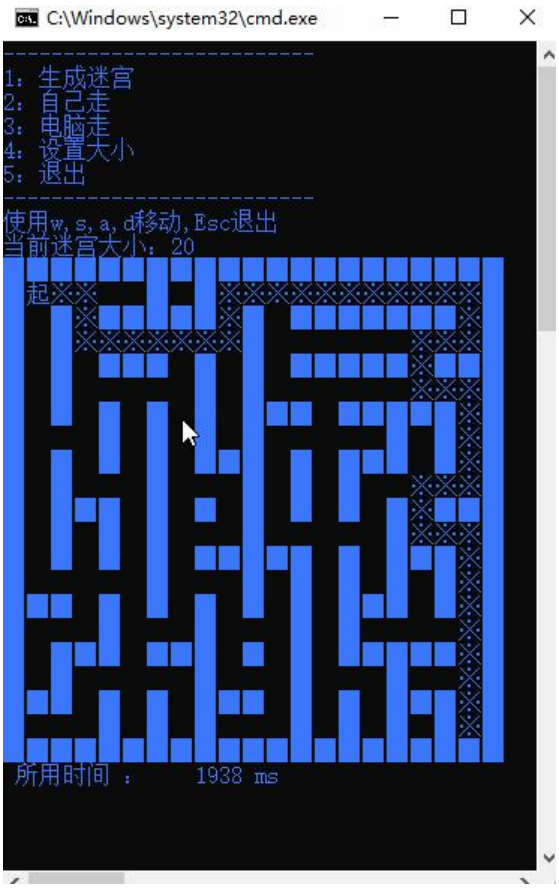
链滴

迷宫生成算法 prime，实现自动走迷宫

作者：jackzhong
原文链接：https://ld246.com/article/1585631733196
来源网站：链滴

从学习迷宫生成算法开始的。程序运行起来的样子：

GitHub地址： GitHub




```
#include <iostream>
```

原文链接：迷宫生成算法 prime，实现自动走迷宫

```
#include <ctime>
#include <cstdlib>
#include <cstdio>
#include<bits/stdc++.h>
#include <windows.h>
#include <tchar.h>
#include <SDKDDKVer.h>
#include<conio.h>
using namespace std;
#define MAZE_MAX 1000
int mz[MAZE_MAX+2][MAZE_MAX+2];
int hs[MAZE_MAX+2][MAZE_MAX+2];
int Move[MAZE_MAX+2][MAZE_MAX+2];
int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
int vis[MAZE_MAX];
int fa[MAZE_MAX];
int X = 12, Y = X;
void make_path();
struct node{
    int x,y;
};
struct node f[MAZE_MAX+2],ppk,endd;
int cnt=0;
void gotoxy(int x, int y){
    HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD pos;
    pos.X = x;
    pos.Y = y;
    SetConsoleCursorPosition(handle, pos);
}
void hideCursor(){
    CONSOLE_CURSOR_INFO cursor_info = { 1, 0 };
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cursor_info);
}
void Print_Maze()
{
    printf("使用w,s,a,d移动,Esc退出\n");
    printf("当前迷宫大小：%d\n",X );
    for(int i=0; i<=X; i++){
        for(int j=0;j<=X;j++){
            if(mz[i][j]){
                printf("%s","■");
             }
             else  if(Move[i][j]==1){
                printf("%s","※");
              }
             else  if(i==1&&j==1){
                printf("%s","起");
              }
            else if(i==X-1&&j==Y-1){
                printf("%s","终");
             }
             else if(i==ppk.x&&j==ppk.y){
                printf("人");
```

```
            }
             else{
                 printf("%s","  ");
             }
        }
        printf("\n");
    }
}
void updata_Maze(){
    hideCursor();
    gotoxy(0, 7);
    Print_Maze();
}
void Init_Maze(){
    cnt=0;
    for(int i=0;i<=X;i++){
        for(int j=0;j<=Y;j++){
            Move[i][j]=0;
            if(i==0||j==0||j==Y||i==X)
                hs[i][j]=-2;//墙壁为1
            else
                hs[i][j]=-1;//能打通的路为-1
            if(i==0||i==X||j==0||j==Y||(i%2==0)||(j%2==0)){
                 mz[i][j]=1;//
            }
            else{
                node tp;
                tp.x=i,tp.y=j;
                hs[i][j]=cnt;
                fa[cnt]=cnt;
                f[cnt++]=tp;
            }
        }
    }
    endd=f[cnt-1];
}
int find(int x){
    while(fa[x]!=x) x=fa[x];
    return x;
}
int merge(int x,int y){
    int f1=find(x);
    int f2=find(y);
    if(f1<f2) fa[f2]=f1;
    else if(f1>f2) fa[f1]=f2;
    else return 0;
    return 1;
}
int check_wall(int x,int y){
    if(x<=0||x>=X||y>=Y||y<=0){
        return 1;
    }
    return 0;
}
```

```
int check_choke(int x,int y){//检查是否全是阻塞的
    if(mz[x+1][y]==1 && mz[x-1][y]==1 &&mz[x][y+1]==1 &&mz[x][y-1]==1){
        return 1;
    }
    return 0;
}
void make_path(){
    srand(time(NULL));
    for(int p=0;p<cnt*8;p++){
        if(find(0)==find(cnt-1)){
            return ;
        }
        int road=vis[p%cnt];
        int next=rand()%4;
        int x=f[road].x,y=f[road].y;
        if(check_choke(x,y)){
            for(int i=0;i<4;i++){
                int cx=x+dir[next][0]*2,cy=y+dir[next][1]*2;
                int next_road=hs[cx][cy];
                if(!check_wall(cx,cy)){
                merge(road,next_road);
                mz[x+dir[next][0]][y+dir[next][1]]=0;
                }
            }
        }
        else{
            int cx=x+dir[next][0]*2,cy=y+dir[next][1]*2;
            int next_road=hs[cx][cy];
            if(!check_wall(cx,cy)){
                merge(road,next_road);
