|  |  |  |  |
| --- | --- | --- | --- |
| **学生学号** | 0122210880507 | **实验课成绩** |  |

**武汉理工大学**

**学 生 实 验 报 告 书**

**课 程 名 称 数据结构与算法综合实验**

**开 课 学 院 计算机科学与技术学院**

**指 导 老 师 胡燕**

**学 生 姓 名 吴肸昊**

**专 业 班 级 软件zy2201**

2023 — 2024 学年 第 二 学期

实验课程名称： 数据结构与算法综合实验

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 二叉树与哈夫曼图片压缩 | | | 实验成绩 |  |
| 实验者 | 吴肸昊 | 专业班级 | 软件zy2201 | 组别 |  |
| 同组者 |  | | | 实验日期 | 2024年2月2日 |
| 第一部分：实验分析与设计（可加页）   1. **实验目的和要求** 2. 目的：通过“二叉树与哈夫曼图片压缩”的编程实践，学习树、遍历二叉树、哈夫曼树、哈夫曼编码和他们的编程应用。 3. 掌握树的存储结构； 4. 掌握二叉树的三种遍历方法； 5. 掌握并理解Huffman树、Huffman编码等知识和应用； 6. 掌握文件的操作； 7. 使用Huffman算法实现图像压缩程序。 8. 要求：使用Huffman压缩算法，对一幅BMP格式的图片文件进行压缩。图片文件名为 “Pic.bmp”,压缩后保存为“Pic.bmp.huf”文件。使用VS Studio作为开发工具，开发一个 控制台程序，使用Huffman压缩算法对图片文件“Pic.bmp”进行压缩。具体如下： 9. 读取原文件，统计权值：以“Pic.bmp”文件为例。若文件存放在F盘根目录下，输入文件完整路径“F：\Pic.bmp”。 按回车结束。以字节流的方式，只读打“Pic.bmp”文件。逐字节读取文件，统计文件中256种字节重复的次数，保存到一个数组中int weight[256] 中。 10. 生成Huffman树：根据（1）中统计的结果，构建Huffman树。定义一个结构体来记录每个节点的权值、 父节点、左孩子和右孩子。使用结构体数组来存储这个Huffman树。 11. 生成Huffman编码：遍历（2）中生成的Huffman树，记录256个叶子节点的Huffman编码，保存在字符串 数组中。 12. 压缩原文件：使用Huffman编码对原文件中的字节重新编码，获得压缩后的文件数据。 13. 保存压缩文件：将编码过的数据，保存到文件“Pic.bmp.huf”中。 14. **分析与设计** 15. 数据结构设计 16. 文件头HEAD结构体：用于记录压缩文件类型，原文件长度和原文件中各个字节的权重。  1. HuffmanNode结构体：作为哈夫曼树的节点，用于构建整个文件的Huffman树，从而得到哈夫曼编码。  1. HuffmanCode结构体：用于记录对应字节的哈夫曼编码。  1. 整个程序主要包括哈夫曼编码生成，文件压缩以及解压缩几个部分，不同的功能分别写在不同的文件。 2. 程序中所构建的哈夫曼树以及记录字节的哈夫曼编码均是采用数组的形式，根据数组的下标进行快速定位与访问。 3. 核心算法设计 4. 在“Huffman.h”文件中，定义HuffmanNode和HuffmanCode两个结构体，分别用来构造Huffman树和存储Huffman编码信息。然后根据字节对应的权重信息，构造Huffman树，此处Huffman树采用HuffmanNode数组的方式进行存储，每个结构体元素中记录着其父系节点的索引以及其左右孩子的索引。大致流程是先构造叶子节点，即将每个字节的权重信息写入数组的前256个元素中；之后构造非叶子节点，从构造好的节点中每次选出两个最小的节点进行构造，而其对应的父节点则是从未构造的数组元素中依次获取，并完善结构体中信息。最后构造得到的数组的第511个元素即为Huffman树的根节点。  1. 对字节进行哈夫曼编码则是根据构造，构造得到的Huffman树进行递归遍历，进入左子树编码为0，进入右子树编码为1，直到到达叶子节点；此处需要注意的是需要采用位移运算。 2. 在“Compress.cpp”文件中，首先根据Compress()函数计算压缩后的编码字节数；然后在Encode()函数中根据字节数创建一个缓冲区，逐个读取文件字节，并将其Huffman编码信息存入缓冲区之中。