

# プログラミング演習 期末レポート

学籍番号：09425566

氏名：戸塚佑太

出題日：2014/07/14

提出日：2014/07/14

締切り日：2014/07/14

## 1 概要

このレポートでは、標準入力からカンマ区切りの 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過程を示すものである。

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する。SCV は次の形式とする。

```
0,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Howks
1,Yuta Totsuka,1993-04-24,Okayama,Kurashiki
2,Kubo Shota,1993-04-16,Ehime,Matuyamakita
3,Oigawa Satoshi,1993-04-18,Shimane,Matueminami
:
:
```

3. % から始まる文は CSV 入力ではなくコマンドとみなして処理を行う。% Q, % C, % P, % R, % W, % F, % S, % E, % H コマンドを実装し、それぞれのコマンドが入力されたとき、次の動作を行うこととする。

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 からデータを読み込む	
% W file	file ヘデータを書き出す	
% F word	検索結果の表示	% P と同形式で表示
% S n	CSV の n 番目の項目でソート	表示はしない
% E	指定した id のデータを編集	
% H	ヘルプ画面の表示	

## 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、いくつかの処理に分けて関数を作成する。処理の概要は以下の通りに定め、下記でそれぞれについて解説する。

- (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

まず、(1) 格納を行う構造体の宣言部 については、概要で示した通りにデータを格納できるよう宣言する。

```
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) 標準入力からの文章を 1 行読み込む は主に `get_line`, `subst`, `perse_line` の部分で処理を行っている。標準入力されたデータを `char *line` で 1 行分読み込み、1 文字目が % であれば 2 文字目以降のコマンドと引数を別関数の引数とし、各コマンドに応じた処理を行う。また、1 文字目が % でない場合はこの 1 行を CSV 形式の文とみなし、カンマ ‘,’ を区切りとして 5 つの文字列として分割する。

(3) 標準入力データが CSV の場合の処理 は `new_profile`, `new_date`, `split` の部分で処理を行っている。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、項目毎に適切な方に変換し、それぞれ構造体のメンバに代入する。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` を使い変数変換を行い代入・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、数値変換を行ってから変数に格納する。また、分割して送られてきた文字列は `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) 標準入力データがコマンドの場合の処理 は各コマンドの実現部分であり、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) をそれぞれ場合分けして `printf` で表示させる。場合分けの方法は、概要の示している通りに行っている。また登録件数を越えた引数 (`|nitems|>n`) が送られた場合は `error` が表示されるようになっている。

### 3 プログラムリストおよび、その説明

完成したプログラムを末尾に添付する。このセクションでは、プログラムの主な構造について説明する。

まず、8-20 行では `struct data` 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 26-53 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 55-63, 425-443, 446~454 行の `get_line`, `perse_line`, `main` では標準入力され文章を 1 行ごと読み込み、解析し、データが % から始まっている場合は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。

また 142-180,102-115 行の `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` で分割した無事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 プログラムの使用例・テスト

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に応じた処理をし、処理結果を標準出力に表示する。入力形式については概要を参照。まず、本プログラム (`main.c`) を `gcc` によりコンパイルし、`a.out` という実行ファイルを作成する。`test.csv` という CSV ファイルの読み込み (入力) を行う場合は、下のように `./a.out | test.csv` と入力する。

```
% gcc main.c
% ./a.out < test.csv
```

`test.csv` は以下のものであった場合を想定する。

```
1,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Howks
2,Yuta Totsuka,1993-04-24,Okayama,Kurashiki
3,Kubo Shota,1993-04-16,Ehime,Matuyamakita
4,Oigawa Satoshi,1993-04-18,Shimane,Matueminami
%P 0
%P 2
%P -2
%P 5
%C
```

このとき以下のように、ユーザがより読み取りやすいように出力を得ることができる。

