Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра вычислительных систем

ОТЧЕТ
по практической работе №5

по дисциплине «**Программирование**»

Выполнил: Гердележов Д.Д.
студент гр. ИВ-122
«17» мая 2022 г.

Проверил: Фульман В.О.
Старший преподаватель
кафедры ВС.
«18» мая 2022 г.

Оценка «_____»

Новосибирск 2022

Оглавление

Задание:

При разработке подпрограмм в соответствии с заданием следует учитывать, что:

1. Исходная задача должна быть разбита на четыре основные подзадачи:
    0. ввод данных – функция input() – предусматривает взаимодействие с пользователем и возвращает строку, содержащую входные данные (путь);
    1. проверка корректности данных – check() – которая обеспечивает проверку допустимости длины входной строки и используемых в ней символов. Значения, возвращаемые функцией check(), должны позволять определить тип ошибки и (если возможно) номер символа, в которой она обнаружена;
    2. обработка – process() – входных данных согласно заданию;
    3. вывод данных – output() – на экран, обеспечивает отображение полученных результатов или сообщений об ошибке.

Взаимодействие между указанными функциями осуществляется через данные. Каждая из них, при необходимости, также разбивается на подзадачи. Разбиение производится до тех пор, пока подзадачи не становятся тривиальными, например, вычисление длины строки.

2. Каждая подзадача оформляется в виде функции.
3.
4. Не допускается использования стандартных функций обработки строк. Все операции над строками должны быть реализованы самостоятельно в виде отдельных подпрограмм. Эти подпрограммы необходимо разместить в отдельном файле strings.c, а прототипы функций – в файле strings.h.

Примерами типичных функций, которые должны присутствовать в каждой программе являются:

0. функция вычисления длины строки (slen);
1. функция разбиения строки на элементы-токены (stok), разделенные заданным символом (например, символ “/” при анализе пути или символ “.”, при разборе IP-адресов или доменных имен);
2. функция проверки символа на принадлежность заданному множеству символов (sspn), необходимая для проверки допустимости используемых символов.
3. функция сравнения строк (scmp);
4. функция копирования строк (scpy).

Набор тестов должен обеспечивать проверку поведения программы для всех классов входных данных:

1. некорректный файловый путь;
2. превышение допустимой длины пути;
3. допустимый путь, который не удовлетворяет указанным в задании условиям;
4. допустимый путь, удовлетворяющий условиям.

Вариант 5

Преобразовать все Windows пути во входном списке к Linux-путям, используя формат эмулятора Cygwin.

Вход:

```
delim: +
paths: C:\Windows\system32+E:\Distrib\msoffice.exe+/home/alex/prog/lab4.c
```

Выход:

```
new paths:
/cygdrive/c/Windows/system32+/cygdrive/e/Distrib/msoffice.exe+/home/alex/prog/lab4.c
```

Выполнение работы:

```
159  int main()
160  {
161      system("clear");
162
163      char* path = malloc(sizeof(char) * MAX_PATH);
164      char* delim = malloc(sizeof(char) * 1);
165
166      input(path, delim);
167
168      int t = check(path, delim);
169      if (!t) {
170          printf("\nПолученные пути:\n%s\n", process(path, delim));
171      } else {
172          output(t);
173      }
174
175      return 0;
176  }
177
```

Вожу с клавиатуры строку и разделитель (path и delim):

```
 8  void input(char* path, char* delim)
 9  {
10      printf("delim: ");
11      scanf("%s", delim);
12      printf("path: ");
13      scanf("%s", path);
14      printf("\n");
15  }
```

Вызываю функцию проверки входных данных (check), в которой существуют проверки на допустимые символы, допустимую длину пути.

