МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 51

ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

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

|  |
| --- |
| ОТЧЕТ О КУРСОВОЙ РАБОТЕ |
| Крисс-кросс |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 5013 |  |  |  | Н. И. Лучкин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

# Содержание

# [1 Постановка задачи 3](#__RefHeading___Toc1200_2025493032)

## [1.1 Задание 3](#__RefHeading___Toc1527_2025493032)

## [1.2 Смысл задачи 3](#__RefHeading___Toc1529_2025493032)

# [2 Алгоритм 4](#__RefHeading___Toc1202_2025493032)

## [2.1 Идея 4](#__RefHeading___Toc1531_2025493032)

## [2.2 Описание алгоритма 4](#__RefHeading___Toc1533_2025493032)

## [2.3 Псевдокод 4](#__RefHeading___Toc1535_2025493032)

## [2.4 Анализ сложности 8](#__RefHeading___Toc1537_2025493032)

# [3 Инструкция пользователя 8](#__RefHeading___Toc1204_2025493032)

## [3.1 Запуск программы 8](#__RefHeading___Toc1539_2025493032)

## [3.2 Формат входного файла 9](#__RefHeading___Toc1541_2025493032)

## [3.3 Формат выходного файла 9](#__RefHeading___Toc1543_2025493032)

# [4 Тестовые примеры 9](#__RefHeading___Toc1206_2025493032)

## [Test1 9](#__RefHeading___Toc648_3548852760)

## [Test2 10](#__RefHeading___Toc650_3548852760)

## [Test3 11](#__RefHeading___Toc652_3548852760)

# [Список литературы 12](#__RefHeading___Toc1208_2025493032)

# Постановка задачи

## Задание

Задачей данной курсовой работы является реализовать программу, которая по набору слов создает решетку кроссворда. Решетка должна быть компактная и красивая, а также иметь максимум пересечений. На вход программа берет список слов, на выходе записывает решетку в текстовый файл.

Алгоритм используемый программой описан в [1].

## Смысл задачи

Задача имеет смысл, как создание головоломки для учебных или развлекательных целей.

# Алгоритм

## Идея

Классический перебор с возвратами. Вписывания слов в фиксированную схему, пока в списке есть подходящие слова. Когда они кончатся, возвращение на шаг назад, с удалением последнего вписанного слова, и попытка вписать другое слово.

## Описание алгоритма

Первым этапом идёт запись слов из файла с проверкой их длинны (не меньше 3 и не более 15 символов ), и проверкой на повторения. Далее слова вписываются в матрицу и сравнивается коэффицент оптимальности матрицы, и сохраняется наиболее связанная матрица. Если больше в матрицу нельзя вписать слов, удаляется последнее слово. После прохода всех итераций, возвращается матрица с наибольшим коэфицентом оптимальности.

## Псевдокод

main() {

создание матрицы (15,15);

ввод названия файла ввода;

проверка слов(файла ввода;);

ввод названия файла вывода;

обертка алгоритм крисскросс(слова, марцица);

}

создание матрицы(высота, широта) {

создать двумерный массив чисел размером высота\*ширина;

проинициализировать его 1;

создать двумерный массив символов размером высота\*ширина;

проинициализировать его „\0“;

вернуть матрица;

}

проверка слов (файл){

создать список\_слов\_и\_статусов;

while (не конец файла) {

получить новое слово;

if (размер слова < 3 || размер слова > 15) {

вывести слова как неподходящее;

continue;

}

if (слова нет в списке)

добавить слово в список;

}

задать все статусы слов равными false;

задать все координаты слов как неиспользуемые;

}

обертка алгоритма крисскросс(слова, матрица){

создать список вошедших слов;

создать матрицу проверки;

создать список сохранения вошедших слов;

алгоритм крисскросс(слова, матрица, список вошедших слов, матрица проверки, список сохранения вошедшыих слов);

матрица = матрица проверки;

вывод слов не вошедших в список сохранения слов;

замена в матрице символов „\0“ на пробелы;

сокращение размеров матрицы до оптимального значения;

вывод матрицы в файл;

}

алгоритм крисскросс(слова, матрица, список вошедших слов, матрица проверки, список сохранения вошедшыих слов){

for (а в списке слов)

if (a не использовано){

if (a можно добавить в матрицу){

добавить в матрицу(а, матрица);

номер а добавить в список вошедших слов;

алгоритм крисскросс (слова, матрица, список вошедших слов, матрица проверки, список сохранения вошедших слов);

