МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА 25

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

доцент, канд. тех. наук		Е. М. Линский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

КРИСС-КРОСС

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 2352		А. А. Сметанина
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ

Задачей данной курсовой работы является разработка программы, которая по набору слов создает решетку кроссворда, обеспечивая компактность и максимальное количество пересечений.

Каждый крисс-кросс состоит из списка слов, разбитых для удобства на группы в соответствии с длиной и упорядоченных по алфавиту внутри каждой группы, а также из схемы, в которую нужно вписать слова. Схема подчиняется тому же правилу, что и в кроссворде, в местах пересечения слова имеют общую букву, однако номера отсутствуют, поскольку слова известны заранее, требуется лишь вписать их в нужные места.

Обычно в схемах крисс-кросса гораздо меньше пересечений по сравнению с кроссвордами, а не заполняемые клетки не заштриховываются, если это не приводит к путанице. Крисс-кросс всегда имеет единственное решение, в котором используются все перечисленные слова [1].

Задача имеет практический и логический смысл, так как требует решения оптимизационной задачи, т.е. компактность решётки и максимизации пересечений, это придаст интересность кроссворда.

ОПИСАНИЕ АЛГОРИТМА

Основная цель алгоритма – создание кроссворда из набора слов. Слова должны быть размещены в двумерной матрице таким образом, чтобы они пересекались в общих буквах, как это принято в кроссвордах.

Идея пересечения – слова пересекаются, если имеют одинаковую букву. Алгоритм пытается найти такие пересечения для каждого слова, чтобы они могли быть размещены либо горизонтально, либо вертикально.

Алгоритм можно разделить на несколько этапов:

1. Слова считываются из файла и записываются в массив строк «Array», затем проверяется корректность данных (все слова должны быть из одного алфавита (кириллица или латиница)), так же слова не должны содержать некорректные символы (например, лишние пробелы, запятые, точки), после этого слова сортируются по убыванию длины, удаляются дубликаты и переводятся в заглавные буквы. На Рисунок 1 представлена иллюстрация выполнения первого этапа алгоритма.

