プログラミング演習 1
中間レポート

氏名: 今田　将也 (Imada, Masaya)
学生番号: 09430509

出題日: 2019 年 04 月 10 日
提出日: 2019 年 04 月 24 日
締切日: 2019 年 05 月 08 日

1　はじめに

プログラミング演習 1 での名簿管理プログラムの作成課題に取り組むにあたり, 必要な関数を作成した.

2　作成した関数の説明

2.1　subst 関数

関数 `int subst(char *str,char c1,char c2)`

行数 4 章のソースコードの 43 行目から 53 行目に記述してある.

概要　引数として与えられた文字列 `str` の特定の文字を別の文字に置き換え, 置き換えた文字数をカウントする.

戻り値 整数型で引数 `str` の置き換えた文字数を返す.

引数 `char *str` は置き換えたい元の文字列を与える. `char c1` は置き換え対象の文字を与える. `char c2` は置き換え後の文字を与える.

使用例

```
int a = 0;
printf("before:%s\ncount:%d\n",test,a);
a=subst(''test'',’t’,’f’);
printf("after:%s\ncount:%d\n",test,a);
```

関数 `subst` の作成にあたっては, 扱う引数が文字列と文字とがあることに注意しなければならない. 文字列の場合には引数として `str` の最初の文字をポインタとして与えている.
4 章の 26 行目からの while 文の判定式は, `*str` のみでも問題はないが, 後に見返す際に少しでもわかりやすくするためにあえて空文字を指定した.
関数の役割を区別するために `subst` 関数には結果を画面に表示する機能はつけていない.

2.2 split 関数

関数 `int split(char *str,char *ret[],char sep, int max)`

行数 4 章のソースコードの 55 行目から 81 行目に記述してある.

概要 引数として与えられた文字列 `str` を指定された文字 `sep` で文字列 `str` の終わりまで順に区切っていく.

戻り値 整数型で引数 `str` を分割した回数を返す.

引数 `char *str` は分割したい元の文字列を与える. `char *ret[]` は分割後の文字列を保存する. `char sep` は区切る対象の文字を与える. `int max` は区切る最大の回数を与える.

使用例

```
printf("test %s\n",str);
count=split(str,ret,',',max);
for(int i = 0; i < count; i++){;
    printf("%d:%s\n",i+1, ret[i]);
}
printf("count is %d.\n\n",count);
```

関数 `split` の内容としては，分割したい文字列のポインタアドレスを分割した文字列を保存する `ret` に格納している．そうすることで，元の文字列 `str` をコピーしなくとも分割後の文字列を保存することが可能になっている.

2.3 error_split 関数

関数 `void error_split(int check)`

行数 4 章のソースコードの 83 行目から 97 行目に記述してある.

概要 引数として与えられた整数値 `int check` の値に応じてエラー処理を行う.

戻り値 なし.

引数 `int check` はエラー処理を行わせたい値を与える.

使用例

```
if(count > max)count = -2;
error_split(count);
```

関数 `error_split` は関数 `split` の分割数が `max` より大きくなった際のエラー処理を行わせている．関数 `split` 内でエラー処理を行わせない理由としては関数の役割が曖昧になることを回避するためである.

2.4 testprint_split 関数

関数 void testprint_split(char *str)

行数 4 章のソースコードの 99 行目から 113 行目に記述してある.

概要 関数 split の内容をテストして結果を表示するための関数である.

戻り値 なし.

引数 この引数は前述の関数 split の引数を引き継ぐためのものである.

使用例

```
char test1[] = "";//分割したい文字列
char test2[] = ",,,";
char test3[] = ",oka,yama";
char test4[] = "o,ka,ya,,ma,a";

testprint_split(test1);
testprint_split(test2);
testprint_split(test3);
testprint_split(test4);
```

関数 testprint_split は関数 split が正しく目的通りに動作しているかのテストを目的として作成した．split 内では関数 error_split 同様に関数の役割を区別するために，画面表示機能を持たせてた本関数に分離した．引数を変えるだけでよいため，テストしたい文字列ごとに関数 split を適用させ，分割後の文字列を表示することを繰り返し記述する必要がなくなる．関数内に分割後の文字列 ret を定義した．また，最大分割数をグローバル変数 static int maxsplit =5 として別に定義している.

2.5 get_line 関数

関数 int get_line(char *input)

行数 4 章のソースコードの 115 行目から 125 行目に記述してある.

概要 引数として標準入力から与えられた文字列を取得し，文字列 char *input に格納する.

戻り値 空文字もしくは改行のみの文字列が与えられた場合 0 を返し失敗を意味する．それ以外の場合は 1 を返し成功を意味する.

引数 char *input は標準入力から入力された文字列の先頭ポインタを与える.

