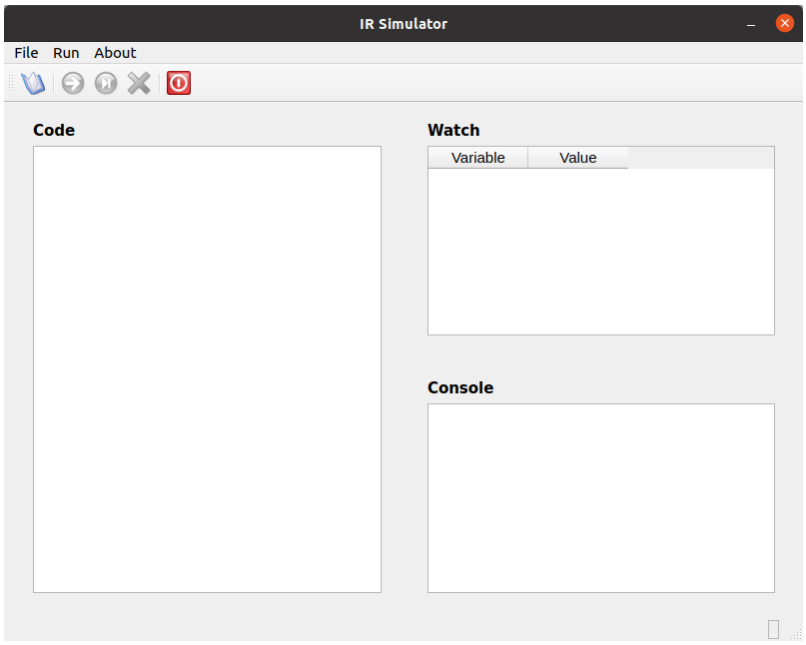

图22. 虚拟机小程序IR Simulator的界面。

8. 附录B：虚拟机小程序使用说明

我们提供的虚拟机小程序叫做IR Simulator，它本质上就是一个中间代码解释器。这个程序使用Python写成，图形界面部分则借助了跨平台的诺基亚Qt库。由于实验环境为Linux，因此该程序只在Linux下发布。注意，运行虚拟机小程序之前要确保本机上装有Qt运行环境。你可以在Linux终端下使用“sudo pip install sip PyQt5 PyQt5-tools”命令以获取Qt运行环境（也可参考附录C中的离线安装方式），然后通过我们课程网站下载虚拟机小程序（请参考附录C中的资源下载和安装介绍）。如果出现了qt.qpa.plugin:xcb插件问题，可以使用“sudo apt install libxcb-xinerama0”命令下载依赖库。

IR Simulator的运行界面如图22所示。整个界面分为三个部分：左侧的**代码区**、右上方的**监视区**和右下方的**控制台区**。代码区显示已经载入的中间代码，监视区显示中间代码中当前执行函数的参数与局部变量的值，控制台区供WRITE函数进行输出用。我们可以点击上方工具栏中的第一个按钮或通过菜单选择File → Open来打开一个保存有中间代码的文本文件（注意该文件的后缀名必须是ir）。如果中间代码中包含语法错误，则IR Simulator会弹出对话框提示出错位置并拒绝继续载入代码；如果中间代码中没有错误，那么你将看到类似于图23所示的内容。

