

## 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

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

```

3  #include<string.h>
4  typedef struct _iobuf FILE;
5  #define maxsize 100
6
7  typedef struct  //定义图结构
8  {
9      int n,e;      //n是点, e是边
10     char arc[maxsize];
11     int edge[maxsize][maxsize];
12 }gragh;
13 char* fun(char *str)    //去除字符串中的空格
14 {
15     int i=0,j=0;
16     char* res = (char*)malloc(sizeof(char)*strlen(str));
17     //动态分配空间
18     for(;str[i]!='\0';++i)
19     {
20         if(str[i]==' '||str[i]=='\n')
21             continue;
22         res[j++] = str[i];
23     }
24     return res;
25 void output(gragh *g)    //输出图
26 {
27     printf("This is your map:\npoints:\t%d\tedges:\t%d",g->n,g->e);
28     for(int i=0;i<g->n;++i)
29     {
30         printf("\n%c:\t",g->arc[i]);
31         for(int j=0;j<g->n;++j)
32         {
33             if(g->edge[i][j]==1)
34             {
35                 printf("%c--%c\t",g->arc[i],g->arc[j]);
36             }
37         }
38     }
39 }
40 int main()
41 {
42     gragh g;      //int a;int* a;(四个字节大小的空间,然后把空间名字
43     //改为a,然后空间存储的是地址,存储的地
44     int i,j;
45     char str[100],ch;    //用来向txt中读取一行数据
46     FILE *fp = fopen("../graph_test.txt","r");
47     if(fp==0)    //0 和 null 的区别是啥 如果表示对错,
48     {
49         printf("Open file error.\n");

```

```

49     return 0;
50 }
51 fscanf(fp,"%d %d",&g.n,&g.e);    //读取点和边
52 i = 0;
53 while(!feof(fp))    //没有到达文件末尾的情况
54 {
55     fgets(str,100,fp);    //读取一行数据
56     char *c = fun(str);
57     int k = 0;
58     g.arc[i] = c[k++];    //输入点
59     for(;c[k]!='\0';++k)
60     {
61         j = c[k]-'0';
62         g.edge[i][j] = 1;    //输入边
63     }
64     ++i;
65 }
66 output(&g);    //输出边并验证
67
68 return 0;
69 }
70

```

### 邻接矩阵初始化

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  typedef struct _iobuf FILE;
5  #define maxsize 100
6
7  typedef struct    //定义图结构
8  {
9      int n,e;
10     char arc[maxsize];
11     int edge[maxsize][maxsize];
12 }graph;
13 char* fun(char *str)    //去除字符串中的空格和括号
14 {
15     int i=0,j=0;
16     char *res = (char*)malloc(sizeof(char)*strlen(str));
17     for(;str[i]!='\0';++i)
18     {
19         if(str[i]==' '||str[i]=='('||str[i]==')'||str[i]=='\n')
20             continue;
21         res[j++] = str[i];
22     }
23     return res;
24 }

```

