Отчет по лабораторной работе № 5

по курсу "Фундаментальная информатика"

Выполнил студент группы М8О-114БВ-24: Дробышев Егор Павлович, № по списку 29

Контакты e-mail: tru.899@yandex.ru

Работа выполнена: «30» ноября 2024 г.

Преподаватель: доцент каф. 806 Никулин Сергей Петрович

Отчет сдан « » 2024г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема: «**Множества»
2. **Цель работы:** научиться составлять программы на языке Си, которые выполняют обработку текста при помощи множеств.
3. **Задание**: Есть ли слово, содержащее более одной прописной буквы (Вариант – 29)
4. **Оборудование: оборудование ПЭВМ студента, если использовалось:**

Процессор _Intel Core i5_ с ОП 8 Гб НМД ___256___Гб. Монитор 1920x1080~60Hz. Другие устройства не использовались

5. **Программное обеспечение: программное обеспечение ЭВМ студента:**

Операционная система семейства _Linux_, наименование _Ubuntu_ версия _24.04 _____

интерпретатор команд _GNU bash_ версия _5.2.21(1)_.

Редактор текстов ____emacs ___версия _____29.3 _____

Утилиты операционной системы: gcc, gdb

Местонахождение и имена файлов программ и данных на домашнем компьютере:

/home/tru__

6. **Идея, метод, алгоритм решения задачи** (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Используем unsigned int для представления множества прописных букв в слове. Каждая прописная буква будет представлена как бит в целочисленном значении ('A' — 1 << 0 (бит 0), 'B' — 1 << 1 (бит 1) и тд). Функция bit_count подсчитывает количество установленных битов в числе, которое будет использоваться для подсчета прописных букв в слове. Функция char_to_set проверяет, является ли символ прописной буквой и возвращает соответствующее множество. Для этого можно использовать 1u << (c - 'A'), где c — это символ в верхнем регистре. Если символ не является прописной буквой, возвращаем 0. Функция is_up_word обрабатывает строку посимвольно. Для каждого слова будем отслеживать прописные буквы. Если в слове больше одной прописной буквы (то есть более одного бита установлено в маске), возвращаем 1. Будем использовать флаг in_word для отслеживания, находимся ли мы в слове, и сбрасывать этот флаг при встрече разделителей. В процессе обработки строки можно использовать два состояния:

Состояние 0, где находимся на разделителе. Если встречаем прописную букву, переходим в состояние 1.

Состояние 1, где находимся в слове. Если встречаем разделитель, проверяем количество прописных букв в слове.

После завершения ввода передаем строку в функцию is_up_word. Если она возвращает 1, выводим сообщение: «Есть слово с более чем одной прописной буквой». Иначе — «Нет слова с более чем одной прописной буквой». Освобождаем выделенную память после обработки.

7. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

int bit_count(unsigned int word) {
    int count = 0;
    while (word) {
        count += word & 1;
        word >>= 1;
    }
    return count;
}

int char_to_set(char c) {
```

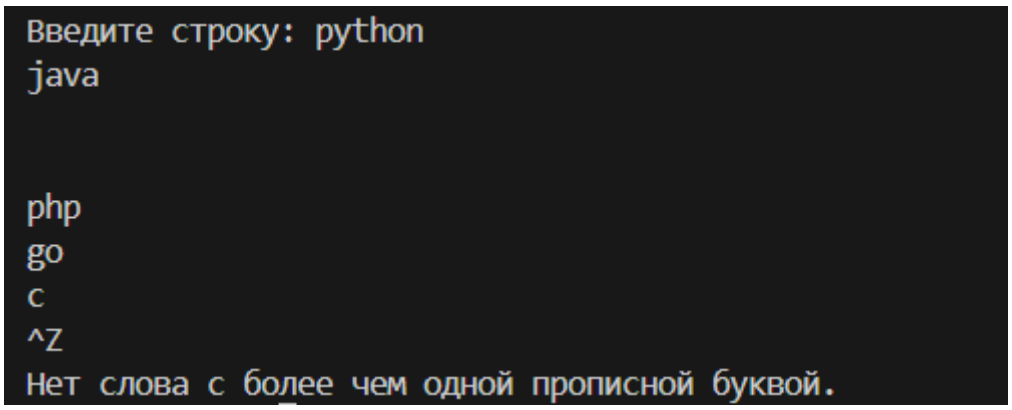
```c
    if (c >= 'A' && c <= 'Z') {
        return 1u << (c - 'A');
    }
    return 0;
}

int is_up_word(const char *str) {
    unsigned int word = 0;
    int in_word = 0;

    for (int i = 0; str[i] != '\0'; i++) {
        char c = str[i];
        if (isupper(c)) {
            word |= char_to_set(c);
            in_word = 1;
        }

        if (!isalpha(c) && !isspace(c)) {
            if (in_word) {
                if (bit_count(word) > 1) {
                    return 1;
                }
            }
            in_word = 0;
            word = 0;
        }

        if (isspace(c)) {
            if (in_word) {
                if (bit_count(word) > 1) {
                    return 1;
                }
            }
            in_word = 0;
            word = 0;
        }
    }

    if (in_word && bit_count(word) > 1) {
        return 1;
    }

    return 0;
}


int main() {
    size_t capacity = 1024;
    char *str = malloc(capacity);

    size_t length = 0;
```

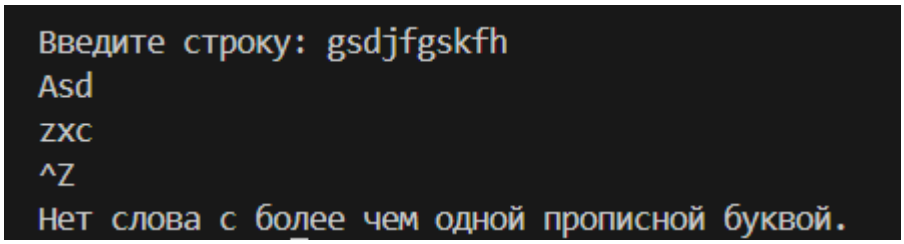
```
    int ch;
    printf("Введите строку: ");
    while ((ch = getchar()) != EOF) {
        if (length >= capacity - 1) {
            capacity *= 2;
            char *temp = realloc(str, capacity);
            str = temp;
        }
        str[length++] = ch;
    }
    str[length] = '\0';

     if (is_up_word(str)) {
        printf("Есть слово с более чем одной прописной буквой.\n");
    } else {
        printf("Нет слова с более чем одной прописной буквой.\n");
    }

    free(str);
    return 0;
}
```

8. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

```
Введите строку: qwerty
HELLO WorlD
GOOD,bye

ok.OK
^Z
Есть слово с более чем одной прописной буквой.
```

```
Введите строку: AB
cd
eF
^Z
Есть слово с более чем одной прописной буквой.
```





9. **Дневник отладки** должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

_____

_____

_____

11. Выводы:

В ходе выполнения лабораторной работы я научился работать со множествами на языке Си. Недочёты при выполнении задания могут быть устранены следующим образом:

_____

_____

_____

Подпись студента _____