}

}

}

if (матрица.получить\_коэффициент > матрица проверки.получить\_коэффициент){

матрица проверки = матрица;

список охранения слов = список вошедших слов;

}

удалить слово(последнее слово из списка вошедших);

}

добавить в матрицу(слово, матрица){

if (статус == true)

return;

оптимальные координаты слова;

коэффициент = матрица.получить\_коэффициент;

for (h = 0; h < высота; h++)

for(w = 0; w < ширина; w++)

добавить слово по координатам(h, w);

if (матрица.получить\_коэффициент > коэффициент){

оптимальные координаты слова = h, w;

коэффициент = матрица.получить\_коэффициент;

}

удалить слово;

if (коэффициент > матрица.получить\_коэффициент){

добавить слово по координатам(h, w);

}

}

удалить слово(слово){

if (статус == true)

return;

очистить клетки занимаемые словом если в матрице чисел клетки не делятся на used\*used;

разделить значения клеток занимаемых словом на used;

для горизонтальных слов значения клеткок сверху и снизу разделить на gorizontal;

для вертикальных слов значения клеток справа и слева разделить на vertical;

статус = false;

}

добавить слово по координатам(h, w){

if (слово не влезает || символы в матрице не совпадают с символами слова)

return;

if (ячейки для слова не делятся на gorizontal){

записать слово по координатам;

в матрице чисел умножить значения по которым находится слово на used;

в матрице чисел умножить значения над словом и под словом на gorizontal;

статус = true;

координаты слова = h, w

ориентация горизонтальная;

return;

}

if (ячейки для слова не делятся на vertical){

записать слово по координатам;

в матрице чисел умножить значения по которым находится слово на used;

в матрице чисел умножить значения над словом и под словом на vertical;

статус = true;

координаты слова = h, w

ориентация вертикальная;

return;

}

}

получить\_коэффициент(){

if (матрица пустая)

return -1;

return количество значений used \* used / (высота \* ширину);

}

## Анализ сложности

Пусть N количество слов, K количество символов, ki длина этик слов, H\*W размеры матрицы;

* Анализ сложности создания списка слов

Добавление слов имеет сложность

* Анализ сложности вставки слова по координатам

Сложность вставки слова по координатам

* Анализ сложности вставки слова

Вставка слова происходит по результатам полного перебора координат по которым можно встваить слово и имеет сложность

* Анализ сложности алгоритма крисскросс

Алгоритм крисскросс рекурсивно вызывает самого себя при этом уменьшая количество неиспользованных слов и имеет сложность =

Запись в файл имеет сложность

* Итоговая сложность

# Инструкция пользователя

## Запуск программы

Для запуска программы нужно запустить файл *run*. После запуска программы появится приглашение ввести имя файл содержащий слова далее появится приглашение ввести имя файла, в который запишется крисскросс. В ходе работа программа сначала выведет неправильные слова (неподходящие по размеру или дубликаты). Затем выведет неиспользованные слова.

## Формат входного файла

Текстовый файл.

## Формат выходного файла

Текстовый файл.

# Тестовые примеры

Тестовые примеры располагаются в папке test. Они представляют файлы содержащие списки слов и файлы содержащие заполненные кроссворды.

## Test1

* + input.txt:

abc cedder rtg gut tyq qlp auter herfatgas ketf ssfer kitg losd tower losskner

* + вывод в консоли:

Input file: input.txt

Output file: output.txt

Wrong words:

List of not used words: kitg

* + output.txt:

abc qlp

u e o h

t d ssfer

e d s r

r e ketf

rtg n a

u e tyq

tower g

a

losd

## Test2

* + input.txt:

board steel flash feel like like ab bang crash street run rare optima update system flag cashe strong

* + вывод в консоли:

Input file: input.txt

Output file: output.txt

Wrong words: like (dublicate) ab

List of not used words:

* + output.txt:

crash

a t

s feel

h e

e flash

y s

street

o r t u r

update n board

t r m n

like flag

m

bang

## Test3

* + input.txt:

hotline maiami wrong number perturbator sound young star size optimal language split killer second shock tower

* + вывод в консоли:

Input file: input.txt

Output file: output.txt

Wrong words:

List of not used words: young

* + output.txt:

shock

e p l

c t split a

o i e o n

number wrong

d a t e u

l u r star

r g

b k size

maiami o

t l u

hotline

r e d

r

# Список литературы

1. А. У. Эзрелл, “Этюды для программистов” (1982). Москва: Мир