Engintime

DS Lab 数据结构实验指导

北京英真时代科技有限公司

DS Lab数据结构实验指导

北京英真时代科技有限公司

地址：北京市房山区拱辰大街 98号 7 层 0825
邮编：102488
电话：010-60357081
邮箱：964515564@qq.com
网址：http://www.engintimes.com

目 录

DS Lab简介

获得人生中的成功需要的专注与坚持不懈多过天才与机会。
——C.W.Wendte

一、概述

DS Lab是一款专用于高等院校数据结构实验教学的集成环境，具有可视化程度高、操作简便、扩展性强等特点。DS Lab主要由两部分组成：

- 一个功能强大的 IDE环境。
- 一套精心设计的数据结构实验源代码。

DS Lab 所提供的 IDE 环境可以在 Windows 操作系统上快速安装、卸载，其用户界面和操作方式与 Microsoft Visual Studio 完全类似，有经验的读者可以迅速掌握其基本用法。该 IDE环境提供的强大功能可以帮助读者顺利的完成数据结构实验，主要功能包括对实验源代码的演示功能、编辑功能、编译功能、调试功能和验证功能。

DS Lab提供了一套精心设计的数据结构实验源代码，这些源代码使用 C 语言和 C++语言编写，可以与主流数据结构教材配套使用。涵盖了从线性表、字符串、二叉树、图到查找、排序等所有重要的数据结构和算法。这些源代码以模块化的方式进行组织，并配有完善的中文注释，可读性好，完全符合商业级的编码规范。

使用 DS Lab进行数据结构实验的过程可以参见图 0-1。

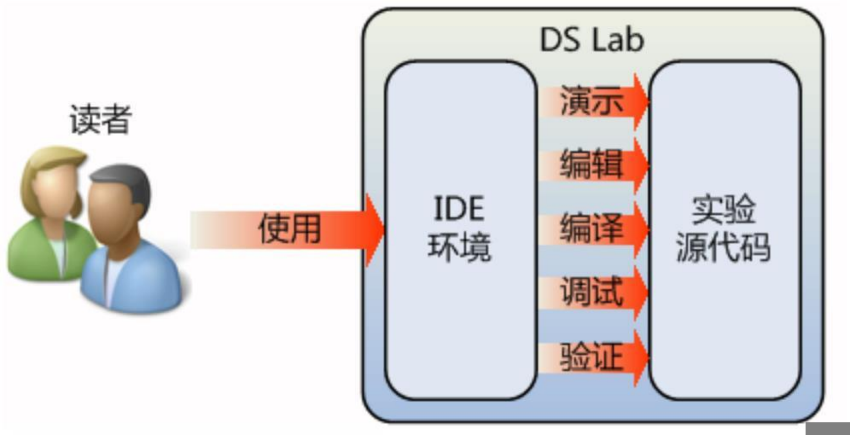


图 0-1：使用 DS Lab进行数据结构实验

二、数据结构（C 语言）实验题目清单

序号	概念	实验名称	性质	学时
1	线性表	实验环境的使用	验证	1
2		线性表的顺序表示及插入操作	验证	1
3		线性表的顺序表示及删除操作	验证	1
4		单链表的插入操作	验证+设计	1
5		单链表的删除操作	验证+设计	1
6		双向循环链表的插入操作	验证+设计	1
7		双向循环链表的删除操作	验证+设计	1
8	字符串	字符串的顺序表示及插入操作	验证+设计	1
9	树	二叉树的先序遍历操作	验证+设计	1
10		二叉树的中序遍历操作	验证+设计	1
11		二叉树的后序遍历操作	验证+设计	1
12		线索二叉树	验证+设计	1
13		赫夫曼树的构造	验证+设计	1
14		一般树的遍历	验证+设计	1
15	图	图的深度优先搜索	验证+设计	1
16		图的广度优先搜索	验证+设计	1
17		拓扑排序	验证+设计	1
18		关键路径	验证+设计	1
19		最短路径(迪杰斯特拉算法)	验证+设计	1
20		最短路径(弗洛伊德算法)	验证+设计	1
21	查找	折半查找	验证+设计	1
22		二叉排序树的构造	验证+设计	1
23		创建哈希表(线性探测再散列)	验证+设计	1
24		创建哈希表(二次探测再散列)	验证+设计	1
25		创建哈希表(伪随机探测再散列)	验证+设计	1
26	内部排序	冒泡排序	验证+设计	1
27		快速排序	验证+设计	1
28		堆排序	验证+设计	1

三、数据结构（C++语言）实验题目清单

序号	概念	实验名称	性质	学时
1	线性表	实验环境的使用	验证	1
2		简单线性链表的插入和删除操作	验证	1
3		双向循环链表的插入和删除操作	验证	1
4	字符串	实现字符串类的常用操作	验证+设计	1
5	树	先序线索二叉树及先序遍历	验证+设计	1
6		中序线索二叉树及中序遍历	验证+设计	1
7		后序线索二叉树及后序遍历	验证+设计	1
8		赫夫曼树	验证+设计	1
9	图	图的深度优先搜索	验证+设计	1
10		图的广度优先搜索	验证+设计	1
11		拓扑排序	验证+设计	1
12		关键路径	验证+设计	1
13		最短路径(迪杰斯特拉算法)	验证+设计	1
14		最短路径(弗洛伊德算法)	验证+设计	1
15	查找	折半查找	验证+设计	1
16		二叉排序树	验证+设计	1
17		使用除留余数法构造散列表	验证+设计	1
18	内部排序	冒泡排序	验证+设计	1
19		快速排序	验证+设计	1
20		堆排序	验证+设计	1

四、开始使用

读者迅速掌握 DS Lab 的基本使用方法，是顺利完成数据结构实验的重要前提。虽然 DS Lab 提供了非常人性化的操作界面，而且读者只需要掌握很少的几个核心功能，就可以顺利完成实验。但是，为了达到“做中学”的目的，在向读者介绍 DS Lab 的核心功能时，不会使用堆砌大量枯燥文字的方法，而是在后面的“实验 1”中，引导读者在使用 DS Lab 的过程中逐步掌握其基本使用方法，这样达到的效果最好。

第一部分 数据结构实验指导（C 语言）

实验 1 实验环境的使用

实验性质：验证
建议学时：1 学时

一、实验目的

- 熟悉数据结构集成实验环境 DS Lab的基本使用方法。
- 掌握线性表的顺序表示。
- 实现线性表的插入操作。

二、实验内容

请读者按照下面的步骤完成实验内容，同时，仔细体会 DS Lab 的基本使用方法。在本实验题目中，操作步骤会编写的尽量详细，并会对 DS Lab 的核心功能进行具体说明。但是，在后面的实验题目中会尽量省略这些内容，而将重点放在实验相关的源代码上。如有必要，读者可以回到本实验题目中，参考 DS Lab 的基本使用方法。

2.1 启动 DS Lab

在安装有 DS Lab的计算机上，可以使用两种不同的方法来启动 DS Lab：

- 在桌面上双击“Engintime DS Lab”图标。
  或者
- 点击“开始”菜单，在“程序”中的“Engintime DS Lab”中选择“Engintime DS Lab”。

2.2 注册用户并登录

DS Lab每次启动后都会弹出一个“登录”对话框，可以进行以下操作：

- 使用已有用户进行登录
  读者可以在“登录”对话框中填写已有的学号、姓名、密码完成登录。登录成功后，DS Lab的标题栏会显示出读者用来登录的学号和姓名。
- 注册新用户
  读者可以点击“注册”按钮，在弹出的“注册”窗口中填写基本信息、所属机构、密码、密保问题完成注册，并自动登录。

2.3 主窗口布局

DS Lab的主窗口布局由下面的若干元素组成：

- 顶部的菜单栏、工具栏。
- 停靠在左侧和底部的各种工具窗口。
- 余下的区域用来放置“起始页”和“源代码编辑器”窗口。

提示：菜单栏、工具栏和各种工具窗口的位置可以随意拖动。如果想恢复窗口的默认布局，选择“窗口”菜单中的“重置窗口布局”即可。

2.4 新建实验项目

新建一个实验项目的步骤如下：

1. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
2. 在“新建项目”对话框中，选择项目模板“001 线性表-前插入”。注意，其他模板会在后面的实验题目中使用。
3. 在“名称”中输入新项目使用的文件夹名称“lab1”。
4. 在“位置”中输入新项目保存在磁盘上的位置“C:\dslab”。
5. 点击“确定”按钮。

新建完毕后， DS Lab 会自动打开这个新建的项目。在“项目管理器”窗口中（如图 1-1 所示），根节点是项目节点，各个子节点是项目包含的文件夹或者文件。读者也可以使用“Windows 资源管理器”打开磁盘上的“C:\dslab\lab1”文件夹，查看项目中包含的源代码文件。

提示：右键点击“项目管理器”窗口中的项目节点，选择快捷菜单中的“打开所在的文件夹”，即可使用“Windows资源管理器”打开项目所在的文件夹。

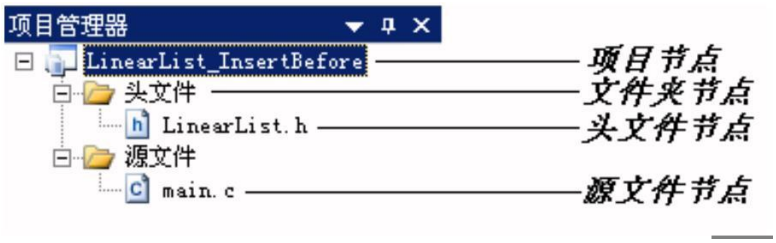


图 1-1：打开项目后的“项目管理器”窗口

2.5 绑定课时

绑定课时功能可以帮助读者将新建的项目与一个课时完成绑定，操作步骤如下：

1. 新建项目后，会自动弹出“绑定课时”对话框。
2. 在“绑定课时”对话框中，读者可以根据课时信息选择一个合适的课时完成绑定。
3. 绑定课时后，DS Lab的标题栏会显示当前项目所绑定的课时信息。
4. 读者可以登录开放实验管理平台，查看更详细的课时信息，平台使用方法请参见附录 2。

注意：如果不为项目绑定课时，将会影响到实验课考评成绩，并且不能提交作业。在未登录的情况下，不会弹出“绑定课时”对话框。

2.6 阅读实验源代码

该实验包含了一个头文件“LinearList.h”和一个 C 源文件“main.c”。下面对这两个文件的主要内容、结构和作用进行说明：

main.c文件

在“项目管理器”窗口中双击“main.c”打开此文件。此文件主要包含了以下内容：

1. 在文件的开始位置，使用预处理命令“#include "LinearList.h"”，包含了 LinearList.h文件。
2. 定义了 main 函数。在其中实现了线性表的初始化，然后调用了两次线性表的插入函数 InsertBefore，第一次调用插入成功，第二次调用插入失败。
3. 在 main函数的后面，定义了线性表的插入函数 InsertBefore。关于此函数的功能、参数和返回值，可以参见其注释。注意，此函数的函数体还不完整，留给读者完成。

LinearList.h文件

在“项目管理器”窗口中双击“LinearList.h”打开此文件。此文件主要包含了以下内容：

1. 包含用到的 C 标准库头文件。目前只包含了标准输入输出头文件“stdio.h”。
2. 包含其他模块的头文件。目前没有其他模块的头文件需要被包含。
3. 定义重要的数据结构。定义了与线性表相关的数据结构。
4. 声明函数。由于在“main.c”文件中，InsertBefore函数定义在 main函数之后，而且在 main函数中又调用了 InsertBefore函数，所以必须在头文件中声明 InsertBefore函数，否则无法通过编译。

提示：请读者认真理解这部分内容，其他实验题目中的源代码文件也严格遵守这些约定，如无特殊情况将不再进行如此详细的说明。

2.7 生成项目

使用“生成项目”功能可以将程序的源代码文件编译为可执行的二进制文件，操作如下：

1. 在“生成”菜单中选择“生成项目”（快捷键是 F7）。

在项目生成过程中，“输出”窗口会实时显示生成的进度和结果。如果源代码中不包含语法错误，会

在生成的最后阶段提示生成成功，如图 1-2所示：

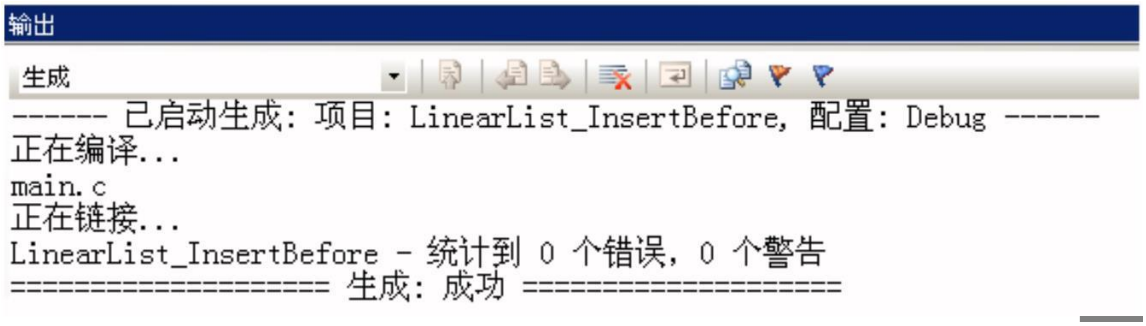


图 1-2：生成项目成功后的“输出”窗口

生成项目的过程，就是将项目所包含的每个 C 源代码文件（.c文件）编译为一个对象文件（.o文件），然后再将多个对象文件链接为一个目标文件（.exe文件）的过程。以本实验为例，成功生成项目后，默认会在“C:\dslab\lab1\Debug”目录下生成 “main.o”文件和 “LinearList_InsertBefore.exe”文件。

提示：读者可以通过修改项目名称的方法来修改生成的.exe文件的名称。方法是在“项目管理器”窗口中右键点击项目节点，选择快捷菜单中的“重命名”。待项目名称修改后，需要再次生成项目才能得到新的.exe文件。

2.8 解决语法错误

如果在源代码中存在语法错误，在生成项目的过程中，“输出”窗口会显示相应的错误信息（包括错误所在文件的路径，错误在文件中的位置，以及错误原因），并在生成的最后阶段提示生成失败。此时，在“输出”窗口中双击错误信息所在的行，DS Lab会使用源代码编辑器自动打开错误所在的文件，并定位到错误所在的代码行。

可以按照下面的步骤进行练习：

1. 在源代码文件中故意输入一些错误的代码（例如删除一个代码行结尾的分号）。
2. 生成项目。
3. 在“输出”窗口中双击错误信息来定位存在错误的代码行，并将代码修改正确。
4. 重复步骤 2、3，直到项目生成成功。

2.9观察点和演示模式

这里介绍DS Lab提供的两个重要功能：观察点和演示模式。

观察点

一个观察点对应一个函数的起始位置和结束位置（称这个函数为观察点函数）。在调试过程中，当程序执行到观察点函数的起始位置和结束位置时就会发生中断，就好像在这两个位置上添加了断点一样。并且，只要在观察点函数内部发生中断（包括命中断点、单步调试等），就会在“转储信息”窗口中显示观察点函数正在操作的数据信息，如果在“演示模式”下，还会在“演示流程”窗口中显示观察点函数的流程信息。

以本实验为例，“观察点”窗口如图 1-3所示（在“调试”菜单的“窗口”中选择“观察点”，可以打开“观察点”窗口），说明 InsertBefore函数是一个观察点函数。启动调试后，在 main.c文件 InsertBefore函数的开始位置和结束位置的左侧空白处，会显示观察点图标（与“观察点”窗口中左侧的图标一致），当程序执行到 InsertBefore函数的开始位置和结束位置时会发生中断。启动调试后，观察点窗口如图 1-4所示，可以显示出观察点所在的“文件”和“地址”。

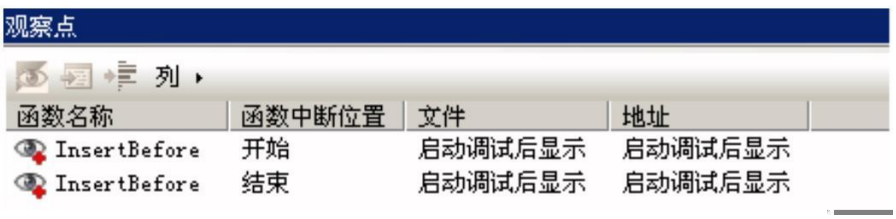


图 1-3：观察点窗口（未启动调试）

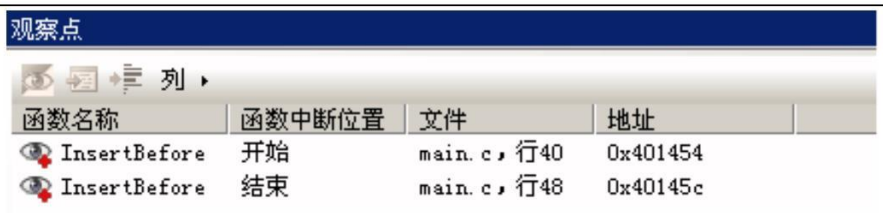


图 1-4：观察点窗口（启动调试）

演示模式

当 DS Lab工具栏上的“演示模式”按钮高亮显示时（如图 1-5所示 ），DS Lab处于演示模式。当在演示模式下调试观察点函数时，会忽略掉其函数体中的所有代码和断点，取而代之的是使用 DS Lab 提供的演示功能对观察点函数的执行过程和返回值进行演示。此特性可使观察点函数在还未完整实现的情况下，让读者了解到其应该具有的功能和执行过程，从而帮助读者正确实现此函数。

当工具栏上的“演示模式”按钮没有高亮显示时（鼠标点击工具栏上的“演示模式”按钮可以使其切换状态），DS Lab处于非演示模式。在非演示模式下调试观察点函数时，会使用其函数体中的代码和断点。

图 1-5：工具栏上的“演示模式”按钮。

2.10在演示模式下调试项目

读者可以按照下面的步骤，练习在演示模式下调试项目（主要是调试观察点函数）：

1. 保证工具栏上的“演示模式”按钮高亮显示。
2. 在“调试”菜单中选择“启动调试”（快捷键是 F5）。

启动调试后，程序会在观察点函数的开始位置处中断，如图 1-6所示。源代码编辑器左侧空白处显示了相应的图标，分别标识了观察点函数的起始位置和结束位置，以及下一行要执行的代码（黄色箭头）。

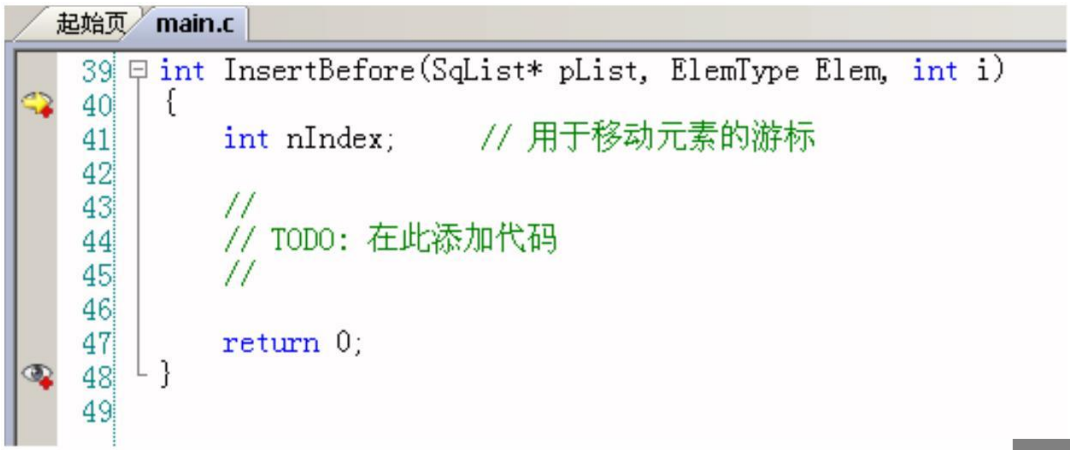


图 1-6：启动调试后，在观察点函数的开始位置中断。

在“可视化数据”窗口中（可以选择“调试”菜单“窗口”中的“可视化数据”打开此窗口），用可视化的方式显示了观察点函数正在操作的数据信息，如图 1-7所示。主要包含了如下内容：

- 由于使用了线性表的顺序表示（使用数组存储），所以可以发现数据元素在内存中的地址是连续的，两个相邻地址间的差值与一个元素占用的字节数（4 个字节）相同。
- 下标从 0 开始计数，位序从 1 开始计数。
- 线性表长度范围内的值使用十进制表示，这些值是线性表（数组）在初始化时存储的有效的值；范围外的值使用十六进制表示，由于未被初始化，所以使用了内存中的随机值，该记录所在表格的填充色为灰色。注意，在不同的计算机上调试时，内存中的随机值可能会不同。

- 并且使用带有绿色边框的记录指定出了元素的插入位置。
- 在调试的过程中，可以看到游标的移动，如果对应元素的当前值与上一次值不同时，该元素的值用红色的字体显示出来。

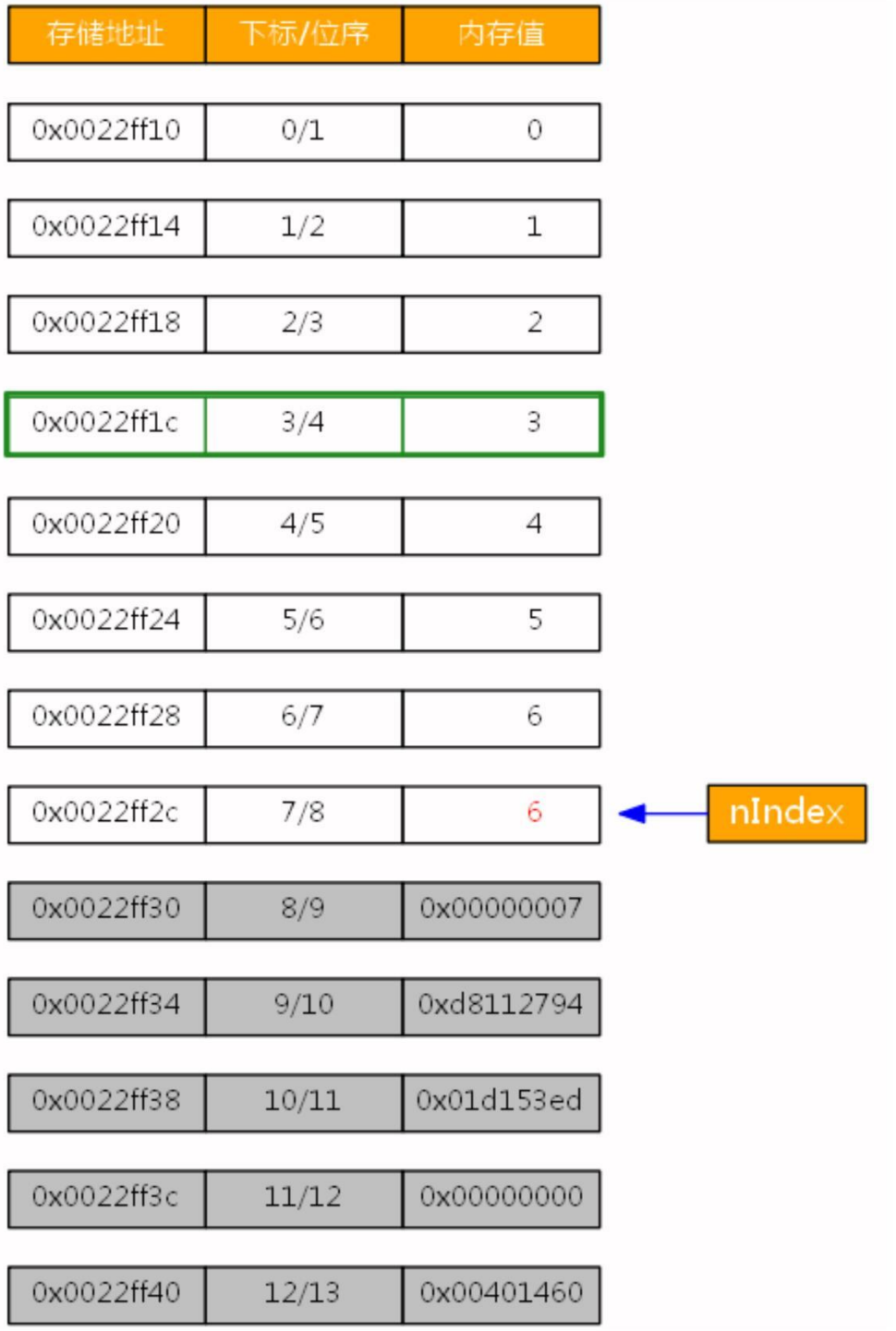


图 1-7：在观察点函数运行时的“可视化数据”窗口。

同时，在“转储信息”窗口中（可以选择“调试”菜单“窗口”中的“转储信息”打开此窗口），使用文本的方式显示了观察点函数正在操作的数据信息，如图 1-8所示。主要包含了如下内容：

1. 函数调用信息。对本次观察点函数的调用信息进行了描述，并显示了参数 i 和参数 Elem的值。
2. 函数返回信息。由于此时刚刚进入观察点函数，所以还无法显示其返回信息。当在观察点函数结束位置中断时，即可显示其返回信息。主要对观察点函数的返回值或者操作结果进行描述。
3. 重要的数据信息。主要是对线性表（pList）的信息进行了描述，包括：

