МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2

по дисциплине: Алгоритмы и структуры данных
тема: Производные структуры данных.
Структура данных типа «строка» (Pascal/C)»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили: асс. Солонченко Роман
Евгеньевич

Белгород 2023 г.

Лабораторная работа №2
Производные структуры данных.
Структура данных типа «строка» (Pascal/C)
Вариант 10

Цель работы: изучение встроенной структуры данных типа «строка», разработка и использование производных структур данных строкового типа.

1. Для СД типа строка определить:

   1.1. Абстрактный уровень представления СД:

      1.1.1. Характер организованности и изменчивости.
      Характер организованности - **линейный**. Характер изменчивости - **динамический**.

      1.1.2. Набор допустимых операций.
      Инициализация, присвоение, получение символа по индексу, получение длины (в СИ доступно не для всех типов массивов).

   1.2. Физический уровень представления СД:

      1.2.1. Схему хранения.
      Схема хранения - **последовательная.**

      1.2.2. Объем памяти, занимаемый экземпляром СД.
      Будем рассматривать СД строку, которая содержит ASCII символы. В этом случае каждый символ будет иметь размер в 1 байт и иметь тип char. Индикатором окончания строки является ноль-символ, который обязательно должен находиться в строке, поэтому размер СД строки будет равен $(length + 1)$ байт, где $length$ - количество символов в строке.

      1.2.3. Формат внутреннего представления СД и способ его интерпретации.
      СД строка состоит из последовательности 8-битных чисел. 8-битному числу соответствует символ в соответствии с таблицей. Содержание таблицы зависит от выбранной кодировки. В данном отчёте была использована кодировка ASCII.

      1.2.4. Характеристику допустимых значений.
      $Car(C) = 1 + Car(char) + Car(char)^2 + ... + Car(char)^{length}$, где C - строка.

      1.2.5. Тип доступа к элементам.
      Тип доступа к элементам - **прямой**.

   1.3. Логический уровень представления СД.

      1.3.1. Способ описания СД и экземпляра СД на языке программирования.

```c
char string1[] = "Hello";
char string2[] = {'H', 'e', 'l', 'l', 'o', '\0'};
char *string3 = "world";
char string4[14];
```

