プログラミング演習期末レポート

氏名：山本康隆
学籍番号：09424563

出題日：2013/05/31
提出日：2013/07/24
締切り日：2013/07/26

1 概要

本演習では，標準入力からのカンマ区切りの CSV 形式のファイル，または CSV データを入力し，それら 1 行ずつ読み込み，区切りごとに id,name,birth,addr,comment の 5 つの項目に分けて格納し，表示するプログラムを作成する．

1. 格納するデータを構造体として表現．指定されたデータ構造は以下の通りである．

ID	学校名	設立年月日	所在地	備考データ
32bit 整数	70bytes	struct date	70bytes	任意長

   この構造体を配列として 10000 件のデータを格納できるように宣言する．

2. 標準入力からの入力を CSV 形式として読み込み，上記に指定された構造体の配列に格納．SCV の形式は次の通り．

```
0,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Howks
1,Yamamoto Yasutaka,1993-07-12,Okayama,kojima
2,Kubo Shota,1993-04-16,Ehime,Matuyamakita
3,Oigawa Satoshi,1993-04-18,Shimane,Matueminami
:
```

3. %から始まる文を CSV 入力ではなく，コマンドとみなし，処理を行う．% Q, % C, % P, % R, % W, % F, % S, % E が入力された際はそれぞれの動作を行う．

コマンド	意味	備考
% Q	終了 (Quit)	
% C	登録件数の表示 (Check)	
% P n	先頭から n 件表示	n=0:全件表示,n<0:後ろから-n 件表示
% R file	file からデータ読み込み	
% W file	file へのデータの書き出し	
% F word	検索結果の表示	% P と同形式で表示
% S n	CSV の n 晩目の項目で整列	表示はしない
% E	指定 id の内容の編集	

2 プログラムの作成方針

今回のプログラムは，大きなプログラムとなるので，基本的にはいくつかの処理に分けて関数を作成する．それらを組み合わせることにより，1 つのプログラムを作成していく．処理は以下の通り．

(1) 格納を行う構造体の宣言部

(2) 標準入力からの文章を 1 行読み込む

(3) 標準入力データが CSV の場合の処理

(4) 標準入力データがコマンドの場合の処理

宣言部については概要で示した通りである．

```
struct date{
  int y;
  int m;
  int d;
};
struct profile{
  int id;
  char name[MAX_STR_LEN+1];
  struct date birth;
  char home[MAX_STR_LEN+1];
  char *comment;
};

struct profile profile_data_store[MAX_PROFILES];
```

(2) は主に get_line,subst,perse_line の部分である．標準入力・%R コマンドから読み込まれたファイルデータを fgets で char *line に 1 行分を読み込む．標準入力の場合*line と標準入力されたという stdin を送り，読み込みを行う．また% R コマンドでファイルから CSV を読み込む場合にはファイルポインタ fp と line を get_line に送り，入力されたファイルから 1 行がなくなるまで読み込みを行う．またこれらの 1 行の 1 文字目が%であれば 2 文字目以降のコマンド，引数を別関数におくり，各コマンドの処理を行う．そうでなければ，この 1 行を CSV 形式の文とみなし，’,’ を区切りとして 5 つの文字列として分割する．

(3) は new_profile,new_date,split の部分である．標準入力されたデータが CSV データだった場合，1 行毎に，文字列として分割し，これらを new_profile に送り，項目毎にデータを変換し，それぞれの構造体の型へと代入を行う．文字列の場合はそのまま代入を行うために strncpy，数値の場合は atoi を使い変換を行い代入・格納する．設立年月日の部分 (2013-6-6) の文字列も new_date に送り，’-’ を区切りとして同様に文字列として分割し，数値変換を行い格納する．概要で示した形式以外での入力，または無効な日付などを入力すると格納されないようになっている．
また，分割して送られてきた文字列は揮発性であるため，strncpy を使用し，メモリ間のコピーを行わなければならないことに注意しなければならない．

