

ソフト系 C言語実習課題 2

祝日(春分の日、秋分の日も含む)を入れた 万年カレンダー

V3.06

本課題は、4つのSetpに分けて実習します。

■ 実習を通じて、StepごとにC言語の基礎と下記のことを学びます。

－Step1:

- ◆ 与えられた、年の1月1日の曜日を求める
- ◆ 自分の誕生日(年月日)を入力して、誕生日の曜日を求める
 - 1月～12月までの各月の日数を定義(31,28,31,30,31,30,31,31,30,31,30,31)
 - うるう年の判定をして、2月の日数を28日か29日に変更
 - 該当する年の1月から誕生日の前月までの月の日数を足し、最後に誕生日を足す。

－Step2:

- ◆ 表示したい西暦の年だけを入力
- ◆ 3次元配列を定義し、与えられた年の1年間分のカレンダーを作成する。

－Step3:

- ◆ 春分、秋分の日を求める
- ◆ 祝日マークを加える。
- ◆ ハッピーマンデー(Happy Monday)の処理をする。

－Step4:

- ◆ 横2ヶ月、縦6ヶ月、あるいは横3ヶ月、縦4ヶ月のカレンダーを表示する

Step1：指定した西暦の年月日の曜日を求める

■ Step1で学ぶこと

－与えられた、年の1月1日の曜日を求める

－自分の誕生日の曜日を求める

◆ 月データのテーブルを作成する

➢ 1月～12月まで大の月、小の月の定義テーブル

➢ 定義例: `char mdays[N_MONTH]={31,28,31,30,31,30,31,31,30,31,30,31};`

◆ うるう年の判定をして、2月の日にちを28日か29日かを確定

－enumの活用

◆ enumは、連続した数字の定義に使用する

◆ 月と曜日はenumで定義して使用すると、プログラムの可読性がよくなる。

◆ 定義例:

➢ `enum M_LIST {JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC, N_MONTH};`

➢ `enum W_LIST {SUN, MON, TUE, WED, THU, FRI, SAT, N_WEEK};`

◆ 実際のカレンダーのプログラムでは、enumで定義した名前を使う

➢ そのため、下記の例ように、プログラムプログラム中に数字はほとんど登場しない

➢ `for(month = JAN ; month <= DEC ; month++)`

カレンダーの基本(元旦の曜日を求める)

■ ユリウス(Julian)暦とグレゴリオ暦

– 現在、多くの国では、グレゴリオ暦を使用

■ グレゴリオ暦で、曜日を求める

– 1. 西暦1年の1月1日から、求める年-1年の12/31日までの日数を求める

– 2. グレゴリオ暦では、下記のように求める。

(1)~(4)の処理は、求める年-1を別名の変数に代入したら、括弧もいない1行の式で書けるよ

◆ (1) 求める年-1に、365日を掛け、累積日数を求める

◆ (2) 求める年-1を4で割った数を、累積日数に足す → 4で割り切れたらうるう年

◆ (3) 求める年-1を100で割った数を、累積日数から引く → 100で割り切れたら平年

◆ (4) 求める年-1を400で割った数を、累積日数に足す → 400で割り切れたらうるう年

◆ 上記(1)~(4)の処理で、求める年の前年の12/31日までの累積日数が求まる

◆ その累積日数に、1を加えると、求める年の1/1日までの日数になる

– 3. 求めた累積日数を7で割り余りが曜日(西暦1年1月1日は、月曜日)

◆ 0=日、1=月、2=火、3=水、4=木、5=金、6=土

西暦1年から計算をする、累積日数が16ビットで足りないため、EXCELでは、1900年を基準年として計算をしている。そのため、EXCELでは、1900年より前の年は正しく計算されない。

月日数と日本の祝日 および 1月1日の曜日確認表

月	月日数	月合計	祝日1	祝日2	祝日3
1月	31	0	1 元旦	第2月曜 (成人の日)	
2月	28	31	11 (建国記念日)	23 (令和天皇誕生日)	
3月	31	59	21日頃 (春分の日)		
4月	30	90	29 (昭和の日)		
5月	31	120	3 (憲法記念日)	4 (緑の日)	5 (子供の日)
6月	30	151	なし		
7月	31	181	第3月曜日 (海の日)		
8月	31	212	11 (山の日)		
9月	30	243	第3月曜日 (敬老の日)	23日頃 (秋分の日)	
10月	31	273	第2月曜 (体育の日)		
11月	30	304	3 (文化の日)	23 (勤労感謝の日)	
12月	31	334	23 (平成天皇誕生日)		

- ・祝日と祝日に挟まれた日は、休日
- ・春分、秋分の日は、別途計算で求める

西暦	数値	曜日	
1995	0	日	1995年1月1日(日)
1996	1	月	1996年1月1日(月)
1997	3	水	1997年1月1日(水)
1998	4	木	1998年1月1日(木)
1999	5	金	1999年1月1日(金)
2000	6	土	2000年1月1日(土)
2001	1	月	2001年1月1日(月)
2002	2	火	2002年1月1日(火)
2003	3	水	2003年1月1日(水)
2004	4	木	2004年1月1日(木)
2005	6	土	2005年1月1日(土)
2006	0	日	2006年1月1日(日)
2007	1	月	2007年1月1日(月)
2008	2	火	2008年1月1日(火)
2009	4	木	2009年1月1日(木)
2010	5	金	2010年1月1日(金)
2011	6	土	2011年1月1日(土)
2012	0	日	2012年1月1日(日)
2013	2	火	2013年1月1日(火)
2014	3	水	2014年1月1日(水)
2015	4	木	2015年1月1日(木)
2016	5	金	2016年1月1日(金)
2017	0	日	2017年1月1日(日)
2018	1	月	2018年1月1日(月)
2019	2	火	2019年1月1日(火)
2020	3	水	2020年1月1日(水)
2021	5	金	2021年1月1日(金)
2022	6	土	2022年1月1日(土)
2023	0	日	2023年1月1日(日)
2024	1	月	2024年1月1日(月)
2025	3	水	2025年1月1日(水)

