

**课 程 设 计**

**课程设计名称： 数据结构课程设计**

**题 目 ： 哈夫曼编码/译码器**

**专 业 班 级 ： 软件1702**

**学 生 姓 名 ： 刘文博**

**学 号 ： 201716040224**

**指 导 教 师 ： 程立**

**课程设计时间： 2018.12.24-2019.1.4**

**软件工程 专业课程设计任务书**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学生姓名** | 刘文博 | **专业班级** | | 软件1702 | **学号** | | 201716040224 |
| **题 目** | **哈夫曼编码/译码器** | | | | | | |
| **课题性质** | 工程设计 | | **课题来源** | | | 自拟课题 | |
| **指导教师** | 程立 | | **同组姓名** | | | 无 | |
| **主要内容** | 综合应用所学知识，设计完成一个哈夫曼编码/译码器。本系统拟实现以下功能：  1. 编码/译码功能；  2. 显示哈夫曼树；  3. 界面设计的优化。  系统要求采用Codeblocks或Dev C++或Visual C++等工具进行开发实现。 | | | | | | |
| **任务要求** | 综合运用和融合所学理论知识，提高分析和解决实际问题的能力，使用C/C++语言设计一个哈夫曼编码/译码器。  完成课程设计报告，报告中对关键部分给出图表说明。要求格式规范，工作量饱满。 | | | | | | |
| **参考文献** | [1] 严蔚敏, 吴伟民. 数据结构（C语言版）[M]. 北京:清华大学出版社, 2012.  [2] 严蔚敏, 吴伟民. 数据结构题集（C语言版）[M]. 北京:清华大学出版社, 2011.  [3] 谭浩强. C语言程序设计[M]. 北京:清华大学出版社, 2014.  [4] [美]Paul Deitel, Harvey Deitel著, 张引等译. C++大学教程（第九版）[M].北京: 电子工业出版社, 2016.  [5] 郭炜. 新标准C++程序设计 [M]. 北京: 高等教育出版社, 2016. | | | | | | |
| **审查意见** | **教研室主任签字： 2018 年 12 月 10 日** | | | | | | |

目录

[1 需求分析 1](#_Toc534542993)

[1.1 添加信息 1](#_Toc534542994)

[1.2 建立哈夫曼树 1](#_Toc534542995)

[1.3 遍历哈夫曼树 1](#_Toc534542996)

[1.4 编码的解密 1](#_Toc534542997)

[1.5 密码的加密 1](#_Toc534542998)

[2 概要设计 2](#_Toc534542999)

[**2.1** 数据类型的定义 2](#_Toc534543000)

[定义节点的元素类型 2](#_Toc534543001)

[2.2功能模块结构图 2](#_Toc534543002)

[3 详细设计 4](#_Toc534543003)

[3.1 添加 4](#_Toc534543004)

[3.2 显示 4](#_Toc534543005)

[3.3 加密 4](#_Toc534543006)

[3.4 解密 4](#_Toc534543007)

[3.5 退出保存 4](#_Toc534543008)

[4 运行结果 5](#_Toc534543009)

[5 调试分析 7](#_Toc534543010)

[6 心得体会 8](#_Toc534543011)

[7 参考文献 9](#_Toc534543012)

# 需求分析

通过和指导老师交流，了解到本系统中的数据来源于标准输入设备（如键盘）或者来自某文件，可以实现对哈夫曼树的节点信息的添加。

## 添加信息

从文件中读取信息，分别读取相应的字符和其权值，生成哈夫曼树。

## 建立哈夫曼树

能够根据输入的节点信息生成相对应的哈夫曼树。

## 遍历哈夫曼树

递归遍历建立好的哈夫曼树，形成节点信息。

## 编码的解密

根据哈夫曼编码对给定的编码进行解密。

## 密码的加密

根据给定的密码，使用哈夫曼编码对其进行加密。

# 概要设计

## 数据类型的定义

## 定义节点的元素类型

typedef struct node

{

int weight;

char str;

struct node\* left;

struct node \*right;

