学生学号	0121710880223	实验课成绩	

武漢理工大學

学 生 实 验 报 告 书

实验课程名称	数据结构
开 课 学 院	计算机科学与技术学院
指导教师姓名	胡燕
学 生 姓 名	刘佳迎
学生专业班级	计算机类 m1702 班

2018 -- 2019 学年 第 一 学期

实验教学管理基本规范

实验是培养学生动手能力、分析解决问题能力的重要环节；实验报告是反映实验教学水平与质量的重要依据。为加强实验过程管理，改革实验成绩考核方法，改善实验教学效果，提高学生质量，特制定实验教学管理基本规范。

1、 本规范适用于理工科类专业实验课程，文、经、管、计算机类实验课程可根据具体情况参照执行或暂不执行。

2、 每门实验课程一般会包括许多实验项目，除非常简单的验证演示性实验项目可以不写实验报告外，其他实验项目均应按本格式完成实验报告。

3、 实验报告应由实验预习、实验过程、结果分析三大部分组成。每部分均在实验成绩中占一定比例。各部分成绩的观测点、考核目标、所占比例可参考附表执行。各专业也可以根据具体情况，调整考核内容和评分标准。

4、 学生必须在完成实验预习内容的前提下进行实验。教师要在实验过程中抽查学生预习情况，在学生离开实验室前，检查学生实验操作和记录情况，并在实验报告第二部分教师签字栏签名，以确保实验记录的真实性。

5、 教师应及时评阅学生的实验报告并给出各实验项目成绩，完整保存实验报告。在完成所有实验项目后，教师应按学生姓名将批改好的各实验项目实验报告装订成册，构成该实验课程总报告，按班级交课程承担单位（实验中心或实验室）保管存档。

6、 实验课程成绩按其类型采取百分制或优、良、中、及格和不及格五级评定。

附表：实验考核参考内容及标准

	观测点	考核目标	成绩组成
实验预习	1. 预习报告 2. 提问 3. 对于设计型实验，着重考查设计方案的科学性、可行性和创新性	对实验目的和基本原理的认识程度，对实验方案的设计能力	20%
实验过程	1. 是否按时参加实验 2. 对实验过程的熟悉程度 3. 对基本操作的规范程度 4. 对突发事件的应急处理能力 5. 实验原始记录的完整程度 6. 同学之间的团结协作精神	着重考查学生的实验态度、基本操作技能；严谨的治学态度、团结协作精神	30%
结果分析	1. 所分析结果是否用原始记录数据 2. 计算结果是否正确 3. 实验结果分析是否合理 4. 对于综合实验，各项内容之间是否有分析、比较与判断等	考查学生对实验数据处理和现象分析的能力；对专业知识的综合应用能力；事实求实的精神	50%

实验课程名称：数据结构

实验项目名称	实验二			实验成绩	
实 验 者	刘佳迎	专业班级	计算机类 m1702 班	组　　别	
同 组 者				实验日期	2018 年 12 月 31 日

一部分：实验预习报告（包括实验目的、意义，实验基本原理与方法，主要仪器设备及耗材，实验方案与技术路线等）

实验目的及方案:

哈夫曼树又称最优二叉树，是带权路径长度最短的树，可用来构造最优编码，用于信息传输、数据压缩等方面，是一种应用广泛的二叉树。为了尽可能使编码精炼，故使用哈夫曼编码来进行最优编码。

哈夫曼树的节点结构

```
typedef struct
{
    int weight;
    int parent, lchild, rchild;

}HTNode,*HuffmanTree;
```

哈夫曼树的建立

1):初始化:根据给定的 n 个权值(W1,W2,…,Wn),构造 n 棵二叉树的森林集合 F={T1,T2,…,Tn}，其中每棵二叉树 Ti 只有一个权值为 Wi 的根节点，左右子树均为空。
2):找最小树并构造新树：在森林集合 F 中选取两棵根的权值最小的树做为左右子树构造一棵新的二叉树，新的二叉树的根结点为新增加的结点，其权值为左右子树的权值之和。
3):删除与插入：在森林集合 F 中删除已选取的两棵根的权值最小的树，同时将新构造的二叉树加入到森林集合 F 中。
4):重复 2)和 3)步骤，直至森林集合 F 中只含一棵树为止，这颗树便是哈夫曼树，即最优二叉树。由于 2)和 3)步骤每重复一次，删除掉两棵树，增加一棵树，所以 2)和 3)步骤重复 n-1 次即可获得哈夫曼树。