```
(line1)Id      : 1
Name   : Takahashi Kazuyuki
Birth  : 1977-04-27
Addr   : Saitama
Com.   : Fukuoka Softbank Howks

(line2)Id      : 2
Name   : Yuta Totsuka
Birth  : 1993-04-24
Addr   : Okayama
Com.   : Kurashiki

(line3)Id      : 3
Name   : Kubo Shota
Birth  : 1993-04-16
Addr   : Ehime
Com.   : Matuyamakita

(line4)Id      : 4
Name   : Oigawa Satoshi
Birth  : 1993-04-18
Addr   : Shimane
Com.   : Matueminami

(line1)Id      : 1
Name   : Takahashi Kazuyuki
Birth  : 1977-04-27
Addr   : Saitama
```

Com. : Fukuoka Softbank Howks

(line2)Id : 2  
Name : Yuta Totsuka  
Birth : 1993-04-24  
Addr : Okayama  
Com. : Kurashiki

(line3)Id : 3  
Name : Kubo Shota  
Birth : 1993-04-16  
Addr : Ehime  
Com. : Matuyamakita

(line4)Id : 4  
Name : Oigawa Satoshi  
Birth : 1993-04-18  
Addr : Shimane  
Com. : Matueminami

登録件数を確認してください。

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" する処理を呼び出すコマンドである。% C は登録件数の表示をする処理を呼び出すコマンドである。

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 ファイルは以下ようになる。

hashi 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

次に検索を行ってみる。

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

% F の後ろに入力された word と一致するものを % P と同様の形式で出力する。出力結果は以下ようになる。

%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 の後にそれぞれ対応するカラムの番号を入力することで、並び替えが行われる。(1:id, 2:name, 3:birth, 4:addr, 5:comment) 以下に、1 でソートした例を示す。

```
\hashi 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
```

また、% S では実際にソートされた結果は表示されない。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上記には示している。% 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 のデータの編集を行う場合である。After の後に

```
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、以前行ったソート後の状態であるとする。

```
shii 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
```

また、% H はヘルプ画面である。実行結果は以下の通りで、本プログラムにおけるコマンド一覧を確認することが出来るようになっている。

%H

```
# HELP
## Commands
- Q: Quit
- C: check the number of registered datas
- P n: Print n elements (n=0:all, n<0:from behind)
- R file: Read from file
- W file: Write as file
- F word: Search by word, print like P
- S n: Sort datas as the column of n
- H: Show usage of commands
```

以上がコマンドの使用例である。

## 5 プログラム作成における考察

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、`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` 関数内では初め、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。

`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よう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

`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。また今回のプログラムでは `n` 件の登録件数に対し、その件数を上回る件数の表示を行おうとすると、登録件数を確認するように促し、表示がされないようにしている。この場合に表示を行った場合に、多少分かりにくくなってしまうのでは無いかと考え、まず登録件数を確認するように促すようにした。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。

コマンド %E

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` が見つけ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。

