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» C»

Выполнил: ст. группы ПВ-223

Игнатьев Артур Олегович

Проверил:

асс. Солонченко Роман Евгеньевич

Белгород 2023г.

**Лабораторная работа №2**

**«Производные структуры данных.**

**Структура данных типа «строка» C»**

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

**Содержание отчета:**

1. Тема лабораторной работы.

2. Цель работы.

3. Характеристика СД «строка» (пункт 1 задания).

4. Индивидуальное задание.

5. Текст модуля для реализации СД типа «строка», текст программы для отладки модуля, тестовые данные, результат работы программы.

6. Текст программы для решения задачи с использованием модуля, те-стовые данные, результат работы программы.

**Задание к лабораторной работе :**

Вариант №3

**Заголовок:** void Center(string1 s1, string1 s2, unsigned l).

**Назначение:** поиск последнего вхождения подстроки s2 в строку s1.

**Входные параметры:** s1,l.

**Выходные параметры:** s2.

**Формат 3:**

#if !defined(\_\_FORM3\_H)

#define \_\_FORM3\_H

const ...; // Определение исключительных ситуаций

typedef char string1[1024]; /\* Первые два байта содержат динамическую длину строки \*/

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);

int StrError; // Переменная ошибок

//...

#endif

1. Для типов данных определить

1.1 Абстрактный уровень представления СД

1.1.1 Характер организованности и изменчивости

• void: встроенный, статический, простейший.

• string: встроенный, динамический, простейший.

• unsigned: встроенный, статический, простейший.

1.1.2 Набор допустимых операций

• void:

Операции взятие адреса, присваивание, приведение типов.

• string:

Операции присваивания, сравнения, конкатенации.• unsigned:

Операции сравнения, инкрементирование, декрементирование, сложение, вычитание, умножение, деление, унарные операции, взятие адреса, логические операции, операции присваивания, приведение типов.

1.2 Физический уровень представления СД

1.2.1 Схема хранения

• void: последовательная.

• string: последовательная.

• unsigned: последовательная.

1.2.2 Объем памяти, занимаемый экземпляром СД

• void: -.

• string: 1 байт.

• unsigned: 4 байта.

1.2.3 Формат внутреннего представления СД и способ его интерпретации

• void: -.

• string: K + 1(1 байт для нулевого символа '\0', который указывает на конец строки), где K — максимальное количество символов в строке.

• unsigned: 4 бит, старший бит не отводиться под знак, все 4 бит отводится под значение.

1.2.4 Характеристика допустимых значений

• void: -.

• string: [-128;127].

•unsigned: [0;4 294 967 295].

1.2.5 Тип доступа к элементам

• void: через указатель на другой тип.

• string: прямой.

•unsigned: прямой.

1.3 Логический уровень представления СД

1.3.1. Способ описания СД и экземпляра СД на языке программирования

• void:

void /\*имя переменной\*/;

•string:

string /\*имя переменной\*/;

• unsigned:

unsigned /\* имя переменной \*/;

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

Файл str.h:

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <windows.h>

#define MAX\_STRING\_LENGTH 1024  
  
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[MAX\_STRING\_LENGTH];  
  
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);

Файл str.c:

#include "str.h"  
  
const int OK = 0;  
const int BUFFER\_OVERFLOW = 1;  
const int INVALID\_FORMAT = 2;  
const int OUT\_OF\_BOUNDS = 3;  
  
int StrError = 0; // Инициализация переменной ошибок  
  
// Функция записи строки в форматированный вид с длиной в начале  
void WriteToStr(string1 st, char \*s) {  
 if (strlen(s) > MAX\_STRING\_LENGTH - 2) {  
 StrError = 1; // Ошибка: превышена максимальная длина строки  
 return;  
 }  
  
 sprintf(st, "%02zu%s", strlen(s), s);  
}  
  
// Функция чтения строки из форматированного вида с длиной в начале  
void WriteFromStr(char \*s, string1 st) {  
 int length;  
 sscanf(s, "%02d", &length);  
 strncpy(st, s + 2, length);  
 st[length] = '\0';  
}  
  
