錬成問題

■ 関数呼出しの際に、呼び出す側が渡す引数を［ (1) ］、呼び出される側が受け取る引数を［ (2) ］と呼び、［ (2) ］には［ (1) ］の値がコピーされる。

■ **int**型の変数*no*の値が10である。*f*が右に示す関数であるとき、これを*f*(*no*)と呼び出した後の*no*の値は［ (3) ］である。
なお、ここでの**( )**を［ (4) ］演算子と呼ぶ。

```
void f(int x)
{
    x = 5;
}
```

■ 以下に示すのは、いずれも受け取った二つの引数*x*, *y*の小さい方の値を返す関数である。

```
int minof(int x, int y)
{
    if ( (5) )
        return (x);
    else
        return (y);
}
```

```
int minof(int x, int y)
{
    return ( (6) ? x : y);
}
```

```
int minof(int x, int y)
{
    int  min;

    if ( (7) )
        min = x;
    else
        min = y;
    return ( (8) );
}
```

■ 以下に示すのは、いずれも*dx*の*no*乗の値を返す関数である。

```
double power(double dx, int no)
{
    int     i;
    double  tmp = (9) ;

    for (i = 0; (10) ; i++)
        tmp *= dx;
    return (tmp);
}
```

```
double power(double dx, int no)
{
    double  tmp = (9) ;

    while ( (11) > 0)
        tmp *= dx;
    return (tmp);
}
```

■ 右に示すのは、正の整数値を読み込んで、その値を返す関数である。なお、正でない値を読み込んだ場合は、再入力を行うこととする。

```
int scan_pint(void)
{
    int  temp;
    do {
        scanf("%d", (12) );
    } while (temp (13) 0);
    return ( (14) );
}
```

■ 右に示すのは、受け取った*x*の絶対値を返す関数である。

```
int abs(int x)
{
    return (  (15)  ? x : -x);
}
```

■ 右に示すのは、受け取った*x*, *y*の差を返す関数である。

```
int diff(int x, int y)
{
    return (  (16)  ? x - y : y - x);
}
```

■ 右に示すのは、受け取った*x*, *y*について、*x*が*y*より大きければ1を、*x*が*y*より小さければ-1を、等しければ0を返す関数である。

```
int intcmp(int x, int y)
{
    return (  (17)  ? 0 :
                  (18)  ? 1 : -1);
}
```

■ 右に示すのは、受け取った*x*の値から0までカウントダウンしながら表示する関数である。

```
void count_down(int x)
{
    while (  (19)  )
        printf("%d ", x--);
}
```

■ 右に示すのは、警報を1回発する関数である。

```
void alert(void)
{
    putchar(  (20)  );
}
```

■ 右に示すのは、受け取った*x*, *y*の平均値を**double**型で返す関数である。

```
double ave(int x, int y)
{
    return ((  (21)  )(x + y) / 2);
}
```

■ 右に示すのは、受け取った*x*, *y*の大きい方の値を返す関数と、それを利用して、四つの値の最大値を返す関数である。

```
int max2(int x, int y)
{
    return (x > y ?  (22)  :  (23)  );
}

int max4(int a, int b, int c, int d)
{
    return (max2(max2(a, b),  (24)  ));
}
```

（次ページへ続く）

■ 右に示すのは、要素数が*no*である配列*x*の要素の最大値を返す関数である。

```
int maxof(int x[], int no)
{
    int  max = [ (25) ];
    while ([ (26) ] > 0)
        if (max < x[no]) max = x[no];
    return (max);
}
```

■ 右に示すのは、要素数が*no*である配列*x*の要素の最小値を返す関数である。

```
int minof(int x[], int no)
{
    int  i, min = [ (27) ];
    for (i = [ (28) ]; i < no; i++)
        if (x[i] < min) min = x[i];
    return (min);
}
```

6

■ 右に示すのは、配列*x*中の*x*[*n1*] ～ *x*[*n2*]の要素に値*v*を格納する関数である。

```
void fill_sub(int x[], int n1, int n2, int v)
{
    int  i;
    for (i = [ (29) ]; i < [ (30) ]; i++)
        x[i] = [ (31) ];
}
```

■ 右に示すのは、要素数が*no*である配列*vc*の全ての要素に*x*を代入する関数と、それを利用して、配列*x*の全ての要素に0を代入する関数である。

```
void set_intary(int vc[], int no, int x)
{
    while ([ (32) ] > 0)
        [ (33) ] = x;
}

void set0_intary(int vc[], int no)
{
    set_intary([ (34) ], no, 0);
}
```

■ 右に示すのは、3行4列の2次元配列*v*の全要素に0を代入する関数である。

ここで*v*の宣言において、[]内の3は [(35)]。また、4は [(36)]。

▶ [(35)]および[(36)]の選択肢

(a) 省略できる

(b) 省略できない

```
void set0_inta34(int v[3][4])
{
    int  [ (37) ];
    for (i = 0; i < [ (38) ]; i++)
        for (j = 0; j < [ (39) ]; j++)
            v[i][j] = 0;
}
```

■ 右に示すプログラムにおいて、*a*は (40) 有効範囲を、*b*と*c*は (41) 有効範囲をもつ。

また、*a*は (42) 記憶域期間を、*b*は (43) 記憶域期間を、*c*は (44) 記憶域期間をもつ。

このプログラムの実行において、必ず0で初期化される変数は、(45) である。

▶ (45) の選択肢 … (a) *a* (b) *a*と*b* (c) *b*と*c* (d) *a*と*c*

```
int a;

int main(void)
{
    int  b;
    static int c;
}
```

■ 右に示すプログラムの実行結果を示せ。

```
a = (46)
b = (47)
a = (48)
b = (49)
a = (50)
b = (51)
```

なお、網掛け部は、関数*fn*の返却値型や引数に関する仕様を知らせるための (52) 宣言である。

```
#include  <stdio.h>

int main(void)
{
    void fn(void);

    int  i;
    for (i = 0; i < 3; i++)
        fn();
    return (0);
}

void fn(void)
{
    int         a = 0;
    static int  b = 0;
    printf("a = %d\n", ++a);
    printf("b = %d\n", ++b);
}
```

■ 以下に示すプログラムの誤りを全て指摘せよ。 (53)

```
#include  <stdi0.h>

#defined    MAX     100

void diffMAX(int x)
{
    return (diff > 100 ? diff - 100 : 100 - diff);
}

void main(int)
{
    int  no;

    pritnf("整数：");            /* 整数値の入力を促す */
    scamf("%f", *no);            /* 整数値を読み込む */

    pritnf("その数と%dとの差は%dです。\n", MAX, diff<no>);

    return (0);
}
```