Если ошибок не найдено, то запускаем основную функцию – process и выводи её результат.

```
 99  {
100      char* buf_path = malloc(sizeof(char) * slen(path) + 2);
101      scpy(buf_path, path);
102      buf_path[slen(path)] = *delim;
103      buf_path[slen(path) + 1] = '\0';
104      char* res = malloc(slen(path)); //итоговая строка
105      char* tmp;
106      for (tmp = stok(buf_path, delim); tmp;
107           tmp = stok(NULL, delim)) { //делим на подстроки весь путь
108          if (slen(res)) {
109              scat(res, delim);
110          }
111          char* buf = malloc(sizeof(char) * slen(tmp) + 2); //хранит подстроку
112          scpy(buf, tmp);
113          buf[slen(buf)] = '\\';
114          buf[slen(buf) + 1] = 0;
115          char* ttmp;
116          int a = 0, b = 0;
```

```
117         if ((ttmp = sstr(buf, ":\\"))) {
118             a = 1;
119         } else if ((ttmp = sstr(buf, "\\"))) {
120             b = 1;
121         }
122         if (a || b) {
123             int c = 1; //для прав работы сток
124             char* tr = malloc(
125                     sizeof(char) * (slen(buf) + 11)); //итоговая подстрока
126             if (a) {
127                 scat(tr, "/cygdrive/");
128                 int n = ttmp - buf;
129                 c = 3;
130                 if (n == 1) { //если одна буква в пути
131                     tr[10] = buf[0] + 32;
132                 } else if (n == 2) { //если две буквы в пути
133                     tr[10] = buf[0] + 32;
134                     tr[11] = buf[1] + 32;
135                 }
136             } else if (b) {
137                 scat(tr, "/cygdrive");
138             }
139             char* tmp_t;
140             char* tmp_lt = save_static_char();
141             for (tmp_t = stok(buf + c, "\\"); tmp_t;
142                  tmp_t = stok(NULL, "\\")) { //добавление в итоговую подстроку
143                                              //путей исходного пути
144                 scat(tr, "/");
145                 scat(tr, tmp_t);
146             }
147             new_static_char(tmp_lt);
148             res = realloc(res, slen(res) + slen(tr));
149             scat(res, tr);
150         } else {
151             res = realloc(res, slen(res) + slen(tmp));
152             scat(res, tmp);
153         }
154     }
155     res[slen(res)] = 0; //конец итоговой строки
156     return res;
157 }
```

В данной функции происходит следующее:

Копируется исходная строка в буфер, куда дописывается разделитель и символ конца строки.

res – итоговая строка, в цикле берется путь из строки, если это windows путь, то преобразуется путь используя переменную tr, после чего записывается в res, если путь Linux, то просто добавляем его в res.

Примеры работы:

1)

```
delim: +
path: C:\Windows\system32+E:\Distrib\msoffice.exe+/home/alex/prog/lab4.c


Полученные пути:
/cygdrive/c/Windows/system32+/cygdrive/e/Distrib/msoffice.exe+/cygdrive/home/alex/prog/lab4.c
```

2)

```
delim: +
path: C:\Windows\qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq.txt

Length too long.
```

3)

```
delim: +
path: C:\Wndows\?abc.txt+C:\qwerty.txt

Problem in the way Windows.
```

Приложение:

Main.c

```c
#include "strings/strings.h"
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>

#define MAX_PATH 260

void input(char* path, char* delim)
{
    printf("delim: ");
    scanf("%s", delim);
    printf("path: ");
    scanf("%s", path);
    printf("\n");
}

int check(char* path, char* delim)
{
    char* buf_path = malloc(sizeof(char) * slen(path) + 2);
    scat(buf_path, path);
    buf_path[slen(buf_path)] = *delim;
    buf_path[slen(buf_path)] = '\0';

    //printf("Ведённая строка разбита на:\n");
    char* str_tmp;
    for (str_tmp = stok(buf_path, delim); str_tmp;
         str_tmp = stok(NULL, delim)) {
        //printf("%s\n", str_tmp);

        if (slen(path) >= MAX_PATH) { // len
            return 3;
        }

        char* th_wi_let = sstr(str_tmp, ":\\"); // windows
        char* th_wi_nlet = sstr(str_tmp, "\\");

        /*if (th_wi_let == 0) {
            return 1;
        }*/
```

```
40
41          if (th_wi_nlet) {
42              if (!isalpha(str_tmp[0]) && str_tmp[0] != '\\') {
43                  return 1;
44              }
45              if (sspn(str_tmp, "*?<\">,/|")) {
46                  return 1;
47              }
48          }
49
50          if ((th_wi_let - str_tmp == 1) || (th_wi_let - str_tmp == 2)) {
51              if (isalpha(str_tmp[0]) && str_tmp[1] == ':') {
52                  if (sspn(str_tmp, "*?<\">,/|")) {
53                      return 1;
54                  }
55              }
56              if (sspn(str_tmp, "*?<\">,/|")) {
57                  return 1;
58              }
59              if (th_wi_let - str_tmp == 1) {
60                  if (str_tmp[0] > 90) {
61                      return 1;
62                  }
63              } else if (th_wi_let - str_tmp == 2) {
64                  if (str_tmp[0] > 90
65                      || str_tmp[1] > 90) { //проверка на заглавные буквы
66                      return 1;
67                  }
68              }
69          } else if (((th_wi_let = sstr(str_tmp, "/")) - str_tmp) == 0) {
70              if (sspn(str_tmp, "*?:\\<\">,|")) {
71                  return 2;
72              }
73          }
74      }
75      return 0;
76  }
77
78  // 0 - good, 1 - trable windows, 2 - trable linux, 3 - problam with len of
79  // path
80  void output(int n)
81  {
82      switch (n) {
83      case 0:
84          //
85          break;
86      case 1:
87          printf("Problem in the way Windows.\n");
88          break;
89      case 2:
90          printf("Problem in the way Linux.\n");
91          break;
92      case 3:
93          printf("Length too long.\n");
94          break;
95      }
96  }
97
```

```
 98 char* process(char* path, char* delim)
 99 {
100     char* buf_path = malloc(sizeof(char) * slen(path) + 2);
101     scpy(buf_path, path);
102     buf_path[slen(path)] = *delim;
103     buf_path[slen(path) + 1] = '\0';
104     char* res = malloc(slen(path)); //итоговая строка
105     char* tmp;
106     for (tmp = stok(buf_path, delim); tmp;
107          tmp = stok(NULL, delim)) { //делим на подстроки весь путь
108         if (slen(res)) {
109             scat(res, delim);
110         }
111         char* buf = malloc(sizeof(char) * slen(tmp) + 2); //хранит подстроку
112         scpy(buf, tmp);
113         buf[slen(buf)] = '\\';
114         buf[slen(buf) + 1] = 0;
115         char* ttmp;
116         int a = 0, b = 0;
117         if ((ttmp = sstr(buf, ":\\"))) {
118             a = 1;
119         } else if ((ttmp = sstr(buf, "\\"))) {
120             b = 1;
121         }
122         if (a || b) {
123             int c = 1; //для прав работы сток
124             char* tr = malloc(
125                     sizeof(char) * (slen(buf) + 11)); //итоговая подстрока
126             if (a) {
127                 scat(tr, "/cygdrive/");
128                 int n = ttmp - buf;
129                 c = 3;
130                 if (n == 1) { //если одна буква в пути
131                     tr[10] = buf[0] + 32;
132                 } else if (n == 2) { //если две буквы в пути
133                     tr[10] = buf[0] + 32;
134                     tr[11] = buf[1] + 32;
135                 }
136             } else if (b) {
137                 scat(tr, "/cygdrive");
138             }
139             char* tmp_t;
140             char* tmp_lt = save_static_char();
141             for (tmp_t = stok(buf + c, "\\"); tmp_t;
142                  tmp_t = stok(NULL, "\\")) { //добавление в итоговую подстроку
143                                             //путей исходного пути
144                 scat(tr, "/");
145                 scat(tr, tmp_t);
146             }
147             new_static_char(tmp_lt);
148             res = realloc(res, slen(res) + slen(tr));
149             scat(res, tr);
150         } else {
151             res = realloc(res, slen(res) + slen(tmp));
152             scat(res, tmp);
153         }
154     }
155     res[slen(res)] = 0; //конец итоговой строки
```

```
156     return res;
157 }
158
159 int main()
160 {
161     system("clear");
162
163     char* path = malloc(sizeof(char) * MAX_PATH);
164     char* delim = malloc(sizeof(char) * 1);
165
166     input(path, delim);
167
168     int t = check(path, delim);
169     if (!t) {
170         printf("\nПолученные пути:\n%s\n", process(path, delim));
171     } else {
172         output(t);
173     }
174
175     return 0;
176 }
```

