TI-Nspire™ 技术程序编辑器简介

欲详细了解 TI 技术，可访问 education.ti.com/eguide 以查看在线帮助。

重要信息

除程序附带的许可证另有明确规定外，德州仪器不作任何明示或暗示的保证，包括但不限于对特定目的的适销性和适用性的隐含保证，涉及任何程序或书籍材料，材料只能按“原样”提供。在任何情况下，德州仪器不对任何人因购买或使用这些材料而造成的特殊，抵押，偶然或后果的损害负责，也不包括德州仪器的唯一和唯一的责任。行动不得超过计划许可证规定的金额。此外，德州仪器不对任何其他方使用这些材料的任何索赔提出任何责任。

商标和版权

TI-Nspire™软件使用Lua作为脚本环境。有关版权和许可证信息，请访问 http://www.lua.org/license.html。

TI-Nspire™软件使用Chipmunk Physics 5.3.4版作为仿真环境。有关许可证信息，请参见http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/。

Microsoft®和Windows®是Microsoft Corporation在美国和/或其他国家/地区的注册商标。

MacOS®，iPad®和OSX®是Apple Inc.的注册商标。

Unicode®是Unicode，Inc.在美国和其他国家/地区的注册商标。

内容

程序编辑器快速入门 1

定义程序或函数 1
查看程序或函数 4
打开一个函数或程序进行编辑 5
从库导入程序 5
创建函数或程序的副本 6
重命名程序或函数 6
更改库访问级别 6
查找文本 7
查找并替换文本 7
关闭当前函数或程序 8
运行程序和计算函数 8
将值代入程序 10
显示信息 12
使用局部变量 13
函数和程序之间的区别 14
从另一程序调用一个程序 15
控制函数或程序流 16
使用 If、Lbl 和 Goto 控制程序流 16
使用循环重复一组命令 18
更改模式设置 21
调试程序和处理错误 22

一般信息 23

在线帮助 23
联络 TI 支持部门 23
维修和保修信息 23

程序编辑器快速入门

您可通过在“计算器”输入行键入定义语句或使用程序编辑器来创建用户定义的函数和程序。程序编辑器有一些优点，在本部分中均有涉及。有关更多信息，请参阅*计算器*。

- 编辑器有一些编程模板和对话框，可以帮助您使用正确的语法定义函数和程序。
- 编辑器可让您输入多行编程语句，无需特殊的键顺序来添加每行。
- 您可以轻松创建私有和公共对象(变量、函数和程序)。更多信息请参阅*库*。

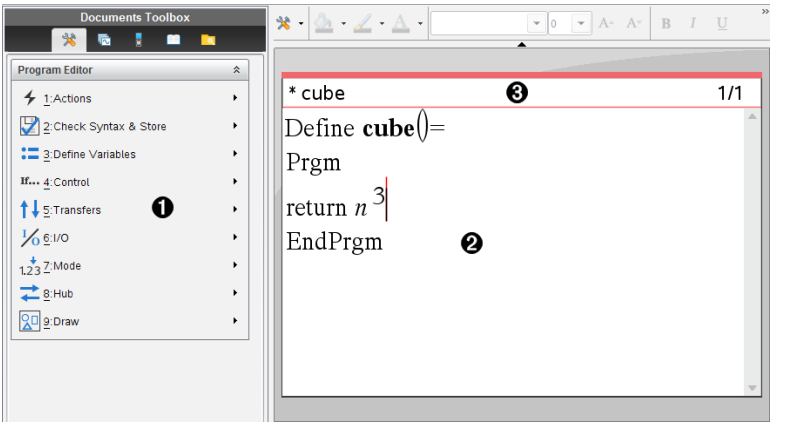
启动程序编辑器

▸ 如要在当前问题中添加新的“程序编辑器”页面：

从工具栏中单击**插入>程序编辑器 >新建**。

手持设备：按 doc，然后选择**插入>程序 编辑器>新建**。

注：通过“计算器”页面的**函数&程序**菜单也能访问编辑器。



❶ “程序编辑器”菜单 – 使用“正常”视图模式下，在“程序编辑器”工作区域中随时可以使用该菜单。

❷ “程序编辑器”工作区

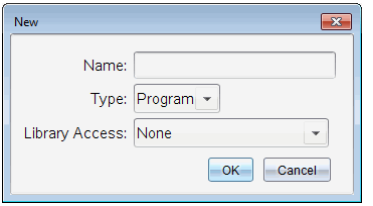
❸ 状态行显示的是正在编辑的函数或程序的行号信息和名称。星号 (*) 表示该函数“不洁”，意思是自上一次检查该函数的句法并将其保存后，该函数发生了改变。