#### コマンド% H

また、新たなコマンドとして% H も追加した。コマンドの確認を仕様書を見直して確認する必要があるよう、全てのコマンドを、プログラム内で参照できるように実装した。



## 7 作成したプログラムのソースコード

Listing 1: main.c

```
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 struct profile profile_data_store[MAX_PROFILES];
23 int profile_data_nitems = 0;
24
25 int subst(char *str, char c1, char c2){
26     int n=0;
27     while(*str!='\0'){
28         if(*str == c1){
29             *str=c2;
30             n++;
31         }
32         *str++;
33     }
34     return n;
35 }
36
37 int split(char *str, char *ret[], char sep, int max)
38 {
39     int n=0;
40
41     ret[n]=str;
42     n = n + 1;
43
44     while(*str && n < max){
45         if(*str == sep){
46             *str = '\0';
47             ret[n] = str + 1;
48             n++;
49         }
50         str++;
51     }
52     return n;
53 }
54
55 int get_line(FILE *fp, char *line)
56 {
57     if(fgets(line, 1025, fp) == NULL)
58         return 0;
59
60     subst(line, '\n', '\0');
61
62     return 1;
63 }
64
```

```

65  /**
66   * Create a new date into D from STR like "2004-05-02".
67   * return: struct date *D itself
68   */
69  struct date *check_date(struct date *d)
70  {
71
72      if((d->m)>12) return NULL;
73
74      if(((d->y)%4) == 0){
75          if(((d->y)%400) == 0 || ((d->y)%100) != 0){
76              if((d->m) == 2 && (d->d)>29) return NULL;
77          }else{
78              if((d->m) == 2 && (d->d)>28) return NULL;
79          }
80      }else{
81          if((d->m) == 2 && (d->d)>28) return NULL;
82      }
83
84      if((d->m) == 4 ||
85         (d->m) == 6 ||
86         (d->m) == 9 ||
87         (d->m) == 11 ){
88          if((d->d)>30) return NULL;
89      }
90
91      if((d->m) == 1 ||
92         (d->m) == 3 ||
93         (d->m) == 5 ||
94         (d->m) == 7 ||
95         (d->m) == 8 ||
96         (d->m) == 10 ||
97         (d->m) == 12){
98          if((d->d)>31) return NULL;
99      }
100 }
101
102 struct date *new_date(struct date *d, char *str)
103 {
104     char *ptr[3];
105
106     if (split(str, ptr, '-', 3) != 3){
107         return NULL;
108     }
109
110     d->y = atoi(ptr[0]);
111     d->m = atoi(ptr[1]);
112     d->d = atoi(ptr[2]);
113
114     return d;
115 }
116
117
118 struct profile *check_id(struct profile *p1)
119 {
120     int i;
121     struct profile *p2;
122
123     if(profile_data_nitems == 0){
124         return p1;
125     }
126
127     for(i=0; i<profile_data_nitems; i++){
128         p2=&profile_data_store[i];
129         if(p1->id == p2->id){
130             return NULL;
131         }
132     }

```

```

133
134
135 }
136
137 /**
138  * Create a new profile into P from CSV string like
139  * "0,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Hawks".
140  * return: struct profile *P itself
141  */
142 struct profile *new_profile(struct profile *p, char *csv,int edt)
143 {
144     char *ptr[5];
145
146     if (split(csv, ptr, ' ', 5) != 5){
147         fprintf(stderr,"error:invalid_style_of_input_or_data\n\n");
148         return NULL;
149     }
150
151     p->id = atoi(ptr[0]);
152     if(check_id(p) == NULL){
153         if(edt!=1){
154             fprintf(stderr,"error:ID_already_exists.\n\nTo_edit_the_data,_use_E_command\n");
155             return NULL;
156         }
157     }
158
159     strncpy(p->name, ptr[1], MAX_STR_LEN);
160     p->name[MAX_STR_LEN] = '\0';
161
162     if (new_date(&p->birth, ptr[2]) == NULL ){
163         fprintf(stderr,"invalid_style_of_input");
164         return NULL;
165     }
166     if(check_date(&p->birth) == NULL){
167         fprintf(stderr,"error:%04d-%02d-%02d_is_invalid_date\n\n",(p->birth).y, (p->birth).
168             m, (p->birth).d);
169         return NULL;
170     }
171
172     strncpy(p->home, ptr[3], MAX_STR_LEN);
173     p->home[MAX_STR_LEN] = '\0';
174
175     p->comment = (char *)malloc(sizeof(char) * (strlen(ptr[4])+1));
176     strcpy(p->comment, ptr[4]);
177
178     profile_data_nitems++;
179
180     return p;
181 }
182
183 void cmd_quit()
184 {
185     printf("Program_has_ended\n");
186     exit(0);
187 }
188
189 void cmd_check()
190 {
191     printf("Number_of_elements:%d\n\n",profile_data_nitems);
192 }
193
194 void print_profile(int i,struct profile *p)
195 {
196     char date[11];
197     printf("(line%d)\n",i+1);
198     printf("id:%d\n",p->id);
199     printf("name:%s\n",p->name);
200     printf("Birth:%04d-%02d-%02d\n", (p->birth).y, (p->birth).m, (p->birth).d);

