数据结构和算法

作者：小甲鱼

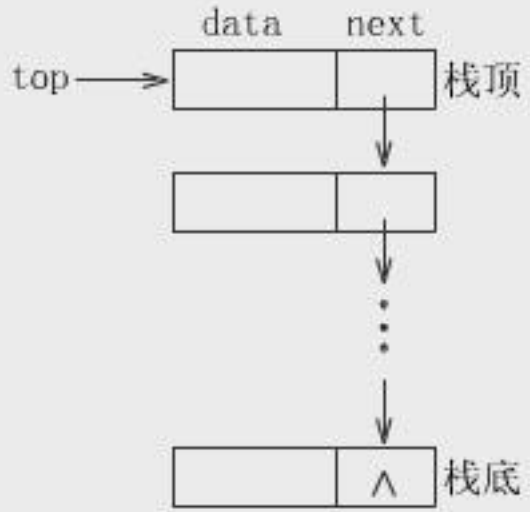
让编程改变世界

Change the world by program

栈的链式存储结构

- 讲完了栈的顺序存储结构，也给大家结合了一些例题演练，相信大家对栈再也不陌生了吧？
- 现在我们来看下栈的链式存储结构，简称栈链。（通常我们用的都是栈的顺序存储结构存储，链式存储我们作为一个知识点，大家知道就好！）
- 栈因为只是栈顶来做插入和删除操作，所以比较好的方法就是将栈顶放在单链表的头部，栈顶指针和单链表的头指针合二为一。
- No pic you say a J8……

栈的链式存储结构



链栈示意图

栈的链式存储结构

```
teypedef struct StackNode
{
    ElemType data;     // 存放栈的数据
    struct StackNode *next;
} StackNode, *LinkStackPtr;
teypedef struct LinkStack
{
    LinkStackPrt top;   // top指针
    int count;          // 栈元素计数器
}
```

进栈操作

- 对于栈链的Push操作，假设元素值为e的新结点是s，top为栈顶指针，我们得到如下代码：

```
Status Push(LinkStack *s, ElemType e)
{
    LinkStackPtr p = (LinkStackPtr) malloc (sizeof(StackNode));
    p->data = e;
    p->next = s->top;
    s->top = p;
    s->count++;
    return OK;
}
```

出栈操作

- 至于链栈的出战Pop操作，假设变量p用来存储要删除的栈顶结点，将栈顶指针下移一位，最后释放p即可。

```
Status Pop(LinkStack *s, ElemType *e)
{
    LinkStackPtr p;
    if( StackEmpty(*s) ) // 判断是否为空栈
        return ERROR;
    *e =s->top->data;
    p =s->top;
    s->top =s->top->next;
    free(p);
    s->count--;
    return OK;
}
```

终极实践

- 在讲解这道例题的时候，请允许小甲鱼花一点点的时间对小学时候的数学老师进行感谢，嗯，谢谢您，让我学会如何计算以下这道表达式，并且认为它十分简单：(1-2)*(4+5)
- 人类早就熟悉这种中缀表达式的计算方式，随便拉一个小朋友过来，给他一颗糖，他会马上告诉你这个结果应该是等于-9，因为括号里边的要先进行计算。
- 但是计算机不喜欢了，因为我们有小括号中括号大括号，还允许一个嵌套一个，这样子计算机就要进行很多次if判断才行决定哪里先计算。

逆波兰表达式

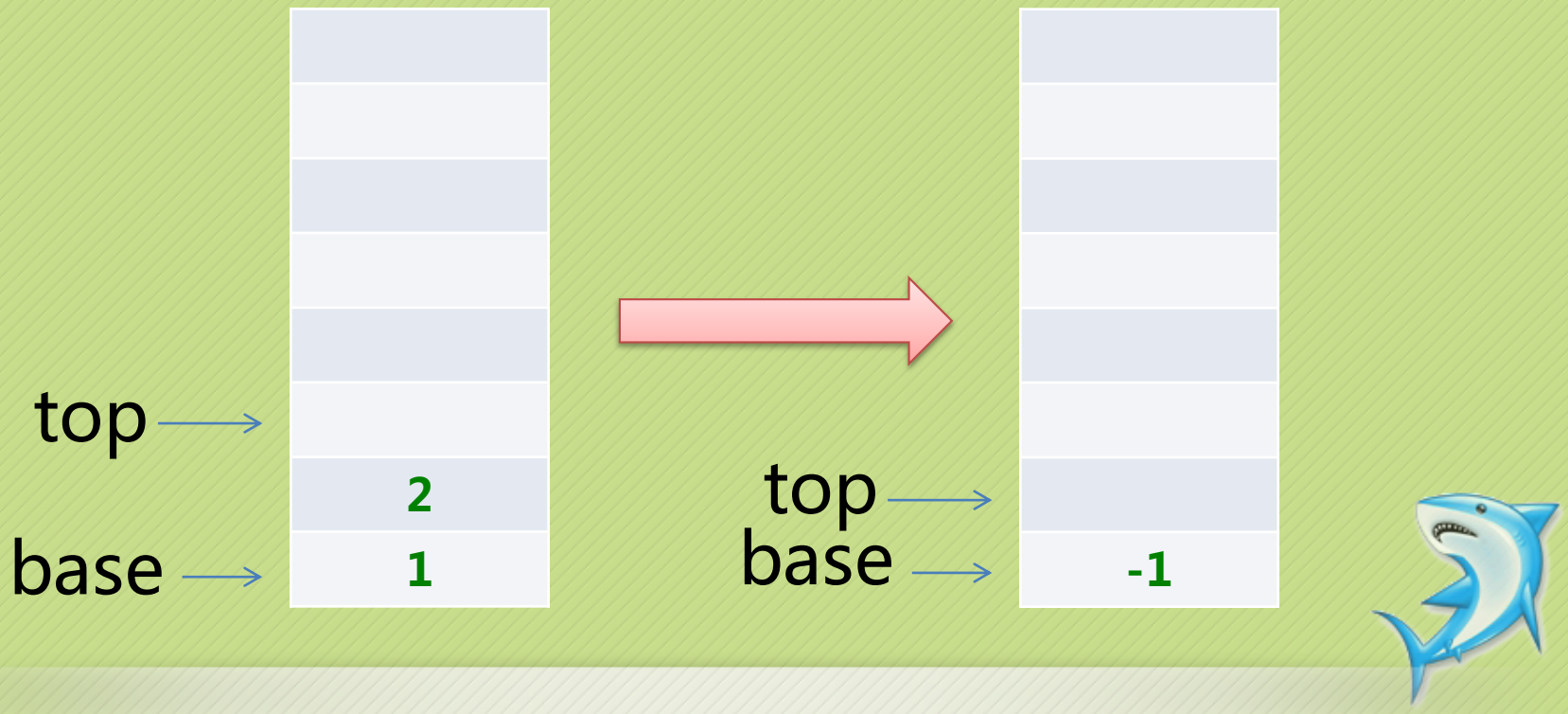
- 后来，在20世纪三十年代，波兰逻辑学家Jan.Lukasiewicz不知道是像牛顿一样被苹果砸到脑袋而想到万有引力原理，或者还是像阿基米德泡在浴缸里突发奇想给皇冠是否纯金做验证，总之他也是灵感闪现了，然后发明了一种不需要括号的后缀表达式，我们通常把它称为逆波兰表达式(RPN)。
- 很多鱼油好奇为什么他发明的东西是以他的国籍而不是以他的名字命名的呢？这也告诉我们，想要流芳百世，名字还得起得朗朗上口才行。