(4) は各コマンド実現部分であり，プログラムの終了，登録件数・登録項目の表示，ファイルの読み込み・書き込み，word 検索，ソート，編集を行う部分である．プログラムの終了は exit(0) を使用することにより，コマンド入力後に処理が停止する．登録件数は printf で表示する．登録項目の表示は 3 文字目以降の引数の件数分 (n 件) をそれぞれの場合分けをして printf で表示させる．場合分けとしては概要の示している通りである．ファイルの読み込みはファイルポインタを使用し，指定のファイルからデータを読み込む．また書き込みは既に登録されているデータを指定のファイルに CSV 形式で書き込みを行う．word 検索は入力された word と同じデータを探し%P と同様の形式で表示をする．ソートは入力された引数に応じて，指定の項目でアルファベット・数字順で並び替えられる．編集は指定の id の内容の書き換え，または削除を行う．

3　プログラムリスト及びその説明

プログラムリストを末尾に添付する．以下ではプログラムの主な構造について説明する．
まず，8-20 行付近は struct data,sturuct profile のデータ型の宣言部とそれを扱う関数の宣言部である．次に，subst,split を 29-59 行付近で宣言している．subst は str の文字列中の c1 を c2 へと変換する．ここでは’,’ を’\0’ へと変換している．split では送られてきた str の文字列中の区切り sep で分割し，subst と同様に’,’ へと’\0’ 変換し，分割したものを ret[] に格納している．これらの文字列を示す複数からなる配列を返す．また"2013-06-06"のような日付を分けるために分割文字を’-’ として struct_date で同様の処理を行っている．
次に 62-72,451-483 付近の get_line,perse_line,main では標準入力，ファイルから読み込まれた文章を 1 行ごと読み込み，解析する．データが%から始まっていればコマンド文字と引数を exec_command に送る．そうでなければ一行を new_profile に送る．

433-448 行付近では% P または他のコマンドを解釈し，適切な関数を呼び出し，実行を行う部分である．
また 120-204 行付近の new_profile,new_date では解析を行い，送られてきた一行を分割し，格納を行う．ここで，"2013/06/07"のように'-' で区切られず，間違った形式で入力された場合は処理されず，はじかれる．上記の split で分割した文字列配列を構造体の宣言部のデータ型に変換し，代入を行っている．文字列は strncpy，数値は atoi 関数を使用．これらを profile_data_store に格納している．profile_data_store に格納できる件数は最大 10000 件となっている

4 プログラムの使用方法

本プログラムは名簿データを管理するためのプログラムである．標準入力された CSV 形式のデータまたはファイル，%から始まるコマンドをうけ，処理し，処理結果を標準出力に出力する．入力形式については概要を参照．gcc によりコンパイル，実行を行う

```
% gcc report1.c
% ./a.out
```

まず% P で csv 形式ファイル test.csv を読み込むこととする．test.csv は以下の通りである．

```
2,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Howks
3,Yamamoto Yasutaka,1993-07-12,Okayama,Kurasikiminami
1,Kubo Shota,1993-04-16,Ehime,Matuyamakita
4,Oigawa Satoshi,1993-04-18,Shimane,Matueminami
```

% P により 1 行ずつ読み込みが行われ，登録される．また% P を使用せずに概要で示した形式で標準入力しても同様に登録される．入力において同じ id が既に存在する場合は入力されないようになっている．例を以下に示す．

```
例）1,Mori Masataka,1993-03,23,Okayama,Amaki
(error) 同じ id が存在します．
内容変更の場合は% E から編集を行ってください．
```