- 线性表中元素在内存中的地址。由于使用了线性表的顺序表示（使用数组存储），所以可以发现数据元素在内存中的地址是连续的，两个相邻地址间的差值与一个元素占用的字节数（4个字节）相同。
- 线性表中元素在数组中的下标。从 0 开始计数。
- 线性表中元素的值。线性表长度范围内的值使用十进制表示，这些值是线性表（数组）在初始化时存储的有效的值；范围外的值使用十六进制表示，由于未被初始化，所以使用了内存中的随机值。注意，在不同的计算机上调试时，内存中的随机值可能会不同。
- 线性表中元素的位序。从 1 开始计数。如果是还未被占用的数组项，就显示为“空闲”。
- 标识出了元素的插入位置和要插入的元素值。

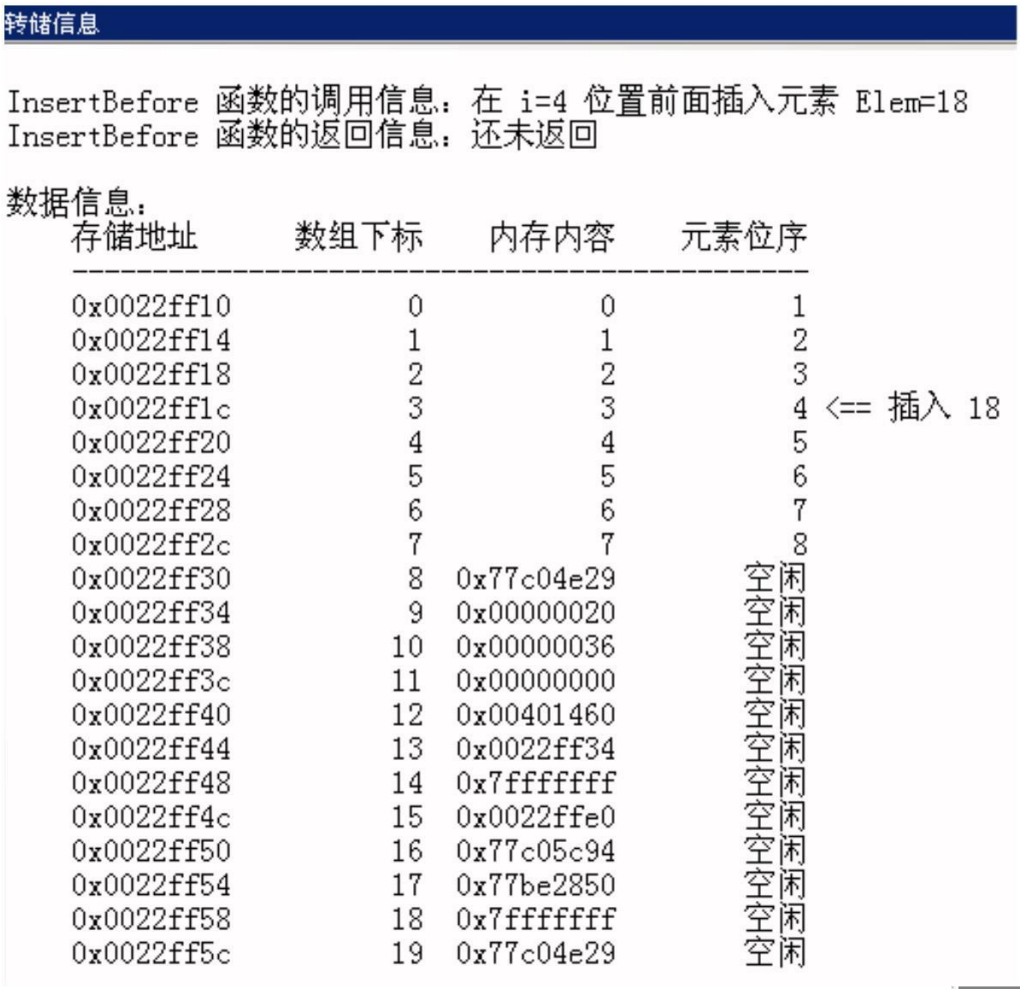


图 1-8：在观察点函数开始位置中断时的“转储信息”窗口。

按照下面的步骤继续调试：

1. 在“调试”菜单中选择“继续”（快捷键是 F5）。

由于是在“演示模式”下调试观察点函数，DS Lab会忽略掉函数体中的所有代码，取而代之的是使用DS Lab提供的演示功能对观察点函数的执行过程进行演示。所以 DS Lab会自动打开“演示流程”窗口（可以选择“调试”菜单“窗口”中的“演示流程”打开此窗口），在其中显示观察点函数的演示流程，如图1-9所示。观察点函数的演示流程通常采用简洁、直观的语言进行描述（一行描述可能会对应多行 C 源代码），偶尔也会在读者理解起来比较困难的地方提供 C 源代码的提示或者直接使用 C 源代码，目的就是为了方便读者将演示流程快速转换为 C 源代码。在“演示流程”窗口左侧的空白处，同样使用黄色箭头标识出了下一行要执行的代码（流程）。

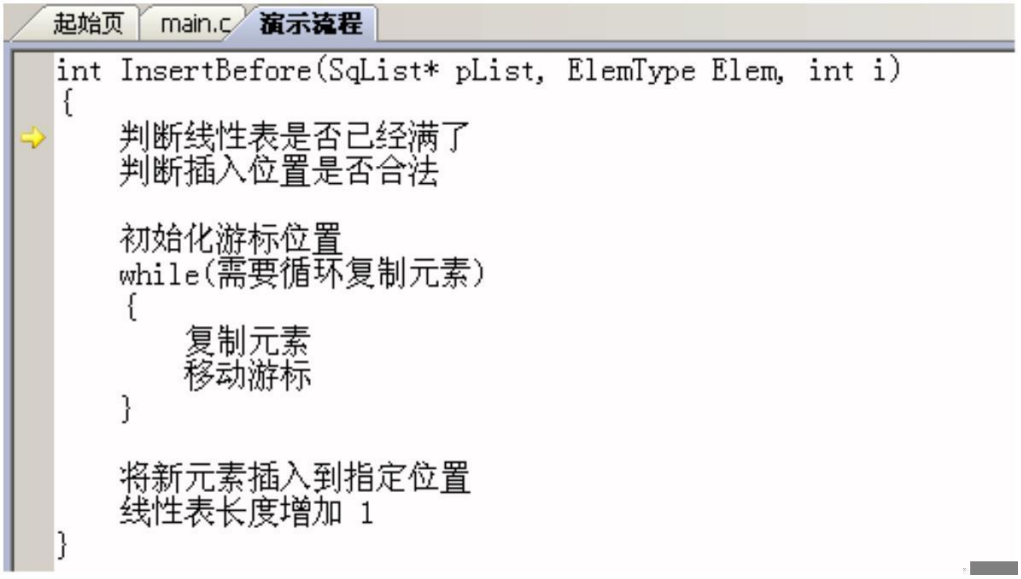


图 1-9：“演示流程”窗口。

按照下面的步骤继续调试：

1. 在“调试”菜单中重复选择“继续”，直到在观察点函数的结束位置中断。DS Lab会单步执行“演示流程”窗口中的每一行（包括循环）。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，例如：线性表元素的复制，当前的内存值与上一次的内存值进行比较，如果发生了变化，用红色的字体标识出来；游标的移动过程等，进而深入理解在线性表中插入元素的执行过程。

按照下面的步骤继续调试观察点函数被第二次调用的过程，理解在线性表中插入元素失败的执行过程：

1. 在“调试”菜单中重复选择“继续”，直到再次在观察点函数的结束位置中断。

按照下面的步骤结束此次调试：

1. 在“调试”菜单中重复选择“继续”，直到调试结束。或者，在“调试”菜单中选择“停止调试”。

读者可以在演示模式下重新启动调试，再次执行以上的步骤，仔细体会在“演示模式”下调试观察点函数的过程，以及在线性表中插入元素的过程。

2.11 验证项目（失败）

这里介绍DS Lab提供的另外一个重要功能：验证功能。

之前提到了 main.c文件中的 InsertBefore函数还不完整，是留给读者完成的。而当读者完成此函数后，往往需要使用调试功能、或者执行功能，来判断所完成的函数是否能够达到预期的效果，即是否与演示时函数的转储信息、执行过程和返回值完全一致。DS Lab提供的验证功能可以自动化的、精确的完成这个验证过程。

验证功能分为下面三个阶段：

1. 在“演示模式”下执行观察点函数（与工具栏上的“演示模式”按钮是否高亮无关），将产生的转储信息自动保存在文本文件 ValidateSource.txt中。
2. 在“非演示模式”下执行观察点函数，将产生的转储信息自动保存在文本文件 ValidateTarget.txt中。
3. 自动使用 DS Lab 提供的文本文件比较工具来比较这两个文件。当这两个文件中的转储信息完全一致时，报告“验证成功”；否则，报告“验证失败”。

当读者完成的函数与演示时函数的执行过程和返回值完全一致时，就会产生完全一致的转储信息，验

证功能就会报告“验证成功”；否则，验证功能就会报告“验证失败”，并且允许读者使用 DS Lab 提供的文本文件比较工具，来查看这两个转储信息文件中的不同之处，从而帮助读者迅速、准确的找到验证失败的原因，进而继续修改源代码，直到验证成功。

按照下面的步骤启动验证功能：

1. 在“调试”菜单中选择“开始验证”（快捷键是 Alt+F5）。在验证过程中，“输出”窗口会实时显示验证各个阶段的执行过程（如清单 1-1 所示），包括转储信息文件的路径、观察点函数的调用信息和返回信息、以及验证结果。由于 InsertBefore函数还不完整，所以验证失败。
2. 使用“输出”窗口工具条上的“比较”按钮（如图 1-10所示）查看两个转储信息文件中的内容，即它们之间的不同之处。

提示：

- ↙ 在转储信息文件中，只保存了观察点函数开始位置和结束位置的转储信息，并使用单线进行分隔。观察点函数多次被调用的转储信息之间，使用双线进行分隔。
- ↙ 在转储信息文件中，为了确保验证功能的准确性，某些信息会被忽略掉（不再显示或使用“N/A”替代），例如内存中的随机值等。

```
------ 已启动验证: 项目: LinearList_InsertBefore, 配置: Debug ------

验证第一阶段：正在使用“演示模式”生成转储信息，并写入源文件...
源文件路径：C:\dslab\lab1\Debug\ValidateSource.txt

InsertBefore 函数的调用信息：在 i=4 位置前面插入元素 Elem=18
InsertBefore 函数的返回信息：插入成功，返回 1

================================================================================

InsertBefore 函数的调用信息：在 i=11 位置前面插入元素 Elem=20
InsertBefore 函数的返回信息：插入失败，返回 0

================================================================================

验证第二阶段：正在使用“非演示模式”生成转储信息，并写入目标文件...
目标文件路径：C:\dslab\lab1\Debug\ValidateTarget.txt

InsertBefore 函数的调用信息：在 i=4 位置前面插入元素 Elem=18
InsertBefore 函数的返回信息：插入失败，返回 0

================================================================================

InsertBefore 函数的调用信息：在 i=11 位置前面插入元素 Elem=20
InsertBefore 函数的返回信息：插入失败，返回 0

================================================================================

验证第三阶段：正在比较转储信息源文件与目标文件的内容...
```

```
比较结果：转储信息源文件与目标文件的内容不同

==================== 验证结果：失败 ====================
```

清单 1-1：在“输出”窗口中显示的验证信息。

图 1-10：“输出”窗口工具栏上的“比较”按钮。

2.12 实现 InsertBefore函数

参考清单 1-2中的源代码，实现 InsertBefore函数。

提示：

- 在“观察点”窗口中，可以在函数名称上点击右键，选择快捷菜单中的“查看演示流程”，DS Lab 会打开“演示流程”窗口，并显示观察点函数的演示流程。这样，即使在没有启动调试的情况下，读者也可以方便的查看观察点函数的演示流程。
- 清单 1-2中的源代码使用“(*pList)”来回避指针操作，如果读者熟悉指针操作，可以进行改写。

```
int InsertBefore(SqList* pList, ElemType Elem, int i)
{
    int nIndex;
    if((*pList).nLength >= MAX_LENGTH)
        return 0;
    if(i < 1 || i > (*pList).nLength)
        return 0;

    nIndex = (*pList).nLength;
    while(nIndex >= i)
    {
        (*pList).Elements[nIndex] = (*pList).Elements[nIndex - 1];
        nIndex--;
    }

    (*pList).Elements[i - 1] = Elem;
    (*pList).nLength++;

    return 1;
}
```

清单 1-2：InsertBefore函数的参考源代码。

2.13 在非演示模式下调试项目

读者在实现了 InsertBefore 函数后，可以按照下面的步骤，练习在非演示模式下调试项目（主要是调试由读者实现的观察点函数）：

1. 在“生成”菜单中选择“生成项目”。如果读者编写的源代码中存在语法错误，修改这些错误，直到可以成功生成项目。
2. 鼠标点击工具栏上的“演示模式”按钮，使其切换到非高亮显示状态。

3. 在“调试”菜单中选择“启动调试”。程序会在观察点函数的开始位置处中断。
4. 在“调试”菜单中重复选择“逐过程”（快捷键是 F10），直到在观察点函数的结束位置中断。DS Lab 会单步执行观察点函数中的每一行源代码。在调试的过程中，每执行一行源代码后，仔细观察“可视化数据”窗口内所发生的变化，例如线性表元素的复制，游标的移动等，理解在线性表中插入元素的执行过程。
5. 在“调试”菜单中选择“继续”，直到再次在观察点函数的开始位置中断（第二次调用 InsertBefore 函数）。
6. 在“调试”菜单中重复选择“逐过程”，直到在观察点函数的结束位置中断。理解在线性表中插入元素失败时的执行过程。

以上的练习说明，DS Lab可以让读者在非演示模式下调试项目，并观察“可视化数据”窗口内容所发生的变化，从而理解每一行源代码对内存数据的操作结果。如果读者发现所编写的源代码存在异常行为（例如死循环、数组越界访问或者验证失败），可以在非演示模式下单步调试项目，来查找异常产生的原因。

2.14 验证项目（成功）

按照下面的步骤启动验证功能：

1. 在“调试”菜单中选择“开始验证”。

如果验证失败，读者可以参考之前的内容来查找原因并修改源代码中的错误，直到验证成功。

2.15 提交作业

如果读者写完了程序并通过了自动化验证，可以使用“提交作业”功能，将读者编写的源代码文件自动提交到服务器，供教师查看。步骤如下：

1. 选择“用户”菜单中的“提交作业”打开“提交作业”对话框。
2. 在“提交作业”对话框中，点击“继续提交”按钮，完成提交作业操作。
3. 如果需要重新绑定课时，点击“重新绑定课时”按钮，绑定符合要求的课时。
4. 提交作业成功后，读者可以登录开放实验管理平台，查看提交作业后的课时信息，平台使用方法参见附录 2。

注意：可以多次执行“提交作业”操作，后提交的作业会覆盖之前提交的作业。在未登录的情况下，不能提交作业。

2.16 总结

读者使用 DS Lab进行数据结构实验的步骤可以总结如下：

1. 启动 DS Lab。
2. 注册新用户，或使用已有用户登录。
3. 新建实验项目，并绑定课时。
4. 在演示模式下调试项目，理解观察点函数的执行过程（通常观察点函数还未完整实现）。
5. 结合观察点函数的演示流程，修改观察点函数的源代码，实现其功能。
6. 生成项目（排除所有的语法错误）。
7. 执行自动化验证。如果验证失败，可以使用“比较”功能在执行结果中查找问题，或者在“非演示模式”下调试项目，从而定位错误的位置，然后回到步骤 5。
8. 提交作业。
9. 退出 DS Lab。

2.17 获得帮助

如果读者在使用 DS Lab的过程中遇到问题需要专业的解答，或者有一些心得体会想和其他 DS Lab用户分享，欢迎加入 DS Lab网上论坛：

↙ 选择 DS Lab“帮助”菜单中的“论坛”。

或者

↙ 直接访问http://www.engintime.com/forum

这里列出了读者在使用 DS Lab 的过程中可能遇到的一些问题和使用技巧，用于帮助读者更好的使用 DS Lab，获得最佳的实验效果。

1. 读者时常会遇到在自己编写的源代码中存在死循环的情况，这就会造成 DS Lab 的调试功能，特别是验证功能无法自行结束。此时，读者可以选择“调试”菜单中的“停止调试”（快捷键是 Shift+F5）来强制结束这些功能。随后，读者可以检查自己编写的源代码，或者在非演示模式下单步调试项目，从而找到造成死循环的原因。
2. 读者时常会遇到的另外一个情况是“数组越界访问”。此时，DS Lab会弹出一个调试异常对话框，读者只要选择对话框中的“是”按钮，就可以立即定位到异常所在的代码行。
3. DS Lab作为一个 IDE环境，提供了强大的调试功能，包括单步调试、添加断点、查看变量的值、查看调用堆栈等。读者在调试过程中可以灵活使用这些功能，提高调试效率。注意，在演示模式下，观察点函数中的断点会被忽略。
4. 在演示模式下，对观察点函数只能进行单步调试（无论是按快捷键 F5，还是 F10），如果观察点函数中存在多次循环，会造成调试过程比较缓慢。此时，读者可以选择“调试”菜单中的“结束观察”（快捷键是 Shift+Alt+F5），直接跳转到观察点函数的结束位置中断。
5. 在“可视化数据”窗口中点击右键，选择菜单中的“图像保存为文件”菜单项，可以很方便的将图像保存保存成文件，用于完成实验报告等工作。
6. “输出”窗口、“演示流程”窗口以及“转储信息”窗口中的文本信息可以被选中并复制（但是不能修改），读者可以很方便的将这些信息保存下来，用于完成实验报告等工作。
7. DS Lab提供的实验项目通常不会在 Windows控制台窗口中打印输出任何信息，因为在“转储信息”窗口、“输出”窗口中已经为读者提供了足够多的信息。当然，读者也可以根据自己的喜好，使用 printf 函数，在 Windows 控制台窗口中打印输出一些信息（这些信息不会对“验证”结果产生任何影响）。如果读者想快速查看程序在 Windows控制台窗口中打印输出的信息，可以使用“调试”菜单中的“开始执行”功能（快捷键是 Ctrl+F5）。
8. 为读者提供了查看用户信息、修改用户信息、修改密码、修改密保问题和找回密码等功能。

实验 2 线性表的顺序表示及插入操作

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握线性表的顺序表示。
- 实现线性表的插入操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“002 线性表-后插入”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中实现了线性表的初始化，然后调用了两次线性表的插入函数 InsertAfter，第一次调用插入成功，第二次调用插入失败。

在 main函数的后面，定义了线性表的插入函数 InsertAfter，此函数的函数体还不完整，留给读者完成。

LinearList.h文件

定义了与线性表相关的数据结构并声明了相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口所发生的变化，例如线性表元素的复制，游标的移动等，理解在线性表中插入元素的执行过程。关于“可视化数据”窗口显示的内容，请参考“实验 1”第 2.10节中的描述。

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 InsertAfter函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 3 线性表的顺序表示及删除操作

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握线性表的顺序表示。
- 实现线性表的删除操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“003 线性表-删除”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中实现了线性表的初始化，然后调用了两次线性表的删除函数 Delete，第一次调用删除成功，第二次调用删除失败。

在 main函数的后面，定义了线性表的删除函数 Delete，此函数的函数体还不完整，留给读者完成。

LinearList.h文件

定义了与线性表相关的数据结构并声明了相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口所发生的变化，例如线性表元素的复制，游标的移动等，理解在线性表中删除元素的执行过程。关于“可视化数据”窗口显示的内容，请参考“实验 1”第 2.10节中的描述；

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 Delete函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 4 单链表的插入操作

实验性质：验证＋设计
建议学时：1 学时

一、 实验目的

- 掌握线性表的链式表示。
- 实现单链表的插入操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“004 单链表-插入”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中实现了单链表的初始化，然后调用了两次单链表的插入函数 InsertBefore，第一次调用插入成功，第二次调用插入失败，并在 main函数的最后销毁了单链表。

在 main 函数的后面，定义了单链表的插入函数 InsertBefore，此函数的函数体还不完整，留给读者完成。

SingleLinkList.h文件

定义了与单链表相关的数据结构并声明了相关的操作函数。

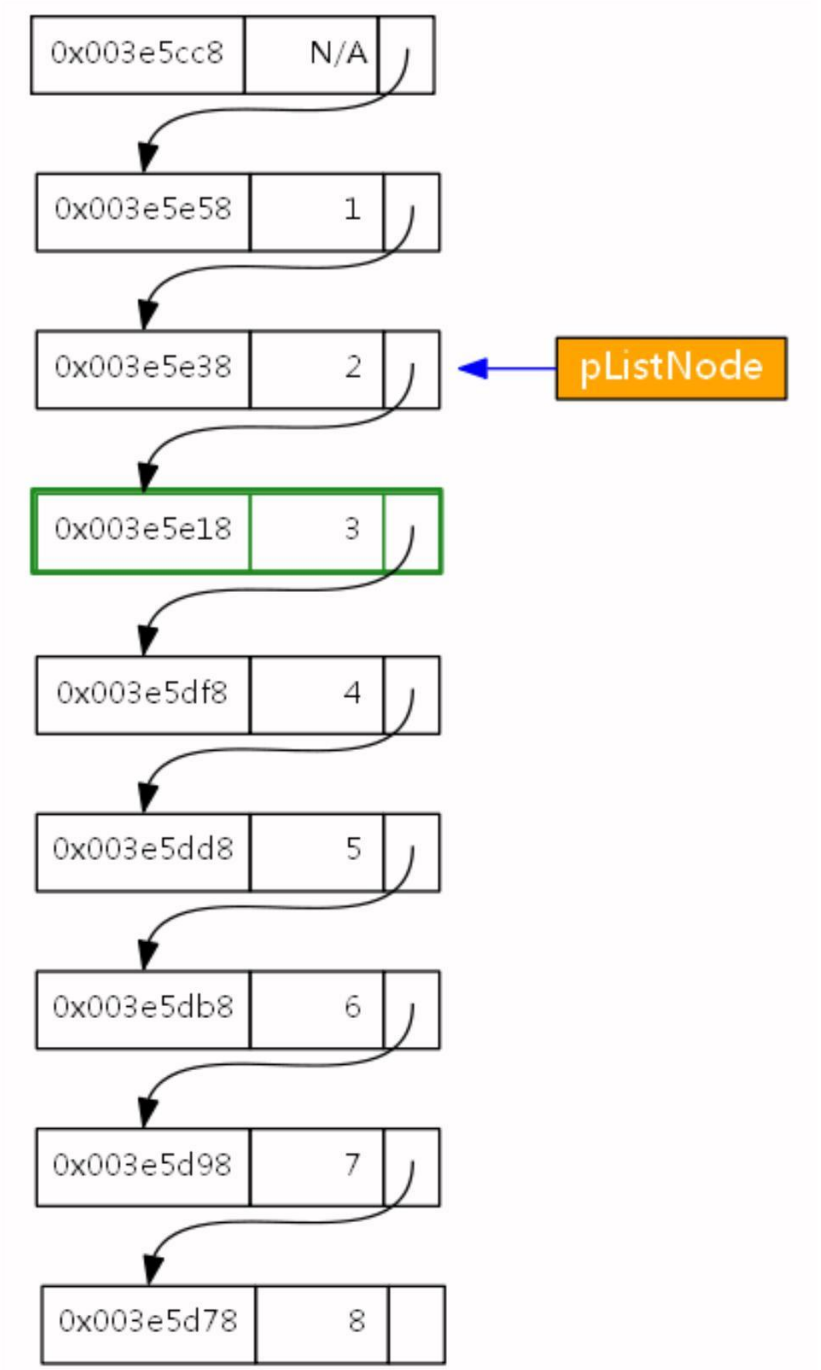
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解在单链表中插入元素的执行过程。“可视化数据”窗口显示了单链表（pListHead）的数据信息（如下图所示），包括：

- 单链表中元素在内存中的地址。由于在初始化时，单链表的元素是连续分配的，所以可能有部分相邻元素的地址是连续的，但是，从所有元素的地址上观察，可以发现地址是非连续的，特别是有新元素插入后。
- 单链表中元素的值。头节点的值被忽略。
- 前一个结点的指针指向了后一个结点的地址。
- 用绿色的边框标识出了结点的插入位置。
- 在调试的过程中，可以看到游标的移动。
- 新插入的结点用红色的边框标识出来。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 InsertBefore函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 5 单链表的删除操作

实验性质：验证＋设计
建议学时：1 学时

一、 实验目的

- 掌握线性表的链式表示。
- 实现单链表的删除操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“005 单链表-删除”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中实现了单链表的初始化，然后调用了两次单链表的删除函数 Delete，第一次调用删除成功，第二次调用删除失败，并在 main函数的最后销毁了单链表。

在 main函数的后面，定义了单链表的删除函数 Delete，此函数的函数体还不完整，留给读者完成。

SingleLinkList.h文件

定义了与单链表相关的数据结构并声明了相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解在单链表中删除元素的执行过程。关于“可视化数据”窗口显示的内容，请参考“实验 4”第2.3节中的描述。

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 Delete函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 6 双向循环链表的插入操作

