

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**КУРСОВАЯ РАБОТА**  
**по дисциплине «Программирование»**  
**Тема: Обработка текста**

Студент гр. 1304

\_\_\_\_\_

Басыров В.А.

Преподаватель

\_\_\_\_\_

Чайка К.В.

Санкт-Петербург

2021

## **ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ (КУРСОВОЙ ПРОЕКТ)**

Студент Басыров В.А.

Группа 1304

Тема работы : Обработка текста

Исходные данные:

Консоль, язык программирования Си, вводится текст, пользователю предлагается набор из 4 действий, которыми он может обработать текст. В любой момент времени пользователь может завершить работу программы.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Работа программы»

«Сборка программы»

«Тестирование»

«Инструкция по использованию»

«Заключение»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 18.12.2021

Дата защиты реферата: 21.12.2021

Студент

Басыров В.А.

## ПРИМЕР РАБОТЫ ПРОГРАММЫ.

```
vova@vova-VivoBook-ASUSLaptop-X515JA-A516JA: ~/Vova/kurc1$ ./menu
Введите Текст:
Чтобы закончить ввод текста нажмите 2 раза подряд Enter
Всегда волнительно тестировать программу. Ведь каждый тест может оказаться провальным.
Но в этом и есть основа правильности. Уверенности. Я готов начать. Готов. ГОТОВ.

Ваш текст после удаления повторных предложений
Всегда волнительно тестировать программу.
Ведь каждый тест может оказаться провальным.
Но в этом и есть основа правильности.
Уверенности.
Я готов начать.
Готов.
Введите 0, если хотите завершить программу
Введите 1, если хотите вывести все предложения, в которых каждое слово удовлетворяет
введенной строке-условию.
Введите 2, если хотите отсортировать предложения по средней длине слов в предложении
(по возрастанию).
Введите 3, если хотите преобразовать предложения так, чтобы слова располагались в по
рядке уменьшения длины слова.
Введите 4, если хотите удалить все предложения, в котором больше 5 или меньше 2 слов
.
Ваша команда-1
```

Рис 1

```
Ваша команда-1
Ваша маска-*ь
тестировать Ведь есть начать
Ваша команда-2
Я готов начать.
Но в этом и есть основа правильности.
Готов.
Ведь каждый тест может оказаться провальным.
Всегда волнительно тестировать программу.
Уверенности.
Ваша команда-3
начать готов Я.
правильности основа этом есть Но в и.
Готов.
провальным оказаться каждый может Ведь тест.
волнительно тестировать программу Всегда.
Уверенности.
Ваша команда-4
начать готов Я.
волнительно тестировать программу Всегда.
Ваша команда-0
vova@vova-VivoBook-ASUSLaptop-X515JA-A516JA: ~/Vova/kurc1$
```

Рис 2

```
Терминал
C6, 18 декабря 21:18
vova@vova-VivoBook-ASUSLaptop-X515JA-A516JA: ~/Vova/kurc1
decide.c decide.o main.o menu print_and_input.h
vova@vova-VivoBook-ASUSLaptop-X515JA-A516JA:~/Vova/kurc1$ ./menu
Введите Текст:
Чтобы закончить ввод текста нажмите 2 раза подряд Enter
Готов.

Ваш текст после удаления повторных предложений
Готов.
Введите 0,если хотите завершить программу
Введите 1,если хотите вывести все предложения, в которых каждое слово удовлетворяет
введенной строке-условию.
Введите 2,если хотите отсортировать предложения по средней длине слов в предложении
(по возрастанию).
Введите 3,если хотите преобразовать предложения так, чтобы слова располагались в по
рядке уменьшения длины слова.
Введите 4,если хотите удалить все предложения, в котором больше 5 или меньше 2 слов
.
Ваша команда-4
Весь ваш текст удален,дальнейшая обработка бессмысленна,нажмите 0,чтобы выйти из п
рограммы
Ваша команда-0
vova@vova-VivoBook-ASUSLaptop-X515JA-A516JA:~/Vova/kurc1$
```

