プログラミング演習 1
第 4 回レポート

氏名: 池田 海斗 (IKEDA, Kaito)
学生番号: 09501502

出題日: 2020 年 05 月 20 日
提出日: 2020 年 05 月 24 日
締切日: 2020 年 05 月 27 日

1　概要

本演習では，C 言語を用いて名簿管理プログラムの制作を行う．このプログラムではコマンド入力によるデータの読み込み・書き出しを行ったり，整列・検索を行うことができる．また，標準入力からコンマで区切られた文字列を入力しデータの登録を行うことができる．

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 6 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
課題 4　CSV データ登録処理の実装
課題 5　コマンド中継処理の実装
課題 6　コマンドの実装：`%P` コマンド

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 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

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，「ID, 氏名, 誕生日, 住所, 備考」を，1 つのデータとして扱えるようにすること．
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，以下の機能を持つコマンドを実装すること．
   （a）プログラムの正常な終了
   （b）登録された名簿データのデータ数表示
   （c）名簿データの全数表示，および，部分表示

(d) 名簿データのファイルへの保存，および，ファイルからの復元

(e) 名簿データの検索と表示

(f) 名簿データの整列

4. 標準入力からのユーザ入力を通して，データ登録やデータ管理等の操作を可能とすること．
5. 標準出力には，コマンドの実行結果のみを出力すること．

(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，構造を持ったデータとしてプログラム中に定義し，利用する．
4. 全名簿データは，“何らかのデータ構造” を用いて，メモリ中に保持できるようにする．
5. コマンドの実行結果以外の出力は，標準エラー出力に出力する．

3 プログラムの説明

プログラムリストは 6 章に添付している．プログラムは全部で 138 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 66–79 行目で宣言し，`main()` 関数を 130–136 行目に記述している．また，これらの関数で利用するために，`stdio.h`，`stdlib.h`，`string.h` のヘッダファイルを読み込んでいる．

```
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)	
`%C`	登録内容のチェック	(Check)	1 行目に登録数を必ず表示
`%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
`%R file`	$file$ から読込み	(Read)	
`%W file`	$file$ への書出し	(Write)	
`%F word`	検索結果を表示	(Find)	`%P` と同じ形式で表示
`%S n`	CSV の n 番目の項目で整列	(Sort)	表示はしない

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() 関数と split() 関数の中で，文字 “\n”，“,” を終端文字 “\0” 置き換えるために，この関数を呼び出している．この関数では，文字 C1 を文字 C2 に置き換えた回数を返り値とする．

split(STR, RET, SEP, MAX) 関数では，STR が指す文字列を，最大 MAX 個まで文字 SEP の箇所で区切り，配列 RET に格納していく．プログラム中では，get_line() 関数で取得した 1 行を，カンマで区切りで配列に格納する際に呼び出される．また，ハイフン区切りで入力される生年月日のデータを，年・月・日に分けて配列に格納する際にも用いる．この関数では，配列に分割された個数を返り値とする．

3.2 構造体や配列を用いた名簿データの定義

本プログラムでは，構造体の配列に名簿データを格納していく．7–11 行目で，date 構造体を定義し，13–19 行目で, profile 構造体を定義している．ID, 氏名, 誕生日, 住所, 備考の 5 つの組み合わせで 1 つの名簿データとなる．そして，21 行目の profile_data_store 変数で全名簿データを管理し，22 行目の profile_data_nitems 変数で，名簿データの個数を管理する．名簿データの誕生日の箇所にあたっては date 構造体を用い，年, 月, 日を別々に保存しておく．

3.3 標準入力の取得と構文解析

get_line(LINE) 関数は，標準入力で読み込まれた名簿データを 1 行ずつ取り出し，LINE に代入する．読み込む行がない場合，文字列が 1024 文字を超えてバッファオーバーランをする場合，ユーザが改行コードを入力した場合に返り値 0 を返す．

parse_line(LINE) 関数では，get_line(LINE) 関数で読み込まれた 1 行がコマンドか入力データであるかで条件分岐を行う．コマンドの場合，コマンド名と引数に分けてその値を exec_command(CMD, PARAM) 関数に引き渡し，入力データであった場合は new_profile(PROFILE_DATA_STORE, LINE) 関数で登録を行う．

3.4 CSV データ登録処理の実装

new_profile(PROFILE_DATA_STORE, LINE) 関数内では，グローバル変数 profile_data_store の任意の配列番号のポインタを受け取り，そこに profile_data_nitems 番目の要素としてデータを保存する．また誕生日に関しては，split 関数を用いて年・月・日に細分化しデータを格納する．

4 プログラムの使用法と実行結果

4.1 プログラムの概要

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．コマンドを入力する際には，% で始まる英字を標準入力に打ち込み，引数で詳細な指示を行う．コマンドの詳細は 2 節に記述してある．また標準入力からのデータ入力も可能としている．

4.2 実行環境

プログラムは，MacOS Catalina 10.15.4 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，MacOS Catalina 10.15.4 におけるターミナルのプロンプトである．

4.3 コンパイル方法

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 とは警告オプションを全て有効にするためのオプションであり，-o とは実行ファイルの名前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，コードの視認性を高めたり無駄なコードを省くことができ，他のソースコードの実行ファイルとの識別が容易である．