实验性质：验证＋设计
建议学时：1 学时

一、 实验目的

- 掌握线性表的链式表示。
- 实现双向循环链表的插入操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“006 双向循环链表-插入”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中实现了双向循环链表的初始化，然后调用了两次双向循环链表的插入函数 InsertBefore，第一次调用插入成功，第二次调用插入失败，并在main函数的最后销毁了双向循环链表。

在 main 函数的后面，定义了双向循环链表的插入函数 InsertBefore，此函数的函数体还不完整，留给读者完成。

DoubleLinkList.h文件

定义了与双向循环链表相关的数据结构并声明了相关的操作函数。

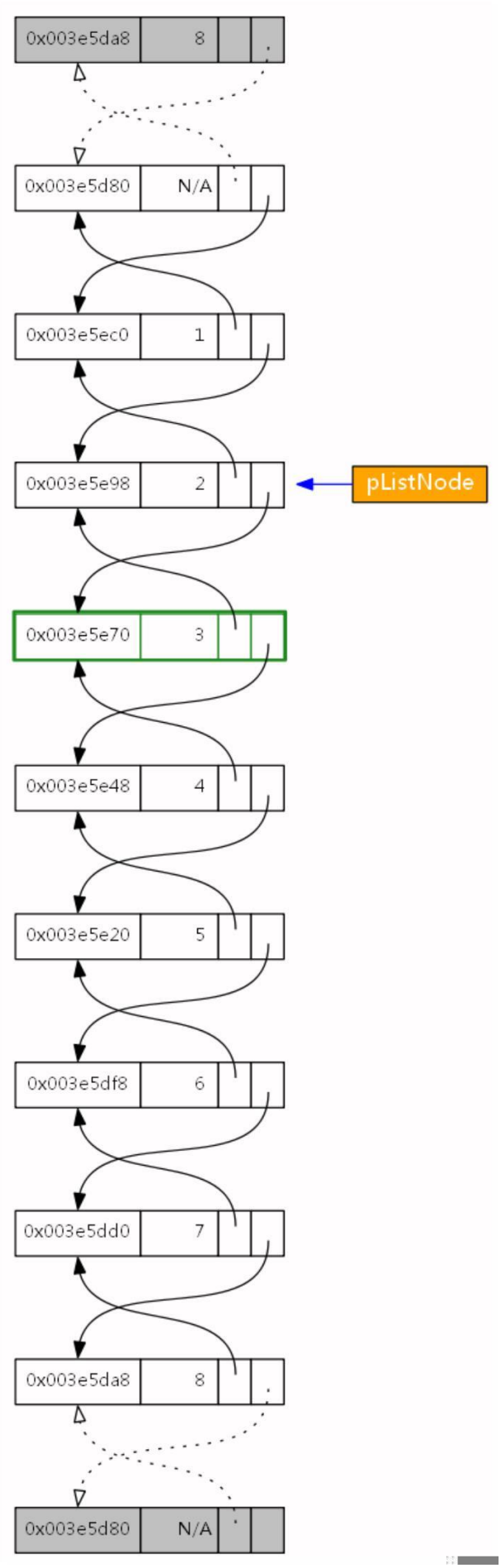
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解在双向循环链表中插入元素的执行过程。“可视化数据”窗口显示了双向循环链表（pListHead）的数据信息（如下图所示），包括：

- 双向循环链表中元素在内存中的地址。由于在初始化时，双向循环链表的元素是连续分配的，所以可能有部分相邻元素的地址是连续的，但是，从所有元素的地址上观察，可以发现地址是非连续的，特别是有新元素插入后。
- 双向循环链表中元素的值。头节点的值被忽略。
- 前一个结点的指针指向后一个结点的地址，后一个结点的指针指向前一个结点的地址。
- 与头节点相连的尾节点，该尾节点的底色为灰色，并且指向地址的指针用虚线表示，该结点与双向循环链表的最后一个结点表示的是同一个结点。与最后一个节点相连的头结点类似。
- 用绿色的边框标识出了元素的插入位置。
- 可以看到游标的移动过程。
- 用红色的边框标识出插入的新元素。


0x003e5da8
8
0x003e5d80
N/A
0x003e5ec0
1
0x003e5e98
2
pListNode
0x003e5e70
3
0x003e5e48
4
0x003e5e20
5
0x003e5df8
6
0x003e5dd0
7
0x003e5da8
8
0x003e5d80
N/A

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 InsertBefore函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 7 双向循环链表的删除操作

实验性质：验证＋设计
建议学时：1 学时

一、 实验目的

- 掌握线性表的链式表示。
- 实现双向循环链表的删除操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“007 双向循环链表-删除”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中实现了双向循环链表的初始化，然后调用了两次双向循环链表的删除函数 Delete，第一次调用删除成功，第二次调用删除失败，并在 main函数的最后销毁了双向循环链表。

在 main 函数的后面，定义了双向循环链表的删除函数 Delete，此函数的函数体还不完整，留给读者完成。

DoubleLinkList.h文件

定义了与双向循环链表相关的数据结构并声明了相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解在双向循环链表中删除元素的执行过程。关于“可视化数据”窗口显示的内容，请参考“实验 6”第 2.3节中的描述。

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 Delete函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 8 字符串的顺序表示及插入操作

实验性质：验证＋设计
建议学时：1 学时

一、 实验目的

- 掌握字符串的顺序表示。
- 实现字符串的插入操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“008 顺序字符串-插入”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中初始化了源字符串与目标字符串，然后调用了两次插入函数 InsertBefore尝试将目标字符串插入到源字符串的不同位置，第一次调用插入成功，第二次调用插入失败。

在 main 函数的后面，定义了字符串插入函数 InsertBefore，此函数的函数体还不完整，留给读者完成。

SeqString.h文件

定义了与顺序存储的字符串相关的数据结构并声明了相关的操作函数。

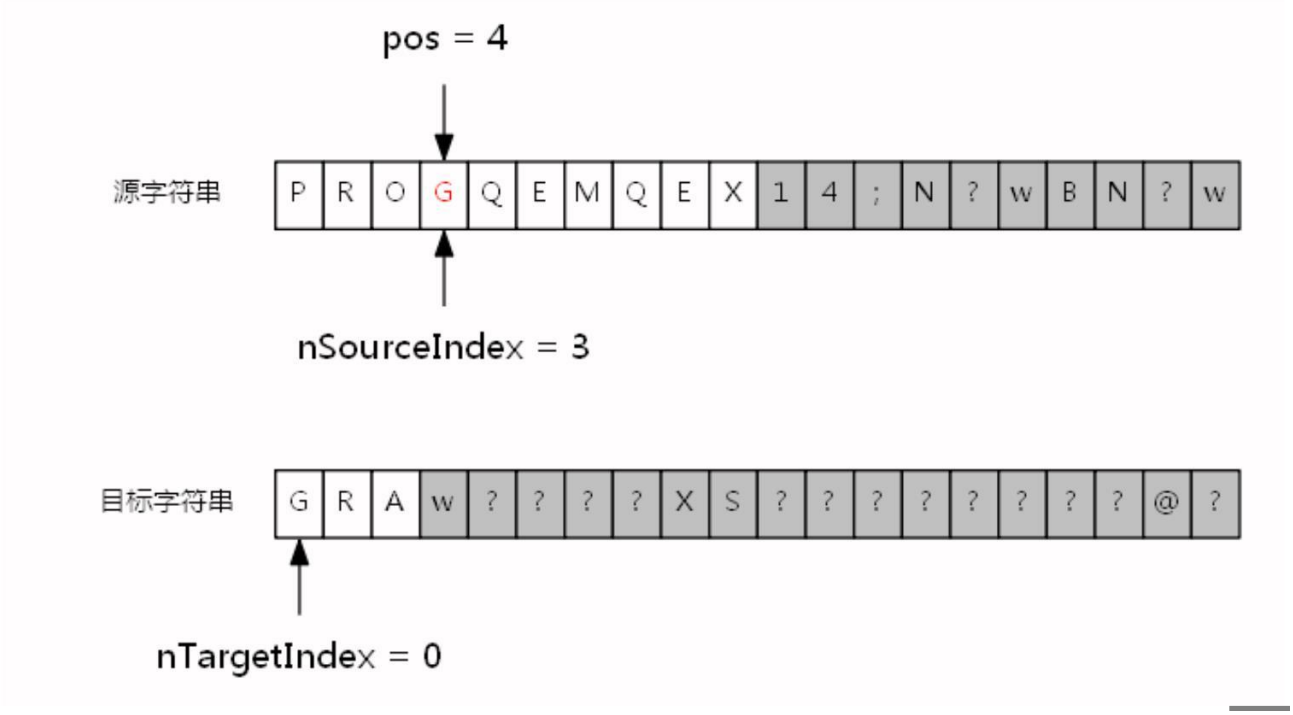
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解在源字符串中插入目标字符串的执行过程。“可视化窗口”窗口显示了源字符串（pSource）和目标字符串（pTarget）的数据信息（如下图所示），包括：

- 字符串的内容。字符串中超过其长度的部分，字符的底色为灰色。注意，由于字符串中超过其长度的部分使用的是内存中的随机数据，如果一个随机数据（一个字节）的值是一个能够显示的ASCII编码，就正常显示其字符；否则，就使用字符“?”代替。
- 在源字符串中，标识出了目标字符串要插入的位置和游标。
- 在执行的过程中，可以看到源字符串的游标或者目标字符串的游标的移动，当前字符串与上一次字符对应的字符不相同时，该字符就会用红色显示。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 InsertBefore函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 9 二叉树的先序遍历操作

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握二叉树的链式存储结构。
- 实现二叉树的先序遍历操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“009 二叉树-先序遍历”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先创建了二叉树，然后调用 PreOrder函数对二叉树进行先序遍历操作，并将产生的先序序列保存在全局变量 g_string中，最后销毁了二叉树。

在 main 函数的后面，分别定义了二叉树的先序遍历函数 PreOrder，以及二叉树的创建函数和销毁函数。其中，先序遍历函数 PreOrder的函数体还不完整，留给读者完成。

BinaryTree.h文件

定义了与二叉树相关的数据结构并声明了相关的操作函数和全局变量。

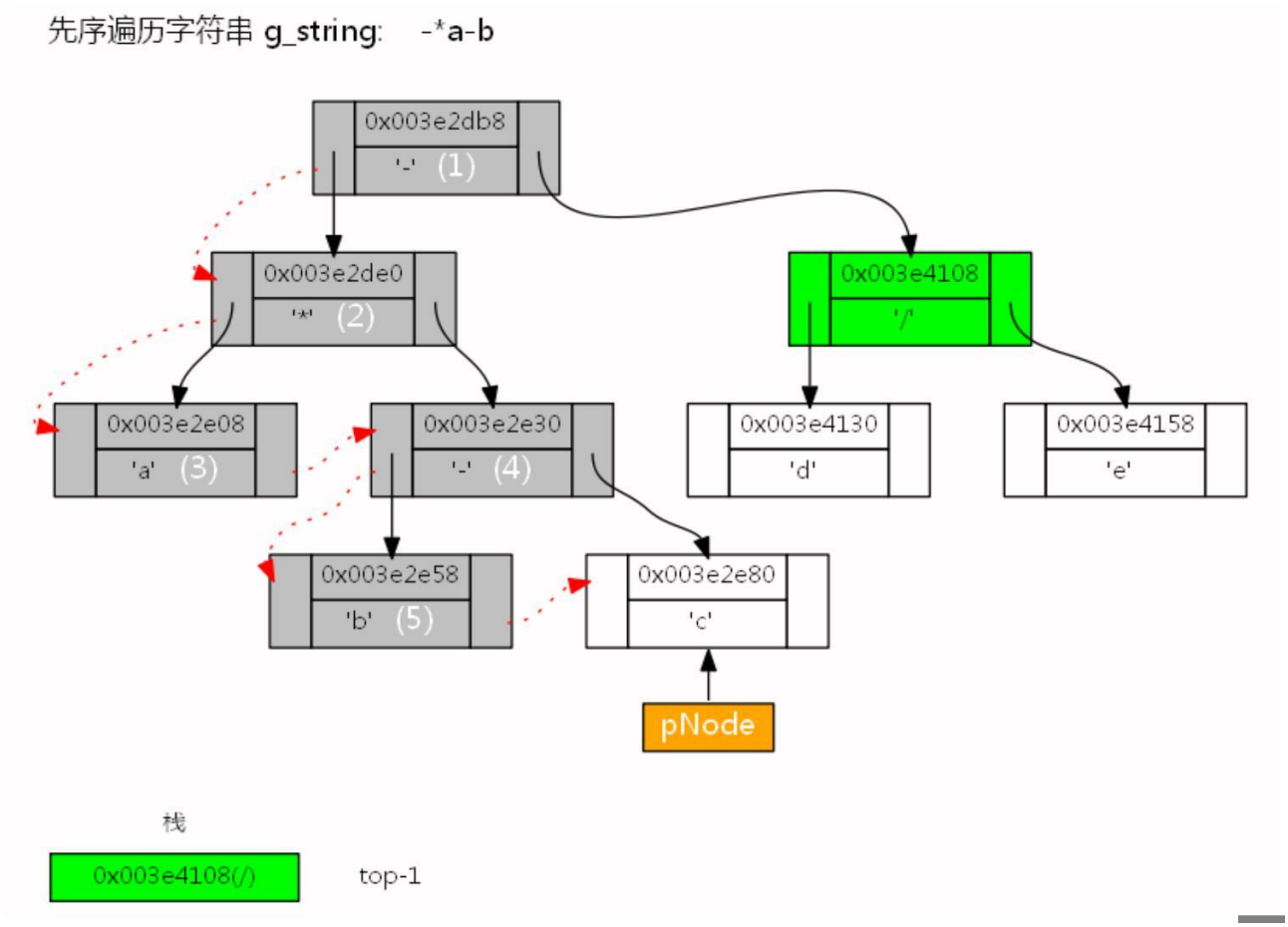
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解先序遍历二叉树的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 先序遍历二叉树时，由二叉树各个节点的值，按照其被访问的先后顺序所组成先序遍历字符串。
- 二叉树的详细信息。包括：
  a) 二叉树节点的值和地址。
  b) 已访问过的结点的底色为灰色，并且在该结点中，用白色的字体标出了访问过的序号。
  c) 在栈中的结点的底色为绿色。
  d) 用红色带箭头的虚线表示出了访问节点的轨迹。
  e) 游标指向了当前的节点。
- 先序遍历二叉树时使用的栈。包括了二叉树节点的地址、值（空地址的值会被忽略），栈的底色为绿色。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 PreOrder函数编写源代码，要求利用栈实现非递归的先序遍历算法，并生成先序序列字符串。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测先序序列字符串生成的是否正确，所以，为了顺利通过验证，请读者在遍历二叉树的同时一定要在全局变量 g_string中生成先序序列字符串。

实验 10 二叉树的中序遍历操作

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握二叉树的链式存储结构。
- 实现二叉树的中序遍历操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“010 二叉树-中序遍历”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中首先创建了二叉树，然后调用 InOrder 函数对二叉树进行中序遍历操作，并将产生的中序序列保存在全局变量 g_string中，最后销毁了二叉树。

在 main函数的后面，分别定义了二叉树的中序遍历函数 InOrder，以及二叉树的创建函数和销毁函数。其中，中序遍历函数 InOrder的函数体还不完整，留给读者完成。

BinaryTree.h文件

定义了与二叉树相关的数据结构并声明了相关的操作函数和全局变量。

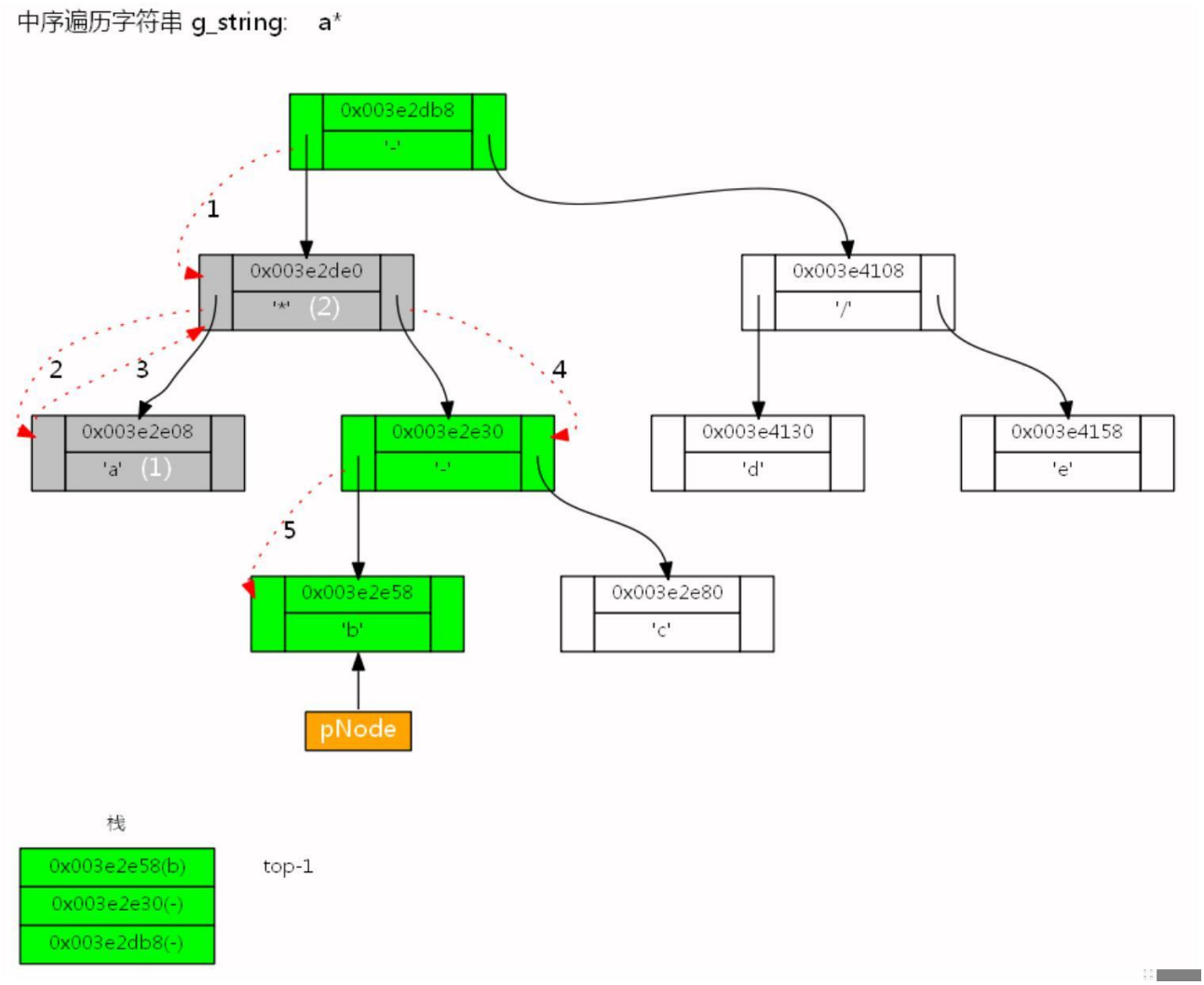
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解中序遍历二叉树的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 中序遍历二叉树时，由二叉树各个节点的值，按照其被访问的先后顺序所组成中序遍历字符串。
- 二叉树的详细信息。包括：
  a) 二叉树节点的值和地址。
  b) 已访问过的结点的底色为灰色，并且在该结点中，用白色的字体标出了访问过的序号。
  c) 在栈中的结点的底色为绿色。
  d) 用红色带箭头的虚线表示出了访问节点的轨迹，在该条虚线的序号表示形成轨迹的先后次序。
  e) 游标指向了当前的节点。
- 中序遍历二叉树时使用的栈。包括了二叉树节点的地址、值，栈的底色为绿色。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 InOrder函数编写源代码，要求利用栈实现非递归的中序遍历算法，并生成中序序列字符串。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测中序序列字符串生成的是否正确，所以，为了顺利通过验证，请读者在遍历二叉树的同时一定要在全局变量 g_string中生成中序序列字符串。

实验 11 二叉树的后序遍历操作

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握二叉树的链式存储结构。
- 实现二叉树的后序遍历操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“011 二叉树-后序遍历”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中首先创建了二叉树，然后调用 PostOrder 函数对二叉树进行后序遍历操作，并将产生的后序序列保存在全局变量 g_string中，最后销毁了二叉树。

在 main函数的后面，分别定义了二叉树的后序遍历函数 PostOrder，以及二叉树的创建函数和销毁函数。其中，后序遍历函数 PostOrder的函数体还不完整，留给读者完成。

BinaryTree.h文件

定义了与二叉树相关的数据结构并声明了相关的操作函数和全局变量。

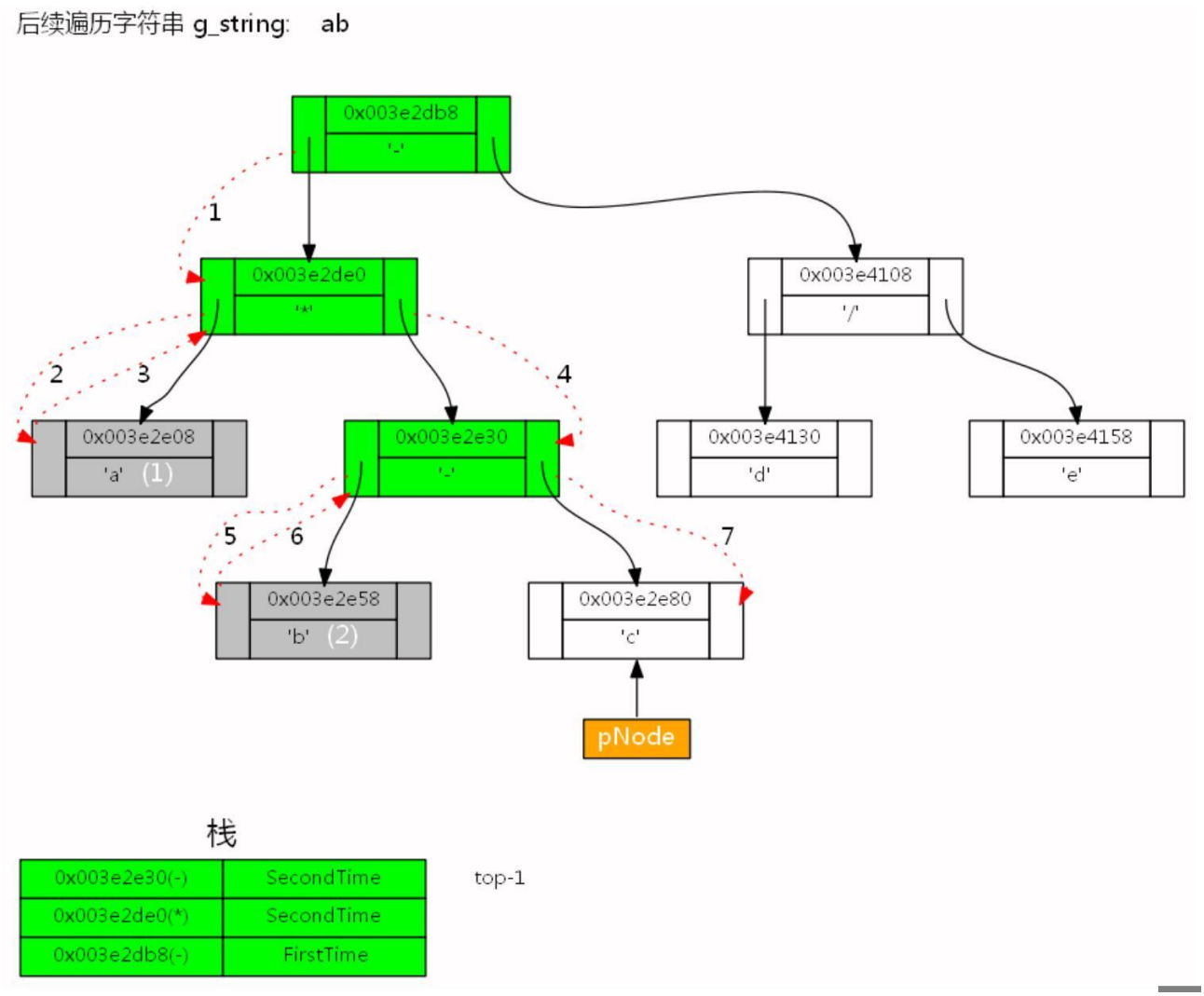
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解后序遍历二叉树的执行过程。“可视化数据”窗口显示的数据信息（如下图），包括：

- 后序遍历二叉树时，由二叉树各个节点的值，按照其被访问的先后顺序所组成的后续遍历字符串。
- 二叉树的详细信息。包括：
  a) 二叉树节点的值和地址。
  b) 已访问过的结点的底色为灰色，并且在该结点中，用白色的字体标出了访问过的序号。
  c) 在栈中的结点的底色为绿色。
  d) 用红色带箭头的虚线表示出了访问节点的轨迹，在该条虚线的序号表示形成轨迹的先后次序。
  e) 游标指向了当前的节点。
- 后序遍历二叉树时所使用的两个栈，一个栈用于存储父节点，另一个栈用于存储父节点入栈的次数。包括了二叉树父节点的地址、值、入栈次数，栈的底色为绿色。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 PostOrder函数编写源代码，要求利用栈实现非递归的后序遍历算法，并生成后序序列字符串。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测后序序列字符串生成的是否正确，所以，为了顺利通过验证，请读者在遍历二叉树的同时一定要在全局变量 g_string中生成后序序列字符串。

