明治大学理工学部情報科学科

ソフトウェア実習 レポート

2章 文字列操作

提出日 2025 年 10 月 5 日

3 年 15組 91番 157R257501

SEO BEOMGYO

はじめに

本課題の目的は，標準Cライブラリに相当する文字列処理関数（検索・複製・連結・比較・部分文字列探索・文字集合探索・数値変換）を自作し，入出力例に基づいて正当性を確認することである。対象関数は strchr（2実装），strlen，strcpy，strcat，strcmp，strstr，strpbrk，atoi の各相当関数であり，いずれもヌル終端文字列を前提に逐次走査する基本戦略を採る。

課題1：strchr1/strchr2

```c
// pointer
char* strchr1(const char* s, int c) {
    while (*s) {
        if (*s == c) return (char*)s;
        s++;
    }
    return NULL;
}

// index
char* strchr2(const char* s, int c) {
    int i = 0;
    while (s[i] != '\0') {
        if (s[i] == c) return (char*)(s + i);
        i++;
    }
    return NULL;
}
```

- 目的
  与えられた文字列中で指定文字の最初の出現位置を指すポインタを返す関数を、ポインタ走査版と添字走査版の二様で実装する。
- 実装要点
  strchr1 はポインタ s を前進させながら一致検査し、一致時に現在位置を返し、終端まで見つからなければ NULL を返す構造である。
  strchr2 は添字 i により s[i] を終端記号まで走査し、一致時に s + i を返す点のみが差分であり、機能的には同等である。
- テストと実行結果
  - 入力：”abcdef”,’d’ → 出力：strchr1 -> ”def”, strchr2 -> ”def”（一致位置から終端までを表示）
  - 入力：”aaaaaa”,’a’ → 出力：strchr1 -> ”aaaaaa”, strchr2 -> ”aaaaaa”
  - 入力：”abcdef”,’z’ → 出力：strchr1 -> NULL, strchr2 -> NULL
  - 入力：”” ,’a’ → 出力：strchr1 -> NULL, strchr2 -> NULL
  - 入力：”12345”,’5’ → 出力：strchr1 -> ”5”, strchr2 -> ”5”

課題2: strcpy1

```c
char* strcpy1(char* dest, const char* src) {
    char* ptr = dest;
    while (*src) {
        *ptr = *src;
        ptr++;
        src++;
    }
    *ptr = '\0';  // last index null
    return dest;
}
```

- 目的
  ソース文字列を終端記号まで逐次コピーして、宛先に同内容の C 文字列を構成する関数を実装する。
- 実装要点
  dest 先頭からポインタを前進させつつ *src を代入し、複写終了後に '¥0' を明示的に付与しているため、空文字列のコピーでも正しい終端が保証される。
- テストと実行結果
  - 入力: dest, src="hello" → 出力: copy result: hello
  - 入力: src="abcde" → 出力: copy result: abcde
  - 入力: src="" → 出力: copy result: （空行、空文字の複写）

課題3: strlen1

```c
int strlen1(const char *s) {
    int len = 0;
    while (*s != '\0') {
        len++;
        s++;
    }
    return len;
}
```

- 目的
  終端記号 '¥0' に到達するまで長さを加算して文字列長を返す関数を実装する。
- 実装要点
  ポインタを前進させながらカウンタ len を加算し、終端で len を返す単純な線形走査である。
- テストと実行結果
  - 入力: "hello" → 出力: length: 5
  - 入力: "abcdef" → 出力: length: 6

- 入力: "" → 出力: length: 0
- 入力: "a" → 出力: length: 1

課題4: strcat1

```c
char* strcat1(char *dest, const char *src) {
    char *ptr = dest;


    while (*ptr != '\0') {
        ptr++;
    }

    while (*src != '\0') {
        *ptr = *src;
        ptr++;
        src++;
    }

    // last index == NULL
    *ptr = '\0';

    return dest;
}
```

