プログラミング 1
Report#07

提出日：2009/07/16(木)

所属：工学部情報工学科
学籍番号：095736E
氏名：玉城　翔

課題１.第２回試験「問Ⅱ：ポインタとアドレス」より、各問題ごとにブロック文を用いて一つの解答確認プログラムとして作成し考察せよ。

(前半１)ソースプログラム：[tester1.c]

```
/*
  Program  : tester1.c
  Comment  : 問２の解答確認プログラム
*/

#include <stdio.h>

#define MAX  25

int main(){

  /*Q1 変数ｖのアドレスを求める式を示せ*/
  {
    int v;

    printf("\n");
    printf("[Q1 => &v = %p]",&v);

  }

  /*１次元配列ｍの０番目から始まる５番目の要素のアドレスを求める式を２つ示せ*/
  {
    int m[] = {};

    printf("\t");
    printf("[Q2 => &m[5] = %p ",&m[5]);
    printf(", m+5 = %p]\n",m+5);

  }

  /*１次元配列ｍの先頭アドレスを求める式を、２つ示せ*/
  {
    int m[] = {};

    printf("\n");
    printf("[Q3 => &m[0] = %p ",&m[0]);
    printf(", m = %p]\n",m);

  }

  /*２次元配列ｄの先頭アドレスを求める式を、３つ示せ*/
  {
    int d[2][2] = { 1, 2, 3, 4 };

    printf("\n");
    printf("[Q4 => &d[0] = %p  ",&d[0][0]);
    printf(", d[0] = %p ",d[0]);
    printf(", *d = %p]\n",*d);

  }

  /*次の文を実行した後の変数ａの値を示せ*/
  {
```

```
    int a = 2, b = 3, c = 5, *p, *q;

    p = &b;  q = &c;  a = *p + *q;

    printf("\n");
    printf("[Q5 => a = %d]",a);

}
```

[出力結果]

```

[Q1 => &v = 0xbffff77c]   [Q2 => &m[5] = 0xbffff790 , m+5 = 0xbffff790]

[Q3 => &m[0] = 0xbffff77c , m = 0xbffff77c]

[Q4 => &d[0] = 0xbffff76c  , d[0] = 0xbffff76c , *d = 0xbffff76c]

[Q5 => a = 8]       [Q6 => a = 5]
```

[考察]

２０ある問題を５問ずつに４つに分けて、解答確認プログラムを考察しようと思う。

[問題ごとの考察]

<Q1>
アドレスは変数に'&'を付けることにより求めることが出来る。結果もアドレスが出力されている。

<Q2>
アドレスなので１つは'&'を付けることが分かる。もう１つは配列なので'm'だけでもアドレスを出力出来るので、５番目だから+5 を付けると出力される。

<Q3>
Q2 とやっていることは一緒である。

<Q4>
これも Q2,3 とやっていることはほぼ一緒で、２次元配列なので'&'をつけること、'd[0]'のようにしてもアドレスが出力されること、もう１つは'd[0]'を置き換えたものと考えればよい。

<Q5>
ポインタの演算で、ようは[a = b + c]をやっている様なもので、5 + 3 を'a'に代入よって 8 が出力される。

(前半 2)ソースプログラム：[tester1.c]

```
  /*次の文を実行した後の変数 a の値を示せ*/
  {
    int a = 2, *p;

    p = &a;  *p = 5;

    printf("\t");
    printf("[Q6 => a = %d]\n",a);

  }

  /*
```

```
    次の文を実行した後の*p、*q、**q の値を示せ。
    但し、*(100) = 200 -> アドレス 100 の値 = 200
          *(200) = 300 -> アドレス 200 の値 = 300 とする。
*/
{
  int a = 200, b = 300, *p, **q, *q1;

  p = &a;  q1 = &a;

  printf("\n");
  printf("[Q7 => *p = %d ",*p);
  printf(", *q = %d ",*q);

  q1 = &b;  q = &q1;

  printf(", **q = %d]",**q);
}

/* 1 次元配列 m において、m[k]と*(m+k)はとのような値か述べよ*/
{
  int m[5] = { 1, 2, 3, 4, 5 }, k;

  k = 3;

  printf("\t");
  printf("[Q8 => m[k] = %d ",m[k]);
  printf(", *(m + k) = %d]\n",*(m + k));

}

/*
  整数型ポインタ変数 p において、p+2 は p の値を何バイト増加させた値か述べよ。但し、整数型データは 4 バイトとする。
*/
{
  int  *p;

  printf("\n");
  printf("[Q9 => p = %d ",p);
  printf(", p + 2 = %d]",p + 2);

}

/*
  次の文章は正しいか述べよ。
  a. 1 次元配列 m は、*m のようにポインタ変数と同じ書式で使用しても良い。
  b.ポインタ変数 p は、p[0]のように配列名と同じ書式で使用しても良い。
*/
{
  int *p, m[5] = { 1, 2, 3, 4, 5 };

  p = m;

  printf("\t");
  printf("[Q10 => m[0] = %d ",m[0]);
  printf(", *p = %d]\n",*p);
}
```

