**Лабораторная работа № 2**

УКАЗАТЕЛИ, ФУНКЦИИ

1. Цель работы

Освоить правила написания и использования функций в языке СИ. Научиться использовать указатели при обработке массивов данных.

3. Методические указания.

В ходе выполнения лабораторной работы потребуется выполнить сортировку массива строк. Для ускорения этой операции обычно используется дополнительный массив указателей. В этом случае вместо перестановки двух строк с помощью функции ***strcpy()*** используется перестановка указателей обычным присваиванием.

Формирование массива указателей можно совместить с вводом строк.

4. Порядок выполнения работы

4.1. Написать программу сортировки массива строк по вариантам (табл. 2.2.). Ввод данных, сортировку и вывод результатов оформить в виде функций. Входные и выходные параметры функции сортировки указаны в таблице. Входные и выходные параметры функций для ввода-вывода приведены в таблице 2.1.

4.2. Модифицировать программу п.1., применив в функциях передачу параметров и возврат результатов по ссылке (с использованием указателей). Сравнить результаты.

Таблица 2.1

|  |
| --- |
| Прототип функции для ввода строк |
| ***length = inp\_str(char\* string, int maxlen);***  ***// length*** - длина строки  ***// string*** - введенная строка  ***// maxlen*** - максимально возможная длина строки (размерность массива ***string***) |
| Прототип функции для вывода строк |
| ***void out\_str(char\* string, int length, int number);***  ***// string -*** выводимая строка  ***// length -*** длина строки  ***// number*** - номер строки |

Таблица 2.2

| Вари ант | Задание | Входные  параметры | Выходные  Параметры |
| --- | --- | --- | --- |
| 1 | Расположить строки по возрастанию длины | 1. Массив 2. Размерность массива | 1. Количество перестановок 2. Длина меньшей строки |
| 2 | Расположить строки по убыванию длины | 1. Массив 2. Размерность массива | 1. Количество перестановок 2. Длина большей строки |
| 3 | Расположить строки в алфавитном порядке | 1. Массив 2. Размерность массива | 1. Количество перестановок  2. Первая буква первой строки |
| 4 | Расположить строки в обратном алфавитном порядке | 1. Массив  2. Размерность массива | 1. Количество перестановок 2. Длина первой строки |
| 5 | Расположить строки по возрастанию количества слов | 1. Массив 2. Размерность массива | 1. Количество перестановок  2. Первый символ последней строки |
| 6 | Расположить строки по убыванию количества слов | 1. Массив  2. Размерность массива | 1. Количество перестановок 2. Максимальное количество слов |
| 7 | Расположить строки по возрастанию количества цифр | 1. Массив  2. Размерность массива. | 1. Количество цифр  2. Вторая цифра строки |
| 8 | Расположить строки по убыванию количества цифр | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Количество цифр |
| 9 | Расположить строки по возрастанию длины первого слова | 1. Массив  2. Размерность массива. | 1. Максимальная длина слова  2. Количество перестановок |
| 10 | Расположить строки по убыванию длины первого слова | 1. Массив  2. Размерность массива. | 1. Минимальная длина слова  2. Последняя буква первого слова |
| 11 | Расположить строки в алфавитном порядке по последней букве строки | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Последняя буква первой строки |
| 12 | Расположить строки в обратном алфавитном порядке по последней букве строки | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Последняя буква последней строки |
| 13 | Расположить строки по возрастанию количества пробелов в строке | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Максимальное количество пробелов в строке |
| 14 | Расположить строки по убыванию количества пробелов в строке | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Максимальное количество пробелов подряд в строке |
| 15 | Расположить строки по возрастанию длины последнего слова | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Максимальная длина последнего слова |
| 16 | Расположить строки по убыванию длины последнего слова | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Минимальная длина последнего слова |
| 17 | Расположить строки по возрастанию длины самого длинного слова | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Максимальная длина самого длинного слова |
| 18 | Расположить строки по убыванию длины самого длинного слова | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Минимальная длина самого длинного слова |
| 19 | Расположить строки по возрастанию длины самого короткого слова | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Максимальная длина самого короткого слова |
| 20 | Расположить строки по убыванию длины самого короткого слова | 1. Массив  2. Размерность массива | 1. Количество перестановок  2. Минимальная длина самого короткого слова |

5. Содержание отчета

5.1. Титульный лист

5.2. Цель работы.

5.3. Задание.

5.4. Текст(ы) программ.

5.5. Полученные результаты.

5.6. Выводы.

6. Контрольные вопросы

**6.1. Правила описания указателей.**

**6.2. Как связаны указатели и массивы?**

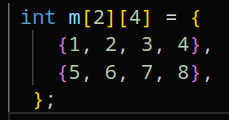
**6.3. Назначение прототипа функции.**

**6.4. Структура описания функции. // описание=прототип**

**6.5. Какие операции допустимы с адресами?**

**6.6. Массивы указателей – описание и использование.**

**6.7. Привести пример описания двумерного массива.**



**6.8. Привести пример доступа к элементу двумерного массива через указатель на него.**

**6.9. Какие классы памяти существуют в языке СИ?**

Классы памяти определяют срок жизни и область видимости переменной или функции.

**Auto** — класс памяти по-умолчанию для локальных переменных. Располагаются на стеке, область видимости ограничивается своим блоком.

int auto x = 10; // auto можно опустить

**Register** — класс памяти для переменных, которые мы хотим расположить на регистрах процессора, а не в оперативной памяти. Не имеют адреса, попытку взять его вызовет ошибку компиляции.

register int x = 20;

**Static** — хранятся в data или bss сегменте. Время жизни совпадает со временем жизни приложения. Область видимости таких переменных ограничена блоком, в котором они объявлены.

static int counter = 0;

**External** — внешние (глобальные) переменные. Срок жизни такой же, как у всей программы в целом, я так понимаю. Прямым текстом это нигде не написано, но это было бы логично.

