プログラミング演習 1
第 2 回レポート

氏名: 重近 大智 (SHIGECHIKA, Daichi)
学生番号: 09501527

出題日: 2020 年 04 月 30 日
提出日: 2020 年 05 月 07 日
締切日: 2020 年 05 月 13 日

1　概要

本演習では，名簿管理機能を有するプログラムを，C 言語で作成する．このプログラムは，標準入力から「ID, 氏名, 誕生日, 住所, 備考」からなるコンマ区切り形式（CSV 形式）の名簿データを受け付けて，それらをメモリ中に登録する機能を持つ．ただし，% で始まる入力行はコマンド入力と解釈し，登録してあるデータを表示したり整列したりする機能も持つ．

本レポートでは，演習中に取り組んだ課題として，以下の課題 1 から課題 3 についての内容を報告する．

課題 1　文字列操作の基礎：`subst` 関数と `split` 関数の実装
課題 2　構造体や配列を用いた名簿データの定義
課題 3　標準入力の取得と構文解析

また，取り組んだ課題のうち，特に以下の課題については，詳細な考察を行った．

課題 1　文字列操作の基礎：`subst` 関数と `split` 関数の実装
課題 3　標準入力の取得と構文解析

2　プログラムの作成方針

本演習で作成したプログラムが満たすべき要件と仕様として，「(1) 基本要件」と「(2) 基本仕様」を示す．

(1) 基本要件

1. プログラムは，その実行中，少なくとも 10,000 件の名簿データをメモリ中に保持できるようにすること．
2. 名簿データは，・・・・
3. プログラムとしての動作や名簿データの管理のために，・・・
   （a）プログラムの正常な終了
   （b）登録された・・・
4. 標準入力からのユーザ入力を通して，，，

(2) 基本仕様

1. 名簿データは，コンマ区切りの文字列（**CSV 入力**と呼ぶ）で表されるものとし，図 1 に示したようなテキストデータを処理できるようにする．
2. コマンドは，% で始まる文字列（**コマンド入力**と呼ぶ）とし，表 1 にあげたコマンドをすべて実装する
3. 1 つの名簿データは，C 言語の構造体 (`struct`) を用いて，・・・

3　プログラムの説明

プログラムリストは 7 章に添付している．プログラムは全部で 70 行からなる．以下では，1 節の課題ごとに，プログラムの主な構造について説明する．

3.1　文字列操作の基礎：subst 関数と split 関数の実装

まず，汎用的な文字列操作関数として，`subst()` 関数を 6–20 行目で宣言し，`split()` 関数を 22–39 行目で宣言している．また，これらの関数で利用するために，`<stdio.h>`というヘッダファイルをインクルードする．

`subst(str, C1, C2)` 関数は，`str` が指す文字列中の，文字 `C1` を文字 `C2` に置き換える．プログラム中では，`get_line()` 関数内の `fgets()` 関数で文字列の入力を受けるとき，末尾に付く改行文字を `NULL` 文字で置き換えるために使用している．呼び出し元には，文字を置き換えた回数を戻り値として返す．

`split(str, ret[], sep, max)` 関数は，他関数から渡された文字列中に文字変数 `sep` の文字に一致する文字があった場合，該当文字を `NULL` 文字で置き換え，該当文字の次の文字が格納されているメモリのアドレスを `ret[]` に書き込む．なお，`ret[0]` に格納されるアドレスの値は，`split()` 関数が呼び出された際の `str` の値である．以降，`sep` の文字に一致する文字があった場合，`ret[]` の添字を 1 ずつ増やしながらアドレスを格納していく．呼び出し元には，アドレスが格納されている `ret[]` の内，添字が最も大きいものの添字を戻り値として返す．

3.2　構造体や配列を用いた名簿データの定義

本名簿管理プログラムでは，構造体の配列を名簿データとして扱う．18–27 行目で，`date` 構造体を定義し，29–48 行目で，`profile` 構造体を定義している．この・・・・が，名簿データ 1 つに相当する．そして，xxx 行目の `xxxxx` 変数で，全名簿データを管理し，xxx 行目の `xxxxx` 変数で，名簿データの個数を管理する．

```
1:  5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,SEN Unit 2.0 Open
2:  5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,01955 641225 ...
3:  5100224,Canisbay Primary School,1928-7-5,Canisbay Wick,01955 611337 Primary 56 3...
4:  5100321,Castletown Primary School,1913-11-4,Castletown Thurso,01847 821256 01847...
```

