数组的每个索引做为一个阶梯，第 `i`个阶梯对应着一个非负数的体力花费值 `cost[i]`(索引从0开始)。

每当你爬上一个阶梯你都要花费对应的体力花费值，然后你可以选择继续爬一个阶梯或者爬两个阶梯。

您需要找到达到楼层顶部的最低花费。在开始时，你可以选择从索引为 0 或 1 的元素作为初始阶梯。

示例 1:

输入： `cost = [10, 15, 20]`
输出： `15`
解释： 最低花费是从`cost[1]`开始，然后走两步即可到阶梯顶，一共花费`15`。

示例 2:

输入： `cost = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]`
输出： `6`
解释： 最低花费方式是从`cost[0]`开始，逐个经过那些`1`，跳过`cost[3]`，一共花费`6`。

注意：

1. `cost` 的长度将会在 `[2, 1000]`。
2. 每一个 `cost[i]` 将会是一个Integer类型，范围为 `[0, 999]`。

思路，动态规划解题，第n个阶梯，自己阶梯耗力最小为自己阶梯所需要的耗力+前两个阶梯中最小的耗力。需要注意运算优先级。

```
class Solution {
public:
    int minCostClimbingStairs(vector<int>& cost) {
        int n=cost.size();
        for(int i=2;i<n;++i){
            cost[i]=cost[i]+(cost[i-1]<cost[i-2]?cost[i-1]:cost[i-2]);
        }
        return cost[n-1]<cost[n-2]?cost[n-1]:cost[n-2];
    }
};
```