錬成問題

■ 単項 **&** 演算子は［ (1) ］演算子と呼ばれ、オブジェクトへの［ (2) ］を生成する演算子であり、単項 ***** 演算子は［ (3) ］演算子と呼ばれ、［ (2) ］が指すオブジェクトのエイリアスすなわち別名を生成する。

■ 右に示すのは、**int**型変数*no*のアドレスを表示する関数呼出しである。

```
printf("  (4)  ",  (5)  )
```

■ 右に示すプログラム部分の実行結果を示せ。

```
int  i, j;
int *ptr = &i;

printf("ptr == &i  → %d\n", ptr == &i);
printf("ptr == &j  → %d\n", ptr == &j);
```

```
ptr == &i → (6)
ptr == &j → (7)
```

■ 右に示すプログラム部分の実行結果を示せ。

```
int  x = 100;
int  y = 500;
int *ptr = &x;

printf("*ptr = %d\n", *ptr);

*ptr = 400;
printf("x = %d\n", x);
printf("y = %d\n", y);
```

```
*ptr = (8)
x = (9)
y = (10)
```

10

■ 右に示すのは、**int**へのポインタ*nx*, *ny*の値を入れかえるプログラム部分である。たとえば、*nx*が*x*を指し、*ny*が*y*を指している状態で、右に示すプログラム部分を実行した後は、*nx*は*y*を、*ny*は*x*を指すことになる。

```
int  *temp;
temp = (11) ;
nx = (12) ;
ny = (13) ;
```

■ 右に示すのは、*nx*が指す変数の値を二乗する関数である。

```
void sqrf(int *nx)
{
    (14) ;
}
```

■ 右に示すのは、*a*を*b*で除した商（小数点以下は切り捨て）を、*np*が指す変数に格納する関数である。

```
void mod(int a, int b, int *np)
{
    (15) = (16) ;
}
```

■ 右に示すのは、*nx*, *ny*が指す**int**型変数の値を入れかえる関数である。

```
void swap(int *nx, int *ny)
{
    int  temp = *nx;
    *nx = (17) ;
    *ny = (18) ;
}
```

■ 右に示すのは、*a*, *b*が指す**int**型変数を昇順になるように並べかえる関数である。ただし、前問の関数*swap*が、与えられているとする。

```
void sort2(int *a, int *b)
{
    if (*a > *b)
        swap( (19) , (20) );
}
```

■ 以下に示すのは、いずれも、要素型が**int**型である配列*vc*の先頭要素に0を代入する関数である。

```
void set0(int vc (21) )
{
    vc[ (22) ] = 0;
}
```

```
void set0(int vc (21) )
{
    (23) vc = 0;
}
```

```
void set0(int (24) vc)
{
    (25) (vc + (26) ) = 0;
}
```

```
void set0(int (24) vc)
{
    0[ (27) ] = 0;
}
```

■ 右に示すのは、要素数が*no*である配列*vc*の全要素に0を代入する関数である。

```
void setary0(int (24) vc, int no)
{
    while ( (28) > 0)
        (29) (vc + (30) ) = 0;
}
```

■ 右に示すプログラム部分の実行結果を示せ。

```
int  v[10], *p = v;

printf("p == v         → %d\n", p == v);
printf("p != &v[0]     → %d\n", p != &v[0]);
printf("p == &v[1]     → %d\n", p == &v[1]);
printf("p == &v[2]     → %d\n", p == &v[2]);
printf("&v[1] - &v[0] → %d\n", &v[1] - &v[0]);
printf("&v[2] - &v[0] → %d\n", &v[2] - &v[0]);
```

```
p == v         → (31)
p != &v[0]     → (32)
p == &v[1]     → (33)
p == &v[2]     → (34)
&v[1] - &v[0] → (35)
&v[2] - &v[0] → (36)
```