使用例

```
int i=0;
printf("please input line:");
while (get_line(input)) {
    printf("******test %d*****\n", ++i);
```

```
        testprint_split(input);
        printf("input line:");
    }
```

関数 get_line は，標準入力から 1 行分を読み込ませるための関数である．標準入力からの入力が空文字の場合は，エラー表示を行わせて失敗を意味する 1 を返している．なお，改行のみの入力の場合もエラーを返すようにしている．また，成功した際は，\n を関数 subst を用いて，空文字へと置き換えている．この関数単体では，繰り返し入力ができないため，繰り返し入力させる際は使用例のように，while 文を用いることに注意されたい．

2.6 testprint_get_line_split 関数

関数 void testprint_get_line_split(char *input)

行数 4 章のソースコードの 115 行目から 125 行目に記述してある．

概要 引数から与えられた文字列 char *input の値に応じてテストを入力が尽きるまで行う．

戻り値 なし．

引数 char *input は文字列配列の先頭アドレスを与える．

使用例

```
char input[LIMIT+1];
testprint_get_line_split(input);
```

関数 get_line と関数 split を結合させて意図した動作をしているか確認するために実装した．while 文の条件判定により繰り返し入力を可能にしている．その中で，split の動作を確認し，結果を表示させている．

3 感想

今回作成した関数 split は，ポインタにポインタのアドレスを紐づけて文字列を丸々メモリにコピーするような手間が省けるようにしたり，本来必要ではないが，後に変更しやすいよう関数に役割を持たせるために関数を分割するということを意識して取り組めたように思う．ポインタは理解を深められていないところが多いため改めて学習に励みたい．今後の課題としては，関数 get_line の入力をファイルからの入力に対応させていくことがあげられる．名簿管理プログラムの完成に向けて努力していきたい．

4 作成したプログラム

```
1 /*
2  * File:   meibo.c
3  * Author: 09430509
4  *
5  * Created on 2019/04/10
```

```
 6  * update on 2019/04/24
 7  */
 8
 9 #include <stdio.h>
10
11 #define LIMIT 1024
12
13 int subst(char *str,char c1,char c2);
14
15 int split(char *str,char *ret[],char sep,int max);
16 void testprint_split(char *str);
17 void error_split(int check);
18
19 int get_line(char *input);
20 void testprint_get_line_split(char *input);
21
22 static int maxsplit = 5;
23
24 int main(void){
25     char test1[] = "";//分割したい文字列
26     char test2[] = ",,,";
27     char test3[] = ",oka,yama";
28     char test4[] = "o,ka,ya,,ma,a";
29     //入力用文字配列
30     char input[LIMIT+1];
31
32     testprint_split(test1);
33     testprint_split(test2);
34     testprint_split(test3);
35     testprint_split(test4);
36
37     testprint_get_line(input);
38
39     return 0;
40
41 }
42
43 int subst(char *str,char c1,char c2){
44     int count = 0;
45     while(*str != '\0'){
46         if(*str == c1){
47             *str = c2;
48             count++;
49         }
50         str++;
51     }
52     return count;
53 }
54
55 int split (char *str,char *ret[],char sep,int max){
56     int count = 0;//分割数
57
58     while (1) {
59         if(*str == '\0') {
60             break;//からもじなら抜ける
61         }
62
63         ret[count++] = str;//str をいじれば ret も変わるように分割後の文字列にはポイン
タを入れる
64
65         while( (*str != '\0') && (*str != sep) ){//区切り文字が見つかるまでポインタ
すすめる
```

```
 66             str++;
 67         }
 68
 69         if(*str == '\0') {
 70             break;//区切り文字がなかったら抜ける＝文字列はそのまま
 71         }
 72
 73         *str = '\0';//必ず区切り文字のはずだからくぎる
 74     str++;//インクリメントさせる
 75     }
 76     //if(count<max) count = -1;
 77     if(count>max)count = -2;
 78     error_split(count);
 79
 80     return count;
 81 }
 82
 83 void error_split(int check){
 84     switch(check){
 85         case -1:
 86             printf("luck.\n");
 87             break;
 88
 89         case -2:
 90             printf("over.\n");
 91             break;
 92
 93         default:
 94             break;
 95     }
 96     return;
 97 }
 98
 99 void testprint_split(char *str){
100     int count;
101     char *ret[maxsplit];
102
103     printf("test %s\n",str);
104
105     count=split(str,ret,',',maxsplit);
106
107     for(int i = 0; i < count; i++){
108         printf("%d:%s\n",i+1, ret[i]);
109     }
110
111     printf("count is %d.\n\n",count);
112     return;
113 }
114
115 int get_line(char *input){
116
117     if (fgets(input, LIMIT + 1, stdin) == NULL || input[0] == '\n'){//何かしら入力
させて、改行のみは認めない
118         printf("error:NULL or input is \\n.\n");
119         return 0; /* 失敗 orEOF */
120     }
121
122     subst(input, '\n', '\0');
123     return 1; /*成功*/
124
125 }
126
```

```
127 void testprint_get_line_split(char *input){
128     int i = 0;
129 
130     printf("please input line:");
131     while (get_line(input)) {
132         printf("******test %d*****\n", ++i);
133         testprint_split(input);
134         printf("input line:");
135     }
136     return;
137 }
```