```c
#pragma warning(disable:4996)   // scanf()のエラー回避用

/*****************************************************************************************
*
* 万年カレンダー(サンプル)
*
* 処理内容
*   指定された西暦から、指定された年数のカレンダーを作成する
*
* 入力  : なし
*
* 出力  : なし
*
* 戻り値: なし
*
* 変更履歴
* 変更日付    Rev.   変更者       変更内容
* ----------  ----  ----------  ------------------------------------------------
* 2017/05/30  1.0   M.Hoshi     新規作成(万年カレンダーのサンプル)
* 2019/04/22  1.1   M.hoshi     令和天皇の即位により、天皇誕生日を2/23日に変更
* 2020/04/16  1.2   M.Hoshi     カラー表示を追加
*
*****************************************************************************************/

#include <stdio.h>
#include <string.h> // memset()で使用

////////////////////////////////////////////////////////////////
//
// 定数宣言
//

#define MAX_ROWS    7   // 1ケ月分のカレンダー配列の行数
#define NUM_OF_HORI 4   // 祝日テーブルの最大日数
#define SPRING_EQ   0   // 春分の日の祝日データの配列位置(最初の祝日)
#define FALL_EQ     1   // 秋分の日の祝日データの配列位置(2回目の祝日)

#define HOLIDAY_MARK  0x20  // 祝日マークのビットフラグ
#define TRANS_MARK    0x40  // 振替休日マークのビットフラグ
#define BIRTHDAY_MARK 0x80  // 誕生日マーク(今回は未使用)
#define DAY_VALUE     0x1f  // 31日の最大数のマスク

#define HOLI_CHAR  "*"      // 祝日マーク
#define TRANS_CHAR "+"      // 振り替え祝日マーク
#define BIRTH_CHAR "&"      // 誕生日マーク

#define DISP_MONTH -2       // xxxx年xx月の表示 (行番号をこの数から開始)
#define DISP_WEEK  -1       // 曜日の表示 (行番号をこの数のとき曜日の表示)
#define DISP_SPACE  3       // 横表示時の月と月のスペース数

////////////////////////////////////////////////////////////////
```

```
52  //
53  // エスケープシーケンスによるカラー指定
54  //
55
56  #define COLOR_BLACK     "¥033[30m"  // 黒色
57  #define COLOR_RED       "¥033[31m"  // 赤色
58  #define COLOR_GREEN     "¥033[32m"  // 緑色
59  #define COLOR_YELLO     "¥033[33m"  // 黄色
60  #define COLOR_BLUE      "¥033[34m"  // 青色
61  #define COLOR_MAGENTA   "¥033[35m"  // 紫色
62  #define COLOR_CYAN      "¥033[36m"  // 水色
63  #define COLOR_WHITE     "¥033[37m"  // 白色
64  #define COLOR_NORM      "¥033[39m"  // 通常に戻す
65
66  enum M_LIST { JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC, N_MONTH };
67  enum W_LIST { SUN, MON, TUE, WED, THU, FRI, SAT, N_WEEK };
68
69
70  ////////////////////////////////////////////////////////////////
71  //
72  // プロトタイプ宣言
73  //
74
75  int make_calen(char calen_tbl[N_MONTH][MAX_ROWS][N_WEEK], int daywk);
76  void print_calen(char calen_tbl[N_MONTH][MAX_ROWS][N_WEEK], int year, int num_of_pair, char color_mode);
77  int datainput(int* s_year, int* num_of_pair, char* color_mode);
78  int calc_calen(int s_year);
79  int  chk_leapyear(int year);
80  void calc_EQday(int yy);
81  void put_char(char ch, int num_of_cnt);
82
83
84  ////////////////////////////////////////////////////////////////
85  //
86  // グローバル変数
87  //
88
89  char mdays[N_MONTH] = { 31,28,31,30,31,30,31,31,30,31,30,31 };          // 月の日数(2月は閏年判定で変更)
90  char* wkday[N_WEEK] = { " 日"," 月"," 火"," 水"," 木"," 金"," 土" };    // 曜日データ
91
92  //*****************************************************
93  //
94  // 祝日、曜日、月のデータテーブル
95  //
96
97  char holidays[N_MONTH][NUM_OF_HORI] = {
98          {1,-2,0,0},     //  1月、元旦、成人の日
99          {11,23,0,0},    //  2月、建国記念日、令和天皇誕生日
100         {21,0,0,0},     //  3月、春分の日
101         {29,0,0,0},     //  4月、昭和の日
102         {3,4,5,0},      //  5月、憲法記念日、みどりの日、子供の日
```

```
103         { 0,0,0,0},     //  6月、なし
