1. 第2回試験の解答プログラム

1.1 ソースコード : answer.c

```
001  /*
002    Program : answer.c
003    Comment : pointer and address
004  */
005
006  #include<stdio.h>
007
008  #define MAX 5
009
010  int main(){
011
012      puts("Q01 --------------------------------");
013      {
014          int v;
015          printf("&v = %x\n\n",&v);
016      }
017
018      puts("Q02 --------------------------------");
019      {
020          int m[5];
021          printf("&m[5] = %x\t",&m[5]);
022          printf("m+5 = %x\n\n",m+5);
023      }
024
025      puts("Q03 --------------------------------");
026      {
027          int m[5];
028          printf("&m[0] = %x\t",&m[0]);
029          printf("  m = %x\n\n",m);
030      }
031
032      puts("Q04 --------------------------------");
033      {
034          int d[2][3] = {{1,2,3},{4,5,6}};
035          printf("&d[0][0] = %x\n",&d[0][0]);
036          printf("    d[0] = %x\n",d[0]);
037          printf("      *d = %x\n\n",*d);
038      }
039
040      puts("Q05 --------------------------------");
041      {
042          int a=2, b=3, c=5, *p, *q;
043          p = &b;  q = &c;
044          a = *p + *q;
045         printf("a = %d\n\n",a);
046      }
047
048      puts("Q06 --------------------------------");
049      {
050          int a=2, *p;
051          p = &a;  *p = 5;
052          printf("a = %d\n\n",a);
053      }
054
```

```
055     puts("Q07 --------------------------------");
056     {
057         int a=200, b=300, *p, **q, *i;
058         p = &a;        i = &b;          q = &i;
059
060         printf("*p = %d\n",*p);
061         printf("*q = %x(= Instead of address 200)\n",*q);
062         printf("**q = %d\n\n",**q);
063     }
064
065     puts("Q08 --------------------------------");
066     {
067         int k=3, m[5] = {12,3,5,25,30};
068         printf("m[k] = %d\t",m[k]);
069         printf(" *(m+k) = %d\n\n",*(m+k));
070     }
071
072     puts("Q09 --------------------------------");
073     {
074         int *p;
075         printf("p = %x\t",p);
076         printf("p+2 = %x\n\n",p+2);
077     }
078
079     puts("Q10 --------------------------------");
080     {
081         int m[5] = {10,26,3,9,30};
082         char *p;
083
084         printf("m[0] = %d\t\t",m[0]);
085         printf(" *m = %d\n",*m);
086
087         p = "ABCDE";
088         printf("*p = %c\t\t",*p);
089         printf("   p[0] = %c\n\n",p[0]);
090     }
091
092     puts("Q11 --------------------------------");
093     {
094         static int m[5] = {10,20,40,50,30};
095
096         printf("       *m = %d\n",*m);
097         printf("   *(m+3) = %d\n",*(m+3));
098         printf("     *m+3 = %d\n",*m+3);
099         printf("*m+*(m+3) = %d\n\n",*m+*(m+3));
100     }
101
102     puts("Q12 --------------------------------");
103     {
104         static int d[][3] = {{1,2,3},{5,6,7},{4,6,8},{9,7,5}};
105
106         printf("    *d[2] = %d\n",*d[2]);
107         printf("*(d[2]+2) = %d\n",*(d[2]+2));
108         printf("  *d[2]+2 = %d\n",*d[2]+2);
109         printf("      **d = %d\n",**d);
110         printf("  *(*d+3) = %d\n",*(*d+3));
111         printf("    **d+6 = %d\n",**d+6);
112         printf("*(d[1]+2) = %d\n",*(d[1]+2));
113         printf("  **(d+2) = %d\n\n",**(d+2));
114     }
```

```
115
116     puts("Q13 --------------------------------");
117     {
118         char *str = "abcdefg", *p;
119         p = str + 3;
120
121         printf("*p = %s\n\n",p);
122     }
123
124     puts("Q14 --------------------------------");
125     {
126         char *p;
127         p = "abc";
128
129         printf("     p = %x\n",p);
130         printf("    *p = %c\n",*p);
131         printf("*(p+2) = %c\n\n",*(p+2));
132     }
133
134     puts("Q15 --------------------------------");
135     {
136         static char m[] = "abcd";
137         char *p, *q;
138         p = &m[0];     q = m;
139
140         printf("*m = %c\t\t",*m);
141         printf("*p = %c\t\t",*p);
142         printf("*q = %c\n\n",*q);
143     }
144
145     puts("Q16 --------------------------------");
146     {
147         static char m[] = "abcd";
148         char *p;
149         p = &m[2];
150
151         printf("*p = %c\t",*p);
152         printf("*(m+2) = %c\t",*(m+2));
153         printf("  *m+2 = %c\n\n",*m+2);
154     }
155
156     puts("Q17 --------------------------------");
157     {
158         char *p, m[] = "abcd";
159         p = m;         *(p + 1) = 'x';
160
161         printf ("*p = %s\n\n",p);
162     }
163
164     puts("Q18 --------------------------------");
165     {
166         int x;
167         char *p; p = "abcd";
168         if(p == "abcd")      x = 0;
169         else                 x = 1;
170
171         printf("x = %d\n\n",x);
172     }
173
```

```
174     puts("Q19 --------------------------------");
175     {
176         char m[MAX] = {5,6,7,8,9}, *p;
177         for(p=m; *p; ++p)    *p += 1;
178
179         int i=0;
180         for(p=m; i < MAX; p++,i++)      printf("p+%d =%d ",i,*p);
181         puts("");
182     }
183     {
184         int k;
185         char m[MAX] = {5,6,7,8,9};
186         for(k=0; m[k]; k++)             m[k] += 1;
187         for(k=0; k < MAX; k++)          printf("m[%d]=%d ",k,m[k]);
188         printf("\n\n");
189     }
190
191     puts("Q20 --------------------------------");
192     {
193         static char *q[] = {"abcd","12345","ABCDEFG","987"};
194
195         printf("     *q[2] = %c\n",*q[2]);
196         printf("   q[3][2] = %c\n",q[3][2]);
197         printf("  *(q[2]+2 = %c\n",*(q[2]+2));
198         printf("*(*(q+3)+2 = %c\n",*(*(q+3)+2));
199         printf("   **(q+1) = %c\n\n",**(q+1));
200     }
201 }
```