最后在WriteFile文件中将缓冲区中的信息写入压缩文件并保存。  1. 文件的解压操作在“Unpack.cpp”文件中进行。首先读取压缩文件信息，将其中的字节信息转换为“0”“1”表示的字符信息并存入到一个临时文件中，此临时文件中存储的信息为字符串表示的Huffman编码信息。然后逐个读取临时文件中的字符，同时在Huffman树中进行查找，解析Huffman编码对应的原字节信息，并将其写入解压文件中保存。  1. 测试用例设计 2. 实验中测试的用例主要是几张bmp图片，可以从网络上下载一些bmp格式的图片，或者可以自己通过画图工具来生成几张bmp格式文件。 3. 根据图片的具体内容不同，对应的压缩率也会有所不同，简单的图片压缩率较高，而颜色样式比较复杂的图片的压缩率则会比较低。   三、**实验软硬件环境**  PC机、Microsoft Visual Studio 2022  第二部分：实验过程与结果（可加页）   1. **实验说明（所建工程文件等的说明）** 2. 首先构建工程项目“HuffmanCompressCPro”，并在其中添加文件。其中“Main.cpp”文件中编写main函数作为程序的主函数；“Huffman.cpp”文件中编写构建Huffman树与生成Hunffman编码的相关代码；“Compress.cpp”文件中编写将图片根据Huffman编码进行压缩并生成压缩文件的相关代码；“Unpack.cpp”文件中则编写将压缩文件解压并还原为原图片的相关代码。源文件都有与之对应的头文件且在其中进行了函数声明。 3. 压缩的图片需要手动进行输入文件路径，然后读取进行哈夫曼编码，经过压缩得到的文件为原文件名后面加上后缀“.huf”，并保存在与原文件相同的路径下。 4. 在解压过程中的文件包括一个记录Huffman编码字符形式的临时文件（.temp文件）和解压缩得到的文件，都放置在与原文件相同的路径下。 5. **调试说明（调试手段、过程及结果分析）** 6. 在本次实验中的难点主要在于原字节信息与Huffman编码之间的转换以及解压时Huffman编码的还原： 7. 由于Huffman编码的长度并不相同，而在存储文件时则需要一个一个字节的写入，所以要对Huffman编码和需要写入的细节进行处理。此处的解决方法是对Huffman编码的字符表示依次进行拼接，当字符串长度大于8时，则截取前8位字符并转化为一个字节写入缓冲区之中。对于最后剩余的字符串的长度可能不足8个，则需要在其后面进行补0操作再转换为字节写入。 8. Huffman编码还原时，若是直接读取字节中的0和1并遍历Huffman树，则会难以处理。所以此处我将所有字节信息转换为字符信息后依次读取字符。而在Huffman编码转换为原字节信息时也容易出错。当判断一个Huffman编码结束时，此时读取字符的位置已经位于下一个Huffman编码的首个字符，再转换为原字节后不能将这个首字符遗漏。同时转换得到的字节编码需要根据读取的上一个字符来决定，而不是当前字符。否则都会出现解压错误的情况。 9. 在调试过程中，主要是查看对应变量值的变化，查看对应内存中的数据是否正确。在程序预编译失败时，则需要找到报错的地方，在相关代码部分打断点，一步一步调试运行。 10. 在本次实验中，是将各个功能依次实现，而在实现的过程中，则会相应地编写一些对应的函数，将得到的一些数据进行输出，从而判断程序是否运行正确。 11. 对于压缩完成的图片，由于无法显示，所以想要验证压缩是否正确，则需要对压缩文件进行解压缩，恢复成原来的图片。在此过程中经过了多次调试，一方面是对比压缩前与解压时得到的哈夫曼编码是否相同，另一方面是将哈夫曼编码与解读出来的原码进行对照，观察是否一致。 12. **软件测试（测试效果、界面、综合分析和结论）** 13. 根据输入的文件地址，统计文件字节数并输出字节对应的权重信息：  1. 构建哈夫曼树并输出其节点信息：      1. 先序遍历哈夫曼树并输出哈夫曼编码信息：      1. 输出压缩文件相关信息：   第三部分：实验小结、收获与体会   1. 通过本次实验，是我对二叉树以及Huffman编码的知识进行回顾复习，对其中的相关知识点也有了进一步的理解与掌握。 2. 此次实验也使我将所学的相关知识应用于实际的图片压缩场景中，完成了从知识到应用的转化。在此实践过程中也收获了喜悦与成就感。 3. 同时，在本次的编程实践中，我也遇到了很多的问题，并经过不断的调试与修改得到解决；使我明白了一些小细节的问题也决定了程序的成功与否。在此后的实践过程中，我也会吸取教训，注重代码中的细节问题，不断锻炼自身编写代码的严谨性与高效性。 | | | | | |