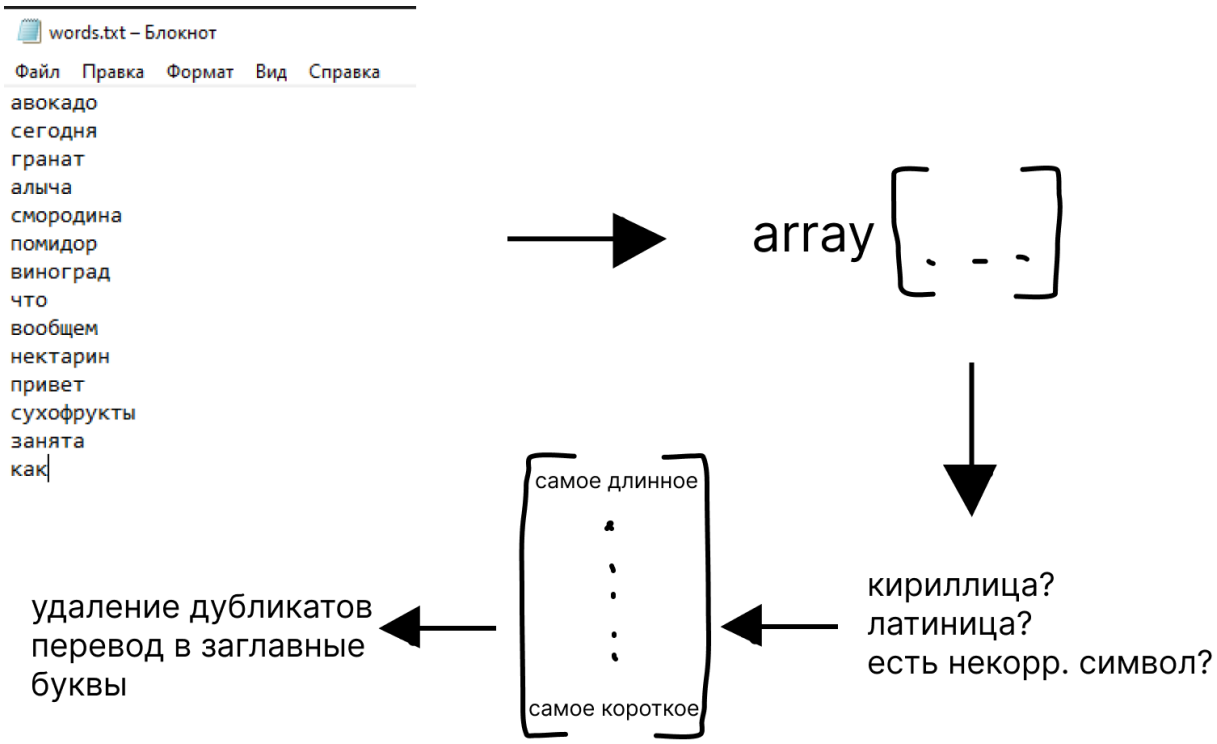


Рисунок 1 – Подготовка данных для создания решетки

2. Создается пустой двумерный массив-решетка Matrix, размер которого зависит от длины самого длинного слова, и заполняется символами пробела (Matrix_symbol). На Рисунок 2 представлена иллюстрация второго этапа алгоритма.

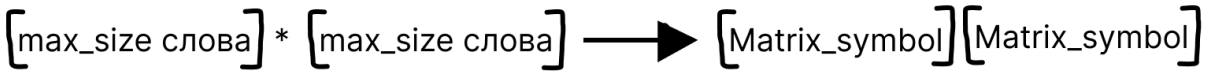


Рисунок 2 – Создание массива и заполнение

3. Первое слово размещается в центре решетки горизонтально. Для каждого следующего слова проверяются возможные пересечения с уже размещенными словами, если пересечение найдено, слово вставляется в решетку горизонтально или вертикально в зависимости от доступного места. Если слово не может быть размещено, оно пропускается. На Рисунок 3 представлена иллюстрация третьего этапа алгоритма.

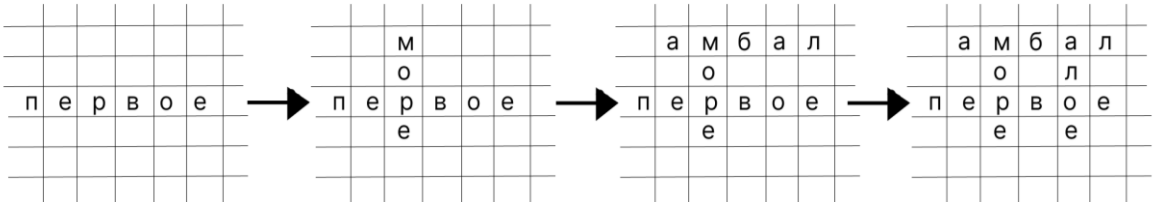


Рисунок 3 – Наглядное размещение слов

4. Удаляются пустые строки и столбцы вокруг кроссворда, чтобы уменьшить его размер.

5. Решетка-кроссворд выводится на экран через консоль и записывается в файл. На Рисунок 4 представлена иллюстрация последнего этапа алгоритма.

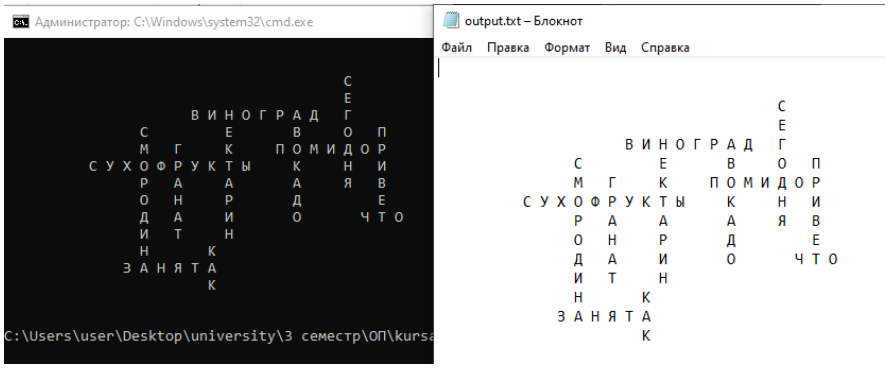


Рисунок 4 – Вывод результата

Пошаговое выполнение алгоритма с примером

Пример:

Входные слова: «синий», «красный», «мясо», «яркий».

Для начала проверим слова на корректность – все слова написаны на кириллице и нет некорректных символов. Теперь можем отсортировать по длине: «красный», «яркий», «синий», «мясо» и перевести в заглавные буквы: «КРАСНЫЙ», «ЯРКИЙ», «СИНИЙ», «МЯСО».

