目录

关于使用说明：

六个代码中有四个由本人亲手编写，使用时（如果可以）请为笔者的GPA送上小小的祝福

数据结构的算法题目其实都很经典，如果笔者的算法有任何问题，不要着急，CSDN网站上几乎都可以搜索到原题

考虑到需要写一个实验报告，请使用者最好自己起码写一个算法并独立完成实验报告（本文件中并没有附上实验报告）

期末考试与敲代码能力关系不大，请使用时不要有太大负罪感（应付期末方法详见“数据结构水课指导（仅限ypb老师）”）

1.关键词检索（注意要使用KMP算法）

```
#include <stdio.h>
#include <stdlib.h>

char filename[50];
char keyword[20];
char tempstr[1024];
int next[20];
```

```
int length;
void getNext()
{
int j=0,k=0;next[0]=-1;
while(j<=length-1)
if(k==-1||next[j]==next[k])
{
j++;k++;
if(next[j]!=next[k])next[j]=k;
else next[j]=next[k];
}
else k=next[k];
}
int Length(char S[])
{
int m;
for(m=0;;m++)
{
if(S[m]=='\0')return m;
}
}
int index()
{
int i=0,j=0,k=0,flag=1;

FILE *fp;
if((fp=fopen(filename,"r"))==NULL)
{
printf("打开文件%s出错\n",filename);
return 0;
}
while(fgets(tempstr,1024,fp)!=NULL)
{k++;i=0;j=0;
while(j<Length(keyword)&&i<Length(tempstr))
{
if(j==-1||tempstr[i]==keyword[j])  {i++;j++;}
else j=next[j];
if(j==Length(keyword))
{
printf("第%d行第%d列\n",k,i-length+1);j=0;}
}



//if(tempstr[i+length-1]=='\0')break;
```

```
}


 fclose(fp);
}

int main()
{
printf("请分别输入文件名以及关键词(必须在结尾加斜杠n):\n");
gets(filename);


gets(keyword);
length=Length(keyword);
getNext();
index();
}
```

2.管道问题

```
#include "iostream"
#include "stdlib.h"
#define MAX_VERTEX_NUM 20
typedef float WeightType;
typedef struct ArcNode{
int adjvex;
WeightType weight;
struct ArcNode*nextarc;
}ArcNode;
typedef struct VertexNode{
char data;
ArcNode*firstarc;
}VertexNode,AdjList[MAX_VERTEX_NUM];
typedef struct{
AdjList vertices;
int vexnum,arcnum;
int kind;
}ALGraph;
int LocateVex(ALGraph G,char v)
{
int i;
for(i=0;i<G.vexnum;i++)
```

```
{
if(G.vertices [i].data == v)
return i;
}
return -1;
}
void CreateGraph(ALGraph &G)
{
int i,j,k;
char vi,vj;
WeightType weight;
ArcNode*p,*q;
std::cout<<"请输入顶点个数,边数和图的类型:\n";
std::cin>>G.vexnum >>G.arcnum >>G.kind ;
for(i=0;i<G.vexnum ;i++)
{
std::cout<<"请输入各个顶点:\n";
std::cin>>G.vertices[i].data ;
G.vertices[i].firstarc=NULL;
 }
 for(k=0;k<G.arcnum ;k++)
 {
        std::cout<<"请输入两顶点和其边的权值:\n";
        std::cin>>vi>>vj>>weight;
        i=LocateVex(G,vi);
        j=LocateVex(G,vj);
        p=(ArcNode*)malloc(sizeof(ArcNode));
p->adjvex=j;
p->weight=weight;
p->nextarc=G.vertices [i].firstarc;
G.vertices [i].firstarc=p;
if(G.kind == 2)
{
q=(ArcNode*)malloc(sizeof(ArcNode));
q->adjvex=i;
q->weight=p->weight;
q->nextarc=G.vertices [j].firstarc;
G.vertices [j].firstarc=q;
}                                                                                                    }
 }
int MinEdge(WeightType lowcost[],int vexmun)
{
int i,k;
WeightType j;
```

```
k=0;
while(lowcost[k]==0)
{
k++;
}
j=lowcost[k];
for(i=k+1;i<vexmun;i++)
{
if(lowcost[i]!=0&&lowcost[i]<j)
{
j=lowcost[i];
k=i;
}
}
return k;
}
void Prim(ALGraph G,int v0, int adjvex[])
{
WeightType lowcost[MAX_VERTEX_NUM];
int i,k;
ArcNode*p;
for(i=0;i<G.vexnum ;i++)
{
    if (i!=v0)
{
lowcost[i]=999;
adjvex[i]=v0;
 }
 }
 p=G.vertices [v0].firstarc;
 while(p)
 {
        lowcost[G,p->adjvex]=p->weight;
        p=p->nextarc;
 }
 lowcost[v0]=0;
 for(i=0;i<G.vexnum ;i++)
 {
        k=MinEdge(lowcost,G.vexnum );
        if(k>=G.vexnum )
        return;
        std::cout<<"("<<k<<","<<adjvex[k]<<"),"<<lowcost[k]<<'\n';
        lowcost[k]=0;
        p=G.vertices [k].firstarc;
```

```
        while(p)
        {
                if(p->weight<lowcost[p->adjvex])
                {
                        adjvex[p->adjvex]=k;
                        lowcost[p->adjvex]=p->weight;
 }
 p=p->nextarc;
 }
 }
}
int main()
{
int adjvex[MAX_VERTEX_NUM];
ALGraph G;
G.kind =2;
CreateGraph(G);
Prim(G,0,adjvex);
return 0;
}
```