[出力結果]

```
[Q5 => a = 8]       [Q6 => a = 5]

```

```
3  [Q7 => *p = 200 , *q = 200 , **q = 300]      [Q8 => m[k] = 4 , *(m + k) = 4]
4
5  [Q9 => p = -1880951540 , p + 2 = -1880951532][Q10 => m[0] = 1 , *p = 1]
6
```

[問題ごとの考察]

<Q6>
ポインタにアドレスを入れた後に、ポインタが示す値の方に5を代入している。よって、ポインタが示している先は変数aであり、答えは5となる。

<Q7>
ポインタp,qはアドレスが共に一緒な為、'**q'は一度別のアドレスに入れることによって可能となる。

<Q8>
'm[k]'は配列のk番目の値を示し、*(m+k)はアドレスmからk番目の値を示しており、どちらとも同一となる。

<Q9>
アドレスの番地の変化を見れば分かるので、結果より8バイトということが分かる。

<Q10>
配列とポインタは互いに同じ書式で使用しても良い為、配列mとポインタpが互いに作用したときに結果が同じになれば、使用しても大丈夫ということになる。

(後半１)ソースプログラム：[tester1.c]

```
1   /*次の定義による、*m, *(m+3), *m+3, *m+*(m+3)の値を示せ。*/
2   {
3     static int m[5] = { 10, 20, 40, 50, 30 };
4
5     printf("\n");
6     printf("[Q11 => *m = %d ",*m);
7     printf(", *(m+3) = %d ",*(m+3));
8     printf(", *m+3 = %d ",*m+3);
9     printf(", *m+*(m+3) = %d]\n",*m+*(m+3));
10
11  }
12
13  /*
14    次の定義による、*d[2], *(d[2]+2), *d[2]+2, **d, *(*d+3), **d+6,
15    *(d[1]+2), **(d+2)の値を示せ。
16  */
17  {
18    static int d[][3] = {{ 1, 2, 3 }, { 5, 6, 7 }, { 4, 6, 8 }, { 9, 7, 5 }};
19
20    printf("\n");
21    printf("[Q12 => *d[2] = %d ",*d[2]);
22    printf(", *(d[2]+2) = %d ",*(d[2]+2));
23    printf(", *d[2]+2 = %d ",*d[2]+2);
24    printf(", **d = %d ",**d);
25    printf(", *(*d+3) = %d \n",*(*d+3));
26    printf(", **d+6 = %d ",**d+6);
27    printf(", *(d[1]+2) = %d ",*(d[1]+2));
28    printf(", **(d+2) = %d]\n",**(d+2));
29
30  }
31
32  /*次の文を実行した後のポインタ変数pの文字列を示せ。*/
```

```
  {
    char *str = "abcdefg", *p;

    p = str + 3;

    printf("\n");
    printf("[Q13 => *p = %s]",p);

  }

  /*次の文を実行した後の p, *p, *(p+2)の値を示せ。但し、&(*p)=100 とする。*/
  {
    char *p;

    p = "abc";

    printf("\t");
    printf("[Q14 => *p = %c ",*p);
    printf(", *(p+2) = %c ",*(p+2));

    &(*p)=100;

    printf(", p = %c]\n",p);

  }

  /*次の文を実行した後の *m, *p, *q の値を示せ。*/
  {
    static char m[] = "abcd";
    char *p, *q;

    p = &m[0];  q = m;

    printf("\n");
    printf("[Q15 => *m = %c ",*m);
    printf(", *p = %c ",*p);
    printf(", *q = %c]",*q);

  }
```

[出力結果]

```
[Q11 => *m = 10 , *(m+3) = 50 , *m+3 = 13 , *m+*(m+3) = 60]

[Q12 => *d[2] = 4 , *(d[2]+2) = 8 , *d[2]+2 = 6 , **d = 1 , *(*d+3) =
5 , **d+6 = 7 , *(d[1]+2) = 7 , **(d+2) = 4]

[Q13 => *p = defg]     [Q14 => *p = a , *(p+2) = c , p = 100]

[Q15 => *m = a , *p = a , *q = a] [Q16 =>  *p = c , *(m+2) = c , *m+2
= c]

```

[問題ごとの考察]

<Q11>
*m = m[0] = 10
*(m+3) = m[3] = 50
*m+3 = m[0]+3 = 10+3 = 13
m+(m+3) = m[0]+m[3] = 10+50 = 60 となる。

