プログラミング演習 I（第 02 回）

課題：ポインタに関する演習

■レポート提出期限：　月　日（　）　時

1．文字列の動的確保とポインタ配列

キーボードから文字列を読み込みファイルに保存するプログラムを，以下の条件に従って作成せよ．作成したプログラムが正常に動作するかを確認するため，任意の入力を用いて実行せよ．プログラム全体と実行結果（画面への出力とファイルの中身）を解答せよ．

プログラムの条件

i. キーボードから英数字（最長で MAX_LEN(100)-1 文字）を入力して文字列（文字配列）data に格納後，画面に表示する．

ii. 入力された文字列と同じ長さの文字列を格納する領域を動的に確保し，文字列 data をその領域にコピーする．なお，必要な文字配列の長さは文字列の長さ＋1バイトである点に注意する．

iii. 文字列 end が入力されるか，入力された文字列が NUM_STRING(20)個になるまで i～ii の処理を繰り返す．

iv. 各文字列へのポインタを格納する（char *）型ポインタの配列 str_p（サイズ:NUM_STRING）を定義して利用すること．

v. i～iii の文字列入力処理が終了した後，メモリに格納された全ての文字列を画面に出力する．出力は最初の行に文字列の個数を，次の行以降に入力された順番と「逆の順番で」文字列を出力することとし，以下の関数 output_strings_reverse()を作成して利用すること．

```
void output_strings_reverse(char **array_p, int m)
```

array_p: 文字列へのポインタの配列

m: 文字列の個数

vi. グローバル変数（関数の外部で宣言する変数）は用いてはいけない．

vii. エラーのチェックを行うこと．

viii. コメントを入れるなどして，見やすいプログラムを書くこと．

```
Input strings －> st22
st22
Input strings －> st333
st333
Input strings －> st1
st1
Input strings －> end
end
```

実行例 1（下線部はキーボードからの入力を指す）

```
3
st1
st333
st22
```

画面への出力

※一部の機能を実現したプログラム（pointer-kadai1.c）を利用する．

2. ポインタの応用演習

課題1のプログラムを任意の数の文字列を扱えるようにし，さらに文字列の長さの短い順番に並べ替えた後，画面に出力するように改修せよ．プログラムは以下の条件（処理）に従って作成せよ．作成したプログラムが正常に動作するかを確認するため，複数の文字列を入力して画面出力を確認せよ．少なくとも課題1の入力例は実行すること．プログラム全体と実行結果（画面への出力）を解答せよ．

プログラムの条件

i. キーボードから入力する文字列の個数を `int` 型の変数（変数名：`n`）に読み込む．

ii. 文字列へのポインタを格納する（`char *`）型ポインタの配列（配列名：`str_p`，サイズ：文字列の個数 `n`）を「動的に」確保する．

iii. キーボードから英数字（最長で`MAX_LEN`（`100`）`-1`文字）を入力して文字列（文字配列）`data`に格納後，画面に表示する．

iv. 読み込んだ文字列と同じ長さの文字列を格納する領域を動的に確保し，文字列 `data` をその領域にコピーする．なお，領域へのポインタを保存するため，`ii` で確保したポインタの配列を利用する．

v. 文字列が `n` 個入力されるまで `iii`～`iv` の処理を繰り返す．

ix. 全ての文字列の入力が終了後，文字列の個数と全ての文字列を順番に画面に出力する．出力は最初の行に文字列の個数を，次の行以降に入力された順番と同じ順番で文字列を出力することとし，以下の関数 `output_strings()`を作成して利用すること．（課題1のように逆順とする必要はないので注意すること．）

```
void output_strings(char **array_p, int m)
```

`array_p:` 文字列へのポインタの配列　　`m:` 文字列の個数

vi. 文字列を，文字の長さが短い順番に並べ替える．並べ替えは，文字列へのポインタを入れ替えることにより行うこと．

vii. 並べ替えの後，文字列の個数と全ての文字列を `output_strings()`を用いて再度順番に画面に出力する．（実行例2）

viii. 課題1の `vi`～`viii` と同じ条件を満たすこと．

```
Number of strings －> 3
3
Input strings －> st22
st22
Input strings －> st333
st333
Input strings －> st1
st1

Before sorting
3
st22
st333
st1

After sorting
3
st1
st22
st333
```

実行例２

※一部の機能を実現したプログラム（pointer-kadai2.c）を利用する．

プログラミング演習Ⅰ（第02回） 補足資料2

・必要に応じて以下の関数を利用すること.

・何れの関数も，使用する際には,「#include <string.h>」をプログラムの先頭部に追加すること.

●文字列の比較

```
int strcmp(const char *s1,  const char *s2)
```

・引数　s1:文字列へのポインタ，　s2:文字列へのポインタ

・返り値　文字コードにより文字列の大小関係を求め，s1 の指す文字列が s2 の指す文字列と等しい場合には 0，大きい場合は正の整数値，小さい場合は負の整数値を返す.

●文字列のコピー

```
char *strcpy(char *s1,  const char *s2)
```

・引数　s1:コピーする先の文字列へのポインタ，　s2:コピーする文字列へのポインタ

・返り値　コピーする先の文字列へのポインタ（s1）

●文字列の長さの取得

```
size_t strlen(const char *s)
```

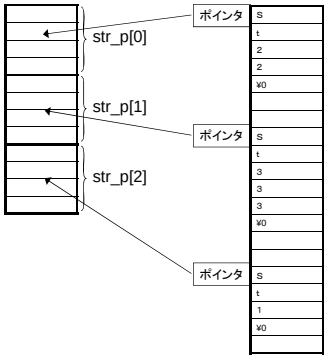
・引数　s:文字列へのポインタ

・返り値　s の指す文字列の長さ（終端を表す NULL 文字（¥0）は長さに含まない）

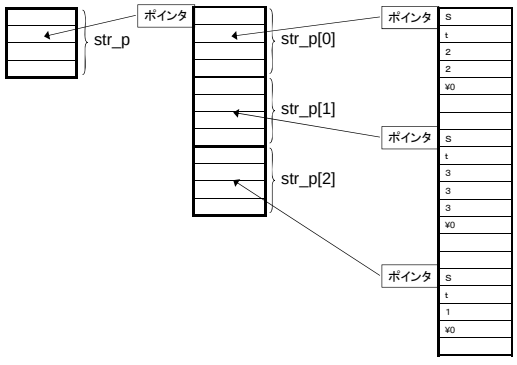
プログラミング演習Ⅰ　第02回

補足資料1

課題1のデータ構造例



課題2のデータ構造例
（文字列をファイルから読み込んだ直後）



課題2のデータ構造例
（文字列を長さの短い順番に並べ替えた後）