104         {-3,0,0,0},     //  7月、海の日
105         {11,0,0,0},     //  8月、山の日
106         {-3,23,0,0},    //  9月、敬老の日、秋分の日
107         {-2,0,0,0},     // 10月、体育の日
108         {3,23,0,0},     // 11月、文化の日、勤労感謝の日
109         { 0,0,0,0}      // 12月、なし(天皇誕生日は2月に移動)
110 };
111
112
113 //*******************************************************
114 //
115 //  メインルーチン
116 //
117 int main(void) {
118
119     char calen_tbl[N_MONTH][MAX_ROWS][N_WEEK];  // 12ケ月分のカレンダーテーブル
120     int s_year;         // 作成する西暦
121     int num_of_pair;    // 横に表示する列数
122     int daywk;          // 元旦の曜日
123     char color_mode;    // カラー表示モード 'y'ならカラー表示
124
125     //*******************************************************
126     //
127     //  データ入力
128     //
129     printf(COLOR_NORM); //文字の色を通常色にする
130     if ((datainput(&s_year, &num_of_pair, &color_mode)) != 0) {
131         printf("入力にエラーがありました");
132         return 0;
133     }
134
135     //*******************************************************
136     //
137     //  指定した年の1月1日の曜日、春分、秋分、うるう年の判定
138     //
139     daywk = calc_calen(s_year); // 入力した西暦から開始年の1月1日の曜日を求める
140
141     //*******************************************************
142     //
143     //  1年分のカレンダーを作成(戻り値は翌年の1月1日の曜日)
144     //
145     memset(calen_tbl, 0, sizeof(calen_tbl));    // カレンダーのテーブルを0で初期化
146     daywk = make_calen(calen_tbl, daywk);       // 1年分のカレンダーの作成
147
148     //*******************************************************
149     //
150     //  1年分のカレンダーを印字
151     //
152     print_calen(calen_tbl, s_year, num_of_pair, color_mode);            // 1年分のカレンダーの表示
153 }
```

```
154
155
156 //*********************************************************************************************
157 //
158 // make_calen
159 //
160 // 処理内容
161 //  祝日を含めた、1年分のカレンダーを作成する
162 //
163 // 入力 : mday      = 各月の最大日数のテーブル
164 //      : holidays  = 祝日テーブル(12 x 4 の配列)
165 //
166 // 引数 : calen_tbl = カレンダの3次元テーブル
167 //      : daywk     = 当該年の元旦の曜日
168 //
169 // 出力 : calen_tbl = 3次元のカレンダーテーブルデータを入れる
170 //
171 // 戻り値: 次の年の元旦の曜日
172 //
173
174 int make_calen(char calen_tbl[N_MONTH][MAX_ROWS][N_WEEK], int daywk) {
175
176     int month;
177     int day;        // 開始の日にち
178     int wkcnt;      // カレンダー日にちを入れる最初の行
179     int mondayCnt;  // 月曜日のカウント
180     int holicnt;    // 祝日のカウント
181
182     for (month = JAN; month <= DEC; month++) {  // 1年のループ
183         day = 1;        // 日付の初期化
184         wkcnt = 0;      // 行番号
185         mondayCnt = 0;  // 月曜日のカウント
186         holicnt = 0;    // 祝日番号の初期化
187
188         while (day <= mdays[month]) {   // 1ケ月のループ
189             if (daywk == MON) {         // 曜日が月曜日であれば(Happy Mondayのために月曜日をカウント)
190                 mondayCnt++;            // 月曜日のカウントアップ
191             }
192             calen_tbl[month][wkcnt][daywk] |= day;  // 日付を入れる
193
194 //**************************************
195 //
196 // 祝日の処理
197 //
198             if (holidays[month][holicnt] != 0) {            // 祝日データがあれば、祝日の処理をする
199
200                 if (holidays[month][holicnt] == day) {      // 固定の祝日であれば、
201                     calen_tbl[month][wkcnt][daywk] |= HOLIDAY_MARK;         // 祝日マークを入れる
202
203                     if ((daywk == SUN) || (calen_tbl[month][wkcnt][daywk] & TRANS_MARK)) {  // 日曜日あるいはすでに振替になっていれば、