2. Реализовать СД строкового типа в соответствии с вариантом индивидуального задания (см. табл.8) в виде модуля. Определить и обработать исключительные ситуации.
main.c (тесты)

```
#include "../../libs/alg/algc.h"

#include <stdio.h>
#include <assert.h>
#include <string.h>

void testWriteFromToStr() {
    string1 emptyString;
    WriteToStr(emptyString, "");
    assert(!strcmp(emptyString, ""));
    assert(StrError == OK);
    char emptyStringChar[256];
    WriteFromStr(emptyStringChar, emptyString);
    assert(StrError == OK);
    assert(!strcmp(emptyStringChar, ""));

    string1 nonEmptyString;
    WriteToStr(nonEmptyString, "Cool val");
    assert(!strcmp(nonEmptyString, "Cool val"));
    assert(StrError == OK);
    char nonEmptyStringChar[256];
    WriteFromStr(nonEmptyStringChar, nonEmptyString);
    assert(StrError == OK);
    assert(!strcmp(nonEmptyStringChar, "Cool val"));

    string1 tooBigString;
    WriteToStr(tooBigString,
↪   "..........................................................................................
    assert(StrError == BUFFER_OVERFLOW);
}

void testComp() {
    string1 t1s1 = "";
    string1 t1s2 = "";
    assert(Comp(t1s1, t1s2) == 0);
    assert(StrError == OK);

    string1 t2s1 = "A";
    string1 t2s2 = "";
```

```
    assert(Comp(t2s1, t2s2) == 1);
    assert(StrError == OK);

    string1 t3s1 = "";
    string1 t3s2 = "A";
    assert(Comp(t3s1, t3s2) == -1);
    assert(StrError == OK);

    string1 t4s1 = "AAAdasd";
    string1 t4s2 = "AAB";
    assert(Comp(t4s1, t4s2) == -1);
    assert(StrError == OK);
}

void testDelete() {
    string1 str1 = "";
    Delete(str1, 0, 0);
    assert(StrError == OK);
    assert(!strcmp(str1, ""));

    string1 str5 = "";
    Delete(str5, 0, 1);
    assert(StrError == OUT_OF_BOUNDS);

    string1 str2 = "A";
    Delete(str2, 0, 1);
    assert(StrError == OK);
    assert(!strcmp(str1, ""));

    string1 str3 = "A";
    Delete(str3, 0, 2);
    assert(StrError == OUT_OF_BOUNDS);

    string1 str4 = "I do not love algorithms and data structures";
    Delete(str4, 1, 7);
    assert(StrError == OK);
    assert(!strcmp(str4, "I love algorithms and data structures"));

    string1 str6 = "I am about to be deleted";
    Delete(str6, 0, 24);
    assert(StrError == OK);
    assert(!strcmp(str6, ""));
}
```

```
void testInsert() {
    string1 t1sb = "";
    string1 t1s = "";
    Insert(t1sb, t1s, 0);
    assert(StrError == OK);
    assert(!strcmp(t1s, ""));

    string1 t2sb = "love ";
    string1 t2s = "I algorithms and data structures";
    Insert(t2sb, t2s, 2);
    assert(StrError == OK);
    assert(!strcmp(t2s, "I love algorithms and data structures"));

    string1 t3sb = "I ";
    string1 t3s = "love algorithms and data structures";
    Insert(t3sb, t3s, 0);
    assert(StrError == OK);
    assert(!strcmp(t3s, "I love algorithms and data structures"));

    string1 t4sb = "pizza";
    string1 t4s = "I love ";
    Insert(t4sb, t4s, 7);
    assert(StrError == OK);
    assert(!strcmp(t4s, "I love pizza"));

    string1 t5sb =
↪   "1izzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzap
    string1 t5s =
↪   "2izzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzap
    Insert(t5sb, t5s, 1);
    assert(StrError == OK);
    assert(!strcmp(t5s,
↪   "21izzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzap

    string1 t6sb =
↪   "1231231izzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzap
    string1 t6s =
↪   "2izzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzap
    Insert(t6sb, t6s, 1);
    assert(StrError == BUFFER_OVERFLOW);
}
```

```
void testConcat() {
    string1 t1s1 = "";
    string1 t1s2 = "";
    string1 res1;
    Concat(t1s1, t1s2, res1);
    assert(StrError == OK);
    assert(!strcmp(res1, ""));

    string1 t2s1 = "42";
    string1 t2s2 = "";
    string1 res2;
    Concat(t2s1, t2s2, res2);
    assert(StrError == OK);
    assert(!strcmp(res2, "42"));

    string1 t3s1 = "";
    string1 t3s2 = "42";
    string1 res3;
    Concat(t3s1, t3s2, res3);
    assert(StrError == OK);
    assert(!strcmp(res3, "42"));

    string1 t4s1 = "I love algorithms and data structures";
    string1 t4s2 = " and pizza";
    string1 res4;
    Concat(t4s1, t4s2, res4);
    assert(StrError == OK);
