Отчет по лабораторной работе № 13 по курсу “Фундаментальная информатика”

Студент группы М80-103Б-21 Белоносов Кирилл Алексеевич, № по списку 3

Контакты почта kirillbelonosov@yandex.ru, telegram: @KiRiLLBElNOS
Работа выполнена: «24» ноября 2021г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

Отчет сдан « » _____20__ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Множества

2. **Цель работы:** Составить программу на языке Си выполняющую выданное преподавателем задания, при этом реализовав математическую абстракцию множества.

3. **Задание** (Вариант 3):
На вход подаётся произвольный набор английских слов, разделённых пробелами, запятыми, знаками табуляции и границами строк. Букву 'y' для простоты считать согласной (гласными или согласными бывают звуки, но не буквы). Необходимо проверить выполнение следующего условия: есть ли слово, хотя бы одна гласная которого повторяется? При решении задачи необходимо реализовать математическую абстракцию множества (сам тип, полное/пустое множество, пересечение, симметрическая разность и т.д.). Для реализации множества необходимо использовать битовые операции, но при этом запрещается нарушать принципы абстракции и инкапсуляции: пользовательский код должен зависеть только от интерфейса, но не от его реализации. При имплементации множеств можно считать, что все они имеют мощность не больше 32. Помимо обычных заголовочных файлов разрешается использовать ctype.h, inttypes.h и stdbool.h. Код функции main должен зависеть только от интерфейса АТД множество.

4. **Оборудование** (студента):
Процессор Intel Core i7-1165G7 @ 4x2.8GH с ОП 16384 Мб, НМД 512 Гб. Монитор 1920x1080

5. **Программное обеспечение (**студента**):**

Операционная система семейства: *linux*, наименование: *ubuntu*, версия 20.04.3 *LTS*
интерпретатор команд: *bash* версия *5.0*.17(1)
Система программирования Visual studio code
Редактор текстов *emacs* версия *27.1*
Утилиты операционной системы --
Прикладные системы и программы
Местонахождение и имена файлов программ и данных на домашнем компьютере –

6. **Идея, метод, алгоритм**
Основная идея заключается в том, что мы посимвольно считываем слова и если находим гласную, то записываем ее во множество, если же она уже во множество выводим Yes, если же до конца до конца программы не встречается слово с двумя гласными, то выводим No. На каждом знаке-разделителе (‘\n’, ‘\t’, ‘ ‘, 13) очищаем множество.

7. **Сценарий выполнения работы.**

Входные данные	Выходные данные
Ada bfd Ikt kfdjkkr	Yes
Jjkfk Lolr NBgu eregd ioi	Yes
ToTr fdtev KKKl pOirf Erv ngner fgew aB Ab	No

8. **Распечатка протокола**

```
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

typedef unsigned set_data_elem;
enum {
```

```
    bits_per_char = 8,
    bits_per_elem = sizeof(set_data_elem) * bits_per_char,
    datalen = (1 << bits_per_char) / bits_per_elem
};

typedef struct {
    set_data_elem data[datalen];
} set;

void set_clear(set *s)
{
    memset(s -> data, 0, sizeof(s->data));
}

void set_insert(set *s, int c)
{
    s -> data[c / bits_per_elem] |= 1u << (c % bits_per_elem);
}
void set_generate(set *s, bool indicator(int))
{
    set_clear(s);
    for (int i = 0; i != 1 << bits_per_char; ++i) {
        if (indicator(i)) {
            set_insert(s, i);
        }
    }
}

void set_erease(set *s, int c)
{
    s -> data[c / bits_per_elem] &= ~(1u << c % bits_per_elem);
}

bool set_in(const set *s, int c)
{
    return (s->data[c / bits_per_elem] & (1u << c % bits_per_elem)) != 0;
}

int set_size(const set *s)
{
    int size = 0;
    for (int i = 0; i != 1 << bits_per_char; ++i) {
        if (set_in(s, i)) {
            ++size;
        }
    }
    return size;
}

bool set_equal(const set *s1, const set *s2)
{
    for (int i = 0; i != datalen; ++i) {
        if (s1->data[i] != s2->data[i]) {
            return false;
        }
    }
    return true;
}

bool set_includes(const set *s1, const set *s2)
{
    for (int i = 0; i != datalen; ++i) {
        if ((s1 -> data[i] | s2->data[i]) != s1 -> data[i]) {
            return false;
```

```
        }
    }
    return true;
}

set set_union(const set *s1, const set *s2)
{
    set result;
    for (int i = 0; i != datalen; ++i) {
        result.data[i] = s1 -> data[i] | s2 -> data[i];
    }
    return result;
}

set set_intersection(const set *s1, const set *s2)
{
    set result;
    for (int i = 0; i != datalen; ++i) {
        result.data[i] = s1 -> data[i] & s2 -> data[i];
    }
    return result;
}

set set_difference(const set *s1, const set *s2)
{
    set result;
    for (int i = 0; i != datalen; ++i) {
        result.data[i] = s1 -> data[i] & ~(s2 -> data[i]);
    }
    return result;
}

set set_symmetric_difference(const set *s1, const set *s2)
{
    set result;
    for (int i = 0; i != datalen; ++i) {
        result.data[i] = s1 -> data[i] ^ s2 -> data[i];
    }
    return result;
}
bool is_alpha(int c)
{
    return isalpha(c);
}
bool is_digit(int c)
{
    return isdigit(c);
}

int main(void)
{
    set s1;
    set_clear(&s1);
    char Symbol;
    bool Repeat = 0;
    while (scanf("%c", &Symbol) != EOF) {
        Symbol = tolower(Symbol);
        switch (Symbol) {
            case 'a':
                if (set_in(&s1, Symbol)) {
                    Repeat = 1;
                } else {
                    set_insert(&s1, 'a');
                }
```

```
                break;

            case 'e':
                if (set_in(&s1, Symbol)) {
                    Repeat = 1;
                } else {
                    set_insert(&s1, 'e');
                }
                break;

            case 'i':
                if (set_in(&s1, Symbol)) {
                    Repeat = 1;
                } else {
                    set_insert(&s1, 'i');
                }
                break;

            case 'o':
                if (set_in(&s1, Symbol)) {
                    Repeat = 1;
                } else {
                    set_insert(&s1, 'o');
                }
                break;

            case 'u':
                if (set_in(&s1, Symbol)) {
                    Repeat = 1;
                } else {
                    set_insert(&s1, 'u');
                }
                break;

            case ' ':
                set_clear(&s1);
                break;

            case '\n':
                set_clear(&s1);
                break;

            case '\t':
                set_clear(&s1);
                break;

            case ',':
                set_clear(&s1);
                break;
            case 13:
                set_clear(&s1);
                break;
            default:
                break;

        }
    }
    if (Repeat == 1) {
        printf("Yes\n");
    } else {
        printf("No\n");
    }
    return 0;
}
```

9. Выводы

В данной лабораторной работе я познакомился с реализацией множеств на языке Си, изучил битовые операции и использовав полученные знания выполнил поставленное задание. Работа интересна тем, что в процессе её выполнения был создан новый тип данных отвечающий за символьное множество и набор функций, отвечающий за работу с этим типом.

Подпись студента

_____