**Отчет по лабораторной работе № 13** по курсу Фундаментальная информатика

Студент группы: **М8О-101Б-22, Чапалда Мария Олеговна**, № по списку: **24**, Контакты **mariyaaach03@gmail.com** Работа выполнена: « » \_\_\_\_\_\_\_\_\_\_\_\_202\_\_\_г.

Преподаватель: **каф. 806 Крылов Сергей Сергеевич**, Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет сдан « » \_\_\_\_\_\_\_\_\_202 \_\_ г., итоговая оценка \_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Тема: Битовые множества в языке Си**

**Цель работы** Составить программу на языке Си ,которая проверяет наличие/отсутствие среди входных слов слова, все согласные буквы которого глухие

Задание (*вариант №* **24**)**:** Есть ли слово, согласные которого глухие

4 **Оборудование**

**Оборудование *ПЭВМ студента, если использовалось****:* **16 Гб оперативной памяти, экран диагональю 17.3", с разрешением экрана 1920x1080 Пикс**

Процессор  **AMD Ryzen 7 3750H with Radeon Graphics 3.20 GHz**

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_AMD Ryzen 7 2700\_ с ОП 16 Гб НМД \_\_\_5\_\_ Тб. Монитор 1920x1080~60Hz

Другие устройства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Алгоритм:

1. Напишем функцию **set\_symbol()**, создающую битовые множества - числа типа undigned long long, состоящие из цифр 0 и 1, причем 1 ставится в номер разряда числа, соответствующего номеру буквы в алфавите.
2. Так как на вход могут поступать и заглавные, и строчные буквы, для простоты будем приводить все заглавные буквы к строчному виду. Для этого напишем функцию **to\_lowercase()**
3. Напишем функцию **is\_alpha()**, которая проверяет, является ли символ разделителем.
4. Из двух массивов (в первом находятся звонкие согласные, во втором - все остальные согласные) составим 2 битсета; создадим пустой битсет.
5. Напишем функцию, которая проверяет слово согласно условию (возвращает 0 или 1).
6. Используя метод конечных автоматов и описанные выше функции, проверим поток ввода согласно условию.

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

1. Составил алгоритм, описан в пункте 6
2. Реализовать его в программе
3. Провести тесты

*Пункты 1-7 отчета составляются строго до начала лабораторной работы*

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

Код программы:

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#define set unsigned long long

set son\_set = 0;

set not\_set = 0;

set lu = 1;

set set\_symbol(set s, char c)

{

return s | (lu << ((set) c - (set) 'a'));

}

char to\_lowercase(char c)

{

if (c >= 'A' && c <= 'Z') {

return c - 'A' + 'a';

} else {

return c;

}

}

bool is\_alpha(char c)

{

c = to\_lowercase(c);

if (c == ' ' || c == '\t' || c == '\n' || c == EOF || c == ',' || c == '\r') {

return false;

}

return true;

}

int is\_son(set n\_set)

{

set pr = 0;

if ((n\_set & not\_set) == pr) {

return 1;

} else {

return 0;

}

}

int main(void)

{

char son[] = {'b', 'v', 'g', 'd', 'z'};

char notson[] = {'c', 'f', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'x', 'y', 'w'};

for (int i = 0; i < 5; i++) {

son\_set = set\_symbol(son\_set, son[i]);

}

for (int i = 0; i < 16; i++) {

not\_set = set\_symbol(not\_set, notson[i]);

}

set n\_set = 0;

char c;

bool result = false, flag = true, empt = true , iseof = false;

int state = 0;

while (flag) {

c = getchar();

c = to\_lowercase(c);

switch (state) {

case 0:

if (c == EOF) {

iseof = true;

} else {

state = 1;

}

break;

case 1:

if (is\_alpha(c)) {

state = 2;

n\_set = set\_symbol(n\_set, c);

} else {

state = 1;

}

break;

case 2:

empt = false;

if (is\_alpha(c)) {

n\_set = set\_symbol(n\_set, c);

} else {

result = result || is\_son(n\_set);

n\_set = 0;

state = 1;

}

break;

}

if (c == EOF) {

flag = false;

}

}

if ((result || empt) && !(iseof)) {

printf("Yes\n");

return 0;

}

printf("No\n");

return 0;

}

**Тесты:**

|  |  |
| --- | --- |
| Ввод | Вывод |
| alter ego | Yes |
| anathemus | No |
| EOF | No |
| \n EOF | Yes |

**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. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Выводы**: В ходе выполнения работы я познакомилась с битовыми множествами в языке Си.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_