```

24 void output(gragh *g)           //输出图
25 {
26     printf("This is your map:\npoints:\t%d\tedges:\t%d",g->n,g-
27 >e);
28     for(int i=0;i<g->n;++i)
29     {
30         printf("\n%c:\t",g->arc[i]);
31         for(int j=0;j<g->n;++j)
32         {
33             if(g->edge[i][j]!=0)
34             {
35                 printf("%c--%c(power=%d)\t",g->arc[i],g-
36 >arc[j],g->edge[i][j]);
37             }
38         }
39     }
40 }
41 int main()
42 {
43     gragh g;
44     int i,j;
45     char str[100],ch;           //用来向txt中读取一行数据
46     FILE *fp = fopen("../power_graph.txt","r");
47     if(fp==0)
48     {
49         printf("Open file error.\n");
50         return 0;
51     }
52     fscanf(fp,"%d %d ",&g.n,&g.e);           //读取点和边
53
54     i = 0;
55     while(!feof(fp))           //没有到达文件末尾的情况
56     {
57         fgets(str,100,fp);           //读取一行数据
58         char *c = fun(str);
59         int k = 0;
60         g.arc[i] = c[k++];           //输入点
61         for(;c[k]!='\0';++k)
62         {
63             j = c[k++]-'0';
64             g.edge[i][j] = c[k]-'0';           //输入边权值
65         }
66         ++i;
67     }
68     output(&g);           //输出边并验证
69
70     return 0;
71 }

```

## 邻接表初始化图

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  #define maxsize 100
5  typedef struct _iobuf FILE;
6
7  typedef struct node          //结点
8  {
9      int adjvex;
10     struct node *next;
11 }node;
12 typedef struct              //头结点
13 {
14     char vex;
15     node* next;
16 }vexNode;
17 typedef struct              //图
18 {
19     int n,e;
20     vexNode adjlist[maxsize];
21 }graph;
22
23 char * fun(char *str)        //去除字符串中的空格
24 {
25     int i=0;
26     char *ch = (char*)malloc(sizeof(char)*strlen(str));
27     for(int j=0;str[j]!='\0';++j)
28     {
29         if(str[j]==' '||str[j]=='\n')    continue;
30         ch[i++] = str[j];
31     }
32     ch[i] = '\0';
33     return ch;
34 }
35 void output(graph *g)        //输出图
36 {
37     int i,j;
38     printf("This is your map:\npoints:%d\tarcs:%d",g->n,g->e);
39     for(i=0;i<g->n;++i){
40         printf("\n%c:\t",g->adjlist[i].vex);
41         node *temp = g->adjlist[i].next;
42         while (temp)
43         {
44             printf("%c--%c\t",g->adjlist[i].vex,g->adjlist[temp-
45 >adjvex].vex);
46             temp = temp->next;
47         }
```

```

47     }
48 }
49 int main()
50 {
51     FILE *fp = fopen("../graph_test.txt","r");
52     if(fp==0)
53     {
54         printf("fail to open file.\n");
55         return 0;
56     }
57     graph g;
58     char s[100];
59     int i=0,j;
60     node *temp;
61     fscanf(fp,"%d %d ",&g.n,&g.e);
62     for(j=0;j<g.n;++j) //初始化
63         g.adjlist[j].next = 0;
64     while(!feof(fp)) //一行一行的开始赋值进去
65     {
66         fgets(s,100,fp);
67         char *str = fun(s);
68         int k=0; //k用来读取字符串
69         g.adjlist[i].vex = str[k++];
70         for(;str[k]!='\0';++k)
71         {
72             temp = (node*)malloc(sizeof(node)); //新建一个结点
73             temp->adjvex = str[k]-'0';
74             temp->next = 0;
75             temp->next = g.adjlist[i].next; //头插法
76             g.adjlist[i].next = temp;
77         }
78         ++i;
79     }
80     output(&g);
81
82     return 0;
83 }
84
85

```

### 邻接表初始化有向权值图

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #define maxsize 100
5 typedef struct _iobuf FILE;
6
7 typedef struct node //结点

```

```

8 {
9     int adjvex;
10    int power;                //网结构得加上权值
11    struct node *next;
12 }node;
13 typedef struct    //头结点
14 {
15     char vex;
16     node* next;
17 }vexNode;
18 typedef struct    //图
19 {
20     int n,e;
21     vexNode adjlist[maxsize];
22 }graph;
23
24 char * fun(char *str)        //去除字符串中的空格和括号
25 {
26     int i=0;
27     char *ch = (char*)malloc(sizeof(char)*strlen(str));
28     for(int j=0;str[j]!='\0';++j)
29     {
30         if(str[j]==' '||str[j]=='('||str[j]==')'||str[j]=='\n')
31             continue;
32         ch[i++] = str[j];
33     }
34     ch[i] = '\0';
35     return ch;
36 }
37 void output(graph *g)        //输出图
38 {
39     int i,j;
40     printf("This is your map:\npoints:%d\tarcs:%d",g->n,g->e);
41     for(i=0;i<g->n;++i)
42     {
43         printf("\n%c:\t",g->adjlist[i].vex);
44         node *temp = g->adjlist[i].next;
45         while (temp)
46         {
47             printf("%c--%c(power=%d)\t",g->adjlist[i].vex,g->adjlist[temp->adjvex].vex,temp->power);
48             temp = temp->next;
49         }
50     }
51 }
52 int main()
53 {
54     FILE *fp = fopen("../directed_power_graph.txt","r");
55     if(fp==0)

```