定义程序或函数

启动新程序编辑器

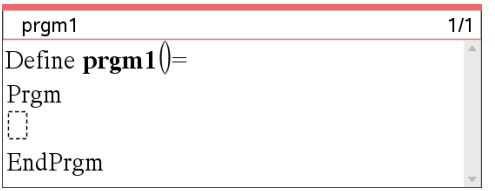
1. 确保您处于要在其中创建程序或函数的文档和问题中。

2. 单击应用程序工具栏上的**插入**按钮，然后选择**程序编辑器 > 新建**。(在手持设备上，按 doc，然后选择**插入 > 程序编辑器 > 新建**。)



3. 键入要定义的函数或程序的名称。
4. 选择**类型(程序**或**函数)**。
5. 设置**库访问**:
   - 要仅从当前文档和问题使用函数或程序，请选择**无**。
   - 要使函数或程序可从任何文档访问但不在目录中显示，请选择 **LibPriv**。
   - 要使函数或程序可从任何文档访问并在目录中显示，请选择 **LibPub(在目录中显示)**。更多信息请参阅*库*。
6. 单击**确定**。

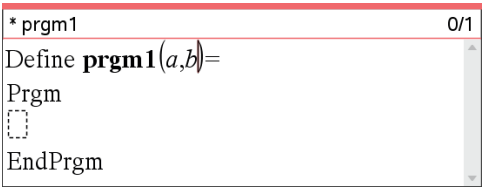
   将打开程序编辑器的新实例，其中包含与您所做选择匹配的模板。



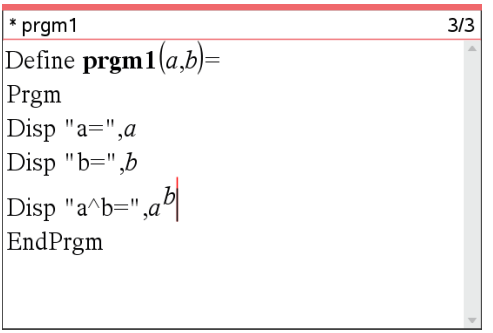
在函数或程序中输入行

在您键入时，程序编辑器不会执行命令或计算表达式。仅在您计算函数或运行程序时才会执行它们。

1. 如果您的函数或程序要求用户提供自变量，请在名称后面的括号中键入参数名称。用逗号分隔参数。



2. 在 Func 和 EndFunc(或 Prgm 和 EndPrgm) 行之间，键入构成函数或程序的语句行。



- 您可以键入函数和命令的名称，也可以从目录中插入它们。
- 行可以长于屏幕的宽度;在这种情况下，您可能必须执行滚动才能查看整个语句。
- 键入每一行后，按 **Enter**。将插入新的空白行，并允许您继续输入另一行。
- 使用 ◄、►、▲ 和 ▼ 箭头键滚动函数或程序以输入或编辑命令。

插入注释

注释信息对于查看或编辑程序的人员很有用。它们不会在程序运行时显示，对程序流没有影响。在带注释的行的开头会显示 © 符号。

```
Define LibPub volcyl(ht,r) =
Prgm
©volcyl(ht,r) => volume of cylinder ❶
  Disp "Volume =", approx(π • r^2 • ht)
©这是另一个注释。
EndPrgm
```

❶ 显示所需句法的注释。因为此库对象是公共的，并且此注释是 Func 或 Prgm 块中的第一行，所以此注释作为帮助信息显示在目录中。更多信息请参阅*库*。

要插入注释:

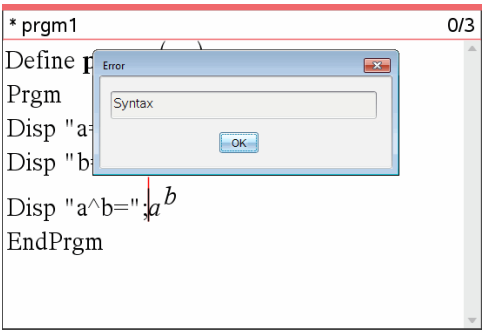
1. 将光标置于要插入注释的行的末尾。
2. 在**操作**菜单中，单击**插入命令**，或按 **Ctrl+T**。
3. 在 © 符号后面键入注释文本。