誕生日の曜日を求める

■ 求める日が1月1日以降の場合

– 求める月の前月のまでの累計日数を加える

◆ (2月以降は求める年のうるう年判定が必要)。

– その後、求める日を加えて、該当年月日までの累積日数を求める

■ うるう年の判定 (うるう年判定の関数を作成)

– 判定する年が、4で割り切れたらうるう年

– 判定する年が、100で割り切れたら平年

– 判定する年が、400で割り切れたらうるう年

西暦2000年は、400年に一度の
左記のすべての条件が成立した年

2月末日の値を変更する例

```
enum M_LIST {JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC, N_MONTH};
```

```
enum W_LIST {SUN, MON, TUE, WED, THU, FRI, SAT, N_WEEK};
```

```
char mdays[N_MONTH]={31,28,31,30,31,30,31,31,30,31,30,31}; ← グローバル変数にしても良い
```

```
mdays[FEB] = うるう年判定(year) ? 29 : 28;
```

3項演算子(条件演算子)

条件演算子を使うと、右の
if文が1行で表現できる。

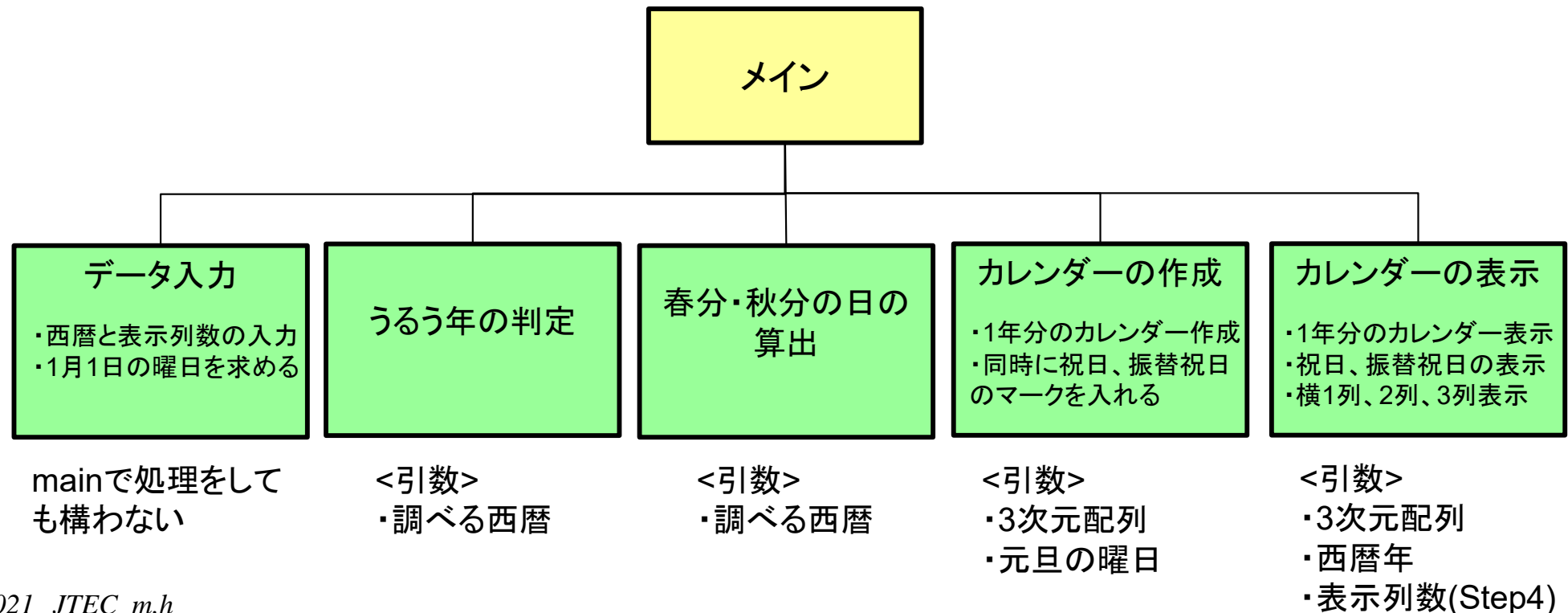
```
if ( うるう年判定(year) == true )  
    mdays[FEB] =29;  
else  
    mdays[FEB] =28
```

Step2 1年間のカレンダーの作成

プログラム構造

■ カレンダーのプログラム構造の例

- データ入力は、メインの中でやってもOK。
- 単にカレンダーの作成であれば、1月1日の曜日が分かれば作成できる
 - ◆ 作成する年の1月1日(元旦)を曜日を引数で渡す(年は不要)。
 - ◆ メインでループをすれば、何年分のカレンダーも作成ができる
 - ◆ うるう年判定、春分・秋分の日算出は、毎年必要なので関数にする



1ヶ月分の2次元配列のカレンダーテーブル

6行 x 7列の2次元配列

	日	月	火	水	木	金	土
0					1	2	3
1	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17
3	18	19	20	21	22	23	24
4	25	26	27	28	29	30	31
5							
	0	1	2	3	4	5	6
enumの定義値	SUN	MON	TUE	WED	THU	FRI	SAT

曜日は、配列に不要

赤枠の範囲が実際の配列

Step2: 1年分のカレンダーを作成


■ Step2で学ぶこと

– 多次元配列の定義

- ◆ 1年間分のカレンダーを入れる多次元配列
 - 3次元配列で作成する
- ◆ 最初に、3次元配列の中をすべて0でクリアする ← main関数で初期化をする
- ◆ 通常3次元配列の操作は3重ループで行うが、カレンダーの作成は2重ループで行う。
 - 一番外側は、月のループ。その内側は、1日から月末までの日にちのループ
 - 日にちのループの中で、日にちを3次元配列に代入し、その後、曜日を更新
 - 曜日が土曜日を越えたら行を更新する

– 配列の内容を出力(画面に表示)

- ◆ 縦12ヶ月で表示する
- ◆ 3次元配列の出力は、単純にfor文の3重ループ(すべての要素を出力する)
- ◆ そのとき、配列の中が0(日付が入っていない)の場合、空白を出力する