Рис 3

```
Терминал
C6, 18 декабря 21:20
vova@vova-VivoBook-ASUSLaptop-X515JA-A516JA: ~/Vova/kurc1
vova@vova-VivoBook-ASUSLaptop-X515JA-A516JA:~/Vova/kurc1$ ./menu
Введите Текст:
Чтобы закончить ввод текста нажмите 2 раза подряд Enter
Готов к проверке.
Тогда начинаем.
Начинай.

Ваш текст после удаления повторных предложений
Готов к проверке.
Тогда начинаем.
Начинай.
Введите 0,если хотите завершить программу
Введите 1,если хотите вывести все предложения, в которых каждое слово удовлетворяет введенной строке-условию.
Введите 2,если хотите отсортировать предложения по средней длине слов в предложении(по возрастанию).
Введите 3,если хотите преобразовать предложения так, чтобы слова располагались в порядке уменьшения длины слова.
Введите 4,если хотите удалить все предложения, в котором больше 5 или меньше 2 слов.
Ваша команда-2p
Ваша команда не распознана
Ваша команда-2
Готов к проверке.
Тогда начинаем.
Начинай.
Ваша команда-3
проверке Готов к.
начинаем Тогда.
Начинай.
Ваша команда-0
vova@vova-VivoBook-ASUSLaptop-X515JA-A516JA:~/Vova/kurc1$
```

Рис 4

## **АННОТАЦИЯ**

Разработана программа, в которой пользователю предлагается ввести текст(кириллицей или латиницей) и выбрать одно из действий: вывести все слова по маске, удалить все повторные предложения или в которых встречается больше 5 или меньше 2 слов, отсортировать по средней длине слов в предложении, отсортировать предложение по длине слова. Выполнение программы осуществлено с использованием стандартных библиотек языка Си.

## **SUMMARY**

A program has been developed in which the user is prompted to enter text (in Cyrillic or Latin) and select one of the actions: display all words by mask, delete all repeated sentences or in which there are more than 5 or less than 2 words, sort by the average length of words in a sentence, sort sentence by word length. The program is executed using the standard C libraries.

## СОДЕРЖАНИЕ

	Введение	7
1.	Работа программы	8
1.1.	Инициализация структур.	8
1.2.	Функции очищения памяти и вывода текста на экран.	8
1.3	Функции, решающие первую подзадачу	9
1.4	Функция, решающая вторую подзадачу	9
1.5	Функции, решающие третью подзадачу	9
1.6	Функция удаляющая повторные предложения и функция решения 4 подзадачи.	10
1.7	Функция main	10
2.	Сборка программы	11
3.	Тестирование	11
4.	Инструкция по использованию	12
5.	Заключение	13
6.	Список использованных источников	14
7.	Приложение А. Программа	15

## ВВЕДЕНИЕ

Целью данной работы является изучение основных управляющих конструкция языка Си, а также основных функций стандартных библиотек. Изучение структур и практика программирования. Для решение поставленных задач нам требуется.

- 1) Изучение основных теоретических положений, функций, конструкция языка Си.
- 2) Знать особенности makefile и файлов с расширением .h , .c, .o.
- 3) Создать удобный интерфейс для пользователя.
- 4) Рационально подойти к вопросу использования памяти.
- 5) Учесть всевозможные ошибки при вводе пользователем.

## **1. РАБОТА ПРОГРАММЫ.**

### **1.1. Инициализация структур.**

Для того, чтобы разработать инициализацию структуры текста, разобьем нашу задачу по частям. Сначала стоит реализовать ввод структуры предложения. Для этого нужно выделить память под строку `temp`, и посимвольно считывать введенный текст, в случае недостатка начальной памяти, следует перевыделить память. Когда нам встретится `.` или перевод строки, мы завершим считывание предложения. Далее нужно выделить динамически память под структуру предложения, в нее записать нашу строку считывания, а также количество символов (которое посчитано в цикле).

В функции `strtok_for_you` для предложения надо найти массив слов, в котором хранится размер слова. Воспользуемся функцией `wcstok`, которая делит предложения по словам, в цикле посчитаем количество предложений. Также надо посчитать общее количество слов. Так как `strtok_for_you` «испортит» строку, надо перед вызовом этой функции скопировать строку, полученную до этого, и после выполнения функции присвоить значение строки.

