[1] 課題内容

プログラミング（C言語）第3回試験の[Ⅰ]演算式と[Ⅱ]ポインタとアドレスに関する問題について、その解答を確認するプログラムを複数の関数を用いた、解答確認プログラムを作成し、考察せよ。可能ならば複数の関数を用いた1つのプログラムとして作成せよ。

[2] 演算式プログラム

a.プログラム

```
--------------------------------------------------------------------------------
/****************************************
  Program    : answer.c
  Student-ID : 045713C
  Author     : OSHIRO,Kazuya
  Date       : 04/06/21
  Comment    : 演算回答プログラム
****************************************/

#include <stdio.h>

#define NUM1 14                 /* NUM1を14に定義する */
#define NUM2  3                 /* NUM2を 3に定義する */
#define MAX  30                 /* MAX を30に定義する */

struct sample{                  /* 構造体sampleを宣言する */
  int  id;
  char name[MAX];
  int  age;
};

main()
{
  int    a,b,c,d,e,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y; /* 整数型の変数宣言 */
  float  f,g;                                          /* 浮動小数の変数宣言 */
  int    test,*answer;                                 /* 整数型の変数とポインタを宣言
*/
  char   work[MAX], *dest, *src;                /* char型の配列とポインタを宣言
*/
  static char cbuf[] = "var test program";             /* 静的変数の配列 */
  static int  ibuf[] = {1,2,4,7,9,11};                 /* 静的変数の配列 */

  a = NUM1;                     /* NUM1(14) */
  b = NUM2;                     /* NUM2( 3) */
  c = a / b;                    /* aをbで割った商 */
  d = a % b;                    /* aをbで割ったときの余り */
  e = (a+b)*(a-b);              /* ()内の計算を先に行いそれをかける */
  f = 8.0 / 3.0;         /* 浮動小数で表すため2.666667となる */
  g = 8 / 3;
  /* 浮動小数で宣言されているが数字が整数であるため処理された値は2.00000となる。 */
  h = a - a%b;                  /* h = a - d とおなじ計算。*/
  i = a > b;                    /* a=14,b=3よりa > bは真(0以外) */
  j = a < b;                    /* a < bは偽(0) */
  k = i||!j;
```

```
l = !i||j;
/* !は否定の意。kの場合i、もしくはjでない値が入る。lは逆でiでない、あるいはjの値 */

m = a | b;
n = a & b;
/* ビット演算子|は二進数で比べたときどちらか一方が1のときあるいは互いが1のとき */
/* 1をとるのでmの場合、aは二進数で1110、bは0011であるため比較して1111となる。 */
/* この値は10進数で15である。ビット演算子&はどちらも1のとき1をとる。これより */
/* nの値は0010。これは10進数で2となる。 */

o = b << 2;
p = a >> 1+b;
/* 指定されたビット数だけデータを動かす。oの場合は0011を左に2ビット */
/* pの場合は1110を1+b、つまり4ビット右に動かす。なおはみ出したビットは削除 */
/* されるためpの値は0000となり値は0となる。 */

q = sizeof(work);
/* 引数のサイズをバイト数で返すchar型のため30となる */

for (r=0, src=cbuf, dest=work; *src!=NULL; r++, src++, dest++){
     *dest = *src;
}
/* rを0に初期化、srcにcbufの初期アドレス(&cbuf[0])を代入 */
/* destにはworkの初期アドレスを代入。 */
/* ポインタsrcのさす要素の値がNULL文字になるまでr,src,destに1を足す。 */
/* *srcがNULLになるのはcbufの要素の最後。それまでにrは16足されることになる */
/* *dest=*srcこれはsrcのさす値、つまり文字列の要素でありこのプログラムを通して */
/* work[]にcbuf[]をコピーするような形になっている。 */

s = src - cbuf;
/* 前のfor文によりsrcはNULL文字のアドレスを指しており、cbufは初期アドレスを */
/* 指している。この二つの差は前のfor文より16であることがわかる。 */

t = cbuf[a] - cbuf[b];

u = sizeof(struct sample);
/* int型(4バイト)が2つ、char型(1バイト)が全体で30なのだが解は38でなく40と出る。*/
/* これはint型が4の倍数でしかバイトをとれないことからおこるズレである。 */

v = ibuf[2] - ibuf[4];
w = cbuf[5] - cbuf[8];
x = 's' - 'a';
src = cbuf;
y = *(src+4) - 'b';
/* t,v,w,x,yは全部、文字の計算となっているASCIIコードの10進表示で演算したものが */
/* それぞれの解となる。ちなみに*(src+4)はcbuf[4]と同じ意味である。 */

for(test='a',answer=&a;test<='e';test++,answer++){
  printf("%c = %3d",test,*answer);
  if((test-1)%5 == 0)
    printf("¥n");
  else
    printf("¥t¥t");
  }

printf("f =   %f¥t",f);
printf("g =   %f¥n",g);

for(test='h',answer=&h;test<='y';test++,answer++){
```

```
    printf("%c = %3d",test,*answer);
    if((test-3)%5 == 0)
      printf("¥n");
    else
      printf("¥t¥t");
    }
  printf("¥n");
}
```