黒いコンソール画面の横幅のデフォルトは80文字になっているため、横3ヶ月の表示をすると横幅が超えてしまう。

そこで、コンソール画面の横幅を100文字程度に広げる必要がある。

変更方法は、後述の「付録:コンソール画面の大きさを変更する」を参照こと。

カレンダー作成のヒント

■ 2重ループでカレンダーを作成する

ー 番外側は、1月～12月までのループ

```
for(month=JAN ; month<=DEC ; month++){
```

ここから新しい月が始まる(月初めにすることは、**行を0**に、**日付を1日**にする)

ー 次のループは、日にちが1日～月末になるまでループ

- ◆ 元旦の曜日の位置に、日付(最初は1)を入れる(曜日はmainから引数で受け取る)
- ◆ 日にちと曜日をそれぞれ+1する
- ◆ 曜日が土曜日を超えたら、行を+1して、曜日を日曜日にする。

3次元配列は、通常3重ループで操作するが、
カレンダーの作成では、2重ループで処理をする

	④			①	②	----->	③
	↓			↓	↓		
	0	1	2	3	4	5	6
	日	月	火	水	木	金	土
→ 0行目				1	2	3	4
⑤ → 1行目	5	---	---	---	---	---	→
→ 2行目	---	---	---	---	---	---	→
→ 3行目	---	---	---	---	---	---	→
→ 4行目	---	---	---	---	→	31	
5行目							

曜日が水曜日(3)の場合、

① 曜日の位置に日付を入れる

② 日付と曜日をそれぞれ+1する

上記、①、②を繰り返す

③ 曜日が土曜を超えたら(曜日>SAT)、

④曜日を日曜日戻す(曜日=SUN)

⑤行を+1する

上記を月末まで繰り返す

1ヶ月分が終わると、曜日は翌月の1日の曜日になっている。

カレンダー3次元配列出力のヒント

- 作成したカレンダーの3次元配列の出力
 - 単純にfor文の3重ループで配列のすべての要素を出力

```
月のループ{                                     // JAN ~ DEC
    《ここで、年月、曜日を表示》
    行のループ{                                   // 0 ~ 6行
        曜日のループ{                             // SUN~SAT
            ここで、日にちが入っていれば日にちを出力、
            日にちが 0 の場合は日にちの代わりに空白を出力
        }
        《1週間分の出力が終わったので、改行する》
    }
}
```

Step3 祝日を入れる

Step3: 祝日を入れる

■Step3で学ぶこと

ー祝日のデータテーブルを用意する

- ◆ 色々な手法がありますが、今回は次ページに示すものを使います
- ◆ 計算式によって、春分と秋分の日を求める

ー求めた、祝日をカレンダー配列に入れる(マークする)

- ◆ 祝日のマークを、ANDとORのビット操作で行う
 - (1) 最初は、固定日の祝日を入れてみる
 - ・ その日が日曜日なら次の日(月曜日)を振替にする
 - (2) ハッピーマンデーの処理
 - (3) 例外処理(祝日と祝日に挟まれた日は休日)

ー祝日を含めたカレンダー印字

- ◆ 祝日は'*'を、振替祝日には '+' を、日付の前に付ける
- ◆ 余裕があったら、誕生日マーク(例えば、"&"など)表示する

祝日のデータテーブル

- 祝日テーブルは、グローバル変数として、main関数の前で定義してもよい。

```
#define MAX_HOLI_TBL 4

//*****
//
// 祝日のデータテーブル
//
char holidays[N_MONTH][MAX_HOLI_TBL] = {
    { 1, -2, 0, 0},    // 1月、元旦、成人の日
    {11, 23, 0, 0},    // 2月、建国記念日、令和天皇誕生日
    {21, 0, 0, 0},    // 3月、春分の日(計算で算出)
    {29, 0, 0, 0},    // 4月、昭和の日
    { 3, 4, 5, 0},    // 5月、憲法記念日、みどりの日、子供の日
    { 0, 0, 0, 0},    // 6月、なし
    {-3, 0, 0, 0},    // 7月、海の日
    {11, 0, 0, 0},    // 8月、山の日
    {-3, 23, 0, 0},    // 9月、敬老の日、秋分の日(計算で算出)
    {-2, 0, 0, 0},    // 10月、体育の日
    { 3, 23, 0, 0},    // 11月、文化の日、勤労感謝の日
    { 0, 0, 0, 0}     // 12月、なし(23日の平成天皇誕生日はなし)
};
```


春分、秋分の日の計算式(下記をコピーして使用)

```
#define SPRING_EQ 0
#define FALL_EQ 1

// 以下を関数にする。引数は、int year
int springEQ;           // 求める春分の日
int fallEQ;             // 求める秋分の日

if (year <= 1899) {
    springEQ = (int) (19.8277 + 0.242194 * (year - 1980.0) - ((year - 1983) / 4));
    fallEQ = (int) (22.2588 + 0.242194 * (year - 1980.0) - ((year - 1983) / 4));
} else if (year >= 1900 && year <= 1979) {
    springEQ = (int) (20.8357 + 0.242194 * (year - 1980.0) - ((year - 1983) / 4));
    fallEQ = (int) (23.2588 + 0.242194 * (year - 1980.0) - ((year - 1983) / 4));
} else if (year >= 1980 && year <= 2099) {
    springEQ = (int) (20.8431 + 0.242194 * (year - 1980.0) - ((year - 1980) / 4));
    fallEQ = (int) (23.2488 + 0.242194 * (year - 1980.0) - ((year - 1980) / 4));
} else if (year >= 2100) {
    springEQ = (int) (21.851 + 0.242194 * (year - 1980.0) - ((year - 1980) / 4));
    fallEQ = (int) (24.2488 + 0.242194 * (year - 1980.0) - ((year - 1980) / 4));
}

holidays[MAR][SPRING_EQ] = springEQ;           // 祝日テーブルに春分の日を入れる
holidays[SEP][FALL_EQ] = fallEQ;               // 祝日テーブルに秋分の日を入れる
```