实验课程名称： 数据结构与算法综合实验

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 景区信息管理系统 | | | 实验成绩 |  |
| 实验者 | 吴肸昊 | 专业班级 | 软件zy2201 | 组别 |  |
| 同组者 |  | | | 实验日期 | 2024年2月9日 |
| 第一部分：实验分析与设计（可加页）  **一、实验目的和要求**   1. 目的 2. 掌握图的定义和图的存储结构； 3. 掌握图的创建方法； 4. 掌握图的两种创建方法； 5. 理解迪杰斯特拉 (Dijkstra)算法； 6. 理解最小生成树的概念和普里姆 (Prim)算法； 7. 掌握文件操作； 8. 使用 C++语言，利用图的数据结构，开发景区信息管理系统。 9. 要求   现有一个景区，景区里面有若干个景点，景点之间满足以下条件：   1. 某些景点之间铺设了道路（相邻） 2. 这些道路都是可以双向行驶的（无向图） 3. 从任意一个景点出发都可以游览整个景区（连通图）   开发景区信息管理系统，对景区的信息进行管理。使用图的数据结构来保存景区景 点信息，为用户提供创建图、查询景点信息、旅游景点导航、搜索最短路径、铺设电路 规划等功能。  **二、分析与设计**   1. 数据结构设计 2. 图的存储结构由邻接矩阵和顶点数组组成；邻接矩阵是一个二维数组，其大小根据景区的景点数量确定；每个元素表示对应两个景点之间是否存在道路以及道路的长度；顶点数组则存储了所有景点的详细信息。 3. 图的存储结构：   **邻接矩阵：**项目中使用了二维数组（邻接矩阵）来表示图的边的信息。每个元素的值代表两个顶点之间的权值，即景点之间的距离。如果两个景点之间没有直接的道路连接，则对应的权值为0。  顶点数组：使用一维数组来存储图的顶点信息，每个顶点包含一个结构体，其中包含景点的编号、名称和介绍。   1. 景点信息的数据结构：   Vex结构体：用于存储单个景点的信息，包含景点编号、名称和介绍以及相邻景点的集合。这个结构体在顶点数组中被使用，以表示图的顶点。  Edge结构体：用于存储景点之间的道路信息，包含两个顶点的编号和它们之间的距离（权值）。这个结构体用于表示图中的边。  Path结构体：用于存储一个顶点到另一个顶点的路径信息，包含表示一条路径的数组和指向下一条不同路径的指针。   1. 核心算法设计 2. 深度优先搜索（DFS）：   该算法用于旅游景点导航功能，通过遍历图来寻找从一个景点出发，无回路游览整个景区的路线。  算法从指定的起始景点开始，访问其所有未访问的邻接景点，然后递归地对这些邻接景点执行相同的操作，直到所有景点都被访问过。为了生成多条路线，算法需要进行相应的优化，比如在回溯时尝试其它可能的路径并存储。   1. 迪杰斯特拉算法（Dijkstra）：   该算法用于搜索两个景点之间的最短路径。  算法通过维护一个优先队列来记录从起始景点到其他所有景点的最短距离和路径。它逐步扩展已知的最短路径集合，直到找到目标景点的最短路径。算法的关键步骤包括初始化、选择最小距离的未访问顶点、更新相邻顶点的距离和路径、以及输出最终结果。   1. 普里姆算法（Prim）：   该算法用于铺设电路规划功能，目的是构造一个最小生成树，使得所有景点都能通电，同时总线路长度最短。  算法从一个景点开始，逐步增加新的边和顶点到生成树中，每次都选择连接生成树和未包含顶点之间权值最小的边。这个过程重复进行，直到所有顶点都被包含在生成树中。算法需要一个有效的数据结构来管理已访问的顶点和边，以及进行权值的比较和选择。   1. 算法改进与优化：   这些算法在项目中的具体实现涉及到图的数据结构，包括邻接矩阵和顶点数组。所以在实验的具体实践过程中需要根据需求对算法进行改进与优化。其中需要改进深度优先搜索算法，将所有符合的路线存储到链表中；同时避免重复访问和存储已访问顶点的信息。   1. 测试用例设计   本次实验的测试用例采用指导书中提供的景点地图，将与其相关的顶点与边信息写入两个文件中，通过读取文件将景点信息存入定义的景点图类中进行相关处理。  也可以自己构建一个景点地图，将其相关信息写入文件中从而进行景区的相关操作。  三、**实验软硬件环境**  PC机、Microsoft Visual Studio 2022  第二部分：实验过程与结果（可加页）   1. **实验说明（所建工程文件等的说明）** 2. 