```

55     {
56         printf("fail to open file.\n");
57         return 0;
58     }
59     graph g;
60     char s[100];
61     int i=0,j;
62     node *temp;
63     fscanf(fp,"%d %d",&g.n,&g.e);
64     for(j=0;j<g.n;++j) //初始化
65         g.adjlist[j].next = 0;
66     while(!feof(fp)) //一行一行的开始赋值进去
67     {
68         fgets(s,100,fp);
69         char *str = fun(s);
70         int k=0; //k用来读取字符串
71         g.adjlist[i].vex = str[k++];
72         for(;str[k]!='\0';++k)
73         {
74             temp = (node*)malloc(sizeof(node)); //新建一个结点
75             temp->adjvex = str[k++]-'0';
76             temp->power = str[k]-'0';
77             temp->next = 0;
78             temp->next = g.adjlist[i].next; //头插法
79             g.adjlist[i].next = temp;
80         }
81         ++i;
82     }
83     output(&g);
84
85     return 0;
86 }
87
88

```

## 矩阵转为邻接表

```

1  #define maxsize 100;
2
3  //邻接表存储结构
4  typedef struct node //结点
5  {
6      int adjvex;
7      int val; //权值
8      struct node *next;
9  }node;
10 typedef struct //头结点
11 {
12     char vex;

```



```

13     int val;           //权值
14     node* next;
15 }vexNode;
16 typedef struct        //图
17 {
18     int n,e;
19     vexNode adjlist[maxsize];
20 }graph1;
21
22 //矩阵存储机构
23 typedef struct        //定义图结构
24 {
25     int n,e;           //n是点, e是边
26     char arc[maxsize];
27     int edge[maxsize][maxsize];
28 }gragh2;
29
30 // 矩阵转邻接表
31 graph1* function(graph2* g) {
32     if(!g){
33         return 0;
34     }
35     graph1* res = (graph1*)malloc(sizeof(graph1));
36     res->n = g->n;
37     res->e = g->e;
38     // 转换
39     for(int i=0;i<g->n;++i){
40         res->adjlist[i].vex = g->arc[i];           //转顶点
41         res->adjlist[i].next = 0;
42         for(int j=0;j<g->n;++j){                   //转边
43             if(g->edge[i][j]){                       //边存在的情况
44                 node* temp = (node*)malloc(sizeof(node)); //新建
一个结点
45                 temp->adjvex = g->edge[i][j];
46                 temp->next = 0;
47                 temp->next = g->adjlist[i].next;       //头插法
48                 g->adjlist[i].next = temp;
49             }
50         }
51     }
52     return res;
53 }
54 // 邻接表转矩阵
55 graph2* function2(graph1* g) {
56     if(!g){
57         return 0;
58     }
59     graph2* res = (graph2*)malloc(sizeof(graph2));
60     res->n = g->n;

```

```
61     res->e = g->e;
62     //初始化
63     for(int i=0;i<g->n;++i){
64         for(int j=0;j<g->n;++j){
65             res->edge[i][j] = 0;
66         }
67     }
68     //转换
69     for(int i=0;i<g->n;++i){
70         res->arc[i] = g->adjlist[i].vex;
71         node* temp = g->adjlist[i].next;
72         while(temp) {
73             res->edge[i][temp->adjvex] = temp->val;
74             temp = temp->next;
75         }
76     }
77     return res;
78 }
```