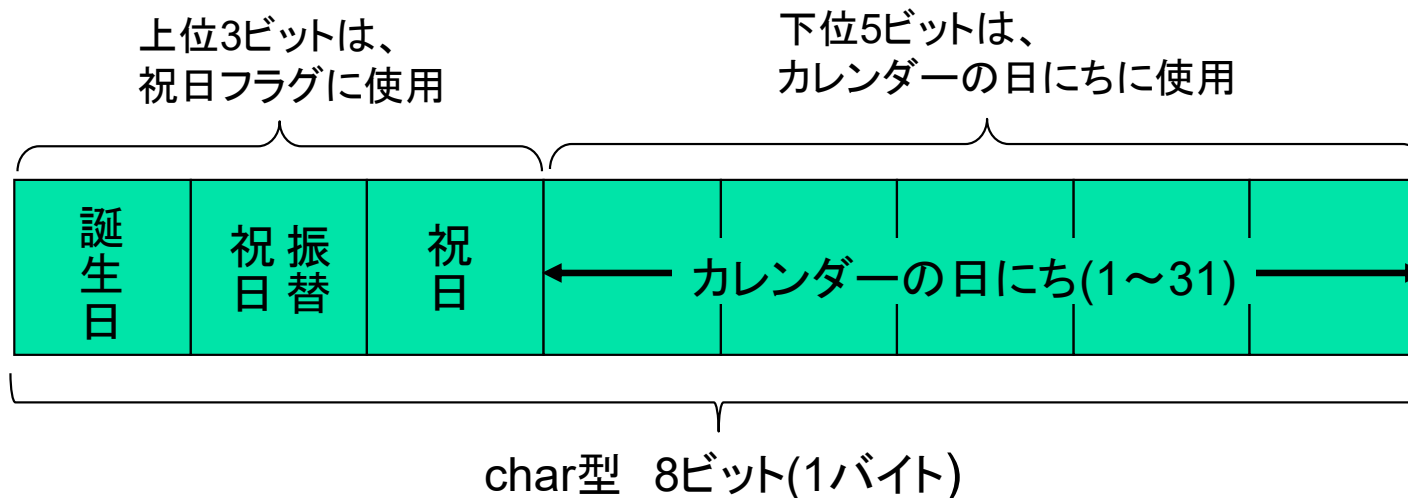
祝日処理のヒント(ビット操作でマークを入れる)

■ カレンダーの日にち(1~31)に必要なビット数は? → 5ビット

– char 型は何ビット? 8ビット(1バイト) → カレンダーのテーブルはchar型で十分

◆ 配列のサイズは、 $12 \times 6 \times 7 = 504$ バイト。int型はバイト? → int型すると2016バイト使う

– そのため、上位3ビットは使用しないため、祝日マークに利用する。



下記のようなビットを定義をして、3次元配列のカレンダーテーブルにマークを入れる

```
#define BIRTH_MARK    0x80    // 誕生日マーク
#define TRANS_MARK    0x40    // 振替祝日マーク
#define HOLI_MARK     0x20    // 祝日マーク
```

マークを入れるときはビット操作のORを、マークがあるかの確認はANDを利用する

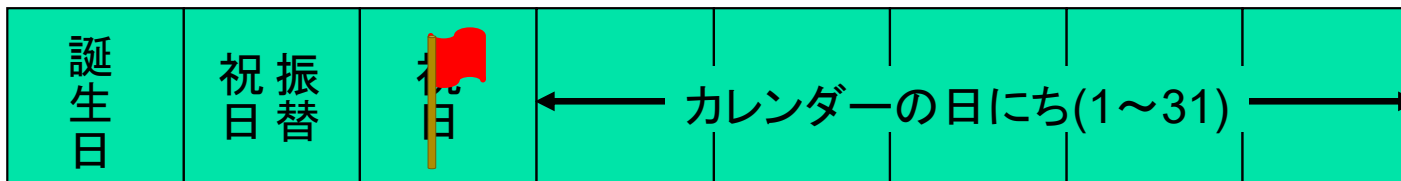
ビット操作

■ ビット操作とは、

- 指定した任意のビット(複数ビットも可)を 0 にしたり 1 にしたりすること → ANDまたはOR操作
- 指定した任意のビット(複数ビットも可)が 0 であるか 1 であるかを調べること → AND操作

■ 祝日マークを入れるとは？

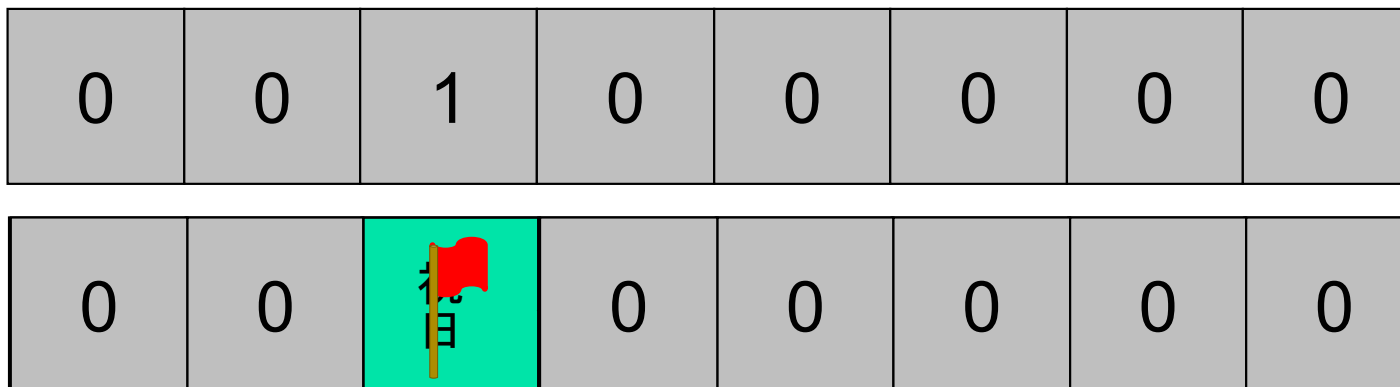
- 祝日マークのフラグをセット(1にする) → OR 操作でビットを1にする
- 変数 `|= HORI_MARK`; ← ビットをセットする