3.哈希表

```
#include<iostream>
#include<string.h>
#include<fstream>
#include<math.h>
#include<iomanip>

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
```

```
#define N 100000

#define HASHSIZE 35
#define   Size      7

using namespace std;

int hash1[60],hash2[60];


typedef unsigned int uint;
typedef struct Node{
const char* key;
const int *value;
Node *next;
}Node;

class HashTable{
private:
     Node* node[HASHSIZE];
public:

HashTable();
~HashTable();
int hash(const char* key);
Node* lookup(const char* key);
bool install(const char* key,const int* value);
const char* get(const char* key);
void display();
};

HashTable *ht = new HashTable();


HashTable::HashTable(){
for (int i = 0; i < HASHSIZE; ++i)
{
node[i] = NULL;
}
}
HashTable::~HashTable(){
cout<<"\n";
```

```
}
int HashTable::hash(const char* key){

return (int)((*key)-97);
}

Node* HashTable::lookup(const char* ch){
Node *np;
uint index;
index = hash(ch);
//np=node[index];
for(np=node[index];np;np=np->next){
if(!strcmp(ch,np->key))
return np;

}

return NULL;
}

bool HashTable::install(const char* key,const int* value){
uint index;
Node *np;
if(!(np=lookup(key))){
index = hash(key);
np = (Node*)malloc(sizeof(Node));
if(!np) return false;
np->key=key;
np->next = node[index];
node[index] = np;
}
np->value=value;
return true;
}

void HashTable::display(){
Node* temp;
for (int i = 0; i < HASHSIZE; ++i)
{
if(!node[i]){
printf("%d\n",i);
}
else
{
```

```
printf("");
for (temp=node[i]; temp; temp=temp->next)
{
cout<<i<<setw(50)<<temp->key<<endl;
//printf("%d    %s",i,temp->key);
//cout<<i<<setw(10)<<temp->key<<setw(5)<<*(temp->value)<<endl;
//        printf("\n");
}

}
}
}
```

```
static void judege_c(int hash[],const char* ch) {
        Node   *np;
        int x;
        x=ht->hash(ch);
         if(x>=0&&x<=25)
{
        np=ht->lookup(ch);
        //!strcmp(ch,np->key)
        if(np!=NULL)
  if(!strcmp(ch,np->key))
      {hash[*(np->value)]++;

    //cout<<np->key<<"    "<<endl;
 }
       // if(x==8&&ht->lookup(ch))
     }
}
/*
uint HashTable::hash(const char* key){
//        uint hash=0;
//        for (; *key; ++key)
//        {
//                hash=hash*33+*key;
//        }

return ((*key)-97);
}
```

```
*/

/*Node* HashTable::lookup(const char* key){
Node *np;
uint index;
index = hash(key);
for(np=node[index];np;np=np->next){
if(!strcmp(key,np->key))
return np;
}
return null;
}*/

static int deal_data1(char data[]) {
    char ch;
    char filename[50];
    cout<<"the first file:"<<endl;
    cin>>filename;
    ifstream infile(filename,ios::in);
    ofstream outfile("dealt_data1.txt",ios::out);
    if(!infile) {
        cout<<"Open first file error!"<<endl;
        exit(0);
}
    if(!outfile) {
        cout<<"Open dealt_data1 error!"<<endl;
        exit(0);
}

    while(infile.get(ch)) {
        if(ch=='('||ch==')'||ch=='{'||ch=='}'||ch=='['||ch==']'||ch==','||ch=='<'||ch=='>'||ch==';')
            ch=' ';
        outfile.put(ch);
    }
    infile.close();
