

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-211

Чувилко Илья Романович

Проверил:

Синюк Василий Григорьевич

Белгород 2022 г.

Вариант: 23. Формат: 5. Задача: 8.

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

Задания

1. Для СД типа строка определить:

1.1 Абстрактный уровень представления СД:

1.1.1 Характер организованности и изменчивости: динамическая линейная последовательность..

1.1.2 Набор допустимых операций: операции доступа, операции инициализации, операции присваивания, операции сравнения, операция конкатенация.

1.2 Физический уровень представления СД:

1.2.1 Схему хранения: последовательная.

1.2.2 Объём памяти, занимаемый экземпляром СД: $V_{\text{стр}} = K + 1$, где K - — максимальное количество символов в строке.

1.2.3 Формат внутреннего представления СД и способ его интерпретации: последовательность из n однобайтовых значений кодов символов и числа 0 –признака окончания строки .

1.2.4 Характеристику допустимых значений: $CAR(string) = 1 + 256 + 256^2 + \dots + 256^K$, где K — максимальное количество элементов в строке.

1.2.5 Тип доступа к элементам: прямой.

1.3 Логический уровень представления СД.

1.3.1 Способ описания СД и экземпляра СД на языке программирования:

```
1 char *str;  
2 char str[]
```

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

3. Разработать программу для решения задачи в соответствии с вариантом индивидуального задания (см. табл.8) с использованием модуля, полученного в результате выполнения пункта 2.

Задание 8. Заголовок: `function WordCmp(s1,s2:string):boolean/ int WordCmp(string1 s1, string1 s2).`

Назначение: сравнение строк(с игнорированием множественных пробелов).

Входные параметры: `s1,s2`. Выходные параметры: нет.

```

#ifdef __FORM5_H
#define __FORM5_H
const ...; // Определение исключительных ситуаций
typedef struct str
{
    char *s; // Указатель на строку
    unsigned max; /* Максимальное количество символов в
строке, определяющееся при инициализации*/
    unsigned N; // Динамическая (текущая) длина строки
};
typedef str *string1;
void InitStr(string1 st, unsigned n);
void WriteToStr(string1 st, char *s);
void WriteFromStr(char *s, string1 st);
void InputStr(string1 st);
void OutputStr(string1 st);
int Comp(string1 s1, string1 s2);
void Delete(string1 s, unsigned Index, unsigned Count);
void Insert(string1 Subs, string1 s, unsigned Index);
void Concat(string1 s1, string1 s2, string1 srez);
void Copy(string1 s, unsigned Index, unsigned Count, string1 Subs);
unsigned Length(string1 s);
unsigned Pos(string1 SubS, string1 s);
void DoneStr(string1 s);
    int StrError; // Переменная ошибок
//...
#endif

```

Выполнение работы:

Переменные ошибок

```

//Операция прошла успешно
static const int STR_SUCCESSFUL = 0;

//Выход за границу максимально разрешенного размера строки
//при вводе в нее данных
static const int STRING_INPUT_ERROR = -1;

//Выход за границу максимально разрешенного размера строки
//при вставке данных из одной строки в другую
static const int STRING_INSERT_ERROR = -2;

//Попытка вставить элемент на место которое не существует
static const int STRING_NO_PLACE = -3;

//Выход за границу максимально разрешенного размера строки
//при склеивании в нее данных
static const int STRING_CONCAT_ERROR = -4;

//Ошибка поиска в меньшей строки большей подстроки
static const int STRING_POS_ERROR = -5;

// Ошибка выхода за пределы массива при удалении подстроки
static const int STRING_DELETE_ERROR = -6;

static int STRING_ERROR = 0;

typedef struct str {
    char *s; // Указатель на строку
    unsigned max; /* Максимальное количество символов в
строке, определяющееся при инициализации*/
    unsigned N; // Динамическая (текущая) длина строки
} str;

typedef str *string1;

