ポインターの応用 (関数の引数)

山本昌志*

2007 年 2 月 23 日

概 要

ポインターを関数の引数に使う方法を学習する．

1 前回の復習と本日の内容

1.1 前回の復習

先週は，以下のようなことを学習した．

- ポインターを使うためには，初期化が必要である．
- 可能なポインターの演算は，整数の加算と減算のみである．1 加算や減算はそのデータのバイト数分のアドレスが移動する．
- 配列名は，配列への先頭アドレスを示すポインターである．
- 文字列リテラル (定数) の先頭アドレスはポインターへ代入できる．

1.2 本日の学習内容

本日の学習の範囲は，教科書 [1] の p.287–292 である．学習のゴールは，以下のとおりである．

- 関数の引数に変数へのポインターを使う方法が分かる．
- 関数の引数に配列を使う方法が分かる．
- 関数の引数に文字列へのポインターを使う方法がわかる．

2 関数の引数にポインターを使う

2.1 変数

2.1.1 参照渡しと値渡しの違い

関数呼出しの時，次のふたつの方法で値—処理すべきデータ—を渡すことができる．

*独立行政法人　秋田工業高等専門学校　電気情報工学科

- 値そのものを渡す方法を**値渡し** (call by value) と言う．呼出側の実引数の値は，呼び出された関数の仮引数 (変数) にコピーされる．この方法では，呼出側の関数と呼び出された側の関数のそれぞれが，記憶領域—変数—を持つので，関数の独立性が高くなる．すなわち，呼び出された側で，呼出側の変数の値を変えることができない．

- アドレスを渡す方法を**参照渡し** (call by reference) と言う．呼出側の引数はアドレスである．呼び出された関数の仮引数のポインターにそのアドレスがコピーされる．この方法では，呼び出された側の関数で呼出側の関数のデータ領域の操作ができる．そのため，関数の独立性が失われる．すなわち，呼び出された側で，呼出側の変数の値を変えることができる．

これらの例をリスト 1 とリスト 2 に示す．いずれの場合も swap() 関数により値を交換しようとしている．値渡しであるリスト 1 の場合，呼び出された関数で呼出側の変数の値を変化させることができないので意図した通りに動作していない．

一方，リスト 2 では，呼び出された関数は呼出元の変数を操作している．呼び出された側の関数は呼出元の変数のアドレスを知っているから，呼出元の変数の操作ができる．

リスト 1: 値渡しを使った swap．意図した通りに動作しない．

```
1  #include <stdio.h>
2  void swap(int a, int b);
3
4  //-----------------------------------------
5  int main(void){
6    int hoge=1111, fuga=2222;
7
8    printf("hoge=%d\tfuga=%d\n",hoge,fuga);
9    swap(hoge, fuga);
10   printf("hoge=%d\tfuga=%d\n",hoge,fuga);
11
12   return 0;
13 }
14
15 //-----------------------------------------
16 void swap(int a, int b){
17   int temp;
18
19   temp=b;
20   b=a;
21   a=temp;
22 }
```

実行結果

```
hoge=1111       fuga=2222
hoge=1111       fuga=2222
```

リスト 2: 参照渡しを使った swap．意図した通りに動作している．

```
1  #include <stdio.h>
2  void swap(int *a, int *b);
3
```

```
4  //-----------------------------------------
5  int main(void){
6    int hoge=1111, fuga=2222;
7
8    printf("hoge=%d\tfuga=%d\n",hoge,fuga);
9    swap(&hoge, &fuga);
10   printf("hoge=%d\tfuga=%d\n",hoge,fuga);
11
12   return 0;
13 }
14
15 //-----------------------------------------
16 void swap(int *a, int *b){
17   int temp;
18
19   temp=*b;
20   *b=*a;
21   *a=temp;
22 }
```

