用于计算四则混合运算表达式的递归函数 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/134/2021_2022__E7_94_A8_E4_BA_8E_E8_AE_A1_E7_c97_134112.htm AnsiString __fastcall Calc(String sExp) { // 计算不带变量的四则混合运算表达式(只含数字、小数点、 -*/号和括号) // 正数不许带正号 int posL, pos, posR. // pos->当前考虑的运算符的位置 // posL->当前考虑的运算符之前最近的运算符的位置 // posL->当前考虑的运算符之前后近的运算符的位置 String sTmp, sL, sR. // sL->当前考虑的运算符的左操作数字符串，sR->当前考虑的运算符的右操作数字符串 bool IsMinus. // IsMinus->当前*/序列的符号 if(sExp.AnsiPos("error")) return(sExp). while(pos = sExp.AnsiPos(" ")) sExp = sExp.Delete(pos, 1). // 去除表达式中的空格 if(sExp.IsEmpty()) return("0"). while((pos = sExp.AnsiPos("[")) > 0 || (pos = sExp.AnsiPos("{")) > 0) // 统一左括号为(sExp = sExp.SubString(1, pos - 1) "(" sExp.SubString(pos 1, sExp.Length()). while((pos = sExp.AnsiPos("]")) > 0 || (pos = sExp.AnsiPos("}")) > 0) // 统一右括号为) sExp = sExp.SubString(1, pos - 1) ")" sExp.SubString(pos 1, sExp.Length()). // 处理括号：递归计算括号中的表达式，最后消去括号 while(posL=sExp.LastDelimiter("(")) // 最里层({ sTmp = sExp.SubString(posL 1, sExp.Length()). posR = sTmp.AnsiPos(")"). // 最里层) if(posR == 0) return("error：没有配对的), 公式错！"). sExp = sExp.SubString(1, posL - 1) Calc(sTmp.SubString(1, posR - 1)) sTmp.SubString(posR 1, sTmp.Length()). } // 以下处理不带括号表达式中的*/序列 IsMinus = false. // IsMinus->当前*/

序列的符号 while(sExp.LastDelimiter("*/")) // 存在*或/ { for(pos = 1. !sExp.IsDelimiter("*/", pos) amp. pos if(pos == 1 || pos == sExp.Length()) return("error：首或尾字符是*/运算符, 公式错！"). posL = sExp.SubString(1, pos - 1).LastDelimiter(" -"). // posL->第一个*/之前的第一个 - Minus0: for(posR = pos 1. !sExp.IsDelimiter(" -*/", posR) amp. posR // posR->第一个*/之后的第一个 -*/运算符 if(posR == sExp.Length()) return("error：尾字符是 -*/运算符, 公式错！"). if(sExp.SubString(pos, 2) == "*-" || sExp.SubString(pos, 2) == "/-") // 乘数或除数为负 { sExp.Delete(pos 1, 1). IsMinus = !IsMinus. goto Minus0. } sL = sExp.SubString(posL 1, pos - posL - 1). sR = sExp.SubString(pos 1, posR - pos - 1). if(sExp.IsDelimiter("/", pos) amp. sR == "0") return("error：除数为零，无意义！"). sExp = (posL == 0? String(""): sExp.SubString(1, posL)) (sExp.IsDelimiter("*", pos)? (sL.ToDouble() * sR.ToDouble()): (sL.ToDouble() / sR.ToDouble())) sExp.SubString(posR, sExp.Length()). } if(IsMinus) sExp = String("-") sExp. // 经过上面的系列处理，sExp中的运算符号只剩下 和-了 // 以下处理不带括号表达式中的 -序列 IsMinus = false. // 加数或减数的符号 while(sExp.LastDelimiter(" -")) // 存在 或- { for(pos=2. !sExp.IsDelimiter(" -", pos) amp. pos