204                         calen_tbl[month][wkcnt][daywk + 1] |= TRANS_MARK;   //翌日を振替にする;
```

```
                    }
                    else if (daywk >= WED) {    // 曜日が水曜より後であれば、祝日に挟まれた平日を祝日するチェック
                        if ((calen_tbl[month][wkcnt][daywk - 1] & HOLIDAY_MARK) == 0 && // 前日が平日で、
                            (calen_tbl[month][wkcnt][daywk - 2] & HOLIDAY_MARK)) {      // 前々日が祝日なら、
                            calen_tbl[month][wkcnt][daywk - 1] |= TRANS_MARK;           // 前日を休日にする
                        }
                    }
                    holicnt++;  // 祝日の処理をしたので次の祝日へ
                }
                else if (daywk == MON && (holidays[month][holicnt] * -1) == mondayCnt) {    //ハッピーマンデーのチェック
                    calen_tbl[month][wkcnt][daywk] |= HOLIDAY_MARK; // ハッピーマンデーの祝日マークを入れる
                    holicnt++;  // Happy Mondayの祝日の処理をしたので次の祝日へ
                }
            }
            daywk++;            // 次の曜日にする
            day++;              // 次の日にちにする

            if (daywk > SAT) {  // 曜日が土曜日を超えたら
                daywk = SUN;    // 曜日を日曜日にする
                wkcnt++;        // 行を次の週に
            }
        }   // 1ケ月のループ
    }   // 1年のループ

    return daywk;   // 翌年の1月1日の曜日を戻す
}


//*****************************************************************************************
//
// print_calen
//
// 処理内容
// 祝日を含めた、1年分のカレンダーを表示する
//
// 入力 : なし
//
// 引数 : calen_tbl   = カレンダの3次元テーブル
//      : s_year      = 作成年
//      : num_of_pair = 横に表示する列数
//      : color_mode  = カラー表示 'y'ならカラー表示
//
// 出力 : なし
//
// 戻り値: なし
//

void print_calen(char calen_tbl[N_MONTH][MAX_ROWS][N_WEEK], int s_year, int num_of_pair, char color_mode) {

    int wkcnt;          // 処理する行番号
    int daywk;          // 処理する曜日
```

```
256     int paircnt;        // 横に表示する月数
257     int month;          // 処理する月番号
258     char tmpstr[16];    // 年月表示の一時文字列
259
260     // 1年間のループ
261     for (month = JAN; month < N_MONTH; month += num_of_pair) {  // 月は指定された桁数分だけ増加
262
263         for (wkcnt = DISP_MONTH; wkcnt < MAX_ROWS; wkcnt++) {   // 行のループ
264
265             for (paircnt = 0; paircnt < num_of_pair; paircnt++) {   // 横表示のループ
266
267                 if (wkcnt == DISP_MONTH) {
268                     sprintf(tmpstr, " %d年%d月", s_year, month + paircnt + 1);  // 年月を文字列する
269                     printf("%s", tmpstr);   // 年月を表示
270                     put_char(' ', N_WEEK * 4 - (int)strlen(tmpstr));    // 月の終わりまでのスペースを表示
271                 }
272                 else if (wkcnt == DISP_WEEK) {
273                     for (daywk = SUN; daywk <= SAT; daywk++) {
274                         printf("%s ", wkday[daywk]);    // 曜日の印刷
275                     }
276                 }
277                 else {
278                     for (daywk = SUN; daywk <= SAT; daywk++) {  // カレンダーの日にちを表示
279                         if (color_mode == 'y') {
280                             if (calen_tbl[month + paircnt][wkcnt][daywk] != 0) {
281                                 if ((calen_tbl[month + paircnt][wkcnt][daywk] & HOLIDAY_MARK))
282                                     printf(COLOR_RED);  // 祝日を赤く表示
283                                 else if ((calen_tbl[month + paircnt][wkcnt][daywk] & TRANS_MARK))
284                                     printf(COLOR_RED);  // 振替祝日を赤く表示