1.2 실행결과 : answer.c

```
Q01 --------------------------------
&v = bffffa18

Q02 --------------------------------
&m[5] = bffffa18 m+5 = bffffa18

Q03 --------------------------------
&m[0] = bffff9f0   m = bffff9f0

Q04 --------------------------------
&d[0][0] = bffff9d8
    d[0] = bffff9d8
      *d = bffff9d8

Q05 --------------------------------
a = 8

Q06 --------------------------------
a = 5

Q07 --------------------------------
*p = 200
*q = bffff9c4(= Instead of address 200)
**q = 300
```

```
Q08 --------------------------------
m[k] = 25    *(m+k) = 25

Q09 --------------------------------
p = 8fe004c0      p+2 = 8fe004c8

Q10 --------------------------------
m[0] = 10          *m = 10
*p = A               p[0] = A

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 = 2e39
    *p = a
*(p+2) = c

Q15 --------------------------------
*m = a          *p = a      *q = a

Q16 --------------------------------
*p = c      *(m+2) = c    *m+2 = c

Q17 --------------------------------
*p = axcd

Q18 --------------------------------
x = 0

Q19 --------------------------------
p+0 =6 p+1 =7 p+2 =8 p+3 =9 p+4 =10
m[0]=6 m[1]=7 m[2]=8 m[3]=9 m[4]=10

Q20 --------------------------------
      *q[2] = A
    q[3][2] = 7
  *(q[2]+2) = C
*(*(q+3)+2) = 7
    **(q+1) = 1
```

