C++による
アルゴリズムとプログラミング

Algorithms and Programming in C++

－　２００２年度版　－

福岡工業大学
情報工学部 情報工学科
柴田望洋

BohYoh Shibata
Fukuoka Institute of Technology

本資料について

◆　本資料は、2002 年度・福岡工業大学 情報工学部 情報工学科４年生の講義

『ソフトウェア工学』

の補助テキストとして、福岡工業大学 情報工学部 情報工学科 柴田望洋が編んだものである。

◆　参考文献・引用文献等は、資料の最後にまとめて示す。

◆　諸君が本資料をファイルに綴じやすいように、研究室の学生達（卒研生と大学院生）が時間を割いて、わざわざ穴を開けるという作業を行っている（一度のパンチで開けることのできる枚数は限られており、**気の遠くなるような**時間がかかっている）。
必ずＢ５のバインダーを用意して、きちんと綴じていただきたい。

◆　本資料のプログラムを含むすべての内容は、著作権法上の保護を受けており、著作権者である柴田望洋の許諾を得ることなく、無断で複写・複製をすることは禁じられている。

本資料は、Microsoft 社のワープロソフトウェアである Microsoft Word 2002 を用いて作成した。

★　本講義では、以下に示すホームページ上の、各ドキュメントも参照するので、参考にされたい。
柴田望洋後援会オフィシャルホームページ　http://www.BohYoh.com/

数当てゲーム

以下に示すのは、0 以上 999 以下の乱数を発生させて、その数を当てさせるゲームです。

■ じゃんけんゲーム（Ver.1.0）

```
/*
    数当てゲーム（Ver.1.0）
*/

#include  <ctime>
#include  <cstdlib>
#include  <iostream>

using namespace    std;

int main(void)
{
    const int max = 10;             // 最大入力回数
    int  x;                         // 読み込んだ値
    int  no;                        // この数を当てさせる
    int  cnt = max;                 // 残り何回入力できるか？
    time_t t;                       // 時刻

    srand(time(&t) % RAND_MAX);     // 乱数の種を初期化
    no = rand() % 1000;             // 0〜999の乱数を発生

    cout << "0〜999の数を当ててください。¥n";

    do {
        cout << "残り" << cnt << "回です。¥n";
        cout << "整数を入力せよ：";
        cin >> x;
        cnt--;

        if (x > no)
            cout << "¥a大きいです。¥n";
        else if (x < no)
            cout << "¥a小さいです。¥n";
    } while (x != no && cnt > 0);

    if (x == no) {
        cout << "正解です。¥n";
        cout << max - cnt << "回で当たりましたね。¥n";
    }

    return (0);
}
```

このプログラムは、昨年度の『情報工学ゼミナール』で示したＣ言語のプログラムを C++ に移植したものです。

さて、＜要求は進化する＞ものです。このプログラムを題材に、いろいろと考えていきましょう。たとえば、以下のような仕様変更が考えられます：

［問題 01］　このプログラムは不正解であった場合に何も表示しない。不正解であれば、

　答えは 99 でした。

と表示するように変更せよ。

［問題 02］このプログラムは、0 未満や 1000 以上など、不正な値の入力を無条件に受け付ける。入力された値をチェックし、不正な値が入力された場合は、再入力させるようにせよ。

［問題 03］　ゲーム終了時に、『もう一度やりますか（Yes…1／No…0）：』と表示し、1 が入力されたら、再びゲームできるように変更せよ。

ただし、ゲームは 10 回までとする（1 回のゲームを「ステージ」と呼ぶことにしよう）。

［問題 04］　上記の問いに対して、0 が入力されたら、各ステージが、何回で成功したか（あるいは不正解であったか）を、以下のように表示するように変更せよ。

Stage 1　☆　☆　☆

Stage 2　★　★　★　★　★　★　★　★　★　★

Stage 3　☆　☆　☆　☆　☆

Stage 4　☆　☆　☆　☆　☆　☆　☆　☆　☆　☆

※第 1 ステージは 3 回で成功、第 2 ステージは 10 回でも不正解、第 3 ステージは 5 回で成功、第 4 ステージは 10 回で成功したものとする。また、この場合は、第 4 ステージまでが行われたものである。