逆波兰表达式

- 我们先来看看，对于(1-2)*(4+5)，如果用逆波兰表示法，应该是这样：12-45+*
- 这种方式敢情我们人类是不大好接受的了，不过对于计算机来说，那可是喜爱至极。
- 因为只需要利用栈的特点，就可以将这种后缀表达式的性能发挥到极致。
- 解析来就让小甲鱼图文并茂的解释一下吧！
- No pic you say a J8……

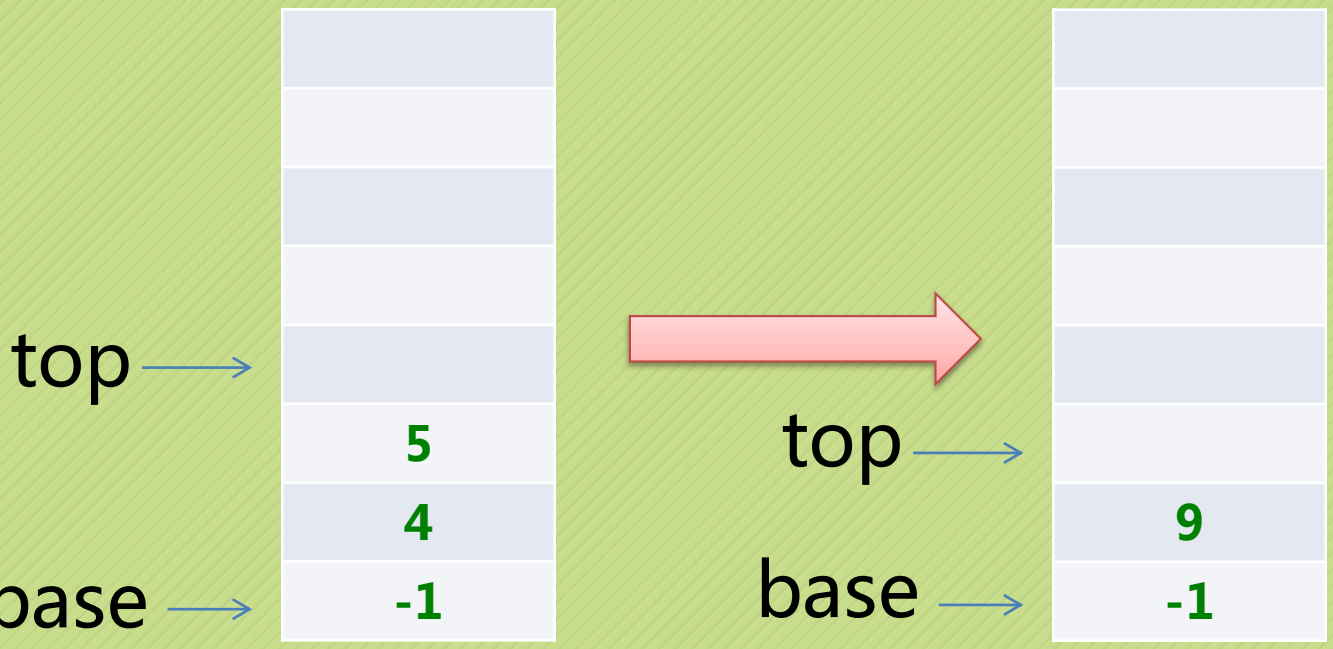
逆波兰表达式

- 数字1和2进栈，遇到减号运算符则弹出两个元素进行运算并把结果入栈。



逆波兰表达式

- 4和5入栈，遇到加号运算符，4和5弹出栈，相加后将结果9入栈。



逆波兰表达式

- 然后又遇到乘法运算符，将9和-1弹出栈进行乘法计算，此时栈空并无数据压栈，-9为最终运算结果！