检查句法

程序编辑器允许您检查函数或程序的句法是否正确。

▸ 从**检查句法并保存**菜单中，单击**检查句法**。

如果句法检查程序发现任何句法错误，它将显示错误消息，并尝试将光标置于第一个错误附近，以便您可以进行更正。



保存函数或程序

您必须保存函数或程序，才能使其可访问。在保存之前，程序编辑器会自动检查句法。

程序编辑器的左上角显示星号 (*)，表示尚未保存函数或程序。

- 从**检查句法并保存**菜单中，单击**检查句法并保存**。

  如果句法检查程序发现任何句法错误，它将显示错误消息，并尝试将光标置于第一个错误附近。

  如果未找到句法错误，则程序编辑器顶部的状态行中将显示“已保存成功”消息。

注:如果函数或程序被定义为库对象，则还必须将文档保存在指定的库文件夹中，并刷新库以使对象可供其他文档访问。更多信息请参阅*库*。

查看程序或函数

1. 从 **动作** 菜单，选择 **视图**。

2. 如果函数或程序为库对象，请从 **位置** 列表选择它的库。
3. 从 **名称** 列表选择函数或程序名称。

   函数或程序将会显示在查看器中。

4. 使用箭头键查看函数或程序。
5. 如果您要编辑程序，单击 **Edit**。

   手持设备：按 tab 突出显示 **Edit**，然后按 enter。

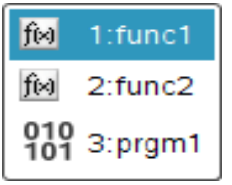
   注意：Edit 选项仅可用于在当前问题中定义的函数和程序。要编辑库对象，您必须首先打开它的库文档。

打开一个函数或程序进行编辑

您只能从当前问题打开函数或程序。

注意：您不能修改锁定的程序或函数。要解锁对象，请转至 计算器 页面并使用 **unLock** 命令。

1. 显示可用函数和程序的列表。
   - 从 **动作** 菜单，选择 **打开**。



2. 选择要打开的条目。

从库导入程序

您可以在当前问题内，将定义为库对象的函数或程序导入 程序编辑器。导入的副本不会被锁定，即使原始副本已锁定。

1. 从 **动作** 菜单，选择 **导入**。

2. 选择 **库名称**。
3. 选择对象的 **名称**。
4. 如果您要将导出的对象命名为别的名称，请在 **导入为** 下键入名称。

创建函数或程序的副本

创建新函数或程序时，您可能发现从现有副本着手会容易一些。您创建的副本不会被锁定，即使原始副本已锁定。

1. 从 **动作** 菜单，选择 **创建副本**。
2. 键入新名称，或单击 **确定** 接受建议的名称。
3. 如果您要更改访问级别，请选择 **库访问**，然后选择新级别。

重命名程序或函数

您可以重命名和(可选) 更难改当前函数或程序的访问级别。

1. 从 **动作** 菜单，选择 **重新命名**。

2. 键入新名称，或单击 **确定** 接受建议的名称。
3. 如果您要更改访问级别，请选择 **库访问**，然后选择新级别。

更改库访问级别

1. 从 **动作** 菜单，选择 **更改库访问**。

2. 选择 **库访问**：
   - 要仅从当前 计算器 问题使用函数或程序，请选择 **无**。
   - 要使函数或程序可以从任何文档访问，但不显示在 目录 中，请选择 **LibPriv**。
   - 要使函数或程序可以从任何文档访问，同时也显示在 目录 中，请选择 **LibPub**。

查找文本

1. 从 **动作** 菜单，选择 **查找**。

2. 键入您要查找的文本，然后单击 **确定**。
   - 如果找到了文本，它将在程序中突出显示。
   - 如果没有找到文本，将会显示通知消息。

查找并替换文本

1. 从 **动作** 菜单，选择 **查找与替换**。

2. 键入您要查找的文本。
3. 键入替换文本。
4. 单击 **替换** 替换光标位置后的第一处文本，或单击 **全部替换** 替换所有出现的文本。

注意：如果在数学模板中找到文本，将会显示一条消息，警告您替换文本将替换整个模板，而不只是找到的文本。

关闭当前函数或程序

▸ 从 **动作** 菜单，选择 **关闭**。

如果函数或程序有尚未存储的更改，在关闭前会提示您检查语法并存储。

运行程序和计算函数

定义和存储程序或函数之后，可以从应用程序进行使用。所有应用程序都可计算函数值，但只有“计算器”和“记事本”应用程序能够运行程序。

程序语句按连续顺序执行（不过某些命令会改变程序流）。输出（如果有）会显示在应用程序的工作区域中。

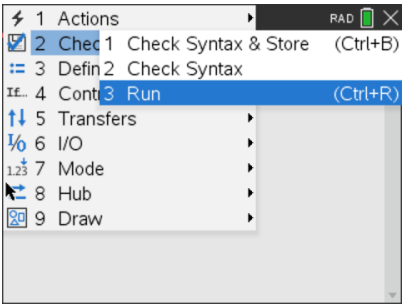
- 程序执行会持续进行，直到它到达最后一条语句或 **Stop** 命令。
- 函数执行会持续进行，直到它到达 **Return（返回）** 命令。