［問題 05］　問題 02 では、ゲーム回数の上限を 10 と限定することによって、成功回数を記憶させるための変数を、単なる 1 次元配列に格納できた。

上限回数を限定させないためには、どのような工夫が必要であるか考察せよ。

例）ファイルに点数を保存する 。線形リスト構造によるバッファを用意するなど。

［問題 06］　ゲーム回数に上限をもたせない場合でも、最後に行われた 10 回のゲームの点数のみを表示する仕様とすれば、成功回数を記憶させるための変数は、単なる 1 次元配列に格納できる。そのようにプログラムを改良せよ。

[問題 06]を解く前に、少し簡単化したプログラムを例に考えましょう。以下に示すのは、最大 10 個の整数値を読み込んで、要素数が 10 である配列に、その値を格納するプログラムです。

■最大 10 個の値を読み込んで要素数が 10 である配列に格納

```
/*
    最大10個の値を読み込んで要素数が10である配列に格納
*/

#include  <iomanip>
#include  <iostream>

using namespace    std;

int main(void)
{
    const int max = 10;          // 配列の要素数
    int   a[max];                // 配列
    int  x;                      // 読み込んだ値
    int  cnt;                    // 読み込んだ個数

    cout << "整数を入力してください。¥n";
    cout << "入力できるのは最大で" << max << "個です。¥n";

    cnt = 0;
    while (1) {
        int  retry;              // もう一度？

        cout << cnt + 1 << "個目の整数：";
        cin >> a[cnt++];

        if (cnt == max)
            break;

        cout << "続けますか（Yes…1／No…0）？";
        cin >> retry;
        if (retry == 0) break;
    }

    for (int i = 0; i < cnt; i++)
        cout << setw(2) << i + 1 << " : " << a[i] << '¥n';

    return (0);
}
```

このプログラムの実行例を示します。「続けますか」と訊ねられた際に 0 を入力すると、その時点で読み込みは終了します。また、10 個の整数を読み込むと、自動的に読み込みは終了します。

実行例（１）

```
整数を入力してください。
入力できるのは最大で10個です。
1個目の整数：15
続けますか（Yes…1／No…0）？1
2個目の整数：31
続けますか（Yes…1／No…0）？1
3個目の整数：64
続けますか（Yes…1／No…0）？0
 1 : 15
 2 : 31
 3 : 64
```

実行例（２）

```
整数を入力してください。
入力できるのは最大で10個です。
1個目の整数：16
続けますか（Yes…1／No…0）？1
2個目の整数：923
続けますか（Yes…1／No…0）？1
3個目の整数：254
続けますか（Yes…1／No…0）？1
4個目の整数：1
続けますか（Yes…1／No…0）？1
5個目の整数：35
続けますか（Yes…1／No…0）？1
6個目の整数：268
続けますか（Yes…1／No…0）？1
7個目の整数：6547
続けますか（Yes…1／No…0）？1
8個目の整数：1234
続けますか（Yes…1／No…0）？1
9個目の整数：25
続けますか（Yes…1／No…0）？1
10個目の整数：35
 1 : 16
 2 : 923
 3 : 254
 4 : 1
 5 : 35
 6 : 268
 7 : 6547
 8 : 1234
 9 : 25
10 : 35
```

それでは、このプログラムを、[問題 06]風に拡張してみてください。すなわち、任意の個数を読み込めるようにします。そして、読み込んだ個数が 10 を超えた場合は、最後の 10 を表示するようにしましょう。

