関数とポインタ

演習問題 14.1　文法事項

1）引数の値渡しの利点と欠点について簡潔に説明しなさい。

解答例　値渡しとは、引数によるデータの受け渡しの際に、呼び出し側の実引数から呼び出される関数側の仮引数に値をコピーする方法でデータを受け渡す方法です。

値渡しの最大の利点は、呼び出された関数側ではコピーされたデータを用いて処理を行うため、呼び出した側のデータを破壊する恐れがなく、安全なプログラムを作りやすいということです。言い換えると、呼び出された関数側では、呼び出した側の実引数の値を書き換えることはできません。

上記の利点は、場合によっては欠点となります。呼び出された関数側から呼び出した側の実引数の値を書き換えたい場合、通常の方法では不可能だからです。この欠点は先に述べた利点と裏腹の関係にあるため、欠点だけを取り除くことはできません。

2）値渡しの欠点を解決する方法について簡潔に答えなさい。

解答例　1) で述べた値渡しの欠点を解消するために、引数としてポインタを渡す方法がよく用いられます。呼び出し側の実引数として変数などのポインタを渡すと、呼び出された関数側では、そのポインタを用いて呼び出し側の変数を書き換えることができます。

3）関数呼び出しの引数に配列を用いる場合の特徴は何ですか。

解答例　引数として配列を与えるとはどういうことかを考えるには、配列名とは何であったかを思い出すとよいでしょう。C のプログラム中では、配列名は配列の先頭要素のアドレス表します。つまり、ポインタです。

このため、配列は普通の変数のように値渡しで渡されるのではなく、先頭要素のポインタだけが関数側に渡されます。従って、呼び出された関数側で配列を操作すると、呼び出し側で実引数として用いられた配列の内容も書き変わってしまいます。

配列の引数渡しがこのようなメカニズムであるのには理由があります。考えてもみて下さい。大量のデータを格納した配列の全ての要素を値渡しで渡すとすると、それらの値をコピーするだけで多くの時間を費やしてしまい、プログラムの実行効率が極端に低下してしまうでしょう。

演習問題 14.2

極座標形式 (r, θ)で表された複素数の虚部と実部を求めるプログラムを書きなさい。極座標形式 (r, θ)で表された複素数の虚部と実部を求める関数を作り、main()関数がその関数を呼び出すような形式でプログラミングしてください。

解答例　解答例を解答プログラム 14.1 に示します。

このプログラムは本文中のプログラム例 14.3 とほとんど同じです。プログラム例 14.3 では **int** 型のデータを扱っていますが、このプログラムでは **double** 型のデータを扱っています。扱うデータ型が違っても、ポインタの扱いは全く同じです。

このプログラムは、標準数学関数を用いていますので、**math.h** のインクルードとコンパイル時の **-l** スイッチを忘れないで下さい。

解答プログラム 14.1　複素数の実部と虚部

```
/************************************************
    p14_1.c
    複素数の実部と虚部
    Tadaaki Shimizu    2007.01.15
************************************************/
#include <stdio.h>
#include <math.h>
void getRealImag(double, double, double *, double *); //実部と虚部を得る

int main(void)
{
    double radius, theta;       // 半径と偏角
    double realPart, imagPart; // 実部と虚部

    printf("複素数入力：半径 >>");
    scanf("%lf", &radius);
    printf("複素数入力：偏角 >>");
    scanf("%lf", &theta);

    getRealImag(radius, theta, &realPart, &imagPart);

    printf("実部 : %lf\n", realPart);
    printf("居部 : %lf\n", imagPart);
    exit(0);
}

/************************************************
    関数名： getRealImag()
    機能   : 複素数の実部と虚部を得る
    引数   : double radius, theta 半径と偏角
             double *real,  *imag 実部と虚部
************************************************/
void getRealImag(double radius, double theta, double *real, double *imag)
{
    *real = radius * cos(theta);
    *imag = radius * sin(theta);
    return;
}
```

演習問題 14.3

文字列の長さを求めるプログラムを作りなさい。ただし、文字列の長さを求める関数 strl()を作り、main()関数がその関数を呼び出すような形式でプログラミングしてください。

1）プログラム例 14.4 を参考にして、プログラムしなさい。

解答例 解答例を解答プログラム 14.2 に示します。
プログラム例 14.4 と良く似ていますが、関数に返り値がある点が大きく異なります。配列を引数として渡す場合のプログラムとして、非常に判りやすい例なので、全体を通してよく理解して下さい。
for 文に空文が従属している点にも注意して下さい。この部分が判らない方は、本文 10.2.2 節を復習してみて下さい。

解答プログラム 14.2　文字列の長さ

```
/************************************************
    p14_2.c
    文字列の長さ
    Tadaaki Shimizu    2007.01.15
************************************************/
#include <stdio.h>
int strl(char []);

int main(void)
{
    char string[100];  // 入力される文字列
    int  length;       // 文字列の長さ

    printf(" Input string >>");
    scanf("%s", string);

    length = strl(string);

    printf("[%s] の長さは %d です。\n", string, length);
    exit(0);
}

/************************************************
    関数名 : strl()
    文字列の長さ：str[] の長さを返す
    引数：   char str[]  文字列用の配列
    返り値： int   文字列の長さ
************************************************/
int strl(char str[])
{
    int length; // 文字列の長さ

    for(length = 0; str[length] != '\0'; length++) ; //空文

    return length;
}
```