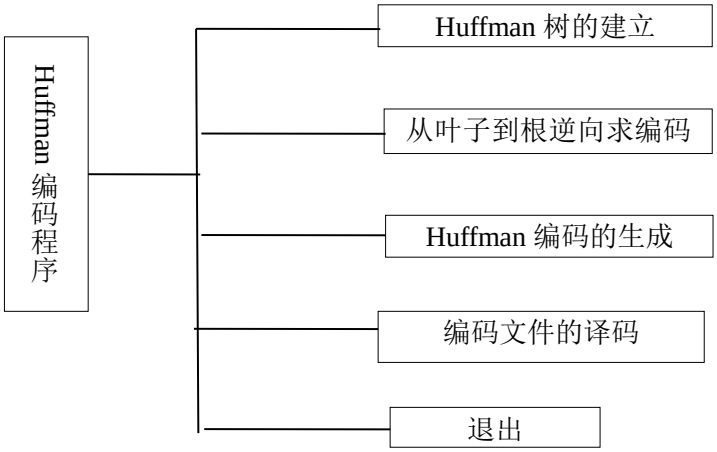
哈夫曼树的编码

为每个字符求解哈夫曼编码，从叶子到根逆向求解每个字符的哈夫曼编码。

实验主要实现函数：

1. 统计字符串中字符的种类以及各类字符的个数的函数
2. 构造 Huffman 树的函数
3. Huffman 编码的函数
4. 建立正文的编码文件的函数
5. 代码文件的译码函数
6. 主函数

实验功能模块



使用说明

1. 运行环境：vs2017

2. 首先选择主控菜单中的操作 1 ，即建表，然后进行其它操作.

第二部分：实验过程记录（可加页）（包括实验原始数据记录，实验现象记录，实验过程发现的问题等）

实验数据

1.哈夫曼树的构造和编码

//实验具体代码如下

//model.h

```
#pragma once
typedef struct
{
    int weight;
    int parent, lchild, rchild;

}HTNode,*HuffmanTree;
typedef char** HuffmanCode;
void setHuffmantree(HuffmanTree &HT, int *w, int n);//构建哈弗曼树
void select(HuffmanTree HT, int n, int &s1, int &s2);//在HT[1~I-1】选择parent为零且为最小的两个数，序号分别为s1,s2
void Huffcoding(HuffmanTree &HT, HuffmanCode &HC, int *w, int n);
void setHuffmancode(HuffmanTree &HT, HuffmanCode &HC, int *w, int n);
```

//main.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<stdlib.h>
#include"model.h"
#include<string.h>
int main()
{
    int n; char *str; int *w; char c_tmp;
    HuffmanCode HC; HuffmanTree HT;
    printf("输入字符个数");
    scanf_s("%d", &n);
    while ((c_tmp = getchar() != '\n') && c_tmp != EOF);
    str = (char *)malloc(n * sizeof(char));
    w = (int *)malloc(n * sizeof(int));
    printf("输入%d个权重", n);
    for (int i = 0; i < n; i++)
    {
        scanf_s("%d", w + i);
    }
    do { c_tmp = getchar(); } while (c_tmp != '\n'&&c_tmp != EOF);

    Huffcoding(HT,HC,w,n);
```

```
    printf("赫夫曼树编码如下:\n\n");
    for (int i = 1; i <= n; i++)
    {
        printf("%d :%s\n", HT[i].weight, HC[i]);
    }
    return 0;
}
void Huffcoding(HuffmanTree &HT, HuffmanCode &HC, int *w, int n)//构建哈夫曼编码
{
    setHuffmantree(HT, w, n);
    setHuffmancode(HT, HC, w, n);
}
void setHuffmantree(HuffmanTree &HT, int *w, int n)//构建哈弗曼树
{
    if (n <= 1) return;
    int m = 2 * n - 1; int s1 = 0, s2 = 0;
    HT = (HuffmanTree)malloc((m + 1) * sizeof(HTNode));//不用零号单元
    HuffmanTree p; int i ;
    HT->lchild = HT->parent = HT->rchild = HT->weight = 0;
    for (p = HT+1, i = 1; i <= n; i++, p++, w++)
    {
        p->weight = *w;
        p->lchild = 0; p->parent = 0; p->rchild = 0;
    }
    for (1; i <= m; i++,p++)
    {
        p->weight =0;
        p->lchild = 0; p->parent = 0;  p->rchild = 0;
    }
    for (i = n + 1; i <= m; i++)
    {
        select(HT, (i - 1), s1, s2);
        HT[s1].parent = i; HT[s2].parent = i;
        HT[i].lchild = s1; HT[i].rchild = s2;
        HT[i].weight= HT[s1].weight + HT[s2].weight;
    }
/*
    for (i = 1; i <= m; i++)
    {
        printf(" %d %d %d %d ", HT[i].weight, HT[i].lchild, HT[i].rchild, HT[i].parent);
        printf("\n");
    }*/
```

