例子

- 有向图

6 11
A 1 3 4
B
C 0 3 4 5
D
E 1 3
F 1 4

- 有向网图

6 11
A 1(6) 3(2) 4(3)
B
C 0(2) 3(1) 4(2) 5(2)
D
E 1(3) 3(2)
F 1(2) 4(2)

- 无向图

6 11
A 1 2 3 4
B 0 4 5
C 0 3 4 5
D 0 2 4
E 0 1 2 3 5
F 1 2 4

- 无向网图

6 11
A 1(6) 2(2) 3(2) 4(3)
B 0(6) 4(3) 5(2)
C 0(2) 3(1) 4(2) 5(2)
D 0(2) 2(1) 4(2)
E 0(3) 1(3) 2(2) 3(2) 5(2)
F 1(2) 2(2) 4(2)

邻接矩阵初始化图

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

```c
#include<string.h>
typedef struct _iobuf FILE;
#define maxsize 100

typedef struct  //定义图结构
{
    int n,e;        //n是点，e是边
    char arc[maxsize];
    int edge[maxsize][maxsize];
}gragh;
char* fun(char *str)     //去除字符串中的空格
{
    int i=0,j=0;
    char* res = (char*)malloc(sizeof(char)*strlen(str));
 //动态分配空间
    for(;str[i]!='\0';++i)
    {
        if(str[i]==' '||str[i]=='\n')
            continue;
        res[j++] = str[i];
    }
    return res;
}
void output(gragh *g)       //输出图
{
    printf("This is your map:\npoints:\t%d\tedges:\t%d",g->n,g->e);
    for(int i=0;i<g->n;++i)
    {
        printf("\n%c:\t",g->arc[i]);
        for(int j=0;j<g->n;++j)
        {
            if(g->edge[i][j]==1)
            {
                printf("%c--%c\t",g->arc[i],g->arc[j]);
            }
        }
    }
}
int main()
{
    gragh g;        //int a;int* a;(四个字节大小的空间，然后把空间名字改为a，然后空间存储的是地址，存储的地
    int i,j;
    char str[100],ch;      //用来向txt中读取一行数据
    FILE *fp = fopen("../graph_test.txt","r");
    if(fp==0)       //0 和 null 的区别是啥  如果表示对错，
    {
        printf("Open file error.\n");
```

```
        return 0;
    }
    fscanf(fp,"%d   %d ",&g.n,&g.e);      //读取点和边
    i = 0;
    while(!feof(fp))    //没有到达文件末尾的情况
    {
        fgets(str,100,fp);       //读取一行数据
        char *c = fun(str);
        int k = 0;
        g.arc[i] = c[k++];         //输入点
        for(;c[k]!='\0';++k)
        {
            j = c[k]-'0';
            g.edge[i][j] = 1;        //输入边
        }
        ++i;
    }
    output(&g);          //输出边并验证

    return 0;
}

```

邻接矩阵初始化网

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct _iobuf FILE;
#define maxsize 100

typedef struct  //定义图结构
{
    int n,e;
    char arc[maxsize];
    int edge[maxsize][maxsize];
}gragh;
char* fun(char *str)     //去除字符串中的空格和括号
{
    int i=0,j=0;
    char *res = (char*)malloc(sizeof(char)*strlen(str));
    for(;str[i]!='\0';++i)
    {
        if(str[i]==' '||str[i]=='('||str[i]==')'||str[i]=='\n')
continue;
        res[j++] = str[i];
    }
    return res;
}
```

```
void output(gragh *g)       //输出图
{
    printf("This is your map:\npoints:\t%d\tedges:\t%d",g->n,g->e);
    for(int i=0;i<g->n;++i)
    {
        printf("\n%c:\t",g->arc[i]);
        for(int j=0;j<g->n;++j)
        {
            if(g->edge[i][j]!=0)
            {
                printf("%c--%c(power=%d)\t",g->arc[i],g->arc[j],g->edge[i][j]);
            }
        }
    }
}
int main()
{
    gragh g;
    int i,j;
    char str[100],ch;      //用来向txt中读取一行数据
    FILE *fp = fopen("../power_graph.txt","r");
    if(fp==0)
    {
        printf("Open file error.\n");
        return 0;
    }
    fscanf(fp,"%d   %d ",&g.n,&g.e);     //读取点和边

    i = 0;
    while(!feof(fp))    //没有到达文件末尾的情况
    {
        fgets(str,100,fp);      //读取一行数据
        char *c = fun(str);
        int k = 0;
        g.arc[i] = c[k++];         //输入点
        for(;c[k]!='\0';++k)
        {
            j = c[k++]-'0';
            g.edge[i][j] = c[k]-'0';        //输入边权值
        }
        ++i;
    }
    output(&g);         //输出边并验证

