**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский

технический университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

Кафедра систем автоматизированного проектирования

09.03.01. «Информатика и вычислительная техника»

КУРСОВАЯ РАБОТА

по дисциплине: «Методы программирования систем автоматизированного проектирования »

на тему: «Трассировка межсоеденительных элементов СБИС, на основе построения кратчайших покрывающих деревьев, с использованием первого алгоритма Штейнера»

Обучающийся 4314 \_\_\_\_\_\_\_\_\_\_\_\_ Бикмуллин А.Р.

(номер группы) (подпись, дата) (Ф.И.О.)

Руководитель старший преподаватель Суздальцев И.В.

(должность) (Ф.И.О.)

Курсовая работа (проект) зачтена (зачтен) с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

(подпись, дата)

Казань 2023

**СОДЕРЖАНИЕ**

1. Введение.......................................................................................................3
2. Содержательная постановка задачи...........................................................4
3. Математическая постановка задачи...........................................................5
4. Описание алгоритма решения задачи........................................................7
5. Решение задачи на контрольном примере...............................................11
6. Описание программы.................................................................................15
7. Примеры решения задач о нахождении минимального остовного дерева с помощью нашей программы.................................................................17
8. Оценка временной сложности алгоритма................................................24
9. Заключение.................................................................................................31
10. Источники...................................................................................................32
11. Приложения................................................................................................34

**Введение**

В данной курсовой работе проводится полный анализ реализации трассировки в СБИС с использованием первого алгоритма Штейнера для построения кратчайшего покрывающего дерева.

Выполнена содержательная постановка задачи, математическая постановка задачи, произведено описание алгоритма решения задачи и решение задачи на контрольном примере, а также была проанализирована временная сложность алгоритма.

Также мы выяснили насколько алгоритм актуален в наши дни, и где он используется.

Выполнена разработка программы, которая способна решать задачу (строить минимальное покрывающее дерево) используя в основе первый алгоритм Штейнера. Программа написана на языке разработки - С++

**2 Содержательная постановка задачи**

Исходными данными решаемой задачи являются список цепей, параметры конструкции элементов и коммутационного поля, размещение элементов на коммутационном поле.

Покрывающее дерево состоит из некоторого подмножества рёбер графа, таких, что из любой вершины графа можно попасть в любую другую вершину, двигаясь по этим рёбрам, и в нём нет циклов, то есть из любой вершины нельзя попасть в саму себя, не пройдя какое-то ребро дважды.

В неформальной форме задача о нахождении минимального покрывного дерева трактуется следующим образом: Связать все существующие в данном графе вершины так, чтобы суммарный вес связывающих ребер был минимальным. Важные условия: в получившимся графе не должно быть циклов и каждая вершина должна быть связана

Области применения:

Задача о нахождении минимального покрывного дерева используется в главной мере в решениях задач трассировки межсоединительных элементов СБИС. Нахождение кратчайшего дерева это очень важный элемент автоматизированного проектирования СБИС: на этом этапе проводится планирование всех межсоединительных связей до определения их окончательной конкретной топологии.

Также задача о нахождении кратчайшего покрывного дерева часто встречается решении логистических задач по транспортировке: допустим, есть *n* городов, которые необходимо соединить дорогами, так, чтобы можно было добраться из любого города в любой другой (напрямую или через другие города). Разрешается строить дороги между заданными парами городов и известна стоимость строительства каждой такой дороги. Требуется решить, какие именно дороги нужно строить, чтобы минимизировать общую стоимость строительства.

**3 Математическая постановка задачи**

Для математической (формальной) постановки задачи, введём следующие представленные в таблице 1.

|  |  |  |
| --- | --- | --- |
| **Обозначения элементов в математической модели задачи** | **Описание соответствующих элементов математической модели** | **Примечание** |
|  | Множество точек на плоскости, соответсвующих выводам произвольной цепи. |  |
| G(X, U) | Исходный граф (Все элементы расположенные на печатной плате) |  |
| X | Множество выводов цепи (множество элементов СБИС) |  |
| U | Множество ребер соединяющих выводы x (межсоеденительные связи между элементами) |  |
| n | Количество вершин графа (количество элементов СБИС) |  |
| dij | Вес ребра, инцидентного i-ой и j-ой вершинам графа (длина соединения между i-ым и j-ым элементами) | i = 1.. n;  j = 1.. n;  i≠j |
| xij | Элемент матрицы X | i = 1.. n;  j = 1.. n;  i≠j |
| F | Целевая функция |  |

Таб. 1. Перечень обозначений элементов в математической модели задачи.

Тогда математическая постановка задачи о построении покрывающего дерева минимальной длинны имеет следующий вид:

Покрывающее дерево D\* связного графа G называется кратчайшим, если его вес d(D\*) является наименьшим среди весов всех покрывающих деревьев графа G.

Алгоритм построения кратчайшего покрывающего дерева:

Вход: связный граф G = (X, U), |X| = n, |U| = q,

функция весов d : X → R+.

Выход: какое-то кратчайшее остовное дерево D\* = (X, U\*) графа G, U\* ⊆ U

F=

**4 Описание алгоритма решения задачи**

### **1) Общее описание алгоритма**

Задача Штейнера о минимальном [дереве](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) состоит в поиске кратчайшей сети, соединяющей заданный конечный набор точек плоскости. Задача получила своё название в честь [Якоба Штейнера](https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80,_%D0%AF%D0%BA%D0%BE%D0%B1) (1796—1863).

Альтернативное условие задачи заключается в поиске минимального подграфа, соединяющего конечное число заданных вершин (терминалов) и образующего таким образом сеть кратчайших путей между этими вершинами. Задача является таким образом обобщением задачи поиска [минимального остовного дерева](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BE%D1%81%D1%82%D0%BE%D0%B2%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE).

### **2) Математическое описание алгоритма**

Сформируем первый алгоритм Штейнера в ортогональной метрике.

1. Пусть P = {p1, 2p,…,pn} - выводы одной цепи. Построим базовую ортогональную сетку магистралей, проходящих через точки p с координатами (xn, yn)
2. Пронумеруем по часовой стрелке по спирали точки pi, из множества P, начиная с такой точки p1 для которой координата x=xmin и y = ymin/
3. Присвоим каждой точке p один из двух признаков: 0- если трасса пройдет вдоль оси x и 1- если трасса пройдет вдоль оси y. Точка p1 получает признак 0. Далее признаки присваивают всем точками попеременно, причем точки одного участка, находящиеся на одной горизонтали (вертикали) получают один признак
4. Для точки p = p1 найдем такую точку px, для которой будет справедливо r1 = |xx –x1| + |yx – y1| стремится к минимуму. Строим фрагмент дерева, присваиваем наименьший из номеров концевых точек фрагмента.
5. Всем вершинам ортогональной сетки, через которые прошел фрагмент дерева присваиваем наименьший из концевых точек фрагмента.
6. Выполним третий и четвертый пункты для всех точек p принадлежащие P в порядке возрастания их номера
7. Будем повторять с третьего по пятый пункты до тех пор пока все точки pi не получат номер, равный единице.

Важное замечание. Если же на плоскости разрешается размещение точек Штейнера, то длинну кратчайшего острова можно уменьшить соответствующим подбором таких точек.

Рассмотрим наиболее важные свойства:

1. Точка Штейнера S имеет локальную степень P(S) = 3
2. Для вершины p1 принадлежащая P степень d(p1) <=3
3. Число точек Штейнера в наикратчайшем дереве Штейнера ровно k: 0<=k<=n2, где n = |P|

### **3) Доказательство.**

Для доказательства того что задача Штейнера на графах является NP-полной нам следует:

1. Показать, что Р принадлежит классу NP

2. Выбрать известную NP-полную задачу из Р’

3. Построить преобразование f из P’ в P

4. Доказать, что f-полиномиальное преобразование

### **4) Сложность алгоритма**

[Эффективного](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_P) (сложность выражается функцией, ограниченной сверху некоторым полиномом от параметра задачи, в данном случае число вершин графа) алгоритма, дающего точное решение проблемы Штейнера, не существует при условии [неравенства классов P и NP](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B2%D0%B5%D0%BD%D1%81%D1%82%D0%B2%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2_P_%D0%B8_NP), так как проблема является [NP-полной](https://ru.wikipedia.org/wiki/NP-%D0%BF%D0%BE%D0%BB%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0).

Из этого следует, что при предположении о несовпадении классов Р и АР не существует полиномиального алгоритма решения исследуемой задачи.