2）プログラム例 14.5 を参考にして、プログラムしなさい。

解答例　解答例を解答プログラム 14.3 に示します。

これは、解答プログラム 14.2 を、仮引数にポインタを用いて書き換えたものです。全く同じ働きの関数が、配列でもポインタでも作れることがよく判るでしょう。どちらを用いる方が良いか、この場合は微妙なところです。どちらでも大差無いのですから、読者の好きな方を採用すれば良いでしょう。

解答プログラム 14.3　文字列の長さ (ポインタの利用)

```
/************************************************
    p14_3.c
    文字列の長さ ( ポインタの利用 )
    Tadaaki Shimizu    2007.01.15
************************************************/
#include <stdio.h>
int strl(char *);

int main(void)
{
    char string[100];  // 入力される文字列
    int  length;       // 文字列の長さ

    printf(" Input string >>");
    scanf("%s", string);

    length = strl(string);

    printf("[%s] の長さは %d です。\n", string, length);
    exit(0);
}

/************************************************
    関数名 : strl()
    文字列の長さ：*ptr の指す文字列の長さを返す
    引数：    char *ptr  文字列用へのポインタ
    返り値： int   文字列の長さ
************************************************/
int strl(char *ptr)
{
    int length; // 文字列の長さ

    for(length = 0; *ptr != '\0'; length++, ptr++) ; //空文

    return length;
}
```

演習問題 14.4

文字列を接続するプログラムを作りなさい。ただし、文字列を接続する関数 strjoin()を作り、main()関数がその関数を呼び出すような形式でプログラミングしてください。

1）プログラム例 14.4 を参考にして、プログラムしなさい。

解答例　解答例を解答プログラム 14.4 に示します。

解答プログラム 14.4　文字列の接続

```
/************************************************
    p14_4.c
    文字列の接続
    Tadaaki Shimizu    2007.01.15
************************************************/
#include <stdio.h>
void strjoin(char [], char []);

int main(void)
{
    char aStr[100];  // 入力される文字列
    char bStr[100];  // 複写先の文字列

    printf(" Input string >>");
    scanf("%s", aStr);
    printf(" Input string >>");
    scanf("%s", bStr);

    strjoin(aStr, bStr);

    printf("接続結果 :%s\n", bStr);
    exit(0);
}

/************************************************
    関数名 : strjoin()
    文字列の接続：bStr[] に aStr[] を接続する
    引数： char aStr[], bStr[]  文字列用の配列
************************************************/
void strjoin(char aStr[], char bStr[])
{
    int aCnt, bCnt;

    // bStr[] 文字列の終端を探す
    for(bCnt = 0; bStr[bCnt] != '\0'; bCnt++)
        ;    // 空文

    // 文字列のコピー
    for(aCnt = 0; aStr[aCnt] != '\0'; aCnt++)
        bStr[bCnt + aCnt] = aStr[aCnt];
    bStr[bCnt + aCnt] = '\0';

    return;
}
```

2）プログラム例 14.5 を参考にして、プログラムしなさい。

解答例　解答例を解答プログラム 14.5 に示します。

この演習問題では、仮引数として配列を用いるよりも、ポインタを用いる方がプログラムがずっとスマートになっていることに気付くと思います。解答プログラム 14.4 と解答プログラム 14.5 の **strjoin()** 関数をよく見比べてみて下さい。

この例のように、配列要素を順番にアクセスし、数を数える必要がないような手続きでは、多くの場合、配列よりもポインタを使う方がずっとスマートになります。

解答プログラム 14.5　文字列の接続 (ポインタの利用)

```
/************************************************
    p14_4.c
    文字列の接続 ( ポインタの利用 )
    Tadaaki Shimizu    2007.01.15
************************************************/
#include <stdio.h>
void strjoin(char *, char *);

int main(void)
{
    char aStr[100];  // 入力される文字列
    char bStr[100];  // 複写先の文字列

    printf(" Input string >>");
    scanf("%s", aStr);
    printf(" Input string >>");
    scanf("%s", bStr);

    strjoin(aStr, bStr);

    printf("接続結果 :%s\n", bStr);
    exit(0);
}

/************************************************
    関数名 : strjoin()
    文字列の接続：*bPtr の文字列に *aPt の文字列を接続する
    引数： char *aPtr, *bPtr  文字列のポインタ
************************************************/
void strjoin(char *aPtr, char *bPtr)
{
    // bStr[] 文字列の終端を探す
    for(; *bPtr != '\0'; bPtr++)   ; // 空文

    // 文字列のコピー
    for(; *aPtr != '\0'; aPtr++, bPtr++)
       *bPtr = *aPtr;
    *bPtr = '\0';

    return;
}
```