    return 0;
}

```

邻接表初始化图

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define maxsize 100
typedef struct _iobuf FILE;

typedef struct node         //结点
{
    int adjvex;
    struct node *next;
}node;
typedef struct   //头结点
{
    char vex;
    node* next;
}vexNode;
typedef struct      //图
{
    int n,e;
    vexNode adjlist[maxsize];
}graph;

char * fun(char *str)           //去除字符串中的空格
{
    int i=0;
    char *ch = (char*)malloc(sizeof(char)*strlen(str));
    for(int j=0;str[j]!='\0';++j)
    {
        if(str[j]==' '||str[j]=='\n')     continue;
        ch[i++] = str[j];
    }
    ch[i] ='\0';
    return ch;
}
void output(graph *g)       //输出图
{
    int i,j;
    printf("This is your map:\npoints:%d\tarcs:%d",g->n,g->e);
    for(i=0;i<g->n;++i){
        printf("\n%c:\t",g->adjlist[i].vex);
        node *temp = g->adjlist[i].next;
        while (temp)
        {
            printf("%c--%c\t",g->adjlist[i].vex,g->adjlist[temp-
>adjvex].vex);
            temp = temp->next;
        }
```

```
    }
}
int main()
{
    FILE *fp = fopen("../graph_test.txt","r");
    if(fp==0)
    {
        printf("fail to open file.\n");
        return 0;
    }
    graph g;
    char s[100];
    int i=0,j;
    node *temp;
    fscanf(fp,"%d   %d ",&g.n,&g.e);
    for(j=0;j<g.n;++j)      //初始化
        g.adjlist[j].next = 0;
    while(!feof(fp))        //一行一行的开始赋值进去
    {
        fgets(s,100,fp);
        char *str = fun(s);
        int k=0;         //k用来读取字符串
        g.adjlist[i].vex =  str[k++];
        for(;str[k]!='\0';++k)
        {
            temp = (node*)malloc(sizeof(node));      //新建一个结点
            temp->adjvex = str[k]-'0';
            temp->next = 0;
            temp->next = g.adjlist[i].next;           //头插法
            g.adjlist[i].next = temp;
        }
        ++i;
    }
    output(&g);

    return 0;
}


```

邻接表初始化有向权值图

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define maxsize 100
typedef struct _iobuf FILE;

typedef struct node         //结点
```

```
{
    int adjvex;
    int power;                //网结构得加上权值
    struct node *next;
}node;
typedef struct   //头结点
{
    char vex;
    node* next;
}vexNode;
typedef struct      //图
{
    int n,e;
    vexNode adjlist[maxsize];
}graph;

char * fun(char *str)            //去除字符串中的空格和括号
{
    int i=0;
    char *ch = (char*)malloc(sizeof(char)*strlen(str));
    for(int j=0;str[j]!='\0';++j)
    {
        if(str[j]==' '||str[j]=='('||str[j]==')'||str[j]=='\n')
    continue;
        ch[i++] = str[j];
    }
    ch[i] ='\0';
    return ch;
}
void output(graph *g)        //输出图
{
    int i,j;
    printf("This is your map:\npoints:%d\tarcs:%d",g->n,g->e);
    for(i=0;i<g->n;++i)
    {
        printf("\n%c:\t",g->adjlist[i].vex);
        node *temp = g->adjlist[i].next;
        while (temp)
        {
            printf("%c--%c(power=%d)\t",g->adjlist[i].vex,g-
>adjlist[temp->adjvex].vex,temp->power);
            temp = temp->next;
        }
    }
}
int main()
{
    FILE *fp = fopen("../directed_power_graph.txt","r");
    if(fp==0)
```

```
    {
        printf("fail to open file.\n");
        return 0;
    }
    graph g;
    char s[100];
    int i=0,j;
    node *temp;
    fscanf(fp,"%d   %d ",&g.n,&g.e);
    for(j=0;j<g.n;++j)      //初始化
        g.adjlist[j].next = 0;
    while(!feof(fp))        //一行一行的开始赋值进去
    {
        fgets(s,100,fp);
        char *str = fun(s);
        int k=0;         //k用来读取字符串
        g.adjlist[i].vex =  str[k++];
        for(;str[k]!='\0';++k)
        {
            temp = (node*)malloc(sizeof(node));      //新建一个结点
            temp->adjvex = str[k++]-'0';
            temp->power = str[k]-'0';
            temp->next = 0;
            temp->next = g.adjlist[i].next;          //头插法
            g.adjlist[i].next = temp;
        }
        ++i;
    }
    output(&g);

    return 0;
}


```

矩阵转为邻接表

```
#define maxsize 100;

//邻接表存储结构
typedef struct node          //结点
{
    int adjvex;
    int val;         //权值
    struct node *next;
}node;
typedef struct   //头结点
{
    char vex;
```

```
    int val;        //权值
    node* next;
}vexNode;
typedef struct      //图
{
    int n,e;
    vexNode adjlist[maxsize];
}graph1;

//矩阵存储机构
typedef struct  //定义图结构
{
    int n,e;        //n是点，e是边
    char arc[maxsize];
    int edge[maxsize][maxsize];
}gragh2;

// 矩阵转邻接表
graph1* function(graph2* g) {
    if(!g){
        return 0;
    }
    graph1* res = (graph1*)malloc(sizeof(graph1));
    res->n = g->n;
    res->e = g->e;
    // 转换
    for(int i=0;i<g->n;++i){
        res->adjlist[i].vex = g->arc[i];        //转顶点
        res->adjlist[i].next = 0;
        for(int j=0;j<g->n;++j){        //转边
            if(g->edge[i][j]){      //边存在的情况
            node* temp = (node*)malloc(sizeof(node));     //新建一个结点
            temp->adjvex = g->edge[i][j];
            temp->next = 0;
            temp->next = g.adjlist[i].next;          //头插法
            g.adjlist[i].next = temp;
            }
        }
    }
    return res;
}
// 邻接表转矩阵
graph2* function2(graph1* g) {
    if(!g){
        return 0;
    }
    graph2* res = (graph2)malloc(sizeof(graph2));
    res->n = g->n;
```

```
    res->e = g->e;
    //初始化
    for(int i=0;i<g->n;++i){
        for(int j=0;j<g->n;++j){
            res->edge[i][j] = 0;
        }
    }
    //转换
    for(int i=0;i<g->n;++i){
        res->arc[i] = g->adjlist[i].vex;
        node* temp = g->adjlist[i].next;
        while(temp) {
            res.edge[i][temp->adjvex] = temp->val;
            temp = temp->next;
        }
    }
    return res;
}
```