図 1　名簿データの CSV 入力形式の例．1 行におさまらないデータは... で省略した．

表 1　実装するコマンド

コマンド	意味	備考
%Q	終了　(Quit)	
%P n	先頭から n 件表示　(Print)	n が 0 → 全件表示， n が負 → 後ろから $-n$ 件表示
*****	（サンプルのため　省略）	

date 構造体の定義にあたっては，・・・（以降，サンプルのため，省略）
・・・（サンプルのため，省略）

3.3 標準入力の取得と構文解析

標準入力を取得するための get_line() 関数は 41–47 行目で宣言している．標準入出力のため，<stdio.h>というヘッダファイルをインクルードする．

get_line(line) 関数は，標準入力 stdin を fgets() 関数で取得し，1024 文字以上を越えた場合は，次の行として処理を行うことでバッファオーバーランを防止している．標準入力が NULL または，ESC キーを 1 文字目に入力した場合は，呼び出し元に戻り値 0 を返す．それ以外の場合，subst() 関数で末尾の改行文字を NULL 文字に置き換えた後，呼び出し元に戻り値 1 を返す．

4 プログラムの使用法と実行結果

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．CSV 形式のデータと % で始まるコマンドを標準入力から受け付け，処理結果を標準出力に出力する．入力形式の詳細については，2 節で説明した．

プログラムは，Cent OS 7.6.1810(Core) で動作を確認しているが，一般的な UNIX で動作することを意図している．なお，以降の実行例における，行頭の$記号は，Cent OS 7.6.1810(Core) におけるターミナルのプロンプトである．

まず，gcc でコンパイルすることで，プログラムの実行ファイルを生成する．ここで，-Wall とは通常は疑わしいものとみなされることのない構文に関して警告を出力するためのオプションであり，-o とは出力ファイル名を指定するオプションである．これらのオプションをつけることで，疑わしい構文を発見し，任意の出力ファイル名を指定することができる．

```
$ gcc -Wall -o program1 program1.c
```

次に，プログラムを実行する．以下の実行例は，プログラム実行中の動作例を模擬するため，任意の csv ファイルを標準入力のリダイレクションにより与えることで，実行する例を示している．通常の利用においては，キーボードから文字列を入力してもよい．

```
$ ./program1 < csvdata.csv
```

プログラムの出力結果として，CSV データの各項目が読みやすい形式で出力される．例えば，下記の csvdata.csv に対して，

```
0,Takahashi Kazuyuki,1977-04-27,3,Saitama,184,78
10,Honma Mitsuru,1972-08-25,2,Hokkaidou,180,78
%S3
%P0
```

以下のような出力が得られる．

```
code:   10
name:   Honma Mitsuru
bday:   1972/08/25
type:   2
home:   Hokkaidou
height: 180
weight: 78

code:   0
name:   Takahashi Kazuyuki
```

```
bday:   1977/04/27
type:   3
home:   Saitama
height: 184
weight: 78
```

まず，入力データについて説明する．入力中の最初の 2 行で，2 つの CSV データを登録している．CSV データは，・・・で示したように，7 つの項目からなる．3 行目の %S3 は，これまでの入力データを 3 番目の項目（生年月日）でソートすることを示している．4 行目の %P0 は，入力した項目の全ての項目 (1–7) を表示することを示している．

（※サンプルのため省略）

5 考察

3 章のプログラムの説明，および，4 章の使用法と実行結果から，演習課題として作成したプログラムが，1 章で述べた基本要件と基本仕様のいずれも満たしていることを示した．ここでは，個別の課題のうち，以下の 3 つの項目について，考察を述べる．

1. 文字列操作の基礎：subst 関数と split 関数の実装
2. 標準入力の取得と構文解析
3. ・・・（サンプルのため，省略）

5.1 「文字列操作の基礎：subst 関数と split 関数の実装」に関する考察

subst() 関数が，他関数から文字列の配列を受け取ることを想定して，ポインタを使用して他関数内の配列を参照できるようにした．また，入力文字列の途中に NULL 文字が出現することは標準入力では起こりにくいため，文字置き換えのループ処理の継続条件に NULL 文字を用いることで，確実に入力文字列の終端である NULL 文字手前まで文字の置き換えを行えるようにした．c1 と c2 に同じ文字が入力された場合，文字の置き換え処理は実質行われないに等しい．12 行目に break 文を書くことにより，文字の置き換えと置き換えられた文字のカウントを行わず，処理の簡略化及び高速化ができた．