Для решения мощностной задачи Штейнера на ориентированном градуированном графе следует применять весь спектр методов, использующихся для исследования АР-трудных задач:

1. Алгоритмы уменьшения размера задачи.

2. Точные алгоритмы решения задачи:

а) метод ветвей и границ,

б) метод динамического программирования.

3. Поиск классов графов с полиномиальной разрешимостью задачи.

4. Приближенные алгоритмы.

Настоящая работа представляет собой обзор полученных при этом результатов.

Блок-схема данного алгоритма показана на (рис. 1) ниже.

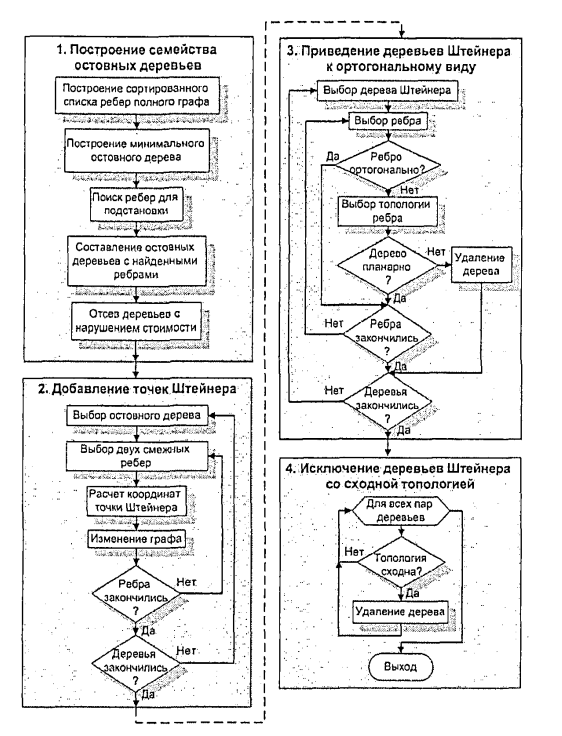


Рис.1. Блок-схема формирования деревьев штейнера в графе.

**5 Решение задачи на контрольном примере**

**Дается набор вершин (набор элементов СБИС)** (рис 2).

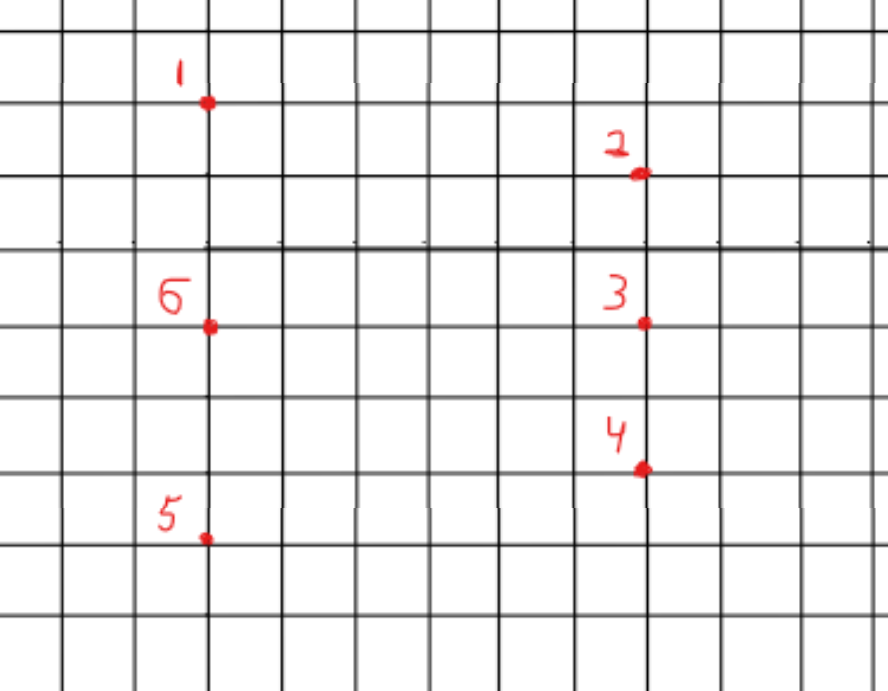


Рис. 2. Неориентированный несвязный граф .

**Решение:**

Решаем контрольный пример с помощью первого алгоритма Штейнера о минимальном дереве.

Алгоритм состоит из нескольких шагов:

1. Пронумеруем вершины по часовой стрелке по спирали и построим сетку
2. Присваиваем каждой точке индексы 0 и 1 в зависимости от горизонтальности или вертикальности трассы.
3. Рекурсивно находим и добавляем вершины в граф пока не обойдем все элементы платы, где r1 = |xx –x1| + |yx – y1| наименьшее. При необходимости добавляем точку Штейнера.

Решение:

1. Присваиваем каждой точке индексы 0 и 1 в зависимости от горизонтальности или вертикальности трассы.

(рис 3)

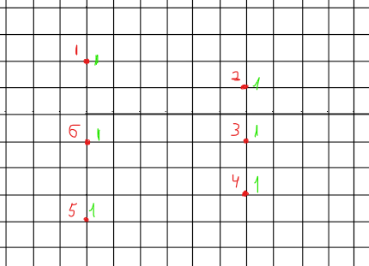


Рис. 3. Проиндексированный граф

2) Рекурсивно находим и добавляем вершины в граф, где r1 = |xx –x1| + |yx – y1| наименьшее (рис.4)

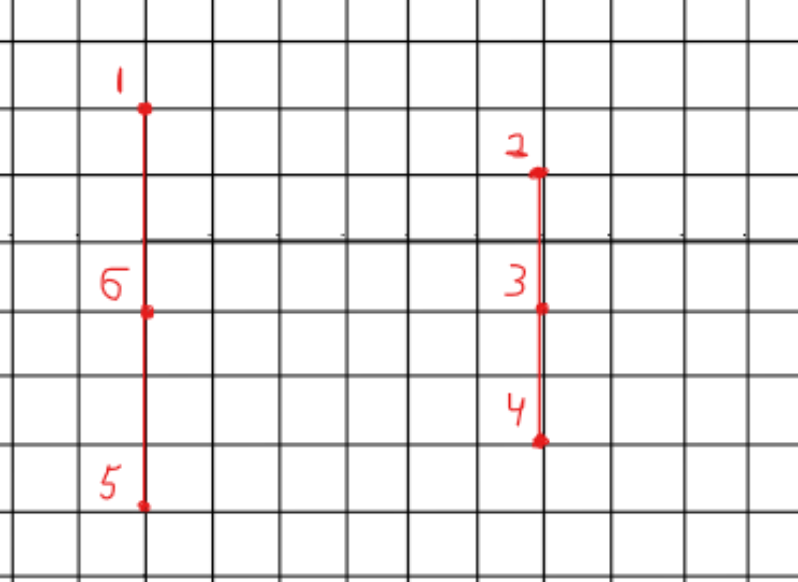


Рис. 4. Граф с соединениями

3) Проверяем на обход всех элементов платы

Так как связаны не все элементы повторяем пункт 2,3

4) Присваиваем каждой точке индексы 0 и 1 в зависимости от горизонтальности или вертикальности трассы. (рис.5)

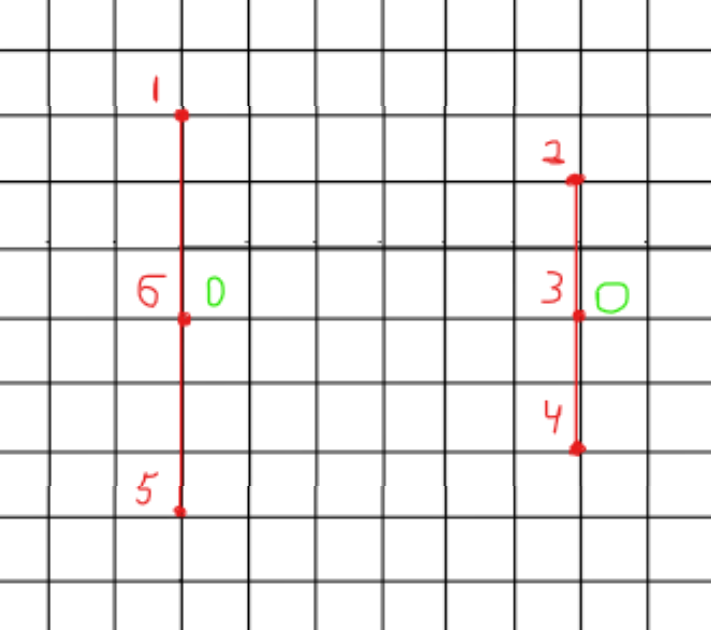


Рис. 5. Проиндексированный граф

5) Рекурсивно находим и добавляем вершины в граф, где r1 = |xx –x1| + |yx – y1| наименьшее (рис 6)