Заполнив структуру предложения, следует перейти к реализации структуры текста. Сделаем тоже самое для текста, как для предложения, не забывая при этом выделять и перевыделять память.

Стоит отметить, что в случае невозможности выделения памяти, функции будут возвращать либо нулевой указатель (функция `strtok_for_you()` и `readsen()`) или структуру `text`, в которой количество символов будет `-1(readText)`. В `main()` эти случаи будут отлавливаться, освобождалась память от частично заполненного текста, программа будет экстренно завершаться, выводить ошибку памяти.

### **1.2. Функции очищения памяти и вывода на экран.**

Функция `free_for_two` вспомогательная и стандартная, призванная очистить двумерный массив. Функция `free_for_text`, пробегает по предложениям



освобождает в них массив количества слов и строку. Затем освобождает массив предложений. Функция `printlist()` идет по предложением и распечатывает строки.

### **1.3 Функции, решающая 1 подзадачу.**

Нужно выводить все слова соответствующее маски. Для этого следует поделить сначала все предложения на слова, с помощью функцию `strtok_for_mask`, копировать строку ,чтобы ее не испортить, в этой строке надо разбить все на слова, каждое слово проверять на маску с помощью функции `mask_fun`(см ниже), сразу после выполнения надо очищать «плохую» строку.

В функции `mask_fun`, учитывая, что `*` может быть только одна, нужно взять нужно использовать следующее соображение: если в маске нет звездочки, то проверить длину,если длина маски и слова не равна- не рассматривать, иначе посимвольно сравнивать. Если же звездочки есть — начать сравнивать предложение- дойти до звездочки и в слове перевести сравниваемый элемент на позицию <размер маски>- <положение звездочки>, продолжив сравнивать посимвольно. Если встретился в маске `?` считать- продолжить сравнение. В случае если цикл прошелся по всем символам вывести слово- иначе не выводить.

`Strtok_for_mask` — возвращает 1 если программа отработала корректно и 0 -если нам не хватило памяти.

### **1.4 Функция, решающая 2 подзадачу.**

Задача требует отсортировать предложение по средней длине предложения. Следует Воспользоваться функцией `qsort`, которая отсортирует наш массив по средней длине. Для этого напишем функцию сравнений `str`. Так как в каждом предложении заполнена информация о количестве символов и количестве слов, количество символов , которые являются буквами это общее количество символовов минус количество слов(пробелов,запятых и точек). Надо найти среднюю длину и реализовать сравнения.

### **1.5 Функции,решающие 3 подзадачу.**

Для этой подзадачи также стоит воспользоваться в функции `qsort_for_3`, которая будет оперировать строкой. Функцией `wcstok`, все слова надо записать в двумерный массив(посредством цикла и функции `wcsncpy`) и оперировать им. Отсортировать сначала с помощью `qsort` предложение по словам, для этого написав функцию `cmp_1`, которая принимает 2 слова, высчитывает их длину, выводит их, а также функцию `cmp_2`, которая сортирует массив длины слов. Вызвать два раза `qsort` с `cmp1` и с `cmp2`. Выделить память под результирующую строку и в нее с помощью функции `wcsncpy`(копирует n символов) , копировать слово целиком. Таким образом получится наша итоговая отсортированная строка. В `main` в тексте пройдемся по предложениям, чтобы каждое предложение было отсортировано по нашему требованию.

#### **1.6 Функция удаляющая повторные предложения и функция решения 4 подзадачи.**

Для решения удаления повторных предложений в функции `removeSentence` воспользуемся функцией `wscasestr`, которая сравнивает 2 предложения без учёта регистра, если они одинаковы, последовательно очиститься массив размеров слов, строка, также все предложения в массиве предложений начиная с позиции удаленного передвинутся на одну позицию влево, если же предложения не равны ,функция получит следующее предложения. Посредством вложенного цикла таким образом сравниваются все предложения. Аналогично работает решение 4 подзадачи функция `del_sentence`, только она удаляет по другому критерию- количество слов больше 5 или меньше 2.