ここで以下のコマンドを入力するとする．

```
%P 0
%P 2
%P -2
%C
```

以下の出力を得る．

```
%P 0
(line1)
id:2
name:Takahashi Kazuyuki
Birth:1977-04-27
addr:Saitama
comment:Fukuoka Softbank Howks
(line2)
id:3
name:Yamamoto Yasutaka
Birth:1993-07-12
addr:Okayama
comment:Kurasikiminami
(line3)
id:1
name:Kubo Shota
Birth:1993-04-16
addr:Ehime
```

```
comment:Matuyamakita
(line4)
id:4
name:Oigawa Satoshi
Birth:1993-04-18
addr:Shimane
comment:Matueminami
%P 2
(line1)
id:2
name:Takahashi Kazuyuki
Birth:1977-04-27
addr:Saitama
comment:Fukuoka Softbank Howks
(line2)
id:3
name:Yamamoto Yasutaka
Birth:1993-07-12
addr:Okayama
comment:Kurasikiminami
%P -2
(line3)
id:1
name:Kubo Shota
Birth:1993-04-16
addr:Ehime
comment:Matuyamakita
(line4)
id:4
name:Oigawa Satoshi
Birth:1993-04-18
addr:Shimane
comment:Matueminami
```

登録件数：4 件

入力中の% P 2, % P 0, % P -2 はそれぞれ前から 2 件表示，全件表示，後ろから 2 件表示することを示している．% C は登録件数の表示を意味する．

次に以下の新しいデータを入力し，以下のコマンドを入力したとする．

```
5,Mori Masataka,1993-03-24,Okayama,Amaki
%W test.csv
```

% W は指定ファイルに書き込みを行うコマンドである．書き込みを行われた test.csv ファイルは以下のようになる．

```
2,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Howks
3,Yamamoto Yasutaka,1993-07-12,Okayama,Kurasikiminami
1,Kubo Shota,1993-04-16,Ehime,Matuyamakita
4,Oigawa Satoshi,1993-04-18,Shimane,Matueminami
5,Mori Masataka,1993-03-24,Okayama,Amaki
```

次に word 検索を行う．

```
%F Saitama
%F 1993-07-12
%F 4
```

% F の後に入力された word と一致するものを% P と同様の形式で表示する．出力結果は以下のようである．

```
%F Saitama
(line1)
id:2
name:Takahashi Kazuyuki
Birth:1977-04-27
addr:Saitama
comment:Fukuoka Softbank Howks
```

```
%F 1993-07-12
(line2)
id:3
name:Yamamoto Yasutaka
Birth:1993-07-12
addr:Okayama
comment:Kurasikiminami

%F 4
(line4)
id:4
name:Oigawa Satoshi
Birth:1993-04-18
addr:Shimane
comment:Matueminami
```

% S はソートコマンドである．% S の後に 1(id),2(name),3(birth),4(addr),5(comment) で並び替えが行われる．以下には 1(id) でソートした例を示す．

```
ソート前）
2,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Howks
3,Yamamoto Yasutaka,1993-07-12,Okayama,kojima
1,Kubo Shota,1993-04-16,Ehime,Matuyamakita
4,Oigawa Satoshi,1993-04-18,Shimane,Matueminami
5,Mori Masataka,1993-03-24,Okayama,Amaki

ソート後）
1,Kubo Shota,1993-04-16,Ehime,Matuyamakita
2,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Howks
3,Yamamoto Yasutaka,1993-07-12,Okayama,kojima
4,Oigawa Satoshi,1993-04-18,Shimane,Matueminami
5,Mori Masataka,1993-03-24,Okayama,Amaki
```

上記はソートしたものを% W を使い csv ファイルに書き込んだものを示している．% F を実行した後に表示などはされない．

% E は以下のような実行が行われる．

```
%E
id を入力してください．
 id:1
Before
1,Kubo Shota,1993-04-16,Ehime,Matuyamakita
After
```

上記は id:1 のプロフィールの編集の場合である．After の部分に

```
1,Ishii Isamu,1993-08-09,Okayama,Konan
```

と入力した場合，以下のように編集される．また前のソート後に編集を行ったものとする．

```
1,Ishii Isamu,1993-08-09,Okayama,Konan
2,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Howks
3,Yamamoto Yasutaka,1993-07-12,Okayama,kojima
4,Oigawa Satoshi,1993-04-18,Shimane,Matueminami
5,Mori Masataka,1993-03-24,Okayama,Amaki
```