实验 12 线索二叉树

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握线索二叉树的链式存储结构。
- 实现二叉树的中序线索化。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“012 线索二叉树”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先创建了一棵二叉树，然后调用函数 InOrderThreading对二叉树进行线索化。

在 main函数的后面，定义了函数 InOrderThreading， 此函数的函数体还不完整，留给读者完成。

ThreadBinaryTree.h文件

定义了与线索二叉树相关的数据结构并声明了相关的操作函数和全局变量。注意，此头文件中还包含了栈模块的头文件 Stack.h。

Stack.h文件

定义了与栈相关的数据结构并声明了相关的操作函数。

Stack.c文件

定义了与栈相关的操作函数。

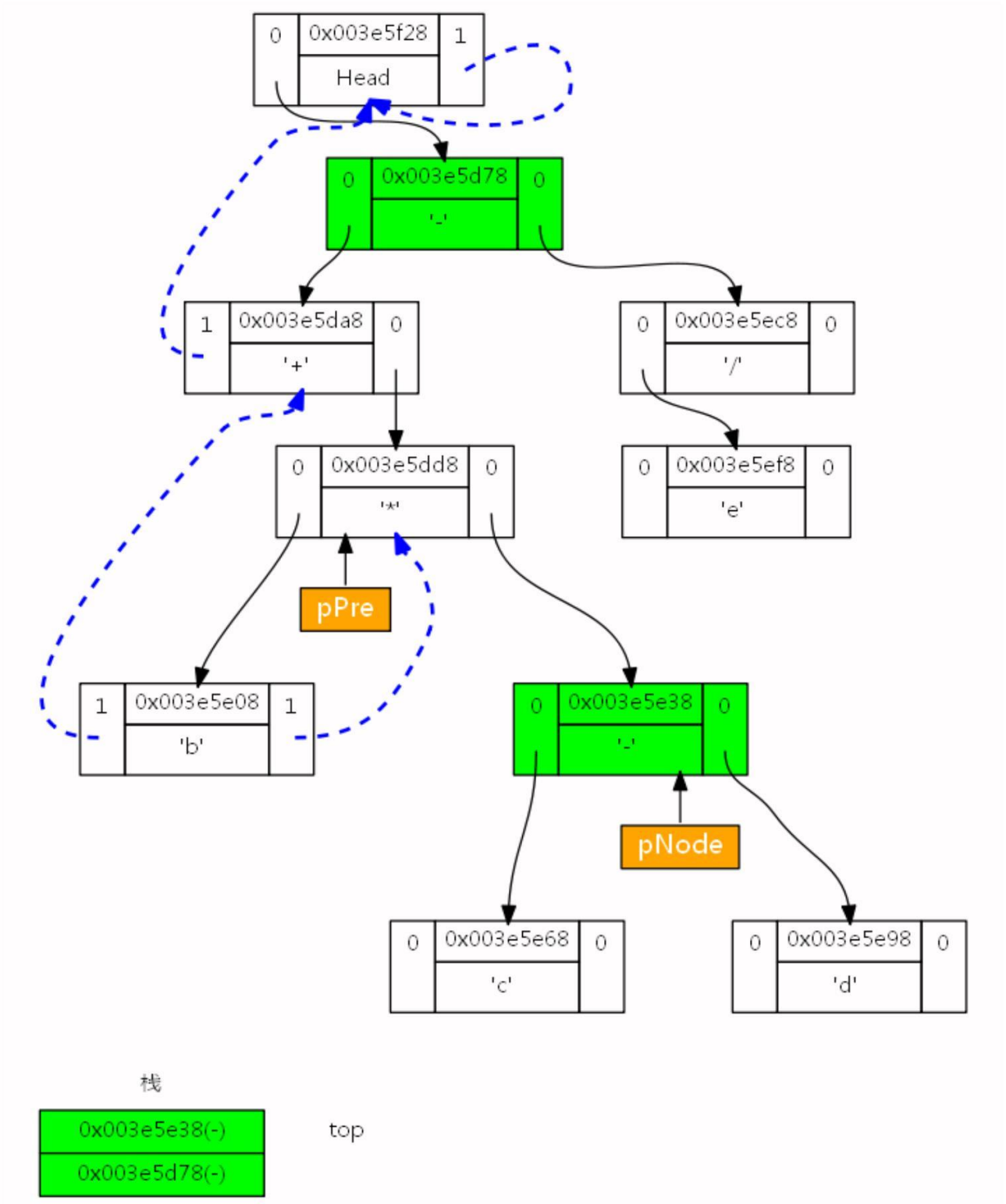
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解二叉树线索化的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 头结点的详细信息。包括结点地址及前驱后继的指向。
- 二叉树的详细信息。
  a) 二叉树节点的值和地址。
  b) 使用指针指向了该结点的左孩子和右孩子。
  c) 在栈中的结点的填充色为绿色。
  d) 用蓝色带箭头的虚线表示出了二叉树的线索。
  e) pPre游标指向了刚刚访问过的结点，pNode游标指向了当前的结点。
- 线索化二叉树时使用的栈。显示了已入栈的二叉树节点的地址、值（空地址的值会被忽略）。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 InOrderThreading 函数编写源代码。注意，尽量不要使用递归算法，而是使用栈来保存还未线索化的子树的根结点，并尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5启动调试后重复按 F10单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 13 赫夫曼树的构造

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握赫夫曼树的存储结构。
- 实现赫夫曼树的构造过程。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“013 赫夫曼树”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先创建了赫夫曼树的初始状态，然后调用函数 HuffmanTree构造赫夫曼树。

在 main函数的后面，定义了赫夫曼函数 HuffmanTree，此函数的函数体还不完整，留给读者完成。

HuffmanTree.h文件

定义了与赫夫曼树相关的数据结构并声明了相关的操作函数。

2.3 在演示模式下调试项目

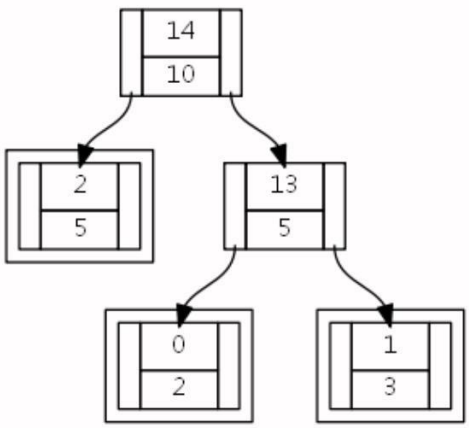
按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解赫夫曼函数的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 赫夫曼树的存储结构，包括内部结点和外部结点。
- 可以看到最小和次小节点权值及游标。可以显示最小和次小结点的查找过程，已查找出的最小和次小结点的填充色为灰色。
- 赫夫曼树的存储结构，对应的当前值与上一次值不相同时，用红色的字体显示出来。
- 在根据已选出的最小和次小结点，构造的哈夫曼树中，内部结点使用双线的边框，外部结点使用单线的边框。

	下标	权值	parent	lchild	rchild	
	0	2	13	-1	-1	
i→	1	3	13	-1	-1	
	2	5	14	-1	-1	←Index1
	3	7	-1	-1	-1	
	4	11	-1	-1	-1	
	5	13	-1	-1	-1	
	6	17	-1	-1	-1	
	7	19	-1	-1	-1	
	8	23	-1	-1	-1	
	9	29	-1	-1	-1	
	10	31	-1	-1	-1	
	11	37	-1	-1	-1	
	12	41	-1	-1	-1	
	13	5	14	0	1	←Index2
j→	14	10	-1	2	13	
	15	-1	-1	-1	-1	
	16	-1	-1	-1	-1	
	17	-1	-1	-1	-1	
	18	-1	-1	-1	-1	
	19	-1	-1	-1	-1	
	20	-1	-1	-1	-1	
	21	-1	-1	-1	-1	
	22	-1	-1	-1	-1	
	23	-1	-1	-1	-1	
	24	-1	-1	-1	-1	



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为HuffmanTree函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按F5启动调试后重复按F10单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤1。

实验 14 一般树的遍历

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握一般树的二叉链表表示法。
- 实现一般树的先序遍历算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“014 一般树的遍历”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先创建了一般树，然后调用 PreOrder函数对一般树进行先序遍历操作，并将产生的先序序列保存在全局变量 g_string中。

在 main函数的后面，定义了一般树的先序遍历函数 PreOrder， 此函数的函数体还不完整，留给读者完成。

CSTree.h文件

定义了与一般树相关的数据结构并声明了相关的操作函数和全局变量。注意，此头文件中还包含了栈模块的头文件 Stack.h。

Stack.h文件

定义了与栈相关的数据结构并声明了相关的操作函数。

Stack.c文件

定义了与栈相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

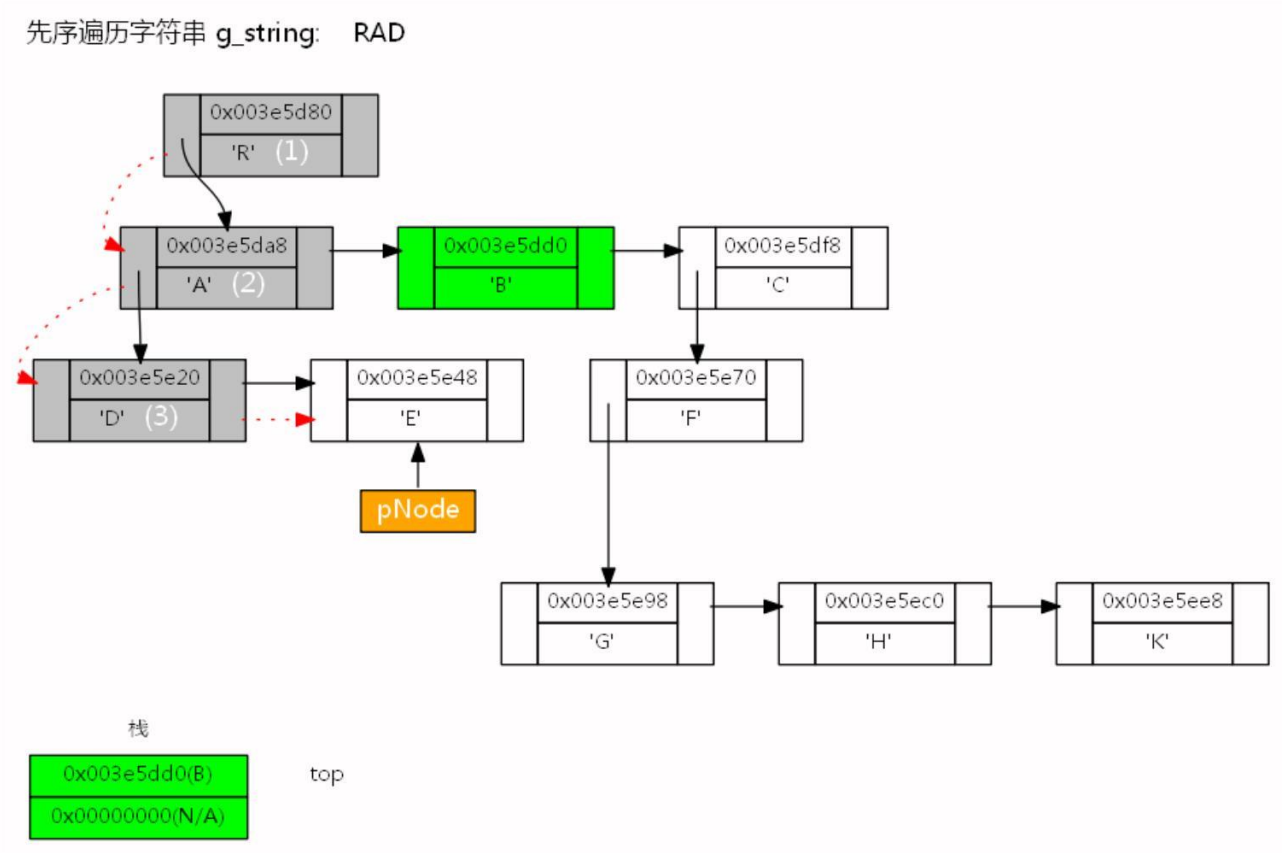
1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解一般树先序遍历的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 先序遍历一般树时，树节点按照其被访问的先后顺序所组成的先序遍历字符串。
- 一般树的详细信息。
  a) 一般树节点的值和地址。
  b) 已访问过的结点的底色为灰色，并且在该结点中，用白色的字体标出了访问过的序号。
  c) 在栈中的结点的底色为绿色。
  d) 用红色带箭头的虚线表示出了访问节点的轨迹。

e） 游标指向了当前的节点。

↙ 先序遍历一般树时使用的栈，栈的填充色为绿色。包括了一般树节点的地址、值（空地址的值会被忽略）。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 PreOrder函数编写源代码，要求利用栈实现非递归的先序遍历算法，并生成先序序列字符串。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5启动调试后重复按 F10单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测先序序列字符串生成的是否正确，所以，为了顺利通过验证，请读者在遍历一般树的同时一定要在全局变量 g_string中生成先序序列字符串。

实验 15 图的深度优先搜索

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握图的邻接表存储结构。
- 实现图的深度优先搜索。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“015 图-深度优先搜索”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先初始化了图，然后调用 DepthFirstSearch函数对图进行深度优先搜索操作，并在全局变量 visited中记录下了顶点访问的先后次序,最后销毁了图。

在 main 函数的后面，分别定义了图的深度优先搜索函数 DepthFirstSearch，以及图的初始化函数和销毁函数。其中，深度优先搜索函数 DepthFirstSearch的函数体还不完整，留给读者完成。

Graph.h文件

定义了与图相关的数据结构并声明了相关的操作函数和全局变量。

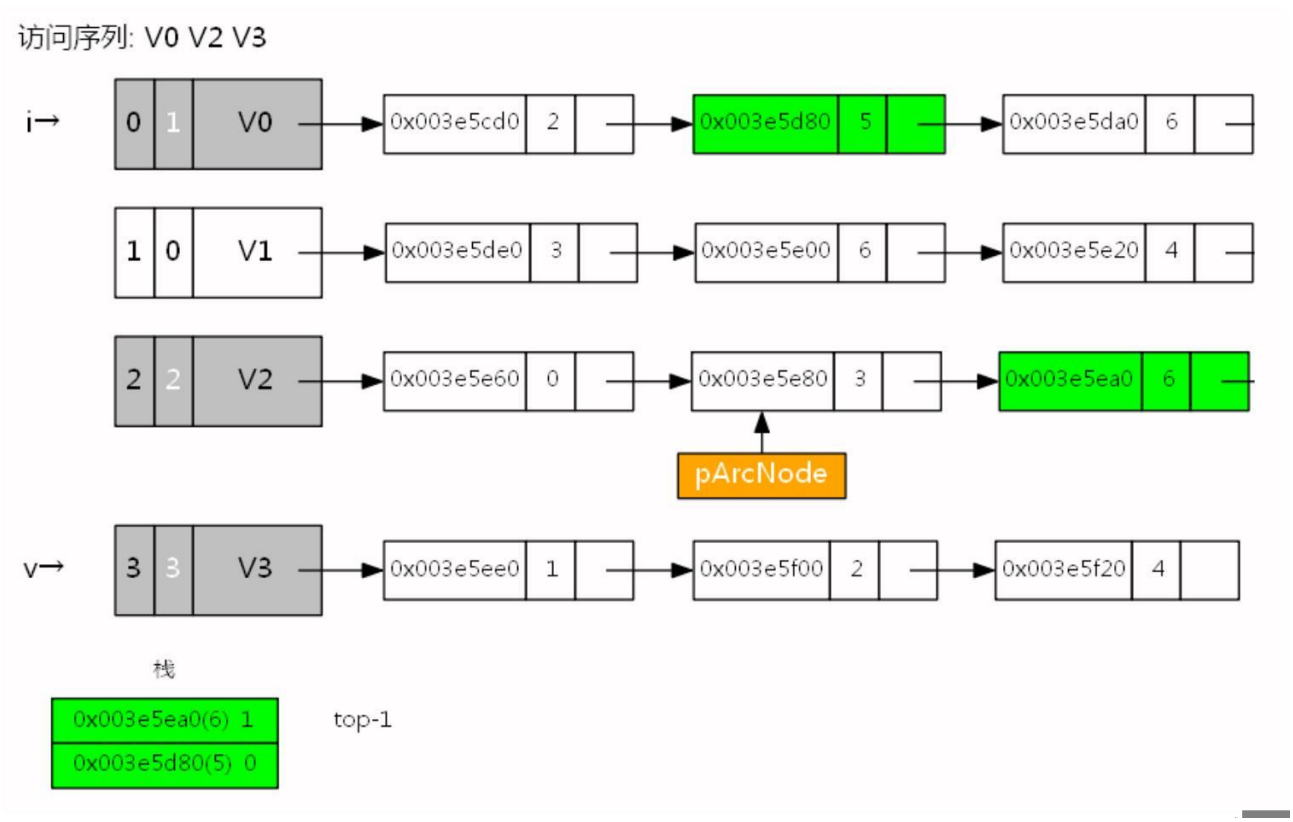
2.3 在演示模式下调试项目

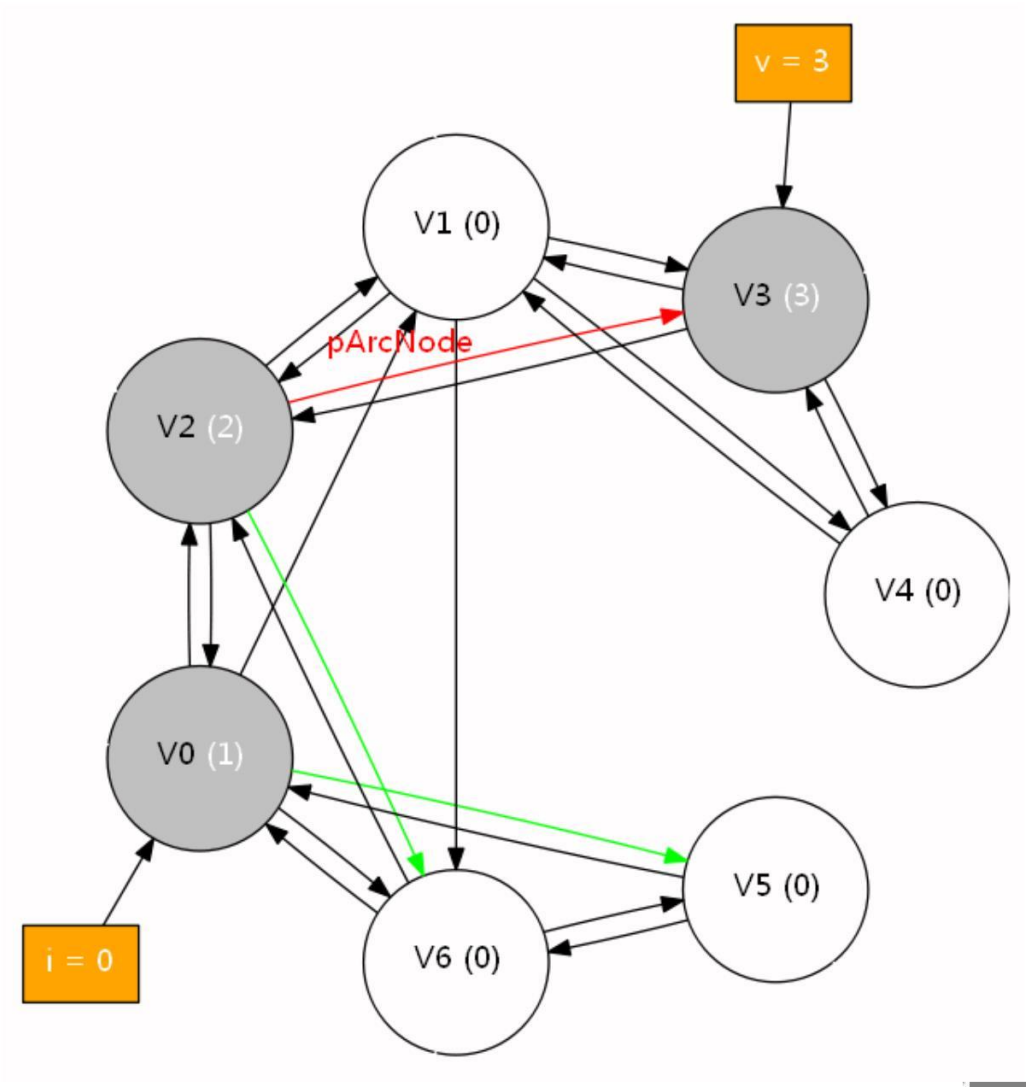
按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解深度优先搜索的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 对图进行深度优先搜索时产生的顶点访问序列。
- 图的邻接表。
  a) 包括顶点访问标志（0 表示顶点未被访问，大于 0 表示顶点被访问的先后次序，使用白色的字体表示）、顶点名称字符串，已访问过的顶点的填充色为灰色。
  b) 顶点的邻接点（即图中的边、或者弧），如果该邻接顶点在栈中，填充色为绿色。
  c) 在调试的过程中可以看到游标的移动。
- 深度优先搜索使用的栈。包括了邻接点的地址、值，以及数组下标。
- 在图中，顶点与边的关系。
  a) 已访问过的顶点的填充色为灰色，并使用白色的字体标出了顶点的访问序列。
  b) 相邻顶点连接的边，如果是游标指向的当前边，用红色表示，如果在栈中，用绿色表示。
  c) 在调试的过程中，可以看到游标的移动。


访问序列: V0 V2 V3
i→
0 1 V0
0x003e5cd0 2
0x003e5d80 5
0x003e5da0 6
1 0 V1
0x003e5de0 3
0x003e5e00 6
0x003e5e20 4
2 2 V2
0x003e5e60 0
0x003e5e80 3
0x003e5ea0 6
pArcNode
v→
3 3 V3
0x003e5ee0 1
0x003e5f00 2
0x003e5f20 4
栈
0x003e5ea0(6) 1
0x003e5d80(5) 0
top-1


v = 3
V1 (0)
V3 (3)
pArcNode
V2 (2)
V4 (0)
V0 (1)
V5 (0)
V6 (0)
i = 0

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 DepthFirstSearch 函数编写源代码，要求利用栈实现非递归的深度优先搜索算法。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测顶点访问序列、以及顶点访问数组生成的是否正确，所以，为了顺利通过验证，请读者在对图进行搜索的同时一定要在全局变量 visited中记录下顶点访问的先后次序。

实验 16 图的广度优先搜索

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握图的邻接表存储结构。
- 实现图的广度优先搜索。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“016 图-广度优先搜索”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先初始化了图和队列，然后调用 BreadthFirstSearch函数对图进行广度优先搜索操作，并在全局变量 visited中记录下了顶点访问的先后次序,最后销毁了图。

在 main 函数的后面，分别定义了图的广度优先搜索函数 BreadthFirstSearch，以及图的初始化函数和销毁函数。其中，广度优先搜索函数 BreadthFirstSearch的函数体还不完整，留给读者完成。