1.3 考　察 : answer.c

Q01
- 変数 v のアドレスを求めるには、&(アンド)を変数の前に付けるので、答えは、&v

Q02
- 1次元配列の5番目の要素のアドレスを求める式は、2つある。
- 単純に、m[]を変数と考え、m[]の前に&(アンド)を付けるので、答えは、&m[5]
- m は配列の先頭アドレスを示すので、配列の5番目を表すには、答えは、m + 5

Q03
- m[]を変数と考え、m[]の前に&(アンド)を付ければいいので、答えは、&m[0]
- m は配列の先頭アドレスを示すので、答えは、m

Q04
- 2次元配列も同様に考え、m[][]の前に&(アンド)を付ければいいので、答えは、&d[0][0]
- d[0]と *d は同じ。両方とも先頭アドレスを表すので、答えは、d[0] と *d

Q05
- p = &b で変数b のアドレスをアドレス変数p に代入。
- q = &c で変数c のアドレスをアドレス変数q に代入。
- *p + *q は、b + c と同じなので、5 を a に代入なので、答えは、a = 8

Q06
- 変数a のアドレスをポインタp に渡し、*p に 5 を代入なので、答えは、a = 5

Q07
- ポインタ p は、変数 a のアドレスを持っているので、*p の値は、*p = 200
- ① ポインタ i に、変数 b のアドレスを渡す。
  ② ポインタ i のアドレスをポインタ q に渡す。
  したがって、*q の値は変数b のアドレスなので、*q = bffff6f4（アドレス200の代わり）
- *qが、変数b のアドレスを持っているので、**qは変数b と同じ値になる。
  したがって、**q の値は変数b の値なので、**q = 300

Q08
- int k = 3, m[5] = {12,3,5,25,30}; と定義した。
- m[k] , *(m+k) をそれぞれ printf関数で出力すると、m[k] = *(m+k) = 25 であった。
  したがって、m[k] , *(m+k) は同値である。

Q09
- int *p; ポインタ変数p を宣言し、ポインタ変数p と p+2 のアドレスを16進数で出力した。
- アドレスを出力してみるとそれぞれ、p = 8fe004c0 , p+2 = 8fe004c8 となった。
  したがって、アドレスの値の差が 8 なので、p+2 はp の値を8バイト増加させた値である。

Q10
- a について、int m[5] = {10,26,3,9,30}; と定義した。
- m[0] , *m の値を出力すると、m[0] = *m = 10 となった。
  したがって、「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 について、char *p; p = "ABCDE"; と定義した。
- *p , p[0] の値を出力すると、*p = p[0] = A となった。
  したがって、「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。」

Q11
- static int m[5] = {10,20,40,50,30}; と定義した。

- *m は、m[0]と同じなので、*m の値は、*m = 10 となる。

- *(m+3) は、m[3]と同じ意味なので、*(m+3)の値は、*(m+3) = 50 となる。

- *m+3 は、演算子の優先順が + より * が高いので、m[0]の値に +3 することになる。
  したがって、*m+3の値は、*m+3 = 13 となる。

- *m + *(m+3)は、*m = m[0] = 10 , *(m+3) = m[3] = 50 なので、*m + *(m+3) = 60 となる。

Q12
- static int d[][3] = {{1,2,3},{5,6,7},{4,6,8},{9,7,5}}; と定義した。

d[][]	[0]	[1]	[2]
[0]	1	2	3
[1]	5	6	7
[2]	4	6	8
[3]	9	7	5

- *d[2] は、d[2][0]と同じ意味なので、*d[2]の値は、*d[2] = 4 となる。

- *(d[2]+2) は、d[2][2]と同じなので、*(d[2]+2)の値は、*(d[2]+2) = 8 となる。

- *d[2]+2 は、演算子の優先順が + より * が高いので、*d[2]の値に +2 することになる。
  したがって、*d[2]+2 の値は、*d[2]+2 = 4 + 2 = 6 となる。

- **d は、2次元配列の先頭 d[0][0] を表すので、**d = 1 となる。

- *(*d+3) は、d[0][3]を表すが、定義されていない。d[0][0]から3つ先のデータと考えると、
  d[1][0]となり、実行結果と同様の値になるので、*(*d+3) = 5 となる。