以上がコマンドの使用例である．

5　プログラムの作成過程に関する考察

プログラムの作成過程での考察は，分割して返された文字列を代入する際に，揮発性であることに注意し，`strncpy` を使うようにした．数値の代入をするためには `atoi` 関数を使い値を直接代入するようにした．また `cmd_print` 関数内でははじめ，すべての `n` の場合分けを行いループを考え，その中のすべてで表示させていたが，記述量も多くなり，効率的では無いと考えたために，`print` で表示させる部分だけを別関数で作成し，ループ内に返されるように変更した `struct profile *newprofile` のように構造体の宣言にポインタがついているものがある．これはポインタを付けることによって，格納し，蓄積させたデータのすべてを返すのではなく先頭アドレスだけを返している．構造体内のすべての数値，文字列を返すよりも，効率が上がると考えたためである．`%P` で登録件数を越えた件数表示をしようとした際に表示させないようにしていたが，登録件数を確認し再度表示させる手

間を省くためにすべて表示させるようにするループを採用した．また最大の登録件数を越えて，新たなデータを登録しようとしたさいに，perse_line 内で条件文により，最大登録件数になってしまっていることを伝え，そこで処理を終えるようになっている．cmd_find 関数では入力された引数が文字列であるため，strcmp で比較を行うようにした．これはそれぞれの型の変換を行うことなく比較ができるためにこの様な比較方法を採用した．ソートにおいても別関数で合分けを行い，どのような大小関係（数字，アルファベット順）においても正，0，負のなどの統一の値が返せるようにした．そうすることにより，並び代えの処理も同じになり，記述量を減らせるようにした．またバブルソートを採用しているのは，なるべく簡単な記述を採用し，分かりやすいプログラムを作成しようと考えたためである．

6　得られた結果に関する，あるいは諮問に対する回答

考察問題

1：Size of 考察

```
void cmd_size(struct profile p){
printf("id=%d, name=%d, birth=%d, addr=%d, comment=%d\n",
         sizeof(p.id), sizeof(p.name), sizeof(p.birth), sizeof(p.home), sizeof(p.comment));
  printf("struct profile=%d\n", sizeof(p));
}
```

```
%O 0
id=4, name=70, birth=12, addr=70, comment=4
struct profile=164
```

上記は構造体の各メンバのサイズと struct progile のメモリ中のバイトの比較を行うプログラムを作成し，結果を出力したものである．結果を見てみると各メンバのサイズの合計は 160，struct profile のサイズは 164 であり，この 2 つのサイズは一致しない．これから考えられることは，メンバを隙間をあけ，配置しているために全体のメモリが大きくなっているからであると考える．この様な現象が起こっているのは，何か効率や，アクセスの面での利益があるからであると考える．

2：新たなコマンド
新たなコマンドとして%E，編集を作成した．プログラムの使用例でも記述したように，編集したい id のデータを再び入力し，上書きするコマンドである．入力した id を atoi 関数で数値変換し，既に登録されている id を数値として比較し，同じ id が見つかれば既に登録されているデータを CSV 形式で表示させる．表示例はプログラム使用例に示したとおりである．
その後，変更したいデータの入力を行う．この時に’\0’ のみや間違った形式で入力すると弾かれるようになっている．新たに入力したデータの分割には new_profile を使用しているが，分割が正常に終了すれば profile_data_nitem++を行うようになっている．そのため編集の際は成功した後に，profile_data_nitems--を行うようにし，データの総数は増えないようにしている．

3:プログラムの改良案と実装方法.
プログラムの改良案としては% P，% F などにおいて表示，検索を 1 度行うごとに処理を終了していたのでは，複数回処理を行いたい場合に効率が悪くなり，不便である．これらのプログラムの改良案としては，’\0’ が入力された際に終了するように処理する．間違って’\0’ を入力した場合に備え，’\0’ が入力された場合に終了の処理を行うか否かを 1:yes,2:no などで分岐するようにする．’\0’ の入力の場合に終了処理を行うように考えたのは% F において，何か特定の文字で (end など) で終了処理をしようとすると，その文字も検索結果に含まれてしまい，区別がつかない可能性を考えたからである．また同処理を行うならば% P においても同様の処理を行う仕様にしていたほうが使いやすいと考える．今回は時間の関係上，実装できていない．

4:エラー処理について
本課題において起こりうる問題は以下のようなものが考えられる．
・ありえない日付の入力された場合.
・間違った形式のデータが入力された場合.
・既に登録されたデータ (id) が入力された場合.
・%R においてファイルが開けなかった場合.
・検索で 1 つのデータも見つからなかった場合.
・%F で 1~5 以外の引数が入力された場合.