    outfile.close();
    ifstream infile1("dealt_data1.txt",ios::in);
    if(!infile1) {
        cout<<"open dealt_data1 error!"<<endl;
    }
    int cnt=0;
    while(infile1.get(ch)) {
        data[cnt++]=ch;
    }
```

```
    infile1.close();
    return cnt;
}

static int deal_data2(char data[]) {
    char ch;
    char filename[50];
    cout<<"the second file:"<<endl;
    cin>>filename;
    ifstream infile(filename,ios::in);
    ofstream outfile("dealt_data2.txt",ios::out);
    if(!infile) {
        cout<<"Open second file error!"<<endl;
        exit(0);
}
    if(!outfile) {
        cout<<"Open dealt_data2 error!"<<endl;
        exit(0);
}
    while(infile.get(ch)) {
        if(ch=='('||ch==')'||ch=='{'||ch=='}'||ch=='['||ch==']'||ch==','||ch=='<'||ch=='>'||ch==';')
            ch=' ';
        outfile.put(ch);
    }
    infile.close();
    outfile.close();
    ifstream infile2("dealt_data2.txt",ios::in);
    if(!infile2) {
        cout<<"open dealt_data2 error!"<<endl;
    }
    int cnt=0;
    while(infile2.get(ch)) {
        data[cnt++]=ch;
    }
    infile2.close();
    return cnt;
}

static void calcunum_c(const char data[],const int n,const int num) {
    char ch[100];
    int i,j,x;
    for(i=0; i<n; i++) {
        if(data[i]!=' '&&data[i]!='\n'&&data[i]!='\t') {
            j=0;
```

```
                while(data[i]!=' '&&data[i]!='\n'&&data[i]!='\t') {
                    ch[j]=data[i];
                    i++;
                    j++;
                }
                ch[j]='\0';

                if(num==1)
                   {

 judege_c(hash1,ch);
   }
                else
                 {

      judege_c(hash2,ch);
 }
          }
     }
}

static double possibality(int hash1[],int hash2[])
{
     int i;
     double sum=0,min=0,max=0,pos;
     for(i=0;i<Size;i++)
     {
          sum=sum+(hash1[i])*(hash2[i]);
          min=min+(hash1[i])*(hash1[i]);
          max=max+(hash2[i])*(hash2[i]);
     }
     pos=sum/sqrt(min*max);
     return pos;
}


int main(void)
{

const char* key[]={"void","int","for","char","if","else","while"
                        };
const int value[]={1,2,3,4,5,6,7};
// char key1[7]={'a','a'};
cout<<"the list of hash:       "<<endl<<endl<<endl;
```

```
for (int i = 0; i < Size; ++i)
{
ht->install(key[i],&value[i]);
}
ht->display();

char data1[N];
    char data2[N];
    const int cnt1 = deal_data1(data1);
    const int cnt2 = deal_data2(data2);

    calcunum_c(data1,cnt1,1);
    calcunum_c(data2,cnt2,2);



cout<<"the number of key of first file:"<<endl;
    for(int i=0; i<Size; i++){
        cout<<setw(10)<<key[i]<<":"<<hash1[i+1];
        if((i+1)%5==0)
            cout<<endl;
    }
    cout<<endl;
    cout<<"the number of key of second file:"<<endl;
    for(int i=0; i<Size; i++){
        cout<<setw(10)<<key[i]<<":"<<hash2[i+1];
        if((i+1)%5==0)
            cout<<endl;
    }
cout<<endl;cout<<endl;cout<<endl;
cout<<"两源程序的相似性为: "<<possibality(hash1,hash2)<<endl;
delete ht;
return 0;
}
```

