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: студент группы ПВ-223
Мелехов Артём Дмитриевич

Проверил:
асс. Солонченко Роман Евгеньевич

Белгород 2023 г.

Лабораторная работа №1 «Производные структуры данных. Структура данных типа «строка» (Pascal/C)»

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

Содержание отчета:

- Тема лабораторной работы;

- Цель лабораторной работы;

- Условия задач и их решение;

- Вывод.

Задания к лабораторной работе: вариант №7

Номер формата: 7

Задача: 7

Задание 1. Для СД типа строка определить:

I. Абстрактный уровень представления СД: характер организованности и изменчивости, набор допустимых операций.

II. Физический уровень представления СД: схему хранения, объём памяти, занимаемый экземпляром СД, формат внутреннего представления СД и способ его интерпретации, характеристику допустимых значений, тип доступа к элементам.

III. Логический уровень представления СД: способ описания СД и экземпляра СД на языке программирования.

I. Для СД типа строка (в языке программирования Си) абстрактный уровень представления включает характер организованности и изменчивости. Строка является последовательностью символов, которые могут быть организованы в различные структуры, такие как массивы символов или указатели на символы. Набор допустимых операций включает операции чтения, записи, копирования, сравнения и конкатенации строк.

II. Физический уровень представления включает схему хранения строки в памяти компьютера. Строка может быть хранена как массив символов, где каждый символ занимает один байт памяти. Объем памяти, занимаемый строкой, зависит от ее длины и типа данных. Формат внутреннего представления строки и способ его интерпретации зависят от конкретной реализации языка программирования. Характеристика допустимых значений включает ограничения на длину строки и допустимые символы. Тип доступа к элементам строки может быть произвольным, где каждый символ может быть доступен по индексу, или последовательным, где каждый символ доступен только последовательно.

III. Логический уровень представления включает способ описания строки на языке программирования Си. Строка может быть объявлена как массив символов или как указатель на символы. Операции над строками могут быть выполнены с помощью стандартных функций библиотеки Си, таких как `strlen`, `strcpy`, `strcat` и др.

Задание 2. Реализовать СД строкового типа в соответствии с вариантом индивидуального задания в виде модуля. Определить и обработать исключительные ситуации.

Файл `str.h`:

```
#if !defined(__FORM7_H)
#define __FORM7_H
const short str_ok = 0; // Определение исключительных ситуаций
const short str_empty = 1;
const short str_full = 2;

typedef struct str {
    char *s; /* Указатель на строку. Первые два байта строки s
со-держат динамическую длину строки */
    unsigned max; /* Максимальное количество символов в строке,
определяющееся при инициализации */
} str;

typedef str *string1;

void init_str(string1 st, unsigned n);

void write_to_str(string1 st, char *s);

void write_from_str(char *s, string1 st);

void input_str(string1 st);

void output_str(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 sub_s, string1 s);

void done_str(string1 s);
```

```
int str_error; // Переменная ошибок//...
#endif
```

Файл str.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

#include "str.h"

void init_str(string1 st, unsigned n) {
    st->max = n;
    st->s = (char *) malloc(n + 2);

    if (st->s == NULL) {
        str_error = 1; // Нет памяти

        return;
    }

    *(unsigned *) st->s = 0;
    st->s[2] = '\0';
}

void write_to_str(string1 st, char *s) {
    unsigned len = strlen(s);

    if (len > st->max) {
        str_error = 2; // Слишком длинная строка

        return;
    }

    *(unsigned *) st->s = len;
    strcpy(st->s + 2, s);
}

void write_from_str(char *s, string1 st) {
    strcpy(s, st->s + 2);
}

void input_str(string1 st) {
    gets(st->s + 2);
    *(unsigned *) st->s = strlen(st->s + 2);
}

void output_str(string1 st) {
    printf("%s", st->s + 2);
}

int comp(string1 s1, string1 s2) {
```

```
    return strcmp(s1->s + 2, s2->s + 2);
}

void delete(string1 s, unsigned index, unsigned count) {
    unsigned len = *(unsigned *) s->s;

    if (index > len || index + count > len) {
        str_error = 3; // Ошибка параметров

        return;
    }

    memmove(s->s + index + 2, s->s + len + 2 - count, count);
    *(unsigned *) s->s = len - count;
}

void insert(string1 subs, string1 s, unsigned index) {
    unsigned len = *(unsigned *) s->s;

    if (index > len) {
        str_error = 3; // Ошибка параметров

        return;
    }

    if (len + *(unsigned *) subs->s > s->max) {
        str_error = 2; // Слишком длинная строка

        return;
    }

    memmove(s->s + len + 2 + *(unsigned *) subs->s, s->s + index
+ 2, len - index);
    memcpy(s->s + index + 2, subs->s + 2, *(unsigned *) subs-
>s);
    *(unsigned *) s->s = len + *(unsigned *) subs->s;
}

void concat(string1 s1, string1 s2, string1 srez) {
    unsigned len = *(unsigned *) s1->s + *(unsigned *) s2->s;

    if (len > srez->max) {
        str_error = 2; // Слишком длинная строка
        return;
    }

    memcpy(srez->s + 2, s1->s + 2, *(unsigned *) s1->s);
    memcpy(srez->s + 2 + *(unsigned *) s1->s, s2->s + 2,
*(unsigned *) s2->s);
    *(unsigned *) srez->s = len;
}

void copy(string1 s, unsigned index, unsigned count, string1
```

```
subs) {
    unsigned len = *(unsigned *) s->s;

    if (index > len || index + count > len) {
        str_error = 3; // Ошибка параметров

        return;
    }

    if (count > subs->max) {
        str_error = 2; // Слишком длинная строка

        return;
    }

    memcpy(subs->s + 2, s->s + index + 2, count);
    *(unsigned *) subs->s = count;
}

unsigned length(string1 s) {
    return *(unsigned *) s->s;
}

unsigned pos(string1 sub_s, string1 s) {
    char *p = strstr(s->s + 2, sub_s->s + 2);

    if (p == NULL)
        return 0;
    else
        return p - s->s - 2;
}

void done_str(string1 s) {
    free(s->s);
}
```

Задание 3. Разработать программу для решения задачи в соответствии с вариантом индивидуального задания с использованием модуля, полученного в результате выполнения задания 2.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

#include "str.h"

string1 *sud_word(char *s, unsigned n) {
    string1 *s1 = (string1 *) malloc(sizeof(string1));

    init_str(s1, strlen(s) - n + 1);
    write_to_str(s1, s + n);
```

```
    return s1;
}
```

Вывод: в ходе выполнения лабораторной работы была изучена встроенная структура данных типа «строка», также была разработана производная структура данных строкового типа.