プログラミング演習 1
第 3 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 13 日
提出日: 2020 年 05 月 16 日
締切日: 2020 年 05 月 20 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 169 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 22–36 行目で宣言し，`split()` 関数を 38–55 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 57–63 行目で宣言している．構文解析のための parse_line() 関数は，65–82 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，制御文字 ESC 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．

parse_line(line) 関数は，他関数からの文字列配列をポインタ line で取得し，入力内容が特定のコマンドとその引数であるか，単なる文字列であるかを判定し，それぞれ exec_command() 関数を呼び出すか，new_profile() 関数を呼び出す．get_line() 関数で入力された入力文字列の 1 文字目が % である場合，2 文字目の文字を文字変数 cmd に代入，4 文字目以降の文字列をポインタ param で参照できるようにする．入力文字列がコマンドで引数が与えられなかった場合，ポインタ param は文字列 (Null Parameter) を参照し，ポインタ param が何も指していない状態を防止している．exec_command(cmd, *param) 関数が呼び出す一部の自作関数では引数が必要となるので，この処理を行う．必要に応じて exec_command() 関数内でキャストを行ってから，コマンドの処理をする関数に引数を送る．

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 program1.c -o program1 -Wall
```

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 [3]．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 test.csv に対して，

```
Microsoft,Windows,7
%Q
y
```

以下のような出力が得られる．

```
line number:1
input:"Microsoft,Windows,7"
split[0]:"Microsoft"
split[1]:"Windows"
split[2]:"7"