---

b. 実行結果

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
[nw0413:~/prog1/report/#5] j04013% ./answer
a =  14         b =   3         c =   4         d =   2         e = 187
f =   2.666667  g =   2.000000
h =  12         i =   1         j =   0         k =   1         l =   0
m =  15         n =   2         o =  12         p =   0         q =  30
r =  16         s =  16         t =  65         u =  40         v =  -5
w =  69         x =  18         y =  18
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

c. 考察

＊＊＊今回使われた演算子について＊＊＊

`+` ------ **加算演算子。足し算する。**

`-` ------ **原産演算子。引き算する。**

`/` ------ **除算演算子。割り算の商を表す。**

`%` ------ **剰余演算子。割り算をしたときの余りを表す。**

`*` ------ **蒸散演算子。かけ算の積を表す。**

`!` ------ **否定演算子。!aと書いたときaではないという意味になる。**

`<` ------ **関係演算子。小なりの意。**
左辺が右辺より小さいとき真(0以外)を表し、そうでないとき偽(0)を返す。

`>` ------ **関係演算子。大なりの意。**
右辺が左辺より小さいとき真(0以外)を表し、そうでないとき偽(0)を返す。

`|` ------ **ビット演算子(OR)。**
二進数にして互いを比較し、どちらか1つまたは両方のビット　　が1なら、結果を1とする。

`&` ------ **ビット演算子(AND)。**
二進数にして互いを比較し、どちらも1なら、結果を1とする。

`<<`------ **左シフト演算子。**
指定されたビット数だけデータを左に動かす。なお、左側にはみ出したビットは削除され、右側の開いたビットには0が詰められる。

`>>`------ **右シフト演算子。**
左シフト演算とは逆に指定されたビット数だけ右に動かす。右側にはみ出したビットは削除、左の開いたビットには0が詰められる。

`sizeof()` --- **sizeof演算子。引数のサイズをバイト数で返す演算。**

＊＊＊構造体について＊＊＊
構造体とは異なるデータ型(例えば整数型と文字列型のようなもの)をグループとして格納するものである。配列と違い、構造体では一つ一つの要素に名前を付けてそれぞれを個別のデータ型にできる。

構造体定義の一般的な形式

```
struct structure-name {
    field-type field-name;
    field-type field-name;
     ・・・
} variable-name;
```