用程序编辑器运行程序或函数

1. 确保程序或函数已经定义好，并且程序编辑器是当前可用窗格（计算机）或页面（手持设备）。

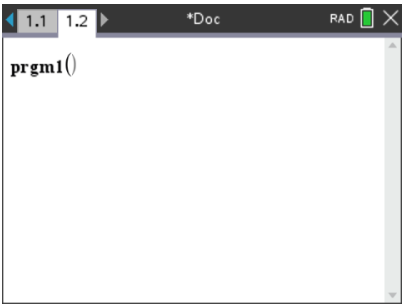
2. 在工具栏上，单击 **Document Tools（文档工具）** 按钮  ，然后选择 **Check Syntax & Store（检查语法并存储）> Run（运行）**。
   —或—
   按 **Ctrl+R**。

   手持设备：按 menu 2 3，或按 ctrl R。



   这会自动：

   - 检查语法并存储程序或函数，
   - 在紧接着程序编辑器后的计算器应用程序的第一个可用行上粘贴程序或函数名称。如果该位置上不存在计算器，则会插入新计算器。



3. 如果程序或函数需要提供一个或多个参数，请在圆括号内键入值或变量名称。
4. 按 enter。

注意：还可以通过键入带有圆括号的程序名称以及任何所需参数并按 enter，在“计算器”或“记事本”应用程序中运行程序或函数。

使用短名称和长名称

在同一问题里，凡定义过的对象，都可以通过输入其短名称(在对象的 **Define** 命令中提供的名称) 来访问它。对于所有已定义对象(包括私有、公共和非库对象) 都是这种情况。

可以通过键入对象的长名称，从任何文档访问库对象。长名称包含对象库文档的名称，后跟反斜杠“\”，再后跟对象的名称。例如，在库文档 **lib1** 中定义为 **func1** 的对象的长名称是 **lib1\func1**。要在手持设备上键入“\”字符，请按 shift ÷。

注意：如果无法记住私有库对象的确切名称或所需参数顺序，则可以打开库文档或使用程序编辑器查看对象。还可以使用 **getVarInfo** 查看库中的对象列表。

使用公共库程序或函数

1. 确保在文档的第一个问题中定义对象，存储对象，将库文档保存在 MyLib 文件夹中并刷新库。
2. 打开 TI-Nspire™ 应用程序，使用其程序或函数。

   注意：所有应用程序都可计算函数值，但只有“计算器”和“记事本”应用程序能够运行程序。
3. 打开“Catalog(目录) ”，使用库选项卡查找并插入对象。
   —或—
   键入对象的名称。对于程序或函数，始终在名称后加上圆括号。

   ```
   libs2\func1()
   ```
4. 如果某程序或函数需要提供一个或多个参数，请在圆括号内键入值或变量名称。

```
libs2\func1(34,power)
```

5. 按 enter。

使用私有库程序或函数

要使用私有库对象，必须知道其长名称。例如，在库文档 **lib1** 中定义为 **func1** 的对象的长名称是 **lib1\func1**。

注意：如果无法记住私有库对象的确切名称或所需参数顺序，则可以打开库文档或使用程序编辑器查看对象。

1. 确保在文档的第一个问题中定义对象，存储对象，将库文档保存在 MyLib 文件夹中并刷新库。
2. 打开要在其中使用程序或函数的 TI-Nspire™ 应用程序。

   注意：所有应用程序都可计算函数值，但只有"计算器"和"记事本"应用程序能够运行程序。
3. 键入对象的名称。对于程序或函数，始终在名称后加上圆括号。

```
libs2\func1()
```

4. 如果对象需要提供一个或多个参数，请在圆括号内键入值或变量名称。

```
libs2\func1(34,power)
```

5. 按 enter。

中断运行程序或函数

在程序或函数运行期间，会显示繁忙指针 ⧖。

- 要停止程序或函数，
  - Windows®：按住 **F12** 键并反复按 **Enter** 键。
  - Mac®：按住 **F5** 键并反复按 **Enter** 键。
  - 手持设备：按住 ⌂ on 键并反复按 enter 。

  会显示一个消息。要在程序编辑器中编辑程序或函数，请选择 **Go To(转到)**。光标会出现在发生中断的命令处。

将值代入程序

您可以从多种方法中选择一种，提供函数或程序在计算中使用的值。

将值嵌入程序或函数内

如果程序或函数每次都使用相同的值时，这种方法非常有用。

1. 定义程序。

```
Define calculatearea()=
```

```
Prgm
w:=3
h:=23.64
area:=w*h
Disp area
EndPrgm
```

2. 运行程序。

```
calculatearea()
                                    70.92
```

让用户赋值给变量

程序或函数可以引用预先创建的变量。此方法要求用户记住变量名称并在使用对象之前赋值给它们。

1. 定义程序。

```
Define calculatearea()=
Prgm
area:=w*h
Disp area
EndPrgm
```

2. 提供变量，然后运行函数。

```
w:=3 : h:=23.64
calculatearea()
                                    70.92
```

