情報処理論2a

2章「日数／曜日計算」

（Fairfield公式）

中島康彦

§2. 1 今日の作業ディレクトリを作る

1．% cd ⇒ ホームディレクトリへ移動

2．% mkdir chap14 ⇒ ディレクトリchap14を作成

3．% cd chap14 ⇒ ディレクトリchap14へ移動

4．% netscapeを使ってdata14をchap14へダウンロード

5．% tar xvf data14 ⇒ サンプルデータの複写

```
days1.c
days2.c
ymd1.c
ymd2.c
```

§2. 2 日付 ⇒ 日数／曜日

機能設計(入出力形式)

標準入力　　… 年, 月, 日
標準出力　　… 日数, 曜日(Sun～Sat)
標準エラー出力　… 入力異常の通知

構成設計(内部データ構造)

年を記憶する変数　… int型 y
月を記憶する変数　… int型 m
日を記憶する変数　… int型 d
日数を記憶する変数 … int型 days

§2. 3 日数／曜日を計算するには

西暦元年1月1日(月曜日)からの日数計算

毎年365日増加, 4年毎に1日増加, 100年毎に1日減少, 400年毎に1日増加

西暦y年3月1日まで 365*(y-1)+[y/4]-[y/100]+[y/400]+31+28+1

4月1日まで	+ 31日	[306*(4+1)/10]-122
5月1日まで	+ 61日	[306*(5+1)/10]-122
6月1日まで	+ 92日	[306*(6+1)/10]-122
7月1日まで	+122日	[306*(7+1)/10]-122
8月1日まで	+153日	[306*(8+1)/10]-122
9月1日まで	+184日	[306*(9+1)/10]-122
10月1日まで	+214日	[306*(10+1)/10]-122
11月1日まで	+245日	[306*(11+1)/10]-122
12月1日まで	+275日	[306*(12+1)/10]-122
13月1日まで	+306日	[306*(13+1)/10]-122
14月1日まで	+337日	[306*(14+1)/10]-122

[]はガウス記号

一般化

m月1日まで +[306*(m+1)/10]-122
m月d日まで +[306*(m+1)/10]-122+d-1
西暦y年m月d日まで 365*(y-1)+[y/4]-[y/100]+[y/400]+60+[306*(m+1)/10]-122+d-1

§2. 3 日数／曜日を計算するには(続き)

日数計算式(Fairfield公式)

$$365*y+[y/4]-[y/100]+[y/400]+[306*(m+1)/10]+d-428$$

ただし, $3 \leqq m \leqq 14$

曜日計算式(0:Sun ～ 6:Sat)

$$(365*y+[y/4]-[y/100]+[y/400]+[306*(m+1)/10]+d-428)\%7$$

§2. 4 詳細設計(ソースプログラム days1.c)

```
static char RcsId[] = "$Header$"; /* RCSを使うと日付/版数/作者などに置き換わる */
#include <stdio.h>
char *wday[] = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
main()
{
  int y, m, d;
  int days;

  if (scanf("%d %d %d", &y, &m, &d) != 3) {
    fprintf(stderr, "usage: year month date\n");
    exit(1);
  }

  if (m < 3) { y--; m += 12; }
  days = 365*y+y/4-y/100+y/400+306*(m+1)/10+d-428;

  if (days < 0) {
    fprintf(stderr, "error: days<0\n");
    exit(1);
  }

  printf("%d %s\n", days, wday[days%7]);
  exit(0);
}
```

§2. 5 解説

```
static char RcsId[] = "$Header$";
```

版数表示

- ファイルに局所的な文字配列RcsId[]の定義.
- RCSを使うと，$Header$が版数/日付等に置き換わる.
- 具体的には，ci -l を実行する.

```
static char RcsId[] = "$Header: days1.c,v 1.1 2000/08/19 15:00:18
nakashim Exp nakashim $";
```

- 関数の外側で定義すると外部変数(関数間で共有)
- staticの場合，他ファイルの関数から参照できない

コメント /* ... */

- コメントは，いくら書いても多過ぎることはない.
- 良いソースプログラムとは，仕様書がなくても他人が容易に理解できるもの.
- RCSによる変更履歴も重要な情報源.

