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-202
Аладиб язан

Проверил:

Кабалянц Петр Степанович
Маньшин Илья Михайлович

Лабораторная работа № 2
«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.

Задания к работе:

1. Для СД типа строка определить:
    1.1. Абстрактный уровень представления СД:
        1.1.1. Характер организованности и изменчивости.
        1.1.2. Набор допустимых операций.
    1.2. Физический уровень представления СД:
        1.2.1. Схему хранения.
        1.2.2. Объем памяти, занимаемый экземпляром СД.
        1.2.3. Формат внутреннего представления СД и способ его интерпретации.
        1.2.4. Характеристику допустимых значений.
        1.2.5. Тип доступа к элементам.
    1.3. Логический уровень представления СД.
    Способ описания СД и экземпляра СД на языке программирования.
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 варианта №2:

Номер варианта	Номер формата	Задача
2	2	2

1.1. Абстрактный уровень представления СД:

Характер организованности и изменчивости :
Характер организованности – последовательность.
Изменчивость – динамическая.

Набор допустимых операций :
1) Выделение динамической памяти под строку
2) Запись данных в строку из указателя на char
3) Запись данных из строки по указателю на char
4) Ввод строки с клавиатуры
5) Вывод строки на экран
6) Сравнение строк
7) Удаление определенного количества символов из строки, начиная с определенной позиции
8) Вставка подстроки в строку, начиная с определенной позиции
9) Выполняет конкатенацию строк.
10) Запись определенного количества символов в строку из другой строки, начиная с определенной позиции
11) Определение позиции, в которой находиться подстрока
12) Вставка подстроки в строку, начиная с определенной позиции

1.2. Физический уровень представления СД:

Схему хранения : последовательность.

Объем памяти, занимаемый экземпляром СД :
Объем памяти зависит от зависит от максимального количества элементов (символов) в строке и определяется формулой
Vстр = K + 1, где K — максимальное количество символов в строке.

Формат внутреннего представления СД и способ его интерпретации :
Массив элементов типа char, которые нумеруются от 1 до k, заканчивается нулевым символом '\0', элемент, с индексом K равен 0 как признак конца строки.

Характеристику допустимых значений :
Количество допустимых значений СД типа строка = CAR(string) = $1 + 256^1 + 256^2 + \ldots + 256^k$
где K – максимальное число элементов в строке.

Тип доступа к элементам : Прямой

1.3. Логический уровень представления СД :

String1 <Итендификатор> = CreateStr();

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

Код программы FORM2.h:

```
#ifndef lab2_FORM2_H
#define lab2_FORM2_H
#include <stdio.h>
#include <malloc.h>

typedef struct {
    char *s;
    unsigned max;
} str;
typedef str *string1;
//Создает строку возвращает указатель на строку
string1 CreateStr();
//Выделение динамической памяти под строку st, содержащую от 0 до n символов.
void InitStr(string1 st, unsigned n);
/* Запись данных в строку st из строки s.
 Строка s заканчивается нулевым символом '\0' */
void writeToStr(string1 st, char *s);
/* Запись данных в строку s из строки st.
 Строка s заканчивается нулевым символом '\0' */
void writeFromStr(char *s, string1 st);
//Ввод строки s с клавиатуры
void inputStr(string1 s);
//Вывод строки st на экран монитора
void outputStr(string1 st);
/* Сравнивает строки строки s1 и s2. Возвращает 0, если s1 = s2, если s1 > s2,
 то 1, если s1 < s2, то -1. */
int comp(string1 s1, string1 s2);
// Удаляет count символов из строки s, начиная с позиции index
void delete(string1 s, unsigned index, unsigned count);
// Вставляет строку подстроку subs в строку s начиная с позиции index
void insert(string1 subs, string1 s, unsigned index);
//Выполняет конкатенацию строк s1 и s2. Результат помещает в srez.
void Concat(string1 s1, string1 s2, string1 srez);
// Записывает count символов в строку subs из строки s, начиная с позиции index
void copy(string1 s, unsigned index, unsigned count, string1 subs);
// Возвращает текущую длину строки s
unsigned length(string1 s);
// Возвращает позицию, начиная с которой в строке s располагается подстрока
// subs, или -1, если s не содержит subs
unsigned pos(string1 subs, string1 s);
#endif //lab2_FORM2_H
```

Код программы FORM2.c:

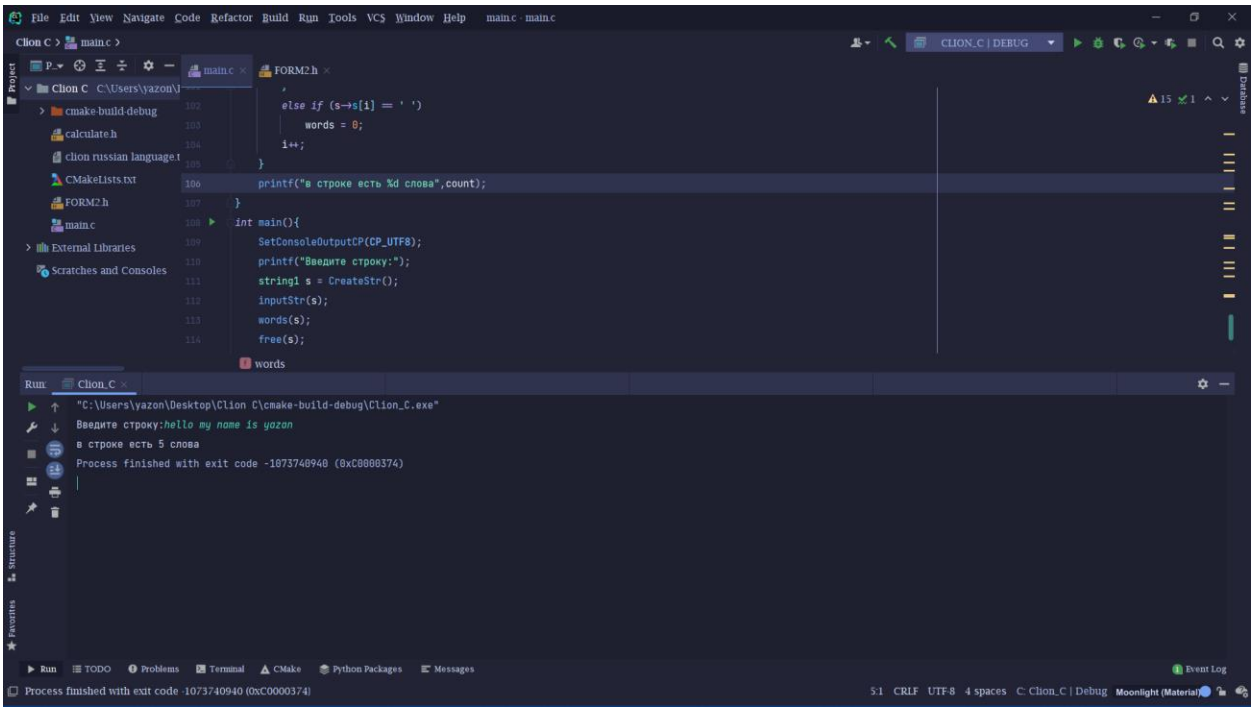
```
#include "FORM2.h"

string1 CreateStr(){
    return (string1)malloc(sizeof(str));
}
void writeToStr(string1 st, char *s){
    st->max = 0;
    while (*s){
        st->s[st->max++] = *(s++);
    }
}
void writeFromStr(char *s, string1 st){
    unsigned i;
    for (i = 0; i < st->max; ++i) {
        s[i] = st->s[i];
    }
    s[++i] = '\0';
}
void inputStr(string1 s){
    unsigned len = 0;
    char c;
    while ((c = getchar()) != '\n')
        s->s[len++] = c;
    s->max = len;
}
void outputStr(string1 st){
    for (unsigned i = 0; i < st->max; i++)
        putchar(st->s[i]);
}
int comp(string1 s1, string1 s2){
    unsigned i = 0;
    while (s1->s[i] == s2->s[i]){
        if (s1->s[i] == '\0')
            return 0;
        i++;
    }
    return s1->s[i] - s2->s[i];
}
void copy(string1 s, unsigned index, unsigned count, string1 subs){
    int i;
    for (i = 0; i < count; ++i) {
        subs->s[i] = s->s[index+i];
    }
    subs->s[i] = '\0';
}
unsigned length(string1 s){
    return s->max;
}
unsigned pos(string1 subs, string1 s){
    unsigned i = 0;
    if (subs->max > s->max)
        return -1;
    while (s->s[i] != subs->s[0] && s->s[i])
        i++;
    int result = (int)i;
    for (int j = 0; j < subs->max; ++j, ++i) {
        if (subs->s[j] != s->s[i])
            return -1;
    }
    return result;
}
```

```
void delete(string1 s, unsigned index, unsigned count){
    for (int i = count + index; i < s->max; i++){
        s->s[i - count] = s->s[i];
    }
    s->max -= count;
}
void insert(string1 subs, string1 s, unsigned index){
    string1 end = CreateStr();
    writeToStr(end, &(s->s[index]));
    for (int i = 0; i < subs->max; ++i) {
        s->s[index + i] = subs->s[i];
    }
    s->max += subs->max;
    for (int i = index + subs->max, j = 0; i < s->max && j < end->max; i++, j++)
        s->s[i] = end->s[j];
}
void Concat(string1 s1, string1 s2, string1 srez){
    for (int i = 0; i<s1->max; i++){
        srez->s[i] = s1->s[i];
    }
    int i = s1->max;
    for (int j = 0; j<s2->max; j++){
        srez->s[j+i] = s2->s[j];
    }
    srez ->max = s1->max + s2->max;
    srez->s[srez->max] = '\0';
}
```

Решение задача :

```
#include "FORM2.h"
#include <windows.h>
void words(string1 s){
    unsigned i = 0,words =0,count=0;
    while (s->s[i] == ' ' && s->s[i] != '\0')
        i++;
    words =0;
    while (s->s[i] != '\0') {
        if (s->s[i] != ' ' && words == 0)
        {
            words = 1;
            count++;
        }
        else if (s->s[i] == ' ')
            words = 0;
        i++;
    }
    printf("в строке есть %d слова",count);
}
int main(){
    SetConsoleOutputCP(CP_UTF8);
    printf("Введите строку:");
    string1 s = CreateStr();
    inputStr(s);
    words(s);
    free(s);
}
```



Вывод

В ходе выполнения лабораторной работы была изучена встроенная структура данных типа «строка», а также разработана и использована производная структура данных строкового типа.