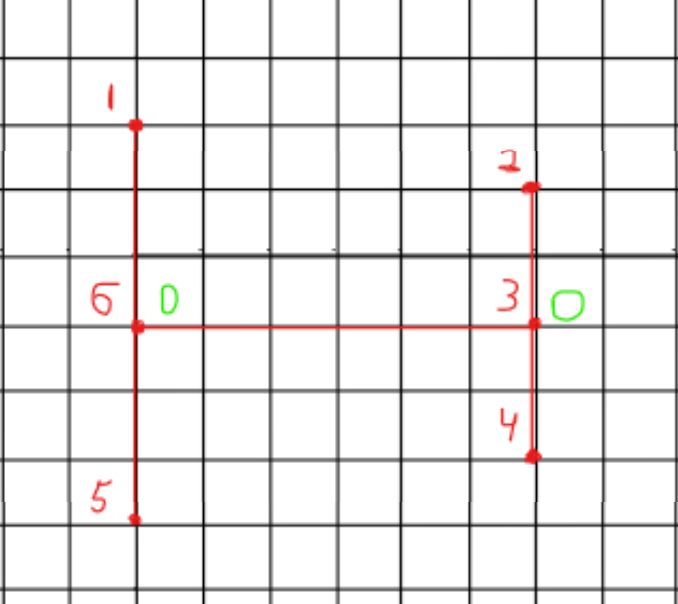


Рис. 6. Полученный граф

6 ) Проверяем на полноту графа

Граф полный!

7) Считаем вес полученного дерева

3+3+2+2+6 = 16

Задача решена! Мы нашли минимальное покрывающее дерево и его вес с помощью первого алгоритма Штейнера на графе .

**6 Описание работы программы.**

## Представим описание работы программы для решения задачи о поиске минимального покрывного дерева в виде структуры программы (с помощью UML-диаграммы классов программы).(рис. 9)

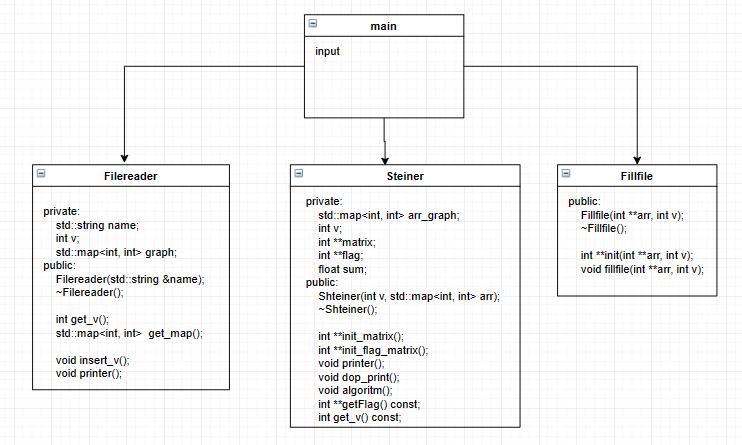
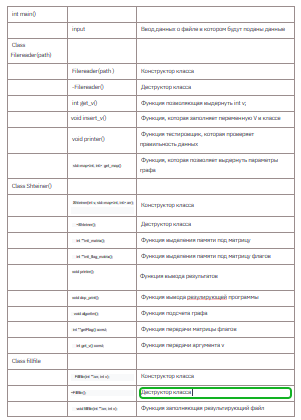


Рис. 9. UML диаграмма

## Программно-модифицированный алгоритм реализован в виде консольного приложения на языке C++. Описание классов и методов представлено ниже в таблице 2.



Таб. 2. Полное описание используемых методов

**7 Примеры решения задачи о поиске минимального остовного дерева.**

Пример №1

**Решение:**

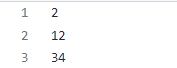
1) Подан файл с количеством вершин и набором координат 

Рис. 10 Файл

2) Запускаем программу через команду bash start.sh

3) Результат выполнения программы (рис 12):

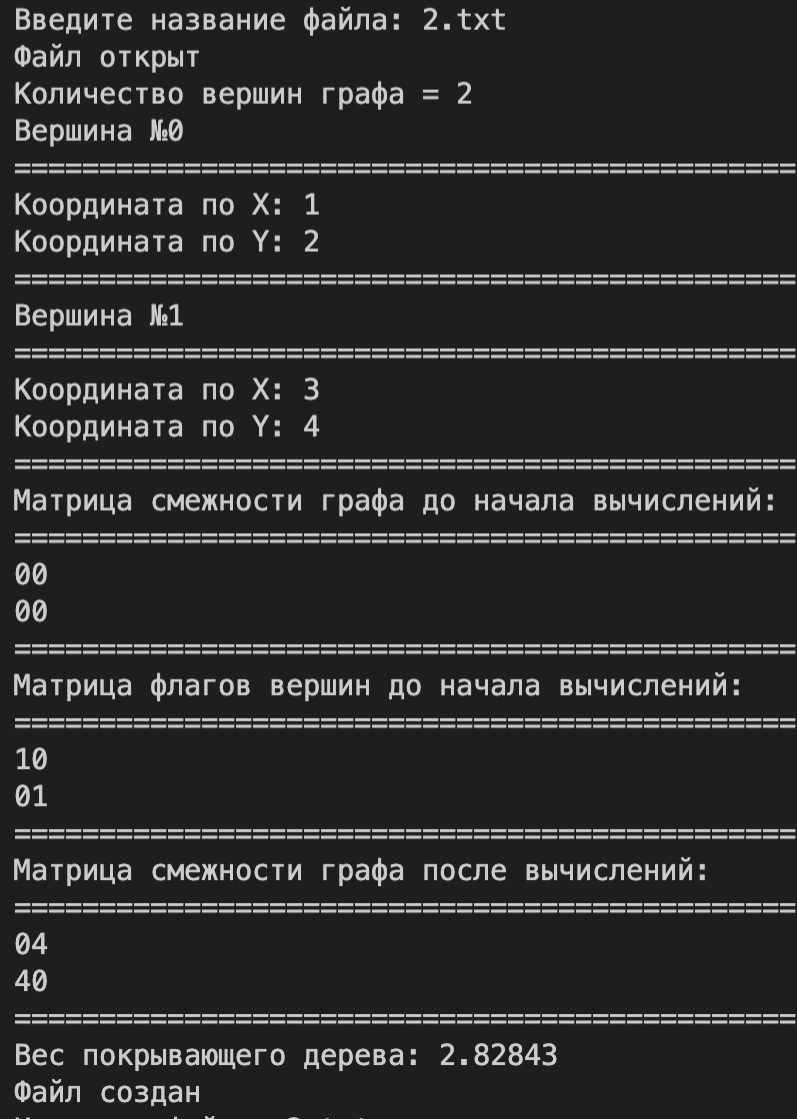


Рис. 12. Результат выполнения программы

4)Матрица построенная визуальной частью программы (рис 13):

Программа ставит на координатах i=j единицу не потому что там есть петли, а потому чтобы алгоритм работал быстрее и не обработывал лишние кустки кода.

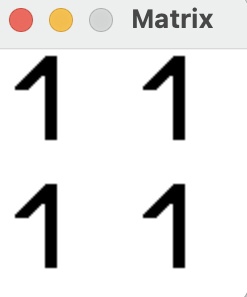


Рис. 13. Визуализированное решение программы.