4.huffman编码解码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct{
char character='0';//记录节点的字母
int alaways=-1,rchild=-1,lchild=-1,parent;//左右孩子与双亲的物理指针
char num[20];
int c=0;//判断是否进入树
}Node;
Node node[100];//树的节点存放的数组
int n;//字符集的大小
int gen,o,p=-1;//根的下标
char num1[20],aid11[100],aid22[1024],aid33[100],aid44[1024];//num1用来暂存01编码
int min()
{
int i,min;
for(i=0;i<=gen;i++)
{
if(node[i].c==0){min=i;break;
}
}
for(i=min+1;i<=gen;i++)
{
if(node[i].c==1)continue;
if(node[i].alaways<node[min].always)min=i;
}
node[min].c=1;
return min;
```

```
}
void Initialization()
{
printf("请输入字符集的大小n：");scanf("%d",&n);getchar();gen=n-1;
int i,j,k,he,l=n,temp;
for(i=0;i<n;i++)
{
printf("请输入第%d个字符的类型以及权值：",i+1);
scanf("%c%d",&node[i].character,&node[i].alaways);getchar();
printf("%c%d\n",node[i].character,node[i].alaways);//检验用
}//前i+1个都是字符以及权值，接下来建立树
j=min();///*********************************************
for(i=0;i<n-1;i++)
{
k=min();node[l].alaways=node[j].alaways+node[k].alaways;//jk是两个子叶
if(node[k].alaways<node[j].alaways) {temp=k;k=j;j=temp;}//进行交换
node[l].lchild=j;node[l].rchild=k;
j=min();
l++;gen=l-1;
}//完成了初始的构建树，现在总共2*n-1=l个节点， 所以l--即可表示根
l--;gen=l;o=l;//现在l表示根
}
void bianma(int aim)
{
p++;
if(node[aim].lchild!=-1)
{
num1[p]='0';
bianma(node[aim].lchild);
p--;
num1[p]='1';
bianma(node[aim].rchild);
p--;
}
else{

num1[p]='\0';
strcpy(node[aim].num,num1);
printf("%c    %s\n",node[aim].character,node[aim].num);}
}
void   goujianma()//aid11来存放英语语句 aid22存放编码后的数字
{
int i,j;aid22[0]='\0';
printf("请输入需要编译的语句：");fflush(stdin);
```

```
gets(aid11);
for(i=0;aid11[i]!='\0';i++)
{
for(j=0;j<n;j++)
{
if(aid11[i]==node[j].character)strcat(aid22,node[j].num);
}
}
printf("%s编码后的二进制为：\n%s\n",aid11,aid22);
FILE *fp;
if( (fp=fopen("C:\\Users\\asus\\Desktop\\CodeFile.txt","w+")) == NULL )
{
        printf("Fail to open file!\n");
        exit(0);   //退出程序（结束程序）
}
fprintf(fp,"%s",aid22);
fclose(fp);
}

void jiema()//aid33[100],aid44[1024]来存放语句与二进制
{aid33[0]='\0';
char shiyong[2];shiyong[1]='\0';
int i,j,k=0,l,flag=0;
FILE *fp;
if( (fp=fopen("C:\\Users\\asus\\Desktop\\CodeFile.txt","r")) == NULL )
{
        printf("Fail to open file!\n");
        exit(0);   //退出程序（结束程序）
}
fgets(aid44,1024,fp);
for(i=0;aid44[i]!='\0';i++)//把aid44中的取出到num1中去比较
{
for(l=k;l<i+1;l++)
{
num1[l-k]=aid44[l];//k要变
}num1[l-k]='\0';

for(j=0;j<n;j++)
{
if(strcmp(num1,node[j].num)==0)
{shiyong[0]=node[j].character;
strcat(aid33,shiyong);k=i+1;}
}
}
```

```
printf("%s解码后是：\n%s",aid44,aid33);
fclose(fp);



}
void dayin()
{
int i=0,j;
while(aid22[i]!='\0')
{
for(j=0;j<50;j++,i++)
 {
 if(aid22[i]=='\0')break;
 printf("%c",aid22[i]);
 }
printf("\n");
 }
}
void dayinhuffman()
{
int i;printf("下标-左子树下标-右子树下标\n");
for(i=0;node[i].alaways!=-1;i++)printf("%d    %d                %d\n",i,node[i].lchild,node[i].rchild) ;
}
int main()
{
int x=-1,hui;
while(x){
printf("0.结束程序\n1.初始化\n2.编码\n3.解码\n4.打印代码文件\n5.打印huffman树\n");
scanf("%d",&x);
switch(x)
{
case 1:Initialization();bianma(o);break;
case 2:goujianma();break;
case 3:jiema();break;
case 4:dayin();break;
case 5:dayinhuffman();break;
default:break;
}
scanf("%d",&hui);
system("cls");
}
}
```

5.停车场

```
#include <stdio.h>
#include <stdlib.h>
typedef struct{
int *car,*time;

}SqStack;SqStack stop;
int top=-1;

typedef struct Node{
int Car;
struct Node *next;
}list,*List;

List Creat(int n,int m)
{
stop.car=new int[n];
stop.time=new int[n];
int i;
List head=(List)malloc(sizeof(list));
head->next=NULL;head->Car=0;
for(i=1;i<n;i++)
{
List p=(List)malloc(sizeof(list));
p->Car=0;p->next=head;head=p;
}
return head;
}

void Delete(List head,int n)
{
delete []stop.car;delete []stop.time;
int i;List p;
for(i=0;i<n;i++)
{
p=head;
head=head->next;
free(p);