```
}
void select(HuffmanTree HT,int nn,int &s1,int &s2)//在HT[1~I-1】选择parent为零且为最小的两个数，序号分别为s1,s2
{
    int i=1, v1, v2;
    v2 = 32767;
    v1 = 32767;
    s2 = 0;
    s1 = 0;
    for (i = 1; i <= nn; i++)
    {
        if (HT[i].parent == 0)
        {
            if (HT[i].weight < v1)
            {
                s2 = s1; v2 = v1;
                v1 = HT[i].weight;
                s1 = i;
            }
            else if (HT[i].weight < v2)
            {
                v2 = HT[i].weight;
                s2 = i;
            }
        }
    }
    /*
    for (i = 1; i <= n; i++)
    printf(" %d ", HT[i].weight);
    printf("\n");
    */
}
void setHuffmancode(HuffmanTree &HT, HuffmanCode &HC, int *w, int n)
{
    int f, c, i;
    HC = (HuffmanCode)malloc((n + 1) * sizeof(char *)); //分配n个字符编码的头指针向量
    char * cd = (char *)malloc(n * sizeof(char));        //分配求编码的工作空间
    cd[n - 1] = '\0';                                    //编码结束符
for ( i = 1; i <= n; ++i) {
        int start = n - 1;
        for ( c = i, f = HT[i].parent; f != 0; c = f, f = HT[f].parent)
    //从叶子到根逆向求编码
```

