func(cos);
12
13   return 0;
14 }
15
16 //========== 関数の値を表示する関数 ============================
17 void print_func(double (*f)(double))
18 {
19   int i;
20   double dx=0.1;
21
22   printf("-----------------------------\n");
23   for(i=0; i<=5; i++){
24     printf("%f\t%f\n", i*dx, f(i*dx));
25   }
26 }
```

実行結果

```
-----------------------------
0.000000        0.000000
```

```
0.100000        0.099833
0.200000        0.198669
0.300000        0.295520
0.400000        0.389418
0.500000        0.479426
-------------------------
0.000000        1.000000
0.100000        0.995004
0.200000        0.980067
0.300000        0.955336
0.400000        0.921061
0.500000        0.877583
```

3.5 ポインターの配列

複数組の文字列を扱う場合，ポインターの配列は便利である．リスト 6 にプログラム例を，図 3 にポインターと文字列定数の関係を示す．ここでの動作は次のようになる．

- プログラムの実行に先だって，プログラムがロードされたときにメモリーのどこかに文字列定数の文字列が格納される．
- そして，文字列のポインター—先頭アドレス—がポインターの配列に格納される．
- これらの処理が完了した後に，プログラムが動作する．

リスト 6: ポインターの配列を使ったプログラム例．

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    char *animal[]={"cat", "dog", "rabbit", "horse"};
6
7    printf("1st : %s\n", animal[0]);
8    printf("2nd : %s\n", animal[1]);
9    printf("3rd : %s\n", animal[2]);
10   printf("4th : %s\n", animal[3]);
11
12   return 0;
13 }
```

実行結果

```
1st : cat
2nd : dog
3rd : rabbit
4th : horse
```

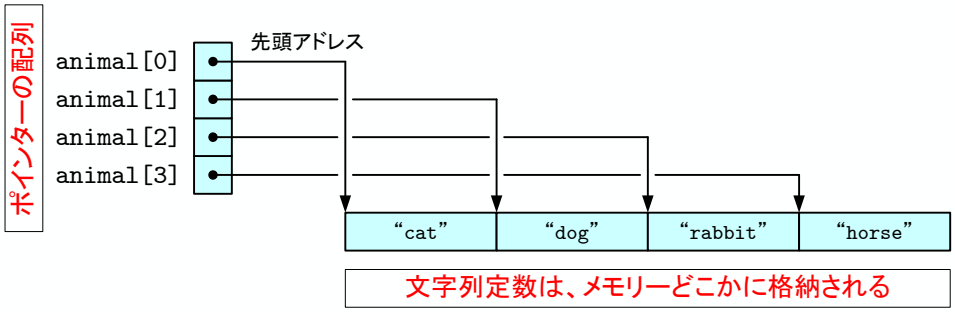

図 3: 文字列定数とポインターの配列 animal の関係.

3.6 ポインターのポインター

ポインターを指し示すポインターがある．これは，つぎのように書く．

```
int **hoge, *fuga, a;

fuga=&a;
hoge=&fuga;
```

あまり深入りしないことにする．

4 プログラム作成の練習

[練習 1] 次のような構造体を作成し，適当な値を入れ，ポインターを用いてその値を取り出して表示せよ．

- 構造体のメンバーは,「国名」と「首都」,「人口」とする．人口の単位は，[万人] または [億人]，あるいはこれ以外でも良い．各自，勝手に考えよ．

[練習 2] 次のような文字列の並びを作成して，ポインターを使って表示せよ．

```
organg, apple, grape, peach, pineapple
```

[練習 3] リスト 5 を参考にして，ふたつの関数の合計値を計算し，表示するプログラムを作成しなさい．ひとつは，ユーザー定義関数 x^2 とする．もうひとつは，$\cos(x)$ とする．表示は区間 $[0, 1]$ とする．

[練習 4] 配列へのポインターに関して，`hoge[i]` と `*(hoge+i)` が等しいことを確認しなさい．さらに，`hoge[i]` と `i[hoge]` の値が等しいことを確認しなさい．

[練習 5] 時間の余った者は，教科書を読め．

5　課題

以下の課題を実施し，レポートとして提出すること．

[問 1] (復予) 教科書 [2] の第 4 章を 3 回読め．レポートには「3 回読んだ」と書け．

[問 2] (復) 本日配布したプリントを 2 回読め．レポートには「2 回読んだ」と書け．

[問 3] (復) 次のような構造体を作成し，適当な値を入れ，ポインターを用いてその値を取り出して表示するプログラムを作成せよ．

- 構造体のメンバーは,「地方」と「県名」,「人口」とする．

[問 4] (復) ここでの学習内容でわからないところがあれば，具体的に記述せよ．

提出要領はいつものとおり．ただし，期限は 6 月 5 日 (火)AM 8:45，課題名は「課題　ポインター」とすること．

参考文献

[1] ハーバート・シルト．独習 C 第 3 版．(株) 翔泳社, 2003.

[2] 内田智史監修, (株) システム計画研究所編．C 言語によるプログラミング　応用編　第 2 版．(株) オーム社, 2006.