该实验的工程名为 GraphCPro，解决方案与工程名相同。工程中包含多个文件，每个文件承担不同的职责和功能。 3. 所包含的文件： 4. CGraph.h和CGraph.cpp文件：CGraph.h文件定义了顶点、边和路径三个结构体以及表示图及其相关操作的CGraph类；CGraph.cpp文件中则是CGraph类中所包含方法的具体实现。包括图的初始化、获取顶点或边的信息、深度优先搜索、搜索最短路径、构造最小生成树等方法。 5. CTourism.h和CTourism.cpp文件：定义了CToursim类的声明以及其中方法的具体实现。包括创建景区景点图、查新指定顶点信息、查询旅游导航路线、搜索两个顶点的最短路径和查询铺设电路规划图方法。 6. Header.h文件：其中包含了整个项目所需的所有头文件，方便统一管理头文件。 7. Main.cpp文件：其中定义了程序的入口点，通过在控制台显示菜单来与用户进行交互。 8. 核心代码介绍： 9. DFS函数（深度优先搜索得到两个顶点间的所有可行路径）：   参数：nVex（当前顶点索引）、aVisited[]（记录顶点是否被访问过的顶点数组）、nIndex（当前路径的长度）、nexts（相邻顶点的集合）、pList（存储所有可行路径的链表）。  思路：如果pList为空，直接返回；如果nIndex等于顶点数nVexNum，说明已经找到了一条完整的路径，将新路径加入链表。遍历当前顶点的邻接顶点集合nexts[nVex]，将未访问过的顶点加入vec向量；复制aVisited和nIndex数组，以便在递归过程中保留原始状态。对于vec中的每个顶点，执行以下操作：如果sign为false，表示当前顶点是路径的第一个顶点，将其加入路径中，并标记为已访问；如果sign为true，表示当前顶点不是路径的第一个顶点，需要创建一个新的路径对象p，并将前面已经得到的路径复制到p中；遍历当前顶点的邻接顶点集合nexts[nVex]，将未访问过的顶点加入新路径中，并标记为已访问；递归调用DFS函数，继续搜索下一个顶点；如果递归调用返回后，nIndex不等于nVexNum，说明当前路径不完整，删除新创建的路径对象p，并返回；如果当前顶点是路径的第一个顶点，将其加入set集合中，表示已经处理过该顶点。重复上一步骤，直到vec中的所有顶点都被处理完毕。   1. FindShortPath函数（迪杰斯特拉算法得到两点间的最短路径）：   思路：初始化一个visited数组，用于记录每个顶点是否被访问过，初始值都为0。将起始顶点nVexStart标记为已访问，即visited[nVexStart] = 1。初始化一个path数组，用于记录从起始顶点到每个顶点的最短路径距离，初始值都为-1。将起始顶点nVexStart到自身的距离设为0，即path[nVexStart] = 0。初始化一个preIndex数组，用于记录每个顶点的前驱顶点，初始值都为-1。初始化一个visitedEdge数组，用于记录每条边是否被访问过，初始值都为0。遍历所有的边，根据起始顶点nVexStart更新路径距离和边的访问信息。如果某条边的一个顶点是起始顶点，则将该边的权重作为起始顶点到另一个顶点的距离，并将该边的索引存入preIndex数组中，同时将visitedEdge数组中对应位置的值设为1。进行nVexNum-1次循环，每次循环中找到未访问的最近的点的下标minIndex，并将其标记为已访问。以找到的点minIndex为中间媒介，更新其他点的距离。遍历所有的边，如果边已经读取过，则直接跳过；否则，根据边的权重和路径距离，更新路径距离和前驱顶点的信息。循环结束后，将路径下标添加到vector aPath中，从终点nVexEnd开始，依次获取前驱顶点，直到回到起始顶点nVexStart。最后返回路径距离path[nVexEnd]。   1. FindMinTree函数（构造最小生成树来规划电路铺设路线）：   思路：初始化一个空集合set，将起始顶点0加入集合中。创建一个数组visitedEdge，用于记录每条边是否被访问过，初始值都为0。初始化变量minWeight和minIndex，分别表示当前找到的最小权重和对应的边的索引。进入循环，直到集合set的大小等于顶点数nVexNum为止。在每次循环中，遍历所有的边，找到一条不构成闭环的最小边。具体判断方法是：如果集合set中包含某条边的其中一个顶点，而另一个顶点不在集合中，则这条边满足条件。