ソース

```
#include <stdio.h>



int main(){
/*Q1.変数vのアドレスを求める式を示せ。*/
	{
		int v, *p;
		p=&v;/*ポインタpに変数vのアドレスを示す。*/
		printf("Q1.変数vのアドレスを求める式を示せ。¥n");
		printf("&v = 0x%x¥n",p);
	}
/*Q2.1次元配列mの番目から始まる番目の要素のアドレスを求める式をつ示せ。*/
	{
		int m[10];
		printf("Q2.1次元配列mの番目から始まる番目の要素のアドレスを求める式をつ示せ。
¥n");
		printf("m+5 = %p¥n",m+5);/* m+5は、mの先頭アドレスから番目を表す。*/
		printf("&m[5] = %p¥n",&m[5]);/*&m[5]は、配列mの番目を表す。*/


	}
/*Q3.1次元配列mの先頭アドレスを求める式をつ示せ。*/
	{
		char m[10];
		printf("Q3.1次元配列mの先頭アドレスを求める式をつ示せ¥n");
		printf("&m[0] = %p¥n",&m[0]);
		printf("m = %p¥n",m);
	}
/*Q4.2次元配列dの先頭アドレスを求める式をつ示せ。*/
	{
		char d[2][3];
		printf("Q4.2次元配列dの先頭アドレスを求める式をつ示せ。¥n");
		printf("&d[0][0] = %p¥n",&d[0][0]);
		printf("d[0] = %p¥n",d[0]);
		printf("d = %p¥n",d);
	}
```

```
/*Q5.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;/*ポインタpに変数bのアドレスを代入*/
		q = &c;/*ポインタcに変数cのアドレスを代入*/
		a = *p + *q;/* *p+*qはb+cとなるのでa=3+5となりa=8となる*/
		printf("Q5.次の文を実行した後の変数aの値を示せ。¥n");
		printf("a = %d¥n",a);
	}
/*Q6.次の文を実行した後の変数aの値を示せ。*/
	{
		int a=2,*p;
		p = &a;/*ポインタpに変数aの値を代入*/
		*p = 5;/* *pは「ポインタpが指すアドレスの中身」のこと、つまりaの中身がとなる。
*/
		printf("Q6.次の文を実行した後の変数aの値を示せ。¥n");
		printf("a = %d¥n",a);
	}
/*Q7.次の文を実行した後の*p,*q,**qの値を示せ。*/
	{
		int a=200,b=300;
		int *p,**q;
		p=&a;/*ポインタpに変数aのアドレスを代入*/
		q=&a;/*ポインタpに変数aのアドレスを代入*/
		printf("Q7.次の文を実行した後の*p,*q,**qの値を示せ。¥n");
		printf("*p = %d¥n",*p);
		printf("*q = %d¥n",*q);
		a=&b;
		printf("**q = %d¥n",**q);
	}
/*Q8.一次元配列において、m[k]と*(m+k)はどのような値か述べよ。*/
	{
		int m[10] = {12,34,56,78,90,23,45,67,89,01},k;
		printf("Q8.一次元配列において、m[k]と*(m+k)はどのような値か述べよ。¥n");
		for(k=0;k<=9;k++){/*kの初期値は,kが以下なら繰り返す。kの値を増やす。*/
			printf("m[%02d] = %02d¥t",k,m[k]);
```

```
                printf("*(m+%02d) = %02d¥n",k,*(m+k));
            }
        }
/*Q9.変数pにおいてp+2はpの値を何バイト増加させたものか。*/
        {
            int *p;
            printf("Q9.変数pにおいてp+2はpの値を何バイト増加させたものか。¥n");
            printf("&p = %p¥n",&p);
            printf("&(p+2) = %p¥n",&p+2);
        }
/*Q10.次の文章が正しいか述べよ。*/
        {
            int m[3]= {10,20,30};
            char *p;
            p = "abcdef";
            printf("Q10.次の文章が正しいか述べよ。¥n");
            printf("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。
¥n");
            printf("*(m+1) = %d¥n",*(m+1));
            printf("m[1] = %d¥n",m[1]);
            printf("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よい。¥n");
            printf("*(p+4) = %c¥n",*(p+1));
            printf("p[4] = %c¥n",p[1]);
        }
/*Q11.次の定義による、*m,*(m+3),*m+3,*m+*(m+3)の値を示せ。*/
        {
            static int m[5] = {10,20,50,40,30};
            printf("Q11.次の定義による、*m,*(m+3),*m+3,*m+*(m+3)の値を示せ。¥n");
