情報処理 2 - 前期第 11 回課題

柴田健琉

提出日：2026 年 07 月 06 日
2026 年 07 月 02 日

1　はじめに

この課題のプログラムは以下の環境での動作が確認されている:

- OS: NixOS 26.05 Yarara, Linux 7.1.2 x86_64
- CC: GCC 15.2.0
- CFLAGS: `-g -O1 -Wall -Wpedantic`

2　課題 - 回文判定

入力された英文文字列が回文になっているか判定する `kaibun` 関数を作成する.

回文判定プログラム

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

#define BUFF_SZ 128

// Obtain the lenght of string
int my_strlen(char *str, int max_len) {
    if (str == NULL) return -1;

    int i = 0;
    while (*(str + i) != '\0' && i < max_len)
        i++;

    return i;
}

// convert ASCII CAPITAL CASE ALPHABETS into lower case alphabets
void to_lower(char *str, int sz) {
    if (str == NULL) return;

    for (int i = 0; i < sz; i++) {
        if (*(str + i) >= 'A' && *(str + i) <= 'Z') {
            *(str + i) = *(str + i) + 0x20;
        }
    }
}

// remove every non-alphabet letters from string
int strip(char *str, char *dest, int sz) {
    if (str == NULL || dest == NULL) {
        return -1;
    }

    int ret_sz = 0;
```

```
35
36      for (int i = 0; i < sz; i++) {
37          if (*(str + i) >= 'a' && *(str + i) <= 'z') {
38              *(dest + ret_sz) = *(str + i);
39              ret_sz++;
40          }
41      }
42
43      *(dest + ret_sz) = '\0';
44
45      return ret_sz;
46  }
47
48  // check if string is a palindrome
49  bool kaibun(char *str, int sz) {
50      if (str == NULL) return false;
51
52      for (int i = 0; i < (sz - 1)/2; i++) {
53          if (*(str + i) != *(str + sz - i - 2)) {
54              return false;
55          }
56      }
57
58      return true;
59  }
60
61  int main(void) {
62      // BUFF_SZ letters + null terminator
63      char *buff = (char*)malloc(sizeof(char) * (BUFF_SZ + 1));
64      char *processed_str = (char*)malloc(sizeof(char) * (BUFF_SZ + 1));
65
66      printf("Input␣string:␣");
67      char* res = fgets(buff, BUFF_SZ, stdin);
68      if (res == NULL) {
69          printf("Error␣while␣reading␣from␣stdin\n");
70          return 1;
71      }
72
73      int len = my_strlen(buff, BUFF_SZ);
74      if (len == -1) {
75          printf("Error␣on␣retrival␣of␣string␣length.");
76          return 1;
77      }
78      to_lower(buff, len);
79      len = strip(buff, processed_str, len);
80
81      bool isPalindrome = kaibun(processed_str, len + 1);
82
83      printf("isPalindrome:␣%s\n", isPalindrome ? "true" : "false");
84
85      return 0;
86  }
```

2.1 実行結果

```
~/Documents/nit-info-proc-2-S1/src/build/cls11 % ./main
Input string: NaN
isPalindrome: true
~/Documents/nit-info-proc-2-S1/src/build/cls11 % ./main
Input string: Able was I ere I saw Elba.
isPalindrome: true
~/Documents/nit-info-proc-2-S1/src/build/cls11 % ./main
Input string: sudo rm -rf /
isPalindrome: false
~/Documents/nit-info-proc-2-S1/src/build/cls11 % ./main
Input string: NULL
isPalindrome: false
~/Documents/nit-info-proc-2-S1/src/build/cls11 %
```

図1: 回文判定プログラムの実行結果

2.2 処理の流れ

2.2.1 `my_strlen` 関数

1. `str` に対しヌルポインタチェックを実施し, エラーであれば異常値を返却する
2. 変数 `i` を宣言し, 0 で初期化する
3. 文字へのポインタ `str` から `i` 分ずれたアドレスを指す文字がヌル文字でない, かつ `i` が最大文字カウント `max_len` 以下であれば, `i` をインクリメントする
4. 条件を満たすまで 3 を繰り返す
5. `i` を返却する

2.2.2 `to_lower` 関数

1. `str` に対しヌルポインタチェックを実施し, エラーであればサブルーティーンを早期終了させる
2. カウンタ変数 `i` を 0 から文字列の長さ `sz` まで 3 を繰り返す
3. 文字型ポインタ `str` から `i` 分ずれたアドレスを指す文字が ASCII の'A' 以上, 'Z' 以下であるならば, 十六進数値 0x20 を足し, ASCII の'a' から'z' の範囲にある文字に変更する

2.2.3 `strip` 関数

1. `str` と `dest` に対しヌルポインタチェックを実施し, エラーであれば異常値を返却する
2. 変更後の文字列の長さを格納する整数型変数 `ret_sz` を宣言・初期化する

3. カウンタ変数 i を 0 から文字列の長さ sz まで 4 を繰り返す
4. 文字型ポインタ str から i 分ずれたアドレスを指す文字が ASCII の'a' 以上, 'z' 以下であるならば, dest 文字列に該当文字を追加し, ret_sz をインクリメントする
5. dest 文字列の終端にヌル文字を追加する
6. ret_sz を返却する

2.2.4 kaibun 関数

1. str に対しヌルポインタチェックを実施し, エラーであれば論理偽を返却する
2. カウンタ変数 i を 0 から文字列の長さ sz の半分まで 3 を繰り返す
3. 文字ポインタ str の始点と終点から中央に向って i 分ずれたアドレスが指す値が一致しなかった場合に論理偽を返却する
4. for ループを抜けたら論理真を返却する

2.2.5 main 関数

1. buff と processed_str の 2 つそれぞれに BUFF_SZ + 1 文字分のメモリを確保する
2. 標準入力から最大 BUFF_SZ 文字の 1 行を取得し, buff に格納する
3. 2 でのエラー処理を行う
4. 文字列 buff の長さを取得, エラー処理も行う
5. 文字列 buff を to_lower 関数と strip 関数で加工, processed_str に格納
6. processed_str に対し kaibun 関数を適用, 結果を isPalindrome に格納
7. 標準出力に isPalindrome の値を出力する

2.3 書く際の難点

Off by one error で kaibun 関数内のポインタ演算の処理が一発で通らず, 何回か修正・試行した.

2.4 想定されるバグ

多くの関数で文字列の長さを指定する必要があるが, この長さを間違えると意図しない動作が発生してしまう. 例えば, kaibun 関数で 1 文字多く・少なく指定してしまったため, 回文であっても正常に判定されないという現象が考えられる.

しかし, ユーザ側では長さの指定はできないので, プログラマが責任をもって記述することになる.