・無効なコマンドが入力された場合
・データが入力されなかった場合
・登録限度を越えた場合
などが考えられる．
まず優先順位から考えると，登録限度を越えた場合，データが入力されなかった場合，間違った形式のデータが入力された場合，ありえない日付が入力された場合，既に登録されたデータが入力された場合 (id)，無効なコマンドが入力された場合の 6 つは優先順位が高い．これらはデータ登録の時点で影響があるエラーであるからであると考えた．その後の優先順位はコマンド内などのエラーなどであるために順次行うのがよいと思われる．

登録件数においては，parseline に一行が送られ，new_profile に一行を送る前に profile_data_nitems と最大登録限度数の比較を行い，達していたらエラー表示を行う．同じくデータ入力がされなかった場合は入力データの最初の文字が'\0' であればエラー表示を行う．またありえない日付，間違った形式が入力された場合には split できちんと分けられなかった場合やありえない日付が入力された時点で NULL を返すようにした．入力データが最後まで問題なく格納されたときに profile_data_nitems++ が行われるようになっているので無効なデータの場合は登録されないようにしている．また既に登録されている id が重複して登録される場合には，new_pofile 内で分割してデータの格納を行う際に，既に登録されている id と入力された id の比較を行い，重複している場合には NULL を返し，エラー表示を行う．これは% E(編集) の際に一つのプロフィールだけを検索できる方が使用の際に効率が良いのではないかと考えたためこの様な行った．無効なコマンドが入力された際は exec_command において swich 関数で分岐を行っており，指定のコマンド以外が入力されるとエラー表示させれば良い．ここまでが優先順位の比較的高いエラーの処理であると考えられる．

その他のエラー処理はコマンド入力後などにコマンド内で起こるエラーであるために上記の 6 つよりは比較的優先順位の低いものであると考える．
%R,%W においてファイルが開けなかった場合，つまりファイルポインタが NULL の際にはエラー表示，-1 を返すようにし，処理から抜けるようになっている．また検索で 1 つのデータも見つからなかった場合については比較の際にデータが見つからなかった場合にあらかじめ定義したカウンタを一つ増やすようにし，それがもし profile_data_nitems と一致するならば検索できなかったものとし，エラー表示を行う．%F の際に 1~5 の引数が入力されなかった場合は引数を profile_compare に送る前に，1~5 以外であれば，エラー表示を行うようにした．
これらが今回行ったエラー処理である．