#### **1.7 Функция `main`.**

Это фактически меню пользователя. Здесь выводятся подсказки, стоит отметить, что переменная `a` — команда пользователя, которая считывает элемент и символ перевода строки. Если оказывается длина не равной 2 — сразу выводится ошибка, иначе с помощью оператора множественного выбора `switch` выбирается нужный нам элемент. Предусмотрен случай выхода из программы.

## 2. СБОРКА ПРОГРАММЫ

Будем использовать Makefile, в целях all зададим цель menu — бинарный файл. Чтобы его выполнить, нужно выполнить 3 файла:

- 1) main.c — подсказки и меню программы.
- 2) decide.c — функции обработки текста.
- 3) print\_and\_input.c — функции ввода и вывода

Разделив линковку и компиляцию будем выполнять цели. Также есть эти же файлы с расширением .h, в которых содержатся необходимые директивы и заголовки функций. Стоит выделить директивы pragma once, которая не позволяет повторно включить файл в итоговый файл и директиву ifndef endif, которые определяют структуры и не дает повторно включать структуры в файл.

## 3. ТЕСТИРОВАНИЕ.

Номер теста	Входные данные	Выходные Данные	Пояснения
1	гшпккр	Команда не распознана	Нет такой команды
2	1 *	Ночь Улица Фонарь Аптека Бессмысленный и тусклый свет Живи ещё хоть четверть века Все будет так Исхода нет	Выведены все слова.
3	1 *a	Улица Аптека века Исхода	Все слова, оканчивающие на a
4	2	Все будет так. Ночь. Исхода нет. Живи ещё хоть четверть века. Улица. Фонарь. Аптека. Бессмысленный и тусклый свет.	Все предложения отсортированы по возрастанию средней длины.
5	3	будет Все так. Ночь. Исхода нет. четверть Живи хоть века ещё.	Помимо сортировки 4, также отсортированы по убыванию размера слова в каждом предложении.

		Улица. Фонарь. Аптека. Бессмысленный тусклый свет и. будет Все так. Исхода нет. четверть Живи хоть века ещё. Бессмысленный тусклый свет и. Команда не распознана	Удалены предложения из 1 слова.      Нет такой команды Окончания работы программы.
6	4		
7	1шпг		
8	0		

Выше приведена таблица для последовательного тестирования следующего текста:

Ночь.Улица.Фонарь.Аптека.  
Бессмысленный и тусклый свет.  
Живи ещё хоть четверть века.  
Все будет так.Исхода нет.

#### **4.Инструкция по использованию.**

- 1)Запустите программу, введя в терминале /menu**
- 2)Введите ваш текст, по завершению нажмите 2 Enter подряд.**
- 3) Следуйте подсказкам, все команды описаны выше и в тексте, который будет выведен на экран.**
- 4) Для выхода из программы нажмите 0.**

## **5.ЗАКЛЮЧЕНИЕ**

Подводя итоги, можно сказать, что все поставленные задачи были выполнены, изучено и закреплено на практике использование стандартных библиотек и конструкций языка Си, а также изучены структуры и основы работы со сборкой программы.

## 6. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Правила оформления пояснительной записки//  
se.moevm.<http://se.moevm.info/doku.php/courses:programming:report>. (дата  
обращения 18.12.2021)

2. Функции стандартных библиотек Си.//Wikipedia.  
<https://ru.wikipedia.org/wiki/> (дата обращения 18.12.2021)

3. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство  
"Невский Диалект", 2001. 352 с. (дата обращения 18.12.2021)