※すなわち 25 個読み込んだ場合は、16 個目から 25 個目の 10 個を表示します。

プログラムは次のようになります。

■値を読み込んで要素数が 10 である配列に最後の 10 個を格納

```
/*
    値を読み込んで要素数が10である配列に最後の10個を格納
*/

#include  <iomanip>
#include  <iostream>

using namespace  std;

int main(void)
{
    const int max = 10;          // 配列の要素数
    int  a[max];                 // 配列
    int  x;                      // 読み込んだ値
    int  cnt;                    // 読み込んだ個数

    cout << "整数を入力してください。¥n";
    cout << "入力できるのは最大で" << max << "個です。¥n";

    cnt = 0;
    while (1) {
        int  retry;           // もう一度？

        cout << cnt + 1 << "個目の整数：";
        cin >> a[(cnt++) % max];

        cout << "続けますか（Yes…1／No…0）？ ";
        cin >> retry;
        if (retry == 0) break;
    }

    int i = cnt - max;
    if (i < 0) i = 0;

    for ( ; i < cnt; i++)
        cout << setw(2) << i + 1 << " : " << a[i % max] << '¥n';

    return (0);
}
```

読み込んだ値が 10 個以下であれば、配列の先頭から順に格納するだけです。たとえば、三つ読み込んだ場合は、以下のようになります。

1	2	3							
15	32	64							

もちろん、10 個読み込んだ場合は、次のようになります。

1	2	3	4	5	6	7	8	9	10
15	32	64	57	99	21	0	23	24	55

さて、11 個目を読み込むためには、以下に示すように、配列の要素を 1 個ずつ先頭側にずらすという手もあります。

1	2	3	4	5	6	7	8	9	10
15	32	64	57	99	21	0	23	24	55

↓

2	3	4	5	6	7	8	9	10	**11**
32	64	57	99	21	0	23	24	55	**97**

しかし、これだと、これ以降毎回ずらさなければならず効率が悪くなります。このプログラムでは、配列を循環しているものとみなし、11 個目のデータを先頭に格納します。

11	2	3	4	5	6	7	8	9	10
97	32	64	57	99	21	0	23	24	55

もちろん、12 個目のデータは、2 番目に格納されることになります。

11	**12**	3	4	5	6	7	8	9	10
97	**32**	64	57	99	21	0	23	24	55

最後に表示する際は、読み込んだ個数 cnt が

10 個以下の場合：a[0]〜a[cnt - 1]を表示します。

11 個以上の場合：a[cnt - 10]〜a[cnt - 1]を表示します。

短期記憶力チェックプログラム

以下に示すのは、表示されてすぐに消える２桁の数字を瞬時に記憶して、それに 1 を加えた値を表示するプログラムです。

昨年度の『情報工学ゼミナール』で示したＣ言語のプログラムを C++に移植して、若干の改良をほどこしたものです。

■　２桁の数に１を加えた値を当てる（最後の 5 ステージのポイントを記録）

```
/*
    ２桁の数に１を加えた値を当てる（最後の5ステージのポイントを記録）
*/

#include  <ctime>
#include  <cstdlib>
#include  <iomanip>
#include  <iostream>

using namespace    std;

//--- xミリ秒経過するのを待つ ---//
int sleep(unsigned long x)
{
    clock_t  s = clock();
    clock_t  c;
    do {
        if ((c = clock()) == (clock_t)-1)       // エラー
            return (0);
    } while (1000UL * (c - s) / CLOCKS_PER_SEC <= x);
    return (1);
}

int main(void)
{
    const int  pmax = 5;          // 点数を記録するステージ数
    int  point[pmax];             // 点数
    int  level;                   // レベル
    int  no[5];                   // この数を記憶させる
    int  x[5];                    // 読み込んだ値

    time_t t;                     // 時刻

    srand(time(&t) % RAND_MAX);      // 乱数の種を初期化

    cout << "２桁の数値を記憶して１を加えた値を入力しましょう。¥n";
```

```
    do {
        cout << "挑戦するレベル（2〜5）：";
        cin >> level;
    } while (level < 2  ||  level > 5);

    int  success = 0;                   // 成功回数
    int  stage = 0;                     // ステージ

    cout << "1を加えた値を順に入力してください。¥n";

    while (1) {
        int  retry;                     // もう一度？
        int  seikai = 0;                // このステージでの成功数

        cout << "¥r第" << stage + 1 << "ステージ!!¥n";

        for (int i = 0; i < level; i++) {
            no[i] = rand() % 90 + 10;    // 10〜99の乱数を発生
            cout << no[i] << ' ';
        }

        sleep(level * 200);

        cout << '¥r';
        for (i = 0; i < level; i++) {
            cout << i + 1 << "番目の数：";
            cin >> x[i];
        }

        for (i = 0; i < level; i++) {    // 正誤を表示しながらカウント
            if (x[i] != no[i] + 1)
                cout << "x ";
            else {
                cout << "○ ";
                seikai++;
            }
        }
        cout << '¥n';
        for (i = 0; i < level; i++)      // 正解を表示
            cout << no[i] << ' ';

        cout << "  …  " << seikai << "個正解です。¥n";
        point[(stage++) % pmax] = seikai;

        cout << "続けますか（Yes…1／No…0）？";
        cin >> retry;
        if (retry == 0) break;
    }
```

```
    for (int i = (stage < pmax - 1) ? 0 : stage - pmax; i < stage; i++)
        cout << setw(2) << i + 1 << "ステージ："
                        << point[i % pmax] << '¥n';

    return (0);
}
```