如果找到了符合条件的边，将其权重与当前的minWeight进行比较，如果更小，则更新minWeight和minIndex。将找到的最小边添加到路径aPath中，并更新总权重allWeight。将该边标记为已访问，即将visitedEdge数组中对应位置的值设为1。将新增的顶点加入到集合set中，即如果该边的一个顶点不在集合中，则将该顶点加入集合；否则将另一个顶点加入集合。重复以上步骤，直到集合set的大小等于顶点数nVexNum，即所有顶点都被访问过。最后返回最短距离allWeight。   1. **调试说明（调试手段、过程及结果分析）** 2. 本次实验中主要的难点是实现几个算法，所以实验中出现的错误以及调试主要是在实现这几个算法的函数中。 3. 主要的调试手段是先将整个工程进行编译，如果编译通过，则进行交互测试；如果编译不通过，则根据报错信息确定错误代码的位置；然后对代码进行分析，找出其中不合理的代码进行修改。如果无法根据报错提示来定位到错误代码，则需要在合适的位置设置断点，逐行代码进行检查。 4. 在本次实验过程中，遇到的一个问题是在使用深度优先搜索遍历景点时，不能采用传统的方法将所有遍历信息全部储存，需要在无回路的条件下将所有的景点全部遍历，否则最终输出的结果会出错。同时在构建新路径的同时，需要将之前已经走过的路径进行赋值，然后再分支中递归进入下一个景点节点，而当发现此路径不可行时，则要将其删除。在此过程中涉及动态申请内存和回收，需要严格注意。 5. 本次实验中深度优先搜索规划路径我采用的是递归的方式构建路径，同时也可以使用自己构建栈的方法，但是由于不同情况讨论与操作过与复杂，所以直接递归调用系统栈。 6. 实验最终的结果显示都符合预期输出。但是所涉及的几个算法流程有些复杂，应该还有一定的改进空间。 7. **软件测试（测试效果.界面、综合分析和结论）** 8. 菜单页面显示以及景区图原型：      1. 创建景区景点图：      1. 景区路径规划导航、搜索两景点的最短路径以及铺设电路规划：     第三部分：实验小结、收获与体会   1. 在本次实验中，通过实际编写代码实现图的数据结构和相关算法，使我加深了对无向图、邻接矩阵、深度优先搜索（DFS）、迪杰斯特拉算法和普里姆算法等概念的理解。 2. 同时在实验过程中，通过C++的编码实践，提高了我的编程能力，尤其是在处理文件、数据结构操作、算法实现和面向对象。 3. 在编码和调试过程中我也体会到了细节的重要性，一个小小的语法错误或逻辑疏漏都可能导致程序运行失败或结果不正确。 4. 在编码实践中我也体会到了耐心与毅力的重要性，只有具备足够的耐心与伊利，才能在面对复杂算法和难以调试的代码时不放弃，不断克服困难，取得成功。 | | | | | |

实验课程名称： 数据结构与算法综合实验

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 欢乐连连看 | | | 实验成绩 |  |
| 实验者 | 吴肸昊 | 专业班级 | 软件zy2201 | 组别 |  |
| 同组者 |  | | | 实验日期 | 2024年3月30日 |
| 第一部分：实验分析与设计（可加页）  **一、实验目的和要求**   1. 目的   本次实验需要通过“欢乐连连看”项目，达到如下目标：   1. 了解项目业务背景，调研与连连看相同类型的游戏，了解连连看的功能和规则等。 2. 掌握C++开发工具和集成开发环境。 3. 掌握C++面向对象的编程思想和C++的基础编程。 4. 理解MFC基本框架，包括MFC Dialog和GDI 编程。 5. 掌握数据结构，包括算法控制、数组、栈、图；同时掌握算法，比如数组的遍历、图的遍历（如：DFS）、连通判断等。 6. 实施项目的调研和分析，阅读和填充项目的过程文档。 7. 了解项目的开发流程，学习软件工程的，迭代思想，开发GUI应用程序。 8. 养成良好的编码习惯，提高C++语言编程能力，开发应用程序“欢乐连连看”。 9. 要求   本实验需要根据进度，依次完成以下任务：   1. 创建工程、绘制欢迎语、主界面设计、游戏界面设计、绘制游戏地图； 2. 同色消子、程序结构调整、消子判断、数据结构设计； 3. 开始游戏、消子、胜负判断； 4. 提示、重排、计时、帮助、休闲模式、排行榜、设计主题   **二、分析与设计**  **数据结构设计**   1. 在此项目中需要的数据结构设计主要是对游戏地图的操作，由于游戏地图是一个二维的对象，所以最简单的便是使用一个二维数组来表示游戏地图的元素信息。 