让用户提供值作为参数

此方法可让用户在调用程序或函数的表达式内传递一个或多个值作为参数。

以下程序 **volcyl** 将计算圆柱体积。它要求用户提供两个值：圆柱的高度和半径。

1. 定义 **volcyl** 程序。

```
Define volcyl(height,radius) =
Prgm
Disp "体积 =", approx(π • radius^2 • height)
EndPrgm
```

2. 运行该程序可显示高度为 34 毫米，半径为 5 毫米的圆柱体积。

```
volcyl(34,5)                体积 = 534.071
```

注意：您在运行 **volcyl** 程序时不必使用参数名称，但您必须提供两个自变量(如值、变量或表达式)。第一个必须代表高度，第二个必须代表半径。

向用户请求值(仅限程序)

您可以在程序中使用 **Request** 和 **RequestStr** 命令，暂停该程序并显示一个对话框，提示用户提供信息。此方法不要求用户记住变量名称或所需的变量顺序。

您不能在函数中使用 **Request** 或 **RequestStr** 命令。

1. 定义程序。

```
Define calculatearea()=
Prgm
  Request "宽度: ",w
  Request "高度: ",h
area:=w*h
EndPrgm
```

2. 运行程序并回答请求。

```
calculatearea() :
宽度:3        (输入 3 作为响应)
高度:23.64  (输入 23.64 作为响应)

                                    70.92
```

如果您希望程序将用户响应理解为字符串而不是数学表达式，请使用 **RequestStr**，而不要使用 **Request**。这不再要求用户将回答括在引号内 (“”)。

显示信息

运行函数或程序不会显示中间计算结果，除非您包含要显示它们的命令。这是在输入行上执行计算与函数或程序中执行计算之间的重大区别。

例如，以下计算不会在函数或程序中显示结果(但从输入行计算时会显示结果)。

```
  ⋮
x:=12•6
cos(π/4)➞
  ⋮
```

在历史记录中显示信息

您可以在程序或函数中使用 **Disp** 命令在历史记录中显示信息，包括中间结果。

```
  ⋮
Disp 12•6
Disp "导致:",cos(π/4)
  ⋮
```

在对话框中显示信息

您可以使用 **Text** 命令暂停程序运行，并在对话框中显示信息。用户选择 **确定** 可继续，也可选择 **取消** 停止程序。

您不能在函数中使用 **Text** 命令。

```
⋮
Text "区 =" & area
⋮
```

注意：使用 **Disp** 或 **Text** 显示结果将不会存储该结果。如果您希望稍后引用某个结果，请将其保存为全局变量。

```
⋮
cos(π/4)→maximum
Disp maximum
⋮
```

使用局部变量

局部变量是一个临时变量，仅在计算用户定义的函数或运行用户定义的程序时才存在。

局部变量的示例

以下程序段显示一个 **For...EndFor** 循环(将在本模块的后面部分讨论)。变量 *i* 是循环计数器。在大多数情况下，变量 *i* 仅在程序运行时使用。

```
Local i ❶
For i,0,5,1
  Disp i
EndFor
Disp i
```

❶ 声明变量 *i* 为局部变量。

注意：在可能情况下，请将仅在程序内使用并且在程序停止后不需要再用的变量，声明为局部变量。

什么会导致未定义变量错误消息？

当您计算的用户定义函数或运行的用户定义程序中引用了尚未初始化(赋值)的局部变量，则会显示一条 **变量未定义** 变量错误消息。

例如：

```
Define fact(n)=Func
  Local m ❶
  While n>1
    n•m→m: n–1→n
  EndWhile
```

```
  Return m
  EndFunc
```

❶ 局部变量 *m* 未被赋予初始值。

初始化局部变量

在引用所有局部变量之前，必须给它们赋初始值。

```
Define fact(n)=Func
  Local m: 1→m  ❶
  While n>1
    n•m→m: n–1→n
  EndWhile
  Return m
  EndFunc
```

❶ 1 保存作为 *m* 的初始值.

注意 (CAS)：函数和程序不能使用局部变量执行符号计算。

CAS：执行符号计算

如果您要函数或程序执行符号计算，您必须使用全局变量，而不是局部变量。不过，您必须要确定程序之外不存在该全局变量。以下方法可能对您有所帮助。

- 引用一个不可能在函数或程序之外存在的全局变量名称，一般是两或三个字符。
- 在程序内包含 **DelVar** 命令，从而在引用全局变量之前删除全局变量(如果存在) 。(**DelVar** 不会删除锁定或已链接的变量。)

