**Отчет по лабораторной работе №13** по курсу\_1\_\_\_\_\_

|  |  |
| --- | --- |
|  | Студент группы М80-105Б-21 Жилов Андрей, № по списку 4  Контакты www, e-mail: [klzxrcn3692@outlook.com](mailto:klzxrcn3692@outlook.com)\_\_\_\_\_\_\_\_\_\_\_  Работа выполнена: «2» декабря 2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Преподаватель: В. К. Титов каф. 806\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Отчет сдан «3» декабря 2021 г., итоговая оценка \_\_\_\_\_\_\_\_\_  Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

1. **Тема:** Множества, массивы, хедер-файлы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. **Цель работы:** Составить программу на языке Си, выполняющую анализ и обработку вводимого текста в соответствии с выданным преподавателем вариантом задания.
3. **Задание:** (*Вариант №4*): Вывести количество слов с повторяющимися буквами(Регистр не важен), максимальное количество повторяющихся букв в одном слове и последнее встречающееся слово с максимальным количеством повторяющихся букв, считается что слова не превышают в длинну 256 букв
4. **Оборудование(лабораторное)**:

ЭВМ -, процессор -, имя узла сети - с ОП - ГБ,

НМД - ГБ, терминал- адрес -, принтер –

Другие устройства –

*Оборудование ПЭВМ студента, если использовалось:*

Процессор AMD Ryzen 3500U, с ОП 8 ГБ, НМД 256 ГБ. Монитор 1920x1080 пикс.

Другие устройства –

1. **Программное обеспечение:**

Операционная система семейства -, наименование - версия –

интерпретатор команд – версия

Система программирования - версия –

Редактор текстов - версия –

Утилиты операционной системы –

Прикладные системы и программы –

Местонахождение и имена файлов программ и данных -

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства GNU/Linux, наименование Kali версия 5.10.0

интерпретатор команд Bash версия 5.1.8(1)-release

Система программирования \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_версия\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

редактор текстов emacs версия 27.1

Утилиты операционной системы ls, cat, date, cal, who, whoami, uname, ruptime, uptime, tty,hostname, pwd, cd,mkdir,

rmdir, cp, mv, rm, whatis, whereis, chmod, sudo, lsblk, df, free, ps, echo.

Прикладные системы и программы –

Местонахождение и имена файлов программ и данных -

1. **Идея, метод, алгоритм**  решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок,

таблица] или формальные спецификации с пред- и постусловиями)

Считываем символы, пока не закончится ввод. Если символ относится к английским буквам то сначала копирум ее в массив Word, затем если буква заглавная то меняем ее на маленькую(регистр нам не важен) далее проверяем является ли пустым пересечение множества использованных символов и данного, если оно не пустое то добавляем символ в множество повторяющихся символов и прибавляем 1 к количеству повторяющихся символов в данном слове, после этого добавляем символ в множество использованных символов(вне зависимости от условия выше).

Если же символ не относится к алфавиту то мы считаем количество различных повторяющихся букв и их количество добавляем к количеству повторяющихся букв в данном слове(т к иначе наша программа не засчитала бы первый символ за повторяющийся), затем если в слове есть повторяются буквы то добавляем к количеству нужных нам слов 1, а если еще количество повторяющихся букв в данном слове больше максимального, то данное слово становится словом с максимальным количеством повторяющихся букв, затем мы все подчищаем и переходим к следующему слову

В конце выводим получившийся результат.

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

**Хедер-файл(Set.h):**

// for program lab13.cpp

// file Set.h

// --------------------------------------------------

#define N 26

char A='A';

char Z='Z';

char a='a';

char z='z';

char Alph[2\*N]; // alphabet: small and capital letters

int Letter(char c)

{ return c>=a&&c<=z||c>=A&&c<=Z; }

void CreateSet(char W[], char S[])

{ char c; int i;

for(i=0;i<N;i++)S[i]=S[N+i]=0;

for(i=0;c=W[i];i++)

if(c>=a&&c<=z) S[c-a]=1; else

if(c>=A&&c<=Z) S[N+c-A]=1;

}

void DisplaySet(char S[])

{ int i;

Alph[0]=a; Alph[N]=A; Alph[2\*N]=0; // alphabet fills up

for(i=1;i<N;i++)

{ Alph[i]=Alph[i-1]+1;

Alph[i+N]=Alph[i+N-1]+1; } // --"--

printf("\n%s\n",Alph); // and prints

for(i=0;i<2\*N;i++)

printf("%1d",S[i]); printf("\n"); // set prints

}

int EqualSet(char S1[], char S2[])

{ for(int i=0;i<2\*N;i++)

if(S1[i]!=S2[i]) return 0; return 1;

}

void UnionSet(char S1[], char S2[], char S3[])

{ for(int i=0;i<2\*N;i++) S3[i]=S1[i]||S2[i]; }

void IntersSet(char S1[], char S2[], char S3[])