§2. 5 解説(続き)

```
#include <stdio.h>
```

システムファイル/usr/include/stdio.hを取り込む

- 標準入出力に関連する識別子が定義してある.

```
stdin    … 標準入力
stdout   … 標準出力
stderr   … 標準エラー出力
```

- システムファイルには多くの種類がある. 使用する機能に応じて，必要なファイルを#includeする.
- 現ディレクトリのユーザファイルを取り込むには，

```
#include "file"
```

§2. 5 解説(続き)

```
char *wday[] = { "Sun", "Mon", "Tue", "Wed", "Thu",
"Fri", "Sat" };
```

文字列を要素とする配列の宣言(初期値を含む)

- 配列名は`wday`
  `wday[]`
- 要素の型は`char*`(文字を指すポインタ即ち文字列)
  `char *wday[]`
- 配列要素番号は0〜6(C言語の場合0から始まる)
  `char *wday[7]`
  `char *wday[] = { 7個の初期値 }`

§2. 5 解説(続き)

```
main(){}
```

関数の宣言

- 引数/返り値の有無に関わらず関数と呼ぶ.
- プログラムは関数`main()`から実行が開始される.
- `main()`が終了するとプログラムが終了する.

- 関数からの返り値は, 通常, 多くても1つである.
- `main()`からの返り値は, シェルに返る.
- `exit(a)`はプログラムを終了し, `a`をシェルに返す.

§2. 5 解説(続き)

```
int y, m, d;
int days;
```

内部変数の宣言

- 整数型(32ビット)
- main()の中でのみ有効
- 初期値指定が無いので，初期値は不定

§2. 5 解説(続き)

```
if (scanf("%d %d %d", &y, &m, &d) != 3) {
  fprintf(stderr,"usage: year month date\n");
  exit(1);
}
```

関数scanf()

- 標準入力から変数への代入
- 入力文字列を空白により区切られた10進数とみなして，変数y，変数m，変数dへ順に代入する.
- 改行は空白とし，さらに代入を待つ.
- 正常に完了した引数の数を返す.
- 3でなければ，入力エラー

関数fprintf()

- 第一引数により指定した先へ出力する.
- stderrは，標準エラー出力を意味する.

§2. 5 解説(続き)

```
if (m < 3) { y--; m += 12; }
```

年および月データの修正

- 3 ≦ m ≦ 14 に対応する.
- mが3未満である場合，yを1減じ，mを12増加する.

§2. 5 解説(続き)

```
days=365*y+y/4-y/100+y/400+306*(m+1)/10+d-428;
```

日数計算式(Fairfield公式)に基づく計算

- 除算結果の小数点以下は切り捨てられる.
- ガウス記号をそのまま実現していることになる.

§2. 5 解説(続き)

```
if (days < 0) {
  fprintf(stderr, "error: days<0\n");
  exit(1);
}
```

入力データの検査

- C言語では，演算例外が発生する整数演算は，0による除算および0による剰余のみ.

- 除算の前に除数を検査する必要がある.
- 同様に，負数に対する剰余演算の結果は負数となるため，演算結果を配列の添字として使用する場合は，予め符号を検査する必要がある.

§2. 5 解説(続き)

```
printf("%d %s\n", days, wday[days%7]);
```

曜日計算式(0:Sun ～ 6:Sat)に基づく計算

- 日数を7で割った剰余
  `days%7`
- `days`は`int`型なので，書式には`%d`を用いる.

- 曜日名を要素とする配列`wday[]`のうち，剰余により指定される要素
  `wday[days%7]`
- `wday[]`の各要素は`char*`型なので，書式には`%s`を用いる.

