C 实例教学(类的应用-01) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/133/2021_2022_C___E5_AE_9E_E4_BE_8B_E6_c97_133765.htm 类类的概念抓住了程序的本质。程序的基本元素是数据。而函数是围绕数据进行处理和操作。抓住了数据这个“ 纲 ”，程序中关系复杂的各种函数就变得脉络清楚，可以随着相应的数据组合成类，类的使用使得：* 程序设计本身更有条理了；* 程序的可读性更好了；* 程序设计的过程真正象是机器部件的组装；* 由于程序的零部件化，使得程序的可重用性变成切实可行的事。 为了学会OOP方法，首先让我们看看C 程序中类及其对象是怎样工作的。 9.1 设计一个栈类 栈(stack)是程序设计过程中经常遇到朝气一种数据结构形式，它对于数据的存放和操作有下面这样的特点： 1) 它只有一个对数据进行存入和取出的端口；2) 后进者先出，即最后被存入的数据将首先被取出。其形式很象一种存储硬币的小容器，每次只可以从顶端压入一个硬币，而取出也只可以从顶端进行，即后进先出。 这样的数据存储和管理形式在一些程序设计中很有用。例如，编译系统中(这是一类比较复杂的程序)，对于函数调用的处理、对于表达式计算的处理，都利用了栈这样的数据结构。 下面是一个关于栈的程序： // program 6_1.h #include const int maxsize=6. // enum boola{false,true}. /*注：如果在TC中调试，应加上这一句*/ class stack{ float data[maxsize]. int top. public: stack(void). ~stack(void). bool empty(void). void push(float a). float pop(void). }. stack::stack(void) { top=0. cout} stack::~stack(void) { cout} bool stack::empty(void) { return top==0?true:false. } void

stack::push(float a) { if(top==maxsize) { coutreturn. } data[top]=a. top . } float stack::pop(void) { if(top==0) { coutreturn 0. } top--. return data[top]. } void main() { stack s1,s2. for(int i=1.is1.push(2*i). coutfor(i=1.icoutfor(i=1.is1.push(2.5*i). for(i=1.is2.push(s1.pop()). do coutwhile(!(s2.empty())). } 程序运行结果如下： 