                mz[x+dir[next][0]][y+dir[next][1]]=0;
            }
        }
    }
    ppk=f[0];
}
void Init_list(){
    srand(time(NULL));
    for(int i=0;i<cnt;i++){
        vis[i]=i;
    }
    for(int i=0;i<cnt;i++){
        int a=rand()%cnt,b=rand()%cnt;
        swap(vis[a],vis[b]);
    }
}
int check_rode(int x,int y){
    return mz[x][y]==0 ? 1 : 0;
}
void p_Move(){
    int ch;ppk=f[0];
    while (1){
        if(ppk.x==f[cnt-1].x&&ppk.y==f[cnt-1].y){
            printf("你赢了！ \n");
```

```
            break;
        }
        updata_Maze();
        if(_kbhit()){//如果有按键按下，则_kbhit()函数返回真
            ch = _getch();//使用_getch()函数获取按下的键值
            if (ch == 27){
                break;
            }//当按下ESC时循环，ESC键的键值时27.
            else if(ch==119){// w
                if(check_rode(ppk.x-1,ppk.y))
                ppk.x--;
            }
            else if(ch==115){//s
                if(check_rode(ppk.x+1,ppk.y))
                ppk.x++;
            }
            else if(ch==97){//a
                if(check_rode(ppk.x,ppk.y-1))
                ppk.y--;
            }
            else if(ch==100){//d
                if(check_rode(ppk.x,ppk.y+1))
                ppk.y++;
            }
        }
    }
}
node step[MAZE_MAX];
int flag=0,walked[MAZE_MAX][MAZE_MAX];
void dfs_find_path(int stp,node now){
    if(flag) return ;
    if(now.x==endd.x&&now.y==endd.y){
        flag=1;
        printf("%s\n","走到了" );
        for(int i=0;i<stp;i++){
            Move[step[i].x][step[i].y]=1;
            updata_Maze();
        }
        return ;
    }
    node next_road;
    for(int i=0;i<4;i++){
        int cx=now.x+dir[i][0],cy=now.y+dir[i][1];
        if(check_rode(cx,cy)&&!walked[cx][cy]){
            walked[cx][cy]=1;
            next_road.x=step[stp].x=cx;
            next_road.y=step[stp].y=cy;
            dfs_find_path(stp+1,next_road);
            walked[cx][cy]=0;
        }
    }
}
void Print_hello(){
    hideCursor();
```

```
    gotoxy(0, 0);
    printf("-------------------------\n");
    printf("1：生成迷宫\n");
    printf("2：自己走\n");
    printf("3：电脑走\n");
    printf("4：设置大小\n");
    printf("5：退出\n");
    printf("-------------------------\n");
}
int begintime,endtime;
void menu1(){
    system("cls");
    begintime=clock();  //计时开始
    Init_Maze();//初始化迷宫
    Init_list();//生成一个个随机数列
    make_path();//生成迷宫
    updata_Maze();
    endtime = clock();  //计时结束
    printf(" 所用时间： %8d ms", endtime-begintime);
}
void menu2(){
    begintime=clock();  //计时开始
    p_Move();//
    endtime = clock();  //计时结束
    printf(" 所用时间： %8d ms", endtime-begintime);
}

void menu3(){
    begintime=clock();  //计时开始
    ppk=f[0];
    flag=0;
    walked[ppk.x][ppk.y]=1;
    dfs_find_path(1,ppk);
    endtime = clock();  //计时结束
    memset(Move,0,sizeof(Move));
    memset(walked,0,sizeof(walked));
    printf(" 所用时间： %8d ms", endtime-begintime);
}
void menu4(){
    printf("\n请输入迷大小：      ");
    scanf("%d",&X);X%2==1 ? X++ : 1;
    Y=X;
    menu1();
}
void MENU(){
    int ch;
    while (1){
        Print_hello();
        if(_kbhit()){//如果有按键按下，则_kbhit()函数返回真
            ch = _getch();//使用_getch()函数获取按下的键值
            if(ch==49){// w
                menu1();
            }
            else if(ch==50){
```

```
                menu2();
            }
            else if(ch==51){
                menu3();
            }
            else if(ch==52){//d
                menu4();
            }
            else if(ch==53){
                break;
            }
        }
    }
}
int main(){
    system("chcp 65001");
    system("mode con cols=220 lines=100");
    system("color 9");
    MENU();
    return 0;
}
```