```
% gcc -Wall -o eop_04_09501502 eop_04_09501502.c
```

4.4 実行方法

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データの入力を模擬するため，CSV ファイルを標準入力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，%R file によりデータを読み込む．

```
$ ./eop_04_09501502.out < stdin.csv
```

4.5 出力結果

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 stdin.csv に対して,

```
5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,SEN Unit 2.0 Open
%P contoso
5100224,Canisbay Primary School,1928-7-5,Canisbay Wick,01955 611337 Primary 56 3.5 Open
%C
%Q
```

以下のような出力が得られる．

```
%P command is invoked with arg: 'contoso'
%C command is invoked with no arg
```

まず，入力データについて説明する．コマンドは %P, %C, %Q の 3 つを記述している．また，この CSV ファイルには 2 件のデータを保存しており，それぞれの項目はカンマで区切られている．%Q コマンドはプログラムの終了を意味している．

また今回はデータの登録のみをバックグラウンドで行っている．出力結果については，コマンド名とその引数を表示させている．

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. 構造体や配列を用いた名簿データの定義

3. 標準入力の取得と構文解析

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

5.1.1 「subst 関数」に関する考察

ここでは subst 関数について考察を行う．練習問題の解答例を見てみると，main 関数内で複数の文字列を変換できるようにループさせており，また配列に保存した複数の文字を置換した結果も出力できるようにしてある．このことから，最終的に完成するプログラムでは，配列に保存された文字列を複数個 subst 関数にかけるのではないかと考察できる．

また，for 文で囲まれた部分に着目すると，他の言語での foreach 文に似たような動作をしていることがわかった．

5.1.2 「split 関数」に関する考察

次に，split 関数についての考察を行う．今回工夫したところは，カンマを全て終端文字に置き換えて split 関数に渡すことで，文字列の先頭にポインタを合わせることで一気に文字列をコピーできることである．その後 23 行目で，ポインタを文字数 +1(終端文字) 移動させることで，次の文字列の文頭にポインタを移動している．

5.2 「構造体や配列を用いた名簿データの定義」に関する考察

構造体 date・profile についての考察を行う．構造体 date について，年・月・日の 3 つを int 型で保存してある．また構造体 profile 内では，ID を int 型，名前・住所・備考を char 型で保存してある．誕生日については，構造体 date で birthday というメンバ変数を用いて格納してある．

5.3 「標準入力の取得と構文解析」に関する考察

5.3.1 「get_line 関数」に関する考察

get_line 関数についての考察を行う．今回重要なポイントとなってくるところは，fgets の最大文字数を 1024 ではなく 1026 に設定したところである．まず改行文字が格納されるので，1024 文字に +1 してある．また 1024 文字以上であることを確認するために，文字を 1 文字多く取得しそこに文字があるかで返り値を変えている．しかし，strlen 関数は終端文字をカウントしないので，strlen 関数の値が 1025 文字以上を含まないようにしている．

5.3.2 「parse_line 関数」に関する考察

parse_line 関数についての考察を行う．まずコマンドか否かの判別は，文字頭の値が % で始まるかどうかで行っている．これは，データ入力の場合文字列の始めは [0-9] で始まるので，上記の条件に含まれることはないからである．96 行目では，ポインタの 4 文字目以降からヌル終端文字までを配列に格納している．

5.4 「CSV データ登録処理の実装」に関する考察

new_profile 関数についての考察を行う．今回この関数を作成するにあたって一番苦戦したところは，atoi() 関数の存在を知ることであった．配列の要素に 1 桁ずつの値が入っているので，それぞれに 10^n をかけて和を求める関数を最初は制作していた．その後 C 言語関数リファレンスで該当する関数を探し，ようやく atoi() 関数を発見した．また，誕生日の年・月・日を区切る処理も split 関数を再利用した．

5.5 「コマンド中継処理の実装」に関する考察

exec_command 関数についての考察を行う．この関数の場合，if 文の条件分岐よりも swich 文を使う方が短いコードで書けるので採用した．ポインタでパラメータを渡すことで，複数文字の入力にも対応している．また想定外の入力に対しても，処理を実行せずにエラー文を返す処理を挟んでいる．

6 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．

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_LINE_LEN 1024

struct date {
    int y;
    int m;
    int d;
};

struct profile {
    int id;
    char name[70];
    struct date birthday;
    char address[70];
    char note[80];
};

int profile_data_nitems = 0;
struct profile profile_data_store[10000];

void cmd_quit(void) {
    exit(0);
}

void cmd_check(char cmd) {
    fprintf(stderr, "%%%c command is invoked with no arg\n", cmd);
}

void cmd_print(char cmd, char *param) {
    fprintf(stderr, "%%%c command is invoked with arg: '%s'\n", cmd, param);