このプログラムを実行すると、まずレベルを尋ねられます。変数 level に読み込んだ値が 2 以上 5 以下でなければ、do 文が繰り返され再入力が促されます。

2桁の値が表示されて直ぐに（0.2×level秒で）消えます。それぞれに１を加えた値を入力します。

実行例（１）

```
２桁の数値を記憶して１を加えた値を入力
挑戦するレベル（2〜5）：3
１を加えた値を順に入力してください。
第1ステージ!!
22 52 31                    ← この行は表示後に消えます。
1番目の数：22
2番目の数：53
3番目の数：52
○ ○ ×
21 52 31   …  2個正解です。
続けますか（Yes…1／No…0）？1
第2ステージ!!

（中略）

続けますか（Yes…1／No…0）？0
 5ステージ：2
 6ステージ：1
 7ステージ：0
 8ステージ：3
 9ステージ：2
```

各ステージで何個成功したかがポイントとします。

pmax は、何ステージ分のポイントを記憶するかを表すための定数です。このプログラムでは、5 ステージ分のポイントを記憶することになっています。

＊

さて、この値を変数にして任意のステージ分のポインタを記録するようにしましょう。さらに、6 レベル以上にも挑戦できるようにしましょう。

そのためには、プログラム実行時に配列の要素数を決定できるようにしなければなりません。

■　２桁の数に１を加えた値を当てる

（ポイント記録ステージおよびレベルを実行時に決定）

```
/*
    ２桁の数に１を加えた値を当てる（ポイント記録ステージおよびレベルを実行時に決定）
*/

#include  <ctime>
#include  <cstdlib>
#include  <iomanip>
#include  <iostream>

using namespace    std;

//--- xミリ秒経過するのを待つ ---//
int sleep(unsigned long x)
{
       // ---  (…省略…)  ---//
}

int main(void)
{
    int  pmax;                    // 点数を記録するステージ数
    int  level;                   // レベル

    time_t t;                     // 時刻

    srand(time(&t) % RAND_MAX);      // 乱数の種を初期化

    cout << "２桁の数値を記憶して１を加えた値を入力しましょう。¥n";

    do {
        cout << "挑戦するレベル（2〜5）：";
        cin >> level;
    } while (level < 2  ||  level > 5);

    int  *no = new int[level];        // この数を記憶させる
    int  *x = new int[level];         // 読み込んだ値

    cout << "何ステージ分のポイントを記録しますか？";
    do {
        cin >> pmax;
    } while (pmax < 0);
    int *point = new int[pmax];  // 点数

    int  success = 0;          // 成功回数
    int  stage = 0;            // ステージ

    cout << "１を加えた値を順に入力してください。¥n";
```

```
    while (1) {

        // ---  (…省略…)  ---//

    }
    delete[] no;
    delete[] x;

    for (int i = (stage < pmax - 1) ? 0 : stage - pmax; i < stage; i++)
        cout << setw(2) << i + 1 << "ステージ："
                    << point[i % pmax] << '¥n';

    return (0);
}
```