    assert(!strcmp(res4, "I love algorithms and data structures and pizza"));

    string1 t5s1 =
↪   "1231231izzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzap
    string1 t5s2 =
↪   "2izzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapizzapi
    string1 res5;
    Concat(t5s1, t5s2, res5);
    assert(StrError == BUFFER_OVERFLOW);
}

void testCopy() {
    string1 origin1 = "";
    string1 substr1 = "Lorem ipsum";
    Copy(origin1, 0, 0, substr1);
    assert(StrError == OK);
```

```
    assert(!strcmp(substr1, ""));

    string1 origin2 = "A";
    string1 substr2 = "Lorem ipsum";
    Copy(origin2, 0, 0, substr2);
    assert(StrError == OK);
    assert(!strcmp(substr2, ""));

    string1 origin3 = "A";
    string1 substr3 = "Lorem ipsum";
    Copy(origin3, 0, 1, substr3);
    assert(StrError == OK);
    assert(!strcmp(substr3, "A"));

    string1 origin4 = "A";
    string1 substr4 = "Lorem ipsum";
    Copy(origin4, 0, 2, substr4);
    assert(StrError == OUT_OF_BOUNDS);

    string1 origin5 = "There is an answer t42o all questions";
    string1 substr5 = "Lorem ipsum";
    Copy(origin5, 20, 2, substr5);
    assert(StrError == OK);
    assert(!strcmp(substr5, "42"));

    string1 origin6 = "There is no answer to all questions";
    string1 substr6 = "Lorem ipsum";
    Copy(origin6, 33, 2, substr6);
    assert(StrError == OK);
    assert(!strcmp(substr6, "ns"));

    string1 origin7 = "There is no answer to all questions";
    string1 substr7 = "Lorem ipsum";
    Copy(origin7, 33, 3, substr7);
    assert(StrError == OUT_OF_BOUNDS);

    string1 origin8 = "There is no answer to all questions";
    string1 substr8 = "Lorem ipsum";
    Copy(origin8, 35, 3, substr8);
    assert(StrError == OUT_OF_BOUNDS);

    string1 origin9 =
↪   "1PizzaPizzaPizzaPizzaPizzaPizzaPizzaPizzaPizzaPizza1PizzaPizzaPizzaPizzaPizzaPizza
```

```
    string1 substr9 = "Lorem ipsum";
    Copy(origin9, 0, 255, substr9);
    assert(StrError == OK);
    assert(!strcmp(substr9,
↪   "1PizzaPizzaPizzaPizzaPizzaPizzaPizzaPizzaPizzaPizza1PizzaPizzaPizzaPizzaPizzaPizza

    string1 origin10 =
↪   "1111PizzaPizzaPizzaPizzaPizzaPizzaPizzaPizzaPizzaPizza1PizzaPizzaPizzaPizzaPizzaPi
    string1 substr10 = "Lorem ipsum";
    Copy(origin10, 0, 256, substr10);
    assert(StrError == BUFFER_OVERFLOW);
}

void testPos() {
    string1 t1origin = "";
    string1 t1search = "";
    assert(Pos(t1search, t1origin) == 0);
    assert(StrError == OK);

    string1 t2origin = "Some text";
    string1 t2search = "";
    assert(Pos(t2search, t2origin) == 0);
    assert(StrError == OK);

    string1 t3origin = "Some text";
    string1 t3search = "S";
    assert(Pos(t3search, t3origin) == 0);
    assert(StrError == OK);

    string1 t4origin = "Some text";
    string1 t4search = " ";
    assert(Pos(t4search, t4origin) == 4);
    assert(StrError == OK);

    string1 t5origin = "Some textp";
    string1 t5search = "p";
    assert(Pos(t5search, t5origin) == 9);
    assert(StrError == OK);

    string1 t6origin = "Some text";
    string1 t6search = "[";
    assert(Pos(t6search, t6origin) == -1);
    assert(StrError == OK);
```

```
    string1 t7origin = "I love algorithms, data structures, pizza and 42.";
    string1 t7search = "I lo";
    assert(Pos(t7search, t7origin) == 0);
    assert(StrError == OK);

    string1 t8origin = "I love algorithms, data structures, pizza and 42.";
    string1 t8search = "I love algorithms, data structures, pizza and 42.";
    assert(Pos(t8search, t8origin) == 0);
    assert(StrError == OK);

    string1 t9origin = "I love algorithms, data structures, pizza and 42.";
    string1 t9search = "data";
    assert(Pos(t9search, t9origin) == 19);
    assert(StrError == OK);

    string1 t10origin = "I love algorithms, data structures, pizza and 42.";
    string1 t10search = "42.";
    assert(Pos(t10search, t10origin) == 46);
    assert(StrError == OK);

    // Requiem for pizza
    string1 t11origin = "I love algorithms, data structures, pizza and 42.";
    string1 t11search = "pizza";
    assert(Pos(t11search, t11origin) == 36);
    assert(StrError == OK);
}

void test() {
    testWriteFromToStr();
    testComp();
    testDelete();
    testInsert();
    testConcat();
    testCopy();
    testPos();
}

int main() {
    test();

    return 0;
}
```