5.2 「標準入力の取得と構文解析」に関する考察

fgets() 関数で標準入力を取得する際，直接入力では標準入力を NULL にすることができず，改行文字が入ってしまうため，ESC キーを 1 文字目に入力することにより，入力待ちを終了できるように，44 行目に戻り値の条件を追加した．fgets() 関数により取得した文字列は，終端が改行文字になってしまうため，subst() 関数を呼び出し，改行文字を NULL 文字に置き換える処理も含めている．この処理を行うことで，subst() 関数のループ処理の継続条件，split 関数のループ処理の継続条件や printf() 関数による文字列の出力などで文字列が扱いやすくなった．

5.3 「・・・(サンプルのため，省略)」に関する考察

(※サンプルのため省略)

6 感想

(※サンプルのため省略)

7　作成したプログラム

作成したプログラムを以下に添付する．なお，1 章に示した課題については，4 章で示したようにすべて正常に動作したことを付記しておく．

subst，split，get_line 関数を含むプログラムのソースコード (70 行)

```
 1        #include <stdio.h>
 2
 3        #define ESC 27                                          /*文字列 ESC を ESC の ASCII
コードで置換*/
 4        #define MAX_LINE 1025                                   /*文字配列 LINE の
最大入力数の指定用*/
 5
 6        int subst(char *str, char c1, char c2)
 7        {
 8          int i;                                                /*for ループ用*/
 9          int c = 0;                                            /*置き換えた文字数の
カウント用*/
10          for(i = 0; *(str + i) != '\0'; i++)                   /*入力文字列の終端に辿り
着くまでループ*/
11            {
12              if(c1 == c2) break;                               /*見た目上文字列に変化が
ないとき*/
13              if(*(str + i) == c1)                              /*(str + i) の文字が c1 の
文字と同じとき*/
14                {
15                  *(str + i) = c2;                              /*(str + i) の文字を c2 の
文字に置き換える*/
16                  c++;                                          /*置き換えた文字を数える*/
17                }
18            }
19          return c;                                             /*置き換えた文字数を戻り値
とする．*/
20        }
21
22        int split(char *str, char *ret[], char sep, int max)
23        {
24          int i;                                                /*for ループ用*/
25          int c = 0;                                            /*ポインタの配列の指定用*/
26
27          ret[0] = str;                                         /*ret[0] に str の先頭アドレス
を代入*/
28
29          for(i = 0; *(str + i) != '\0'&& c < max; i++)         /*c が max より小さいか，入力
文字列の終端に辿り着いていないときループ*/
30            {
31              if(*(str + i) == sep)                             /*(str + i) が sep のとき*/
32                {
33                  *(str + i) = '\0';                            /*(str + i) に NULL を代入*/
34                  c++;
35                  ret[c] = str + (i + 1);                       /*ret[c] に NULL 文字の"次の"
アドレスを代入*/
36                }
37            }
38          return c;                                             /*文字列をいくつに分割したか
を戻り値とする*/
39        }
40
41        int get_line(char *line)
42        {
43          if(fgets(line, MAX_LINE, stdin) == NULL) return 0;    /*入力文字列が空のとき，
0 を戻り値とする．入力文字列は 1024 文字*/
44          if(*line == ESC) return 0;                            /*直接入力のとき，入力文字列
を空にできないため，ESC キーの単独入力により 0 を戻り値とする*/
45          subst(line, '\n', '\0');                              /*subst 関数により，入力の
```

```
改行文字を終端文字に置き換える*/
    46          return 1;                                   /*入力文字列が存在したとき，
1 を戻り値とする*/
    47        }
    48
    49        int main(void)
    50        {
    51          char line[MAX_LINE] = {0};                  /*入力文字列は最大 1024 文字*/
    52          char *ret[10];
    53          char sep = ',';                             /*csv ファイルからの入力を
想定しているため，カンマ*/
    54          int max = 10;
    55          int c, i, a = 1;
    56
    57          while(get_line(line))                       /*get_line 関数を呼び出し，
戻り値が 0 ならループを終了*/
    58            {
    59              printf("line number:%d\n", a);
    60              printf("input:\"%s\"\n", line);
    61              c = split(line, ret, sep, max);         /*split 関数を呼び出す*/
    62              for(i = 0;i <= c; i++)
    63                {
    64                  printf("split[%d]:\"%s\"\n", i, ret[i]);
    65                }
    66              printf("\n\n");                         /*見やすさのために改行*/
    67              a++;
    68            }
    69          return 0;
    70        }
```

参考文献

[1] 平田富雄，アルゴリズムとデータ構造，森北出版，1990.

[2] 著者名，書名，出版社，発行年.

[3] WWW ページタイトル，URL，アクセス日.