- 目的
  宛先文字列末尾にソース文字列を連結する関数を実装する。
- 実装要点
  まず dest の終端位置までポインタを進め、以降 src を逐次書き込み、最後に '¥0' を付与している。
- テストと実行結果
  - 初期: str="abc" → 連結 src="def" 後の出力: result = abcdef

課題5：strcmpl

```c
int strcmp1(const char *str1, const char *str2) {
    while (*str1 && (*str1 == *str2)) {
        str1++;
        str2++;
    }
    return (unsigned char)*str1 - (unsigned char)*str2;
}
```

- 目的
  二つの文字列を辞書順に比較し、差の符号で大小関係を返す関数を実装する。
- 実装要点
  一致が続く間は双方を前進させ、最初に異なる文字の unsigned char 差を返す実装であり、終端同士であれば 0 を返す。
- テストと実行結果
  - 入力："abc","abc" → 出力：0（等しい）
  - 入力："abc","abcd" → 出力：負値（前者が短い）
  - 入力："bac","abc" → 出力：正値（先頭比較で前者が大）
  - 入力："","" → 出力：0

課題6：strstrl

```c
char* strstr1(const char* s1, const char* s2) {
    if (*s2 == '\0') return (char*)s1;

    for (; *s1 != '\0'; s1++) {
        const char *p1 = s1;
        const char *p2 = s2;
        while (*p1 && *p2 && (*p1 == *p2)) {
            p1++;
            p2++;
        }
        if (*p2 == '\0') {
            return (char*)s1;
        }
    }
    return NULL;
}
```

- 目的
  部分文字列の最初の出現位置を返す関数を実装する。
- 実装要点
  外側で開始位置を一つずつずらし、内側で連続一致を確認し、探索語が空なら先頭

を返す、という二重ループ構成である。

- テストと実行結果
    - 入力: ”abcddddef”,”dde” → 出力: ”ddef”（一致開始以降を表示）
    - 入力: ”abcabc”,”abc” → 出力: ”abcabc”
    - 入力: ”print”,”int” → 出力: ”int”
    - 入力: ”abc”,”” → 出力: ”abc”
    - 入力: ””,”” → 出力: ””（空文字）
    - 入力: ”age:20, name:tom”,”name:” → 出力: ”name:tom”
    - 入力: ”This is a pen”,”never” → 出力: NULL

課題7: atoil

```c
int atoi1(const char* str) {
    int n = 0, sign = 1;

    while (*str == ' ' || *str == '\t') str++;

    if (*str == '+') str++;
    else if (*str == '-') {
        sign = -1;
        str++;
    }

    while (*str >= '0' && *str <= '9') {
        n = n * 10 + (*str - '0');
        str++;
    }

    return sign * n;
}
```

- 目的
  数値文字列を int に変換する簡易関数を実装する。
- 実装要点
  前置空白のスキップ、+/- の符号処理、連続する数字のみに基づく累積変換を行い、非数字に遭遇した時点で停止して符号付き値を返す。
- テストと実行結果
    - 入力: ”123” → 出力: 123
    - 入力: ” -45” → 出力: -45
    - 入力: ”+77abc” → 出力: 77（英字以降は無視）

- 入力：”0” → 出力：0
- 入力：”abc” → 出力：0（数字先頭でないため積は 0）

課題8：strpbrkl

```
char* strpbrk1(const char* s, const char* accept) {
    for (; *s != '\0'; s++) {
        for (const char *a = accept; *a != '\0'; a++) {
            if (*s == *a) return (char*)s;
        }
    }
    return NULL;
}
```

- 目的
  受容集合 accept に含まれるいずれかの文字が最初に現れる位置を返す関数を実装する。
- 実装要点
  外側で対象文字列を一文字ずつ進め、内側で accept 全体を走査して一致すればその位置を返す二重ループである。
- テストと実行結果
  - 入力：”abcdef”,”xyza” → 出力：”abcdef”（先頭 a に一致）
  - 入力：”hello world”,”oW” → 出力：”o world”（小文字 o に一致）
  - 入力：”test”,”xyz” → 出力：NULL