}Node,\*pNode ;

## 2.2功能模块结构图

根据需求分析，为了满足用户的功能需求，按照软件开发方法学中的模块划分原则，我将本系统主要划分为如下模块：添加查询和退出保存，各模块之间的关系如图1所示。

图 1模块结构图

为了实现上述功能模块，分别在顺序表和单链表物理结构上定义了多个函数，本系统定义的函数和功能如下：

/\*\*从文件中读取数据 \*/

void ReadingFile(vector<Node\*> &vec\_node ,char passward[] , char str[]);

/\*\* 建立哈弗曼树 \*/

void CeateHalfTree(Node\* TreeRoot,vector<Node\*> &vec\_node);

/\*\*遍历HuffmanTree 并记录各个字符的路径 \*/

void TravelHalfTree(pNode root,vector<int> &show,char str1[100][50] ,int& i , int& k);

/\*\* 将各个的节点保存到一个向量中，方便下一次的操作 \*/

void StoryHalfTree(char str1[100][50] , vector<string> & element,int& i);

/\*\* 打印各个节点的路径 \*/

void PrintHalfTree(vector<string> & element);

/\*\* 加密 \*/

void EncryptionString(vector<string> & element ,char str[] ,char encryptionstr[]);

/\*\* 解密 \*/

void Decodestring(Node\* TreeRoot, char passworld[] , char decodestr[]);

void SetColor(int x , int y);

void SetPosition(short int x ,short int y);

/\*\* 显示菜单 \*/

void ShowMenu();

/\*\* 将数据保存到文件中 \*/

void OutFile(vector<string> &element ,char encryptionstr[] , char decodestr[]);