}

void cmd_read(char cmd, char *param) {
    fprintf(stderr, "%%%c command is invoked with arg: '%s'\n", cmd, param);
}

void cmd_write(char cmd, char *param) {
    fprintf(stderr, "%%%c command is invoked with arg: '%s'\n", cmd, param);
}

void cmd_find(char cmd, char *param) {
    fprintf(stderr, "%%%c command is invoked with arg: '%s'\n", cmd, param);
}

void cmd_sort(char cmd, char *param) {
    fprintf(stderr, "%%%c command is invoked with arg: '%s'\n", cmd, param);
}

void exec_command(char cmd, char *param) {
```

```
53      switch (cmd) {
54          case 'Q': cmd_quit();   break;
55          case 'C': cmd_check(cmd);  break;
56          case 'P': cmd_print(cmd, param);  break;
57          case 'R': cmd_read(cmd, param);   break;
58          case 'W': cmd_write(cmd, param);  break;
59          case 'F': cmd_find(cmd, param);   break;
60          case 'S': cmd_sort(cmd, param);   break;
61          default:
62          printf("Unregistered Command Is Entered.\n");
63      }
64  }
65
66  int subst(char *str, char c1, char c2) {
67      int diff = 0;
68      char *p;
69
70      p = str;
71      while (*p != '\0') {
72          if (*p == c1) {
73              *p = c2;
74              diff++;
75          }
76          p++;
77      }
78      return diff;
79  }
80
81  int split(char *str, char *ret[], char sep, int max) {
82      int i, count = 0;
83      subst(str, sep, '\0');
84      for (i=0; i<max; i++) {
85          ret[i] = str;
86          str += strlen(str)+1;
87          count++;
88      }
89      return count;
90  }
91
92  int get_line(char *line) {
93      if (fgets(line, MAX_LINE_LEN + 1, stdin) == NULL || strlen(line) > MAX_LINE_LEN
94                                                                          || *line == '\n') {
95          return 0;
96      } else {
97          subst(line, '\n', '\0');
98          return 1;
99      }
100 }
101
102 void new_profile(struct profile *profile_data_store, char *line) {
103     int max_line = 5, max_date = 3;
104     char *ret[80] = {0}, *date[80] = {0}, sep_line = ',', sep_date = '-';
105
106     split(line, ret, sep_line, max_line);
107     split(ret[2], date, sep_date, max_date);
108
109     profile_data_store->id = atoi(ret[0]);
110     strcpy(profile_data_store->name, ret[1]);
111
112     profile_data_store->birthday.y = atoi(date[0]);
113     profile_data_store->birthday.m = atoi(date[1]);
114     profile_data_store->birthday.d = atoi(date[2]);
115
116     strcpy(profile_data_store->address, ret[3]);
117     strcpy(profile_data_store->note, ret[4]);
118 }
119
120 void parse_line(char *line) {
```

```
    char cmd, param[80] = {0};
    if (*line == '%') {
        cmd = *(line+1);
        strcpy(param, line+3);
        exec_command(cmd, param);
    } else {
        new_profile(&profile_data_store[profile_data_nitems++], line);
    }
}

int main(void) {
    char line[MAX_LINE_LEN + 1];
    while (get_line(line)) {
        parse_line(line);
    }
    return 0;
}

```

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] C 言語の atoi で出来ること, https://arma-search.jp/article/clanguage-atoi, 2020/05/20.