```

```

200     printf("addr:%s\n",p->home);
201     printf("comment:%s\n",p->comment);
202 }
203
204 // %P: print
205 int min(int a, int b)
206 {
207     if(a < b) return a;
208     return b;
209 }
210
211 int max(int a, int b)
212 {
213     if(a > b) return a;
214     return b;
215 }
216
217 void cmd_print(int nitems)
218 {
219     int i, start = 0, end = profile_data_nitems;
220
221     if(nitems>0){
222         end = min(nitems,profile_data_nitems);
223     }
224     if(nitems<0){
225         start = max(end - (-nitems),0);
226     }
227
228     for(i = start; i < end; i++){
229         print_profile(i,&profile_data_store[i]);
230         printf("\n\n");
231     }
232 }
233
234 int cmd_read(char *filename)
235 {
236     FILE *fp;
237     char line[MAX_LINE_LEN+1];
238
239     fp = fopen(filename,"r");
240
241     if(fp == NULL){
242         fprintf(stderr,"error: could not read the file\n\n");
243         return -1;
244     }
245
246     while(get_line(fp,line)){
247         parse_line(line);
248     }
249     fclose(fp);
250     return 0;
251 }
252
253 // %W: write
254 void fprint_profile_csv(FILE *fp,struct profile *p)
255 {
256     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);
257 }
258
259 int cmd_write(char *filename)
260 {
261     FILE *fp;
262     char line[MAX_LINE_LEN+1];
263     int i;
264
265     fp = fopen(filename,"w");
266

```

```

267     if(fp == NULL){
268         fprintf(stderr,"error: could not open the file\n\n");
269         return -1;
270     }
271
272     for (i = 0; i < profile_data_nitems; i++){
273         fprintf_profile_csv(fp,&profile_data_store[i]);
274     }
275     fclose(fp);
276     return 0;
277 }
278
279 // %F: find
280 char *date_to_string(char buf[],struct date *date)
281 {
282     sprintf(buf,"%04d-%02d-%02d",date->y,date->m,date->d);
283     return buf;
284 }
285
286 void cmd_find(char *word)
287 {
288     int i;
289     struct profile *p;
290     char s[8];
291     char birthday_str[11];
292     int n=0;
293
294     for(i = 0; i < profile_data_nitems; i++){
295         p = &profile_data_store[i];
296         sprintf(s,"%d",p->id);
297         date_to_string(birthday_str,&p->birth);
298         if(strcmp(s,word) == 0 ||
299            strcmp(p->name,word) == 0 ||
300            strcmp(birthday_str,word) == 0 ||
301            strcmp(p->home,word) == 0 ||
302            strcmp(p->comment,word) == 0){
303             print_profile(i,p);
304             printf("\n");
305         }else{
306             n++;
307         }
308     }
309     if(n == profile_data_nitems){
310         fprintf(stderr,"error: could not find\n\n");
311     }
312 }
313
314
315 // %S sort
316 void swap_profile(struct profile *p1,struct profile *p2)
317 {
318     struct profile temp;
319
320     temp = *p1;
321     *p1 = *p2;
322     *p2 = temp;
323 }
324
325 int compare_date(struct date *d1,struct date *d2)
326 {
327     if(d1->y != d2->y) return d1->y - d2->y;
328     if(d1->m != d2->m) return d1->m - d2->m;
329     return d1->d - d2->d;
330 }
331
332 int profile_compare(struct profile *p1,struct profile *p2,int param)
333 {
334