Do you want to quit?(y/n)
quit success.
```

まず，入力データについて説明する．入力中の最初の 1 行で，1 つの CSV データを登録している．CSV データは自動的に分割されて表示される．%Q コマンドによる終了を実施する際に確認メッセージがあるため，最後に y を csv ファイルから入力している．

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らないため，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．28 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

get_line() 関数内の fgets() 関数で標準入力を取得する際，直接入力では Ctrl+D で標準入力を NULL にすることができるが，デバッグ用の機能として ESC 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 60 行目に戻り値の条件を追加した．ESC は制御文字であり，ファイルリダイレクションによりファイルから入力されることはないため，これをデバッグ用に追加した [4]．ファイルリダイレクションを 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．parse_line() 関数では，例外が発生する可能性が多いので，3 文字目以降に入力がない場合や 3 文字目に誤って入力してしまった場合などの対策が必要だと考えた．3 文字目以降に入力がない場合を考慮し，仮の (Null Parameter) を設定している．また parse_line() 関数自体が，他関数から文字列の配列の先頭アドレスを受け取り，ポインタ line に格納するが，他関数にポインタ line をそのまま渡す可能性もあったため，ポインタの値自体をインクリメントしたり，デクリメントしたりすることは避けるとともに，表記を subst() 関数や split() 関数と統一している．

5.2.1 「・・・(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

毎回の講義でプログラムのソースコードを追記していくとき，どの関数を上から順に書けばいいかを考えていたが，煩雑化してしまうため，関数プロトタイプ宣言をすることで見やすくなり作業がしやすくなった [2]．今までの自作プログラムでは，関数プロトタイプ宣言が必要になるほど自作関数を用意することがなかったので，実際のプログラミングの経験になった．また，プログラム全体を通して表記のゆれが少なくなるように，多くの自作関数において，ポインタの指し先をずらすために int 型変数 i を使用したり，各関数内で似たような役割を持つ変数の変数名を統一したりした．今回の課題プログラムの作成を通して，ポインタへの理解が一層深まり，ポインタのポインタやポインタの配列に関して理解することができたと思う [5, 6]．printf() 関数の書式指定 %s の引数として，任意のアドレスを代入するが，ポインタの配列*ret[] の場合，どのような形で書けばいいのか悩んだ．printf() 関数の書式指定 %c の引数では，引数が文字コードなので，ポインタを利用して引数を指定する場合*を付ける必要があった．一方書式指定 %s の場合は，引数がアドレスのため*は必要なく，単に ret[] と書くので混乱した．

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        #include <stdlib.h>                                        /*exit 関数用*/
    3
    4        #define ESC 27                                             /*文字列 ESC を ESC の ASCII
コードで置換*/
    5        #define MAX_LINE 1025                                      /*文字配列 LINE の最大入力数
の指定用*/
    6
    7        /*関数プロトタイプ宣言（煩雑化防止）*/
    8        int subst(char *str, char c1, char c2);
    9        int split(char *str, char *ret[], char sep, int max);
   10        int get_line(char *line);
   11        void parse_line(char *line);
   12        void exec_command(char cmd, char *param);
   13        void cmd_quit();
   14        void cmd_check();
   15        void cmd_print();
   16        void cmd_read();
   17        void cmd_write();
   18        void cmd_find();
   19        void cmd_sort();
   20        void new_profile(char *line);
   21
   22        int subst(char *str, char c1, char c2)
   23        {
   24          int i;                                                   /*for ループ用*/
   25          int c = 0;                                               /*置き換えた文字数のカウ
ント用*/
   26          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*/
   27            {
   28              if(c1 == c2) break;                                  /*見た目上文字列に変化が
ないとき*/
```

```
    29              if(*(str + i) == c1)                                    /*(str + i) の文字が c1 の
文字と同じとき*/
    30                {
    31                  *(str + i) = c2;                                    /*(str + i) の文字を c2 の
文字に置き換える*/
    32                  c++;                                                /*置き換えた文字を数える*/
    33                }
    34            }
    35          return c;                                                   /*置き換えた文字数を戻り値
とする. */
    36        }
    37
    38        int split(char *str, char *ret[], char sep, int max)
    39        {
    40          int i;                                                      /*for ループ用*/
    41          int c = 0;                                                  /*ポインタの配列の指定用*/
    42
    43          ret[0] = str;                                               /*ret[0] に str の先頭アドレ
スを代入*/
    44
    45          for(i = 0; *(str + i) != '\0'&& c < max; i++)               /*c が max より小さいかつ入力
文字列の終端に辿り着いていないときループ*/
    46            {
    47              if(*(str + i) == sep)                                   /*(str + i) が sep のとき*/
    48                {
    49                  *(str + i) = '\0';                                  /*(str + i) に NULL を代入*/
    50                  c++;
    51                  ret[c] = str + (i + 1);                             /*ret[c] に NULL 文字の"次の"
アドレスを代入*/
    52                }
    53            }
    54          return c;                                                   /*文字列をいくつに分割したか
を戻り値とする*/
    55        }
    56
    57        int get_line(char *line)
    58        {
    59          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*/
    60          if(*line == ESC) return 0;                                  /*ESC を 1 文字目に入力すること
により 0 を戻り値とする（デバッグ用）*/
    61          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*/