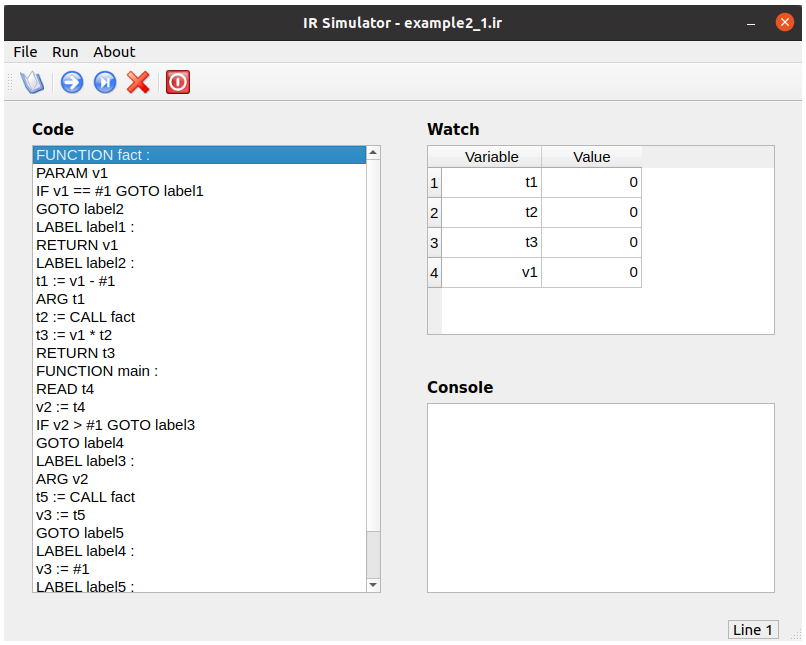

图23. IR Simulator载入中间代码成功的界面。

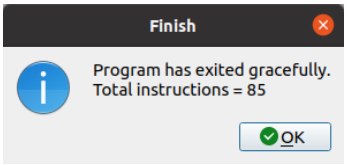

图24. IR Simulator运行中间代码成功的提示。

此时中间代码已被完全载入到左侧的代码区，因为中间代码尚未执行，所以右侧监视区的显示为空。此时你可以单击工具栏上的第二个按钮或通过菜单选择Run → Run（快捷键为F5）直接运行该中间代码，或者单击第三个按钮或通过菜单选择Run → Step（快捷键为F8）来对该中间代码进行单步执行。单击第四个按钮或通过菜单选择Run → Stop可以停止中间代码的运行并重新初始化。如果中间代码的运行出现错误，那么IR Simulator会弹出对话框以提示出错的行号以及原因（一般的出错都是因为访问了一个不存在的内存地址，或者在某个函数中缺少RETURN语句等）；如果一切正常，你将会看到如图24所示的对话框。

这提示我们中间代码的运行已经正常结束，并且本次运行共执行了85条指令。由于实验三会考察优化中间代码的效果，如果你想得到更高的评价，就需要设法使Total instructions后面的数字尽可能小。

另外，关于IR Simulator有几点需要注意：

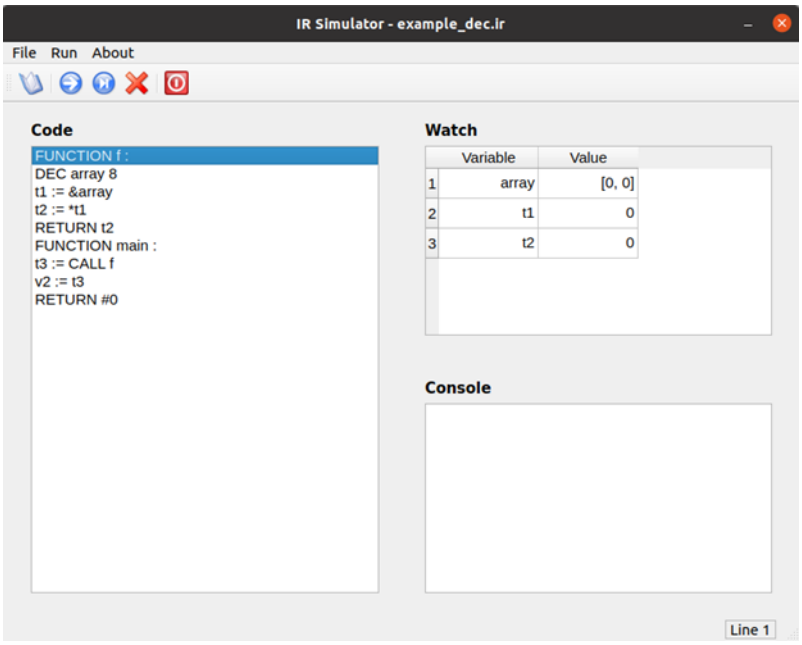

图25. DEC定义的变量与普通变量之间的显示区别。

1) 在运行之前请确保你的中间代码不会陷入死循环或无穷递归中。IR Simulator不会对这类情况进行判断，因此一旦当它执行了包含死循环或无穷递归的中间代码，你就需要将IR Simulator强制退出了。

2) 互相对应的ARG语句和PARAM语句数量一定要相等，否则可能出现无法预知的错误。

3) 由于IR Simulator实现上的原因，单步执行时它并不会在GOTO、IF、RETURN等与跳转相关的语句上停留，而是会将控制直接转移到跳转目标那里，这是正常的情况。

4) 在中间代码的执行过程中，右侧监视区始终显示当前正在执行函数的中间代码中的变量及其值。如图25中所示，当执行到函数“f”的第一条代码时，右侧监视区中会显示此函数对应的中间代码中使用的变量，且变量的初始值默认为0。

5) 使用DEC语句定义的变量在监视区中的显示与普通的变量的区别如图25所示。注意监视区中变量array的值：与其它变量不同，array中的内容被一对中括号“[”和“]”包裹了起来。有时候，因为DEC出来的变量内容较多，Value一栏可能无法显示完全，你可以用鼠标调整Value栏的宽度使其所有内容都能被显示出来。

6) 所有函数的参数与局部变量都只在运行到该函数之后才会去为其分配存储空间并在监视区显示。存储空间采用由低地址到高地址的栈式分配方式，递归调用的局部变量将不会影响到上层函数的相应变量的值。如果想要修改上层函数中变量的值，则需要向被调用的函数传递

该变量的地址作为参数，也就是我们常说的引用调用。不过，监视区中显示的变量总是当前这一层变量的值，只要不退回到上一层，我们就无法看到上层函数局部变量的值是多少。