```

```

335     switch(param){
336     case 1:
337         printf("Sorted by id");
338         return p1->id - p2->id;
339     case 2:
340         return strcmp(p1->name,p2->name);
341     case 3:
342         return compare_date(&p1->birth,&p2->birth);
343     case 4:
344         return strcmp(p1->home,p2->home);
345     case 5:
346         return strcmp(p1->comment,p2->comment);
347     }
348 }
349
350 void cmd_sort(int param)
351 {
352     int i,j,cmp;
353     int left=0,right=profile_data_nitems - 1;
354
355     if(param>5 || 0>=param) fprintf(stderr,"error: input a number of 1 to 5\n");
356
357     for(i=left; i<=right; i++){
358         for(j=left; j<=right-1; j++){
359             cmp = profile_compare(&profile_data_store[j],&profile_data_store[j+1],param);
360             if(cmp > 0){
361                 swap_profile(&profile_data_store[j],&profile_data_store[j+1]);
362             }
363         }
364     }
365 }
366
367 // %E edit
368 void cmd_edit()
369 {
370     int i,id,edt=0;
371     char m[MAX_LINE_LEN],after[MAX_LINE_LEN];
372     char *ptr[5],*aft;
373     struct profile *p;
374
375     printf("What is ID of the data you want to edit?\n id:");
376     fgets(m,MAX_LINE_LEN + 1,stdin);
377     id = atoi(m);
378
379     for(i = 0; i < profile_data_nitems; i++){
380         p = &profile_data_store[i];
381
382         if(id == p->id){
383             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);
384             printf("After\n");
385             fgets(after,MAX_LINE_LEN+1,stdin);
386
387             if(new_profile(&profile_data_store[i],after,edt)!=NULL){
388                 profile_data_nitems--;
389             }
390         }
391     }
392 }
393
394 // %H help
395 void cmd_help()
396 {
397     printf("\n# HELP\n");
398     printf("## Commands\n");
399     printf("-Q: Quit\n");
400     printf("-C: check the number of registered datas\n");
401     printf("-Pn: Print n elements (n=0: all, n<0: from behind)\n");

```

```

402     printf("-R_file:_Read_from_file\n");
403     printf("-W_file:_Write_as_file\n");
404     printf("-F_word:_Search_by_word,_print_like_P\n");
405     printf("-S_n:_Sort_datas_as_the_column_of_n\n");
406     printf("-H:_Show_usage_of_commands\n\n");
407 }
408
409 void exec_command(char cmd, char *param)
410 {
411     switch(cmd){
412         case'Q':cmd_quit(); break;
413         case'C':cmd_check(); break;
414         case'P':cmd_print(atoi(param)); break;
415         case'R':cmd_read(param); break;
416         case'W':cmd_write(param); break;
417         case'F':cmd_find(param); break;
418         case'S':cmd_sort(atoi(param)); break;
419         case'E':cmd_edit(); break;
420         case'H':cmd_help(); break;
421         default:fprintf(stderr,"error\n\n");
422     }
423 }
424
425 int parse_line(char *line)
426 {
427     int cmd,std=1;
428     char *param;
429
430     if(*line == '%'){
431         cmd = line[1];
432         param = &line[3];
433         exec_command(cmd,param);
434     }else if(*line == '\0'){
435         fprintf(stderr,"error:_no_input\n\n");
436         return 0;
437     }else if(profile_data_nitems == MAX_PROFILES){
438         fprintf(stderr,"error:_over_the_limit_of_datas\n\n");
439     }else {
440         new_profile(&profile_data_store[profile_data_nitems],line,std);
441     }
442     return 0;
443 }
444
445
446 int main()
447 {
448     printf("Program_has_sterted.Type_a_command_after_a_symbol_of_percent._To_show_help\n\n");
449     char line[MAX_LINE_LEN+1];
450     while(get_line(stdin,line)){
451         parse_line(line);
452     }
453     return 0;
454 }

```

---