Пример №2

1) Подан файл с количеством вершин и набором координат (рис 14)



Рис. 14. Файл

2) Запускаем программу через команду bash start.sh

3) Результат выполнения программы (рис 15):

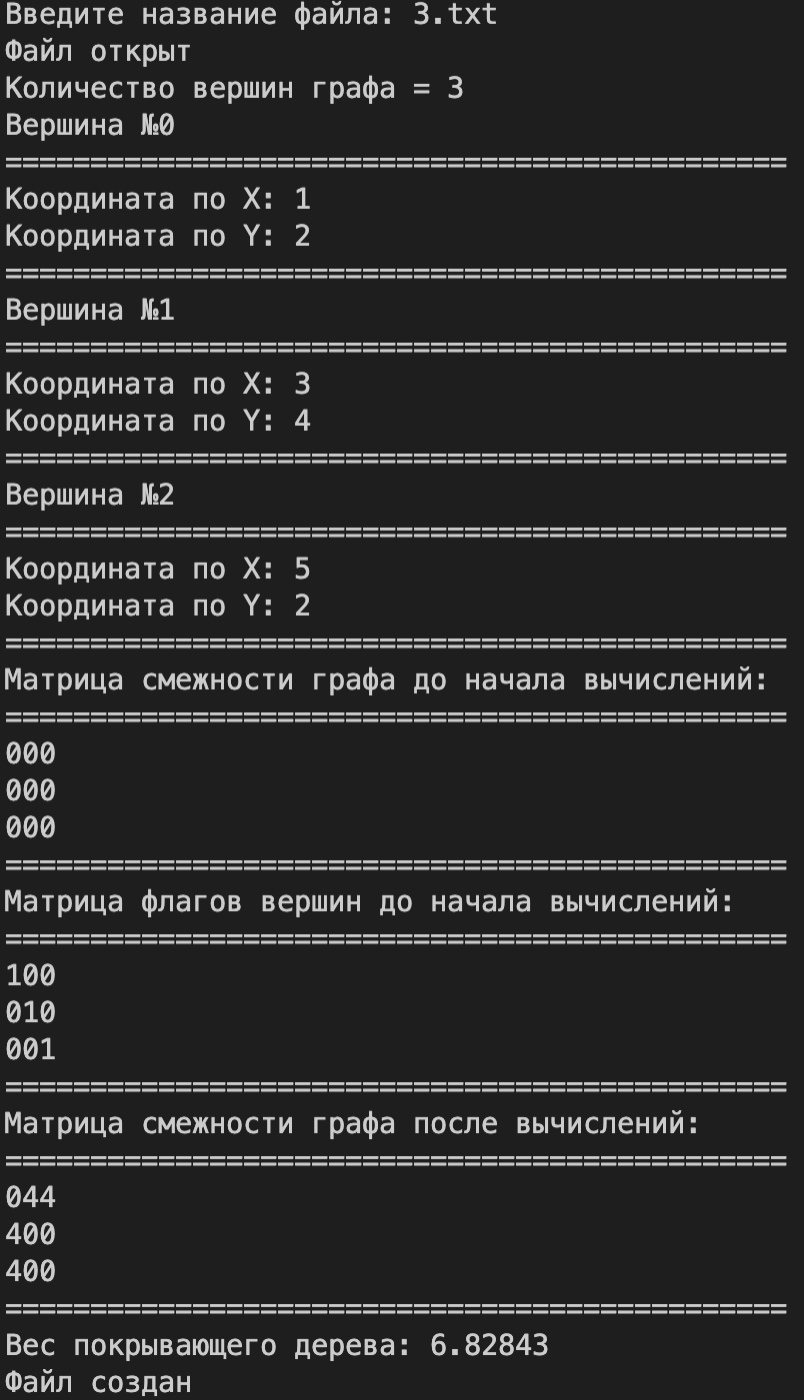


Рис. 15. Результат выполнения программы

4) Матрица построенная визуальной частью (рис 16):

Программа ставит на координатах i=j единицу не потому что там есть петли, а потому чтобы алгоритм работал быстрее и не обработывал лишние кустки кода.

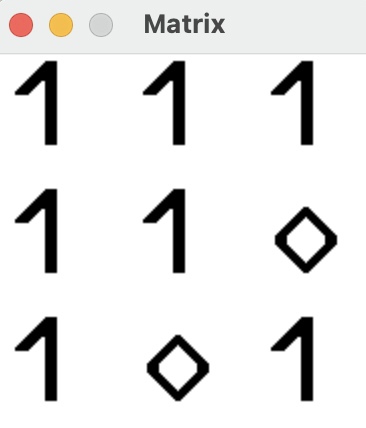


Рис. 16. Визуализированное решение программы.

Пример №3

1) Подан файл с количеством вершин и набором координат на рисунке 17.

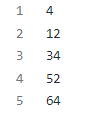


Рис. 17. Граф из варианта

2) Запускаем программу через команду bash start.sh

3) Результат выполнения программы(рис 18):

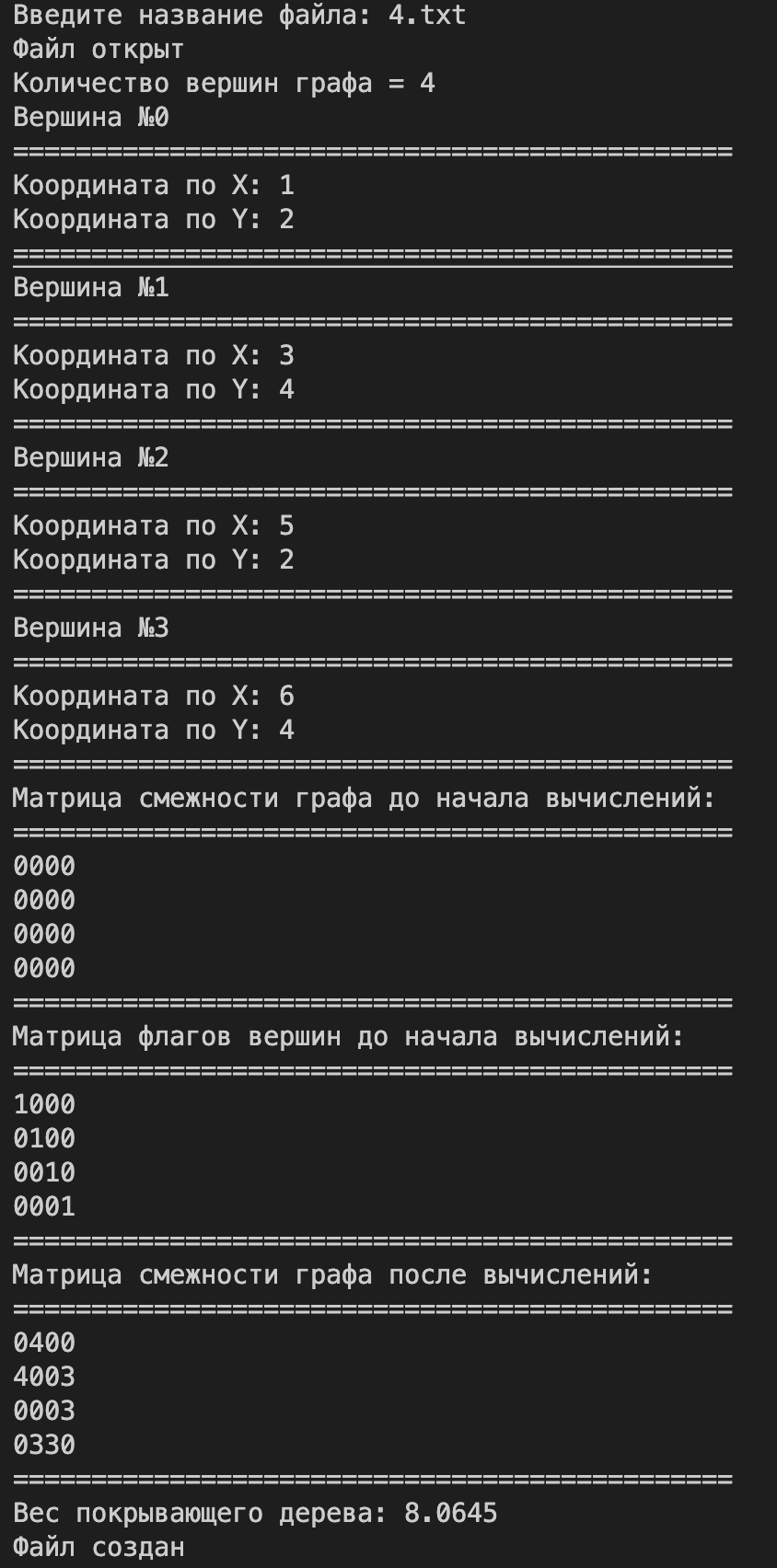


Рис. 18 Результат выполнения программы

4) Представим полученный результат (рис 19):

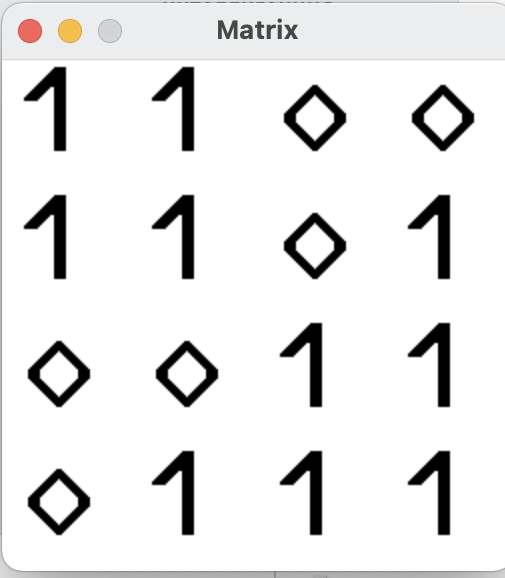


Рис. 19. Визуализированное решение программы.