Graph.h文件

定义了与图相关的数据结构并声明了相关的操作函数和全局变量。注意，此头文件中还包含了队列模块的头文件 Queue.h。

Queue.h文件

定义了与队列相关的数据结构并声明了相关的操作函数。

Queue.c文件

定义了与队列相关的操作函数。

2.3 在演示模式下调试项目

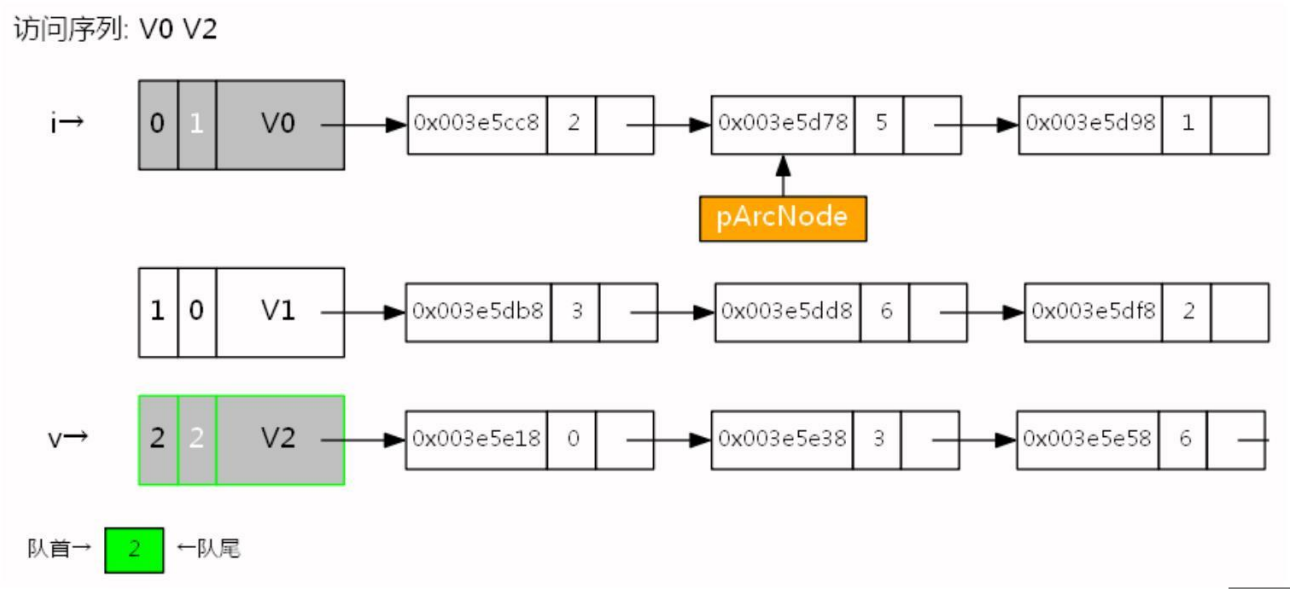
按照下面的步骤调试项目：

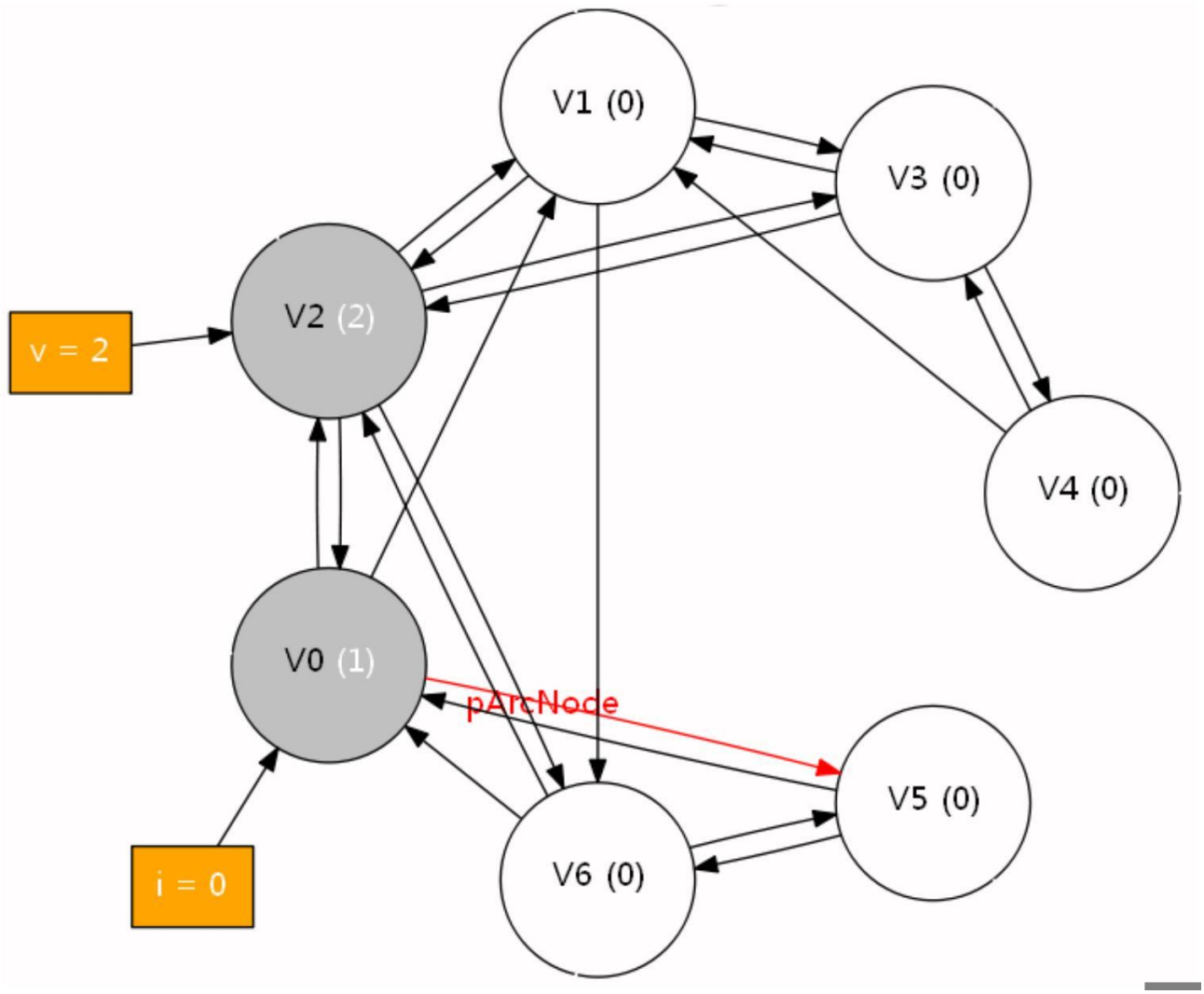
1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解广度优先搜索的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 对图进行广度优先搜索时产生的顶点访问序列。
- 图的邻接表。
  a) 顶点访问标志（0 表示顶点未被访问，大于 0 表示顶点被访问的先后次序，用白色的字体表示）、顶点名称字符串，已访问过的顶点的填充色为灰色，在队列中的顶点使用绿色的边框。
  b) 顶点的邻接点（即图中的边、或者弧）。
  c) 在调试的过程中，可以看到游标的移动。

- 广度优先搜索使用的队列。
- 在图中，已访问过的顶点的填充色为灰色，并且标出了访问顶点的序号；如果两顶点之间的边是游标指向的当前边，就用红色显示该条边，在调试的过程中，可以看到游标的移动。





2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为BreadthFirstSearch函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测顶点访问序列、以及顶点访问数组生成的是否正确，所以，为了顺利通过验证，请读者在对图进行搜索的同时一定要在全局变量 visited中记录下顶点访问的先后次序。

实验 17 拓扑排序

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握图的邻接表存储结构。
- 实现图的拓扑排序操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“017 拓扑排序”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先初始化了图和栈，然后调用 TopoSort函数对图进行拓扑排序操作，并在全局变量 TopoSortResult中记录下了顶点的下标,最后销毁了图。

在 main函数的后面，分别定义了计算图中顶点入度函数 FindInDegree和图的拓扑排序函数 TopoSort，以及图的初始化函数和销毁函数。其中，计算顶点入度函数 FindInDegree和拓扑排序函数 TopoSort的函数体还不完整，留给读者完成。

TopologicalSort.h文件

定义了与图相关的数据结构并声明了相关的操作函数和全局变量。注意，此头文件中还包含了栈模块的头文件 Stack.h。

Stack.h文件

定义了与栈相关的数据结构并声明了相关的操作函数。

Stack.c文件

定义了与栈相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

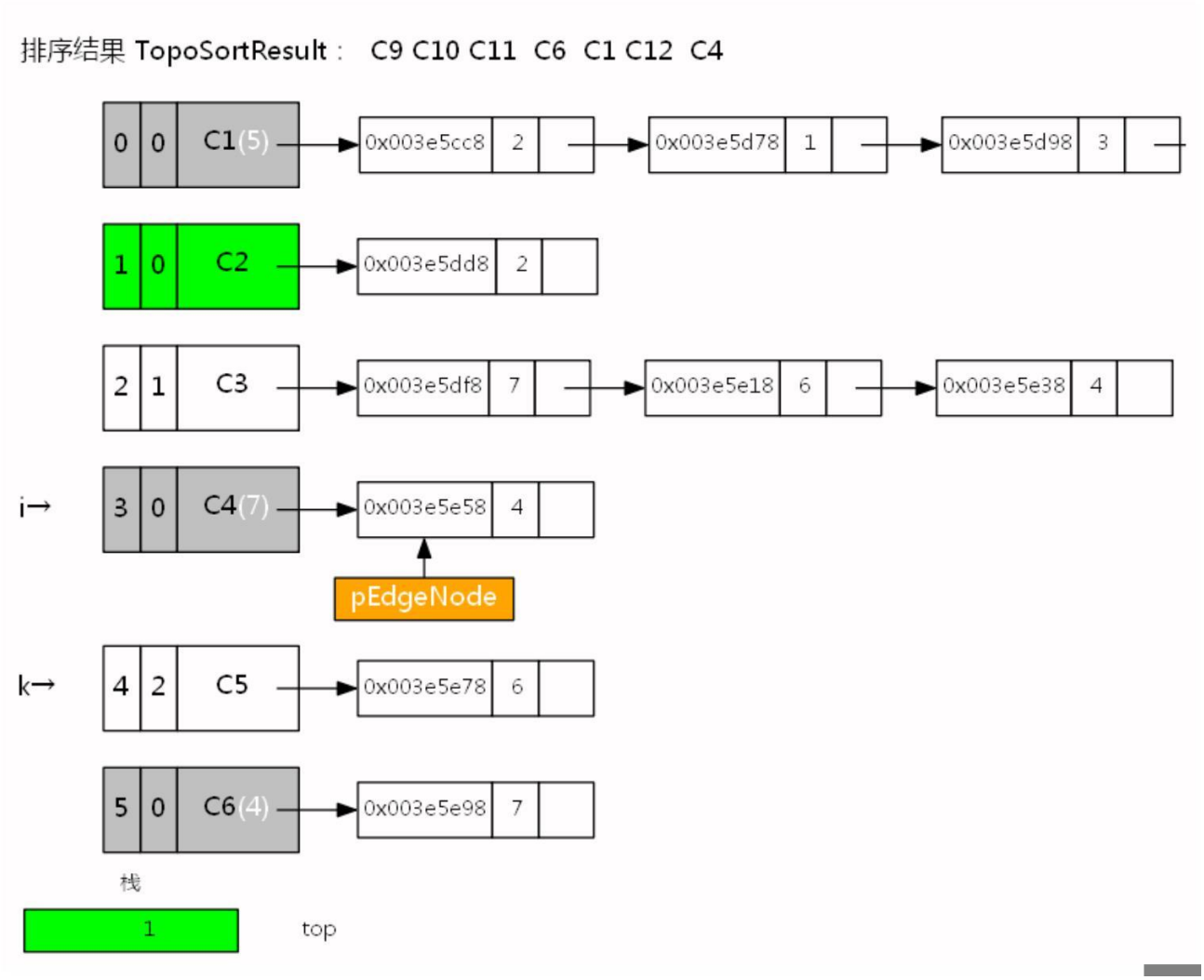
1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

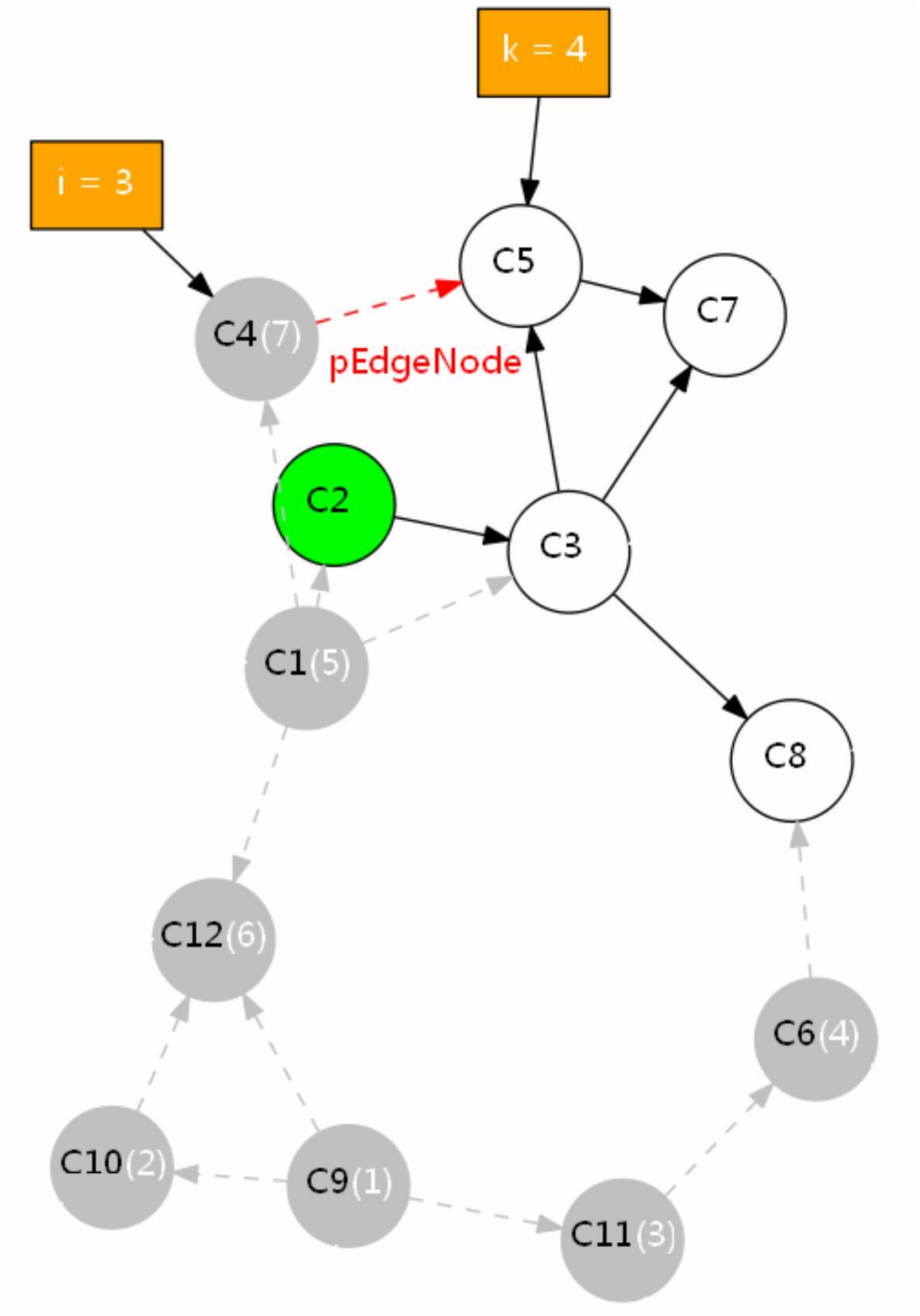
在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解拓扑排序的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 对图进行拓扑排序时产生的排序结果。
- 图的邻接表。
  a) 包括顶点的入度、顶点名称字符串，已排序过的顶点的填充色为灰色，用白色的字体显示出拓扑排序的序号，在栈中的顶点的填充色为绿色。
  b) 顶点的邻接点（即图中的边）。

c） 在调试的过程中，可以看到游标的移动。

- 拓扑排序时使用的栈。
- 在图中，已排序过的顶点的填充色为灰色，用白色的字体显示出拓扑排序的序号，并且与该顶点关联的边为灰色的虚线；在栈中顶点的填充色为绿色，游标指向当前边，该边的颜色为红色；在调试的过程中，可以看到游标的移动。





2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为FindInDegree和TopoSort函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测拓扑排序字符串生成的是否正确，所以，为了顺利通过验证，请读者在拓扑排序的同时一定要在全局变量 g_string中生成字符串。

实验 18 关键路径

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握图的邻接表存储结构。
- 实现图的关键路径查找操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“018 关键路径”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中首先初始化了图和栈，然后调用CriticalPath函数查找图中关键路径,最后销毁了图。

在 main 函数的后面，分别定义了计算图中顶点入度函数 FindInDegree 和图的拓扑排序函数TopologicalSort和图的关键路径查找函数CriticalPath，以及图的初始化函数和销毁函数。其中，计算顶点入度函数 FindInDegree和拓扑排序函数 TopologicalSort以及图中关键路径查找函数 CriticalPath的函数体还不完整，留给读者完成。

CriticalPath.h文件

定义了与图相关的数据结构并声明了相关的操作函数和全局变量。注意，此头文件中还包含了栈模块的头文件 Stack.h。

Stack.h文件

定义了与栈相关的数据结构并声明了相关的操作函数。

Stack.c文件

定义了与栈相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

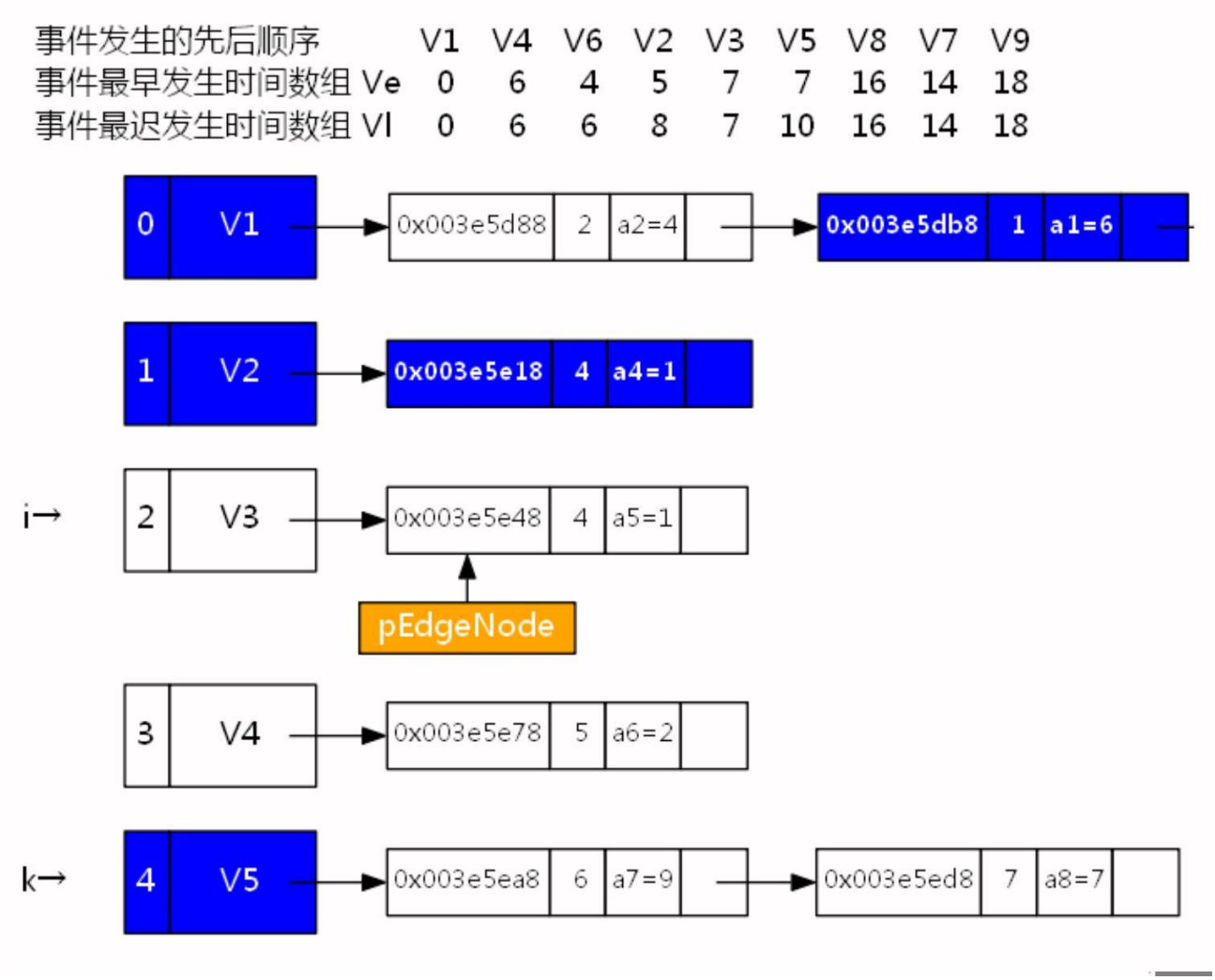
在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解关键路径函数的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

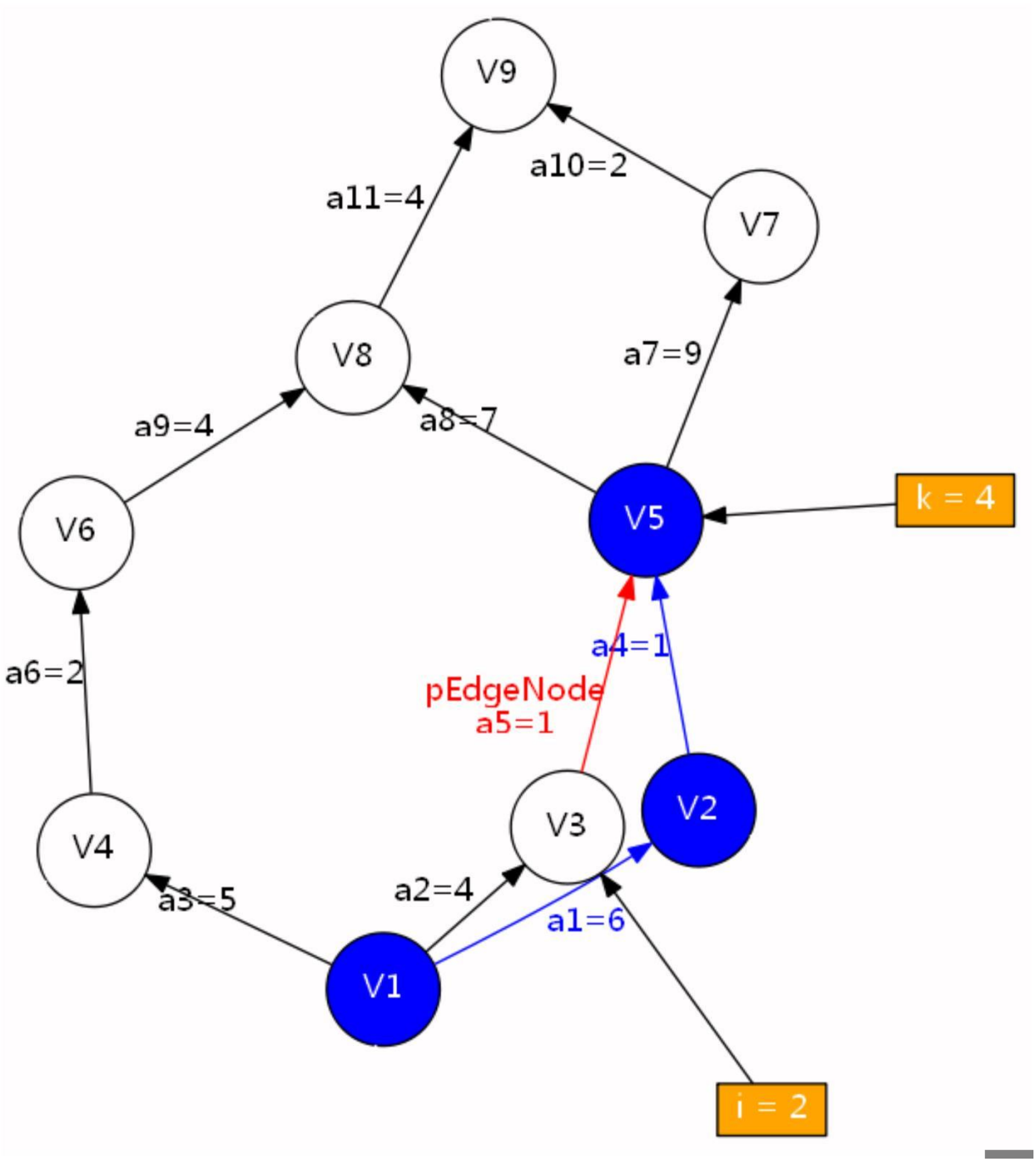
- 求图中关键路径所需的事件最早发生时间数组和事件最迟发生时间数组。
- 图的邻接表。
  a) 包括顶点名称字符串，如果该顶点是关键路径上关联的顶点，该顶点的填充色为蓝色，字体颜色为白色；如果该顶点在栈中，该顶点的填充色为绿色。
  b) 顶点的邻接点（即图中的边、或者弧），和邻接点中包含的权值、 顶点下标和活动名称，如

果该邻接顶点所包含的活动为关键路径上活动，该邻接点的填充色为蓝色，字体颜色为白色。

c) 在调试的过程中，可以看到游标的移动。

- 计算事件最迟发生时间数组时使用的栈。
- 在图中，栈中的顶点的填充色为绿色；关键路径上的顶点的填充色为蓝色，字体的颜色为白色；关键路径上边的颜色为红色；游标指向的当前边的颜色为红色；在调试的过程中，可以看到游标的移动。





2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 FindInDegree、TopologicalSort 和 CriticalPath 函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在搜索关键路径的过程中，把边（或者弧）的 isCritical域设置为 TRUE，表示这条边（或者弧）是关键路径上的关键活动。

实验 19 最短路径（迪杰斯特拉算法）

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握图的邻接矩阵存储结构。
- 实现图的最短路径查找操作（迪杰斯特拉算法）。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“019 最短路径（迪杰斯特拉算法）”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中首先初始化了图，然后调用 Dijkstra函数查找图中最短路径。

在 main 函数的后面，定义了图的最短路径查找函数 Dijkstra，该函数的函数体还不完整，留给读者完成。

ShortestPath.h文件

定义了与图相关的数据结构并声明了相关的操作函数和全局变量。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解查找最短路径函数的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