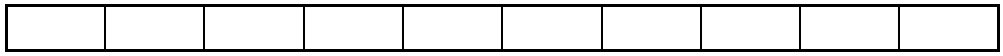
ここで利用しているのが、**new** 演算子と **delete[]**演算子です。一般に、要素数が *n* で、要素型が *Type* 型である配列 a を動的に生成するには、

Type * a = **new** *Type*[*n*];

とします。また、配列を利用して不要になった場合の解放は、次のようにします。

delete[] a;

↓　int *a = new int[10];



↓　delete[] a;

［問題 01］　満点であったステージを pmax 個だけ（該当するステージが pmax を超える場合は最後の pmax 個）を配列 fsno に記憶させよ。たとえば pmax が 3 でレベルが 5 で、各ステージのポイントが順に、

5, 4, 3, 5, 5, 2, 5, 4, 3

であれば、配列 point と配列 fsno は以下のようになる。

配列 point ： 先頭から順に 5　4　3　　※最後の 3 ステージの点数

配列 fsno　： 先頭から順に 4　5　7　　※5 点だったステージ番号

相異なる四つの数字の並びを当てる

相異なる四つの数字の並びを当てさせるプログラムを作りましょう。プログラムは以下のように動作するものとします。

実行例（１）

```
■ 四つの数字の並びを当ててください。
■ 同じ数字が複数含まれることはありません。
■ たとえば4307のように連続して入力してください。
■ スペース文字などを入力してはいけません。

入力してください：1A57
数字以外の文字を入力しないでください。
入力してください：1234
    入力された数字はまったく含まれません。

入力してください：5678
    入力された数字中2個の数字が含まれます。
    ただし位置もあっている数字はありません。

入力してください：5679
    入力された数字中2個の数字が含まれます。
    ただし位置もあっている数字はありません。

入力してください：5670
    入力された数字中2個の数字が含まれます。
    ただし位置もあっている数字はありません。

入力してください：7890
    入力された数字中4個の数字が含まれます。
    その中の1個は位置もあっています。

入力してください：7980
    入力された数字中4個の数字が含まれます。
    ただし位置もあっている数字はありません。

入力してください：7809
    入力された数字中4個の数字が含まれます。
    その中の2個は位置もあっています。

入力してください：9807
正解です。!!
```

■　相異なる四つの数字の並びを当てさせる

```
/*
    相異なる四つの数字の並びを当てる
*/

#include  <ctime>
#include  <cctype>
#include  <iostream>
#include  <cstdlib>

using namespace    std;

//--- 相異なる四つの数字の並びを生成して配列xに格納 ---//
void make4digits(int x[])
{
    int  i, j, val;

    for (i = 0; i < 4; i++) {
        do {
            val = rand() % 10;           // 0〜9の乱数
            for (j = 0; j < i; j++)      // その数が既に得られているか
                if (val == x[j])
                    break;
        } while (j < i);
        x[i] = val;
    }
}

//--- 入力された文字列sの妥当性をチェック ---//
int check(const char s[])
{
    int  i, j;

    for (i = 0; i < 4; i++) {
        if (!isdigit(s[i]))
            return (1);                  // 数字以外の文字が含まれている
        for (j = 0; j < i; j++)
            if (s[i] == s[j])
                return (2);              // 同一の数字が含まれている
    }
    if (s[i] != '¥0')
        return (3);                      // 文字数が4を超えている
    return (0);
}

/*--- 判定結果を表示 ---*/
void print_result(int snum, int spos)
{
```

```
    if (spos == 4)
        cout << "正解です。!!";
    else if (snum == 0)
        cout << "    入力された数字はまったく含まれません。¥n";
    else {
        cout << "    入力された数字中" << snum << "個の数字が含まれます。¥n";
        if (spos == 0)
            cout << "    ただし位置もあっている数字はありません。¥n";
        else
            cout << "    その中の" << spos << "個は位置もあっています。¥n";
    }
    putchar('¥n');
}

int main(void)
{
    int   i, j;
    int   chk;          // 入力された文字列のチェック結果
    int   snum;         // 当たっている数字の個数
    int   spos;         // 位置も当たっている数字の個数
    int   no[4];        // 当てる数字の並び
    char  buff[10];       // 読み込む数字の並びを格納する文字列
    time_t t;           // 時刻

    srand(time(&t) % RAND_MAX);      // 乱数の種を初期化

    puts("■ 四つの数字の並びを当ててください。");
    puts("■ 同じ数字が複数含まれることはありません。");
    puts("■ たとえば4307のように連続して入力してください。");
    puts("■ スペース文字などを入力してはいけません。¥n");

    make4digits(no);              // 相異なる四つの数字の並びを生成

    do {
        do {

            cout << "入力してください：";
            cin >> buff;          // 読み込む
            chk = check(buff);        // 読み込んだ文字列をチェック

            switch (chk) {
             case 1: cout << "数字以外の文字を入力しないでください。¥n"; break;
             case 2: cout << "同一の数字を複数入力しないでください。¥n"; break;
             case 3: cout << "４文字以上入力しないでください。¥n"; break;
            }
        } while (chk != 0);

        snum = spos = 0;
```