{ for(int i=0;i<2\*N;i++) S3[i]=S1[i]&&S2[i]; }

int IncludeSet(char S1[], char S2[])

{ for(int i=0;i<2\*N;i++)

if(S1[i]==1&&S2[i]==0) return 0; return 1;

}

int EmptySet(char S[])

{ for(int i=0;i<2\*N;i++)

if(S[i]==1) return 0; return 1;

}

void InSet(char c, char S[]) // add to set

{

if(Letter(c))

if(c<=z&&c>=a) S[c-a]=1; else S[c+N-A]=1;

}

**Основная программа:**

#include <stdio.h>

#include "Set.h"

#define maxSizeOfWord 256

#define N 26

void CleenSet(char S[])

{ char c; int i;

for(i=0;i<N;i++)S[i]=S[N+i]=0;

}

int main()

{ char Word[maxSizeOfWord],Word\_with\_max\_repetition[maxSizeOfWord];

char Repetition\_Leters[N\*2];

char Used\_Letters[N\*2];

char Letter\_Set[N\*2];

int countOfrepetition=0,maxCountOfrepetition=0;

int final=0;

char c;

int r=0;

CleenSet(Used\_Letters);

CleenSet(Repetition\_Leters);

while(c!=EOF)

{

c=getchar();

if(!Letter(c))

{

for(char i='a';i<='z';++i)

{

CleenSet(Letter\_Set);

InSet(i,Letter\_Set);

IntersSet(Repetition\_Leters,Letter\_Set,Letter\_Set);

if(!EmptySet(Letter\_Set))

countOfrepetition+=1;

}

if(countOfrepetition>0)

{

final+=1;

if(maxCountOfrepetition<=countOfrepetition)

{

maxCountOfrepetition=countOfrepetition;

for(int i=0;i<maxSizeOfWord;++i)

{

Word\_with\_max\_repetition[i]=Word[i];

Word[i]=0;

}

}

}

countOfrepetition=0;

CleenSet(Used\_Letters);

CleenSet(Repetition\_Leters);

r=0;

continue;

}

Word[r]=c;

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

c=c-('A'-'a');

CleenSet(Letter\_Set);

InSet(c,Letter\_Set);

IntersSet(Used\_Letters,Letter\_Set,Letter\_Set);

if(!EmptySet(Letter\_Set))

{

InSet(c,Repetition\_Leters);

countOfrepetition+=1;

}

InSet(c,Used\_Letters);

++r;

}

printf("Максимальное количество повторений в одном слове %d\n",maxCountOfrepetition);

printf("Слово с максимальным количеством повторений: %s\n",Word\_with\_max\_repetition);

printf("Всего слов с повторениями: %d\n",final);

}

**Тесты:**

**test:**

ooo tttt adfdsafasdfasdfasdfa hh gzxcvbnm hh ff dd Gg HhHhFfDdSsRrrrrrrrReeEEEEWWWw

**Ответ:**

Максимальное количество повторений в одном слове: 29

Слово с максимальным количеством повторений: HhHhFfDdSsRrrrrrrrReeEEEEWWWw

Всего слов с повторениями: 9

**test1:**

aAA ll tt y Hh lkcbd

**Ответ:**

Максимальное количество повторений в одном слове: 3

Слово с максимальным количеством повторений: aAA

Всего слов с повторениями: 4

**test2:**

this test two, hello world, happy new year

**Ответ:**

Максимальное количество повторений в одном слове: 2

Слово с максимальным количеством повторений: happy

Всего слов с повторениями: 3

**test3:**

word1,word2,wword3,worrd4

**Ответ:**

Максимальное количество повторений в одном слове 2

Слово с максимальным количеством повторений: worrd

Всего слов с повторениями: 2

*Пункты 1-7 отчета составляются сторого до начала лабораторной работы.*

*Допущен к выполнению работы.*  **Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ cat lab13.txt

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Лаболаторная работа # 10 \*

\* Множества, массивы, хедер-файлы \*

\* Выполнил студент группы \*

\* М8О-105Б-21 \*

\* Жилов Андрей Алексеевич \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ cat Set.h

// for program lab13.cpp

// file Set.h

// --------------------------------------------------

#define N 26

char A='A';

char Z='Z';

char a='a';

char z='z';

char Alph[2\*N]; // alphabet: small and capital letters

int Letter(char c)

{ return c>=a&&c<=z||c>=A&&c<=Z; }

void CreateSet(char W[], char S[])

{ char c; int i;

for(i=0;i<N;i++)S[i]=S[N+i]=0;

for(i=0;c=W[i];i++)

if(c>=a&&c<=z) S[c-a]=1; else

if(c>=A&&c<=Z) S[N+c-A]=1;

}

void DisplaySet(char S[])

{ int i;

Alph[0]=a; Alph[N]=A; Alph[2\*N]=0; // alphabet fills up

for(i=1;i<N;i++)