## 7.ПРИЛОЖЕНИЕ А

### ПРОГРАММА.

Название файла main.c

```
#include "decide.h"
#include "print_and_input.h"
#define Size_Buf 100
int main()
{
    setlocale(LC_ALL, "");
    printf("Введите Текст:\n");
    printf("Чтобы закончить ввод текста нажмите 2 раза подряд Enter\n");
    struct text Text=readtext();
    if (Text.kol_sym== -1)
        {printf("Ошибка памяти\n");
        return 0;}
    wchar_t a[Size_Buf]=L"";
    Text=removeSentense(Text);
    printf("Ваш текст после удаления повторных предложений\n");
    printText(Text);
    printf("Введите 0,если хотите завершить программу\n");
    printf("Введите 1,если хотите вывести все предложения, в которых каждое слово удовлетворяет введенной строке-условию.\n");
    printf("Введите 2,если хотите отсортировать предложения по средней длине слов в предложении(по возрастанию).\n");
    printf("Введите 3,если хотите преобразовать предложения так, чтобы слова располагались в порядке уменьшения длины слова.\n");
    printf("Введите 4,если хотите удалить все предложения, в котором больше 5 или меньше 2 слов.\n");
    do
    {printf("Ваша команда-");
    fgetws(a,Size_Buf,stdin);
    if (wcslen(a)!=2)
    {    printf("Ваша команда не распознана\n");
        continue;}
    switch (a[0])
    {case L'0':break;
    case L'1': printf("Ваша маска-");
        wchar_t *s=malloc(sizeof(wchar_t)*100);
        if (!s)
            {printf("Ошибка памяти\n");
            free_for_text(Text);
            return 0;}
        fgetws(s,100,stdin);
        int len=wcslen(s)-1;
```

```

        s[len]='\0';
        if (!strtok_for_mask(Text,s,len))
            {printf("Ошибка памяти\n");
             free_for_text(Text);
             return 0;}
        printf("\n");
        break;
    case L'2':qsort(Text.sen,Text.kol_sen,sizeof(struct sentence *),cmp);
        break;
    case L'3':for (int i=0;i<Text.kol_sen;i++)
        {Text.sen[i]->str=qsort_for_3(*Text.sen[i]);
          if (!Text.sen[i]->str)
              {printf("Ошибка памяти\n");
               free_for_text(Text);
               return 0;}}
        break;
    case L'4':Text=del_sentence(Text);
        break;
    default:
        printf("Ваша команда не распознана\n");
    }
    if (a[0]==L'2' || a[0]==L'3' || a[0]==L'4')
        printText(Text);
    if (a[0]!='0')
        if (Text.kol_sen==0)
            printf("Весь ваш текст удален,дальнейшая обработка
бессмысленна,нажмите 0,чтобы выйти из программы\n");
        while (a[0]!='0' || wcslen(a)!=2);
        free_for_text(Text);
        return 0;
}

```

### Название файла decide.c

```

#include "decide.h"
struct text removeSentence(struct text t)
{int k=0,j=0;
 for (int i=0;i<t.kol_sen;i++)
 {    j=i+1;
     while (j<t.kol_sen)
         {wchar_t *str1=t.sen[i]->str;
          wchar_t *str2=t.sen[j]->str;
          if (!wcscasecmp(str1,str2))
              {free(t.sen[j]->word);
               free(t.sen[j]->str);

```



```

        free(t.sen[j]);
        for (int k=j;k<t.kol_sen-1;k++)
            t.sen[k]=t.sen[k+1];
        t.kol_sen--;}
    else
        j++;}}
return t;}

void free_for_two(wchar_t **s,int size)
    {for (int i=0;i<size;i++)
        free(s[i]);
    free(s);}

int mask_fun(wchar_t *str,int kol,wchar_t *mask,int size)
{
    int i=0,j=0,k=0;
    if ((size!=kol && !wcschr(mask,(wchar_t) '*')) || size>kol+1)
        return 0;
    while (j<size)
        if (mask[j]!='*' && mask[j]!='?' && str[i]!=mask[j])
            return 0;
        else
            {if (mask[j]=='*')
                {j++;
                 i=kol-size+j;//Синхронизировали элементы j и i(kol-(size-
j))
                }

            else
                {i++;
                 j++;}
            }

    return 1;}

int strtok_for_mask(struct text Text,wchar_t *mask,int size)
{
    for (int i=0;i<Text.kol_sen;i++)
        {
            Text.sen[i]->str;
            wchar_t *token;
            wchar_t *temp=malloc(sizeof(wchar_t)*(Text.sen[i]->kol+1));
            if (!temp)
                return 0;
            wcscpy(temp,Text.sen[i]->str);
            wchar_t *t=wcstok(temp,L" .",&token);
            int j=0;