函数和程序之间的区别

程序编辑器 中定义的函数与内置于 TI-Nspire™ CAS 软件中的函数类似。

- 函数必须返回可以绘制或输入到表格中的结果。程序不能返回结果。
- 您可以在表达式内使用函数(而不是程序) 。例如：**3 • func1(3)** 有效，但 **3 • prog1(3)** 无效。
- 只有“计算器”和“记事本”应用程序能够运行程序。但是，您可以在“计算器”、“记事本”、“列表 & 电子表格”、“图形 & 几何”以及“数据 & 统计”中计算函数值。
- 函数可以引用任何变量；不过，它只能将值存储为局部变量。程序可以将值存储为局部和全局变量。

  注意：用于传递值给函数的参数会被自动视为局部变量。如果您要存储为任何其他变量，您必须从函数内将它们声明为 **Local**。
- 函数不能将程序当作子程序调用，但可以调用另一个用户定义的函数。
- 您不能在函数内定义程序。

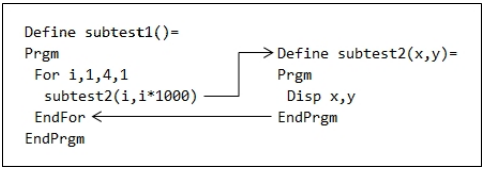
- 函数不能定义全局函数，但可以定义局部函数。

从另一程序调用一个程序

一个程序可以将另一程序作为子程序调用。子程序可以是外部程序(独立程序)，也可以是内部程序(包含在主程序内)。当程序需要在几个不同的地方重复同一组命令时，子程序非常有用。

调用独立的程序

要调用独立的程序，请使用您用于从输入行运行程序的相同语法。



定义和调用内部子程序

要定义内部子程序，请使用带 **Prgm...EndPrgm** 模板的 **Define** 命令。因为必须在调用之前先定义子程序，所以一种好的做法是在主程序开头定义子程序。

内部子程序的调用和执行方法与独立程序相同。

```
Define subtest1()=
  Prgm
  local subtest2 ❶
  Define subtest2(x,y)= ❷
  Prgm
    Disp x,y
  EndPrgm
  ©Beginning of main program
  For i,1,4,1
    subtest2(i,I*1000) ❸
  EndFor
  EndPrgm
```

❶ 声明子程序为局部变量。

❷ 定义子程序。

❸ 调用子程序。

注意：使用 程序编辑器 的 **Var** 菜单输入 **Define** 和 **Prgm...EndPrgm** 命令。

使用子程序的注意事项

执行到子程序结尾处时，将返回调用程序。要在其他时候退出子程序，请使用不含自变量的 **Return**。

子程序不能访问调用程序中声明的局部变量。同样，调用程序不能访问子程序中声明的局部变量。

Lbl 命令在所在程序内为局部变量。因此，调用程序中的 **Goto** 命令不能分支到子程序的标签，反之亦然。

避免循环定义错误

计算用户定义的函数或运行程序时，您可以指定其中包含用于定义函数或创建程序的相同变量的参数。不过，为了避免循环定义错误，您必须给用于计算函数或运行程序的变量赋值。例如：

```
x+1→x ❶
```

- 或 -

```
For i,i,10,1
  Disp i ❶
EndFor
```

❶ 如果 x 或 i 没有被赋值，将导致 **循环定义** 错误消息。如果 x 或 i 已赋值，该错误不会出现。

控制函数或程序流

您在运行程序或计算函数时，程序行会按顺序执行。不过，有些命令会改变程序流。例如：

- **If...EndIf** 命令等控制结构使用条件测试来决定要执行的程序部分。
- **For...EndFor** 等循环命令会重复一组命令。

使用 If、Lbl 和 Goto 控制程序流

If 命令和几个 **If...EndIf** 结构可让您根据条件来执行语句或语句块，也就是根据测试结果执行(例如 x>5) 。**Lbl**(标签) 和 **Goto** 命令可让您从函数或程序中的一个地方分支或跳转到另一个地方。

If 命令和一些 **If...EndIf** 结构存在于 程序编辑器 的 **控制** 菜单上。

当您插入 **If...Then...EndIf** 等结构时，将会在光标位置插入一个模板。光标定位可以方便您输入条件测试。

If 命令

要在条件测试为真时执行一条命令，请使用通用形式：

```
If x>5
  Disp "x是大于5" ❶
Disp x ❷
```

❶ 仅当 x>5 时执行，否则跳过。

❷ 始终显示 x 的值。

在此例中，您必须在执行 **If** 命令前，存储一个值到 x。

If...Then...EndIf 结构

要在条件测试为真时，执行一组命令，请使用结构：

```
If x>5 Then
  Disp "x是大于5" ❶
  2•x→x ❶
EndIf
Disp x ❷
```