{ Alph[i]=Alph[i-1]+1;

Alph[i+N]=Alph[i+N-1]+1; } // --"--

printf("\n%s\n",Alph); // and prints

for(i=0;i<2\*N;i++)

printf("%1d",S[i]); printf("\n"); // set prints

}

int EqualSet(char S1[], char S2[])

{ for(int i=0;i<2\*N;i++)

if(S1[i]!=S2[i]) return 0; return 1;

}

void UnionSet(char S1[], char S2[], char S3[])

{ for(int i=0;i<2\*N;i++) S3[i]=S1[i]||S2[i]; }

void IntersSet(char S1[], char S2[], char S3[])

{ for(int i=0;i<2\*N;i++) S3[i]=S1[i]&&S2[i]; }

int IncludeSet(char S1[], char S2[])

{ for(int i=0;i<2\*N;i++)

if(S1[i]==1&&S2[i]==0) return 0; return 1;

}

int EmptySet(char S[])

{ for(int i=0;i<2\*N;i++)

if(S[i]==1) return 0; return 1;

}

void InSet(char c, char S[]) // add to set

{

if(Letter(c))

if(c<=z&&c>=a) S[c-a]=1; else S[c+N-A]=1;

}

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ cat progav3.c

#include <stdio.h>

#include "Set.h"

#define maxSizeOfWord 256

#define N 26

void CleenSet(char S[])

{ char c; int i;

for(i=0;i<N;i++)S[i]=S[N+i]=0;

}

int main()

{ char Word[maxSizeOfWord],Word\_with\_max\_repetition[maxSizeOfWord];

char Repetition\_Leters[N\*2];

char Used\_Letters[N\*2];

char Letter\_Set[N\*2];

int countOfrepetition=0,maxCountOfrepetition=0;

int final=0;

char c;

int r=0;

CleenSet(Used\_Letters);

CleenSet(Repetition\_Leters);

while(c!=EOF)

{

c=getchar();

if(!Letter(c))

{

for(char i='a';i<='z';++i)

{

CleenSet(Letter\_Set);

InSet(i,Letter\_Set);

IntersSet(Repetition\_Leters,Letter\_Set,Letter\_Set);

if(!EmptySet(Letter\_Set))

countOfrepetition+=1;

}

if(countOfrepetition>0)

{

final+=1;

if(maxCountOfrepetition<=countOfrepetition)

{

maxCountOfrepetition=countOfrepetition;

for(int i=0;i<maxSizeOfWord;++i)

{

Word\_with\_max\_repetition[i]=Word[i];

Word[i]=0;

}

}

}

countOfrepetition=0;

CleenSet(Used\_Letters);

CleenSet(Repetition\_Leters);

r=0;

continue;

}

Word[r]=c;

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

c=c-('A'-'a');

CleenSet(Letter\_Set);

InSet(c,Letter\_Set);

IntersSet(Used\_Letters,Letter\_Set,Letter\_Set);

if(!EmptySet(Letter\_Set))

{

InSet(c,Repetition\_Leters);

countOfrepetition+=1;

}

InSet(c,Used\_Letters);

++r;

}

printf("Максимальное количество повторений в одном слове %d\n",maxCountOfrepetition);

printf("Слово с максимальным количеством повторений: %s\n",Word\_with\_max\_repetition);

printf("Всего слов с повторениями: %d\n",final);

}

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ gcc progav3.c

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ cat test

ooo tttt adfdsafasdfasdfasdfa hh gzxcvbnm hh ff dd Gg HhHhFfDdSsRrrrrrrrReeEEEEWWWw

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ ./a.out < test

Максимальное количество повторений в одном слове 29

Слово с максимальным количеством повторений: HhHhFfDdSsRrrrrrrrReeEEEEWWWw

Всего слов с повторениями: 9

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ cat test1

aAA ll tt y Hh lkcbd

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ ./a.out < test1

Максимальное количество повторений в одном слове 3

Слово с максимальным количеством повторений: aAA

Всего слов с повторениями: 4

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ cat test2

this test two, hello world, happy new year

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ ./a.out < test2

Максимальное количество повторений в одном слове 2

Слово с максимальным количеством повторений: happy

Всего слов с повторениями: 3

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ cat test3

word1,word2,wword3,worrd4

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/lr13]

└─$ ./a.out < test3

Максимальное количество повторений в одном слове 2

Слово с максимальным количеством повторений: worrd

Всего слов с повторениями: 2

1. **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 |
|  | дом | 1.12.21 | 12:00 | В массиве Letter\_Set был мусор из-за которого не работала программа | Создал функцию CleanSet которая очищала множества от мусора |  |

1. **Замечания автора по существу работы:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Выводы**:

В результате выполнения данной лабораторной работы я научился работать с хедер файлами, множествами в языке Си\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Недочеты при выполнении задания могут быть устранены следующим образом:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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