alg.h (заголовки)

```
extern const int OK;
extern const int BUFFER_OVERFLOW;
extern const int INVALID_FORMAT;
extern const int OUT_OF_BOUNDS;
extern int StrError; // Переменная ошибок

typedef char string1[256];

// Признак конца строки - символ '\0'
void WriteToStr(string1 st, char *s);
void WriteFromStr(char *s, string1 st);
void InputStr(string1 s);
void OutputStr(string1 s);
int Comp(string1 s1, string1 s2);
void Delete(string1 s, unsigned Index, unsigned Count);
void Insert(string1 Subs, string1 s, unsigned Index);
unsigned Length(string1 s);
void Concat(string1 s1, string1 s2, string1 srez);
void Copy(string1 s, unsigned Index, unsigned Count, string1 Subs);
unsigned Pos(string1 SubS, string1 s);
unsigned StrCSpn(string1 s, string1 s1);
```

task2.c (реализации функций)

```
#include "../algc.h"

#include <stdio.h>
#include <stdbool.h>

const int OK = 0;
const int BUFFER_OVERFLOW = 1;
const int INVALID_FORMAT = 2;
const int OUT_OF_BOUNDS = 3;

int StrError = OK;

unsigned Length(string1 s)
{
    for (unsigned i = 0; i < sizeof(string1); i++)
    {
```

```
        if (s[i] == '\0')
        {
            StrError = OK;
            return i;
        }
    }

    StrError = INVALID_FORMAT;
}

// Признак конца строки - символ '\0'
void WriteToStr(string1 st, char *s)
{
    for (int i = 0; i < sizeof(string1); i++)
    {
        st[i] = s[i];

        if (s[i] == '\0')
        {
            StrError = OK;
            return;
        }
    }

    // В случае, если s окажется слишком большим, будем укорачивать строку
    // до размера буфера.
    StrError = BUFFER_OVERFLOW;
    st[sizeof(string1) - 1] = '\0';
}

void WriteFromStr(char *s, string1 st)
{
    for (int i = 0; i < sizeof(string1); i++)
    {
        s[i] = st[i];

        if (st[i] == '\0')
        {
            StrError = OK;
            return;
        }
    }
```

```
    // В исходной строке st не было ноль-символа, что говорит о
    // некорректном формате строки. В строку s, куда пишем информацию,
    // запишем ноль-символ для предоствращения дальнейших ошибок.
    StrError = INVALID_FORMAT;
    s[sizeof(string1) - 1] = '\0';
}

void InputStr(string1 s)
{
    for (int i = 0; i < sizeof(string1); i++)
    {
        int input = getchar();

        if (input == '\n')
        {
            s[i] = '\0';
            StrError = OK;
            return;
        }
        else
            s[i] = input;
    }

    s[sizeof(string1) - 1] = '\0';
    StrError = BUFFER_OVERFLOW;
}

void OutputStr(string1 s)
{
    for (int i = 0; i < sizeof(string1); i++)
    {
        if (s[i] == '\0')
        {
            StrError = OK;
            return;
        }

        putc(s[i], stdout);
    }

    // Строка не содержит конца, поэтому присваиваем ошибку
    StrError = INVALID_FORMAT;
}
```

```
int Comp(string1 s1, string1 s2)
{
    for (int i = 0; i < sizeof(string1); i++)
    {
        if (s1[i] != s2[i] || (s1[i] == '\0' || s2[i] == '\0'))
        {
            StrError = OK;
            int diff = s1[i] - s2[i];
            return diff > 0 ? 1 : diff < 0 ? -1
                                           : 0;
        }
    }

    // Строки не содержат конца
    StrError = INVALID_FORMAT;
    return 0;
}

void Delete(string1 s, unsigned Index, unsigned Count)
{
    if (Count == 0)
    {
        StrError = OK;
        return;
    }

    int strlen = Length(s);
    if (StrError != OK)
        return;

    if (Index + Count - 1 >= strlen)
    {
        StrError = OUT_OF_BOUNDS;
        return;
    }

    // +1 для нуль-символа
    for (int i = Index; i < Index + strlen - Count + 1; i++)
    {
        s[i] = s[i + Count];
    }
```

```
    StrError = OK;
}

void Insert(string1 Subs, string1 s, unsigned Index)
{
    int substrlen = Length(Subs);
    if (StrError != OK)
        return;

    int strlen = Length(s);
    if (StrError != OK)
        return;

    if (Index > strlen)
    {
        StrError = OUT_OF_BOUNDS;
        return;
    }

    if (strlen + substrlen + 1 > sizeof(string1))
    {
        StrError = BUFFER_OVERFLOW;
        return;
    }

    for (int i = strlen + substrlen; i >= (int)Index + substrlen; i--)
    {
        s[i] = s[i - substrlen];
    }

    for (int i = 0; i < substrlen; i++)
    {
        s[Index + i] = Subs[i];
    }
}

void Concat(string1 s1, string1 s2, string1 srez)
{
    unsigned s1len = Length(s1);
    if (StrError != OK)
        return;

    int i;
```

```
    for (i = 0; i < s1len; i++)
    {
        srez[i] = s1[i];
    }

    for (int j = 0; i < sizeof(string1); i++, j++)
    {
        srez[i] = s2[j];
        if (srez[i] == '\0')
        {
            StrError = OK;
            return;
        }
    }

    srez[i] = '\0';
    StrError = BUFFER_OVERFLOW;
}

void Copy(string1 s, unsigned Index, unsigned Count, string1 Subs)
{
    if (Count >= sizeof(string1))
    {
        StrError = BUFFER_OVERFLOW;
        return;
    }

    int strlen = Length(s);
    if (StrError != OK)
        return;

    if (Index + Count > strlen && Count != 0)
    {
        StrError = OUT_OF_BOUNDS;
        return;
    }

    for (int i = 0; i < Count; i++)
    {
        Subs[i] = s[i + Index];
    }

    Subs[Count] = '\0';
```

```
}

unsigned Pos(string1 SubS, string1 s)
{
    int strlen = Length(s);
    if (StrError != OK)
        return -1;

    if (SubS[0] == '\0')
    {
        StrError = OK;
        return 0;
    }

    for (int i = 0; i < strlen; i++)
    {
        for (int j = 0; j < sizeof(string1); j++)
        {
            if (SubS[j] == '\0')
            {
                StrError = OK;
                return i;
            }
            else if (s[j + i] == '\0')
            {
                // Дальнейший поиск не имеет смысла, рассматриваемая подстрока больше
                // строки, в которой выполняется поиск.

                StrError = OK;
            }
            else if (s[j + i] != SubS[j])
            {
                break;
            }
        }
    }

    StrError = OK;
    return -1;
}
```