- **d+6 は、演算子の優先順位を考えると、**d の値に +6 すれば良いことがわかるので、
  d = d[0][0] = 1 よって、d+6 = 1 + 6 = 7 となる。

- *(d[1]+2) は、d[1][7]と同じなので、*(d[1]+2) = 7 となる。

- **(d+2) は、d[2][0]と同じ意味なので、**(d+2) = 4 となる。

Q13
- char *str = "abcdefg", *p; と定義。

- ポインタ変数p に str+3(先頭アドレス+3) を代入しているので、*p = defg となる。

Q14
- char *p; p = "abc"; と定義。

- printf関数でポインタ変数p のアドレスを16進数で出力、p = 2e39 となった。

- *p は、*p = aとなり、*(p+2)は、*(p+2) = c となる。

Q15
- static char m[] = "abcd"; char *p, *q; と定義。

- 最初のprintf関数では、*m(=配列の最初の要素)を出力、m = a となる。

- p = &m[0]; 配列の先頭アドレスをポインタ変数p に代入。したがって、*p = aとなる。

- q = m; これも同様に配列の先頭アドレスをポインタ変数q に代入しているので、*q = a となる。

Q16
- static char m[] = "abcd"; char *p; と定義。

- p = &m[2]; ポインタ変数p に配列m[2]のアドレスを代入、*p の値を出力なので、*p = c となる。

- *(m+2) は、m[2]と同じなので、*(m+2) = c となる。

- *m+2 は、演算子の優先順位を考えると、*m の値に +2 なので、*m+2 = a + 2 = c となる。

Q17
- char *p, m[] = "abcd"; と定義。

- ① p = m; で配列m の先頭アドレスをポインタ変数p に代入。
  ② *(p + 1) = m[1] の値 'b'を 'x'に置換する。
  したがって、文字列p の値は、*p = axcdとなる。

Q18
- int x; char *p; p = "abcd"; と定義。

- if文。文字列p の値が、"abcd"と等しいならば、x に 0を代入、そうでなければ、x に 1を代入。
  したがって、変数x の値は、x = 0 となる。

Q19
- 問題のプログラム(177-178行目)は、配列の値を1ずつ増やすプログラム。

- int k; を宣言し、配列を用いて、同様のプログラムを作る。

- 問題では、for文の初期値を p = m , 終値を *p , 増分値を ++p なので、k = 0を初期値として、
  m[k] まで、k を 1ずつ増やす。

- 確認の為、各配列を出力してみる。配列をm[MAX] = {5,6,7,8,9}; と定義する。
  p+0 =6 p+1 =7 p+2 =8 p+3 =9 p+4 =10
  m[0]=6 m[1]=7 m[2]=8 m[3]=9 m[4]=10
  実行結果より、問題のプログラムと、int型変数k と配列を用いたプログラムが同義である。

Q20
- static char *q[] = {"abcd","12345","ABCDEFG","987"}; と定義する。

*q[]	[0]	[1]	[2]	[3]	[4]	[5]	[6]
[0]	a	b	c	d			
[1]	1	2	3	4	5		
[2]	A	B	C	D	E	F	G
[3]	9	8	7				

- *q[2] は、q[2][0]なので、*q[2] = A となる。

- q[3][2] は、上の表よりq[3][2] = 7 となる。

- *(q[2]+2) は、q[2][2] と同じなので、*(q[2]+2) = C となる。

- *(*(q+3)+2) は、q[3][2] となるので、*(*(q + 3)) = 7 となる。

- **(q+1) は、q[1][0] と同じなので、**(q+1) = 1 となる。

2. 構造体について

2.1 構造体の使い方

(1) 構造体とは？

- 複数の異なる型（int型 , char型 , short型 etc..）をまとめたもの。

(2) 構造体の書式

- あらかじめ次のような書式で、構造体の「型枠」を宣言しておく。

```
struct 構造体名 {
        データ型1     変数名1 ;
        データ型2     変数名2 ;
};
```