実行結果

```
hoge=1111       fuga=2222
hoge=2222       fuga=1111
```

2.1.2 複数の計算結果を呼出元へ知らせる

参照渡しを使うと，グローバル変数を使わないで複数の計算結果を呼出元へ知らせることができる．リスト 3 の関数 cal() は，和と差，積，商を一度に計算し，参照渡しを使い 4 つの値を一度に呼出元へ知らせている．実際には，呼出し元から指定されたアドレスに，呼び出された関数 cal() が計算結果を書き込んでいるのである．

リスト 3: 参照渡しを使い，複数の計算結果を戻す．

```
1  #include <stdio.h>
2
3  void cal(int *wa, int *sa, int *seki, int *sho, int a, int b);
4
5  //-----------------------------------------------------------
6  int main(void){
7    int add, sub, mul, div;
8
9    cal(&add, &sub, &mul, &div, 33, 3);
10   printf("add = %d\n", add);
11   printf("sub = %d\n", sub);
12   printf("mul = %d\n", mul);
13   printf("div = %d\n", div);
14
15   return 0;
16 }
17
18 //-----------------------------------------------------------
19 void cal(int *wa, int *sa, int *seki, int *sho, int a, int b){
20
21   *wa = a+b;
```

```
22    *sa = a-b;
23    *seki = a*b;
24    *sho = a/b;
25  }
```

実行結果

```
add = 36
sub = 30
mul = 99
div = 11
```

2.2 配列

関数の引数に配列名を使う場合，参照渡しとなる．前回の講義で述べたように，配列名は配列の先頭アドレスを表すポインターである．すなわち，配列名という変数 (のようなもの) には，その配列の先頭アドレスが格納されている．その配列名—アドレスの値—を渡すのであるから，参照渡しである．

2.2.1 一次元配列

リスト 4 に配列を関数に渡す例を示す．この例を見て分かるように，配列を渡す場合は次のようにする．

- 呼出側の実引数は配列名のみを書く．配列のサイズを書いてはならない．配列名は配列の先頭アドレスが格納されたポインターなので，これにより呼び出された関数へアドレスを渡すことができる．
- 呼び出された関数の仮引数は，配列名にかぎかっこ `[]` をつける．
- 配列のサイズが呼び出された関数で必要であれば，別の引数を使う．

リスト 4 の関数 `reverse()` は，配列に格納されている順序を逆にする関数である．動作については，各自考えよ．

リスト 4: 一次元配列が関数の引数となる場合．

```
1  #include <stdio.h>
2  void reverse(int n, int a[]);
3
4  //-----------------------------------------
5  int main(void){
6    int hoge[3]={1,2,3};
7
8    printf("hoge=\t%d\t%d\t%d\n",hoge[0],hoge[1],hoge[2]);
9    reverse(3, hoge);
10   printf("hoge=\t%d\t%d\t%d\n",hoge[0],hoge[1],hoge[2]);
11
12   return 0;
13 }
14
15 //-----------------------------------------
16 void reverse(int n, int a[]){
17   int i, i_max, temp;
18
```

```
19    i_max=n/2;
20
21    for(i=0; i<i_max; i++){
22      temp=a[i];
23      a[i]=a[n-1-i];
24      a[n-1-i]=temp;
25    }
26  }
```

実行結果

```
hoge=   1       2       3
hoge=   3       2       1
```

2.2.2 多次元配列

多次元配列になると，仮引数の配列の記述方法に気を付ける必要がある．二次元以上になると，一番左側の配列のサイズの記述は不要であるが，左から二番目以降の記述が必要である．配列の先頭アドレスから，その添字に対応するデータのアドレスを計算するときに，配列のサイズのうち最も左側は，その計算に不要である．計算に不要なため，仮引数の配列の最も左側のサイズは書く必要がない．記述しても，コンパイラーはそれを無視する．