❶ 仅当 x>5 时执行。

❷ 显示以下值：
如果 x>5，则显示 2x
如果 x≤55，则显示 x

注意：EndIf 标记在条件为真时执行的 **Then** 块结束。

If...Then...Else...EndIf 结构

要在条件为真时执行一组命令，在条件为假时执行另一组命令，请使用此结构：

```
If x>5 Then
  Disp "x是大于5" ❶
  2•x→x ❶
Else
  Disp "x是小于或等于5" ❷
  5•x→x ❷
EndIf
Disp x ❸
```

❶ 仅当 x>5 时执行。

❷ 仅当 x≤5 时执行。

❸ 显示以下值：
如果 x>5，则显示 2x
如果 x≤5，则显示 5x

If...Then...ElseIf...EndIf 结构

更复杂的 **If** 命令可让您测试多个条件。假定您要一个程序测试用户提供的表示四个选项之一的自变量。

要测试每个选项(If Choice=1，If Choice=2，等等)，请使用 **If...Then...ElseIf...EndIf** 结构。

Lbl 和 Goto 命令

您可以使用 **Lbl**(标签) 和 **Goto** 命令控制程序流。这些命令位于 程序编辑器 的 **转移** 菜单。

使用 **Lbl** 命令标记(指定名称给) 函数或程序中的特定位置。

Lbl *labelName*	要指定给此位置的名称(使用与变量名称相同的命名约定)

您然后可以在函数或程序中的任何位置，使用 **Goto** 命令分支到指定标签对应的位置。

Goto *labelName*	指定要分支到哪个 **Lbl** 命令

因为 **Goto** 是非条件命令(它总是分支到指定的标签) ，所以常与 **If** 命令一起使用，以便您指定条件测试。例如：

```
If x>5
  Goto GT5  ❶
Disp x
--------
--------  ❷
Lbl GT5
Disp "这个数字是大于5"
```

❶ 如果 x>5，直接分支到标签 GT5。

❷ 对于此例而言，程序必须包含命令(如 **Stop**) 来阻止在 x≤5 时执行 **Lbl** GT5。

使用循环重复一组命令

要连续重复同一组命令，请使用一种循环结构。有几类循环可以使用。每种类型都可让您根据条件测试，以不同方式退出循环。

循环以及与循环有关的命令位于 程序编辑器 的 **控制** 和 **转移** 菜单上。

当您插入其中一种循环结构时，其模板即会插入到光标位置。然后，您可以开始输入将在循环内执行的命令。

For...EndFor 循环

For...EndFor 循环使用计数器控制循环的重复次数。**For** 命令的语法为：

注意：如果增量为负，结束值可以小于起始值。

For *变量*, *下限*, *上限* [, *步长*]

❶ ❷ ❸ ❹

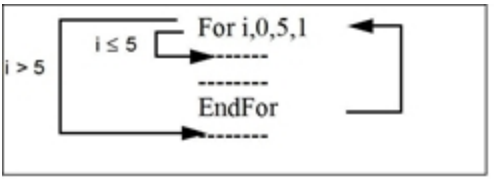
❶ 变量 作为计数器

❷ 第一次执行 **For** 时使用的计数器值

❸ 当 变量 超过此值时退出循环

❹ 以后每执行一次 **For**，计数器的增量(如果忽略可选值，则 *步长* 为 1。)

执行 **For** 时，会将 变量 值与 *上限* 值进行比较。如果 变量 没有超出 *上限*，即会执行循环；否则，控制语句将跳至 **EndFor** 后的命令。



注意：For 命令自动增加计数器变量的值，以便函数或程序在重复执行一定次数后退出循环。

在循环末尾 (**EndFor**)，控制语句跳回 **For** 命令，其中的变量增加并与 *上限* 进行比较。

例如：

```
For i,0,5,1
  Disp i ❶
EndFor
Disp i ❷
```

❶ 显示 0、1、2、3、4 和 5。

❷ 显示 6。当 *变量* 增加到 6 时，就不执行循环。

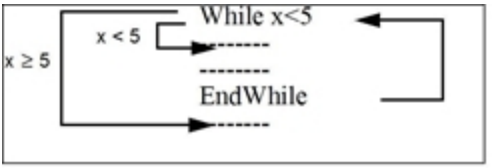
注意：您可以将计数器变量声明为局部变量，如果在函数或程序停止后不需要保存的话。

While...EndWhile 循环

只要指定的条件为真，**While...EndWhile** 循环就会重复命令块。**While** 命令的语法为：

While *条件*

在执行 **While** 时，就会计算 *条件*。如果 *条件* 为真，就会执行循环；否则，控制语句将跳至 **EndWhile** 后的命令。



注意：While 命令不会自动改变条件。您必须包含允许函数或程序退出循环的命令。

在循环末尾 **(EndWhile)**，控制语句将跳回 **While** 命令，在这里重新计算条件。

要开始执行循环，初始条件必须为真。

- 必须在 **While** 命令之前，设置条件中引用的任何变量。(您可以将值置于函数或程序中，也可以提示用户输入值。)
- 循环必须包含改变条件中值的命令，最终导致条件为假。否则，条件始终为真，函数或程序就不能退出循环(称为无限循环) 。