§2. 5 解説(続き)

```
exit(0);
```

プログラムの終了コード

- 正常終了した場合，返り値を0としておく.
- プログラムを起動したシェルに対して，返り値が通知される.
- シェルスクリプトが，変数$statusを参照してプログラムの正常/異常を検出できる.

```
#!/bin/csh
./days1
if ( $status == 0 ) then
  echo "normal end"
else
  echo "error"
endif
```

§2. 6 日数 ⇒ 日付／曜日

機能設計(入出力形式)

標準入力	… 日数
標準出力	… 年，月，日，曜日(Sun～Sat)
標準エラー出力	… 入力異常の通知

構成設計(内部データ構造)

年を記憶する変数	… int型 y
月を記憶する変数	… int型 m
日を記憶する変数	… int型 d
日数を記憶する変数	… int型 days

§2. 7 日付／曜日を計算するには

西暦元年1月1日(月曜)からの日数に基づく日付計算

400年毎に365*400+100-4+1=146097日増加.
途中100年毎に365*100+25-1=36524日増加.
途中4年毎に365*4+1=1461日増加.
途中毎年365日増加.

西暦元年1月1日から

西暦400*K年3月1日まで	146097*K-365+31+28+1日	146097*K-305
西暦100*L年3月1日まで	36524*L-365+31+28+1日	36524*L-305
西暦 4*M年3月1日まで	1461*M-365+31+28+1日	1461*M-305
西暦 N年3月1日まで	365*N-365+31+28+1日	365*N-305
m月1日まで	+[306*(m+1)/10]-122	

一般化

y = [(日数+305)/146097]*400 + [(日数+305)%146097/36524]*100
+ [(日数+305)%146097%36524/1461]*4 + [(日数+305)%146097%36524%1461/365]

西暦y年3月1日からの日数 d'= 日数 - (365*(y-1)+[y/4]-[y/100]+[y/400]+31+28+1);
閏年の場合 d'== -1となるため d'= 365, yを1減ずる
m は 3≦m≦14, [306*(m+1)/10]-122≦d', [306*(m+2)/10]-122>d'を満たすもの
西暦y年m月1日からの日数 d = d' - ([306*(m+1)/10]-122 - 1)
m > 12 であれば m を12減じ, yを1加える

§2. 8 詳細設計(ソースプログラム ymd1.c)

```
static char RcsId[] = "$Header$";
#include <stdio.h>
char *wday[] = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
main()
{
  int y, m, d;
  int days;

  if (scanf("%d", &days) != 1) {
    fprintf(stderr, "usage: days\n");
    exit(1);
  }

  if (days < 0) {
    fprintf(stderr, "error: days<0\n");
    exit(1);
  }

  y = (days+305)/146097*400
    + (days+305)%146097/36524*100
    + (days+305)%146097%36524/1461*4
    + (days+305)%146097%36524%1461/365;
  d = days - (365*(y-1)+y/4-y/100+y/400+31+28+1);
  if (d == -1) { y--; d=365; }

  for (m=3; m<15; m++) {
    if (306*(m+1)/10-122 <= d && 306*(m+2)/10-122 > d)
      break;
  }
  d -= 306*(m+1)/10-122 - 1;
  if (m > 12) { y++; m -= 12; }

  printf("%d %d %d %s\n", y, m, d, wday[days%7]);
  exit(0);
}
```

§2. 9 解説

```
static char RcsId[] = "$Header$";
#include <stdio.h>
char *wday[] = { "Sun", "Mon", ..., "Sat" };
main(){
  int y, m, d;
  int days;
  :
}
```

days1.cに同じ

§2. 9 解説(続き)

```
if (scanf("%d", &days) != 1) {
  fprintf(stderr, "usage: days\n");
  exit(1);
}
```

関数scanf()