<Q12>
*d[2] = d[2][0] = 4
*(d[2]+2) = d[2][2] = 8
*d[2]+2 = d[2][0]+2 = 4+2 = 6
**d = d[0][0] = 1
*(*d+3) = d[1][0] = 5
**d+6 = d[0][0]+6 = 1+6 = 7
*(d[1]+2) = d[1][2] = 7
**(d+2) = d[2][0] = 4 となる。

<Q13>
文字列のアドレスを 3 つプラスしたアドレスをポインタに代入している為、文字列の 3 つ先の方から出力される。

<Q14>
アドレスは指定されている為 100 となる。ポインタは文字列の 1 番目とアドレスにプラス２をした値、３番目となる。
<Q15>
'*m'は m[0]と一緒な為、他のポインタはアドレスに m[0]のアドレスが代入されている為、結果値はどれも同一となっている。

(後半２)ソースプログラム：[tester1.c]

```
1   /*次の文を実行した後の *p, *(m+2), *m+2 の値を示せ。*/
2   {
3     static char m[] = "abcd";
4     char *p;
5
6     p = &m[2];
7
8     printf("\t");
9     printf("[Q16 =>  *p = %c ",*p);
10    printf(", *(m+2) = %c ",*(m+2));
11    printf(", *m+2 = %c]\n",*m+2);
12
13  }
14
15  /*次の文を実行した後のポインタ変数 p の文字列を示せ。*/
16  {
17    char *p, m[] = "abcd";
18
19    p = m;  *(p + 1) = 'x';
20
21    printf("\n");
22    printf("[Q17 => *p = %s]",p);
23
24  }
25
26  /*次の文を実行した後の変数 x の値を示せ。*/
27  {
28    int x;  char *p;
29
30    p = "abcd";
31
32    if(p == "abcd") x = 0;
33    else            x = 1;

    printf("\t");
    printf("[Q18 => x = %d]\n",x);

  }
```

```
34
35    /*次の文をポインタの代わりに、"int k;"を宣言し、配列を用いた文に書き換えよ。*
36  /
37    {
38      char m[MAX], *p;
39
40      for(p=m; *p; ++p) *p += 1;
41
42      printf("\n");
43      printf("[Q19 => *p = %d]",*p);
44
45    }
46
47    {
48      int k;  char m[MAX];
49
50      for(k=0; m[k]; k++) m[k] += 1;
51
52      printf("\t");
53      printf("[Q19 => m[k] = %d]\n",m[k]);
54
55    }
56
57    /*
58      次の定義による *q[2], q[3][2], *(q[2]+2), *(*(q+3)+2), **(q+1)の値を示せ。
59    */
60    {
61      static char *q[] = {"abcd", "12345", "ABCDEFG", "987"};
62
63      printf("\n");
64      printf("[Q20 => *q[2] = %c ",*q[2]);
65      printf(", q[3][2] = %c ",q[3][2]);
66      printf(", *(q[2]+2) = %c ",*(q[2]+2));
67      printf(", *(*(q+3)+2) = %c\n ",*(*(q+3)+2));
68      printf(", **(q+1) = %c]\n",**(q+1));
69
70    }
71  }
72
```

[出力結果]

```
1  [Q15 => *m = a , *p = a , *q = a]      [Q16 =>  *p = c , *(m+2) = c , *m+2 = c]
2
3  [Q17 => *p = axcd] [Q18 => x = 0]
4
5  [Q19 => *p = 0]     [Q19 => m[k] = 0]
6
7  [Q20 => *q[2] = A , q[3][2] = 7 , *(q[2]+2) = C , *(*(q+3)+2) = 7
   , **(q+1) = 1]
8
```

[問題ごとの考察]

<Q16>
'*p'にはm[2]のアドレスが代入されている為、'*(m+2)'はmの2番目である為共に値が同一となっている。もう一方のほうは、値にプラス2をしているが文字である為、値aから順番に数えて2番目のcとなり、同じとなった。

<Q17>
文字列のaから1番目がxに変更していることを示しており、bがxに代わった文字列が出力される。

<Q18>
ポインタには文字列 abcd が代入されており、if 文で文字列 abcd と同じなら 0 を、違うのなら 1 を出力することになっており、結果は 0 となる。

<Q19>
書き換えの問題であり、変更前と変更後の動作が同じである為結果は書き換えがきちんとなされていることになる。

<Q20>
*q[2] = q[2] = A
q[3][2] = 7 そのままの配列と見てみる
*(q[2]+2) = q[2][2] = C
((q+3)+2) = q[3][2] = 7
**(q+1) = q[1] = 1