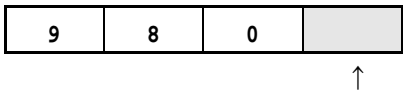
```
        for (i = 0; i < 4; i++) {
            if (buff[i] == '0' + no[i]) {       // 照合成功
                spos++;
                snum++;
            } else {
                for (j = 0; j < 4; j++) {       // 含まれるか
                    if (buff[i] == '0' + no[j])
                        snum++;
                }
            }
        }

        print_result(snum, spos);       // 判定結果を表示

    } while (spos < 4);

    return (0);
}
```

各関数を簡単に解説します。

make4digis

配列 x の四つの要素 x[0], x[1], x[2], x[3]に相異なる 0〜9 の数値を代入します。

代入するのは乱数です。ただし、既に発生した乱数を重複して代入するわけにはいきませんから、そのためのチェックを行っています。



たとえば、x[3]への代入を行う際は、9, 8, 0 以外の乱数が出るまで、必要であれば何度も乱数を発生さることになります。

check

このゲームでは、当てさせるべき数字の並びとして 0935 のように、先頭が 0 で始まる可能性がありますので、整数値として読み込むことはできません(935 となってしまいます)。

そのため、読込みは、文字列で行っています。読み込まれた文字列の妥当性をチェックします。

具体的には、数字以外の文字が含まれていないか、同一の数字が複数含まれていないか、4 文字以上含まれていないかをチェックします。

print_result

ユーザの入力した数の判定結果を表示します。

当てさせるべき数字の並び

9	8	0	7

入力された数字の並び

9	0	2	3

であれば、二つの数が含まれており、その内の一つが位置もあっています。

変数 spos は、値も位置も正しい数字の個数、snum は、位置は不正であるが値は正しい数字の個数です。これをもとに判定結果を表示します。

なお、判定を行うのは、main 関数中の for 文です。

演習

このプログラムをもとに、応用的なゲームを作成しよう。

※ヒント機能をつける。

- 最初の 1 文字を教えてあげる。
- 合っている数の中で最も先頭側の 1 文字を教えてあげる。
- ヒントは 2 回まで　…　等。

※数字の重複を許す。

※４以外の桁数とする。

※数字でなくアルファベット文字とする。

Etc…

カレンダーを表示する

カレンダーを表示するプログラムを作りましょう。

■　カレンダーを表示する

```
//
//  カレンダー表示
//

#include  <iomanip>
#include  <iostream>

using namespace  std;

//--- 各月の日数 ---//
int mday[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

//--- year年month月day日の曜日を求める ---//
int dayofweek(int year, int month, int day)
{
    if (month == 1  ||  month == 2) {
        year--;
        month += 12;
    }
    return ((year + year/4 - year/100 + year/400 + (13*month+8)/5 + day)  % 7);
}

//--- year年は閏年か？（0･･･非閏年 / 1･･･閏年）---//
int isLeap(int year)
{
    return (year % 4 == 0 && year % 100 != 0 || year % 400 == 0);
}

//--- year年month月の日数（28～31）---//
int monthdays(int year, int month)
{
    if (month-- != 2)                    // monthが２月でないとき
        return (mday[month]);
    return (mday[month] + isLeap(year));    // monthが２月であるとき
}

//--- y年m月カレンダーを表示 ---//
void put_calendar(int y, int m)
{
    int  i;
    int  wd = dayofweek(y, m, 1);        /* y年m月の曜日 */
    int  mdays = monthdays(y, m);        /* y年m月の日数 */
```

```
    cout << " 日 月 火 水 木 金 土 ¥n";
    cout << "----------------------¥n";

    for (i = 0; i < wd; i++)
        cout << "   ";

    for (i = 1; i <= mdays; i++) {
        cout << setw(3) << i;
        if (++wd % 7 == 0)
            cout << '¥n';
    }
    cout << '¥n';
}

int main(void)
{
    int y, m;

    cout << "年：";       cin >> y;
    cout << "月：";       cin >> m;

    put_calendar(y, m);

    return (0);
}
```

実行例

```
年：2002
月：7
 日 月 火 水 木 金 土
----------------------
     1  2  3  4  5  6
  7  8  9 10 11 12 13
 14 15 16 17 18 19 20
 21 22 23 24 25 26 27
 28 29 30 31