            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",*m+*(m+3));
        }
/*Q12.次の定義による*d[2],*(d[2]+2),*d[2]+2,**d,*(*d+3),**d+6,*(d[1]+2),**(d+2)の値を示せ。*/
        {
            static int d[4][3] = {{1,2,3},{5,6,7},{4,6,8},{9,7,5}};
```

```
		printf("Q12.次の定義による
*d[2],*(d[2]+2),*d[2]+2,**d,*(*d+3),**d+6,*(d[1]+2),**(d+2)の値を示せ¥n");
		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",**(d+2));
	}
/*Q13.次の文を実行した後のポインタpの文字列を示せ。*/
	{
		char *str = "abcdefg",*p;
		p = str+3;
		printf("Q13.次の文を実行した後のポインタpの文字列を示せ。¥n");
		printf("p = %s¥n",p);
	}
/*Q14.次の文を実行した後のp,*p,*(p+2)の値を示せ。*/
	{
		char *p;
		p = "abc";/* pの先頭アドレスが指すところにaを、次のアドレスが指すところにbを、
そのまた次のアドレスが指すところにcを入れる*/
		printf("Q14.次の文を実行した後のp,*p,*(p+2)の値を示せ。¥n");
		printf("p = %d¥n",p);/*アドレスはコンパイラーによりきまるので、にはならない。
*/
		printf("*p = %c¥n",*p);
		printf("*(p+2) = %c¥n",*(p+2));

	}
/*Q15.次の文を実行した後の*m，*p，*qの値を示せ。*/
	{
		static char m[] = "abcd";/*staticでメモリ領域を確保している。配列mにそれぞれ、
m[0]にaを、m[1]にbをというように文字を代入していく。*/
		char *p,*q;
		p = &m[0];/*ポインタpにm[]の先頭アドレスを代入*/
```

```
		q = m;/*ポインタqにmの先頭アドレスを代入*/
		printf("Q15.次の文を実行した後の*m，*p，*qの値を示せ。¥n");
		printf("*m = %c¥n",*m);
		printf("*p = %c¥n",*p);
		printf("*q = %c¥n",*q);
	}
/*Q16.次の文を実行した後の*p,*(m+2),*m+2の値を示せ。*/
	{
		static char m[] = "abcd";
		char *p;
		p = &m[2];/*ポインタpにmの先頭アドレスから番目を代入。*/
		printf("Q16.次の文を実行した後の*p,*(m+2),*m+2の値を示せ。¥n");
		printf("*p = %c¥n",*p);
		printf("*(m+2) = %c¥n",*(m+2));
		printf("*m+2 = %c¥n",*m+2);
	}
/*Q17.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 *p;
		p = "abcd";
		*(p+1) = 'x';/* pの次の文字つまりp[1]の文字をxに変更する。*/
		printf("Q17.次の文を実行した後のポインタ変数pの文字列を示せ。¥n");
		printf("*p = %s¥n",p);
	}
/*Q18.次の文を実行した後の変数Xの値を示せ。*/
	{
		int x;
		char *p;
		p = "abcd";
		if(p == "abcd")
			x=0;
		else
			x=1;
		printf("Q18.次の文を実行した後の変数Xの値を示せ。¥n");
		printf("x = %d¥n",x);
	}
```

```
/*Q19.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。*/
	{
		char i,*p,m[5] = {1,2,3,4,5};
		printf("Q19.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。¥n");
		printf("書き換え前¥n");
		for(p=m;*p;++p,i++){/*ポインタpのアドレスと配列mの先頭アドレスを同じにする。*pの中身がNULLになったら終了。pのアドレスを増やす。*/
			*p+=1;
			printf("*(p+%d) = %d¥n",i,*p);
		}

	}
	{
		int k;
		char m[6] = {1,2,3,4,5};
		printf("変更後¥n");
		for(k=0;m[k];k++){/*kの初期値をに、m[k]の中身がNULLならば終了*/
			m[k] +=1;/*配列m[]の中身にを足す*/
			printf("m[%d] = %d¥n",k,m[k]);
		}
	}
/*Q20.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。*/
	{
		static char *q[] = {"abcd","12345","ABCDEFG","987"};
		printf("Q20.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。¥n");
		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));
	}
}
```