    62          return 1;                                                   /*入力文字列が存在したとき，
1 を戻り値とする*/
    63        }
    64
    65        void parse_line(char *line)
    66        {
    67          char cmd;                                                   /*% の次の 1 文字を格納用*/
    68          char *param;                                                /*コマンドのパラメータとなる
文字列へのポインタ用*/
    69          char *buffer = "(Null Parameter)";                          /*例外処理用*/
    70
    71          if(*line == '%')                                            /*入力文字列の 1 文字目が % のとき*/
    72            {
    73              cmd = *(line + 1);                                      /*cmd に入力文字列の 2 文字目
の値を代入*/
    74              if(*(line + 3) != '\0') param = (line + 3);             /*ポインタ line に 3 を足した
アドレスをポインタ param に代入*/
    75              else param = buffer;                                    /*入力文字列にパラメータ部が
無いとき，文字列"(Null Parameter)"のアドレスをポインタ param に代入*/
    76              exec_command(cmd, param);
    77            }
    78          else                                                        /*入力がコマンドではないとき*/
    79            {
    80              new_profile(line);
    81            }
```

```
 82        }
 83
 84        void exec_command(char cmd, char *param)
 85        {
 86          switch (cmd) {
 87          case 'Q': cmd_quit();   break;
 88          case 'T': printf("Parameter test:\"%s\"\n", param); break;   /*ポインタ param の参照先
から後ろに向かって，NULL まで文字列を表示する（デバッグ用）*/
 89          case 'C': cmd_check();  break;
 90          case 'P': cmd_print();  break;
 91          case 'R': cmd_read();   break;
 92          case 'W': cmd_write();  break;
 93          case 'F': cmd_find();   break;
 94          case 'S': cmd_sort();   break;
 95          default: fprintf(stderr, "%%%c command is invoked with arg: \"%s\"\n", cmd, param);
 break;/*エラーメッセージを表示*/
 96          }
 97        }
 98
 99        void cmd_quit()
100        {
101          printf("Do you want to quit?(y/n)\n");                    /*確認メッセージ*/
102          if(getchar() == 'y')
103            {
104              printf("quit success.\n");
105              exit(0);
106            }
107          getchar();                                                /*getchar での入力時に改行文字が
残ってしまうため*/
108          printf("quit cancelled.\n");
109        }
110
111        void cmd_check()
112        {
113          fprintf(stderr, "Check command is invoked.\n");
114        }
115
116        void cmd_print()
117        {
118          fprintf(stderr, "Print command is invoked.\n");
119        }
120
121        void cmd_read()
122        {
123          fprintf(stderr, "Read command is invoked.\n");
124        }
125
126        void cmd_write()
127        {
128          fprintf(stderr, "Write command is invoked.\n");
129        }
130
131        void cmd_find()
132        {
133          fprintf(stderr, "Find command is invoked.\n");
134        }
135
136        void cmd_sort()
137        {
138          fprintf(stderr, "Sort command is invoked.\n");
139        }
140
141        void new_profile(char *line)
142        {
143          char *ret[11];
144          char sep = ',';                                           /*csv ファイルからの入力を想定して
いるため，カンマ*/
145          int max = 10;
```

```
  146           int c, i;
  147           static int a = 1;                                   /*値を main 関数終了時まで保持する
必要があるため，static int 型*/
  148
  149           printf("line number:%d\n", a);
  150           printf("input:\"%s\"\n", line);
  151           c = split(line, ret, sep, max);                     /*split 関数を呼び出す*/
  152           for(i = 0;i <= c; i++)
  153             {
  154               printf("split[%d]:\"%s\"\n", i, ret[i]);
  155             }
  156           printf("\n");                                       /*見やすさのために改行*/
  157           a++;
  158         }
  159
  160         int main(void)
  161         {
  162           char LINE[MAX_LINE] = {0};                          /*入力文字列（1 行分）は main 関数で
管理*/
  163
  164           while(get_line(LINE))                               /*文字配列 LINE に文字列を入力する
(get_line 関数)*/
  165             {
  166               parse_line(LINE);                               /*入力文字列がある場合，構文解析
を行う (parse_line 関数)*/
  167             }
  168           return 0;
  169         }
```

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] 林晴比古，明快入門 C，SB クリエイティブ，2013

[3] 入出力のリダイレクションとパイプ，http://web.sfc.keio.ac.jp/ manabu/command/contents/pipe.html，2020．05．14.

[4] IT 用語辞典 E-Words ASCII 文字コード，http://e-words.jp/p/r-ascii.html，2020．05．14

[5] １０－３．ポインタと文字列，http://www9.plala.or.jp/sgwr-t/c/sec10-3.html，2020．05．14.

[6] 4.3 ポインタ配列，http://cai3.cs.shinshu-u.ac.jp/sugsi/Lecture/c2/e_04-03.html，2020．05．14.