■ 祝日マークがあるかどうかは？

祝日と定義したビットだけを抽出して、0か1かの判定をする

`if (変数 & HORI_MARK)` ← ビットがセットされているかを調べる



祝日マークを入れる実際のプログラム例

```
for (month=JAN ; month<=DEC; month++){
```

```
    <月初め>
```

```
    日にちを1に、行を0にする ← 今までの処理
```

```
    祝日カウント = 0;           // 祝日のカウントをクリア
```

```
    月曜日のカウント = 0;       // 月曜日のカウントをクリア
```

```
    while (日にち<=月末){
```

```
        if (曜日 == MON)        // 今日が月曜日であれば、
            月曜日のカウントを+1する
```

```
        3次元配列[month][行][曜日] |= 日にち;
```

祝日が日曜日だった場合、振り替えのマークが先行して入り、日にちはまだ入っていない。そのため、日にちも OR で入れる必要がある

```
        if (祝日データ[month][祝日カウント] == 日にち){
            3次元配列[month][行][曜日] |= HOLI_MARK;
```

```
            if (曜日==SUN){
```

```
                3次元配列[month][行][曜日+1] |= TRANS_MARK;
```

```
            }
```

```
            // ここに、今日が水曜日以上で、祝日と祝日に挟まれた平日は休日にする処理を入れる
```

```
            祝日カウント++;
```

```
            // 祝日カウントを進める
```

```
        }else{
```

```
            // 今日が月曜日であれば、ハッピーマンデーの処理
```

```
            祝日カウント++;
```

```
        }
```

```
    }
```

Step4 カレンダーを複数列で表示する

Step4: 作成したカレンダーの表示を変える

■ Step4で学ぶこと

- Step3までで、縦12ヶ月で表示していたカレンダーを、横2ヶ月、縦6ヶ月、あるいは横3ヶ月、縦4ヶ月のカレンダーを表示する
- 列の指定は、1列、2列、3列の3種類とする。
 - ◆ 横に何列表示するかは、最初に入力して指定する
- 配列のアクセスは、次元数に応じて多重ループになる。
 - ◆ 1次元配列は、1重ループ (1次元配列は九九の表で経験済み)
 - ◆ 2次元配列は、2重ループ (2次元配列はライフゲームで経験済み)
 - ◆ 3次元配列は、3重ループ (3次元配列はカレンダーのStep3で経験済み)
- カレンダーを横に複数列表示するためには、4重ループが必要



ヒント

◆ 行のループと曜日のループの間に、列のループが入る

◆ 行のループを -2 から始める方法も考えてみよう

➤ 年月、曜日の表示が楽になる

カレンダーの複数列出力のヒント

■ Step3で表示していた内容を変更する

- 行と曜日のループの間に、列のループを入れる
- 年月と曜日の表示を列ループの中に移動し、改行はしない

```
for 月のループ{                                // JAN ~ DEC (月の増加は+1でなく、列数を増加する)
  for 行のループ{                                // -2 ~ 6行 (-2から始める)
    for 列のループ{ ← このループを追加(0～指定された列数)
      if (行が-2なら) 年月の表示
      else if(行が-1なら) 曜日の表示           } 年月と曜日の表示は、
      else {                                     列ループの中です
        for 曜日のループ{ // SUN~SAT
          日にちが入っていれば日にちを出力、
          日にちが 0 の場合は日にちの代わりに空白を出力
        }
        ここに月と月の間隔の空白を出力
      }
    }
  }
  列のループが終わったら、改行する
}
```

月の変数に、列ループの変数を足さないと、同じ月のデータが出力される。

表示例(横2ヶ月、縦6ヶ月)

2015年 1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	27	28	29	30	31

2015年 2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	27	28

2015年 3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	27	28
29	30	31				

2015年 4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	27	28	*29	30		

2015年 5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	27	28	29	30
31						

2015年 6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	27
28	29	30				

2015年 7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	27	28	29	30	31	

2015年 8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	27	28	29
30	31					

2015年 9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
27	28	29	30			

2015年10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	27	28	29	30	31

2015年11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	27	28
29	30					

2015年12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
27	28	29	30	31		

表示例(横3ヶ月、縦4ヶ月)

2015年 1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	27	28	29	30	31

2015年 2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	27	28

2015年 3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	27	28
29	30	31				

2015年 4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	27	28	*29	30		

2015年 5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	27	28	29	30
31						

2015年 6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	27
28	29	30				

2015年 7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	27	28	29	30	31	

2015年 8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	27	28	29
30	31					

2015年 9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
27	28	29	30			

2015年10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	27	28	29	30	31

2015年11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	27	28
29	30					

2015年12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
27	28	29	30	31		

Step4.5 おまけ-月と月の間隔を自動化するテクニック

- 横に表示するとき月と月の空白を自動計算する
 - 月と月の間は、空白を入れて間隔の調整したと推測する

2015年 1月							2015年 2月							2015年 3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	1	2	3	4	5	6	7	1	2	3	4	5	6	7
4	5	6	7	8	9	10	8	9	10	*11	12	13	14	8	9	10	11	12	13	14
11	*12	13	14	15	16	17	15	16	17	18	19	20	21	15	16	17	18	19	20	*21
18	19	20	21	22	23	24	22	*23	24	25	26	27	28	22	23	24	25	26	27	28
25	26	27	28	29	30	31								29	30	31				

足りない空白

月と月の間隔

– その月と月の空白を計算で求める方法

- ◆ カレンダーの1日の表示桁を4桁にした場合、1週間に必要な桁数は、
 - 4文字 × 7日 = 28文字
- ◆ 年月日の表示は、例えば、”%4d年2d%月”とした場合、年月で使用する桁数は、
 - 漢字は2文字で数えるため、4 + 2(年) + 2 + 2(月) = 10文字
 - すなわち、年月表示の1週間の文字数で足りない空白は
 - 28文字 – 10文字 = 18文字