課題２.構造体・共有体についても考察せよ。

(構造体)ソースプログラム：[struct1.c]

```
1  /*
2    Program  : struct1.c
3    Comment  : 構造体
4  */
5
6  #include <stdio.h>
7
8  /*構造体の型枠の宣言*/
9  struct seiseki {
10   int no;
11   char name[20];
12   double heikin;
13 };
14
15 int main(void)
16 {
17   int i;
18   struct seiseki seito[20] = {/*構造体の初期化*/
19     { 1, "SAKURAI", 78.6 },
20     { 2, "NAGANO", 57.3 },
21     { 3, "TAKESHITA", 66.4 },
22   };
23
24 /*構造体のポインタの宣言*/
25   struct seiseki *sp;
26
27   sp = seito;
28
29   for(i=0;i<3;i++){/*ポインタの値を変えずにデータを参照*/
30
31     printf("%d\n",(sp+i)->no);
32
33   }
34
35   for(i=0;i<3;i++){/*ポインタの値を変えてデータを参照*/

     printf("%s\n",sp->name);
     ++sp;

   }
```

```
36
37    return 0;

}
```

[出力結果]

```
1  1
2  2
3  3
4  SAKURAI
5  NAGANO
6  TAKESHITA
7
```

[考察]

複数のデータをまとめて扱うには配列を用いましたが、配列では同じ型のデータしかまとめて扱う事はできません。
実際にプログラムを組んでいると、異なる型のデータをまとめて扱いたい場合がしばしばあります。 たとえば、学生の成績を扱うときに、int型の学生番号と、char型配列の氏名と、double型の点数をまとめて扱えれば便利になります。
構造体には、そのような機能が実装されています。

構造体の機能

1.異なるデータをまとめて扱うことができる。

2.構造体をひとまとまりとして扱うことが出来、そのまとまりをまた別の構造体の一部として扱うことが出来る。

3.構造体を配列として見ると、ポインタとしても扱える。ポインタを参照する場合は ->(アロー演算子)を使用する。

4.関数に構造体を渡したり、関数から構造体を受け取ったりすることが可能。

今回の課題がポインタとアドレスだったので、上記のソースプログラムで構造体にポインタを使ったみた。

ポインタの値を変えずにデータを参照する方法と、ポインタの値を実際に変えていってデータを参照する方法をプログラムした。

(共有体)ソースプログラム：[union1.c]

```
1  /*
2    Program  : union1.c
3    Comment  : 共有体
4  */
5
6  #include <stdio.h>
7
8  /* 共有体の型枠の宣言 */
9  union smpl {
10   long l;
11   int i;
12   char c;
13 };
14
15 int main(void)
16 {
17   /* 共有体の宣言と初期化 */
```

```
  /* (long 型メンバ 1 への初期化) */
  union smpl dt = { 0x11111111 }; /* (1) */

  /* 共有体の参照 */
  printf("dt.l = 0x%lx\n", dt.l);
  printf("dt.i = 0x%x\n", dt.i);
  printf("dt.c = 0x%x\n", dt.c);

  dt.c = 0x22;/* (2) */
  printf("\ndt.l = 0x%lx\n", dt.l);
  printf("dt.i = 0x%x\n", dt.i);
  printf("dt.c = 0x%x\n", dt.c);

  dt.i = 0x3333;/* (3) */
  printf("\ndt.l = 0x%lx\n", dt.l);
  printf("dt.i = 0x%x\n", dt.i);
  printf("dt.c = 0x%x\n", dt.c);

  dt.l = 0x44444444;/* (4) */
  printf("\ndt.l = 0x%lx\n", dt.l);
  printf("dt.i = 0x%x\n", dt.i);
  printf("dt.c = 0x%x\n", dt.c);

  return 0;
}

```

[出力結果]

```
dt.l = 0x11111111
dt.i = 0x11111111
dt.c = 0x11

dt.l = 0x11111122
dt.i = 0x11111122
dt.c = 0x22

dt.l = 0x3333
dt.i = 0x3333
dt.c = 0x33

dt.l = 0x44444444
dt.i = 0x44444444
dt.c = 0x44
```

[考察]

共有体は、宣言の「struct」が「union」になるだけで、それ以外は宣言の仕方や使い方などは構造体と全く同じです。 しかし、共有体は各メンバがすべて同じアドレスから割り振られている点が構造体とは異なります。

共有体の機能

1.基本は構造体とほとんど変わりはしない。

2.各メンバがすべて同じ先頭アドレスから割り振られている為、値が変わった時等に他の値にも影響を与えかねない。

参考サイト

初心者のためのポイント学習 C 言語
http://www9.plala.or.jp/sgwr-t/index.html

e0857ポチギ13
http://www.ie.u-ryukyu.ac.jp/~e085713/home/report/syorui/c_rep6.pdf