```

```

        while (t)
            {if (mask_fun(t,Text.sen[i]->word[j],mask,size))
                printf("%ls ",t);
                t=wcstok(NULL,L" .",&token);
                j++;}
        free(temp);
    }
    return 1;
}

int cmp(const void *a,const void *b){
    struct sentence **first=(struct sentence **) a;
    struct sentence **second=(struct sentence **) b;
    int n1=(*first)->kol_word;
    int n2=(*second)->kol_word;
    int sum1=(*first)->kol-(*first)->kol_word,sum2=(*second)->kol-(*second)-
>kol_word;
    float sr1=(float)sum1/n1, sr2=(float) sum2/n2;
    if (sr1>sr2)
        return 1;
    else
        if (sr1<sr2)
            return -1;
    return 0;
}

int cmp_1(const void *a,const void *b){
    wchar_t **first=(wchar_t**) a;
    wchar_t **second=(wchar_t**) b;
    int n1=wcslen(*first),n2=wcslen(*second);
    if (n1>n2)
        return -1;
    else
        if (n1==n2)
            return 0;
    return 1;}

int cmp_2(const void *a,const void *b)
{
    int *first=(int*) a;
    int *second=(int*) b;
    if (*first>*second)

```

```

        return -1;
    else
        if (*first==*second)
            return 0;
    return 1;}

wchar_t * qsort_for_3(struct sentence sen)
{wchar_t *token;
wchar_t *t=wcstok(sen.str,L" .",&token);
wchar_t **temp=malloc(sizeof(wchar_t *)*(sen.kol_word));
if (!temp)
    return NULL;
int i=0,sum=0;
while (i<sen.kol_word)
    {temp[i]=malloc(sizeof(wchar_t)*(sen.word[i]+1));
    if (!temp[i])
        return NULL;
    wcscpy(temp[i],t);
    t=wcstok(NULL,L" .",&token);
    i++;}
qsort(temp,sen.kol_word,sizeof(wchar_t *),cmp_1);
qsort(sen.word,sen.kol_word,sizeof(int),cmp_2);
wchar_t *res=malloc(sizeof(wchar_t)*(sen.kol+1));
if (!res)
    return NULL;
for (int j=0;j<sen.kol_word;j++)
    {wcsncpy(res+sum,temp[j],(sen.word[j]));
    sum+=sen.word[j]+1;
    res[sum-1]=' ';
    }
res[sum-1]='.';
res[sum]='\0';
free_for_two(temp,sen.kol_word);
return res;}

struct text del_sentence(struct text t)
{
    int i=0;
    while (i<t.kol_sen)
    {
        if (t.sen[i]->kol_word>5 || t.sen[i]->kol_word<2)
            {free(t.sen[i]->word);
            free(t.sen[i]->str);
            free(t.sen[i]);
            for (int j=i;j<t.kol_sen-1;j++)
                t.sen[j]=t.sen[j+1];

```

```

        t.kol_sen--;}
    else
        i++;
}
return t;}

```

```

void free_for_text(struct text t)
{
    for(int i=0;i<t.kol_sen;i++)
    {
        free(t.sen[i]->word);
        free(t.sen[i]->str);
    }
    free(t.sen);
}

```

### Название файла print\_and\_input.c

```

#include "print_and_input.h"
int * strtok_for_you(struct sentence *sen)
{
    wchar_t *token;
    wchar_t *t=wcstok(sen->str,L" .",&token);
    while (t)
        t=wcstok(NULL,L" .",&token);
    int *temp=malloc(sizeof(int)*Size);
    if (!temp)
        return NULL;
    t=sen->str;
    int k=0,j=0,size=Size;
    for (int i=0;i<sen->kol;i++)
    {
        if (!t[i])
        {
            temp[k]=j;
            k++;
            if (k>=size)
            {
                int *temp1=realloc(temp,sizeof(int)*(size+Size));
                if (!temp1)
                {
                    free(temp);
                    return NULL;}
                size+=Size;
                temp=temp1;}
            j=0;}
        else
            j++;}
    sen->kol_word=k;
    return temp;}