今回の場合、variable-nameの方は省略されており、つまりは構造体型の定義をし、変数は定義せれていない。

[3] ポインタ,アドレスプログラム

a. プログラム

```
-------------------------------------------------------------------------------
/************************************************
  Program    : answer2.c
  Student-ID : 045713C
  Author     : OSHIRO,Kazuya
  Date       : 04/06/24
  Comment    : ポインタ・アドレスの回答確認プログラム
************************************************/

#include <stdio.h>
void q1 (void),q2 (void),q3 (void),q4 (void),q5 (void);
void q6 (void),q8 (void),q9 (void);
void q11(void),q12(void),q13(void),q14(void),q15(void);
void q16(void),q17(void),q18(void),q20(void);

main()
{
  printf("Q1-- "); q1();
  printf("Q2-- "); q2();
  printf("Q3-- "); q3();
  printf("Q4-- "); q4();
  printf("Q5-- "); q5();
  printf("Q6-- "); q6();
  printf("Q8 ¥n"); q8();
  printf("Q9-- "); q9();
  printf("Q11-- "); q11();
  printf("Q12-- "); q12();
  printf("Q13-- "); q13();
  printf("Q14-- "); q14();
```

```
  printf("Q15-- "); q15();
  printf("Q16-- "); q16();
  printf("Q17-- "); q17();
  printf("Q18-- "); q18();
  printf("Q20-- "); q20();
}

void q1(void)
{
  int v;
  printf("&v = %x¥n¥n",&v);
}

void q2(void)
{
  char m[8];
  printf("&m[5] = %x¥n",&m[5]);
  printf("     m+5   = %x¥n¥n",m+5);
}

void q3(void)
{
  char m[5];
  printf("&m[0] = %x¥n",&m[0]);
  printf("     m     = %x¥n¥n",m);
}

void q4(void)
{
  char d[10][10];
  printf("*d       = %x¥n",*d);
  printf("     &d[0][0] = %x¥n",&d[0][0]);
  printf("     d[0]     = %x¥n¥n",d[0]);
}

void q5(void)
{
  int a=2, b=3, c=5, *p, *q;

  p = &b;
  q = &c;
  a = *p + *q;
  printf("a = %d¥n¥n",a);
}

void q6(void)
{
  int a=2, *p;
  p  = &a;
  *p = 5;
  printf("a = %d¥n¥n",a);
}

void q8(void)
{
  int k;
  char m[] = "answer";
```

```
  for(k=0;m[k] != NULL;k++){
      printf("m[%d]  = %c  ",k,m[k]);
    }
  printf("¥n");
  for(k=0;*(m+k) != NULL;k++){
      printf("*(m+%d) = %c  ",k,*(m+k));
    }
  printf("¥n¥n");
}

void q9(void)
{
  int x,y,*p;
  p = &x;
  printf("before adless = %d¥n",p);
  printf("     after  adless = %d¥n¥n",p+2);
}

void q11(void)
{
  static int m[5] = {10, 20, 40, 50, 30};
  printf("*m        = %d¥n",*m);
  printf("      *(m+3)    = %d¥n",*(m+3));
  printf("      *m+3      = %d¥n",*m+3);
  printf("      *m+*(m+3) = %d¥n¥n",*m+*(m+3));
}

void q12(void)
{
  static int d[][3] = {{1,2,3}, {5,6,7,}, {4,6,8}, {9,7,5}};
  printf("*d[2]     = %d¥n",*d[2]);
  printf("      *(d[2]+2) = %d¥n",*(d[2]+2));
  printf("      *d[2]+2   = %d¥n",*d[2]+2);
  printf("      **d       = %d¥n",**d);
  printf("      *(*d+3)   = %d¥n",*(*d+3));
  printf("      **d+6     = %d¥n",**d+6);
  printf("      *(d[1]+2) = %d¥n",*(d[1]+2));
  printf("      **(d+2)   = %d¥n¥n",**(d+2));
}

void q13(void)
{
  char *str = "abcdefg", *p;
  p = str +3;
  printf("p = %s¥n¥n",p);
}

void q14(void)
{
  char *p;
  p = "abc";
  printf("p      = %d¥n",p);
  printf("      *p     = %c¥n",*p);
  printf("      *(p+2) = %c¥n¥n",*(p+2));
}

void q15(void)
{
```

```
  static char m[] = "abcd";
  char *p, *q;
  p = &m[0];
  q = m;

  printf("*m = %c¥n",*m);
  printf("      *p = %c¥n",*p);
  printf("      *q = %c¥n¥n",*q);
}

void q16(void)
{
  static char m[] ="abcd";
  char *p;
  p = &m[2];

  printf("*p     = %c¥n", *p);
  printf("      *(m+2) = %c¥n",*(m+2));
  printf("      *m+2   = %c¥n¥n",*m+2);
}

void q17(void)
{
  char *p,m[] = "abcd";
  p = m;
  *(p + 1) = 'x';

  printf("p = %s¥n¥n",p);
}

void q18(void)
{
  int x;
  char *p;
  p = "abcd";
  if(p == "abcd")
    x=0;
  else
    x=1;
  printf("x = %d¥n¥n",x);
}

void q20(void)
{
  static char *q[] = {"abcd", "12345", "ABCDEFG", "987"};
  printf("*q[2]       = %c¥n",*q[2]);
  printf("      q[3][2]     = %c¥n",q[3][2]);
  printf("      *(q[2]+2)   = %c¥n",*(q[2]+2));
  printf("      *(*(q+3)+2) = %c¥n",*(*(q+3)+2));
  printf("      **(q+1)     = %c¥n",**(q+1));
}
```

