錬成問題

■ 文字列リテラルは［ (1) ］記憶域期間をもち、その末尾には［ (2) ］文字が付加される。

■ 右に示すプログラム部分の実行結果を示せ。

```
文字列str = [ (3) ]
```

```
char  str[] = "321\0SEC";

printf("文字列str = %s\n", str);
```

■ 右に示すプログラム部分の実行結果を示せ。

```
[ (4) ]
[ (5) ]
[ (6) ]
```

```
printf("%u\n", (unsigned)sizeof(""));
printf("%u\n", (unsigned)sizeof("ZZ"));
printf("%u\n", (unsigned)sizeof("Z\0Z"));
```

■ 以下に示すのは、右に示すように名前を表すための文字列を読み込んで、「Hello, 名前!!」と表示するプログラム部分である。

```
May I have your name : Hiroshi↵
Hello, Hiroshi!!
```

```
char  name[100];

printf("May I have your name : ");          /* 名前の入力を促す */
scanf("[ (7) ]", [ (8) ]);                  /* 文字列を読み込む */

printf("Hello, [ (9) ]!!\n", [ (10) ]);     /* 挨拶する */
```

■ 右に示すのは、文字列*st*を空文字列にする関数である。

```
void null(char st[])
{
    [ (11) ] = '\0';
}
```

■ 右に示すのは、文字列*st*の内容が"ABC"であれば1を、そうでなければ0を返す関数である。

```
int isABC(const char st[])
{
    if ([ (12) ] != 'A')  return (0);
    if ([ (13) ] != 'B')  return (0);
    if ([ (14) ] != 'C')  return (0);
    if ([ (15) ] != '\0') return (0);
    return (1);
}
```

■ 以下に示すのは、文字列*st*に含まれている数字文字の個数を返す関数である。

```
int dgt_num(const char st[])
{
    int  i, dig = 0;

    for (i = 0; [ (16) ]; i++)
        if (st[i] >= [ (17) ]  &&  st[i] <= [ (18) ])
            dig++;
    return ([ (19) ]);
}
```

■ 右に示すのは、文字列*st*の長さ（ナル文字は含まない）を返す関数である。

```
unsigned strleng(const char st[])
{
    unsigned  len = 0;

    while (st[  (20)  ])
        len++;
    return (  (21)  );
}
```

■ 以下に示すのは、いずれも、文字列*st*を表示する関数である。

```
void putstr(const char st[])
{
    unsigned  i = 0;

    while (st[i])
        putchar(  (22)  );
}
```

```
void putstr(const char st[])
{
    unsigned  i = 0;

    while (st[i])
        printf(  (23)  ,  (24)  );
}
```

■ 以下に示すのは、いずれも、要素数が*no*である文字列（要素数が10で要素型が**char**型である配列）の配列*sv*の各文字列を表示する関数である。

```
void putstra(const char sv[][10], int no)
{
    int  i;
    for (i = 0; i < no; i++)
        printf("sv[%d]＝\"%s\"\n", i, sv[  (25)  ]);
}
```

```
void putstra(const char sv[][10], int no)
{
    int  i, j;
    for (i = 0; i < no; i++) {
        printf("sv[%d]＝\"", i);
        for (j = 0;   (26)  ; j++)
            putchar(  (27)  );
        printf("\"\n");
    }
}
```

■ 以下に示すのは、文字列*st*が回文（前から読んでも後ろから読んでも同じ）であれば1を、そうでなければ0を返す関数である。

```
int isPalindrome(const char st[])
{
    unsigned  i, len = 0;
    while (st[len])
        len++;
    for (i = 0; i <   (28)  ; i++)
        if (st[i] !=   (29)  )
            return (  (30)  );
    return (  (31)  );
}
```

（次ページへ続く）

■ 右に示すのは、文字列*st*に含まれる文字*ch*の添字を返す（複数存在する場合は、より先頭側のものとし、一つも存在しない場合は-1とする）関数である。

```
int strindex(const char st[], int ch)
{
    unsigned  i = 0;

    while ( (32) ) {
        if (st[i] == ch)
            return ( (33) );
        i++;
    }
    return ( (34) );
}
```

■ 右に示すのは、文字列*st*に含まれる文字*ch*の添字を返す（複数存在する場合は、より末尾側のものとし、一つも存在しない場合は-1とする）関数である。

```
int strrindex(const char st[], int ch)
{
    unsigned  i = 0, idx = -1;

    while ( (35) ) {
        if (st[i] == ch)
            idx = (36) ;
        (37) ++;
    }
    return ( (38) );
}
```

■ 以下に示すのは、文字列*st*に含まれている数字文字'0'～'9'の個数を、*tx*[0]～*tx*[9]に格納する関数である。

```
void dgt_no(const char st[], int tx[])
{
    int  i;

    for ( (39) = 0; (40) ; i++)
        tx[i] = (41) ;
    for ( (42) = 0; (43) ; i++)
        if ( (44) >= '0'  &&  (45) <= '9')
            (46) ++;
}
```

■ 右に示すのは、文字列*s1*に文字列*s2*をコピーする関数である。

```
void strcopy(char s1[], const char s2[])
{
    int  i;

    for (i = 0; (47) [i]; i++)
        s1[i] = s2[ (48) ];
    (49) = '\0';
}
```

■ 以下に示すのは、文字列*s1*と文字列*s2*が等しければ1を、そうでなければ0を返す関数である。

```
int streq(const char s1[], const char s2[])
{
    int  i;

    for (i = 0; s1[i] (50) s2[i]; i++)
        if (s1[i] == (51) )
            return ( (52) );
    return ( (53) );
}
```

■ 以下に示すのは、数字の並びである文字列 *st* を整数値に変換した値を返す関数である。たとえば、*st*が**"1234"**であれば整数値1234を返す。ただし、文字列中に数字以外の文字が一つでも入っていれば -1 を返す。

```
int strtoint(const char st[])
{
    int  i, no = (54);

    for ((55); st[i]; (56))
        if (st[i] >= '0'  &&  st[i] <= '9')
            no = no * 10 + (57);
        else
            return ((58));
    return ((59));
}
```

■ 以下に示すのは、文字列 *s1* の中に文字列 *s2* が含まれていれば、その先頭の添字を返す関数である（複数存在する場合は、より先頭側のものとし、一つも存在しない場合は -1 とする）。たとえば、*s1* が **"ABCAICCAI"** で、*s2* が **"CAI"** のときは 2 を返す。

```
int strstrindex(const char s1[], const char s2[])
{
    int  i;

    if ((60) == '\0')                /* s2が空のときは探索不要 */
        return ((61));
    for (i = 0; s1[i] != '\0'; i++) {
        if (s1[i] == s2[(62)]) {
            int  j = 0;
            do {
                if (s2[(63)] == '\0')
                    return ((64));
            } while (s1[(65)] == s2[j]);
        }
    }
    return ((66));
}
```

■ 以下に示すのは、文字列 *s2* に含まれない文字を、文字列 *s1* から全て取り除く関数である。たとえば、*s1* が **"ABCKCAE"** で *s2* が **"ACE"** であれば、*s1* を **"ACCAE"** とする。

```
void strinstr(char s1[], const char s2[])
{
    int  i, j, idx = (67);

    for (i = 0; (68); i++) {
        for (j = 0; (69); j++)
            if (s1[i] == s2[j]) {
                s1[(70)] = s1[i];
                break;
            }
    }
    s1[(71)] = '\0';
}
```