実行結果

Q1.変数 v のアドレスを求める式を示せ。

&v = 0x12ff70

Q2.1 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。

m+5 = 0012FF5C

&m[5] = 0012FF5C

Q3.1 次元配列 m の先頭アドレスを求める式を 2 つ示せ

&m[0] = 0012FF38

m = 0012FF38

Q4.2 次元配列 d の先頭アドレスを求める式を 3 つ示せ。

&d[0][0] = 0012FF30

d[0] = 0012FF30

d = 0012FF30

Q5.次の文を実行した後の変数 a の値を示せ。

a = 8

Q6.次の文を実行した後の変数 a の値を示せ。

a = 5

Q7.次の文を実行した後の*p,*q,**q の値を示せ。

*p = 200

*q = 200

**q = 300

Q8.一次元配列において、m[k]と*(m+k)はどのような値か述べよ。

m[00] = 12　　*(m+00) = 12

m[01] = 34　　*(m+01) = 34

m[02] = 56　　*(m+02) = 56

m[03] = 78　　*(m+03) = 78

m[04] = 90　　*(m+04) = 90

m[05] = 23　　*(m+05) = 23

m[06] = 45　　*(m+06) = 45

m[07] = 67　　*(m+07) = 67

m[08] = 89　　*(m+08) = 89

m[09] = 01　　*(m+09) = 01

Q9.変数 p において p+2 は p の値を何バイト増加させたものか。

&p = 0012FED4

&(p+2) = 0012FEDC

Q10.次の文章が正しいか述べよ。

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。

*(m+1) = 20

m[1] = 20

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よい。

*(p+4) = b

p[4] = b

Q11.次の定義による、*m,*(m+3),*m+3,*m+*(m+3)の値を示せ。

*m = 10

*(m+3) = 40

*m+3 = 13

m+(m+3) = 50

Q12.次の定義による*d[2],*(d[2]+2),*d[2]+2,**d,*(*d+3),**d+6,*(d[1]+2),**(d+2)

*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.次の文を実行した後のポインタ p の文字列を示せ。

p = defg

Q14.次の文を実行した後の p,*p,*(p+2)の値を示せ。

p = 4248932

*p = a

*(p+2) = c

Q15.次の文を実行した後の*m，*p，*q の値を示せ。

*m = a

*p = a

*q = a

Q16.次の文を実行した後の*p,*(m+2),*m+2 の値を示せ。

*p = c

*(m+2) = c

*m+2 = c

Q17.次の文を実行した後のポインタ変数 p の文字列を示せ。

*p = axcd

Q18.次の文を実行した後の変数 X の値を示せ。

x = 1

Q19.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。

書き換え前

*(p+0) = 2

*(p+1) = 3

*(p+2) = 4

*(p+3) = 5

*(p+4) = 6

変更後

m[0] = 2

m[1] = 3

m[2] = 4

m[3] = 5

m[4] = 6

Q20.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。

*q[2] = A

q[3][2] = 7

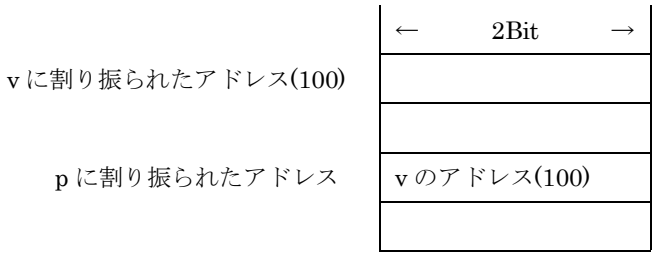
*(q[2]+2) = C

((q+3)+2) = 7

**(q+1) = 1

考察

Q1



p = &v(ここで&v=100 とおく)により、ポイント p に変数 v のアドレスを代入。

*p=5000 遠くと p の箱の中にある数字のアドレスに 5000 が入る。

Q8、10

&m(アドレス)	*m(ポインタ)	m[](配列)	値
0012FED8	*m	m[0]	12
0012FEDC	*(m+1)	m[1]	34
0012FEE0	*(m+2)	m[2]	56
:	:	:	:

:	:	:	:
0012FEFD8+4k	*(m+k)	m[k]	:
:	:	:	:
0012FEFD8+4n	*(m+n)	m[n]	:
:	:	:	:

Int 型で m を宣言しているので、アドレスの差は 4bit である。アドレス、配列 m、ポインタ m、値は図のような関係になっている。

なので、.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く、また.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よい。

Q7

p=&a(=100);変数 a のアドレスはコンパイルにより決まるので 100 と自分で決めることはできないがここでは便宜上変数 a のアドレスを 100 とする。同様にして&b を 200 と定めておく。

	値	
変数 a のアドレス(100)	200	
変数 b のアドレス(200)	300	A=&b
Q のアドレス(0012FF08)	100	q=&a
P のアドレス(0012FF10)	100	P=&a

というような関係となっている。

また自分で*(100)=200；などとやってもエラーが出る理由は、メモリがその領域を確保していないためである。

Q11

配列 m(static)	値
m[0]	10
m[1]	20
m[2]	50
m[3]	40
m[4]	30

*m はポインタ m の指すアドレスの中身なので m[0]の中身 10 を指す。

(m+3)は m[3]とあらわすことができるので、(m+3)の値は 50 である。

*m+3 は*m の値に 3 を加えるので 13。

m+(m+3)は、一見複雑に見えるけれども実際には*m の値と*(m+3)の値の足し算なので*m+*(m+3)の値は 10+50 で 60 となる。

Q12

二次元配列			
d[0]	[0]	[1]	[2]
	1	2	3
d[1]	[0]	[1]	[2]
	5	6	7
d[2]	[0]	[1]	[2]
	4	6	8
d[3]	[0]	[1]	[2]
	9	7	5

*d[2]は、d[2][0]の値をさすので*d[2]=4 となる。

*(d[2]+2)は、d[2][2]の値をさすので(d[2]+2)=8 となる。

*d[2]+2 は*d[2]の値に、2 を足すので、4+2 で 6 となる。

d は d[0][0]と同意なので、d の値は 1 となる。

*(*d+3)は*d+3 の値が 4 となるので*4 となり、先頭アドレスから 4 番目なので*(*d+3)=5 となる。

d＋6 はd=1 なので 1+6=7 となる。

(d[1]+2)は d[1][2]と同意なので、(d[1]+2)=7

**(d+2)、d[2][0]の値をさすので*d[2]=4 となる。

Q13

Char *str =”abcdefg”

*str	a
*(str+1)	b
*(str+2)	c
*(str+3)	d
*(str+4)	e
*(str+5)	f
*(str+6)	g
*(str+7)	NULL

右の図のようになっている。

p = str+3 により str+3 が p の先頭アドレスになるので、下の図のようになる

*p	*(str+3)	d
*(p+1)	*(str+4)	e
*(p+2)	*(str+5)	f
*(p+3)	*(str+6)	g
*(p+)4	*(str+7)	NULL

なので文字列を表示させると”defg”となる。

Q14

配列 m を作り、m の先頭アドレスを 100 とすると下の図のようになる。

p	m のアドレス(100)
(p+1)	m[1]のアドレス
(p+2)	m[2]のアドレス
(p+3)	m[3]のアドレス

m[0]	a
m[1]	b
m[2]	c
m[3]	NULL

右の図のような関係になっている

ここで便宜上 p のアドレスを 100 すると

*p=100 となり、*p=a,* (p+2)=c となる。

Q16

	m[0]	a
	m[1]	b
*P	m[2]	c
*(P+1)	m[3]	d
*(P+2)	m[4]	NULL

p=&m[2]により p の先頭アドレスに m[2]のアドレスを代入するので
右の図のような関係になる。
なので*p=a,*(m+2)=m[2]=2 となる。
*m+2 = 'a'+2 となり ASCll コード表の 0x61+2=0x63 となり c となる。

Q20

q[0]	abcd
q[1]	12345
q[2]	ABCDEFG
q[3]	987

[0]	[1]	[2]	[3]
a	b	c	d

[0]	[1]	[2]	[3]	[4]
1	2	3	4	5

[0]	[1]	[2]	[3]	[4]	[5]	[6]
A	B	C	D	E	F	G

[0]	[1]	[2]
9	8	7

上の図のようになっているので*q[2]=q[2][0]=A, q[3][2]=7, *(q[2]+2)=C,*(*(q+3)+2)= q[3][2]=7,
**(q+1)=q[1][0]=1