Пример №4

1) Подан файл с количеством вершин и набором координа на рисунке 20.

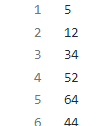


Рис. 20. Граф из варианта

2) Запускаем программу через команду bash start.sh

3) Результат выполнения программы (рис 21):

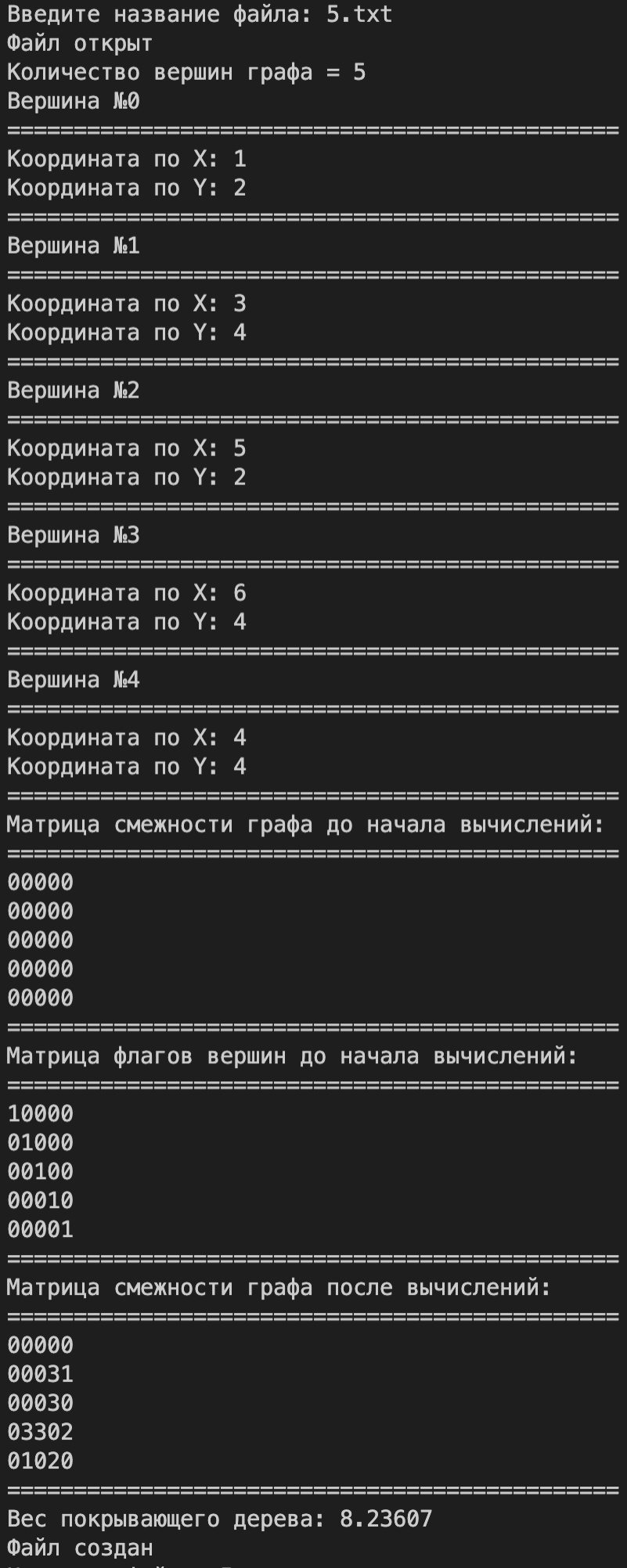


Рис. 21 Результат выполнения программы

4) Полученный результат (рис 22):

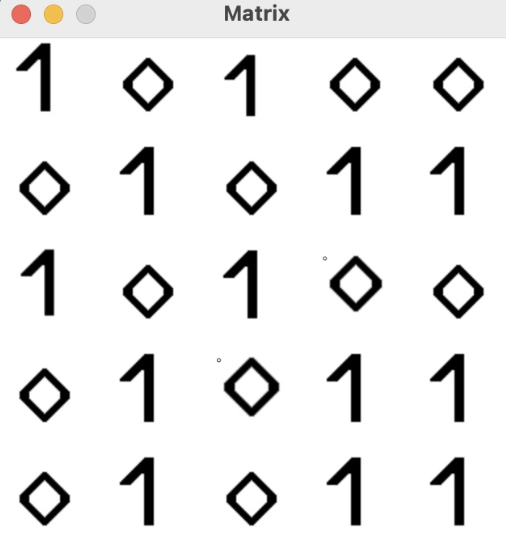
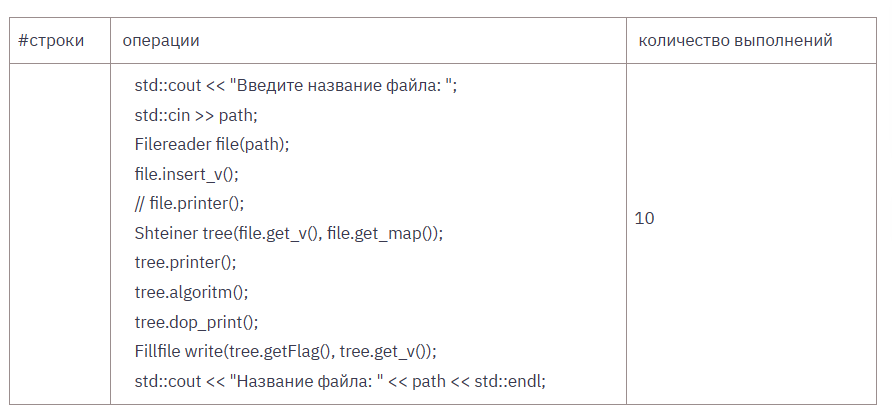


Рис. 22. Визуализированное решение программы.

**6 Оценка временной сложности**

Проведем оценку временной сложности алгоритма для задачи трассировки элементов СБИС на плоскости печатной платы.

1. Проведем декомпозицию алгоритма решения задачи, представив его в виде линейной последовательности процедур.
2. Выполним оценку трудоемкости блока инициализации алгоритма в зависимости от параметра размерности исходных данных задачи. Количественная оценка операций представлена в таблице 3.

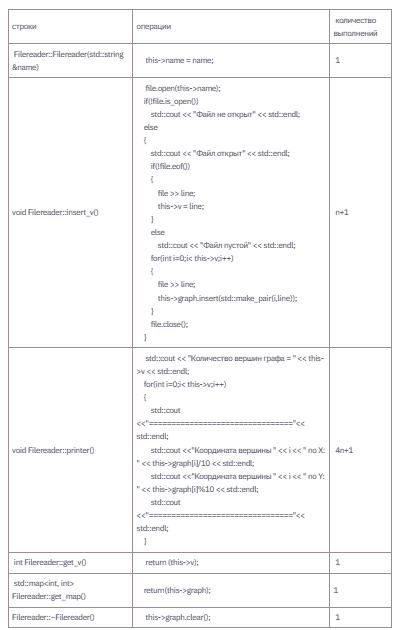
Таблица 3. Оценка трудоемкости блока main.

Таким образом, функция трудоемкости для данной процедуры инициализации будет иметь следующий вид :

T1=10

Выполним оценку трудоемкости для readfile. Количественная оценка операций представлена в таблице 4.