Создаем пустой двумерный массив по максимальной длине слова (слово «КРАСНЫЙ» является самым длинным, размер решетки = 7x7). Размещаем первое максимально длинное слово горизонтально по центру. Затем смотрим пересечение со словом «ЯРКИЙ» – по букве «К». Так проходим по всем оставшимся словам и размещаем наилучшим способом, то есть нужно убедиться, что выбранная буква в текущем слове совпадает с буквой на пересечении в уже размещенном слове, проверить, что вокруг размещаемого слова достаточно места для его размещения, а именно: в начале и конце слова не должны быть буквы других слов и вокруг размещаемого слова (по периметру) не должно быть перекрытий с другими словами, кроме пересечения. Также, если у слова есть несколько вариантов пересечения, выбираем то, где совпадает максимальное количество букв. Если несколько пересечений равнозначны, предпочтение отдается позиции, которая минимизирует пустое пространство вокруг слов.

После размещения всех слов обрезаем матрицу, т.е. удаляем пустые столбцы.

На Рисунок 5 представлена полученная решетка после выполнения всего алгоритма.

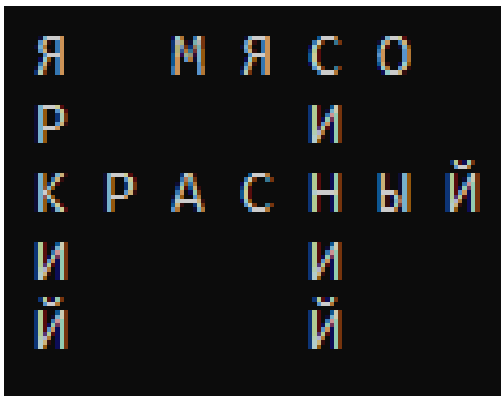


Рисунок 5 – Полученный крисс-кросс

Псевдокод:

```
//чтение слов из файла
int input_from_file(vector<string>& Array, char* words) {
     //инициализация переменных
     //открыть файл по пути file_path
    //пока файл не завершится:
        //прочитать строку из файла
        //добавить строку в Array
    //закрыть файл

     //проверка на корректность слов и минимальное количество
     //преобразование слов в заглавные буквы
}

//создание пустой решетки
char** crisscross_versions::making_matrix(vector<string>& Array)
{
     int mtx_size = max_size(Array) * max_size(Array) + 2;
     //выделение памяти

     //вернуть Matrix;
}

//проверка на пересечение двух слов
int check_words_crisscross_v2(string& Array_1, string& Array_2,
int idx_symb) {
     //инициализация переменных
     //добавление двух строк в массив

     //для каждого символа в word1:
        //если символ равен символу в word2 на позиции idx:
            //вернуть индекс пересечения
     //если строки одинаковые -1
     //если символ не найден -1
}

//нахождение наилучшего пересечения
```

```
int check_words_crisscross(string& Array_1, string& Array_2) {
     //инициализация переменных
     //выделение памяти
     //добавление двух строк в массив
     //создать список для хранения общих символов
     //для каждого символа в word1:
        //для каждого символа в word2:
            //если символы совпадают:
                //добавить символ и его индекс в список
     //если список пуст, вернуть -1

     //для поиска наиболее централизованного пересечения:
        //для каждого общего символа:
            //если символ ближе к центру слова, вернуть его
индекс
}

//проверка размещения слова по горизонтали и вертикали
(аналогично)
int find_w_begin_horizontally(char** Matrix, string Array, int
n, int m) {
     //для каждого символа в word:
        //если символ в Matrix[n][m] совпадает:
            //увеличить счетчик совпадений
     //проверка количеств совпадений с длиной строки
}

//построение кроссворда
void crisscross_versions::making_crisscross(vector<string>&
Array, char** Matrix) {
     //инициализация переменных

     //для каждого слова в Array:
        //для каждого другого слова в Array:
            //если слова не одинаковы:
                //для каждого возможного пересечения символов:
                    //найти пересечение с использованием
check_words_crisscross
                    //если пересечение найдено:
                        //если слово можно разместить
горизонтально или вертикально в матрице:
                            //разместить слово
                            //пометить слово как размещенное
                            //перейти к следующему слову
                //если слово не удалось разместить, вернуть
ошибку
//также для горизонтального размещения слова
```

Сложность алгоритма: O(n^2 + n * m).

ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

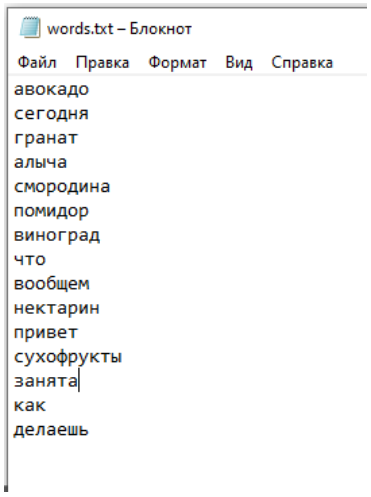
Программа запускается пользователем из командной строки. Командная строка принимает три параметра: первый – имя проекта с кодом (например, criss_cross.exe), второй – имя файла, в котором находятся исходные данные, то есть слова для составления кроссворда (например, words.txt); второй – имя файла, в который будет записан результат после выполнения программы (например: output.txt).

