ASCII 矩形计数器设计说明文档

**1. 项目概述**

**1.1 项目背景**

在软件开发和文本处理中，经常需要对 ASCII 字符图进行分析。本项目旨在开发一个高效、准确的工具，用于识别和统计 ASCII 图中矩形的数量，为相关应用提供基础功能支持。

**1.2 项目目标**

设计并实现一个能够识别由 +、- 和 | 字符构成的矩形的程序，要求：

* 能够处理各种规模和复杂度的 ASCII 网格
* 准确识别符合矩形定义的结构
* 提供友好的用户交互界面
* 保证算法效率和代码可维护性

**1.3 文档目的**

本文档旨在详细描述 ASCII 矩形计数器的设计思路、技术实现和使用方法，为开发人员、测试人员和维护人员提供全面的技术参考。

**2. 需求分析**

**2.1 功能需求**

* **矩形识别**：能够准确识别 ASCII 图中的矩形结构
* **数量统计**：计算并输出识别到的矩形总数
* **输入验证**：确保用户输入的合法性和有效性
* **错误处理**：对非法输入和异常情况提供明确的提示信息

**2.2 非功能需求**

* **性能要求**：在合理时间内处理大规模 ASCII 网格
* **易用性**：提供简洁明了的用户界面和操作指南
* **可维护性**：代码结构清晰，注释完善，便于后续扩展和维护
* **可扩展性**：支持未来添加更多图形识别功能

**2.3 约束条件**

* 仅支持由 +（角）、-（水平边）和 |（垂直边）构成的矩形
* 输入网格必须是规则矩形（每行长度相同）
* 不考虑斜向或不规则形状的矩形

**3. 设计方案**

**3.1 总体架构**

本项目采用模块化设计，主要包括以下几个模块：

* **输入处理模块**：负责读取用户输入的 ASCII 网格数据
* **验证模块**：检查输入数据的合法性和有效性
* **矩形检测模块**：核心算法模块，负责识别和验证矩形
* **计数模块**：统计并输出识别到的矩形数量
* **辅助工具模块**：提供 HTML 可视化工具，帮助用户生成测试数据

**3.2 数据结构设计**

|  |
| --- |
| // 表示网格中的一个点  private class Point {  var x: Int64 // X坐标  var y: Int64 // Y坐标  init(x: Int64, y: Int64) {  this.x = x  this.y = y  }  }  // 存储 ASCII 网格数据  var inputGrid : ArrayList<String> |

**3.3 算法设计**

采用 "点检测法" 识别矩形，具体步骤如下：

1. **遍历所有点**：扫描整个 ASCII 网格，找出所有 + 字符作为可能的矩形起点
2. **确定候选点**：
   * 对于每个起点，水平向右查找所有可能的第二个点（+ 字符）
   * 垂直向下查找所有可能的第三个点（+ 字符）
3. **计算第四点**：根据第二点和第三点的位置，计算第四点的坐标并验证其是否为 + 字符
4. **验证矩形边**：检查四条边是否符合矩形定义（水平边由 - 或 + 组成，垂直边由 | 或 + 组成）

**3.4 模块详细设计**

**3.4.1 输入处理模块**

|  |
| --- |
| // 读取用户输入的网格数据  public func readGridInput() : ArrayList<String> {  // 实现代码...  } |

**3.4.2 验证模块**

|  |
| --- |
| // 验证输入网格是否为规则矩形  private func validateGrid(inputGrid : ArrayList<String>) : Boolean {  // 实现代码...  } |

**3.4.3 矩形检测模块**

|  |
| --- |
| // 从给定点出发查找所有可能的矩形  private func findRectsFormPoint(p1 : Point, inputGrid : ArrayList<String>) : Int64 {  // 实现代码...  }  // 查找水平方向的可能点  private func possiblePointsHorizontal(inputGrid : ArrayList<String>, p :Point) : ArrayList<Point> {  // 实现代码...  }  // 查找垂直方向的可能点  private func possiblePointsVertical(inputGrid : ArrayList<String>, p :Point) : ArrayList<Point> {  // 实现代码...  }  // 计算并验证第四点  private func findP4(inputGrid : ArrayList<String>, p2 :Point, p3 :Point) : Point {  // 实现代码...  } |