配列の先頭アドレスから添字に対応するデータのアドレスの計算方法は，先週の講義ノートを見よ．

リスト 5: 二次元配列が関数の引数となる場合．

```
1   #include <stdio.h>
2   void reverse(int m, int n, int a[][3]);
3
4   //-----------------------------------------
5   int main(void){
6     int hoge[2][3]={{11,12,13},{21,22,23}};
7
8     printf("%d\t%d\t%d\n",hoge[0][0],hoge[0][1],hoge[0][2]);
9     printf("%d\t%d\t%d\n",hoge[1][0],hoge[1][1],hoge[1][2]);
10    reverse(2, 3, hoge);
11    printf("\n");
12    printf("%d\t%d\t%d\n",hoge[0][0],hoge[0][1],hoge[0][2]);
13    printf("%d\t%d\t%d\n",hoge[1][0],hoge[1][1],hoge[1][2]);
14
15    return 0;
16  }
17
18  //-----------------------------------------
19  void reverse(int m, int n, int a[][3]){
20    int i, j, j_max, temp;
21
22    j_max=n/2;
23
24    for(i=0; i<m; i++){
25      for(j=0; j<j_max; j++){
26        temp = a[i][j];
27        a[i][j] = a[i][n-1-j];
28        a[i][n-1-j] = temp;
```

```
29      }
30    }
31  }
```

実行結果

```
    11      12      13
    21      22      23

    13      12      11
    23      22      21
```

2.3 文字列

文字列定数のアドレスをポインターに格納した場合，そのアドレスを呼び出す関数に渡すことができる．リスト 6 に示すように容易に文字列を渡すことができる．ポインターは便利だ．

もちろん，配列に格納された文字列を渡すこともできる．それは前節の整数が格納された配列と同じようにすればよい．

リスト 6: 文字列を渡す方法.

```
1  #include <stdio.h>
2
3  void my_print(char *str);   // プロトタイプ宣言
4  //-----------------------------------------------
5  int main(void){
6    char *s;
7
8    s="Akita National College of Technology";
9
10   my_print(s);
11
12   return 0;
13 }
14
15 //-----------------------------------------------
16 void my_print(char *str){
17
18   printf("ここは, %s です. \n", str);
19
20 }
```

実行結果

```
ここは, Akita National College of Technology です.
```

3　ポインターの他の使い方

ここで紹介したポインターの利用方法は，C 言語で使われる方法のうちのほんの一部分でしかない．2 年生の「情報処理応用」では，ポインターの使い方をさらに学習するので，楽しみにしてください．興味のある者は，適当な教科書を捜して調べると良いだろう．

ここでは学習しなかったが，次のようなポインターの使い方がある．

- 関数の戻り値へのポインター
- 構造体へのポインター
- 関数へのポインター
- ポインターのポインター
- ポインターの配列
- データ構造への応用としてのポインターの使い方

4　プログラム作成の練習

[**練習 1**] 3 つの引数があり，その順序を次のように変える関数を作成せよ．リスト 2 を参考に 3 つの変数に対応した関数を作れ— ということ．

$$\mathrm{a} \to \mathrm{b} \qquad \mathrm{b} \to \mathrm{c} \qquad \mathrm{c} \to \mathrm{a}$$

[**練習 2**] 前問の問題で引数を値渡しにすると，意図した通りにプログラムが動作しないことを確認せよ．

[**練習 3**] ひとつの角度を与えたら，$\cos\theta$ と $\sin\theta$，$\tan\theta$ の値を計算して，それらを呼出元へ知らせる関数を作成せよ．リスト 3 を参考にせよ．

[**練習 4**] 配列の合計を計算して，その合計を戻り値として返す関数を作成せよ．プログラムは次のとおりとする．

- メイン関数で，配列に $0, 2, 4, 6, 8, \cdots, 2000$ と偶数を格納する．
- メイン関数から，合計を計算する関数を呼び出す．引数はデータの数と配列とする．
- 合計は，関数の戻り値を使うこと．
- メイン関数で合計を表示する．

[**練習 5**] 適当な三次元配列を関数に渡して，表示させてみよ．

[**練習 6**] 適当な文字列を関数に渡すプログラムを作成せよ．

参考文献

[1] 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　基礎編　第 2 版. (株) オーム社, 2006.