2005 年度 プログラミング演習 II　演習 5

2005 年 11 月 2 日 (水) (11 月 3 日 (木) 修正)

演習 1　以下は正の整数 x, y $(x \geq y > 0)$ に対して「ユークリッドの互除法を用いて最大公約数を求めるプログラム」である。次の (1) から (3) の問いに答えなさい。

● ユークリッドの互除法を用いて最大公約数を求めるプログラム　　p2exercise0404.c

```
 1: #include <stdio.h>
 2:
 3: int main(void)
 4: {
 5:     int x = 672, y = 204, tmp;  ← tmp : temporary の略で「臨時」という意味
 6:
 7:     printf("gcd(%d,%d)=", x, y);
 8:     while (y != 0) {
 9:         tmp = x % y;
10:         x = y;
11:         y = tmp;
12:     }
13:     printf("%d¥n", x);
14:
15:     return 0;
16: }
```

(1)　ソースプログラム「p2exercise0404.c」を変更し、変数x, y を与えるとその最大公約数を返す関数「gcd(x, y)」を作成しなさい (ファイル名「p2exercise0501.c」)。ヒント：

- プロトタイプの宣言は「int gcd(int x, int y);」と記述する。
- main() 関数は以下のように記述する。

```
int main(void)
{
    int x = 672, y = 204;

    printf("gcd(%d,%d)=%d¥n", x, y, gcd(x, y));

    return 0;
}
```

(2)　ソースプログラム「p2exercise0501.c」のmain() 関数を少し変更すると 3 変数x, y, z の最大公約数を求めるプログラムが作成できる (z= 124 とする)。「ユークリッドの互除法を用いて 3 変数の最大公約数を求めるプログラム」を作成しなさい (ファイル名「p2exercise0502.c」)。ヒント：gcd() 関数を 2 回適用する。

(3)　ソースプログラム「p2exercise0501.c」のgcd() 関数を利用して変数x, y を与えると最小公倍数を返す関数「lcm(x, y)」を作成しなさい (ファイル名「p2exercise0503.c」)。ヒント：

- プロトタイプの宣言「int lcm(int x, int y);」を追加する。
- main() 関数は以下のように記述する。

```
int main(void)
{
    int x = 672, y = 204;

    printf("lcm(%d,%d)=%d¥n", x, y, lcm(x, y));

    return 0;
}
```

演習 2　以下は正の整数 n ($n \geq 0$) に対して「再帰的に $n!$ を求めるプログラム」である。次の (1) から (3) の問いに答えなさい ((2) と (3) は制御文の復習)。

● 再帰的に $n!$ を求めるプログラム　p2exercise0504.c

```
 1: #include <stdio.h>
 2:
 3: int func(int n);
 4:
 5: int main(void)
 6: {
 7:     int n = 5;
 8:
 9:     printf("%d!=%d¥n", n, func(n));
10:
11:     return 0;
12: }
13:
14: int func(int n)
15: {
16:     if (n == 0) {
17:         return 1;
18:     }
19:     else {
20:         return n * func(n-1);
21:     }
22: }
```

(1)　ソースプログラム「p2exercise0504.c」では再帰的に func() 関数が呼び出されている。呼び出す側の関数とその戻り値に注目し、流れを追いながら以下の表を完成しなさい。ヒント：矢印の順に解答する。数学の記号を用いれば以下のように定義されたのと同じである。

$$\texttt{func(n)} = \begin{cases} \texttt{n * func(n-1)}, & \texttt{n} > 0, \\ \texttt{1}, & \texttt{n} = 0 \end{cases}$$

	呼び出す関数	戻り値
main() 関数	func(5)　↓	↑　5 * func(4) = 120
func() 関数	↓	↑　=
func() 関数	↓	↑　=
func() 関数	↓	↑　=
func() 関数	↓	↑　=
func() 関数	→	→　=

(2)　for 文を使って「$n!$ を求めるプログラム」を作成しなさい (ファイル名「p2exercise0505.c」)。

(3)　while 文を使って「$n!$ を求めるプログラム」を作成しなさい (ファイル名「p2exercise0506.c」)。

演習 3　以下は複素数 a+bi, c+di に対して「複素数の四則演算を行なうプログラム」の一部である (i は虚数単位)。和を計算する関数に習って、差・積・商を計算する関数を追記しなさい (0 で割らないと仮定する)。ただし、変数 (演算結果) の受け渡しにはグローバル変数 x, y (x+yi) を使用すること。

* 変数の受け渡しに配列x[2] (x[0]+x[1]i) を使用する場合についても考察しなさい。

● 複素数の四則演算を行なうプログラム (一部抜粋)　p2exercise0507.c

```
 1: #include <stdio.h>
 2:
 3: double x, y;  /* x + yi */
 4:
 5: void add(double a, double b, double c, double d);          ← addition の略
 6: void sub(double a, double b, double c, double d);       ← subtraction の略
 7: void mul(double a, double b, double c, double d);    ← multiplication の略
 8: void div(double a, double b, double c, double d);          ← division の略
 9:
10: int main(void)
11: {
12:     double a = 1.2, b = -0.5;  /* a + bi */
13:     double c = -2.2, d = 6.8;  /* c + di */
14:
15:     add(a, b, c, d);
16:     printf("((%f)+(%f)i)+((%f)+(%f)i)=(%f)+(%f)i", a, b, c, d, x, y);
17:     sub(a, b, c, d);
18:     printf("((%f)+(%f)i)-((%f)+(%f)i)=(%f)+(%f)i", a, b, c, d, x, y);
19:     mul(a, b, c, d);
20:     printf("((%f)+(%f)i)*((%f)+(%f)i)=(%f)+(%f)i", a, b, c, d, x, y);
21:     div(a, b, c, d);
22:     printf("((%f)+(%f)i)/((%f)+(%f)i)=(%f)+(%f)i", a, b, c, d, x, y);
23:
24:     return 0;
25: }
26:
27: void add(double a, double b, double c, double d)
28: {
29:     x = a + c;
30:     y = b + d;
31: }
```

* グローバル変数で値の受け渡しを行なうため、retrun 文を使って戻り値を返す必要がない (void で定義する)。