285                                 else if (daywk == SUN)
286                                     printf(COLOR_RED);  // 日曜日を赤く表示
287                                 else if (daywk == SAT)
288                                     printf(COLOR_BLUE); // 土曜日を青く表示
289                                 else
290                                     printf(COLOR_NORM); // その他は通常色で表示
291                                 printf(" ");
292
293                                 printf("%2d ", (calen_tbl[month + paircnt][wkcnt][daywk] & DAY_VALUE)); // 日付の印字
294                                 printf(COLOR_NORM);     // 表示を通常色に戻す
295                             }
296                             else
297                                 put_char(' ', 4);   // 4個のスペースを印字
298                         }
299                         else {
300                             if (calen_tbl[month + paircnt][wkcnt][daywk] != 0) {
301                                 if ((calen_tbl[month + paircnt][wkcnt][daywk] & HOLIDAY_MARK))
302                                     printf(HOLI_CHAR);  // 祝日マークの印字
303                                 else if ((calen_tbl[month + paircnt][wkcnt][daywk] & TRANS_MARK))
304                                     printf(TRANS_CHAR); // 振替祝日マークの印字
305                                 else
306                                     printf(" ");    // 祝日でなければ空白の印字
```

```
307
308                                 printf("%2d ", (calen_tbl[month + paircnt][wkcnt][daywk] & DAY_VALUE)); // 日付の印字
309                             }
310                             else
311                                 put_char(' ', 4);   // 4個のスペースを印字
312                         }
313                     }
314                 }
315                 put_char(' ', DISP_SPACE);  // 月と月との間のスペースを入れる
316             }
317             printf("¥n");   // 横一列の表示が終了して改行
318         }
319     }
320 }
321
322
323 //*****************************************************
324 //
325 // カレンダーの計算
326 //
327 // 作成するカレンダーの前年度の12/31日までの累積日数
328 // および指定した月日の曜日を求める
329 // 該当する年のう閏年の判定して、2月末日の変更
330 // 該当する年の祝日データ(春分、秋分の日)の作成
331 //
332
333 int calc_calen(int s_year) {
334
335     int yy = s_year - 1;// 対象年の前年
336     int totaldays;              // 西暦1年1月1日から累積日数
337     int s_daywk;
338
339     totaldays = yy * 365 + yy / 4 - yy / 100 + yy / 400;    // 西暦1年1月1日から12月31日までの累積日数を求める
340     totaldays += 1;                                 // 1を加えて該当年の1月1日までのトータル日数にする
341     s_daywk = totaldays % N_WEEK;                   // 該当年の元旦の曜日を求める
342
343
344 //*******************************************
345 //
346 // 閏年の判定
347 //
348 // 結果が平年(0)なら、2月を28日にする
349 // 結果が閏年(1)なら、2月を29日にする
350 //
351     mdays[FEB] = chk_leapyear(s_year) == 0 ? 28 : 29;   // 2月末の日にちを変更する
352
353
354 //*******************************************
355 //
356 // 春分、秋分の日を算出
357 //
```

```
358     calc_EQday(s_year); // 春分の日、秋分の日のデータを更新
359
360     return s_daywk;     // 元旦の曜日を返す
361 }
362
363
364 //*****************************************************************************************
365 //
366 // datainput
367 //
368 // 処理内容
369 //  作成するカレンダーの西暦と横表示の数を入力
370 //
371 // 入力: なし
372 //
373 // 引数: calen_data = yymmdd_tのデータ構造
374 //
375 // 出力  :
376 //  エラー情報 0=正常  1=西暦の値に誤りあり
377 //
378