}
}

void Arrival(List head,int n,int m,int carID,int time)//++++++++++++++++++++++++++
{
int i;
```

```
if(top==(n-1))
{
for(i=0;i<m;i++)
{
if(head->Car==0)
{
head->Car=carID;
printf("%d车在候车场%d号位\n",carID,i+1);
break;
}
head=head->next;
}
}
else
{
top++;
stop.car[top]=carID;stop.time[top]=time;
printf("%d车在停车场%d号位\n",carID,top+1);
}
}
List Leave(List head,int n,int m,int carID,int time)//++++++++++++++++++++++++++
{
int i,j;top--;
for(i=0;i<n;i++)
{
if(stop.car[i]==carID)break;
}
printf("%d车在停车场中停了%d分钟收费%f元\n",carID,time-stop.time[i],(float)(time-stop.time[i])/30);
for(j=i;j<n-1;j++)
{
stop.car[j]=stop.car[j+1];
stop.time[j]=stop.time[j+1];
}stop.car[n-1]=0;stop.time[n-1]=0;
List q=head;
if(head->Car!=0)
{
top++;
stop.car[n-1]=head->Car;
stop.time[n-1]=time;
List p=head;head=head->next;q=head;
free(p);p=(List)malloc(sizeof(list));
while(1)
{
if(head->next==NULL)
```

```
{
head->next=p;p->next=NULL;return q;
}
head=head->next;

}
}
return q;
}
void ShowStop(int n)
{
printf("正在停车的车牌号从里到外为：\n");
int i;
for(i=0;i<n;i++)
{
if(stop.car[i]!=0)
printf("%d\n",stop.car[i]);
}
}
void ShowList(List head,int m)
{
int i;
printf("正在排队的车牌号从前到后为：\n");
for(i=0;i<m;i++)
{
if(head->Car!=0)
{
printf("%d\n",head->Car);
head=head->next;
 }
}
}
int main()
{

int n,m,p;List head;
printf("请按顺序输入停车场容量n、队伍容量m、停车价格p：");
scanf("%d%d%d",&n,&m,&p);
getchar();
head=Creat(n,m);
int carID,time;
char state;
while(1)
{
```

```
printf("请按顺序输入状态、车牌号、时间：");
scanf("%c%d%d",&state,&carID,&time);getchar();
switch(state)
{
case 'A':Arrival(head,n,m,carID,time);break;
case 'D':head=Leave(head,n,m,carID,time);break;
case 'P':ShowStop(n);break;
case 'W':ShowList(head,m);break;
case 'E':break;
default:printf("error\n"); break;
}
if(state=='E'){printf("结束\n"); break;}
}
Delete(head,n);
}
```

6.约瑟夫环

```
#include <stdio.h>
#include <stdlib.h>
typedef struct LNode
{
int num;
int qpy;
struct LNode *next;
}LNode,*Linklist;

Linklist CreatList(int n)
{
int i;
Linklist head,p,q;
head=(Linklist)malloc(sizeof(LNode));
p=head;
for(i=1;i<n;i++)
{
q=(Linklist)malloc(sizeof(LNode));
p->next=q;q->next=NULL;p=q;q=NULL;
}
p->next=head;
return head;
}

void PutInqpy(int n,Linklist p)
{
```

```
int i;
for(i=1;i<=n;i++)
{
p->num=i;
scanf("%d",&p->qpy);
p=p->next;
}
}

void Delete(int n,int m,Linklist head)
{
int i,j,k,o=n;
Linklist p,q;

for(i=1;i<o;i++)
{
if(m!=n)m=m%n;
if(m==1)
{
q=head;p=head;
head=head->next;
m=q->qpy;
printf("%d ",q->num);
for(;;)
{

if(p->next==q)break;
p=p->next;
}
p->next=head;
delete q;
}
else
{
q=head;p=head;
for(j=1;j<m;j++)q=q->next;
for(;;)
{

if(p->next==q)break;
p=p->next;
}p->next=p->next->next;
m=q->qpy;
printf("%d ",q->num);
```

```
head=q->next;
delete q;
}
n--;
}
printf("%d ",head->num);
delete head;

}

int main()
{
int n,m;
Linklist head;
printf("请输入总人数");
scanf("%d",&n);
printf("请输入初始密码");
scanf("%d",&m);
head=CreatList(n);//完成了循环链表的创建
printf("请输入每个人的密码：");
PutInqpy(n,head);
Delete(n,m,head);
return 0;
}
```