プログラミング演習
中間レポート

学籍番号：09425566
氏名：戸塚佑太

出題日：2014/05/19
提出日：2014/05/26
締切り日：2014/05/26

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途中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 の形式は次の通り．

```
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. 今回は% Q, % C, % P コマンドのみ実装し，それぞれのコマンドは次の動作を行うよう実装する．

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

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 行付近は 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 26-56 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 58-67,195-226 付近の 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．
また 73-123 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 は以下のよう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”％Ｐ２”, ”％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.

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.

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.

7　作成したプログラムのソースコード

Listing 1: listManager.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  #define MAX_LINE_LEN 1024 // 一行の最大文字数
5  #define MAX_STR_LEN 69  // 氏名・住所の最大文字数
6  #define MAX_PROFILES 10000 // 蓄積させられる最大件数
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]; // グローバル変数
23 int profile_data_nitems = 0; // 登録%件数を数えるためのカウンタ C
24
25 int subst(char *str, char c1, char c2)
26 {
27   int n=0;
28   while(*str!='\0'){
29     if(*str == c1){
30       *str=c2;
31       n++;
32     }
33       *str++;
34   }
35   return n; //変換した回数の値を返す
36 }
37
38 /*文字列操作関数をへ','\0'*/
39 int split(char *str, char *ret[], char sep, int max)
40 {
41  int n=0;
42
43   ret[n]=str;
44   n = n + 1;
45
46   while(*str && n < max){
47     if(*str == sep){
48       *str = '\0';
49       ret[n] = str + 1;
50       n++;
51     }
52     str++;
53   }
54   return n;
55 }
56
57 /*一行を読み込みへ送る subst*/
58 int get_line(char *line)
59 {
60   if(fgets(line,1025,stdin) == NULL){
61     return 0;
62   }
63   subst(line,'\n','\0');
64   return 1;
```

```
65 }
66
67  /**
68   * Create a new date into D from STR like "2004-05-02".
69   * return: struct date *D itself
70   */
71 /*の分割，数値の代入 Birth*/
72 struct date *new_date(struct date *d, char *str)
73 {
74     char *ptr[3];
75
76     if (split(str, ptr, '-', 3) != 3)
77      return NULL;
78
79     d->y = atoi(ptr[0]);
80     d->m = atoi(ptr[1]);
81     d->d = atoi(ptr[2]);
82
83     return d;
84 }
85
86  /**
87   * Create a new profile into P from CSV string like
88   * "0,Takahashi Kazuyuki,1977-04-27,Saitama,Fukuoka Softbank Hawks".
89   * return: struct profile *P itself
90   */
91
92 /*形式の一文を分割，文字列，値の代入・格納 CSV*/
93 struct profile *new_profile(struct profile *p, char *csv)
94 {
95   char *ptr[5];
96
97   if (split(csv, ptr, ',', 5) != 5)
98       return NULL;
99
100    /* ID: id */
101    p->id = atoi(ptr[0]);
102
103    /* 学校名: name */
104    strncpy(p->name, ptr[1], MAX_STR_LEN);
105    p->name[MAX_STR_LEN] = '\0';
106
107    /* 設立年月日: birthday */
108    if (new_date(&p->birth, ptr[2]) == NULL)
109      return NULL; /* format error */
110
111    /* 所在地: home */
112    strncpy(p->home, ptr[3], MAX_STR_LEN);//上に同じ．
113    p->home[MAX_STR_LEN] = '\0';
114
115    /*備考*/
116    /*：指定バイト分の，メモリ領域を確保 malloc
117     strlen(ptr[4])分のメモリを確保して先頭アドレスを返す．+1返却された先頭アドレスを
118    (char*)とみなし p->に代入 comment*/
119    p->comment = (char *)malloc(sizeof(char) * (strlen(ptr[4])+1));
120    strcpy(p->comment, ptr[4]);
121
122    return p;
123  }
124
125
126 /*終了*/
127 void cmd_quit()
128 {
129   exit(0);
130 }
131
132 /*登録件数*/
```

```
133 void cmd_check()
134 {
135   printf("登録件数：%d件\n",profile_data_nitems);
136 }
137
138 /*先頭から件表示 n n = 0: 全件表示, n < 0 後ろから−件表示 n*/
139 char *date_to_string(char buf[], struct date *date)
140 {
141    sprintf(buf, "%04d-%02d-%02d", date->y, date->m, date->d);
142    return buf;
143 }
144
145 void print_profile(int i,struct profile *p) // 件表示 n
146 {
147  char date[11];
148
149    printf("(line%d)\n",i+1);
150    printf("Id␣␣␣␣:␣%d\n", p->id);
151    printf("Name␣␣:␣%s\n", p->name);
152    printf("Birth␣:␣%s\n", date_to_string(date, &p->birth));
153    printf("Addr␣␣:␣%s\n", p->home);
154    printf("Com.␣␣:␣%s\n", p->comment);
155 }
156
157 void cmd_print(int n) // で場合分け，ループ処理 n
158 {
159   int i;
160
161    if(n>0 && n<=profile_data_nitems){
162     for(i=0;i<n;i++){
163       print_profile(i,&profile_data_store[i]);
164       printf("\n");
165     }
166    }else if(n==0){
167
168     for(i=0;i<profile_data_nitems;i++){
169       print_profile(i,&profile_data_store[i]);
170       printf("\n");
171       }
172     }else if(n<0 && (-n)<=profile_data_nitems){
173
174     for(i=(profile_data_nitems+n);i<profile_data_nitems;i++){
175        print_profile(i,&profile_data_store[i]);
176        printf("\n");
177         }
178     }else{
179       printf("登録件数を確認してください．\n\n");
180       }
181 }
182
183 /*コマンド文字によって，適切な関数に引数などを送るり，処理を行う．*/
184 void exec_command(char cmd, char *param)
185 {
186   switch(cmd){
187   case'Q':cmd_quit(); break;
188   case'C':cmd_check(); break;
189   case'P':cmd_print(atoi(param)); break;
190
191   default:printf("error\n");
192   }
193 }
194
195 /*一行を読み込み，コマンド，形式の解析．それぞれの値を CSVexec_command,に送る．
        new_profile*/
196 int parse_line(char *line)
197 {
198   int cmd;
199   char *param;
```

```
200
201   if(*line == '%'){
202
203     cmd = line[1];
204     param = &line[3];
205     exec_command(cmd,param);
206   }else if(profile_data_nitems == MAX_PROFILES){
207     printf("登録限度を越えています．\n");
208   }else if(line[0] == '\0'){
209     return 0;
210     //printf("muri");
211   } else {
212     new_profile(&profile_data_store[profile_data_nitems++],line);
213 /*profile_data_store[profile_data_nitemsの先頭アドレスを送り，]を増やす．nitems1*/
214   }
215 }
216
217 /*一行を読み込み，へ送る．perse_line*/
218 int main()
219 {
220   int n=0;
221   char line[MAX_LINE_LEN+1];
222   while (get_line(line)){
223
224     parse_line(line);
225     }
226   return 0;
227 }
```