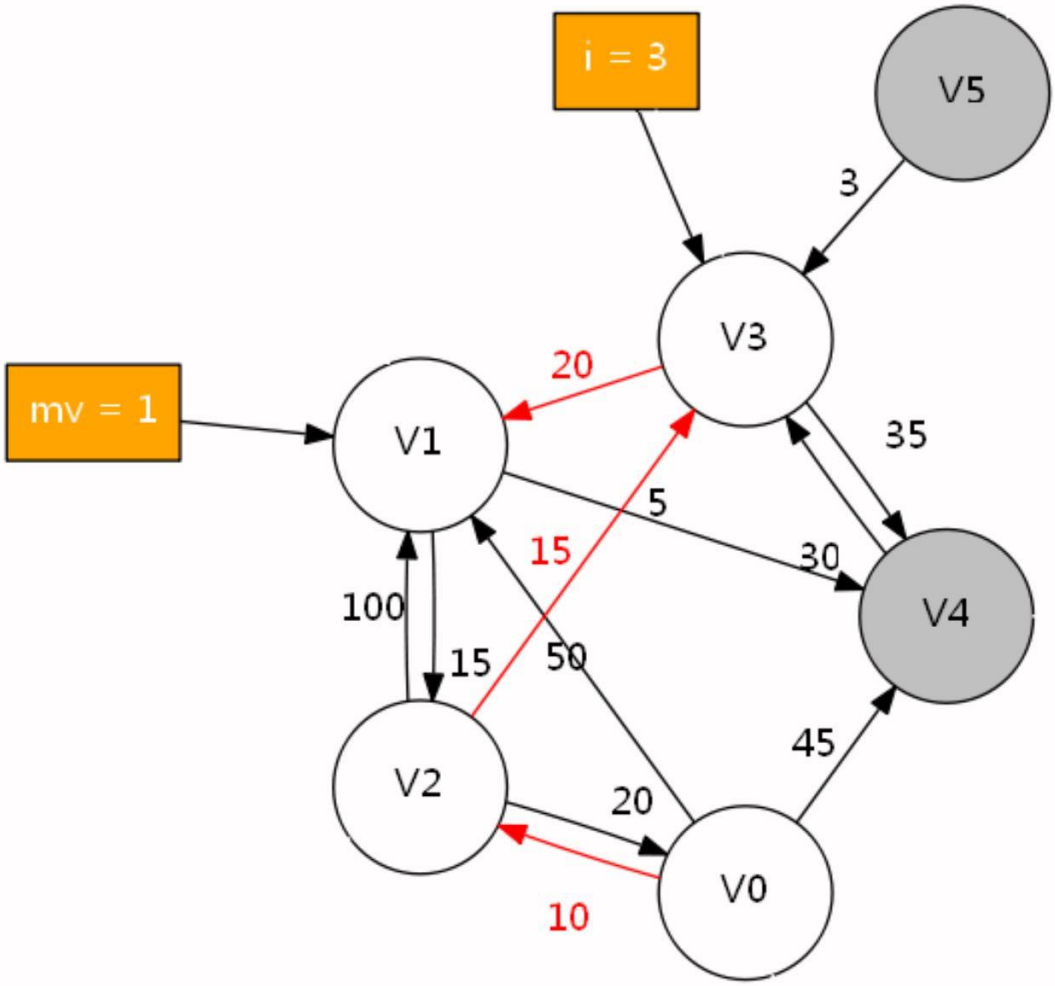
- 图的邻接矩阵。矩阵中的值表示顶点之间的距离。
- 在图中，未选出的顶点的填充色为灰色，已选出的顶点的填充色为白色；各个顶点之间的最短路径用红色的边进行连接。
- 在调试的过程中，可以看到游标的移动。

前驱顶点下标 mv = 1

最短路径长度 minw = 45

关系矩阵

	V0	V1	V2	V3	V4	V5
V0	1	50	10	∞	45	∞
V1	∞	1	15	∞	5	∞
V2	20	100	1	15	∞	∞
i→ V3	∞	20	∞	1	35	∞
V4	∞	∞	∞	30	0	∞
V5	∞	∞	∞	3	∞	0



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为Dijkstra函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 20 最短路径（弗洛伊德算法）

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握图的邻接矩阵存储结构。
- 实现图的最短路径查找操作（弗洛伊德算法）。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“020 最短路径（弗洛伊德算法）”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中首先初始化了图，然后调用 Floyd函数查找图中最短路径。

在 main函数的后面，定义了图的最短路径查找函数 Floyd，该函数的函数体还不完整，留给读者完成。

ShortestPath.h文件

定义了与图相关的数据结构并声明了相关的操作函数和全局变量。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解查找最短路径函数的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 游标 k。用来显示迭代计算最短路径的过程。
- 图的邻接矩阵。矩阵中的值表示顶点之间的距离。
- 存放每对顶点间最短路径长度的二维数组，以及 Vi 到 Vj 最短路径上 Vi 的后继顶点下标值的二维数组，如果对应的当前值与上一次不相同，用红色的字体显示。
- 在图中，分别显示了从某一个顶点（该顶点的边框是红色的双圆环）到各个顶点之间的最短路径，这些顶点之间用红色的边连接；未加入顶点的填充色为灰色，已加入顶点的填充色为白色。

关系矩阵

k→

	V0	V1	V2	V3	V4	V5
V0	0	50	10	∞	45	∞
V1	∞	0	15	∞	5	∞
V2	20	∞	0	15	∞	∞
V3	∞	20	∞	0	35	∞
V4	∞	∞	∞	30	0	∞
V5	∞	∞	∞	3	∞	0

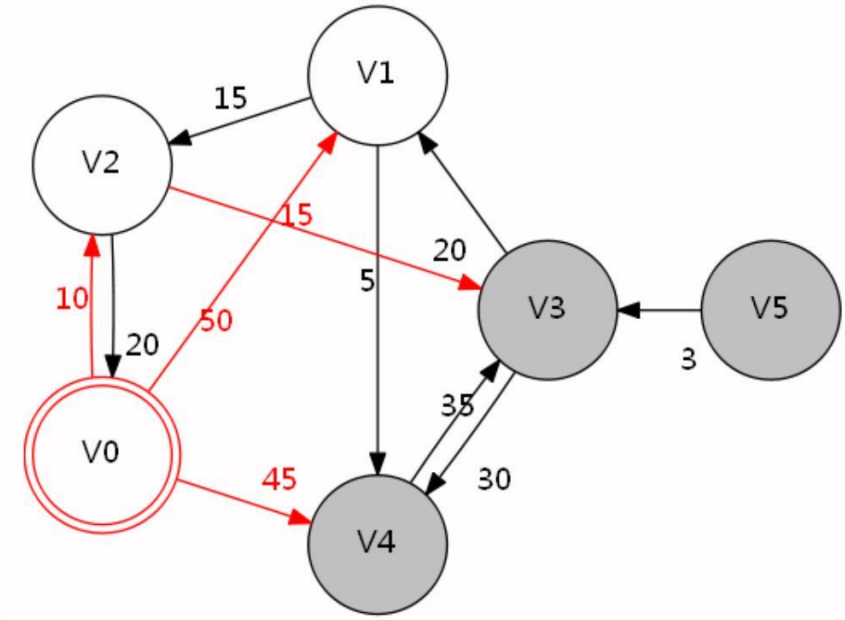
每对顶点间最短路径长度的数组

	V0	V1	V2	V3	V4	V5
V0	0	50	10	25	45	∞
V1	35	0	15	30	5	∞
V2	20	70	0	15	65	∞
V3	55	20	35	0	25	∞
V4	∞	∞	∞	30	0	∞
V5	∞	∞	∞	3	∞	0

后继顶点下标数组

	V0	V1	V2	V3	V4	V5
V0	0	1	2	2	4	-1
V1	2	1	2	2	4	-1
V2	0	0	2	3	0	-1
V3	1	1	1	3	1	-1
V4	-1	-1	-1	3	4	-1
V5	-1	-1	-1	3	-1	5

V0到各个顶点之间的最短路径



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为Floyd函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 21 折半查找

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现折半查找算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“021 折半查找”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中首先初始化了一个已经排序的线性表（注意，没有使用单元 0），然后调用了两次折半查找函数 BinarySearch 查找指定的关键字，第一次查找关键字成功，第二次查找关键字失败，最后销毁了线性表。

在 main函数的后面，分别定义了折半查找函数 BinarySearch，以及线性表的初始化函数和销毁函数。其中，折半查找函数 BinarySearch的函数体还不完整，留给读者完成。

SeqTable.h文件

定义了与线性表相关的数据结构并声明了相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

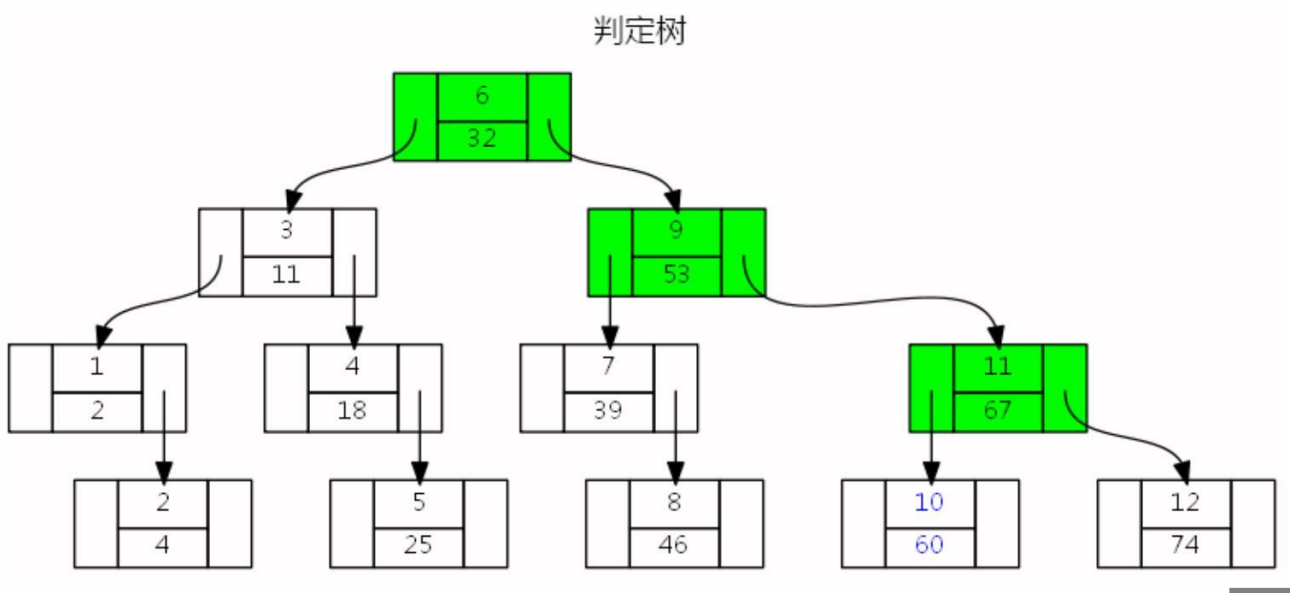
1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解折半查找的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 已经排序的线性表。
  a) 序号从 1 开始计数，要查找的关键字用蓝色的字体显示。
  b) 在调试过程中，可以看到 low、mid和 high三个游标的移动。
  c) 在查找的过程中，被命中的关键字所在的行的填充色为绿色。
- 根据线性表生成的判定树。
  a) 每个节点包括了元素的序号和关键字，并用蓝色的字体显示出要查找的关键字。
  b) 在查找的过程中，被命中的关键字所在的结点的填充色为绿色。

线性表:

序号	关键字	
1	2	
2	4	
3	11	
4	18	
5	25	
6	32	
7	39	
8	46	
9	53	
key→ 10	60	←low
11	67	←mid
12	74	←high



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 BinarySearch函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然

后回到步骤 1。

提示：在验证过程中，主要是检测 BinarySearch函数的返回值是否正确，所以，为了顺利通过验证，请读者一定要注意让 BinarySearch函数返回正确的值。

实验 22 二叉排序树的构造

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 掌握二叉排序树的链式存储结构。
- 实现二叉排序树的构造。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“022 二叉排序树的构造”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中首先定义了二叉排序树的指针，初始化了用于构造二叉排序树的关键码数组，然后调用二叉排序树的构造函数 CreateSearchTree构造二叉排序树，最后销毁了二叉排序树。

在 main 函数的后面，定义了二叉排序树的构造函数 CreateSearchTree，此函数的函数体还不完整，留给读者完成。

BinarySearchTree.h文件

定义了与二叉排序树相关的数据结构并声明了相关的操作函数。

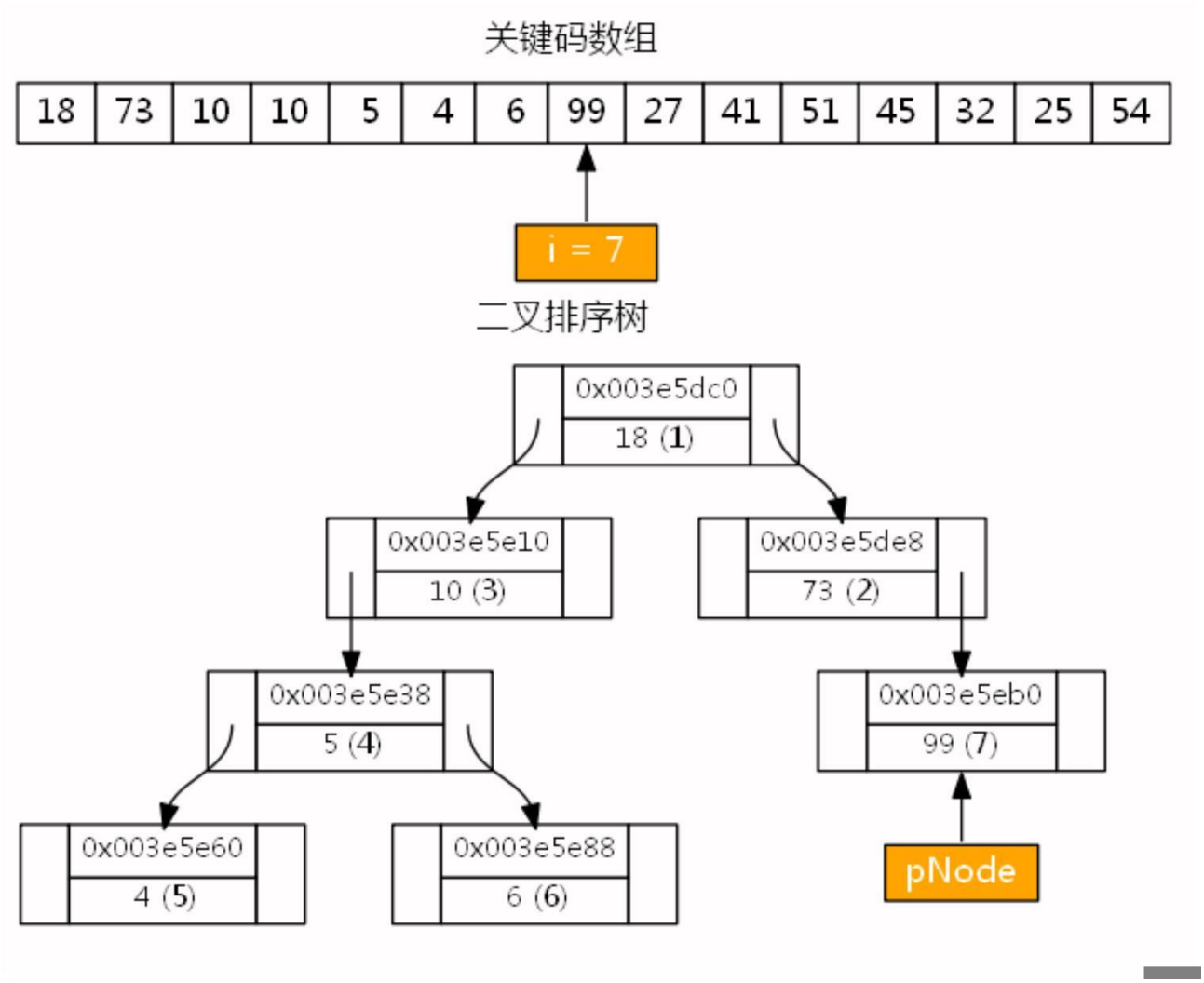
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解二叉排序树构造的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 用于构造二叉排序树的关键码数组和游标 i。
- 二叉排序树的详细信息。
  a) 包括了二叉排序树结点的值和地址，并且标出了结点在构建二叉排序树的序号。
  b) 在调试的过程中，游标指向了当前的节点。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为CreateSearchTree函数编写源代码，注意，尽量使用已定义的局部变量。
2. 按F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 23 创建哈希表（线性探测再散列）

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现哈希表的创建。
- 利用线性探测再散列处理冲突。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“023 创建哈希表（线性探测再散列）”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main.c 文件中，首先在 main 函数前定义了一个全局的哈希表结构体数组，然后在 main 函数中初始化了一个关键字数组并调用哈希表创建函数 Hash 用给定的关键字数组创建哈希表，最后在 Windows 控制台中打印输出哈希表。

在 main函数的后面，定义了哈希表创建函数 Hash，此函数的函数体还不完整，留给读者完成。

SearchHash.h文件

定义了与哈希表相关的数据结构并声明了相关的操作函数。

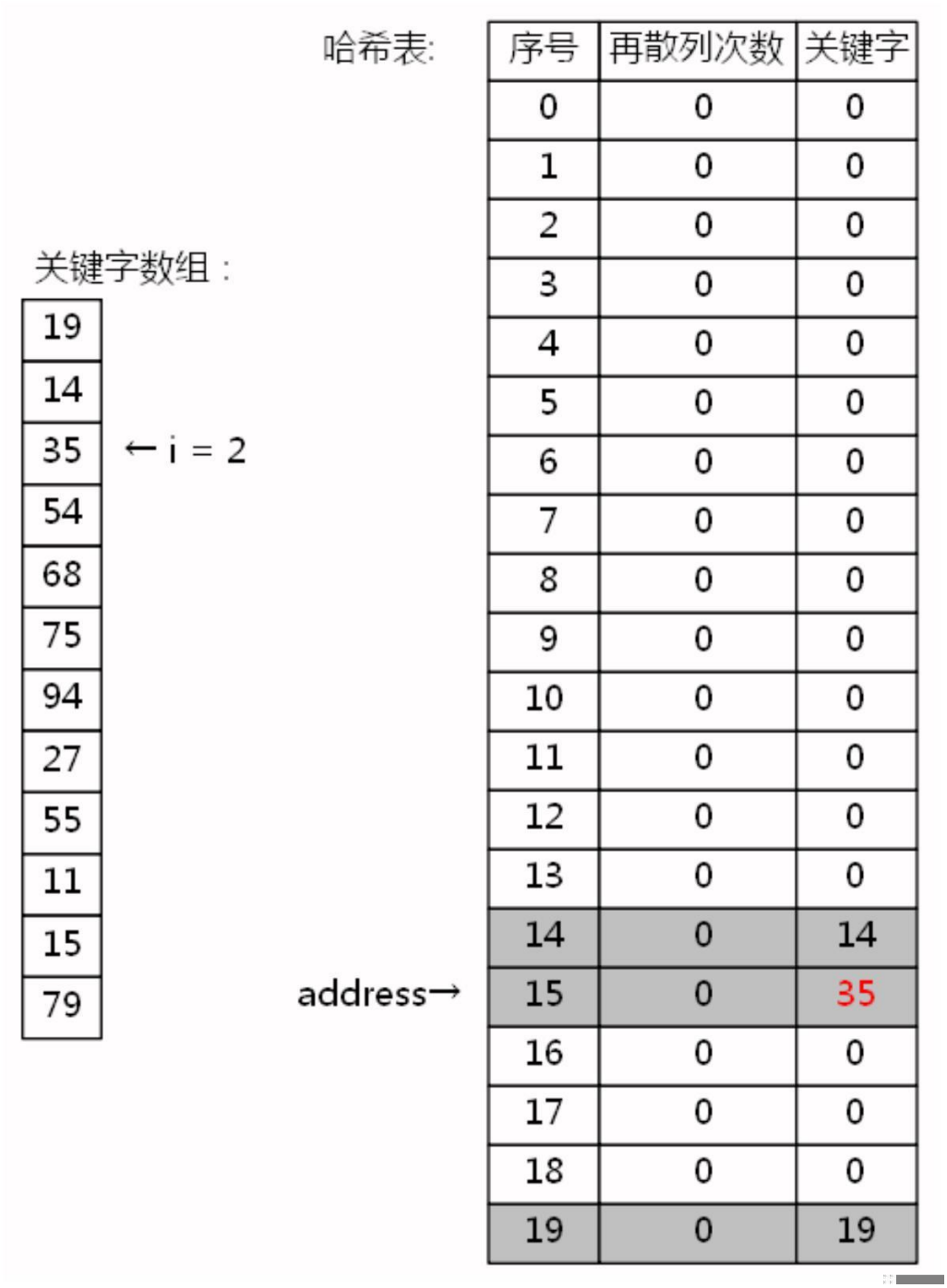
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解哈希表创建的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 用于构造哈希表的关键字数组和游标。
- 哈希表。对应的当前值与上一次值不相同，用红色的字体显示；已加入哈希表的关键字所在行的填充色为灰色。



序号	再散列次数	关键字
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	14
15	0	35
16	0	0
17	0	0
18	0	0
19	0	19

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为Hash函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 24 创建哈希表（二次探测再散列）

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现哈希表的创建。
- 利用二次探测再散列处理冲突。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“024 创建哈希表（二次探测再散列）”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main.c 文件中，首先在 main 函数前定义了一个全局的哈希表结构体数组，然后在 main 函数中初始化了一个关键字数组并调用哈希表创建函数 Hash 用给定的关键字数组创建哈希表，最后在 Windows 控制台中打印输出哈希表。

在 main函数的后面，定义了哈希表创建函数 Hash，此函数的函数体还不完整，留给读者完成。

SearchHash.h文件

定义了与哈希表相关的数据结构并声明了相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解哈希表创建的执行过程。“可视化数据”窗口显示的数据信息，包括：

- 用于构造哈希表的关键字数组和游标。
- 哈希表。可以参考实验 23中 2.3节。

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 Hash函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 25 创建哈希表（伪随机探测再散列）

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现哈希表的创建。
- 利用伪随机探测再散列处理冲突。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“025 创建哈希表（伪随机探测再散列）”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main.c文件中，首先在 main函数前定义了全局的哈希表结构体数组和伪随机数数组，然后在 main 函数中初始化了一个关键字数组并调用哈希表创建函数 Hash 用给定的关键字数组创建哈希表，最后在 Windows控制台中打印输出哈希表。

在 main函数的后面，定义了哈希表创建函数 Hash，此函数的函数体还不完整，留给读者完成。

SearchHash.h文件

定义了与哈希表相关的数据结构并声明了相关的操作函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解哈希表创建的执行过程。“可视化数据”窗口显示的数据信息，包括：

- 用于构造哈希表的关键字数组和游标。
- 用于处理冲突的伪随机数数组和游标。
- 哈希表。可以参考实验 23中 2.3节。

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 Hash函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

实验 26 冒泡排序

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现冒泡排序算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“026 冒泡排序”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先初始化了一个还未排序的线性表，然后调用冒泡排序函数 BubbleSort为线性表进行排序，最后在 Windows控制台中打印输出排序后的线性表。

在 main函数的后面，定义了冒泡排序函数 BubbleSort，此函数的函数体还不完整，留给读者完成。

SeqTable.h文件

声明了 BubbleSort函数。

2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解冒泡排序的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 线性表的长度 length。
- 每趟排序比较的次数 j。在未初始化时，显示为内存中的随机值。
- 排序完成标志 flag（0 表示排序未完成；1 表示排序已完成）。在未初始化时，显示为内存中的随机值。
- 线性表中元素的值。
  a) 可实时显示元素交换及冒泡排序的过程。
  b) 已排序的元素所在单元格的填充色为灰色，字体为白色。
  c) 对应的当前值与上一次值不相同，该元素用红色的字体显示。
  d) 在调试的过程中，可以看到游标的移动。
- 逐次记录每趟排序完成后线性表中元素的值，已排好序的元素所在单元格的填充色为灰色，字体为白色。

每趟结果

初始	12	21	77	65	38	7	38	53
1 趟	12	21	65	38	7	38	53	77
2 趟	12	21	38	7	38	53	65	77
3 趟	12	21	7	38	38	53	65	77
4 趟	12	7	21	38	38	53	65	77

2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 BubbleSort函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测 BubbleSort 函数执行完毕后，线性表中元素的顺序是否正确，所以，为了顺利通过验证，读者只需要完成对线性表的排序即可。

实验 27 快速排序

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现快速排序算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“027 快速排序”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main 函数中首先初始化了一个还未排序的线性表，然后调用快速排序函数 QuickSort 为线性表进行排序，最后在 Windows控制台中打印输出排序后的线性表。