3つのprintf()

- データの出力に最も多く使用するprintf()は3種類ある

`printf ("hello World\n");` → 普通のprintfで画面に出力
`fprintf (fp, "hello World\n");` → fpで指定したファイルに出力
`sprintf (buf, "hello World\n");` → 出力をbufに格納

- 年月日を表示して、月の終わりに満たない空白を求める

- 出力結果をbufに格納

```
char buf[64];  
int len;  
sprintf(buf, "%4d年%d月", year, month);
```

yearが2021、monthが4であれば、
bufには、"2019年4月"が入る

- 格納した出力の文字数を求める (strlen関数を使う)

```
len = strlen(buf);
```

- 月の終わりまでに足りない空白の数は、 $4*N_WEEK - len$ で求まる

- 指定した数だけスペースを出力する関数を作成する

```
printf("%s", buf); ← bufの内容を文字列として出力する  
put_char(' ', 4*N_WEEK - len ); ← 足りない空白を出力
```

put_char()関数は、九九の表で横線を表示する時に作成した関数を参考して作成する。

put_char() は、新たに作成する関数です。標準関数のputchr()ではありません。

付録:ANSIエスケープシーケンス

ライフゲームでは、エスケープシーケンスで画面のクリアとカーソルの移動を行いました。
カレンダーでは、エスケープシーケンスで祝日のカラー表示を試みます。

画面の制御 - エスケープシーケンスとは?

- Unixなどの世界では、画面の制御にエスケープシーケンスというものを使用している。
- Windows10から、コマンドプロンプトでエスケープシーケンスがサポートされた。
- エスケープシーケンスとは、0x1bに続く文字で色々と定められている。
- 下記は、その一例です。

Teratermなどでテストをする場合は、ESCキーを押した後に、「[」「2」「J」と押します。以下同じです。

```
printf("\033[2J"); //画面クリア
printf("\033[0K"); //カーソル位置からその行の右端までをクリア
printf("\033[1K"); //カーソル位置からその行の左端までをクリア
printf("\033[2K"); //カーソル位置の行をクリア
```

```
printf("\033[%d;%dH", 10, 20); //カーソル位置を、高さ10行目、横20文字目に移動
printf("\033[%dC", 10); //カーソルを10文字だけ右に移動
printf("\033[%dD", 10); //カーソルを10行文字だけ左に移動
printf("\033[%dB", 10); //カーソルを10行だけ下に移動
printf("\033[%dA", 10); //カーソルを10行だけ上に移動
```

実際は、各色を定義して使う

```
printf("\033[30m"); //黒の文字に変更
printf("\033[31m"); //赤の文字に変更
printf("\033[32m"); //緑の文字に変更
printf("\033[33m"); //黄色の文字に変更
printf("\033[34m"); //青の文字に変更
printf("\033[35m"); //マゼンダの文字に変更
printf("\033[36m"); //シアンの文字に変更
printf("\033[37m"); //白の文字に変更
printf("\033[39m"); //文字の色を通常の色に戻す
Printf("\033[0m"); //リセット(設定を未設定の状態に戻す)
```

#define COLOR_BLACK	"\033[30m"	// 黒色
#define COLOR_RED	"\033[31m"	// 赤色
#define COLOR_GREEN	"\033[32m"	// 緑色
#define COLOR_YELLOW	"\033[33m"	// 黄色
#define COLOR_BLUE	"\033[34m"	// 青色
#define COLOR_MAGENTA	"\033[35m"	// 紫色
#define COLOR_CYAN	"\033[36m"	// 水色
#define COLOR_WHITE	"\033[37m"	// 白色
#define COLOR_NORM	"\033[39m"	// 通常に戻す

¥033 は、C言語における8進数の定数表現
¥x1b として、16進定数としても良い。
例: "\033[30m" は: "\x1b[30m" と同じ

エスケープシーケンスを使ったカラー表示

■ コマンドプロンプトで、エスケープシーケンスを実行した例

```
R:¥>type ansi_test.txt
```

		40m	41m	42m	43m	44m	45m	46m	47m
m	Color	Color	Color	Color	Color	Color	Color	Color	Color
1m	Color	Color	Color	Color	Color	Color	Color	Color	Color
30m									
1;30m	Color	Color	Color	Color	Color	Color	Color	Color	Color
31m	Color	Color	Color	Color	Color	Color	Color	Color	Color
1;31m	Color	Color	Color	Color	Color	Color	Color	Color	Color
32m	Color	Color	Color	Color	Color	Color	Color	Color	Color
1;32m	Color	Color	Color	Color	Color	Color	Color	Color	Color
33m	Color	Color	Color	Color	Color	Color	Color	Color	Color
1;33m	Color	Color	Color	Color	Color	Color	Color	Color	Color
34m	Color	Color	Color	Color	Color	Color	Color	Color	Color
1;34m	Color	Color	Color	Color	Color	Color	Color	Color	Color
35m	Color	Color	Color	Color	Color	Color	Color	Color	Color
1;35m	Color	Color	Color	Color	Color	Color	Color	Color	Color
36m	Color	Color	Color	Color	Color	Color	Color	Color	Color
1;36m	Color	Color	Color	Color	Color	Color	Color	Color	Color
37m	Color	Color	Color	Color	Color	Color	Color	Color	Color
1;37m	Color	Color	Color	Color	Color	Color	Color	Color	Color

Unsupported ANSI escape codes:

- ansi code 2 (faint)
- ansi code 3 (italic)
- ansi code 4 (underscore)
- ansi code 5 (slow blink)
- ansi code 6 (fast blink)
- ansi code 7 (reverse/negative)
- ansi code 8 (conceal/hidden)
- ansi code 9 (strike-through)