3. Разработать программу для решения задачи в соответствии с вариантом индивиду-

ального задания (см. табл.8) с использованием модуля, полученного в результате выполнения пункта 2.
main.c (основная программа)

```c
#include "../../libs/alg/algc.h"

#include <stdio.h>

int main()
{
    do
    {
        string1 input;
        do
        {
            printf("Enter origin string: ");
            fflush(stdout);
            InputStr(input);
        } while (StrError != OK);

        string1 filter;
        do
        {
            printf("Enter filter string: ");
            fflush(stdout);
            InputStr(filter);
        } while (StrError != OK);

        int left = StrCSpn(input, filter);
        if (StrError == OK)
            printf("Characters left after filter: %d", left);
        else if (StrError == BUFFER_OVERFLOW)
            printf("An error accured during execution: buffer was overflown.\n");
        else if (StrError == INVALID_FORMAT)
            printf("An error accured during execution: invalid format.\n");
        else if (StrError == OUT_OF_BOUNDS)
            printf("An error accured during execution: out of bounds.\n");
    } while (StrError != OK);

    return 0;
}
```

task3.c (реализация функции StrCSpn)

```
#include "../algc.h"

unsigned StrCSpn(string1 s, string1 s1) {
    int sLen = Length(s);
    int count = 0;
    if (StrError) return 0;

    for (int i = 0; i < sLen; i++) {
        string1 search;
        search[0] = s[i];
        search[1] = '\0';

        if (Pos(search, s1) == -1 && !StrError)
            count++;
    }

    return count;
}
```

Вывод: в ходе лабораторной работы изучили встроенную структуры данных типа «строка», разработали и использовали производные структуры данных строкового типа.