```

```

#include "String.h"

// Выделение динамической памяти под строку str, содержащую от 0 до n символов.
void InitStr(string1 str, unsigned n) {
    str->max = n;
    str->N = 0;
    str->s = malloc(sizeof(char) * n);
}

// Запись данных в строку dest из строки source.
void WriteToStr(string1 dest, const char *source) {
    int i = 0;
    while (source[i++] != '\0')
        continue;

    if (dest->max < i)
        STRING_ERROR = STRING_INPUT_ERROR;
    else
        memcpy(dest->s, source, i);
}

// Запись данных в строку dest из строки source.
void WriteFromStr(char *dest, string1 source) {
    memcpy(dest, source->s, source->N);
}

// Ввод строки s с клавиатуры
void InputStr(string1 str) {
    char k = getchar();
    unsigned short i = 0;

    while (k != '\n' && i < str->max) {
        str->s[i] = k;
        ++i;
        k = getchar();
    }

    str->N = i;
    str->s[i++] = '\0';

    if (k != '\n' && i == str->max)
        STRING_ERROR = STRING_INPUT_ERROR;
    else
        STRING_ERROR = STR_SUCCESSFUL;
}

// Вывод строки s на экран монитора
void OutputStr(string1 str) {
    for (int i = 0; i < str->N; i++)
        putchar(str->s[i]);
}

// Сравнивает строки s1 и s2, возвращает 0 если s1 == s2; 1 если s1 > s2; -1 если s1 < s2
int Comp(string1 s1, string1 s2) {
    if (s1->N > s2->N)
        return 1;
    else if (s1->N < s2->N)
        return -1;

    for (int i = 0; i < s1->N; i++) {
        if (s1->s[i] > s2->s[i])
            return 1;
        else if (s1->s[i] < s2->s[i])
            return -1;
    }
}

```

```

    }
    return 0;
}

//Удаляет count символов из строки s начиная с позиции index
void Delete(string1 str, unsigned index, unsigned count) {
    if (index + count >= str->N)
        STRING_ERROR = STRING_DELETE_ERROR;
    else {
        unsigned ptr = index;
        for (unsigned i = index + count; i < str->N; i++)
            str->s[ptr++] = str->s[i];
        str->N -= count;
    }
}

//Вставляет подстроку subS в строку str, начиная с позиции index
void Insert(string1 subS, string1 str, unsigned index) {
    if (index > str->N)
        STRING_ERROR = STRING_NO_PLACE;
    else if (subS->N + str->N > str->max)
        STRING_ERROR = STRING_INSERT_ERROR;
    else {
        for (unsigned i = 0; i < subS->N; i++) {
            str->s[str->N++] = str->s[index + i];
            str->s[index + i] = subS->s[i];
        }
    }
}

//Выполняет конкатенацию строк s1 и s2 результат помещает в sRez
void Concat(string1 s1, string1 s2, string1 sRez) {
    if (s1->N + s2->N > sRez->N)
        STRING_ERROR = STRING_CONCAT_ERROR;

    while (sRez->N < s1->N)
        sRez->s[sRez->N++] = s1->s[sRez->N];
    int i = 0;
    while (i < s1->N)
        sRez->s[sRez->N++] = s2->s[i++];
}

// Записывает count символов в строку subS из строки s начиная с позиции index
void Copy(string1 s, unsigned index, unsigned count, string1 subS) {
    if (index + count > s->N || count > s->max)
        STRING_ERROR = STRING_NO_PLACE;
    else {
        memcpy(subS->s, &s->s[index], count);
    }
}

//Возвращает текущую длину строки s
unsigned Length(string1 s) {
    return s->N;
}

void reverse(string1 s) {
    unsigned length = strlen(s->s) - 1;

    for (int i = 0; i < length / 2; i++) {
        char c = s->s[i];

```

```

        s->s[i] = s->s[length - i];
        s->s[length - i] = c;
    }
}

// Возвращает позицию начиная с которой в строке s располагается строка subS
unsigned Pos(string1 subS, string1 s) {
    if (s->N < subS->N)
        STRING_ERROR = STRING_POS_ERROR;
    else {
        for (unsigned i = 0; i < s->N - subS->N; ++i) {
            unsigned j = 0;
            while ((j < subS->N) && (s->s[i + j] == subS->s[j]))
                ++j;
            if (j == subS->N)
                return i + 1;
        }
        return 0;
    }
}

// Удаляет строку s из динамической памяти.
void DoneStr(string1 s) {
    free(s->s);
    s->N = 0;
    s->max = 0;
}

// сравнение строк(с игнорированием множественных пробелов)
int WordCmp(string1 s1, string1 s2) {
    int s1Ptr = 0;
    int s2Ptr = 0;
    while (s1->s[s1Ptr] != '\0' && s2->s[s2Ptr] != '\0') {
        while (s1->s[s1Ptr] == ' ')
            s1Ptr++;
        while (s2->s[s2Ptr] == ' ')
            s2Ptr++;
        if (s1->s[s1Ptr] != s2->s[s2Ptr])
            return 0;
        s1Ptr++;
        s2Ptr++;
    }
    return 1;
}

```

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