エスケープシーケンスのテストプログラム

テストプログラム

```
#include<stdio.h>

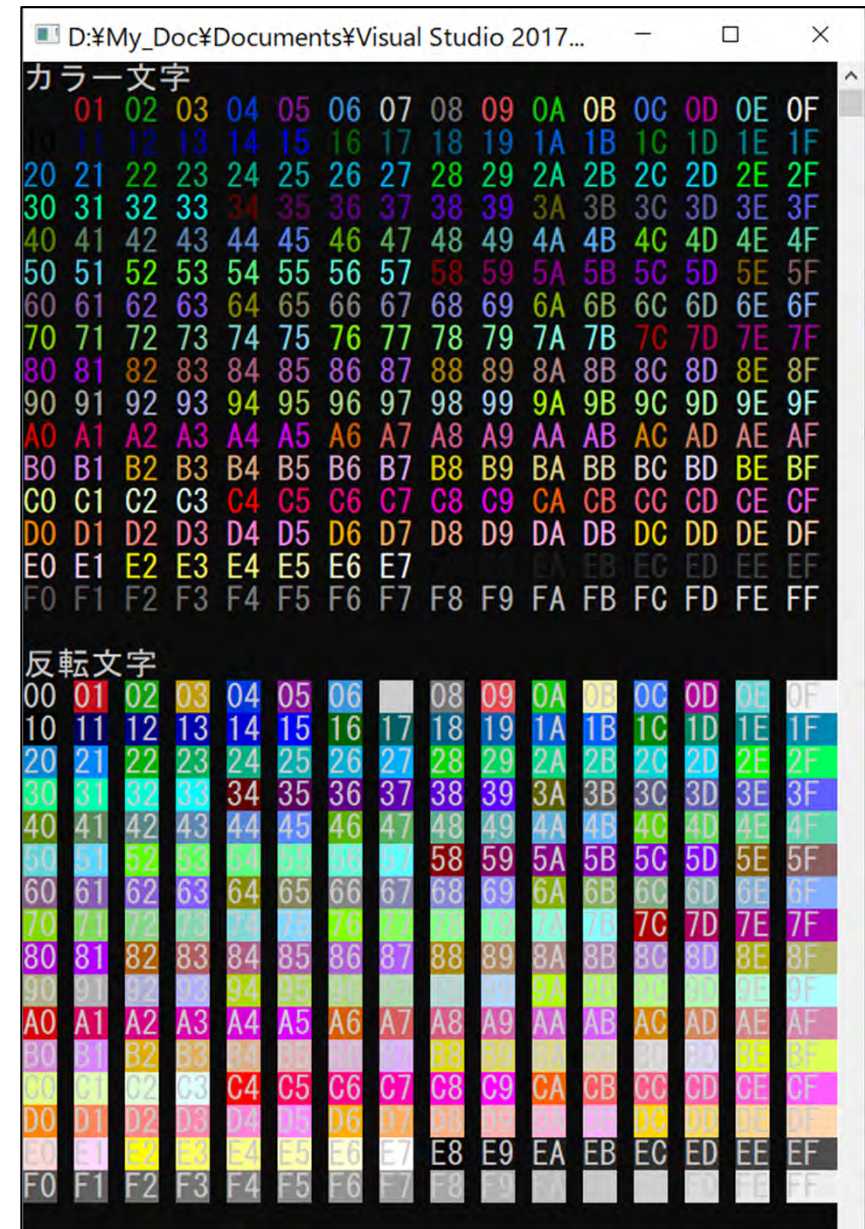
void main(void) {

    int row, col, num;

    printf("カラー文字¥n");
    for (row = 0; row < 16; row++) {
        for (col = 0; col < 16; col++) {
            num = row * 16 + col;
            printf("¥033[38;5;%dm¥033[0m ", num, num);
        }
        printf("¥n");
    }

    printf("¥n反転文字¥n");
    for (row = 0; row < 16; row++) {
        for (col = 0; col < 16; col++) {
            num = row * 16 + col;
            printf("¥033[48;5;%dm¥033[0m ", num, num);
        }
        printf("¥n");
    }
}
```