---

b. 実行結果

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
[nw0413:~/prog1/report/#5] j04013% ./answer2
Q1-- &v = bffffcb0

Q2-- &m[5] = bfffcb5
     m+5   = bfffcb5

Q3-- &m[0] = bfffcb0
     m     = bfffcb0

Q4-- *d       = bffffc50
     &d[0][0] = bffffc50
     d[0]     = bffffc50

Q5-- a = 8

Q6-- a = 5

Q8
m[0]  = a  m[1]  = n  m[2]  = s  m[3]  = w  m[4]  = e  m[5]  = r
*(m+0) = a  *(m+1) = n  *(m+2) = s  *(m+3) = w  *(m+4) = e  *(m+5) = r

Q9-- before adless = -1073742672
     after  adless = -1073742664

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      = 11972
      *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

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

c. 考察

＊＊＊ポインタとアドレスについて＊＊＊
ポインタとは変数のアドレスを指す変数である。
アドレスとはすなわちある変数がおかれているメモリの番地のようなものである。

＊＊＊各、問題の考察＊＊＊

Q1. 変数のアドレスを表示するには前に&をつければ良い。

Q2. 1次元配列mの０番目から始める5番目の要素のアドレスを表す式は&m[5],m+5である。このとき、&m[5]はQ1と同じように表しており、m＋５はポインタを5つずらすような表し方をしている。

Q3. 1次元配列mの先頭アドレスを求める式は&m[0],mである。これはQ2と同じようにして先頭アドレスを求める式がつくれる。

Q4. 2次元配列dの先頭アドレスを求める式は&d[0][0],d[0],*d,である。

Q5. pはbのアドレスを指し、qはcのアドレスを指している。*pは指しているアドレスにあるモノであるため今回の場合はb、つまりは3を表す。*qにも同じことが言え、5を表すことになる。よってaの値は代入された8となる。

Q6. ポインタpは変数aのアドレスを指す。Q5で述べたように*pはアドレスではなく指したアドレスにあるモノである。よって*p = 5はa = 5に相当するものであるため解はa = 5となる

Q7. 問題文は*p,*q,q の値を示せ。**
ただし、*(100)=200 ,*(200)=300とする。

```
int *p, **q;
p = 100;
q = 100;
```

***pの値はアドレス100が示す値なので200となる。次に**qだがこれはポインタqのポインタ**

qを宣言するということだ。それよりポインタqはアドレス100をとる。それより、*q之値はpと同様に200となる。**pはアドレス200の値であるためその値は300となる。

Q8. プログラムからわかるように両方とも同じ値を出力している。このことから`m[k]`と`*(m+k)`とは同一のものということがわかる。

Q9. 整数型のポインタ変数pを`p+2`とおくことはつまりポインタの値を2つずらす、すなわちアドレスを2つずらすことに該当する。この場合、整数型のためひとつのアドレスにとられるバイトは4バイト。よって2ずらすことにより変化するバイトは8バイトとなる。

Q10. 今回、これについてのプログラムは書かなかった。しかし、ほかを見てもこれは明らかである。一次元配列mは＊mとおくこともでき、ポインタ変数pはp[0]のようにおける。

Q11. `*m`は初めの配列の要素を示すのでその値は10となる。`*(m+3)`は配列`m[3]`と同じであるためその値は50となる。`*m+3`は`*m`に3を足した値なので13が出力される。`*m+*(m+3)`は先ほどの`*m`と`*(m+3)`を足した値なので60と出力されている。