在 main函数的后面，定义了快速排序函数 QuickSort函数， 此函数的函数体还不完整，留给读者完成。

SortObject.h文件

声明了 QuickSort函数和全局变量。注意，此头文件中还包含了栈模块的头文件 Stack.h。

Stack.h文件

定义了与栈相关的数据结构并声明了相关的操作函数。

Stack.c文件

定义了与栈相关的操作函数。

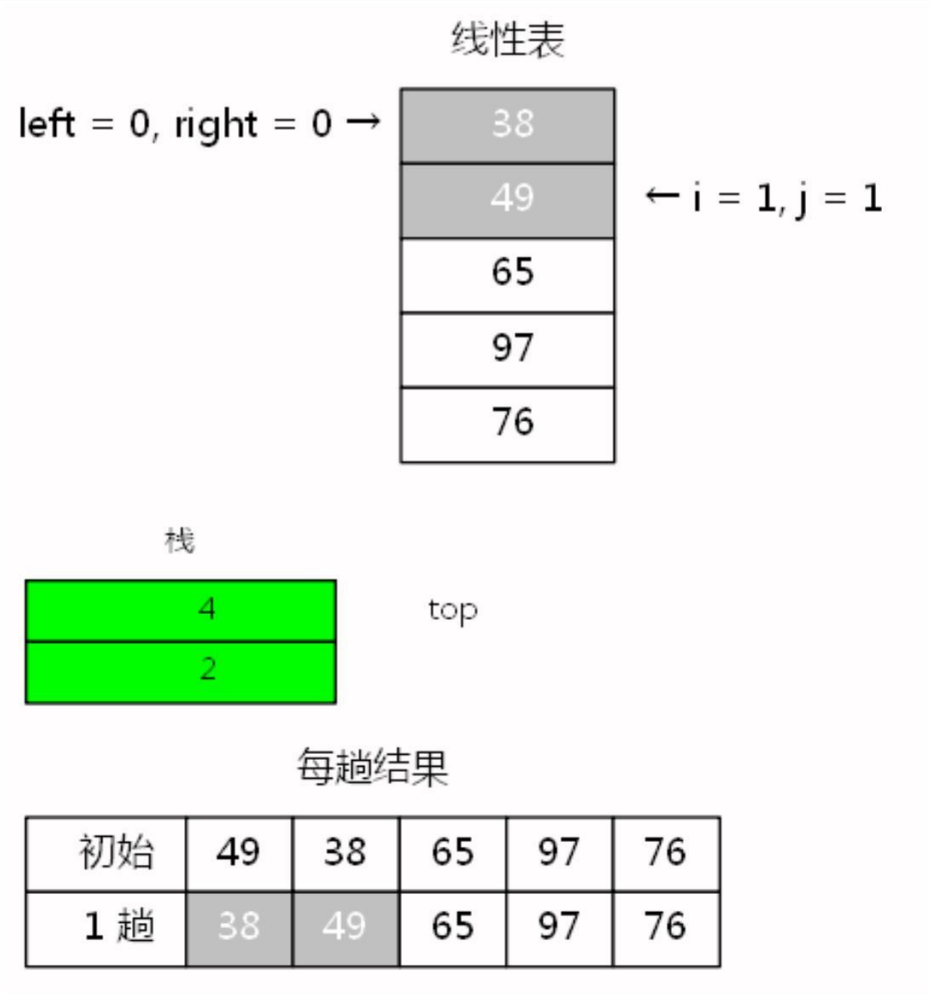
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解快速排序的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 分治范围的左右边界 left和 right。
- 指向分治范围两端的游标 i 和 j。
- 线性表中元素的值。可以实时显示元素之间交换及分区排序的过程。
- 栈中元素的值。显示已经入栈的分治范围。
- 根据每趟排序结果记录线性表的实时变化和分治信息。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 QuickSort函数编写源代码。注意，尽量不要使用递归算法，而是使用堆栈来保存未排序的分区范围，并尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

提示：在验证过程中，主要是检测 QuickSort函数执行完毕后，线性表中元素的顺序是否正确，所以，为了顺利通过验证，读者只需要完成对线性表的排序即可。

实验 28 堆排序

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现堆排序算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“028 堆排序”新建一个项目。

2.2 阅读实验源代码

main.c文件

在 main函数中首先初始化了一个还未排序的线性表，然后调用堆排序函数 HeapSort为线性表进行排序，最后在 Windows控制台中打印输出排序后的线性表。

在 main函数的后面，定义了堆排序函数 HeapSort和 Sift函数， 这两个函数的函数体还不完整，留给读者完成。

SortObject.h文件

定义了与线性表相关的数据结构并声明了相关的操作函数。

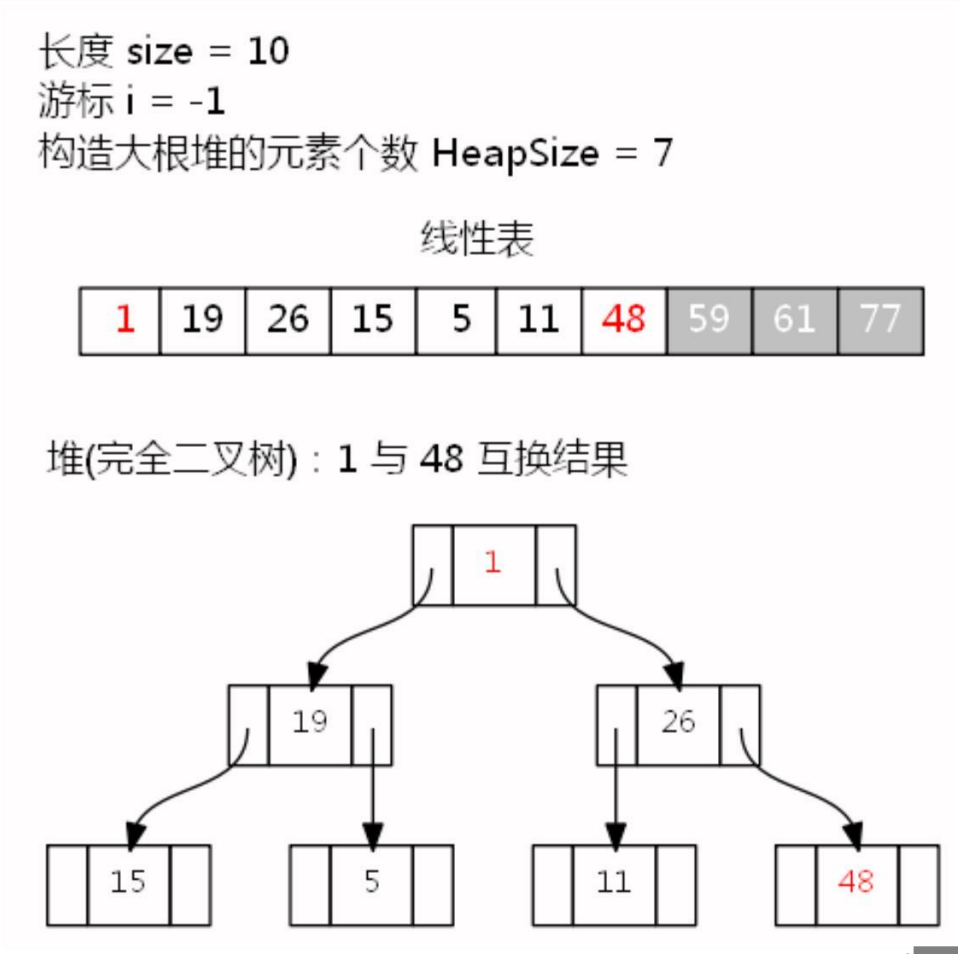
2.3 在演示模式下调试项目

按照下面的步骤调试项目：

1. 按 F7生成项目。
2. 在演示模式下，按 F5启动调试项目。程序会在观察点函数的开始位置中断。
3. 重复按 F5，直到调试过程结束。

在调试的过程中，每执行“演示流程”窗口中的一行后，仔细观察“可视化数据”窗口内容所发生的变化，理解堆排序的执行过程。“可视化数据”窗口显示的数据信息（如下图所示），包括：

- 线性表的长度 size。
- 游标 i 表示建堆元素的起始位置。
- 线性表中用于构造大根堆的元素的个数 HeapSize。
- 线性表中元素的值。可实时显示元素之间交换及堆排序的过程。
- 根据排序过程记录大根堆（完全二叉树）的实时变化和详细信息。



2.4编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 HeapSort函数和 Sift函数编写源代码。注意，尽量使用已定义的局部变量。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，或者在“非演示模式”下按 F5 启动调试后重复按 F10 单步调试读者编写的源代码，从而定位错误的位置，然后回到步骤 1。

第二部分 数据结构实验指导（C++语言）

实验 1 实验环境的使用

实验性质：验证
建议学时：1 学时

一、实验目的

- 熟悉数据结构集成实验环境 DS Lab的基本使用方法。
- 掌握线性表的顺序表示。
- 实现线性表的插入和删除操作。

二、实验内容

请读者按照下面的步骤完成实验内容，同时，仔细体会 DS Lab 的基本使用方法。在本实验题目中，操作步骤会编写的尽量详细，并会对 DS Lab 的核心功能进行具体说明。但是，在后面的实验题目中会尽量省略这些内容，而将重点放在实验相关的源代码上。如有必要，读者可以回到本实验题目中，参考 DS Lab 的基本使用方法。

2.1 启动 DS Lab

在安装有 DS Lab的计算机上，可以使用两种不同的方法来启动 DS Lab：

- 在桌面上双击“Engintime DS Lab”图标。
  或者
- 点击“开始”菜单，在“程序”中的“Engintime DS Lab”中选择“Engintime DS Lab”。

2.2 注册用户并登录

DS Lab每次启动后都会弹出一个“登录”对话框，可以进行以下操作：

- 使用已有用户进行登录
  读者可以在“登录”对话框中填写已有的学号、姓名、密码完成登录。登录成功后，DS Lab的标题栏会显示出读者用来登录的学号和姓名。
- 注册新用户
  读者可以点击“注册”按钮，在弹出的“注册”窗口中填写基本信息、所属机构、密码、密保问题完成注册，并自动登录。

2.3 主窗口布局

DS Lab的主窗口布局由下面的若干元素组成：

- 顶部的菜单栏、工具栏。
- 停靠在左侧和底部的各种工具窗口。
- 余下的区域用来放置“起始页”和“源代码编辑器”窗口。

提示：菜单栏、工具栏和各种工具窗口的位置可以随意拖动。如果想恢复窗口的默认布局，选择“窗口”菜单中的“重置窗口布局”即可。

2.4 新建实验项目

新建一个实验项目的步骤如下：

1. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
2. 在“新建项目”对话框中，选择项目模板“cpp001”。注意，其他模板会在后面的实验题目中使用。
3. 在“名称”中输入新项目使用的文件夹名称“lab1”。
4. 在“位置”中输入新项目保存在磁盘上的位置“C:\dslab”。
5. 点击“确定”按钮。

新建完毕后， DS Lab 会自动打开这个新建的项目。在“项目管理器”窗口中（如图 1-1 所示），根节点是项目节点，各个子节点是项目包含的文件夹或者文件。读者也可以使用“Windows 资源管理器”打开磁盘上的“C:\dslab\lab1”文件夹，查看项目中包含的源代码文件。

提示：右键点击“项目管理器”窗口中的项目节点，选择快捷菜单中的“打开所在的文件夹”，即可使用“Windows资源管理器”打开项目所在的文件夹。

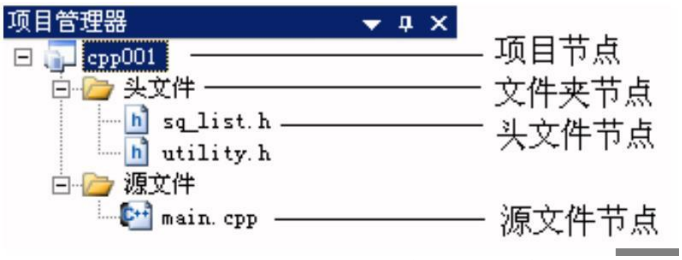


图 1-1：打开项目后的“项目管理器”窗口

2.5 绑定课时

绑定课时功能可以帮助读者将新建的项目与一个课时完成绑定，操作步骤如下：

1. 新建项目后，会自动弹出“绑定课时”对话框。
2. 在“绑定课时”对话框中，读者可以根据课时信息选择一个合适的课时完成绑定。
3. 绑定课时后，DS Lab的标题栏会显示当前项目所绑定的课时信息。
4. 读者可以登录开放实验管理平台，查看更详细的课时信息，平台使用方法请参见附录 2。

注意：如果不为项目绑定课时，将会影响到实验课考评成绩，并且不能提交作业。在未登录的情况下，不会弹出“绑定课时”对话框。

2.6 阅读实验源代码

该实验包含了两个头文件“sq_list.h”、“utility.h”和一个 CPP 源文件“main.cpp”。下面对这三个文件的主要内容、结构和作用进行说明：

main.cpp文件

在“项目管理器”窗口中双击“main.cpp”打开此文件。此文件主要包含了以下内容：

1. 在文件的开始位置，使用预处理命令“#include "utility.h" #include "sq_list.h""，包含了 utility.h文件和 sq_list.h文件。
2. 定义了 main函数。在其中实现了线性表的初始化，调用了若干次线性表的插入函数 Insert，再次调用了若干次线性表的删除函数 Delete,最后调用 Traverse 函数遍历线性表并打印出线性表的内容。

sq_list.h文件

在“项目管理器”窗口中双击“sq_list.h”打开此文件。此文件主要包含了以下内容：

1. 定义了线性表的类模板。
2. 实现了线性表类模板的部分方法，包括构造函数模板、析构函数模板、拷贝构造函数、赋值拷贝构造函数、求线性表的长度等，其中在线性表指定位置插入元素和在线性指定位置删除元素，这两个方法还没有实现，留给读者完成。

utility.h文件

在“项目管理器”窗口中双击“utility.h”打开此文件。此文件主要包含了以下内容：

1. 包含用到的 C++标准库头文件。目前包含了头文件“iostream”、“string”、“fstream”等。
2. 定义了实用函数，包括用于交换两个元素值的 Swap函数，用于显示数据的 Show函数等。
3. 定义了实用类，包括定时器类和随机数类。

提示：请读者认真理解这部分内容，其他实验题目中的源代码文件也严格遵守这些约定，如无特殊情况将不再进行如此详细的说明。

2.7 生成项目

使用“生成项目”功能可以将程序的源代码文件编译为可执行的二进制文件，操作如下：

1. 在“生成”菜单中选择“生成项目”（快捷键是 F7）。

在项目生成过程中，“输出”窗口会实时显示生成的进度和结果。如果源代码中不包含语法错误，会在生成的最后阶段提示生成成功，如图 1-2所示：

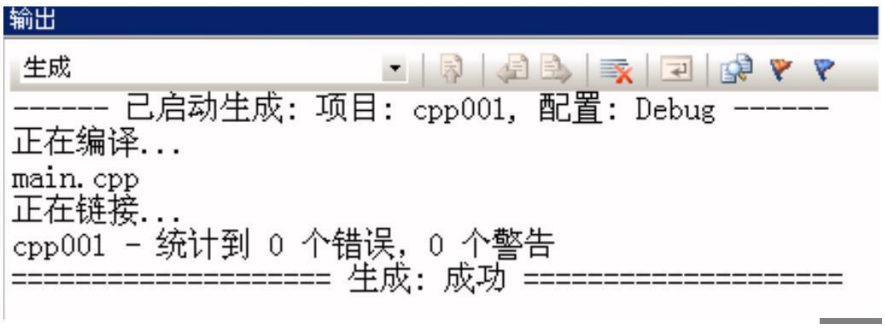


图 1-2：生成项目成功后的“输出”窗口

生成项目的过程，就是将项目所包含的每个 CPP 源代码文件（.cpp 文件）编译为一个对象文件（.o 文件），然后再将多个对象文件链接为一个目标文件（.exe 文件）的过程。以本实验为例，成功生成项目后，默认会在“C:\dslab\lab1\Debug″目录下生成 “main.o”文件和 “cpp001.exe”文件。

提示：读者可以通过修改项目名称的方法来修改生成的.exe文件的名称。方法是在“项目管理器”窗口中右键点击项目节点，选择快捷菜单中的“重命名”。待项目名称修改后，需要再次生成项目才能得到新的.exe文件。

2.8 解决语法错误

如果在源代码中存在语法错误，在生成项目的过程中，“输出”窗口会显示相应的错误信息（包括错误所在文件的路径，错误在文件中的位置，以及错误原因），并在生成的最后阶段提示生成失败。此时，在“输出”窗口中双击错误信息所在的行，DS Lab会使用源代码编辑器自动打开错误所在的文件，并定位到错误所在的代码行。

可以按照下面的步骤进行练习：

1. 在源代码文件中故意输入一些错误的代码（例如删除一个代码行结尾的分号）。
2. 生成项目。
3. 在“输出”窗口中双击错误信息来定位存在错误的代码行，并将代码修改正确。
4. 重复步骤 2、3，直到项目生成成功。

2.9 验证项目（失败）

这里介绍DS Lab提供的另外一个重要功能：验证功能。

之前提到了 sq_list.h文件中的 Insert函数和 Delete函数还不完整，是留给读者完成的。而当读者完成这两个函数后，往往需要使用调试功能、或者执行功能，来判断所完成的函数是否能够达到预期的效果。DS Lab提供的验证功能可以自动化的、精确的完成这个验证过程。

验证功能分为下面三个阶段：

1. 将源程序（软件自带的标准程序）产生的转储信息自动保存在文本文件 ValidateSource.txt中。
2. 将目标程序（即读者编写的程序）产生的转储信息自动保存在文本文件 ValidateTarget.txt中。
3. 自动使用 DS Lab 提供的文本文件比较工具来比较这两个文件。当这两个文件中的转储信息完全一致时，报告“验证成功”；否则，报告“验证失败”。

当读者完成的函数实现的功能与题目要求的功能完全一样，就会产生完全一致的输出信息，验证功能就会报告“验证成功”；否则，验证功能就会报告“验证失败”，并且允许读者使用 DS Lab 提供的文本文件比较工具，来查看这两个转储信息文件中的不同之处，从而帮助读者迅速、准确的找到验证失败的原因，进而继续修改源代码，直到验证成功。

按照下面的步骤启动验证功能：

3. 在“调试”菜单中选择“开始验证”（快捷键是 Alt+F5）。在验证过程中，“输出”窗口会显示验证结果信息。由于 Insert方法和 Delete方法还未实现，所以验证失败。
4. 使用“输出”窗口工具条上的“比较”按钮（如图 1-3所示）查看两个转储信息文件中的内容，即它们之间的不同之处。

图 1-3：“输出”窗口工具栏上的“比较”按钮。

2.10 实现 Insert函数

参考清单 1-1中的源代码，实现 Insert函数。

```
template <class ElemType>
bool SqList<ElemType>::Insert(int position, const ElemType &e)
{
    ElemType tmp;
    if (count == maxSize)
    {
        return false;
    }
    else if (position < 1 || position > Length() + 1)
    {
        return false;
    }
    else
    {
        count++;
        for (int pos = Length(); pos >= position; pos--)
        {
            GetElem(pos, tmp); SetElem(pos + 1, tmp);
        }

        SetElem(position, e);
        return true;
    }
}
```

清单 1-1：Insert函数的参考源代码。

2.11 实现 Delete函数

参考清单 1-2中的源代码，实现 Delete函数。

```
template <class ElemType>
bool SqList<ElemType>::Delete(int position, ElemType &e)
{
    ElemType tmp;

    if (position < 1 || position > Length())
    {
```

```
        return false;
    }
    else
    {   // position合法
        GetElem(position, e);
        for (int pos = position + 1; pos <= Length(); pos++)
        {
            GetElem(pos, tmp); SetElem(pos - 1, tmp);
        }
        count--;
        return true;
    }
}
```

清单 1-2：Delete函数的参考源代码。

2.12 调试项目

DS Lab提供的调试器是一个功能强大的工具，使用此调试器可以观察程序的运行时行为并确定逻辑错误的位置，可以中断程序的执行以检查代码，计算和编辑程序中的变量。为了顺利进行后续的各项实验，应该学会灵活使用这些调试功能。

调试 Insert函数：

1. 打开 main.cpp文件。
2. 在代码行 29行
   la.Insert(1, 68);
   上点击鼠标右键，在弹出的快捷菜单中选择“插入/删除断点”，会在此行左侧的空白处显示一个红色圆点，表明已经成功在此行代码上添加一个断点。
3. 在“调试”菜单中选择“启动调试”，可以看到刚刚添加断点的代码行左侧空白中显示一个黄色箭头，表示程序已经在此行代码处中断执行（注意，黄色箭头指向的代码行是下一个要执行的代码行，此行代码当前还没有执行）。
4. 在“调试”菜单中选择“逐语句”（快捷键是 F11），进入 Insert函数的内部。
5. 在“调试”菜单中选择“窗口”中的“调用堆栈”，在调用堆栈窗口显示出该函数的调用层次信息。
6. 可以按 F10 继续进行“逐过程”调试，也可以选择在“调试”菜单中选择“跳出”，就可以跳出Insert函数，在该线性表的指定位置插入了一个新元素。
7. 在“调试”菜单中选择“停止调试”，可以结束此次调试。

调试 Delete函数：

1. 在代码行 34行
   la.Delete(2, e);
   上点击鼠标右键，在弹出的快捷菜单中选择“插入/删除断点”。
2. 将鼠标移动到源代码编辑器中变量 e 的名称上，会弹出一个窗口显示出变量 e 的值（由于此时还没有给变量 e 赋值，所以是一个随机值）。
3. 在“调试”菜单中选择“逐过程”，会执行黄色箭头当前指向的代码行，并将黄色箭头指向下一个要执行的代码行。
4. 在源代码编辑器中变量 e 的名称上点击鼠标右键，在弹出的快捷菜单中选择“快速监视”，可以使用“快速监视”对话框查看 e 的值。可以点击“关闭”按钮关闭“快速监视”对话框。
5. 在源代码编辑器中变量 e 的名称上点击鼠标右键，在弹出的快捷菜单中选择“添加监视”，变量 e 就被添加到了“监视”窗口中。使用“监视”窗口可以随时查看变量的值和类型。可以看到“监

视”窗口中变量 e 的值已不同于随机值，这时 e 的值是 15。此时，删除了线性表中的元素 15。

6. 在“调试”菜单中选择“停止调试”，可以结束此次调试。

以上的练习说明，DS Lab可以让读者调试项目，从而理解每一行源代码对内存数据的操作结果。如果读者发现所编写的源代码存在异常行为（例如死循环、数组越界访问或者验证失败），可以单步调试项目，来查找异常产生的原因。

2.13 验证项目（成功）

按照下面的步骤启动验证功能：

1. 在“调试”菜单中选择“开始验证”。

如果验证成功，说明读者编写的代码符合实验题目的要求。

如果验证失败，读者可以参考之前的内容来查找原因并修改源代码中的错误，直到验证成功。

2.14 提交作业

如果读者写完了程序并通过了自动化验证，可以使用“提交作业”功能，将读者编写的源代码文件自动提交到服务器，供教师查看。步骤如下：

1. 选择“用户”菜单中的“提交作业”打开“提交作业”对话框。
2. 在“提交作业”对话框中，点击“继续提交”按钮，完成提交作业操作。
3. 如果需要重新绑定课时，点击“重新绑定课时”按钮，绑定符合要求的课时。
4. 提交作业成功后，读者可以登录开放实验管理平台，查看提交作业后的课时信息，平台使用方法参见附录 2。

注意：可以多次执行“提交作业”操作，后提交的作业会覆盖之前提交的作业。在未登录的情况下，不能提交作业。

2.15 总结

读者使用 DS Lab进行数据结构实验的步骤可以总结如下：

1. 启动 DS Lab。
2. 注册新用户，或使用已有用户登录。
3. 新建实验项目，并绑定课时。
4. 根据实验题目的要求，编写代码。
5. 生成项目（排除所有的语法错误）。
6. 使用自动化验证功能，验证读者编写的代码是否符合实验题目的要求。如果验证失败，可以使用“比较”功能，或者调试项目，从而定位错误的位置，修改程序直到正确为止。
7. 提交作业。
8. 退出 DS Lab。

2.16获得帮助

如果读者在使用 DS Lab的过程中遇到问题需要专业的解答，或者有一些心得体会想和其他 DS Lab用户分享，欢迎加入 DS Lab网上论坛：

- ↙ 选择 DS Lab“帮助”菜单中的“论坛”。

  或者

- ↙ 直接访问http://www.engintime.com/forum

这里列出了读者在使用 DS Lab 的过程中可能遇到的一些问题和使用技巧，用于帮助读者更好的使用 DS Lab，获得最佳的实验效果。

1. 读者时常会遇到在自己编写的源代码中存在死循环的情况，这就会造成 DS Lab 的调试功能，特别是验证功能无法自行结束。此时，读者可以选择“调试”菜单中的“停止调试”（快捷键是 Shift+F5）来强制结束这些功能。随后，读者可以检查自己编写的源代码，或者单步调试项目，从而找到造成死循环的原因。
2. 读者时常会遇到的另外一个情况是“数组越界访问”。此时，DS Lab会弹出一个调试异常对话框，读者只要选择对话框中的“是”按钮，就可以立即定位到异常所在的代码行。

3. DS Lab作为一个 IDE环境，提供了强大的调试功能，包括单步调试、添加断点、查看变量的值、查看调用堆栈等。读者在调试过程中可以灵活使用这些功能，提高调试效率。
4. 如果读者想快速查看程序在 Windows控制台窗口中打印输出的信息，可以使用“调试”菜单中的“开始执行”功能（快捷键是 Ctrl+F5）。
5. 为读者提供了查看用户信息、修改用户信息、修改密码、修改密保问题和找回密码等功能。

实验 2 简单线性链表的插入和删除操作

实验性质：验证＋设计
建议学时：1 学时

一、 实验目的

- 实现单链表的插入操作。
- 实现单链表的删除操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp002”新建一个项目。