- { }; 内にいくつでも異なるデータ型（int,char,short, etc..）を宣言することが出来る。
- 構造体名のことを”タグ名”、変数名のことを”メンバ”という。
- 関数外で宣言した場合は、グローバル変数として、関数内で宣言した場合は、
  ローカル変数として、他の変数と同様に使うことが出来る。

(3) 構造体の宣言

- (2) で定義した「型枠」を使って、次のような書式で、データを宣言する。

```
struct タグ名   変数名 ;
```

- 複数の構造体をまとめて「構造体配列」としても宣言できる。

```
sturct タグ名   変数名[] ;
```

(4) 構造体のイメージ

(5) その他の使い方

a. 構造体の初期化

- 配列と同じ様に、各メンバの値をカンマで区切って記述。

b. 構造体の参照

- 構造体の各メンバを参照するには、「構造体.メンバ名」のように、ピリオドを用いる。

c. 構造体の代入

- 同じ型を持った構造体には、普通の変数と同様に代入が出来る。

```
struct july, june = { 25, "English Test", 3 };

july = june;  // juneの持つ値全てをjulyに代入。
```

2.2 構造体のサンプルプログラム : struct.c

```
01  /*
02    Program    : struct.c
03    Comment    : 構造体による閏年判断
04  */
05
06  #include <stdio.h>
07
08  struct smp{
09    int   day;
10    int   month;
11    int   year;
12  }date,*pdate;
13
14  isleap1(struct smp d){
15    int r4,r100,r400;
16
17    r4   = d.year % 4;
18    r100 = d.year % 100;
19    r400 = d.year % 400;
20
21    return ( ((r4 == 0) && (r100 != 0)) || (r400 == 0) );
22  }
23
24  isleap2(struct smp *d){
25    int r4,r100,r400;
26
27    r4   = d->year % 4;
28    r100 = d->year % 100;
29    r400 = d->year % 400;
30
31    return ( ((r4 == 0) && (r100 != 0)) || (r400 == 0) );
32  }
33
34  main(){
35    date.day   = 24;
36    date.month = 2;
37    date.year  = 1900;
38
```

```
39    printf("%4d年は閏年で%s\n",date.year,
40          (isleap1(date) != 0)? "す。":"ない。");
41    printf("%x %x %x\n",&date.day,&date.month,&date.year);
42
43    printf("%4d年は閏年で%s\n",date.year,
44          (isleap2(&date) != 0)? "す。":"ない。");
45    printf("%x %x %x\n",&date.day,&date.month,&date.year);
46  }
```

2.3 実行結果 : struct.c

```
new-host:Report7 kouta$ ./struct
1900年は閏年でない。
2030 2034 2038
1900年は閏年でない。
2030 2034 2038
```

2.4 考 察 : sturuct.c

- このプログラムは閏年かどうか判断するプログラムである。
- 「4で割り切れて、100で割り切れない年」または「400で割り切れる年」は、閏年である。
- プログラム中の isleap1 と isleap2 は同様な処理をしているが、isleap1関数は、構造体の要素を直接参照していて、isleap2 関数は、構造体ポインタを参照している。
- isleap1 関数は、ドット演算子(.)を使用している。「2.5 (5) b」(P10) と同じ使い方。
- isleap2 関数は、アロー演算子(->)を使用している。アロー演算子は、構造体のポインタを参照するために使われる。ポインタの使い方は、配列のポインタと同様。