```

画面の横幅が 80 桁程度あるのであれば、３か月分のカレンダーを横に並べて表示できます。そのようなプログラムを作ってみましょう。

なお、Ｃ言語に移植しやすいように、Ｃ＋＋特有の機能は、極力利用していません。

■　カレンダーを横に最大３個並べて表示する

```
///
//  カレンダー表示
//

#include  <cstdlib>
#include  <iomanip>
#include  <iostream>

using namespace  std;

//--- 各月の日数 ---//
int mday[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

//--- year年month月day日の曜日を求める ---//
int dayofweek(int year, int month, int day)
{
    if (month == 1  ||  month == 2) {
        year--;
        month += 12;
    }
    return ((year + year/4 - year/100 + year/400 + (13*month+8)/5 + day)  % 7);
}

//--- year年は閏年か？（0･･･非閏年 / 1･･･閏年）---//
int isLeap(int year)
{
    return (year % 4 == 0 && year % 100 != 0 || year % 400 == 0);
}

//--- year年month月の日数（28～31）---//
int monthdays(int year, int month)
{
    if (month-- != 2)                    // monthが２月でないとき
        return (mday[month]);
    return (mday[month] + isLeap(year));    // monthが２月であるとき
}

//--- y年m月カレンダーを文字列に格納 ---//
void make_calendar(int y, int m, char s[7][24])
{
    int  k;
    int  wd = dayofweek(y, m, 1);        /* y年m月の曜日 */
```

```
    int  mdays = monthdays(y, m);        /* y年m月の日数 */
    char  tmp[4];

    sprintf(s[0], "      %04d年%02d月      ", y, m);

    for (k = 1; k < 7; k++)
        s[k][0] = '¥0';

    k = 1;
    sprintf(s[k], "%*s", 3 * wd, "");

    for (int i = 1; i <= mdays; i++) {
        sprintf(tmp, "%3d", i);
        strcat(s[k], tmp);
        if (++wd % 7 == 0)
            k++;
    }

    for (wd %= 7 ; wd < 7; wd++)
        strcat(s[k], "   ");

    while (++k < 7)
        sprintf(s[k], "%21s", "");
}

//--- sに格納されたカレンダーを横にn個並べて表示 ---*/
void print(char s[3][7][24], int n)
{
    for (int i = 0; i < n; i++)                // 年・月を表示
        cout << s[i][0] << "  ";
    cout << '¥n';

    for (int i = 0; i < n; i++)
        cout << " 日 月 火 水 木 金 土   ";
    cout << '¥n';

    for (int i = 0; i < n; i++)
        cout << "----------------------  ";
    cout << '¥n';

    for (int i = 1; i < 7; i++) {
        for (int j = 0; j < n; j++)
            cout << s[j][i] << "   ";
        cout << '¥n';
    }
    cout << '¥n';
}
```

```
//--- y1年m1月～y2年m2月のカレンダーを表示 ---//
int put_calendar(int y1, int m1, int y2, int m2)
{
    int  n = 0;
    int  y = y1;
    int  m = m1;
    char cal[3][7][24];

    while (y <= y2) {
        if (y == y2 && m > m2) break;
        make_calendar(y, m, cal[n++]);
        if (n == 3) {
            print(cal, n);
            n = 0;
        }
        m++;
        if (m == 13 && y < y2) {
            y++;  m = 1;
        }
    }
    if (n)
        print(cal, n);
}

int main(void)
{
    int y1, m1, y2, m2;

    int  x;
    cout << "開始年月を入力せよ。¥n";
    cout << "年：";       cin >> y1;
    cout << "月：";       cin >> m1;

    cout << "終了年月を入力せよ。¥n";
    cout << "年：";       cin >> y2;
    cout << "月：";       cin >> m2;

    x = put_calendar(y1, m1, y2, m2);

    return (0);
}
```

