代码逻辑:

定义新数据类型 Location，包含以下数据:

坐标:（double）x，y，用于记录当前点坐标

移动向量:vector，包含（x，y）两个 double 值，用于记录这个点是如何从上一个点移动到当前位置的

值:（double）value，用于记录当前点的函数值

主函数: Init()函数, 生成十个随机的点, 记录其中最优点 best_location, 并将最后一个点记录进整个链表中。作为起始点。

进入循环:最多 1000 次或与目标函数的最大值 1（通过偏导数计算可得）相差小于 0.0001 或者连续 100 次循环没有改变当前最大值节点，三者满足其一即可退出循环。

循环内过程:创建一个新节点（第一次循环时，由 Init（）函数生成的十个点的最后一个作为前节点, 此后即可以用之前生成的节点来作为起点，移动向量用 $\alpha D_k+\beta D_{best}+(1-\alpha-\beta)D_r$ 来确定。）然后比较此点的函数值与当前最大值的大小，若此点函数值更大，则更新最大值点并将连续不变的次数（unchanged）置零，若此点不如当前最大点更大，则连续不变的次数加一。然后将此节点加入到链表末尾并输出当前最大值。

函数逻辑:

Init()函数:创建一个空链表，十次循环，每次循环中，用随机数生成一个点，然后和当前的最大值点（best_location 是全局变量）比较，

判断是否更新最大值点，将最后一个生成的节点添加到链表中。返回此链表。

创建新点（get_new_location）：

生成一个方向向量：根据公式

$$\alpha D_k+\beta D_{best}+(1-\alpha-\beta)D_r$$

将当前已有的点 cur 的信息，cur.v 用于 Dk，best_location 用于 Dbest，再生成一个（0，1）的随机向量 Dr，将生成的向量进行归一化并乘以 step 步长，之后再用 move 函数生成一个新的点，将当前点利用方向 D 进行移动，返回移动后的新点

move 函数：生成新点 new_loc，将原来点 cur 的坐标加上参数 x，y 进行移动，移动后判断是否超界，如果超界，则“反弹”。并计算 value。

第二部分：不同参数对代码结果的影响

α=0.3，β=0.3，step=0.1

最后最优点函数值 0.999595，循环次数 172 次，最优点坐标 (0.999554,0.546128)

α=0.5，β=0.3，step=0.1

最后最优点函数值 0.999736，循环次数 262 次，最优点坐标 (0.682581,0.999505)

α=0.2，β=0.6，step=0.1

最后最优点函数值 0.999941，循环次数 124 次，最优点坐标

(0.484847,0.999923)

α=0.4，β=0.4，step=0.05

最后最优点函数值 0.999914，循环次数 248 次，最优点坐标 (0.998218,0.975733)

以上几次实验表明：在这个函数中，Dbest 的权重较大时，最优点函数值较高，循环次数较少，步长较短时，寻找到的 best_location，比较接近理论计算的（1，1）

优化建议：

可以考虑整个函数的偏导数（如果函数可微），并把偏导数为正的方向作为一个权重方向来确定移动方向