2. 除此之外，还可以将整个游戏地图封装成一个图类，在其中添加对地图操作的一些成员函数，从而到达控制游戏地图的效果。 3. 但是如果根据视频指导上定义的图类，用一个长的一维数组来表示地图元素，再用一个二维数组来表示点与点之间的关系，这样的做法无疑会浪费更多的内存空间，同时图片元素与数据对应的过程中还需要新增一步转换操作，也会增加复杂性。所以在本项目中，对游戏地图的元素操作通过一个二维数组来进行，并在逻辑控制类中实现相关的函数来对此二维数组进行操作。 4. 对于每个图片元素，使用一个Vertex结构体来表示其相关信息，其中包含该元素的行、列、类型等信息。   **核心算法设计**   1. 随机初始化地图：首先获取地图的大小（行数和列数）以及元素数量，并根据地图的大小，动态分配内存空间来存储地图数据；然后检查内存分配是否成功，如果失败则抛出异常。将地图数据初始化为0，检查地图的元素数量是否满足要求，如果不满足则释放内存并抛出异常。若不发生异常，则计算每个元素需要重复的次数，按照指定的重复次数，将元素的值填充到地图中。接着设置随机数种子，根据生成的随机数随机交换地图中的任意两个元素的位置，从而打乱地图的顺序。最后返回初始化后的地图数据。  1. 连通判断：首先获取输入的两个元素的行和列坐标。其次判断两个元素是否在同一行或同一列上，如果是，则判断它们之间的元素是否为空，如果为空，则将这两个顶点添加到路径中，并返回true。如果两个顶点不在同一行或同一列上，判断它们是否可以通过两条线段（一个拐点）进行连接。如果两个顶点不能通过一个拐点连接，判断它们是否可以通过三条直线（两个拐点）连接。如果以上条件都不满足，说明两个顶点之间不连通，返回false。  1. 一个拐点连通判断：首先判断拐点是否为空，再判断是否连通。这里的拐点是两个顶点行列相交的一个空白元素。如果拐点为空，即pGameMap[v1.row][v2.col] == BLANK 或 pGameMap[v2.row][v1.col] == BLANK，那么调用LinkX和LinkY函数来判断拐点与两个顶点之间的元素是否为空。如果为空，则将这三个顶点添加到路径中，并返回true。如果拐点不为空，则返回false。其中LinkX函数用于判断(nRow, nCol1)到(nRow, nCol2)是否连通，即两个元素是否行连通，LinkY函数用于判断(nRow1, nCol)到(nRow2, nCol)是否连通，即两个元素是否列连通。  1. 两个拐点连通判断：该过程首先根据两个元素所在的行列信息交叉得到一个矩形，从而将整个地图分为内部和外部两部分。然后判断中间区域是否相连，该过程会在矩形区域内部通过遍历找寻可能的两个拐点，然后判断拐点与拐点之间以及拐点与元素之间是否全部为空，如果若全部为空，则能够相连，返回true。如果中间区域无法相连，再判断外围区域是否相连；此过程又将外围区域分为上下左右四个区域，通过从内到外依次遍历找寻可能的拐点，判断两个元素能否通过两个拐点相连，如果相连则返回true。如果以上条件都不满足，则说明两个顶点之间不连通，返回false。  1. 寻找可以相连的元素：首先通过两层嵌套循环遍历游戏地图的每一个元素。外层循环遍历行，内层循环遍历列。如果当前遍历到的元素值为BLANK（空白），则跳过当前循环，继续下一个元素的遍历。对于非空白元素，再遍历其下方的所有元素。这里需要注意的是，内层循环的起始列数为当前元素的列数，避免重复检查已经检查过的元素。如果当前遍历到的元素值为BLANK（空白），则跳过当前循环，继续下一个元素的遍历。如果当前遍历到的元素值与之前遍历到的非空白元素值相同，那么创建一个Vertex结构体v1和v2，分别存储这两个元素的行、列和值信息。调用IsLink函数，判断这两个元素之间是否存在有效路径。如果存在有效路径，则返回true。如果遍历完所有元素后都没有找到有效路径，返回false。  1. 重排游戏地图：首先，获取地图的行数nRows和列数nCols。然后设置随机数种子，使用当前时间作为种子值，并计算地图中的顶点总数nVertexNum，即行数乘以列数。接着使用for循环进行nVertexNum次迭代，在每次迭代中，生成两个随机索引nIndex1和nIndex2，范围为0到nVertexNum-1，根据这两个随机索引，找到对应的地图上的位置，并交换这两个位置上的元素。最后，当所有迭代完成后，地图上的元素就已经被随机交换过，从而实现了地图的重排。（与随机生成地图算法相似） 2. 