プログラムの実行例を次ページに示します。

なお、このプログラムは、開始年月と終了年月の妥当性（終了年月の方が後かどうか）はチェックしていません。

```
開始年月を入力せよ。
年：2002
月：7
終了年月を入力せよ。
年：2003
月：4
      2002年07月                2002年08月                2002年09月
 日 月 火 水 木 金 土    日 月 火 水 木 金 土    日 月 火 水 木 金 土
----------------------  ----------------------  ----------------------
     1  2  3  4  5  6                  1  2  3     1  2  3  4  5  6  7
  7  8  9 10 11 12 13     4  5  6  7  8  9 10     8  9 10 11 12 13 14
 14 15 16 17 18 19 20    11 12 13 14 15 16 17    15 16 17 18 19 20 21
 21 22 23 24 25 26 27    18 19 20 21 22 23 24    22 23 24 25 26 27 28
 28 29 30 31             25 26 27 28 29 30 31    29 30


      2002年10月                2002年11月                2002年12月
 日 月 火 水 木 金 土    日 月 火 水 木 金 土    日 月 火 水 木 金 土
----------------------  ----------------------  ----------------------
        1  2  3  4  5                     1  2     1  2  3  4  5  6  7
  6  7  8  9 10 11 12     3  4  5  6  7  8  9     8  9 10 11 12 13 14
 13 14 15 16 17 18 19    10 11 12 13 14 15 16    15 16 17 18 19 20 21
 20 21 22 23 24 25 26    17 18 19 20 21 22 23    22 23 24 25 26 27 28
 27 28 29 30 31          24 25 26 27 28 29 30    29 30 31


      2003年01月                2003年02月                2003年03月
 日 月 火 水 木 金 土    日 月 火 水 木 金 土    日 月 火 水 木 金 土
----------------------  ----------------------  ----------------------
           1  2  3  4                     1                        1
  5  6  7  8  9 10 11     2  3  4  5  6  7  8     2  3  4  5  6  7  8
 12 13 14 15 16 17 18     9 10 11 12 13 14 15     9 10 11 12 13 14 15
 19 20 21 22 23 24 25    16 17 18 19 20 21 22    16 17 18 19 20 21 22
 26 27 28 29 30 31       23 24 25 26 27 28       23 24 25 26 27 28 29
                                                 30 31


      2003年04月
 日 月 火 水 木 金 土
----------------------
        1  2  3  4  5
  6  7  8  9 10 11 12
 13 14 15 16 17 18 19
 20 21 22 23 24 25 26
```

■演習

開始年月と終了年月の妥当性をチェックせよ。