**3.4.4 计数模块**

|  |
| --- |
| // 统计整个网格中的矩形数量  public func countRectangles(inputGrid : ArrayList<String>) : Int64 {  // 实现代码...  } |

**4. 界面设计**

**4.1 命令行界面**

程序启动后显示简洁的提示信息，引导用户输入：

|  |
| --- |
| 请输入网格数据，使用Enter结束输入：  请输入网格的行数：  行 1:  行 2:  ...  找到的矩形数量: X |

**4.2 HTML 辅助工具界面**

HTML 工具提供可视化编辑界面，主要包括：

* 网格编辑区域：用户可点击生成 +、-、| 字符
* 操作按钮：生成代码、清除网格等功能
* 代码输出区域：显示生成的可复制 ASCII 代码

|  |
| --- |
| 图 4.2-1 |

**5. 实现细节**

**5.1 关键算法实现**

矩形检测的核心算法采用四重循环结构：

1. 遍历所有 + 字符作为起点
2. 对每个起点，遍历所有可能的第二点
3. 对每个第二点，遍历所有可能的第三点
4. 验证第四点并检查四条边的有效性

**5.2 边界处理**

* 检查坐标是否超出网格范围
* 处理空输入和无效输入情况
* 确保每行长度一致

**5.3 异常处理**

* 使用 try-catch 块捕获可能的异常
* 对无效输入提供明确的错误提示
* 确保程序在异常情况下能够安全退出

**6. 测试计划**

**6.1 测试目标**

验证程序的功能正确性、性能表现和异常处理能力。

**6.2 测试用例设计**

**6.2.1 基本功能测试**

| **测试用例** | **输入数据** | **预期输出** |
| --- | --- | --- |
| TC-001 | 空网格 | 0 |
| TC-002 | 单个 + 字符 | 0 |
| TC-003 | 简单矩形  `+--+`  `+--+` | 1 |
| TC-004 | 嵌套矩形 | 正确计数所有矩形 |

**6.2.2 边界测试**

| **测试用例** | **输入数据** | **预期输出** |
| --- | --- | --- |
| TC-011 | 最大尺寸网格 | 在合理时间内完成计算 |
| TC-012 | 只有水平或垂直边 | 0 |

**6.2.3 异常处理测试**

| **测试用例** | **输入数据** | **预期输出** |
| --- | --- | --- |
| TC-021 | 非矩形输入 | 错误提示："输入的行数必须大于 0" |
| TC-022 | 包含非法字符的网格 | 错误提示："无效字符" |

**7. 性能分析**

**7.1 时间复杂度**

算法的时间复杂度为 O (n²m²)，其中 n 为网格行数，m 为列数。这是因为需要四重嵌套循环来检查所有可能的矩形组合。

**7.2 空间复杂度**

空间复杂度为 O (nm)，主要用于存储输入的 ASCII 网格数据。

**8. 部署与使用说明**

**8.1 环境要求**

* 仓颉编程语言环境
* 支持 HTML5 的现代浏览器（用于辅助工具）

**8.2 部署步骤**

1. 安装仓颉编译器
2. 克隆项目代码到本地
3. 进入程序文件夹：cd rectangle
4. 编译程序：cjc main.cj -o main
5. 运行程序：./main

或者在控制台直接编译并运行：cjpm run

**8.3 使用方法**

1. 启动程序后，按照提示输入网格行数
2. 逐行输入 ASCII 网格数据
3. 程序将自动计算并输出矩形数量

**8.4 HTML 辅助工具使用**

1. 在浏览器中打开 ascii-generator.html 文件
2. 在网格中绘制所需的矩形结构
3. 点击 "生成代码" 按钮获取可复制的 ASCII 字符串
4. 将字符串粘贴到程序输入界面进行测试

**9. 附录**

**9.1 术语表**

* **ASCII 矩形**：由 +、-、| 字符构成的矩形结构
* **角点**：矩形的四个顶点，由 + 字符表示
* **水平边**：矩形的上下边，由 - 或 + 字符组成
* **垂直边**：矩形的左右边，由 | 或 + 字符组成

**9.2 参考资料**

* 仓颉编程语言官方文档

**10. 修订历史**

| **版本** | **修订日期** | **修订人** | **修订内容** |
| --- | --- | --- | --- |
| 1.0 | 2025-06-07 | 周昕 | 初始版本 |