判断游戏获胜：首先，可以在每次消子后对游戏地图进行判断，若地图元素全为空，则可以断定游戏获胜。但是这种方法的时间复杂度较大，若游戏地图过大，则会影响游戏性能。因此，此处我是在CGameControl类中定义了一个元素个数的变量m\_nElemNum，初始化为开始的元素总数，当每次成功消子后数值就减2。而在每次点击消子后，就只需要判断m\_nElemNum的数值是否为0，就可知道是否消子完全从而游戏获胜。而在每次点击开始游戏时则会根据行列数重新初始化此变量。 3. 判断游戏失败：游戏失败是根据游戏时间和游戏地图元素数来判定的，若游戏时间为0且游戏元素个数不为0，则游戏失败。此判定过程是在与进度条相关联的OnTimer函数中进行的，当游戏剩余时间m\_nTime = 0时，则终止时间进程，并且判断游戏元素个数是否为0，若为0，则弹出游戏失败的界面窗口并重置按钮功能。   **测试用例设计**   1. 本次实验由于是基于MFC框架的连连看游戏设计，所以需要测试的便是游戏界面是否正常显示，同时也要对整个游戏逻辑的正确性进行判断。 2. 前期的准备便是需要一些游戏背景图片和消子元素的图片。分别采用静态加载位图和动态加载位图的方式，观察图片能否正确加载到游戏界面窗口之中。 3. 其次是判断整个游戏的逻辑。首先需要验证点击不同按钮是否会出现预期的窗口界面，其次需要判断游戏过程中是否能够正确消子，消子时的连线是否正确；当游戏成功或失败时是否会出现弹框。同时在设置异常时是否会出现弹窗警告。 4. 同时也需要注意全局与单个模式界面之间的关系，当设置游戏背景或游戏的行列数、元素数和时间时，不同的模式下都应该做出响应。   三、**实验软硬件环境**  PC机、Microsoft Visual Studio 2022  第二部分：实验过程与结果（可加页）  **一、实验说明（所建工程文件等的说明）**   1. 首先在VS2022中新建一个MFC工程，工程名为“LianLianKan”，其中已经预定义了众多的头文件与源文件，整个项目需要在此基础上进行修改和扩展，定义新的头文件与源文件，并编写相关类与函数的代码。      1. LLKDlg.cpp文件和LLKDlg.h文件：文件实现一个名为CLLKDlg的对话框类，用于显示连连看游戏的初始界面。在这个类中，实现了一些成员函数，如DoDataExchange、OnSysCommand、OnPaint、OnQueryDragIcon等，同时也包含了各个按钮的事件处理函数，用于处理对话框的各种事件。此外，类中还定义了InitBackground函数和UpdateWindow函数，用于初始化对话框的背景图片和调整对话框窗口的大小。 2. CGameDlg.cpp文件和CGameDlg.cpp文件：文件是实现了CGameDlg类，该类属于游戏界面层，用于显示在不同模式下的游戏界面。在这个类中，构造函数会根据不同的模式对成员数据进行不同的初始化，InitBackground函数、InitElem函数和UpdateWindow函数则用于对话框的初始化，对话框初始化函数中也对各个按钮的可用功能进行了设置。该类中还包含了开始游戏、暂停游戏、提示、重排、帮助、设置等按钮以及点击游戏区域的事件函数，用于处理对话框的各种事件。 3. CGameControl.cpp文件和CGameControl.h文件：文件实现了CGameControl游戏控制类，该类属于游戏控制层，在CGameDlg类中包含了此类的对象，从而实现了在游戏界面操作时实现对界面的控制。该类包含了开始游戏初始化、设定选中元素、判断游戏是否相连、给予提示、重排地图、判断游戏是否获胜等方法，从而实现对游戏的控制。 4. CGameLogic.cpp文件和CGameLogic.h文件：文件实现了CGameLogic游戏逻辑类，该类属于游戏逻辑层，具体实现表示游戏的数据的各种内在逻辑操作，是对CGameControl类中各个方法的具体逻辑的实现。包含为游戏地图动态申请内存、异常检查等操作，以及游戏逻辑操作的具体方法。 5. global\_var.cpp文件和global.h文件：这两个文件中定义一些该项目的全局变量，包括窗口对话框的宽和高，游戏地图的行、列、元素种类，游戏时长，背景图片路径等。所定义的这些变量主要用于窗口对话框的初始化、游戏地图的初始化、在设置对话框中设置不同的背景图片以及游戏的行列数、元素数和游戏时间。 6. CWinDlg文件和CFailDlg文件：相关文件实现了CWinDlg类和CFailDlg类，这两个类是游戏获胜或游戏失败时所显示的窗口对话框，在CGameDlg的相关方法中使用。 