Входной и выходной файлы должны быть в формате .txt.

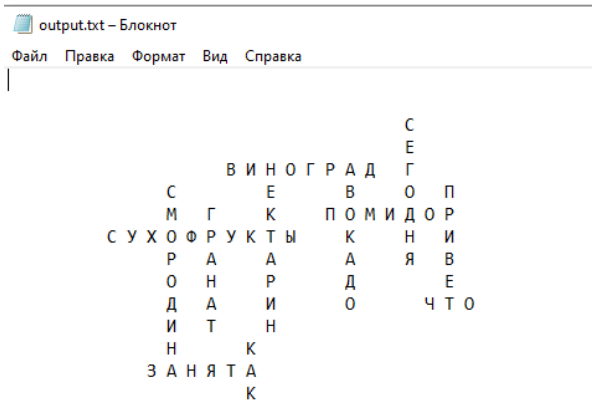
ТЕСТОВЫЕ ПРИМЕРЫ

Тест 1

Входные данные:

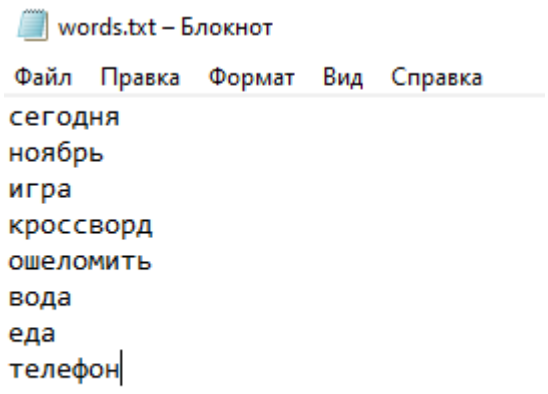


Выходные данные:

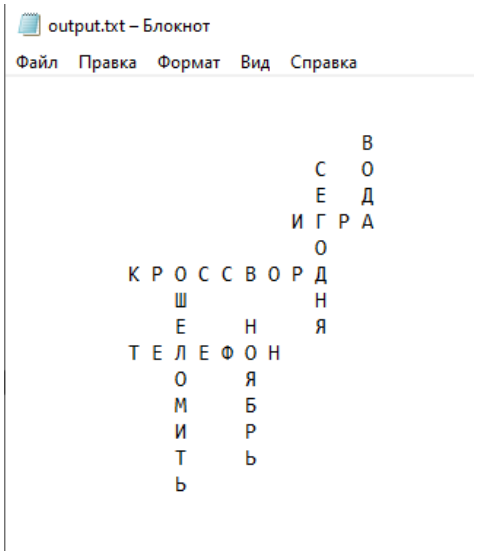


Тест 2

Входные данные:

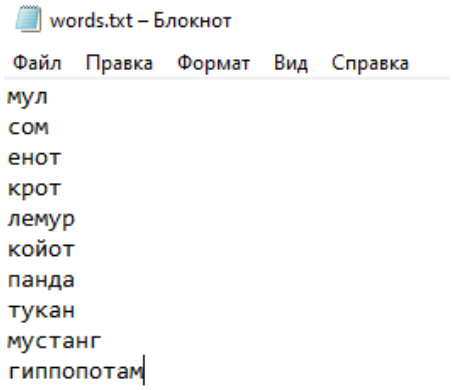


Выходные данные:

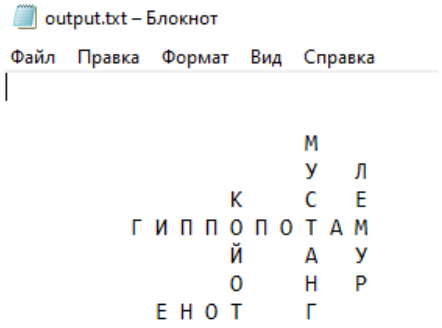


Тест 3

Входные данные:



Выходные данные:



СПИСОК ЛИТЕРАТУРЫ

1. Ч. Уэзрелл, «Этюды для программистов», 1982 г. – 288 с.

2. Методы программирования в задачах и примерах на C/C++ : учебное пособие / В. Д. Валединский, А. А. Корнев. – Москва, Издательство Московского университета, 2023. – 413 с.