Q12. このプログラムは二次元配列を使用している。`*d[2]`は要素`{4,6,8}`内の初期の要素を表しているため4が出力される。`*(d[2]+2)`は`{4,6,8}`内のさらに2つずらした要素になるため8がその値となる。`*d[2]+2`は`*d[2]`に2を足した値、つまり6となる。`*d`は先頭アドレスをとり、その値である`**d`は初めの要素1となる。*(*d+3)は**d+6は**dが1をしめす、それに6を加えるため7となる。*(d[1]+2)は先頭アドレスから一つずらして{5,6,7}そこの先頭アドレスからさらに二つずらすので7が出力される。**(d+2)は先頭アドレスから2つずらして{4,6,8}そこの先頭アドレスをさし、その値を求めるので解は4となる。

Q13. ポインタは配列と同じように使えるため文字列をとれる。pは初期アドレスを3つずらしたところにおかれたため、そこから後のdefgの文字列が出力される。

Q14. ポインタpははじめ先頭アドレスを指すので&a、つまりは100をとる。今回のプログラムではアドレスの値100はとれなかったので本来のaのアドレスが出力されている。*pは先頭アドレスの値、つまりaである。*(p+2) はアドレスを2つずらした後の値なのでcが出力される。

Q15. *mは初めの要素を表すのでその値はaである。ポインタpはm[]の先頭アドレスがあたえられており、その値が*pなので*mと同様にaとなる。ポインタqも同様にm[]の先頭アドレスが与えられている。よって*qはa。この三つは違う表現をしているが表しているのは同じものであることがわかる。

Q16. ポインタpにはm[2]のアドレスが入っている。*pはその値なのでcとなる。*(m+2)はm[2]と同等であるよってその値はc。*mの値はa(aをASCII10進数にして)、それに2を足すことによりcとなる。

Q17. ポインタpを一つずらした値に'x'を代入する。そのためずれたさきに有ったbはxに書き換えられる。そのため、文字列は"axcd"となる。しかし、今回のプログラムはそのままでやるとBus errorと表示される。これは、ポインタpが文字列の先頭アドレスをとれていないためおこるようでそのため今回のプログラムは配列で置き換えることで出力した。

Q18. ポインタpは前で示したように配列と同じようにとれるためif文は真であり、1を出力することになる。これは、コンパイラによってはpをポインタと見る場合もあり、その場合は偽となり0を返すことになる。

Q19. この問題の解は

```
char m[MAX];
int k;
for (k=0;m[k];++k)
  m[k] += 1;
```

となる。これはポインタのかわりに整数型のkを宣言している。ポインタの値を配列の要素で置き換え(*(m+k) = m[k]だから)その要素をkとおいて一つずつ増やすことでもとと同じプログラムを作成した。このプログラムは配列が終わるまで繰り返し行い、配列の要素にはm[0]には0、m[1]には1、のように入っていく。

Q20. *q[2]は"ABCDEFG"内の先頭のアドレスをとったものである、よってAが出力される。
q[3][2]は三つずらして"987"、そこからさらに2つずらしたものつまり7となる。
***(*q[2]+2)はq[2]より"ABCDEFG"、そこの初めの値から2つずらしたCが値となる。**
***(*(q+3)+2)、これは先頭アドレスから3ずらし、"987"をしめす。そこから2つずらすことにより値7を示すことになる。*(q+1)は"12345"の先頭アドレスを示す。**(q+1)はその値なので1となる。**

[4] 感想・反省

今回は久しぶりに二週間の期間もあり、それなりに問題の意味もわかっていたのでわりと楽そうだなと始めたレポートでしたがその問題の多さに正直参りました。わかっていると思っていたものも深く考えると根本的にはわかっていなかったものがいくつかでてきて悩みました。考察だけでなくプログラムを作るのにも予想外の時間がかかり結局余裕なしです・・・。ポインタは難しいですね、まだわかったのかどうか不安だ・・・。まぁ、なんとか終わらすことができました。しかし、自分でも思うが読みにくいな・・・。次は読みやすさを追求してみたいなと思うレポートでした。

参考文献

C実践プログラミング (Steve Oualine　著)
C言語の演算子
(http://www.belution.com/miwaki/developer/c_operator.shtml)