7. CHelpDialog.cpp文件和CHelpDialog.h文件：文件中相关代码实现了一个名为CHelpDialog的对话框类，主要实现了一个帮助对话框，用户可以通过滚动条浏览帮助信息。当用户点击帮助按钮时显示该对话框。 8. CSettleDialog文件和CSetDialog文件：相关文件实现了CSettleDialog类和CSetDialog类，这两个类都是实现设置对话框，但是相关实现有所不同。CSettleDialog类实现的是多界面切换设置窗口，可以切换两个界面，一个界面设置游戏背景，另一个界面设置游戏行列数等，该类在游戏初始界面中点击设置界面时触发。而CSetDialog仅仅是实现了设置游戏行列数、元素个数和时间，该类在点击游戏界面中的设置按钮进行触发。 9. CPage0文件和CPage1文件：相关文件中的代码是CSettleDialog类中两个界面的具体实现。   **二、调试说明（调试手段、过程及结果分析）**   1. 该项目采取的是迭代开发的方法，所以每当完成该项目的一部分功能时可以对代码进行编译运行，查看游戏窗口界面是否符合预期的显示，若不符合预期，则需要对代码进行检查修改，直至界面显示正确。 2. 若编写的相关代码与游戏逻辑相关，则需要在游戏界面多进行交互，检查元素图片框选是否正确、连线是否正确、消子是否正确、游戏成功或失败时是否有相关窗口，以及点击相应按钮时是否会由对应的正确响应。该过程需要从不同的方面进行各种的多次的交互，只有这样才有可能发现所编写游戏逻辑的异常，从而对代码进行改进。 3. 该项目中的主要难点在于一些控件及其相关相应函数的使用和整个游戏逻辑的统筹。在调试过程中，可以设置断点逐行检查代码，或者在网上查找相关资料来弥补一些知识盲区，从而在编写代码时避免一些错误。 4. 经过多次的调试、修改与丰富代码，整个项目能够正确运行练练看游戏，游戏过程中的逻辑也符合预期设计。 5. **软件测试（测试效果.界面、综合分析和结论）** 6. 游戏初始界面以及游戏界面：对于游戏界面会有不同的模式，在不同模式下游戏背景会有所不同，相关的功能也会有所差别，但整体的布局显示都是一致的；基本模式与休闲模式都可以点击设置按钮对游戏地图和时间进行修改，而关卡模式则不能设置，但其在每次胜利后的下一局中都会对地图加以调整，从而不断增加游戏难度。      1. 游戏暂停界面、提示界面、重排界面：暂停界面会有一张图片对游戏地图加以覆盖，同时游戏时间的进度条也会停止计时；提示界面则会框出两个可以消除的地图元素并进行连接；重排界面则是将原有的游戏地图全部打乱，不同种类的元素个数和空白区域的个数不会改变。      1. 游戏设置界面：此游戏有两种设置界面，一种在初始游戏界面显示，一种在游戏界面显示。      1. 游戏帮助界面：主要是介绍游戏功能与游戏规则。  1. 经过多次的测试与调试修改，整个项目能够正确地显示游戏界面。在点击相应的按钮时，界面也会出现对应的界面弹框；在游戏界面中，通过点击能够正确实现图片元素的选中、相连、消子等操作；而通过点击相关按钮，也能够实现游戏开始、暂停、继续、提示、重排、帮助、设置等功能。总体而言，在测试过程中连连看游戏项目符合预期，但是对于一些小细节，仍然有改进优化的空间。   第三部分：实验小结、收获与体会   1. 在这个实验项目中，我深入了解了开发连连看游戏的业务背景和流程。通过使用C++开发工具和集成开发环境Microsoft Visual Studio 2022，我学习并掌握了MFC框架和GDI编程，并对相关数据结构和算法进行了实践。 2. 在项目实验过程中，我大致了解了软件开发流程，并最终成功开发了一个具有GUI界面的连连看游戏应用程序。通过相关内容的学习与实际编写代码的实践，不仅丰富了我的知识储备，同时也提高了我的C++编程能力。 3. 在连连看游戏中我实现了各种功能，包括开始游戏、消子、提示、重排、判断胜负、设置游戏、控制游戏、游戏积分和游戏模式等。我使用了数组、图等数据结构来存储游戏数据，并通过设计不同的算法来组织整个游戏的逻辑，使游戏功能符合预期。此外，我还学习了程序的三层结构：表示层、业务逻辑层和数据访问层。这有助于将不同层次的功能分离，使代码更加模块化和可维护。 4. 总之，这个实验项目对于提升我的软件开发能力和C++编程技巧非常有帮助。我通过实践应用所学知识，加深了对软件开发流程和相关技术的理解。这些经验和技能也将对我未来的编程实践产生积极的影响。 | | | | | |