7 作成したプログラム

```
 1 #include<stdio.h>
 2 #include<stdlib.h>
 3 #include<string.h>
 4 #define MAX_LINE_LEN 1024
 5 #define MAX_STR_LEN 69
 6 #define MAX_PROFILES 10000
 7
 8 struct date{
 9   int y;
10   int m;
11   int d;
12 };
13
14 struct profile{
15   int id;
16   char name[MAX_STR_LEN+1];
17   struct date birth;
18   char home[MAX_STR_LEN+1];
19   char *comment;
20 };
21
22
23 struct profile profile_data_store[MAX_PROFILES];
24 int profile_data_nitems = 0;
25
26
27
28
29 int subst(char *str, char c1, char c2){
30   int n=0;
31   while(*str!='\0'){
32     if(*str == c1){
```

```
33        *str=c2;
34        n++;
35      }
36        *str++;
37    }
38    return n;
39  }
40
41
42
43  int split(char *str, char *ret[], char sep, int max){
44
45   int n=0;
46
47    ret[n]=str;
48    n = n + 1;
49
50    while(*str && n < max){
51      if(*str == sep){
52        *str = '\0';
53        ret[n] = str + 1;
54        n++;
55      }
56      str++;
57    }
58    return n;
59  }
60
61
62  int get_line(FILE *fp,char *line)
63  {
64    if(fgets(line,1025,fp) == NULL){
65
66      return 0;
67    }
68
69    subst(line,'\n','\0');
70
71    return 1;
72  }
73
74
75   /**
76    * Create a new date into D from STR like "2004-05-02".
77    * return: struct date *D itself
78    */
79
80
81  struct date *check_date(struct date *d)
82  {
83
84    if((d->m)>12) return NULL;
85
86
87    if(((d->y)%4) ==0){
88       if(((d->y)%400) == 0 || ((d->y)%100) != 0){
89
90         if((d->m) == 2 && (d->d)>29) return NULL;
91
92       }else{
93
94         if((d->m) == 2 && (d->d)>28) return NULL;
95       }
96     }else{
97         if((d->m) == 2 && (d->d)>28) return NULL;
98     }
99
100    if((d->m)== 4 ||
101       (d->m)== 6 ||
102       (d->m)== 9 ||
103       (d->m)== 11 ){
104      if((d->d)>30) return NULL;
105    }
106    if((d->m)==1 ||
107       (d->m)==3 ||
108       (d->m)==5 ||
```

```
109       (d->m)==7 ||
110       (d->m)==8 ||
111       (d->m)==10||
112       (d->m)==12){
113      if((d->d)>31) return NULL;
114      }
115
116
117    }
118
119
120 struct date *new_date(struct date *d, char *str)
121  {
122    char *ptr[3];
123
124    if (split(str, ptr, '-', 3) != 3){
125
126
127      return NULL;
128    }
129
130    d->y = atoi(ptr[0]);
131    d->m = atoi(ptr[1]);
132    d->d = atoi(ptr[2]);
133
134    return d;
135  }
136
137
138 struct profile *check_id(struct profile *p1)
139 {
140   int i;
141   struct profile *p2;
142
143   if(profile_data_nitems == 0){
144     return p1;
145   }
146
147   for(i=0;i<profile_data_nitems;i++){
148     p2=&profile_data_store[i];
149   if(p1->id == p2->id){
150     return NULL;
151       }
152   }
153
154
155   }
156  /**
157   * Create a new profile into P from CSV string like
158   *  "0,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Hawks".
159   * return: struct profile *P itself
160   */
161
162
163
164 struct profile *new_profile(struct profile *p, char *csv,int edt)
165  {
166    char *ptr[5];
167
168    if (split(csv, ptr, ',', 5) != 5){
169      fprintf(stderr,"(error) 入力形式を確認, またはデータを入力してください.
\n\n");
170      return NULL;
171    }
172
173    p->id = atoi(ptr[0]);
174    if(check_id(p) == NULL){
175      if(edt==1){
176      fprintf(stderr,"(error) 同じ id が存在します.\n 内容を編集する場合には% E
から行ってください.\n\n");
177      return NULL;
178      }
179    }
```

```
180
181     strncpy(p->name, ptr[1], MAX_STR_LEN);
182     p->name[MAX_STR_LEN] = '\0';
183