2.2 阅读实验源代码

simple_lk_list.h文件

定义了简单线性链表类模板，并实现了部分功能。

node.h文件

定义了节点类模板。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试简单线性链表的插入、删除功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 simple_lk_list.h文件中的 Insert函数中添加代码，完成插入元素的功能。
2. 在 simple_lk_list.h文件中的 Delete函数中添加代码，完成删除元素的功能。
3. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
4. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
5. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 3 双向循环链表的插入和删除操作

实验性质：验证＋设计
建议学时：1 学时

一、 实验目的

- ↙ 实现双向循环链表的插入操作。
- ↙ 实现双向循环链表的删除操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp003”新建一个项目。

2.2 阅读实验源代码

dbl_lk_list.h文件

定义了双向循环链表模板，并实现了部分功能。

dbl_node.h文件

定义了双向循环链表节点模板。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试双向循环链表的插入、删除功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 dbl_lk_list.h文件中的 Insert函数中添加代码，完成插入元素的功能。
2. 在 dbl_lk_list.h文件中的 Delete函数中添加代码，完成删除元素的功能。
3. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
4. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
5. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 4 实现字符串类的常用操作

实验性质：验证＋设计
建议学时：1 学时

一、 实验目的

- 实现连接字符串的操作。
- 实现字符串拷贝的操作。
- 实现连字符串查找的操作。
- 实现求子字符串的操作。
- 实现重载下标运算符的操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp004”新建一个项目。

2.2 阅读实验源代码

string.h文件

定义了串类，并实现了串的部分相关操作，例如：连接字符串、拷贝字符串等。

lk_list.h文件

定义了线性链表类模板，并实现了相关功能。

node.h文件

定义了节点类模板。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试串类的连接、拷贝、查找等操作。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 string.h文件中的 Concat函数中添加代码，完成连接字符串的功能。
2. 在 string.h文件中的 Copy函数中添加代码，完成字符串拷贝的功能。
3. 在 string.h文件中的 Index函数中添加代码，完成字符串查找的功能。
4. 在 string.h文件中的 SubString函数中添加代码，完成求子字符串的功能。
5. 在 string.h文件中的 Concat函数中添加代码，完成重载下标运算符的功能。
6. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
7. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
8. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 5 先序线索二叉树及先序遍历

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现先序遍历线索二叉树的操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp005”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了节点类模板。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

bin_tree_node.h文件

定义了二叉树节点类模板。

binary_tree.h文件

定义了二叉树类模板，并实现了相关功能。

thread_bin_tree_node.h文件

定义了线索二叉树结点类模板。

pre_thread_binary_tree.h文件

定义了先序线索二叉树类模板，并实现了部分功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试先序遍历线索二叉树的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在文件 pre_thread_binary_tree.h中的 PreOrder函数中添加代码，完成先序遍历线索二叉树的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 6 中序线索二叉树及中序遍历

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现中序线索化二叉树的操作。
- 实现中序遍历线索二叉树的操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp006”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了节点类模板。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

bin_tree_node.h文件

定义了二叉树节点类模板。

binary_tree.h文件

定义了二叉树类模板，并实现了相关功能。

thread_bin_tree_node.h文件

定义了线索二叉树结点类模板。

in_thread_binary_tree.h文件

定义了中序线索二叉树类模板，并实现了部分功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试中序遍历线索二叉树的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在文件 in_thread_binary_tree.h 中的 InThreadHelp 函数中添加代码，完成中序线索化二叉树的功能。
2. 在文件 in_thread_binary_tree.h中的 InOrder函数中添加代码，完成中序遍历线索二叉树的功能。
3. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
4. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。

5. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 7 后序线索二叉树及后序遍历

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现后序线索化二叉树的操作。
- 实现后续遍历线索二叉树的操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp007”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了节点类模板。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

tri_lk_bin_tree_node.h文件

定义了三叉链表二叉树结点类模板。

tri_lk_binary_tree.h文件

定义了三叉链表二叉树类模板，并实现了相关功能。

post_thread_bin_tree_node.h文件

定义了线索二叉树结点类模板。

post_thread_binary_tree.h文件

定义了后序线索二叉树类模板，并实现了部分功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试后序遍历线索化二叉树的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在文件 post_thread_binary_tree.h 中的 PostThreadHelp 函数中添加代码，完成后序线索化二叉树的功能。
2. 在文件 post_thread_binary_tree.h中的 PostOrder函数中添加代码，完成后序遍历线索二叉树的功能。
3. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
4. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。

5. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 8 哈夫曼树

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现构造哈夫曼树的操作。
- 实现对字符序列进行编码的操作。
- 实现对编码串进行译码的操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp008”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了节点类模板。

lk_list.h文件

定义了线性链表类模板，并实现了相关功能。

string.h文件

定义了串类，并实现了相关功能。

huffman_tree_node.h文件

定义了哈夫曼树结点类模板。

huffman_tree.h文件

定义了哈夫曼树类模板，并实现了部分功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试哈夫曼树的构造、编码和译码的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 huffman_tree.h文件中的 CreatHuffmanTree函数中添加代码，完成构造哈夫曼树的功能。
2. 在 huffman_tree.h文件中的 Encode函数中添加代码，完成对字符序列进行编码的功能。
3. 在 huffman_tree.h文件中的 Decode函数中添加代码，完成对编码串进行译码的功能。
4. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
5. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
6. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 9 图的深度优先搜索

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现图的深度优先搜索。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp009”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了节点类模板。

lk_list.h文件

定义了线性链表类模板，并实现了相关功能。

adj_list_graph_vex_node.h文件

定义了邻接表图顶点结点类模板。

adj_list_dir_graph.h文件

定义了有向图的邻接表类模板，并实现了部分功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试图的深度优先搜索的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在文件 adj_list_dir_graph.h 中的 DFSTraverse 函数中添加代码，请使用递归方式完成图的深度优先搜索的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 10 图的广度优先搜索

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现图的广度优先搜索。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp010”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了节点类模板。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

lk_list.h文件

定义了线性链表类模板，并实现了相关功能。

adj_list_graph_vex_node.h文件

定义了邻接表图顶点结点类模板。

adj_list_dir_graph.h文件

定义了有向图的邻接表类模板，并实现了部分功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试图的广度优先搜索的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在文件 adj_list_dir_graph.h 中的 BFSTraverse 函数中添加代码，请使用队列的方式(在lk_queue.h文件中实现了队列的相关操作)，完成图的广度优先搜索的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 11 拓扑排序

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现图的拓扑排序。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp011”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了结点类模板。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

adj_matrix_dir_graph.h文件

定义了有向图的邻接矩阵类模板，并实现了相关功能。

top_sort.h文件

定义了实现拓扑排序功能的函数，该功能的实现留给读者完成。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试拓扑排序的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 top_sort.h文件中的 TopSort函数中添加代码，完成拓扑排序的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 12 关键路径

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现网的关键路径查找操作。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp012”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了结点类模板。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

lk_stack.h文件

定义了链栈类模板，并实现了相关功能。

adj_matrix_dir_network.h文件

定义了有向网的邻接矩阵类模板，并实现了相关功能。

critical_path.h文件

定义了查找网的关键路径 CriticalPath函数，留给读者实现该功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试网的关键路径查找功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 critical_path.h文件中的 CriticalPath函数中添加代码，完成网的关键路径查找的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 13 最短路径(迪杰斯特拉算法)

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现网的最短路径查找操作（迪杰斯特拉算法）

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp013”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了结点类模板。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

lk_stack.h文件

定义了链栈类模板，并实现了相关功能。

adj_matrix_dir_network.h文件

定义了有向网的邻接矩阵类模板，并实现了相关功能。

shortest_path_dij.h文件

定义了使用迪杰斯特拉算法实现网的最短路径查找功能的函数，该功能的实现留给读者完成。

ass.h文件

定义了显示最短路径功能的函数，并实现了该功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

使用迪杰斯特拉算法,测试网的最短路径查找功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 shortest_path_dij.h文件中的 ShortestPathDIJ函数中添加代码，使用迪杰斯特拉算法，完成网的最短路径查找功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 14 最短路径(弗洛伊德算法)

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现网的最短路径查找操作（弗洛伊德算法）。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp014”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了结点类模板。

lk_list.h文件

定义了线性链表类模板，并实现了相关功能。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

lk_stack.h文件

定义了链栈类模板，并实现了相关功能。

adj_list_network_edge.h文件

定义了邻接表网边数据类模板。

adj_list_network_vex_node.h文件

定义了邻接表网顶点结点类模板。

adj_list_dir_network.h文件

定义了有向网的邻接表类模板，并实现了相关功能。

shortest_path_floyd.h文件

定义了使用迪杰斯特拉算法实现网的最短路径查找功能的函数，该功能的实现留给读者完成。

ass.h文件

定义了显示最短路径功能的函数，并实现了该功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

使用弗洛伊德算法，测试网的最短路径查找功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 shortest_path_floyd.h文件中的 ShortestPathFloyd函数中添加代码，使用弗洛伊德算法，完成网的最短路径查找功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。

3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 15 折半查找

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现折半查找算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp015”新建一个项目。

2.2 阅读实验源代码

bin_search.h文件

定义了实现折半查找算法的函数，该功能的实现留给读者完成。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试折半查找算法的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 bin_search.h文件中的 BinSerach函数中添加代码，完成折半查找算法的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。

实验 16 二叉排序树

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现二叉排序树的构造。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp016”新建一个项目。

2.2 阅读实验源代码

node.h文件

定义了结点类模板。

lk_queue.h文件

定义了链队列类模板，并实现了相关功能。

bin_tree_node.h文件

定义了二叉树结点类模板。

binary_sort_tree.h文件

定义了二叉排序树类模板，并实现了部分功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试二叉排序树的构造功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 binary_sort_tree.h文件中的 Insert函数中添加代码，完成在二叉排序树中插入节点的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 17 使用除留余数法构造散列表

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现使用除留余数法构造散列表。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp017”新建一个项目。

2.2 阅读实验源代码

hash_table.h文件

定义了散列表类模板，并实现了部分功能。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试构造散列表的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 hash_table.h文件中的 Insert函数中添加代码，使用除留余数法构造散列表，并完成在散列表中插入元素的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

实验 18 起泡排序算法

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现起泡排序算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp018”新建一个项目。

2.2 阅读实验源代码

bubble_sort.h文件

定义了起泡排序算法的函数。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试起泡排序算法的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 bubble_sort.h文件中的 BubbleSort函数中添加代码，完成起泡排序算法的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。

实验 19 快速排序算法

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现快速排序算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp019”新建一个项目。

2.2 阅读实验源代码

quick_sort.h文件

定义了快速排序算法的函数。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试快速排序算法的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 bubble_sort.h文件中的 QuickSort函数中添加代码，使用快速排序算法，完成排序的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。

实验 20 堆排序算法

实验性质：验证+设计
建议学时：1 学时

一、 实验目的

- 实现堆排序算法。

二、 实验内容

2.1 准备实验

按照下面的步骤准备实验：

1. 启动 DS Lab。
2. 在“文件”菜单中选择“新建”，然后单击“项目”，打开“新建项目”对话框。
3. 使用模板“cpp020”新建一个项目。

2.2 阅读实验源代码

heap_sort.h文件

定义了堆排序算法的函数。

utility.h文件

定义了一些实用类和实用函数。

main.cpp文件

测试堆排序算法的功能。

2.3编写源代码并通过验证

按照下面的步骤继续实验：

1. 在 heap_sort.h文件中的 HeapSort函数中添加代码，使用堆排序算法，完成排序的功能。
2. 按 F7生成项目。如果生成失败，根据“输出”窗口中的提示信息修改源代码中的语法错误。
3. 按 Alt+F5 启动验证。如果验证失败，可以使用“输出”窗口中的“比较”功能，从而定位错误的位置， 并修改程序中的错误，直到自动化验证通过为止。注意：不要修改 main函数，否则自动化验证会失败。
4. 继续完成其它未实现的函数，并在 main函数中调用这些函数进行测试。

附录 1 使用云模板

读者除了可以使用本地实验模板新建项目，还可以使用云模板新建项目。使用云模板新建项目后的实验过程和本地模板是完全一样的。可以按照下面的步骤使用云模板新建项目，并完成实验：

1. 启动配套实验软件

在安装有配套实验软件的计算机上，可以使用两种不同的方法来启动配套实验软件：

- 在桌面上双击配套实验软件图标。

  或者

- 点击“开始”菜单，在“程序”中选择配套实验软件。

2. 注册用户并登录

配套实验软件每次启动后都会弹出一个“登录”对话框，可以进行以下操作：

- 使用已有用户进行登录

  读者可以在“登录”对话框中填写已有的学号、姓名、密码完成登录。登录成功后，配套实验软件的标题栏会显示出读者用来登录的学号和姓名。

- 注册新用户

  读者可以点击“注册”按钮，在弹出的“注册”窗口中填写基本信息、所属机构、密码、密保问题完成注册，并自动登录。

3. 主窗口布局

配套实验软件的主窗口布局由下面的若干元素组成：

- 顶部的菜单栏、工具栏。
- 停靠在左侧和底部的各种工具窗口。
- 余下的区域用来放置“起始页”和“源代码编辑器”窗口。

提示：菜单栏、工具栏和各种工具窗口的位置可以随意拖动。如果想恢复窗口的默认布局，选择“窗口”菜单中的“重置窗口布局”即可。

4. 从云模板新建实验项目

从云模板新建一个实验项目的步骤如下：

1) 选择“文件”菜单中的“新建”中的“从云模板新建项目”，打开“使用云模板新建项目”对话框，如图 1 所示。

注：如果云模板列表为空，或者未能获取到最新的云模板，可以点击“重新获取云模板”按钮，重新获取云模板，获取完成后，会在云模板列表中显示出所有的实验模板。

图 **1**　使用云模板新建项目对话框

2) 在“使用云模板新建项目”对话框中，首先选择模板作者，例如可以选择作者为“北京英真”，这时在“云模板列表”中，会显示出该作者创建的所有云模板。

3) 在“使用云模板新建项目”对话框中，选择“基本的输入输出”的云模板，此时会显示出该模板的相关信息。

模板的相关信息： 运行时间限制：500ms　　内存限制（字节）：32字节　　下载次数：0　　可验证：是 模板描述： 练习基本的输入输出操作

注：也可以输入模板名称，点击“搜索模板名称”按钮，会自动选中第一个符合要求的模板。

4) 在“名称”中输入新项目使用的文件夹名称“cloudlab1”。
5) 在“位置”中输入新项目保存在磁盘上的位置“D:\CloudTemplatePractice”。
6) 点击“确定”按钮，此时开始从服务器下载已选中的云模板，下载完成并新建项目后，配套实验软件会自动打开这个新建的项目。在“项目管理器”窗口中（如图 2 所示），根节点是项目节点，各个子节点是项目包含的文件夹或者文件。读者也可以使用“Windows 资源管理器”打开磁盘上的“D:\CloudTemplatePractice\cloudlab1”文件夹，查看项目中包含的源代码文件。

提示：右键点击“项目管理器”窗口中的项目节点，选择快捷菜单中的“打开所在的文件夹”，即可使用“Windows资源管理器”打开项目所在的文件夹。

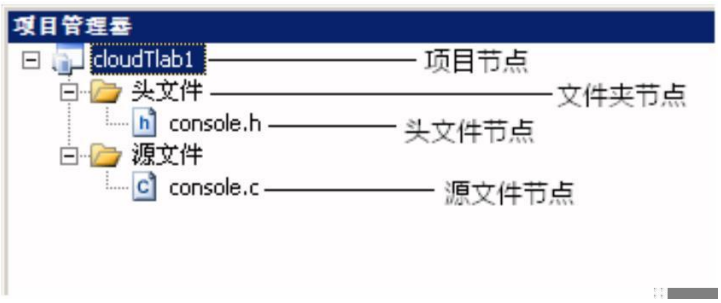


图 2：打开项目后的“项目管理器”窗口

5. 绑定课时

绑定课时功能可以帮助读者将新建的项目与一个课时完成绑定，操作步骤如下：

1. 从云模板新建新建项目后，会自动弹出“绑定课时”对话框。
2. 在“绑定课时”对话框中，读者可以根据课时信息选择一个合适的课时完成绑定。
3. 绑定课时后，配套实验软件的标题栏会显示当前项目所绑定的课时信息。
4. 读者可以登录开放实验管理平台，查看更详细的课时信息，平台使用方法请参见附录 2。

注意：如果不为项目绑定课时，将会影响到实验课考评成绩，并且不能提交作业。在未登录的情况下，不会弹出“绑定课时”对话框。

6. 生成项目

使用“生成项目”功能可以将程序的源代码文件编译为可执行的二进制文件，操作如下：

1) 在“生成”菜单中选择“生成项目”（快捷键是 F7）。

在项目生成过程中，“输出”窗口会实时显示生成的进度和结果。如果源代码中不包含语法错误，会在生成的最后阶段提示生成成功，如图 3 所示：

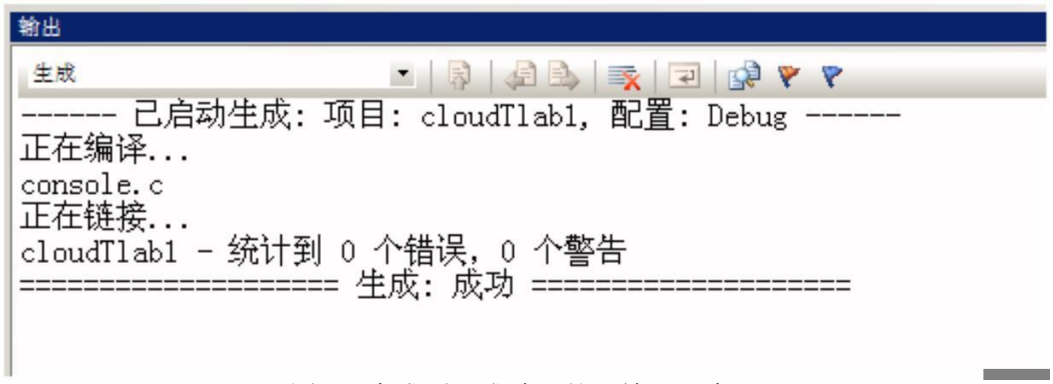


图 3：生成项目成功后的“输出”窗口

生成项目的过程，就是将项目所包含的每个 C 源代码文件（.c文件）编译为一个对象文件（.o文件），

然后再将多个对象文件链接为一个目标文件（.exe文件）的过程。以本实验为例，成功生成项目后，默认会在“D:\CloudTemplatePractice\cloudlab1\Debug”目录下生成“console.o”文件和“cloudlab1.exe”文件。

提示：读者可以通过修改项目名称的方法来修改生成的.exe文件的名称。方法是在“项目管理器”窗口中右键点击项目节点，选择快捷菜单中的“重命名”。待项目名称修改后，需要再次生成项目才能得到新的.exe文件。

7. 解决语法错误

如果在源代码中存在语法错误，在生成项目的过程中，“输出”窗口会显示相应的错误信息（包括错误所在文件的路径，错误在文件中的位置，以及错误原因），并在生成的最后阶段提示生成失败。此时，在“输出”窗口中双击错误信息所在的行，配套实验软件会使用源代码编辑器自动打开错误所在的文件，并定位到错误所在的代码行。

可以按照下面的步骤进行练习：

1) 在源代码文件中故意输入一些错误的代码（例如删除一个代码行结尾的分号）。
2) 生成项目。
3) 在“输出”窗口中双击错误信息来定位存在错误的代码行，并将代码修改正确。
4) 重复步骤 2、3，直到项目生成成功。

8. 验证项目（失败）

这里介绍配套实验软件提供的一个重要功能：验证功能。

console.c文件中的 main函数还不完整，是留给读者完成的。而当读者完成此函数后，往往需要使用调试功能、或者执行功能，来判断所完成的函数是否能够达到预期的效果。配套实验软件提供的验证功能可以自动化的、精确的完成这个验证过程。

验证的过程中，会将用于验证的可执行程序的结果自动保存在文本文件 ValidateSource.txt中，将用户的可执行程序的结果自动保存在文本文件 ValidateTarget.txt中，并自动使用配套实验软件提供的文本文件比较工具来比较这两个文件。当这两个文件中的内容完全一致时，报告“验证成功”；否则，报告“验证失败”。

如果“验证失败”，读者可以使用配套实验软件提供的文本文件比较工具，来查看这两个文件内容的不同之处，从而帮助读者迅速、准确的找到验证失败的原因，进而继续修改源代码，直到验证成功。

按照下面的步骤启动验证功能：

1) 在“调试”菜单中选择“开始验证”（快捷键是 Alt+F5）。在验证过程中，“输出”窗口会实时显示验证的相关信息（如清单 1 所示）。由于 main函数还不完整，所以验证失败。
2) 使用“输出”窗口工具条上的“比较”按钮（如图 4 所示）查看两个文件内容的不同之处。

```
用户程序运行时间：47ms
运行时间限制：500ms

内存峰值：0 字节
内存限制：32字节
内存泄漏：0 字节

==================== 验证结果：失败 ====================

提示：可以使用“输出”窗口工具条上的“比较”按钮查看目标输出文件与
      与标准输出文件的不同之处，从而准确定位导致验证失败的原因。
```

清单 1：在“输出”窗口中显示的验证信息。

图 4：“输出”窗口工具栏上的“比较”按钮。

9. 实现 main函数

参考清单 2 中的源代码，实现 main函数。

```
int main(int argc, char* argv[])
{
    int a = 0, b = 0;
    scanf("%d%d", &a, &b);
    int sum = 0;
    sum = a + b;
    printf("%d\n", sum);

    return 0;
}
```

清单 2：main函数的参考源代码。

10. 调试项目

读者在实现了 main函数后，可以按照下面的步骤，练习调试项目：

1) 在“生成”菜单中选择“生成项目”。如果读者编写的源代码中存在语法错误，修改这些错误，直到可以成功生成项目。
2) 在 main函数中的代码行 int a = 0, b = 0;上点击鼠标右键，在弹出的快捷菜单中选择“插入/删除断点”，会在此行左侧的空白处显示一个红色圆点，表明已经成功在此行代码上添加一个断点。
3) 在“调试”菜单中选择“启动调试”，windows控制台应用程序开始执行，并在刚刚添加断点的代码行左侧空白中显示一个黄色箭头，表示程序已经在此行代码处中断执行（注意，黄色箭头指向的代码行是下一个要执行的代码行，此行代码当前还没有执行）。
4) 在“调试”菜单中选择“逐过程”（快捷键是 F10），会执行黄色箭头当前指向的代码行，并将黄色箭头指向下一个要执行的代码行。
5) 继续使用“逐过程”单步调试源代码，体会此功能的作用。
6) 在“调试”菜单中选择“停止调试”，结束此次调试。

提示：如果读者发现所编写的源代码存在异常行为（例如死循环、数组越界访问或者验证失败），可以单步调试项目，来查找异常产生的原因。

11. 验证项目（成功）

按照下面的步骤启动验证功能：

在“调试”菜单中选择“开始验证”。

如果验证失败，读者可以参考之前的内容来查找原因并修改源代码中的错误，直到验证成功。

12. 提交作业

如果读者写完了程序并通过了自动化验证，可以使用“提交作业”功能，将读者编写的源代码文件自动提交到服务器，供教师查看。步骤如下：

1) 选择“用户”菜单中的“提交作业”打开“提交作业”对话框。
2) 在“提交作业”对话框中，点击“继续提交”按钮，完成提交作业操作。
3) 如果需要重新绑定课时，点击“重新绑定课时”按钮，绑定符合要求的课时。
4) 提交作业成功后，读者可以登录开放实验管理平台，查看提交作业后的课时信息，平台使用方法参见附录 2。

注意：可以多次执行“提交作业”操作，后提交的作业会覆盖之前提交的作业。在未登录的情况下，不能提交作业。

附录 2 使用开放实验管理平台

1. 登录开放实验管理平台。

使用浏览器输入开放实验管理平台的地址，打开登录页面，切换到学生登录选项卡，输入学号和密码，点击“登录”按钮，完成登录。

2. 查看课时信息

1) 选择顶部菜单“课程设置”中的“课时”菜单项，打开“课时列表”页面，在该页面中列出了所有课时信息，可以通过“开课年度”和“所属课程”进行筛选。

2) 在“课时列表”页面中，点击某一课时名称，进入展示课时页面，显示了该课时的详细信息。其中，在实验模板这一项中，包含了本次实验课中需要用到的相关实验模板的详细信息，读者可以根据这些模板信息，使用相应的模板完成实验。

3. 查看已提交的作业

1） 读者可以在“课时列表”页面中，找到需要查看的课时信息，然后在“操作”这一列点击“课时考评”按钮，进入“课时考评”页面。

2） 在“课时考评”页面的“已提交作业”这一列中，可以很方便地查看已提交的作业项。点击某一个作业项，进入“项目信息”所在的页面，可以查看已提交项目的详细信息，在代码查看器中可以查看项目所包含文件的源代码，还可以点击“下载选中项目”按钮，将该项目下载到本地。

4. 修改学生信息和密码

点击右上角的学号，在弹出的菜单中，选择“用户信息”菜单项，可以修改学生信息，选择“修改密码”菜单项，可以修改密码。

参考文献

[1] 张乃孝编著．算法与数据结构——C 语言描述（第 3 版）．北京：高等教育出版社，2011

[2] 严蔚敏编著．数据结构（C 语言版）．北京：清华大学出版社，2007

[3] 唐宁九，游洪跃等编著．数据结构与算法教程（C++版）实验和课程设计．北京：清华大学出版社，2012