# 项目报告

## 项目实验结果：

输出了下列信息到 output.txt 文件中

C1 0 0 # 单元 C1 的位置为(0, 0)

C2 1 0 # 单元 C2 的位置为(1, 0)

C3 2 0 # 单元 C3 的位置为(2, 0)

C4 0 1 # 单元 C4 的位置为(0, 1)

## 技术路线与实现方式：

1. 解析输入文件： 通过读取输入文件，解析出网格的行数和列数，单元数，线网数以及线网内单元的连接方式，并存储到数据结构中。
2. 数据结构： `Unit` 结构体：表示单元，包含名称、位置属性。`Net` 结构体：表示线网，包含名称和与之连接的单元列表。`Placement` 结构体：表示整个布局，包含网格大小、单元数，线网数，以及两个容器 `units` 单元列表和 `nets` 线网列表。
3. 布局算法： 使用穷举法来生成不同的单元排列组合。针对每个排列组合，计算多个线网值的和，根据值的大小衡量布局的好坏，并更新布局结果，选择值最小的排列组合作为最终的布局。
4. 输出文件生成： 将最终布局结果写入输出文件。包括单元名称，具体位置和注释。

## 主要问题及解决方法：

1.解析输入文件并存储数据：通过逐行读取输入文件中包含的信息，包括网格的行数和列数，单元的名称和数目，线网数以及线网内单元的连接方式等数据，存储在`Unit`，`Net`，`Placement`结构体中。还定义了`units`，`nets`的作为容器，用来存储和处理布局中的单元和线网信息

2.布局的算法设计：（1）计算两个单元之间的距离，使用了<cmath>中的函数计算平

方根和四舍五入转化为整数。

（2）计算线网的长度，通过遍历布局和线网中的单元，计算线网中每对单元之间的距离，然后将所有距离累加得到线网的总长度。

（3）寻找最佳布局，使用全排列和穷举的方法，遍历所有可能的布局，并计算每个布局下的总长度，找到总长度最小的布局作为最佳布局，并将其应用到实际布局中。

3.输出文件信息：按照格式要求，输出所有单元的坐标信息。

4.求出单元的初始坐标：通过遍历 placement.units 的容器，根据索引值，取模运算（%）得到单元初始的横坐标， 整除运算（/）得到单元初始的纵坐标。

**主要创新点：**

无

**总结：**

通过本次项目，我更熟悉了 c++及其使用方法。在网上查找资料的过程中，进一步了解了如何读取文件、储存数据、计算线长、实现最优布局和输出文件的方法，并学习了 GitHub 与 git 的简单用法。