184     if (new_date(&p->birth, ptr[2]) == NULL ){
185       fprintf(stderr,"入力形式を確認してください. \n");
186       return NULL;
187     }
188     if(check_date(&p->birth) == NULL){
189       fprintf(stderr,"(error)%04d-%02d-%02dは無効な日付です. \n\n",(p->birth).y
 ,(p->birth).m,(p->birth).d);
190       return NULL;
191     }
192
193
194     strncpy(p->home, ptr[3], MAX_STR_LEN);
195     p->home[MAX_STR_LEN] = '\0';
196
197     p->comment = (char *)malloc(sizeof(char) * (strlen(ptr[4])+1));
198     strcpy(p->comment, ptr[4]);
199
200     profile_data_nitems++;
201
202     return p;
203
204 }
205
206
207 void cmd_quit()
208 {
209   exit(0);
210 }
211
212 void cmd_check()
213 {
214   printf("登録件数：%d 件\n",profile_data_nitems);
215 }
216
217 char *date_to_string(char buf[],struct date *date)
218 {
219   sprintf(buf,"%04d-%02d-%02d",date->y,date->m,date->d);
220   return buf;
221   }
222
223 void print_profile(int i,struct profile *p)
224 {
225   char date[11];
226   printf("(line%d)\n",i+1);
227   printf("id:%d\n",p->id);
228   printf("name:%s\n",p->name);
229   printf("Birth:%04d-%02d-%02d\n",(p->birth).y,(p->birth).m,(p->birth).d);
230   printf("addr:%s\n",p->home);
231   printf("comment:%s\n",p->comment);
232
233 }
234
235 int min(int a, int b)
236 {
237   if(a < b) return a;
238   return b;
239 }
240
241 int max(int a, int b)
242 {
243   if(a > b) return a;
244   return b;
245 }
246
247 void cmd_print(int nitems)
248 {
249   int i, start = 0, end = profile_data_nitems;
```

```
250
251   if(nitems>0){
252     end = min(nitems,profile_data_nitems);
253   }
254   if(nitems<0){
255     start = max(end - (-nitems),0);
256   }
257
258   for(i = start; i < end ; i++){
259     print_profile(i,&profile_data_store[i]);
260     printf("\n");
261   }
262 }
263
264 int cmd_read(char *filename)
265 {
266   FILE *fp;
267   char line[MAX_LINE_LEN+1];
268
269   fp = fopen(filename,"r");
270
271   if(fp == NULL){
272     fprintf(stderr,"(error) ファイルが開けません. \n\n");
273     return -1;
274   }
275
276   while(get_line(fp,line)){
277     parse_line(line);
278   }
279   fclose(fp);
280   return 0;
281 }
282
283 void fprint_profile_csv(FILE *fp,struct profile *p)
284 {
285   fprintf(fp,"%d,%s,%04d-%02d-%02d,%s,%s\n",p->id,p->name,(p->birth).y,p->birth.m
 ,(p->birth).d,(p->home),p->comment);
286 }
287
288 int cmd_write(char *filename)
289 {
290   FILE *fp;
291   char line[MAX_LINE_LEN+1];
292   int i;
293
294   fp = fopen(filename,"w");
295
296   if(fp == NULL){
297     fprintf(stderr,"(error) ファイルが開けません. \n\n");
298     return -1;
299   }
300
301   for (i = 0; i < profile_data_nitems; i++){
302     fprint_profile_csv(fp,&profile_data_store[i]);
303   }
304   fclose(fp);
305   return 0;
306 }
307
308
309 void cmd_find(char *word)
310 {
311   int i;
312   struct profile *p;
313   char s[8];
314   char birthday_str[11];
315   int n=0;
316
317   for(i = 0; i < profile_data_nitems; i++){
318       p = &profile_data_store[i];
319       sprintf(s,"%d",p->id);
```

```
320       date_to_string(birthday_str,&p->birth);
321
322 if(strcmp(s,word) == 0 ||
323    strcmp(p->name,word) == 0 ||
324    strcmp(birthday_str,word) == 0 ||
325    strcmp(p->home,word) == 0 ||
326    strcmp(p->comment,word) == 0){
327
328   print_profile(i,p);
329   printf("\n");
330 }else{
331   n++;
332 }
333   }
334   if(n = profile_data_nitems){
335     fprintf(stderr,"(error) 見つかりませんでした.\n\n");
336   }
337   }
338
339
340 // %S ソート
