# 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 18 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 33 行目から 43 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 45 行目から 59 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 に格納すしている．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．あなお，後述の error_split で分割数が max より大きくなった際の処理を行わせている.

### 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 73 行目から 87 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 のエラー処理を，split 内で行わせるとことで関数の役割が曖昧になることを回避するために作成した.

### 2.4 testprint_split 関数

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

行数 4 章のソースコードの 89 行目から 102 行目に記述してある.

概要　関数 split の内容をテストするための関数.

戻り値　なし.

引数 これら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

```
int max = 5;
char test1[] = "";
char test2[] = ",,,";
char test3[] = ",oka,yama";
char test4[] = "o,ka,ya,,ma,a";
char *ret[max];

testprint_split(test1,ret,',',max);
testprint_split(test2,ret,',',max);
testprint_split(test3,ret,',',max);
testprint_split(test4,ret,',',max);
```

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.

## 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．前回同様未だに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.

## 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 2019/04/18
 7  */
 8
 9 #include <stdio.h>
10
```

```
11  int subst(char *str,char c1,char c2);
12  int split(char *str,char *ret[],char sep,int max);
13  void testprint_split(char *str,char *ret[],char sep,int max);
14  void error_split(int check);
15
16  int main(void){
17      int max = 5;
18      char test1[] = "";//分割したい文字列
19      char test2[] = ",,,";
20      char test3[] = ",oka,yama";
21      char test4[] = "o,ka,ya,,ma,a";
22      char *ret[max];//分割後に入れる文字配列個数
23
24      testprint_split(test1,ret,',',max);
25      testprint_split(test2,ret,',',max);
26      testprint_split(test3,ret,',',max);
27      testprint_split(test4,ret,',',max);
28
29      return 0;
30
31  }
32
33  int subst(char *str,char c1,char c2){
34      int count = 0;
35      while(*str != '\0'){
36          if(*str == c1){
37              *str = c2;
38              count++;
39          }
40          str++;
41      }
42      return count;
43  }
44
45  int split (char *str,char *ret[],char sep,int max){
46      int count = 0;//分割数
47
48      while (1) {
49          if(*str == '\0') {
50              break;//からもじなら抜ける
51          }
52
53          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
54
55          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
56              str++;
57          }
58
59          if(*str == '\0') {
60              break;//区切り文字がなかったら抜ける＝文字列はそのまま
61          }
62
63          *str = '\0';//必ず区切り文字のはずだからくぎる
64              str++;//インクリメントさせる
65      }
66      //if(count<max) count = -1;
67      if(count>max)count = -2;
68      error_split(count);
69
70      return count;
```

```
 71  }
 72
 73  void error_split(int check){
 74      switch(check){
 75          case -1:
 76              printf("luck.\n");
 77              break;
 78
 79          case -2:
 80              printf("over.\n");
 81              break;
 82
 83          default:
 84              break;
 85      }
 86      return;
 87  }
 88
 89  void testprint_split(char *str,char *ret[],char sep,int max){
 90      int count;
 91
 92      printf("test %s\n",str);
 93
 94      count=split(str,ret,',',max);
 95
 96      for(int i = 0; i < count; i++){
 97          printf("%d:%s\n",i+1, ret[i]);
 98      }
 99
100      printf("count is %d.\n\n",count);
101      return;
102  }
```