例如：

```
0→x ❶
While x<5
  Disp x ❷
  x+1→x ❸
EndWhile
Disp x ❹
```

❶ 初始设置 x。

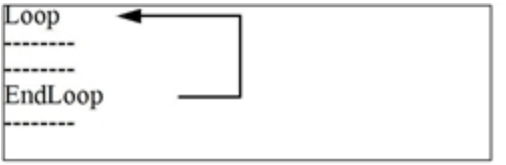
❷ 显示 0、1、2、3 和 4。

❸ 增加 x。

❹ 显示 5。当 x 增加到 5 时，就停止执行循环。

Loop...EndLoop loops

Loop...EndLoop 构成无限循环，这将无穷无尽地执行。**Loop** 命令不含任何自变量。



通常，您要在循环中插入可让程序退出循环的命令。常用命令有：**If**、**Exit**、**Goto** 和 **Lbl** (标签) 。例如：

```
0→x
Loop
  Disp x
  x+1→x
  If x>5  ❶
    Exit
EndLoop
Disp x        ❷
```

❶ **If** 命令会检查条件。

❷ 当 *x* 增加到 6 时，退出循环并跳转到此处。

注意：Exit 命令从当前循环退出。

在本例中，**If** 命令可出现在循环的任何位置。

当 If 命令为：	循环为：
循环的开头	仅在条件为真时执行。
循环的末尾	已执行至少一次，并且在条件为真时重复执行。

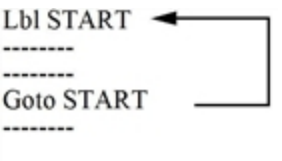
If 命令可能也使用 **Goto** 命令将程序控制语句转移至指定的 **Lbl(** 标签**)** 命令。

立即重复循环

Cycle 命令立即将程序控制语句转至循环的下一次迭代(当前迭代完成之前) 。此命令与 **For...EndFor**、**While...EndWhile** 和 **Loop...EndLoop** 一起使用。

Lbl 和 Goto 循环

虽然 **Lbl(** 标签**)** 和 **Goto** 命令不是严格的循环命令，但他们可以形成无限循环。例如：



至于 **Loop...EndLoop**，该循环应该包含可让函数或程序退出循环的命令。

更改模式设置

函数和程序可以使用 **setMode()** 函数，暂时设置特定计算或结果模式。程序编辑器 的 **模式** 菜单可让您轻松输入正确的语法，无需您记忆数值代码。

注意：函数或程序定义内进行的模式更改不会扩展到函数或程序外。

设置模式

1. 将光标定位至您要插入 **setMode** 函数的地方。

2. 从 **模式** 菜单，选择要更改的模式，然后选择新的设置。

   手持设备：按 菜单 7，选择要更改的模式，然后选择新的设置。

   正确的语法即会被插入光标位置。例如：

```
setMode(1,3)
```

调试程序和处理错误

您在编写函数或程序后，您可以使用几种方法查找并更正错误。您可以将错误处理命令内置于函数或程序本身。

如果函数或程序允许用户从多个选项选择，请确保运行它并测试每个选项。

调试方法

运行时错误消息可以查找语法错误，但不能查找程序逻辑错误。以下方法可能很有用。

- 暂时插入 **Disp** 命令以显示关键变量的值。
- 要确认循环已执行正确的次数，请使用 **Disp** 命令显示计数器变量或条件测试中的值。
- 要确认子程序已执行，请使用 **Disp** 命令在子程序开头或结尾处显示"进入子程序"和"退出子程序"等消息。
- 手动停止程序或功能：
  - Windows®:按住 **F12** 键，并反复按 **Enter** 键。
  - Macintosh®:按住 **F5** 键，并反复按 **Enter** 键。
  - 手持设备：按住 开机 键，并反复按 enter 键。

错误处理命令

命令	说明
Try...EndTry	定义一个块，可让函数或程序执行命令，并且在需要时，从该命令生成的错误恢复。
ClrErr	清除错误状态，并将系统变量 *errCode* 设为零。要查看 *errCode* 的使用范例，请参阅 *参考指南*中的 **Try** 命令。
PassErr	将错误传递至 **Try...EndTry** 块的下一级。

一般信息

在线帮助

education.ti.com/eguide

选择您的国家，获取更多产品信息。

联络 TI 支持部门

education.ti.com/ti-cares

选择您的国家，获取技术和其他支持资源。

维修和保修信息

education.ti.com/warranty

选择您所在的国家/地区，了解有关保修期限和条款或产品服务的信息。

保修期内不会影响您的法定权利。