```

```

struct sentence* readsen()
{
    int size=Size;
    wchar_t *temp=malloc(sizeof(wchar_t)*size);
    if (!temp)
        return NULL;
    int i=0;
    wchar_t c;
    do{
        c=fgetwc(stdin);
        if (i>=size-2)
            { wchar_t *t=realloc(temp,sizeof(wchar_t)*(size+Size));
              if (!t)
                  { free(temp);
                    return NULL;}
              size+=Size;
              temp=t;
            }
        temp[i]=c;
        i++;}
    while (c!='.' && c!='\n');
    temp[i]='\0';
    struct sentence *Sen=malloc(sizeof(struct sentence));
    if (!Sen)
        return NULL;
    Sen->str=temp;
    Sen->kol=i;
    wchar_t *temp1=malloc(sizeof(wchar_t)*size);
    if (!temp1)
        return NULL;
    wcscpy(temp1,temp);
    Sen->word=strtok_for_you(Sen);
    if (!Sen->word)
        return NULL;
    Sen->str=temp1;
    free(temp);
    return Sen;
}

```

```

struct text readtext()
{
    struct text Text;
    struct sentence **text=malloc(sizeof(struct sentence*)*Size);
    if (!(text))
        {Text.kol_sym=-1;

```

```

        return Text;}
int size=Size,i=0,count=0,n=0;
struct sentence *temp;
do{ if (size<=i+2)
        {struct sentence **t=realloc(text,sizeof(struct
sentence*)*(size+Size));
        if (!t)
                {for(int k=0;k<i;k++)
                        free(text[i]);
                free(text);
                Text.kol_sym=-1;
                return Text;}
        size+=Size;
        text=t;}
temp=readsen();
if (!temp)
        {for(int k=0;k<i;k++)
                free(text[i]);
        free(text);
        Text.kol_sym=-1;
        return Text;}
if (temp->str[0]=="\n")
        count++;
else
        {count=0;
        text[i]=temp;
        n+=temp->kol;
        i++;};
}
while (count!=2);
scanf(" ");
Text.sen=text;
Text.kol_sen=i;
Text.kol_sym=n;
return Text;
}

```

```

void printText(struct text t)
{
        for(int i=0;i<t.kol_sen;i++)
                if (wcscmp(t.sen[i]->str,(wchar_t*)"")
                        printf("%ls\n",t.sen[i]->str);
}

```

### Название файла decide.h

```
#pragma once
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <stdio.h>
#define Size 50
#ifndef __Structs__
    #define __Structs__
    struct sentence{
        wchar_t *str;
        int kol;
        int *word;
        int kol_word;};

    struct text{
        struct sentence **sen;
        int kol_sen;
        int kol_sym;};
#endif
void free_for_two(wchar_t **s,int size);
int mask_fun(wchar_t *str,int kol,wchar_t *mask,int size);
int strtok_for_mask(struct text Text,wchar_t *mask,int size);
int cmp(const void *a,const void *b);
int cmp_1(const void *a,const void *b);
int cmp_2(const void *a,const void *b);
wchar_t * qsort_for_3(struct sentence sen);
struct text del_sentence(struct text t);
void free_for_text(struct text t);
struct text removeSentence(struct text t);
```

### Название файла print\_and\_input.h

```
#pragma once
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <stdio.h>
#define Size 50
#ifndef __Structs__
    #define __Structs__
    struct sentence{
        wchar_t *str;
        int kol;
        int *word;
```

```

        int kol_word;};

    struct text{
        struct sentence **sen;
        int kol_sen;
        int kol_sym;};
#endif
int * strtok_for_you(struct sentence *sen);
struct sentence* readsen();
struct text readtext();
void printText(struct text t);

```

### Название файла Makefile

```

all:menu
menu:main.o print_and_input.o decide.o
        gcc main.o print_and_input.o decide.o -o menu
main.o:main.c print_and_input.h decide.h
        gcc -c main.c
print_and_input.o:print_and_input.c print_and_input.h
        gcc -c print_and_input.c
decide.o:decide.c decide.h
        gcc -c decide.c

```