Таблица 4. Оценка трудоемкости блока filereader

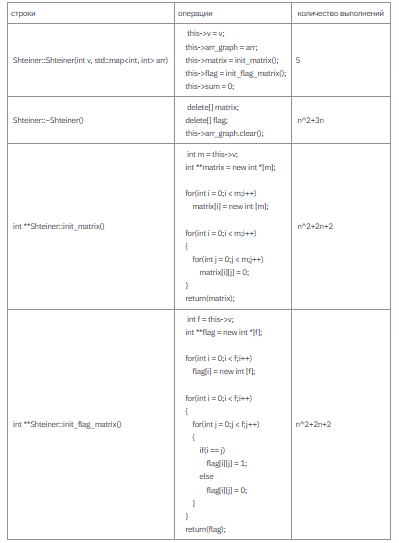


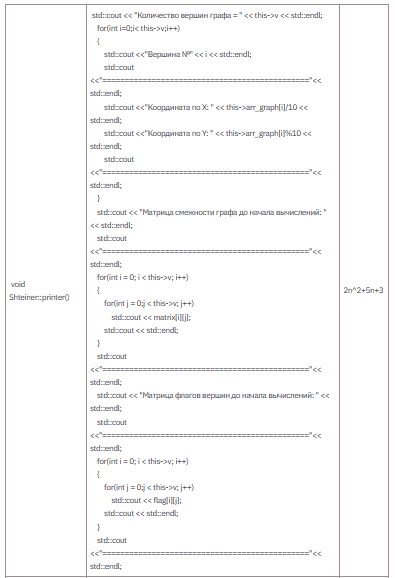
Таким образом, функция трудоемкости для данной процедуры будет иметь следующий вид :

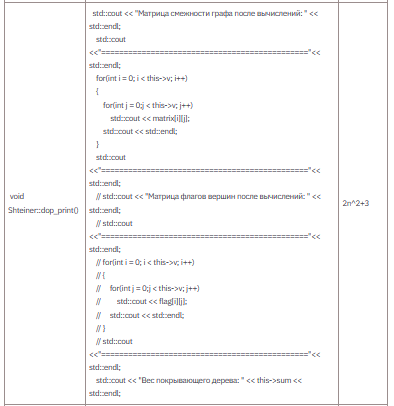
T2=5n+5

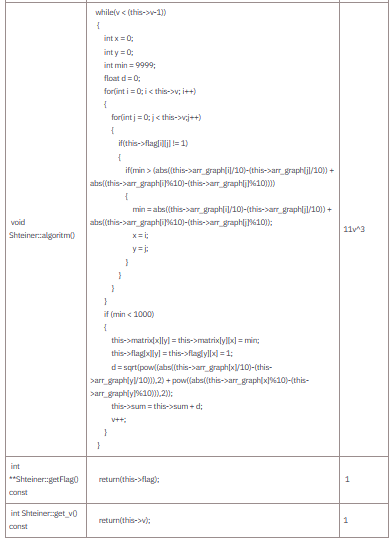
Проведем оценку трудоемкости блока Shteiner. Количественная оценка представлена на таблице 5.

Таблица 5. Оценка трудоемкости блока Shteiner.







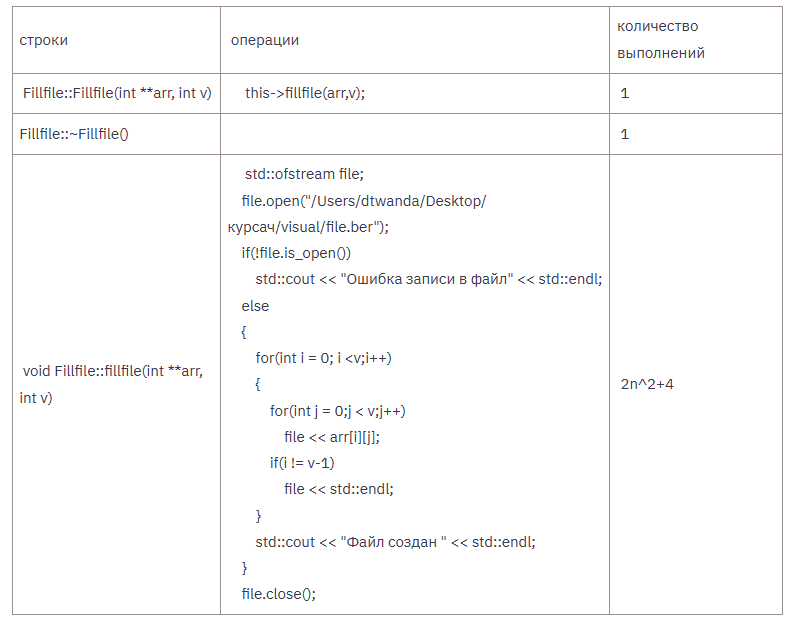


Таким образом, функция трудоемкости для данной процедуры будет :

T3= 11n^3+7n^2+12n+17

Проведем оценку трудоемкости для fillfile. Количественная оценка представлена на таблице 6.

Таблица 6. Оценка трудоемкости блока fillfile.



Таким образом, функция трудоемкости для данной процедуры будет :

T4=2n^2+6

Выполним композицию трудоемкости для всех блоков рассматриваемого алгоритма :

T=T1+T3+T4=

=10+ 5n+5+11n^3+7n^2+12n+17+ 2n^2+6 =

= 11n^3+9n^2+17n+38

1. Выполним асимптотическую оценку генетического алгоритма в виде O нотации:

O(11n^3+9n^2+17n+38)=O(n^3)

Таким образом, данный алгоритм имеет временную сложность пропорциональную O(n^3). Следовательно, наш алгоритм зависит от количества вершин графа.

**ЗАКЛЮЧЕНИЕ**

В данной курсовой работе мной была изучена задача о поиске минимального покрывающего дерева на основе первого алгоритма Штейнера для графа.

Была сформулирована содержательная и математическая постановка задачи. Также мы произвели описание алгоритма решения задачи и решили задачу на контрольном примере.

Мы разработали программу на языке программирования С++, которая реализует поведение алгоритма и строит визуальную модель получившейся матрицы графа. С помощью этой программы мы прорешали 4 примера различных задач трассировки месоединительных связей элементов СБИС.

Мы доказали актуальность алгоритма Штейнера в наши дни.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Воронова. В. В Информационные технологии проектирования электронных средств[ Электронный ресурс]: учеб-метод. пособие. Изд-во КГТУ им. А. Н. Туполева 2007 - 207стр. :URL: https://elibs.kai.ru/\_docs\_file/795827/HTML/index.html
2. Гальперин Александр Леонидович. Комбинаторные алгоритмы. Минимальный остов. 2018. Стр 18. Режим доступа: (creewick.github.io).
3. Заглядин Г.Г. Исследование реализации этапа декомпозиции при создании учебно-исследовательской САПР топологии СБИС. // Сб. науч. Трудов «Проектирование электронной компонентной базы и систем на кристалле» под ред. М.Г. Путри, М.: МИЭТ, 2007. с. 49-55.
4. Заглядин Г.Г, Разработка метода многокритериальной глобальной трассировки СБИС. // Микроэлектроника и информатика-2010, М.: МИЭТ, 2010. стр. 73.
5. Заглядин Г. Г., Сырцов И. А. Глобальная трассировка заказных СБИС с использованием множества остовных деревьев. // Естественные и технические науки. №4(48). М: «Спутник+», 2010. стр. 322-326.
6. Заглядин Г.Г. Метод глобальной трассировки цепей субмикронных СБИС с использованием семейства деревьев Штейнера. // Проектирование систем на кристалле: тенденции развития и проблемы, М.: МИЭТ, 2010. Стр.13
7. Заглядин Глеб Георгевич Исследование и разработка метода планироки цепей СБИС с равномерным распределением области трассировки 2011. Стр 5-17
8. [Задача Штейнера о минимальном дереве](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%A8%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D0%B0_%D0%BE_%D0%BC%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%B5). Научно-образовательный портал Википедия (остовное дерево) [Электронный ресурс]:URL: [Задача Штейнера о минимальном дереве — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%A8%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D0%B0_%D0%BE_%D0%BC%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%B5)
9. NP-полнота и сложность задачи Штейнера. [Электронный ресурс]:URL: [NP-полнота и сложность Задачи Штейнера Нечаева Инна 7381 (present5.com)](https://present5.com/np-polnota-i-slozhnost-zadachi-shtejnera-nechaeva-inna-7381/?ysclid=ld0t0i4z6r31429212)
10. В. А. Щербакова. Мощностная задача Штейнера на ориентированном градуированном графе. Стр 128-144

## ПРИЛОЖЕНИЕ А

## **Листинг программы.**

## Start.sh

#!/bin/bash

make re

./Shteiner

cd visual

./so\_long file.ber

Makefile

SRC = main.cpp Shteiner.cpp Filereader.cpp Fillfile.cpp

OBJ = $(SRC:.cpp=.o)