String.c

```
 1 #include "strings.h"
 2 #include <stdint.h>
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5 #include <string.h>
 6
 7 int slen(char* str)
 8 {
 9     int count = 0;
10     for (int i = 0; str[i] != '\0'; i++) {
11         count++;
12     }
13     return count++;
14 }
15
16 static char* lt;
17
18 char* stok(char* str, const char* delim)
19 {
20     if (str != NULL) {
21         lt = str;
22     }
23     for (int i = 0; lt[i] != 0; i++) {
24         for (int j = 0; delim[j] != 0; j++) {
25             if (lt[i] == delim[j]) {
26                 lt[i] = 0;
27                 str = lt;
28                 lt = lt + i + 1;
29                 return str;
30             }
31         }
32     }
33     return str;
```

```
34 }
35
36 int sspn(char* str, const char* substr)
37 {
38     int count = 0;
39     for (int i = 0; str[i] != 0; i++) {
40         for (int j = 0; substr[j]; j++) {
41             if (str[i] == substr[j]) {
42                 count++;
43             }
44         }
45     }
46     return count;
47 }
48
49 int scmp(const char* str, const char* strc)
50 {
51     int ncount = 0, pcount = 0;
52     for (int i = 0; str[i] != 0; i++) {
53         if (str[i] < strc[i]) {
54             ncount++;
55         } else if (str[i] > strc[i]) {
56             pcount++;
57         }
58     }
59     if (ncount < pcount) {
60         return pcount;
61     } else if (ncount > pcount) {
62         return -ncount;
63     }
64     return 0;
65 }
66
67 char* scpy(char* des, const char* src)
68 {
69     int i;
70     for (i = 0; src[i] != 0; i++) {
71         des[i] = src[i];
72     }
73     des[i] = 0;
74     return des;
75 }
76
77 void new_static_char(char* str)
78 {
79     lt = str;
80 }
81
82 char* save_static_char(void)
83 {
84     return lt;
85 }
86
87
88 char* scat(char* des, char* src)
89 {
90     int j = slen(des);
91     for (int i = 0; src[i] != 0; i++) {
```

```
 92         des[j] = src[i];
 93         j++;
 94     }
 95     des[j] = 0;
 96     return des;
 97 }
 98
 99 char* sstr(char* string1, const char* string2)
100 {
101     char* strptr = string1;
102     int j = 0;
103     for (int i = 0; string1[i] != 0; i++) {
104         if (string2[j] == 0) {
105             return strptr;
106         }
107         if (string1[i] != string2[j]) {
108             j = 0;
109             continue;
110         }
111         if (string1[i] == string2[j]) {
112             if (j == 0) {
113                 strptr = string1 + i;
114             }
115             j++;
116         }
117     }
118     if (string2[j] == 0) {
119         return strptr;
120     } else {
121         return NULL;
122     }
123 }
```

String.h

```
 1 #pragma once
 2
 3 int slen(char* str);
 4 char* stok(char* str, const char* delim);
 5 int sspn(char* str, const char* substr);
 6 int scmp(const char* mstr, const char* cpstr);
 7 char* scpy(char* des, const char* src);
 8 char* scat(char* des, char* src);
 9 char* sstr(char* str1, const char* str2);
10 void new_static_char(char* str);
11 char* save_static_char(void);
```