3. 共用体について

3.1 共用体の使い方

(1) 共用体とは？

- 同一のデータ領域を複数のデータ型で共用(共有)するようにしたもの。

(2) 共用体の書式・宣言

- 構造体の「struct」が「union」になるだけで、書式、宣言は構造体と同様なので省略。

3.2 共用体のサンプルプログラム : union.c

```
01  /*
02    Program    : union.c
03    Comment    : 共用体
04  */
05
06  #include <stdio.h>
07
08  union smp{
09    char  b08;
10    short b16;
11    int   b32;
12  };
13
14  main(){
15    union smp var;
16
17    var.b08=2;
18    puts("----------------");
19    puts("var.b08=2");
20    printf("var.b08=0x      %02x=%d\n",var.b08,var.b08);
21    printf("var.b16=0x    %04x=%d\n",var.b16,var.b16);
22    printf("var.b32=0x%08x=%d\n",var.b32,var.b32);
23    printf("var ADDRESS\n");
24    printf("var.b08=0x%x +=%%0x%x\n",&var.b08,&var.b08+1);
25    printf("var.b16=0x%x +=%%0x%x\n",&var.b16,&var.b16+1);
26    printf("var.b32=0x%x +=%%0x%x\n",&var.b32,&var.b32+1);
27
28    var.b16=260;
29    puts("----------------");
30    puts("var.b16=260");
31    printf("var.b08=0x      %02x=%d\n",var.b08,var.b08);
32    printf("var.b16=0x    %04x=%d\n",var.b16,var.b16);
33    printf("var.b32=0x%08x=%d\n",var.b32,var.b32);
34
35    var.b32=66000;
36    puts("----------------");
37    puts("var.b32=66000");
38    printf("var.b08=0x      %02x=%d\n",var.b08,var.b08);
39    printf("var.b16=0x    %04x=%d\n",var.b16,var.b16);
40    printf("var.b32=0x%08x=%d\n",var.b32,var.b32);
41  }
```

3.3 実行結果 : union.c

```
new-host:Report7 kouta$ ./union
---------------
var.b08=2
var.b08=0x      02=2
var.b16=0x    0002=2
var.b32=0x00000002=2
var ADDRESS
var.b08=0xbffffa4c +=%0xbffffa4d
var.b16=0xbffffa4c +=%0xbffffa4e
var.b32=0xbffffa4c +=%0xbffffa50
---------------
var.b16=260
var.b08=0x      04=4
var.b16=0x    0104=260
var.b32=0x00000104=260
---------------
var.b32=66000
var.b08=0xffffffd0=-48
var.b16=0x    01d0=464
var.b32=0x000101d0=66000
```

3.4 考　察 : union.c

- PC環境の影響なのか、谷口先生と同様の結果が得られなかった。
- 自分のPC環境では、char型変数b08 に2 を代入すると、それぞれの最上位ビットから格納されず、最下位ビットから、順に格納されている。
  したがって、どの型の変数の値も2 となっていると考えられる。
- short型変数b16 に260を代入すると、char型は 8bit なので、4 という値なった。
  int型は、short型よりビット数が多いので、short型同様 260 となった。
- int型変数b32 に6600 を代入すると、char型は、 $(1011\,0000)_2$ となるので、-48 になった。
  short型は、下位16bitだけ int型と同じなので、464 となった。
- char型は、0xbffffa4c から 0xbffffad までの 1バイト。
- short型は、0xbffffa4c から 0xbffffa4e までの 2バイト。
- int型は、0xbffffa4c から 0xbffffa50 までの 4バイト。
- 以上の様に、アドレスは問題なく確保されていたので、値を最上位ビットから格納するか、最下位ビットから格納するかは、PC環境の影響だと思われる。

4. 感　想

今回のレポートは、試験の解答確認プログラムを作るということで、プログラムの行数も今までで一番長くなった。また問題数も多く、一つ一つ説明するのは苦労した。

後半の構造体と共用体については、解答確認プログラムの作成にかなりの時間を費やしたので、しっかりまとめることができなかった。でも、ある程度、構造体と共用体の違いや特徴が分かったので、プログラムによって、使い分けれるようになりたいと思いました。
あと、次回のレポートがプログラミングの最後のレポートらしいので、しっかり理解してまとめたいと思います。

5. 参考文献

- C実践プログラミング 第3版 谷口 功(訳)　望月 康司(監訳)　Steve Oualline(著)
- 初心者のためのポイント学習C言語　　http://www9.plala.or.jp/sgwr-t/
- ProgI/2009　　http://www.osn.u-ryukyu.ac.jp/lecture/wiki/index.php