341 void swap_profile(struct profile *p1,struct profile *p2)
342 {
343   struct profile temp;
344
345   temp = *p1;
346   *p1 = *p2;
347   *p2 = temp;
348 }
349
350 int compare_date(struct date *d1,struct date *d2)
351 {
352   if(d1->y != d2->y) return d1->y - d2->y;
353   if(d1->m != d2->m) return d1->m - d2->m;
354   return d1->d - d2->d;
355 }
356
357 int profile_compare(struct profile *p1,struct profile *p2,int param)
358 {
359
360   switch(param){
361   case 1:
362     return p1->id - p2->id;
363   case 2:
364     return strcmp(p1->name,p2->name);
365   case 3:
366     return compare_date(&p1->birth,&p2->birth);
367   case 4:
368     return strcmp(p1->home,p2->home);
369   case 5:
370     return strcmp(p1->comment,p2->comment);
371   }
372 }
373
374 void cmd_sort(int param)
375 {
376   int i,j,cmp;
377   int left=0,right=profile_data_nitems - 1;
378
379   if(param>5 || 0>=param) fprintf(stderr,"(error)1~5 の数字を入力してくださ
い.\n");
380
381   for(i=left;i<=right;i++){
382       for(j=left;j<=right-1;j++){
383 cmp = profile_compare(&profile_data_store[j],&profile_data_store[j+1],param);
384 if(cmp > 0){
385   swap_profile(&profile_data_store[j],&profile_data_store[j+1]);
386
387 }
388       }
389     }
```

```
390 }
391 //%E 編集
392 void cmd_edit()
393 {
394   int i,id,edt=0;
395   char m[MAX_LINE_LEN],after[MAX_LINE_LEN];
396   char *ptr[5],*aft;
397   struct profile *p;
398
399   printf("id を入力してください.\n id:");
400   fgets(m,MAX_LINE_LEN + 1,stdin);
401   id = atoi(m);
402
403   for(i = 0; i < profile_data_nitems; i++){
404     p = &profile_data_store[i];
405
406     if(id == p->id){
407       printf("Before\n%d,%s,%04d-%02d-%02d,%s,%s\n",p->id,p->name,(p->birth).y
 ,p->birth.m,(p->birth).d,(p->home),p->comment);
408       printf("After\n");
409       fgets(after,MAX_LINE_LEN+1,stdin);
410
411
412       if(new_profile(&profile_data_store[i],after,edt)!=NULL){
413 profile_data_nitems--;
414       }
415     }
416   }
417 }
418
419
420 void cmd_size(struct profile p){
421
422 printf("id=%d, name=%d, birth=%d, addr=%d, comment=%d\n",
423          sizeof(p.id), sizeof(p.name), sizeof(p.birth), sizeof(p.home)
 , sizeof(p.comment));
424   printf("struct profile=%d\n", sizeof(p));
425
426 }
427
428
429
430
431
432 //コマンドの分岐
433 void exec_command(char cmd, char *param)
434 {
435
436   switch(cmd){
437   case'Q':cmd_quit();  break;
438   case'C':cmd_check(); break;
439   case'P':cmd_print(atoi(param)); break;
440   case'R':cmd_read(param);  break;
441   case'W':cmd_write(param); break;
442   case'F':cmd_find(param);  break;
443   case'S':cmd_sort(atoi(param));  break;
444   case'E':cmd_edit(); break;
445   case'O':cmd_size(profile_data_store[atoi(param)]); break;
446   default:fprintf(stderr,"error\n\n");
447   }
448 }
449
450
451 int parse_line(char *line)
452 {
453   int cmd,std=1;
454   char *param;
455
456   if(*line == '%'){
457
458     cmd = line[1];
```

```
459     param = &line[3];
460     exec_command(cmd,param);
461
462   } else if(*line == '\0'){
463     fprintf(stderr,"(error) データを入力してください. \n\n");
464     return 0;
465   } else if(profile_data_nitems == MAX_PROFILES){
466     fprintf(stderr,"(error) 登録限度を越えています. \n\n");
467   } else {
468     new_profile(&profile_data_store[profile_data_nitems],line,std);
469   }
470   return 0;
471 }
472
473
474 int main()
475 {
476   int n=0;
477   char line[MAX_LINE_LEN+1];
478   while (get_line(stdin,line)){
479
480     parse_line(line);
481     }
482   return 0;
483 }
```