379 int datainput(int* s_year, int* num_of_pair, char* color_mode) {
380
381     int errflag = 0;
382     char buf[256];
383
384     printf("万年カレンダーの作成¥n");
385     printf("西暦で年を入れて下さい  : ");
386     (void)scanf("%d", s_year);
387
388     if (*s_year < 0 && *s_year > 9999) {
389         printf("西暦が正しくありません");
390         errflag = 1;
391     }
392     else {
393         printf("一度に表示する月数(1-3) : ");
394         (void)scanf("%d", num_of_pair);
395
396         if (*num_of_pair > 3)       // 横に並べる数字が3よりを大きければ、
397             *num_of_pair = 3;       // 強制的に3列にする
398         else if (*num_of_pair <= 0) // 横に並べる数字が負数であれば、
399             *num_of_pair = 1;       // 強制的に1列にする
400
401         printf("カラー表示をしますか(y/n)?");
402         (void)scanf("%s", buf);
403         *color_mode = buf[0] | 0x20;        // 強制的に小文字にする
404     }
405     printf("¥n");
406
407     return errflag;
408 }
```

```
409
410
411 //*****************************************
412 //
413 // 閏年の判定
414 //
415 // 入力 : なし
416 // 引数 : year = 調べる年の西暦
417 //
418 // 戻り値 : 0 = 平年
419 //          1 = 閏年
420 //
421
422 int chk_leapyear(int year) {
423
424     int leap = 0;   // 判定を平年に初期化
425
426     if (year % 4 == 0)          // 4で割り切れたら閏年
427         leap = 1;
428     else if (year % 100 == 0)   // 100で割り切れたら平年
429         leap = 0;
430     else if (year % 400 == 0)   // 400で割り切れたら閏年
431         leap = 1;
432
433     return leap;    // 結果をリターン
434 }
435
436
437 //*********************************************************
438 //
439 // 春分、秋分の日を算出
440 //
441 // 引数 : yy = 求める西暦
442 //      : holidays = 祝日テーブル(12 x 4 の配列)
443 //
444 // 出力 : 祝日テーブルの3月の1番目に春分の日を入れる
445 //      : 祝日テーブルの9月の2番目に秋分の日を入れる
446 //
447 // 戻り値 : なし
448 //
449
450 void calc_EQday(int yy) {
451
452     int springEQ;               // 春分の日
453     int fallEQ;                 // 秋分の日
454
455     if (yy <= 1899) {
456         springEQ = (int)(19.8277 + 0.242194 * (yy - 1980.0) - ((yy - 1983) / 4));   // 春分の日
457         fallEQ = (int)(22.2588 + 0.242194 * (yy - 1980.0) - ((yy - 1983) / 4));     // 秋分の日
458     }
459     else if (yy >= 1900 && yy <= 1979) {
```

```
460         springEQ = (int)(20.8357 + 0.242194 * (yy - 1980.0) - ((yy - 1983) / 4));   // 春分の日
461         fallEQ = (int)(23.2588 + 0.242194 * (yy - 1980.0) - ((yy - 1983) / 4));     // 秋分の日
462     }
463     else if (yy >= 1980 && yy <= 2099) {
464         springEQ = (int)(20.8431 + 0.242194 * (yy - 1980.0) - ((yy - 1980) / 4));   // 春分の日
465         fallEQ = (int)(23.2488 + 0.242194 * (yy - 1980.0) - ((yy - 1980) / 4));     // 秋分の日
466     }
467     else if (yy >= 2100) {
468         springEQ = (int)(21.851 + 0.242194 * (yy - 1980.0) - ((yy - 1980) / 4));    // 春分の日
469         fallEQ = (int)(24.2488 + 0.242194 * (yy - 1980.0) - ((yy - 1980) / 4));     // 秋分の日
470     }
471
472     holidays[MAR][SPRING_EQ] = springEQ;    // 春分の日を入れる
473     holidays[SEP][FALL_EQ] = fallEQ;        // 秋分の日を入れる
474 }
475
476
477 //*********************************************************
478 //
479 // 指定文字の指定数の出力
480 //
481 // 引数 : ch = 出力する文字
482 //      : num_of_cnt = 出力する数
483 //
484 // 戻り値 : なし
485 //
486
487 void put_char(char ch, int num_of_cnt) {
488
489     for (int cnt = 0; cnt < num_of_cnt; cnt++) {
490         printf("%c", ch);
491     }
492 }
```