実行結果



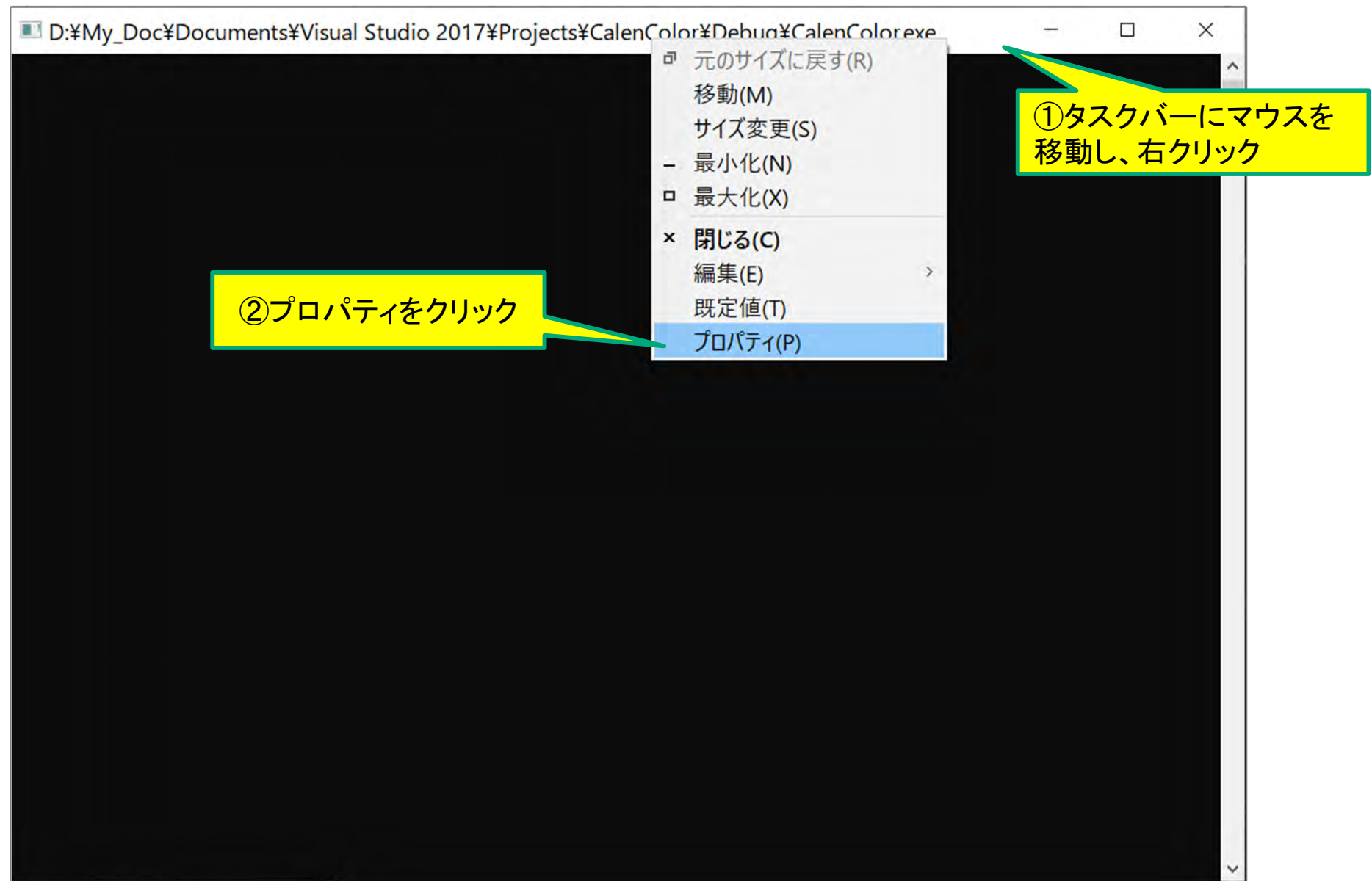
文字による祝日表示

2021年 1月							2021年 2月							2021年 3月						
日	月	火	水	木	金	土	日	月	火	水	木	金	土	日	月	火	水	木	金	土
					* 1	2		1	2	3	4	5	6		1	2	3	4	5	6
3	4	5	6	7	8	9	7	8	9	10	* 11	12	13	7	8	9	10	11	12	13
10	* 11	& 12	13	14	15	16	14	15	16	17	18	19	20	14	15	16	17	18	19	* 20
17	18	19	20	21	22	23	21	22	* 23	24	25	26	27	21	22	23	24	25	26	27
24	25	26	27	28	29	30	28							28	29	30	31			
31																				
2021年 4月							2021年 5月							2021年 6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							1			1	2	3	4	5
4	5	6	7	8	9	10	2	* 3	* 4	* 5	6	7	8	6	7	8	9	10	11	12
11	12	13	14	15	16	17	9	10	11	12	13	14	15	13	14	15	16	17	18	19
18	19	20	21	22	23	24	16	17	18	19	20	21	22	20	21	22	23	24	25	26
25	26	27	28	* 29	30		23	24	25	26	27	28	29	27	28	29	30			
							30	31												
2021年 7月							2021年 8月							2021年 9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	7				1	2	3	4
4	5	6	7	8	9	10	1	2	3	* 11	12	13	14	5	6	7	8	9	10	11
11	12	13	14	15	16	17	8	9	10	11	12	13	14	12	13	14	15	16	17	18
18	* 19	20	21	22	23	24	15	16	17	18	19	20	21	19	* 20	21	22	* 23	24	25
25	26	27	28	29	30	31	22	23	24	25	26	27	28	26	27	28	29	30		
							29	30	31											
2021年10月							2021年11月							2021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	6				1	2	3	4
3	4	5	6	7	8	9	7	8	9	10	11	12	13	5	6	7	8	9	10	11
10	* 11	12	13	14	15	16	14	15	16	17	18	19	20	12	13	14	15	16	17	18
17	18	19	20	21	22	23	21	22	* 23	24	25	26	27	19	20	21	22	23	24	25
24	25	26	27	28	29	30	28	29	30					26	27	28	29	30	31	

カラーによる祝日表示

2021年 1月							2021年 2月							2021年 3月						
日	月	火	水	木	金	土	日	月	火	水	木	金	土	日	月	火	水	木	金	土
					1	2		1	2	3	4	5	6		1	2	3	4	5	6
3	4	5	6	7	8	9	7	8	9	10	11	12	13	7	8	9	10	11	12	13
10	11	12	13	14	15	16	14	15	16	17	18	19	20	14	15	16	17	18	19	20
17	18	19	20	21	22	23	21	22	23	24	25	26	27	21	22	23	24	25	26	27
24	25	26	27	28	29	30	28							28	29	30	31			
31																				
2021年 4月							2021年 5月							2021年 6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							1			1	2	3	4	5
4	5	6	7	8	9	10	2	3	4	5	6	7	8	6	7	8	9	10	11	12
11	12	13	14	15	16	17	9	10	11	12	13	14	15	13	14	15	16	17	18	19
18	19	20	21	22	23	24	16	17	18	19	20	21	22	20	21	22	23	24	25	26
25	26	27	28	29	30		23	24	25	26	27	28	29	27	28	29	30			
							30	31												
2021年 7月							2021年 8月							2021年 9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	7				1	2	3	4
4	5	6	7	8	9	10	1	2	3	4	5	6	7	5	6	7	8	9	10	11
11	12	13	14	15	16	17	8	9	10	11	12	13	14	12	13	14	15	16	17	18
18	19	20	21	22	23	24	15	16	17	18	19	20	21	19	20	21	22	23	24	25
25	26	27	28	29	30	31	22	23	24	25	26	27	28	26	27	28	29	30		
							29	30	31											
2021年10月							2021年11月							2021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	6				1	2	3	4
3	4	5	6	7	8	9	7	8	9	10	11	12	13	5	6	7	8	9	10	11
10	11	12	13	14	15	16	14	15	16	17	18	19	20	12	13	14	15	16	17	18
17	18	19	20	21	22	23	21	22	23	24	25	26	27	19	20	21	22	23	24	25
24	25	26	27	28	29	30	28	29	30					26	27	28	29	30	31	

カラーが表示されないときの確認 その1



カラーが表示されないときの確認 その2

オプションタグ

ここにチェックがないことを確認