NAME = Shteiner

LIB = visual/so\_long

CC = clang++

FLAGS = -Wall -Werror -Wextra -std=c++98 -W

all: $(NAME) visual

visual:

@make -C visual/

$(NAME): $(OBJ)

$(CC) $(FLAGS) $(OBJ) -o $(NAME)

@make -C visual/

%.o: %.c

$(CC) $(FLAGS) -c $< -o $@ -I $(INCLUDE)

clean:

@rm -rf $(OBJ) $(OBJ\_B)

make -C visual/ clean

fclean: clean

@make -C visual/ fclean

rm -f $(NAME)

re: fclean all

Main.cpp

#include <iostream>

#include "Filereader.hpp"

#include "Shteiner.hpp"

#include "Fillfile.hpp"

int main()

{

std::string path;

std::cout << "Введите название файла: ";

std::cin >> path;

Filereader file(path);

file.insert\_v();

// file.printer();

Shteiner tree(file.get\_v(), file.get\_map());

tree.printer();

tree.algoritm();

tree.dop\_print();

Fillfile write(tree.getFlag(), tree.get\_v());

std::cout << "Название файла: " << path << std::endl;

}

Shteiner.hpp

#pragma once

#include <iostream>

#include <map>

#include "Filereader.hpp"

#include <cmath>

class Shteiner

{

private:

std::map<int, int> arr\_graph;

int v;

int \*\*matrix;

int \*\*flag;

float sum;

public:

Shteiner(int v, std::map<int, int> arr);

~Shteiner();

int \*\*init\_matrix();

int \*\*init\_flag\_matrix();

void printer();

void dop\_print();

void algoritm();

int \*\*getFlag() const;

int get\_v() const;

};

Shteiner.cpp

#include "Shteiner.hpp"

Shteiner::Shteiner(int v, std::map<int, int> arr)

{

this->v = v;

this->arr\_graph = arr;

this->matrix = init\_matrix();

this->flag = init\_flag\_matrix();

this->sum = 0;

}

Shteiner::~Shteiner()

{

delete[] matrix;

delete[] flag;

this->arr\_graph.clear();

}

int \*\*Shteiner::init\_matrix()

{

int m = this->v;

int \*\*matrix = new int \*[m];

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

matrix[i] = new int [m];

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

{

for(int j = 0;j < m;j++)

matrix[i][j] = 0;

}

return(matrix);

}

int \*\*Shteiner::init\_flag\_matrix()

{

int f = this->v;

int \*\*flag = new int \*[f];

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

flag[i] = new int [f];

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

{

for(int j = 0;j < f;j++)

{

if(i == j)

flag[i][j] = 1;

else

flag[i][j] = 0;

}

}

return(flag);

}

void Shteiner::printer()

{

std::cout << "Количество вершин графа = " << this->v << std::endl;

for(int i=0;i< this->v;i++)

{

std::cout <<"Вершина №" << i << std::endl;

std::cout <<"=============================================="<< std::endl;

std::cout <<"Координата по X: " << this->arr\_graph[i]/10 << std::endl;

std::cout <<"Координата по Y: " << this->arr\_graph[i]%10 << std::endl;

std::cout <<"=============================================="<< std::endl;

}

std::cout << "Матрица смежности графа до начала вычислений: " << std::endl;

std::cout <<"=============================================="<< std::endl;

for(int i = 0; i < this->v; i++)

{

for(int j = 0;j < this->v; j++)

std::cout << matrix[i][j];

std::cout << std::endl;

}

std::cout <<"=============================================="<< std::endl;

std::cout << "Матрица флагов вершин до начала вычислений: " << std::endl;

std::cout <<"=============================================="<< std::endl;

for(int i = 0; i < this->v; i++)

{

for(int j = 0;j < this->v; j++)

std::cout << flag[i][j];

std::cout << std::endl;

}

std::cout <<"=============================================="<< std::endl;

}

void Shteiner::dop\_print()

{

std::cout << "Матрица смежности графа после вычислений: " << std::endl;

std::cout <<"=============================================="<< std::endl;

for(int i = 0; i < this->v; i++)

{

for(int j = 0;j < this->v; j++)

std::cout << matrix[i][j];

std::cout << std::endl;

}

std::cout <<"=============================================="<< std::endl;

// std::cout << "Матрица флагов вершин после вычислений: " << std::endl;

// std::cout <<"=============================================="<< std::endl;

// for(int i = 0; i < this->v; i++)

// {

// for(int j = 0;j < this->v; j++)

// std::cout << flag[i][j];

// std::cout << std::endl;

// }

// std::cout <<"=============================================="<< std::endl;

std::cout << "Вес покрывающего дерева: " << this->sum << std::endl;

}

void Shteiner::algoritm()

{

int v = 0;

while(v < (this->v-1))

{

int x = 0;

int y = 0;

int min = 9999;

float d = 0;

for(int i = 0; i < this->v; i++)

{

for(int j = 0; j < this->v;j++)

{

if(this->flag[i][j] != 1)

{

if(min > (abs((this->arr\_graph[i]/10)-(this->arr\_graph[j]/10)) + abs((this->arr\_graph[i]%10)-(this->arr\_graph[j]%10))))

{

min = abs((this->arr\_graph[i]/10)-(this->arr\_graph[j]/10)) + abs((this->arr\_graph[i]%10)-(this->arr\_graph[j]%10));

x = i;

y = j;

}

}

}

}

if (min < 1000)

{

this->matrix[x][y] = this->matrix[y][x] = min;

this->flag[x][y] = this->flag[y][x] = 1;

d = sqrt(pow((abs((this->arr\_graph[x]/10)-(this->arr\_graph[y]/10))),2) + pow((abs((this->arr\_graph[x]%10)-(this->arr\_graph[y]%10))),2));

this->sum = this->sum + d;

v++;

}

}

}

int \*\*Shteiner::getFlag() const

{

return(this->flag);

}

int Shteiner::get\_v() const

{

return(this->v);

}

Fillfile.hpp

#pragma once

#include "Shteiner.hpp"

#include "fstream"

class Fillfile

{

public:

Fillfile(int \*\*arr, int v);

~Fillfile();

int \*\*init(int \*\*arr, int v);

void fillfile(int \*\*arr, int v);

};

Fillfile.cpp

#include "Fillfile.hpp"

Fillfile::Fillfile(int \*\*arr, int v)

{

this->fillfile(arr,v);

}

Fillfile::~Fillfile()

{

}

void Fillfile::fillfile(int \*\*arr, int v)

{

std::ofstream file;

file.open("/Users/dtwanda/Desktop/курсач/visual/file.ber");

if(!file.is\_open())

std::cout << "Ошибка записи в файл" << std::endl;

else

{

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

{

for(int j = 0;j < v;j++)

file << arr[i][j];

if(i != v-1)

file << std::endl;

}

std::cout << "Файл создан " << std::endl;

}

file.close();

}

Filereader.hpp

#pragma once

#include <iostream>

#include <map>

#include <fstream>

class Filereader

{

private:

std::string name;

int v;

std::map<int, int> graph;

public:

Filereader(std::string &name);

~Filereader();

int get\_v();

std::map<int, int> get\_map();

void insert\_v();

void printer();

};

Filereader.cpp

#include "Filereader.hpp"

Filereader::Filereader(std::string &name)

{

this->name = name;

}

void Filereader::insert\_v()

{

std::ifstream file;

int line;

file.open(this->name);

if(!file.is\_open())

std::cout << "Файл не открыт" << std::endl;

else

{

std::cout << "Файл открыт" << std::endl;

if(!file.eof())

{

file >> line;

this->v = line;

}

else

std::cout << "Файл пустой" << std::endl;

for(int i=0;i< this->v;i++)

{

file >> line;

this->graph.insert(std::make\_pair(i,line));

}

file.close();

}

}

void Filereader::printer()

{

std::cout << "Количество вершин графа = " << this->v << std::endl;

for(int i=0;i< this->v;i++)

{

std::cout <<"================================"<< std::endl;

std::cout <<"Координата вершины " << i << " по X: " << this->graph[i]/10 << std::endl;

std::cout <<"Координата вершины " << i << " по Y: " << this->graph[i]%10 << std::endl;

std::cout <<"================================"<< std::endl;

}

}

int Filereader::get\_v()

{

return (this->v);

}

std::map<int, int> Filereader::get\_map()

{

return(this->graph);

}

Filereader::~Filereader()

{

this->graph.clear();

