

2011 年度 情報科学演習 演習 4

2011 年 6 月 1 日 (水)

演習 1 以下の (1)~(3) のプログラムを入力しなさい。さらに、コンパイルし、オブジェクトモジュールの実行しなさい。なお、「新訂 新 C 言語入門 ビギナー編」に実行結果が載っている場合は、オブジェクトモジュールの実行結果と同じになることを確認しなさい。

(1) ■ p.180 の「配列を関数に渡す 1」 (ファイル名「`exercise0401.c`」) を入力しなさい。

(2) ■ p.183 の「配列を関数に渡す 2」 (ファイル名「`exercise0402.c`」) を入力しなさい。

(3) ■ p.185 の「文字列を関数に渡す」 (ファイル名「`exercise0403.c`」) を入力しなさい。

演習 2 以下は正の整数 x, y ($x \geq y > 0$) に対する「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)~(4) 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

`exercise0404.c`

```
1: #include <stdio.h>
2:
3: int main(void)
4: {
5:     int x = 672, y = 204, t;
6:
7:     printf("gcd(%d,%d)=", x, y);
8:     while (y != 0) {
9:         t = x;
10:        x = y;
11:        y = t % x;
12:    }
13:    printf("%d\n", x);
14:
15:    return 0;
16: }
```

(1) 「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」の第8行～第12行を関数「gcd(x, y)」(ファイル名「exercise0405.c」)に書き換えなさい。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;
}
```

- 関数「gcd(x, y)」の中では、「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」の第8行～第12行をそのまま利用すること。

(2) ソースプログラム「exercise0405.c」の関数「main()」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 (例えば 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」(ファイル名「exercise0406.c」)を作成しなさい。ヒント: 関数「gcd()」を2回利用する (i.e. 「gcd(x, y, z) = gcd(x, gcd(y, z))」)。

(3) ソースプログラム「exercise0405.c」の関数「gcd()」を利用することで最小公倍数を返す関数「lcm(x, y)」を新たに追加し、「ユークリッドの互除法を用いて2変数の最小公倍数を求めるプログラム」(ファイル名「exercise0407.c」)を作成しなさい。ヒント:

- プロトタイプの宣言「int lcm(int x, int y);」を追加する (関数「gcd(x, y)」を含む)。
- 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;
}
```

(4) ソースプログラム「exercise0405.c」の関数「gcd()」を再帰的なプログラム (ファイル名「exercise0408.c」)に書き換えなさい。なお、次の「再帰的に n! を求めるプログラム」(ファイル名「exercise0409.c」)も参考にすること。ヒント:

$$\text{gcd}(x, y) = \begin{cases} x & , \quad y = 0, \\ \text{gcd}(y, x \% y), & y \neq 0 \end{cases}$$

演習 3 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)～(3) の問いに答えなさい。

● 再帰的に $n!$ を求めるプログラム exercise0409.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「**exercise0409.c**」では再帰的に関数「**func()**」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。

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

	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 (`for` 文)」(ファイル名「`exercise0410.c`」)を作成しなさい (関数にしない; 制御文の復習)。

(3) `while` 文を使って「 $n!$ を求めるプログラム (`while` 文)」(ファイル名「`exercise0411.c`」)を作成しなさい (関数にしない; 制御文の復習)。

演習 4 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 1」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。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。

● 複素数の四則演算を行なうプログラム 1 (未完成)

`exercise0412.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\n", a, b, c, d, x, y);
17:     sub(a, b, c, d);
18:     printf("((%f)+(%f)i)-((%f)+(%f)i)=(%f)+(%f)i\n", a, b, c, d, x, y);
19:     mul(a, b, c, d);
20:     printf("((%f)+(%f)i)*((%f)+(%f)i)=(%f)+(%f)i\n", a, b, c, d, x, y);
21:     div(a, b, c, d);
22:     printf("((%f)+(%f)i)/((%f)+(%f)i)=(%f)+(%f)i\n", 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、`return` 文を使って戻り値を返す必要がない (`void` で定義する)。

演習 5 次のプログラム「ハノイの塔－再帰的解法－」の第 16 行～第 18 行の空欄を埋め、プログラムを完成しなさい。なお、**実行結果**も参考にすること。

● ハノイの塔－再帰的解法－

exercise0413.c

```
1: #include <stdio.h>
2:
3: void hanoi(int n, char x, char y, char z);
4:
5: int main(void)
6: {
7:     int n = 5;
8:
9:     hanoi(n, 'A', 'B', 'C');
10:
11:     return 0;
12: }
13:
14: void hanoi(int n, char x, char y, char z)
15: {
16:     if (      ) hanoi(      ,      ,      );
17:     printf("円盤 %d を %c から %c に移す\n",      ,      ,      );
18:     if (      ) hanoi(      ,      ,      );
19: }
```

実行結果：

```
$ ./hanoi.exe 
円盤 1 を A から C に移す
円盤 2 を A から B に移す
円盤 1 を C から B に移す
円盤 3 を A から C に移す
円盤 1 を B から A に移す
... 省略 ...
円盤 3 を A から C に移す
円盤 1 を B から A に移す
円盤 2 を B から C に移す
円盤 1 を A から C に移す
```

```
$
```