# 详细设计

在概要设计的基础上，对每个模块进行内部逻辑处理部分详细设计。下面分别列出各个模块具体实现流程图：

## 添加

定义一个结构体指针向量，每次读取完数据，存入结构体，让后使用push\_back函数将节点插入向量末尾。

## 显示

根据哈夫曼树的建立规则建树，建好以后，遍历该树，记录节点信息。将节点信息输出到屏幕上

## 加密

根据各个节点对应的密码，对输入的字符进行加密操作，并输出加密后的密码。

## 解密

根据遍历树左转为0，右转为1，按照接受的密码进行对树的遍历，每当查询到字符的时候，重新从树根开始遍历。知道密码结束为止。

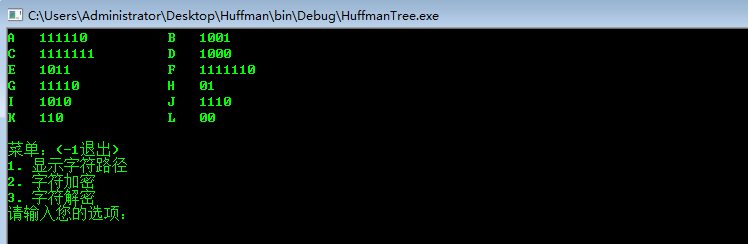
## 退出保存

判断用户输入，当输入结束标志的时候，程序结束

# 运行结果

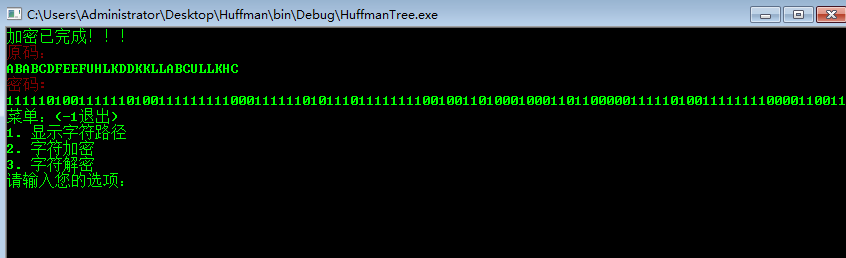
**查询测试：**

输入：1

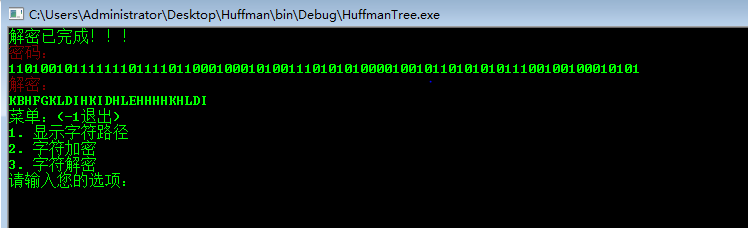


**加密测试：**

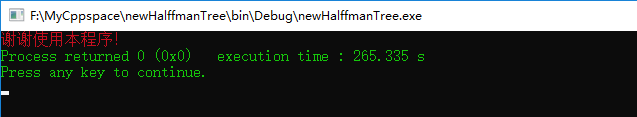
输入：2

**解密测试：**

输入：3

**退出保存函数测试：**

输入：-1



# 调试分析

在执行建树的过程中，每次都会遇到每当建树的函数结束，但是树却没有任何信息，经过查询资料，各种帖子，知道了在函数中申请的空间都是在栈内存中申请的，但是如果需要在函数结束数据仍然保存，那就需要通过new/molloc函数申请堆内存，这样的话在函数结束，内存也不会被释放。

# 心得体会

这次最突出的问题是：代码的复用问题，因为之前建立结构体数组用的是全局变量，本以为在函数操作中会简单，到后来在建立基于单词压缩方法时就出现了复用问题，后来是自己动手改了每个函数参数之间的传递。我觉得这对我是一次警告，警告我以后编程序时要充分考虑代码复用性及函数的封装性。

还有一个地方，是方法的选择问题，在寻找权值最小的两个结点时，我面临很多方法，可以用实验一用过的插入排序把数组按权值排序后直接取前两个，也可以直接遍历两次寻找到最小的，然而这两种方法的时间复杂度都是O（n）,如果再进一步考虑，思考老师给的后面的加分题，完全可以想到用最小堆来优化使时间复杂度降为O（logn），这也是刚学过的知识。基于字符的压缩所用的存储空间固定，但由于每个字符都要编码，相比基于单词的压缩在文章字数较少时更加适用；基于单词的压缩适用于专业类文章或者其他相关术语等有单词出现频率高的文章的压缩，这样编码的长度会减少很多。对于不同的文章，应综合考虑，或者在大量统计的基础上得出普适规律。无论是基于字符的压缩还是基于单词的压缩如果应用K叉树，则空间利用率都会提高，而应用最优堆则会使时间复杂度降低以及整个变量的控制。通过学习课本上的基础编码方法，再加上老师所讲的内容，整理修改后得到这个编码系统。

通过本次试验，掌握了树和哈夫曼树的基本操作，以及各个程序的算法。也复习了前面所学习的参数调用和控制变量范围。

这次课程设计，在编辑中犯了不应有的错误，统计字符时忘记了应该怎样保存数据，对文件的操作也很生疏，在不断的分析后明确并改正了错误和疏漏，是程序有了更高的质量。  总的来说，不仅是实验的结果，更重要的是过程和思考，是我学到了很多的知识，真的是受益匪浅。

# 参考文献

1. 严蔚敏, 吴伟民. 数据结构（C语言版）[M]. 北京:清华大学出版社, 2012.
2. 严蔚敏, 吴伟民. 数据结构题集（C语言版）[M]. 北京:清华大学出版社, 2011.
3. 谭浩强. C语言程序设计[M]. 北京:清华大学出版社, 2014.
4. [美]Paul Deitel, Harvey Deitel著, 张引等译. C++大学教程（第九版）[M].北京: 电子工业出版社, 2016.
5. 郭炜. 新标准C++程序设计 [M]. 北京: 高等教育出版社, 2016.