}

Приложение B

Makefile

NAME = so\_long

CFLAGS = -Wall -Wextra -Werror

CC = gcc

MLX\_PATH = ./minilibx

HEADER = so\_long.h

RM = rm -f

SRCS = main.c image.c map.c mem.c\

get\_next\_line/get\_next\_line.c\

get\_next\_line/get\_next\_line\_utils.c\

OBJS = ${SRCS:.c=.o}

all: ${NAME}

.c.o:

${CC} ${CFLAGS} -c $< -o ${<:.c=.o}

$(NAME): $(OBJS) $(HEADER)

$(MAKE) --directory=minilibx

$(CC) $(OBJS) $(CFLAGS) -Lminilibx -lmlx -framework OpenGL -framework AppKit -o $(NAME)

bonus: all

clean:

${RM} ${OBJS}

fclean: clean

${RM} ${NAME}

make -C $(MLX\_PATH) clean

mlx\_re:

make -C $(MLX\_PATH) re

re: fclean all

.PHONY: all clean fclean re bonus

So\_long.h

#pragma once

# include "get\_next\_line/get\_next\_line.h"

# include "stdio.h"

# include "minilibx/mlx.h"

# include <string.h>

# define PIXEL 64

typedef struct s\_coord

{

int x;

int y;

} t\_coord;

typedef struct s\_image

{

void \*image\_wall;

void \*image\_floor;

} t\_image;

typedef struct s\_vars

{

void \*mlx;

void \*win;

char \*\*arr;

int length;

int count;

int fd;

int flag\_dir;

t\_image image;

void \*tmp\_image;

char \*c;

} t\_vars;

void error(int n);

void fill(t\_vars \*vars, char \*argv);

void memory(t\_vars \*vars);

int map(t\_vars \*vars, char \*argv);

void map\_check(t\_vars \*vars, int \*flag\_n, char \*line);

int map\_check2(t\_vars \*vars, char \*line, int flag\_n);

int check\_symbols(char \*line);

void init(t\_vars \*vars);

void image(t\_vars \*vars, int h, int w);

int draw\_map(t\_vars \*vars);

void draw\_images(t\_vars \*vars, char c, int i, int j);

void draw\_images\_e\_c\_v(t\_vars \*vars, char c, int i, int j);

int close\_window(t\_vars \*vars);

Main.c

void error(int n)

{

if (n == 1)

printf("Error: wrong number of arguments\n");

else if (n == 2)

printf("Error: file read error\n");

else if (n == 3)

printf("Map error: invalid string length\n");

else if (n == 5)

printf("Map error: invalid symbols in the map\n");

else if (n == 6)

printf("Map error: empty lines in file\n");

exit(1);

}

void init(t\_vars \*vars)

{

vars->flag\_dir = 0;

vars->mlx = NULL;

vars->win = NULL;

vars->arr = NULL;

vars->c = "1";

}

int close\_window(t\_vars \*vars)

{

mlx\_destroy\_window(vars->mlx, vars->win);

vars->win = NULL;

exit (0);

}

int main(int argc, char \*\*argv)

{

t\_vars vars;

if (argc != 2)

{

error (1);

exit (1);

}

init(&vars);

map(&vars, argv[1]);

memory(&vars);

fill(&vars, argv[1]);

vars.mlx = mlx\_init();

image(&vars, 0, 0);

vars.win = mlx\_new\_window(vars.mlx, PIXEL \* vars.length,

PIXEL \* vars.count, "Matrix");

mlx\_hook(vars.win, 17, 1L << 5, close\_window, &vars);

mlx\_loop\_hook(vars.mlx, draw\_map, &vars);

mlx\_loop(vars.mlx);

return (0);

}

Mem.c

#include "so\_long.h"

void memory(t\_vars \*vars)

{

int i;

int k;

vars->length = vars->length - 1;

vars->arr = (char \*\*)malloc(vars->count \* sizeof(int \*));

if (!vars->arr)

{

perror(NULL);

exit (1);

}

i = 0;

while (i < vars->count)

{

vars->arr[i] = malloc((vars->length) \* sizeof(char));

if (!vars->arr[i])

{

perror(NULL);

k = 0;

while (k < i)

free(vars->arr[k++]);

free(vars->arr);

exit (1);

}

i++;

}

}

void fill(t\_vars \*vars, char \*argv)

{

int i;

int j;

char \*line;

vars->fd = open(argv, O\_RDONLY);

if (vars->fd < 0)

error(2);

line = get\_next\_line(vars->fd);

i = 0;

while (line)

{

j = 0;

while (j < vars->length)

{

vars->arr[i][j] = line[j];

j++;

}

i++;

free(line);

line = get\_next\_line(vars->fd);

}

close(vars->fd);

}

Map.c

#include "so\_long.h"

int map(t\_vars \*vars, char \*argv)

{

char \*line;

int flag\_n;

vars->fd = open(argv, O\_RDONLY);

if (vars->fd < 0)

error(2);

line = get\_next\_line(vars->fd);

vars->length = ft\_strlen(line);

vars->count = 0;

while (line)

{

map\_check(vars, &flag\_n, line);

free(line);

line = get\_next\_line(vars->fd);

}

if (flag\_n == 1)

error(6);

close(vars->fd);

return (0);

}

void map\_check(t\_vars \*vars, int \*flag\_n, char \*line)

{

\*flag\_n = 0;

if (line[ft\_strlen(line) - 1] == '\n')

\*flag\_n = 1;

vars->count++;

if ((\*flag\_n && ((int)ft\_strlen(line) != vars->length))

|| (!(\*flag\_n) && ((int)ft\_strlen(line) != vars->length - 1)))

error(3);

map\_check2(vars, line, \*flag\_n);

}

int map\_check2(t\_vars \*vars, char \*line, int flag\_n)

{

if (flag\_n)

line[vars->length - 1] = '\0';

if (vars->count > 1 && flag\_n)

{

check\_symbols(line);

}

return (0);

}

int check\_symbols(char \*line)

{

int i;

i = -1;

while (line[++i])

{

if (line[i] != '0' && line[i] != '1')

error(5);

}

return (0);

}

Image.c

#include "so\_long.h"

void image(t\_vars \*vars, int h, int w)

{

vars->image.image\_wall

= mlx\_xpm\_file\_to\_image(vars->mlx, "./images/1.xpm", &h, &w);

vars->image.image\_floor

= mlx\_xpm\_file\_to\_image(vars->mlx, "./images/0.xpm", &h, &w);

}

void draw\_images(t\_vars \*vars, char c, int i, int j)

{

if (c == '0')

{

vars->tmp\_image = vars->image.image\_floor;

mlx\_put\_image\_to\_window(vars->mlx, vars->win,

vars->tmp\_image, j \* PIXEL, i \* PIXEL);

}

if (c == '1')

{

vars->tmp\_image = vars->image.image\_wall;

mlx\_put\_image\_to\_window(vars->mlx, vars->win,

vars->tmp\_image, j \* PIXEL, i \* PIXEL);

}

else if ( c != '0')

draw\_images\_e\_c\_v(vars, c, i, j);

}

int draw\_map(t\_vars \*vars)

{

int i;

int j;

i = -1;

while (++i < vars->count)

{

j = -1;

while (++j < vars->length)

draw\_images(vars, vars->arr[i][j], i, j);

}

return (0);

}

void draw\_images\_e\_c\_v(t\_vars \*vars, char c, int i, int j)

{

if(c != '0')

{

mlx\_put\_image\_to\_window(vars->mlx, vars->win,

vars->tmp\_image, j \* PIXEL, i \* PIXEL);

}

}

0.xpm

/\* XPM \*/

static char \*\_d5baf52e4464a64ebe10f04b2410c49IpcmBfNcBWFKu3rK[] = {

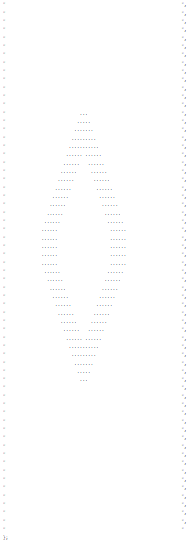
/\* columns rows colors chars-per-pixel \*/

"64 64 2 1 ",

" c white",

". c black",

/\* pixels \*/



1.xpm

/\* XPM \*/

static char \*\_d5baf52e4464a64ebe10f04b2410c49IpcmBfNcBWFKu3rK[] = {

/\* columns rows colors chars-per-pixel \*/

"64 64 2 1 ",

" c white",

". c black",

/\* pixels \*/

