题目要求：<25-P1101-单词方阵-题目要求.docx>

## 题目分析：

这道题是一个DFS的题目，从八向搜索。可以设置一个八向的常量数组，搜索每一个方向，如果满足条件就递归，否则结束递归。当搜索到第7个单词‘g’时，用vis保存路径，此次DFS也就结束了。

首先在矩阵中搜索，找到y和i相连的方向k，以k方向进行DFS。

## 二、我的代码：

附上文件路径：<25-P1101-单词方阵.cpp>

1. #include<stdio.h>
2. #include<string.h>
3. int n,check[101][101]={0};
4. *//n记录方阵边长，check记录出现的正确单词的位置*
5. int dirx[]={0,1,1, 1, 0,-1,-1,-1};
6. *//搜索时x轴上移动的方向*
7. int diry[]={1,1,0,-1,-1,-1, 0, 1};
8. *//搜索时y轴上移动的方向*
9. char now[10],right[]="yizhong",squre[101][101];
10. *//now记录现在已得到的单词*
11. *//right记录正确的待匹配的单词*
12. *//squre记录输入的单词方阵*
13. int judge(int x,int y);
14. *//judge函数用来判断是否越界*
15. int judge2(char c);
16. *//优化1：judge2函数用来判断当前位置的字母是否在待匹配单词中，减少循环次数*
17. void display();
18. *//display函数用来输出最终得到的新的单词方阵*
19. void dfs(void);
20. *//dfs函数用来搜索单词并修改check数组*
21. int main(void)
22. {
23. scanf("%d",&n);
24. for(int i=0;i<n;i++) scanf("%s",squre[i]);
25. *//读入数据*
26. dfs();
27. *//调用函数搜索*
28. display();
29. *//调用函数输出*
30. }
31. int judge(int x,int y)
32. {
33. if(x>=0&&x<=n&&y>=0&&y<=n) return 1;
34. else return 0;
35. }
36. int judge2(char c)
37. {
38. for(int i=0;right[i];i++)
39. if(c==right[i]) return 1;
40. return 0;
41. }
42. void dfs()
43. {
44. int i,j;
45. for(i=0;i<n;i++)
46. {
47. for(j=0;j<n;j++)
48. {
49. if(squre[i][j]=='y')
50. {*//如果待匹配单词的首字母'y'出现则开始搜索单词*
51. for(int k=0;k<8;k++)
52. {*//循环变量k用来控制搜索的方向*
53. int I=i,J=j;
54. *//记录下首字母出现的位置，防止改变变量i和j*
55. for(int p=0;p<7&&(judge(I+dirx[k],J+diry[k])||judge(I,J))&&judge2(squre[I][J]);p++)
56. {*//在一个方向上一直移动7次得到一个长度与待匹配单词相等的单词*
57. *//控制条件有移动次数，边界溢出判断，单词与字母含于关系判断来减少语句执行次数*
58. now[p]=squre[I][J];*//读入当前经过的字母*
59. I+=dirx[k];
60. J+=diry[k];*//下标继续向该方向移动*
61. }
62. I-=dirx[k];
63. J-=diry[k];*//下标回溯一个单位，来正确地标记已经查找到的单词*
64. if (strncmp(now,"yizhong",7)==0)
65. {*//如果单词匹配则回溯标记check数组*
66. for(int p=0;squre[I][J]!='y';p++)
67. {*//控制何时标记结束*
68. check[I][J]=1;*//标记*
69. I-=dirx[k];
70. J-=diry[k];*//回溯*
71. }
72. check[I][J]=1;*//标记首字母*
73. }
74. for(int i=0;i<10;i++) now[i]='a';
75. *//     printf("\n");*
76. *//     display();*
77. }
78. }
79. }
80. }
81. }
82. void display()
83. {
84. *// printf("----------OUTPUT----------\n");*
85. for(int i=0;i<n;i++)
86. {
87. for(int j=0;j<n;j++)
88. {
89. if(check[i][j])printf("%c",squre[i][j]);
90. else printf("\*");
91. }
92. printf("\n");
93. }
94. }

输入样例：<P1101_2.in> 输出样例：<P1101_2.out>

输入样例：<P1101_4.in> 输出样例：<P1101_4.out>

## 三、优化

1.在dfs函数中我们可以利用常规的递归算法来实现使代码更为简洁。

2.我们可以通过判断每一步对应的子母是不是相同位次下待匹配单词的字母，如果不是即可退出循环，而无需像之前的代码中判断字母是否属于单词最后再比较整个的单词与路径

1. #include <bits/stdc++.h>
2. using namespace std;
3. const int maxn=100+10;
4. struct node
5. {
6. int x,y;
7. }c[maxn];*//记录路径*
8. char fz[maxn][maxn],stand[]="yizhong";*//fz保存单词矩阵，stand保存保准的“yizhong”便于匹配*
9. int vis[maxn][maxn];*//保存路径，是否该点为答案*
10. int dir[][2]={{-1,-1},{-1,0},{-1,1},{0,-1},{0,1},{1,-1},{1,0},{1,1}};*//八向的常量数组*
11. void dfs(int x,int y,node c[],int k,int cur)
12. {
13. if(cur==7){
14. for(int i=0;i<7;i++)
15. vis[c[i].x][c[i].y]=1;
16. }
17. else{
18. int dx=x+dir[k][0];*//沿着正确的k方向搜索*
19. int dy=y+dir[k][1];
20. if(cur==6||fz[dx][dy]==stand[cur+1]){
21. c[cur].x=x;c[cur].y=y;
22. dfs(dx,dy,c,k,cur+1);
23. }
24. }
25. }
26. int main()
27. {
28. freopen("input.txt","r",stdin);
29. int n;
30. scanf("%d",&n);
31. for(int i=0;i<n;i++) scanf("%s",fz[i]);
32. memset(vis,0,sizeof(vis));
33. for(int i=0;i<n;i++)*//搜索y，i相连的可能的方向k，以k为方向进行DFS*
34. for(int j=0;j<n;j++)
35. if(fz[i][j]=='y') for(int k=0;k<8;k++){
36. int x=i+dir[k][0];
37. int y=j+dir[k][1];
38. if(fz[x][y]=='i')
39. dfs(i,j,c,k,0);
40. }
41. for(int i=0;i<n;i++){*//输出结果*
42. for(int j=0;j<n;j++)
43. if(vis[i][j]) printf("%c",fz[i][j]);
44. else printf("\*");
45. printf("\n");
46. }
47. return 0;
48. }