// Функция ввода строки с клавиатуры и записи в форматированный вид  
void InputStr(string1 st) {  
 printf("Введите строку: ");  
 scanf("%s", st);  
 sprintf(st, "%02zu%s", strlen(st), st);  
}  
  
// Функция вывода строки без длины в начале  
void OutputStr(string1 st) {  
 printf("%s\n", st + 2);  
}  
  
// Функция сравнения двух строк без длины в начале  
int Comp(string1 s1, string1 s2) {  
 return strcmp(s1 + 2, s2 + 2);  
}  
  
// Функция удаления подстроки из строки по индексу и количеству символов  
void Delete(string1 s, unsigned Index, unsigned Count) {  
 if (Index >= strlen(s + 2)) {  
 StrError = 2; // Ошибка: индекс превышает длину строки  
 return;  
 }  
  
 memmove(s + 2 + Index, s + 2 + Index + Count, strlen(s + 2 + Index + Count) + 1);  
 sprintf(s, "%02zu%s", strlen(s + 2), s + 2);  
}  
  
// Функция вставки подстроки в строку по указанному индексу  
void Insert(string1 Subs, string1 s, unsigned Index) {  
 if (Index > strlen(s + 2)) {  
 StrError = 2; // Ошибка: индекс превышает длину строки  
 return;  
 }  
  
 char temp[MAX\_STRING\_LENGTH];  
 strncpy(temp, s + 2, Index);  
 temp[Index] = '\0';  
 strcat(temp, Subs + 2);  
 strcat(temp, s + 2 + Index);  
 strcpy(s + 2, temp);  
 sprintf(s, "%02zu%s", strlen(s + 2), s + 2);  
}  
  
// Функция конкатенации двух строк  
void Concat(string1 s1, string1 s2, string1 srez) {  
 strcpy(srez, s1);  
 strcat(srez, s2 + 2);  
 sprintf(srez, "%02zu%s", strlen(srez + 2), srez + 2);  
}  
  
// Функция копирования подстроки из строки по индексу и количеству символов  
void Copy(string1 s, unsigned Index, unsigned Count, string1 Subs) {  
 if (Index + Count > strlen(s + 2)) {  
 StrError = 2; // Ошибка: индекс + количество превышает длину строки  
 return;  
 }  
  
 strncpy(Subs + 2, s + 2 + Index, Count);  
 Subs[Count + 2] = '\0';  
 sprintf(Subs, "%02zu%s", strlen(Subs + 2), Subs + 2);  
}  
  
// Функция определения длины строки без учета длины в начале  
unsigned Length(string1 s) {  
 return strlen(s + 2);  
}  
  
// Функция поиска позиции подстроки в строке без учета длины в начале  
unsigned Pos(string1 SubS, string1 s) {  
 char \*pos = strstr(s + 2, SubS + 2);  
 return pos ? pos - (s + 2) : 0;  
}

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

Файл str.h:

void Center(string1 s1, string1 s2, unsigned l);

Файл str.c:

void Center(string1 s1, string1 s2, unsigned l) {  
 unsigned s1Length = Length(s1);  
  
 if (s1Length >= l) {  
 printf("Ошибка: Невозможно центрировать строку. Длина s1 больше или равна l.\n");  
 return;  
 }  
  
 unsigned leftPadding = (l - s1Length) / 2;  
  
 // Заполнение s2 пробелами  
 for (unsigned i = 0; i < l; ++i) {  
 s2[i] = ' ';  
 }  
  
 // Копирование s1 в центр s2  
 Copy(s1, 0, s1Length, s2 + leftPadding);  
 s2[leftPadding + s1Length] = '\0'; // Завершаем строку s2  
}

Файл main.c:

#include "../../libs/alg/alg.h"  
  
int main() {  
 SetConsoleOutputCP(CP\_UTF8);  
  
 string1 s1, s2;  
 unsigned l;  
  
 InputStr(s1);  
  
 printf("Введите длинну l: ");  
 scanf("%u", &l);  
  
 // Вызов функции Center  
 Center(s1, s2, l);  
  
 // Вывод результата  
 printf("Центрированная строка s2: ");  
 OutputStr(s2);  
  
 return 0;  
}

Вывод: на этой лабораторной работе я изучил встроенной структуры данных типа «строка», разработал и использовал производные структуры данных строкового типа.