```
            if (HT[f].lchild == c)
                cd[--start] = '0';
            else
                cd[--start] = '1';
        HC[i] = (char *)malloc((n - start) * sizeof(char));
        strcpy(HC[i], &cd[start]);      //
    }
    free(cd);
}
```

实验中出现的问题

在 scanf 函数中读入字符时，会将缓冲区的剩余字符读入导致数据异常
解决方法：

- 第一种方法： fflush(stdin);

  在后来的更新的标准中被删去了，并不具有可移植性。

- 第二种方法：调用 setbuf(stdout，buf);

  如果 buf 是一个大小适当的字符数组，那么：

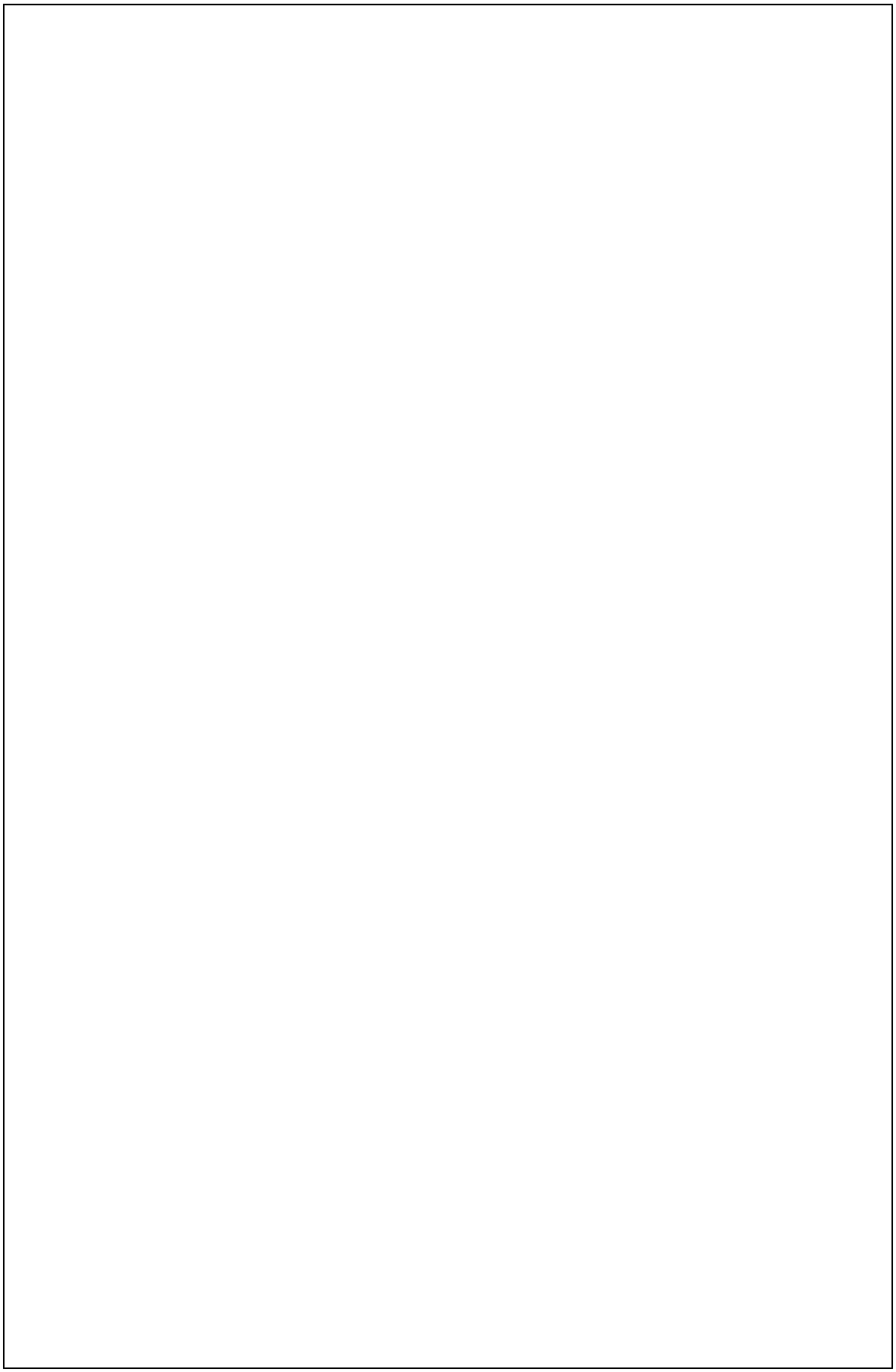
  语句将通知输入/输出库，所有写入到 stdout 的输出都应该使用 buf 作为输出缓冲区，直到 buf 缓冲区被填满或者程序员直接调用 fflush（译注：对于由写操作打开的文件，调用 fflush 将导致输出缓冲区的内容被实际地写入该文件），buf 缓冲区中的内容才实际写入到 stdout 中。缓冲区的大小由系统头文件<stdio.h>中的 BUFSIZ 定义。

  当 buf 是 NULL 时，即直接关闭缓冲区，我们知道缓冲区的这个东西的存在就是为了提高计算机的运算速度的（具体原因就是计算机对缓冲区的操作快于对磁盘的操作），所以关闭缓冲区的后果可想而知。

- 第三种方法：

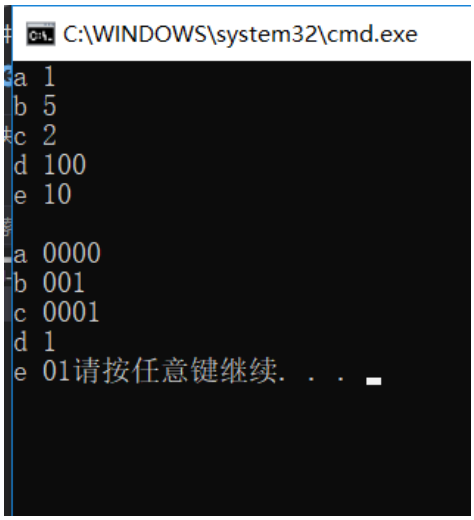
  使用 getchar()函数可以有效地清除缓冲区的内容，并且具有可移植性。

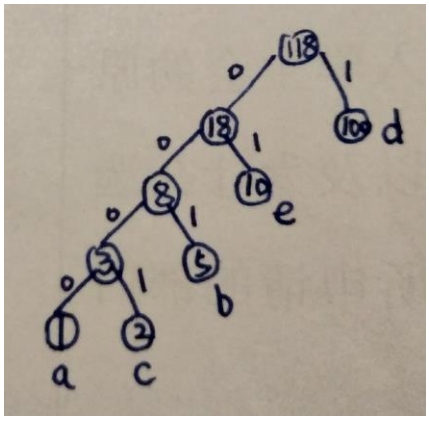
教师签字_____



第三部分　结果与讨论（可加页）

一、实验结果分析（包括数据处理、实验现象分析、影响因素讨论、综合分析和结论等）





二、小结、建议及体会

此次实验一开始老师讲的时候，又要创建哈夫曼树，又要给哈夫曼树编码，期间调试时出现了一些小失误，但是经过断点监视来进行调试，终于排查了一些小 bug，编写代码一定要细心，要勤思考。

三、思考题

哈夫曼编码的解码问题：从哈夫曼树的根节点出发，按字符'0'或'1'确定找其左孩子或右孩子，直至找到叶子节点即可，便求得该字串相应的字符。

```
void HuffmanTranslateCoding(HuffmanTree HT, int n, char* ch,char *N)
{//译码过程
    int m = 2 * n - 1;
    int i, j = 0;

    printf("After Translation:");
    while (ch[j] != '\0')//ch[]:你输入的要译码的0101010串
```

```
    {
        i = m;
        while (0 != HT[i].lchild && 0 != HT[i].rchild)//从顶部找到最下面
        {
            if ('0' == ch[j])//0 往左子树走
            {
                i = HT[i].lchild;
            }
            else//1 往右子树走
            {
                i = HT[i].rchild;
            }
            ++j;//下一个路径
        }
        printf("%c", N[i - 1]);//打印出来
    }
    printf("\n");

    }
```