- 入力文字列を10進数とみなして，変数daysに代入する.
- 正常に完了した引数の数を返す.
- 1でなければ，入力エラー

§2. 9 解説(続き)

```
if (days < 0) {
  fprintf(stderr, "error: days<0\n");
  exit(1);
}
```

入力データの検査

- daysが負であればエラー

§2. 9 解説(続き)

```
y = (days+305)/146097*400
  + (days+305)%146097/36524*100
  + (days+305)%146097%36524/1461*4
  + (days+305)%146097%36524%1461/365;
d = days-(365*(y-1)+y/4-y/100+y/400+31+28+1);
if (d == -1) { y--; d=365; }
```

日付計算式に基づくyおよびd'の計算

- 正しいyを求める.
- 西暦y年3月1日からの日数dを求める.
- dが-1である場合(閏年), 西暦y-1年2月29日に読み替える.

§2. 9 解説(続き)

```
for (m=3; m<15; m++) {
  if (306*(m+1)/10-122 <= d
   && 306*(m+2)/10-122 >  d)
    break;
}
d -= 306*(m+1)/10-122 - 1;
if (m > 12) { y++; m -= 12; }
```

日付計算式に基づくmおよびdの計算

- dを守備範囲とするmを求める.
- 西暦y年m月1日からの日数dを求める.
- mが12より大きい場合, 西暦y+1年m-12月d日に読み替える.

§2. 9 解説(続き)

```
printf("%d %d %d %s\n",y,m,d,wday[days%7]);
exit(0);
```

結果の表示

- 年月日および曜日を表示する.
- 正常終了したので0を返す.

§2. 10 機能の改良(ソースプログラム days2.c)

連続して日数を求める.

- scanf()の返り値が3である間繰り返す.

```
while (scanf("%d %d %d", &y, &m, &d) == 3) {
  if (m < 3) { y--; m += 12; }
  days = 365*y+y/4-y/100+y/400+306*(m+1)/10+d-428;

  if (days < 0) {
    fprintf(stderr, "error: days<0\n");
    continue;
  }

  printf("%d %s\n", days, wday[days%7]);
}
```

§2. 11 機能の改良(ソースプログラム ymd2.c)

連続して日付を求める.

- scanf()の返り値が1である間繰り返す.

```
while (scanf("%d", &days) == 1) {
  if (days < 0) {
    fprintf(stderr, "error: days<0\n");
    continue;
  }

  :

  printf("%d %d %d %s\n", y, m, d, wday[days%7]);
}
```

§2. 12 コンパイルと実行

1．コンパイルする.

```
% gcc days2.c -o days2
% gcc ymd2.c -o ymd2
```

2．RCSのidentを使ってみる.（ci後は以下のような表示が得られる）

```
% ident days2.c days2  ymd2.c ymd2
days2.c:
 $Header: days2.c,v 1.1 2000/08/25 10:46:59 ..

days2:
 $Header: days2.c,v 1.1 2000/08/25 10:46:59 ..

ymd2.c:
 $Header: ymd2.c,v 1.1 2000/08/25 10:46:59 ..

ymd2:
 $Header: ymd2.c,v 1.1 2000/08/25 10:46:59 ..
```

§2. 12 コンパイルと実行(続き)

3．実行

```
% ./days2
1900 2 29
693655 Thu
2000 2 29
730179 Tue
e

% ./ymd2
693655
1900 3 1 Thu
730179
2000 2 29 Tue
e
```

§2. 13 例題

days1.cとymd1.cの一部を使って，入力した年月日(例えばxxxx年2月29日)が実在するかどうかを判定するプログラムを作成することができる．処理手順を箇条書きにてわかりやすく説明せよ．

§2. 14 今日の課題

以上の方法により2000年問題の1つは解決できたが，日付にまつわるコンピュータ誤動作問題はこれから頻出する．今後対応が必要な2xxx年問題，および，有効な解決策について調査し，簡潔にまとめなさい．

宛先: nakashim@econ.kyoto-u.ac.jp
件名: unix2-学生番号

今日はここまで