

C 语言实验上机指导

周凯波 周纯杰 何顶新 高常鑫 左峥嵘 彭刚 陈忠

华中科技大学自动化学院

2017 年 3 月 2 日

目 录

第一部分 环境配置.....	1
1 VC++6.0 集成开发环境.....	1
1.1 VC++6.0 环境中开发程序的过程.....	1
1.2 程序调试.....	6
2 Borland C++V3.1 集成开发环境.....	10
2.1 Borland C++V3.1 环境中开发程序的过程.....	10
2.2 程序调试.....	24
2.3 建立工程文件.....	29
第二部分 C 语言实验指导.....	32
实验一 简单程序设计实验.....	32
1 实验目的.....	32
2 实验相关知识.....	32
3 实验范例.....	32
4 实验内容.....	34
实验二 选择程序设计实验.....	37
1 实验目的.....	37
2 实验相关知识.....	37
3 实验范例.....	38
4 实验内容.....	41
实验三 循环程序设计实验.....	43
1 实验目的.....	43
2 实验相关知识.....	43
3 实验范例.....	44
4 实验内容.....	47
实验四 函数与变量的存储类型实验.....	49
1 实验目的.....	49
2 实验相关知识.....	49
3 实验范例.....	50
4 实验内容.....	54
实验五 数组实验.....	56
1 实验目的.....	56
2 实验相关知识.....	56
3 实验范例.....	56
4 实验内容.....	63
实验六 指针实验.....	65
1 实验目的.....	65
2 实验相关知识.....	65
3 实验范例.....	66
4 实验内容.....	70
实验七 指针的综合应用.....	72
1 实验目的.....	72

2	实验相关知识.....	72
3	实验范例.....	73
4	实验内容.....	76
实验八	字符串实验.....	78
1	实验目的.....	78
2	实验相关知识.....	78
3	实验范例.....	79
4	实验内容.....	82
实验九	结构和联合实验.....	84
1	实验目的.....	84
2	实验相关知识.....	84
3	实验范例.....	85
4	实验内容.....	90
实验十	文件操作与图形实验.....	93
1	实验目的.....	93
2	实验相关知识.....	93
3	实验范例.....	94
4	实验内容.....	99

第一部分 环境配置

1 VC++6.0 集成开发环境

C 语言程序的集成开发环境较多，较常用的为 VC++6.0 集成开发环境和 Borland C++V3.1 集成开发环境。以下简介在 VC++6.0 集成开发环境中设计实现及调试 C 语言程序的方法。

1.1 VC++6.0 环境中开发程序的过程

Visual C++是 Microsoft 公司的 Visual Studio 开发工具箱中的一个 C++程序开发包，是基于 Windows 平台的可视化开发环境。从最早期的 1.0 版本，发展到最新的 6.0 版本，Visual C++已经有了很大的变化，在界面、功能、库支持等方面都有了许多的增强。最新的 6.0 版本在编译器、MFC 类库、编辑器以及联机帮助系统等方面都比以前的版本有了较大的改进。

Visual C++安装完成后，在开始菜单的程序选单中选择 Microsoft Visual Studio 6.0 图标，点击其中的 Microsoft Visual C++ 6.0 即可运行（也可在 Window 桌面上建立一个快捷方式，以后可双击运行）。第一次运行时，将提示如图 1.1 所示对话框。单击“下一提示”按钮，将看到各种操作提示；如果下次运行不需要此对话框，则取消选中“再启动时显示提示”复选框。单击“结束”按钮，关闭提示对话框，进入 Visual C++ 6.0 开发环境见图 1.2。

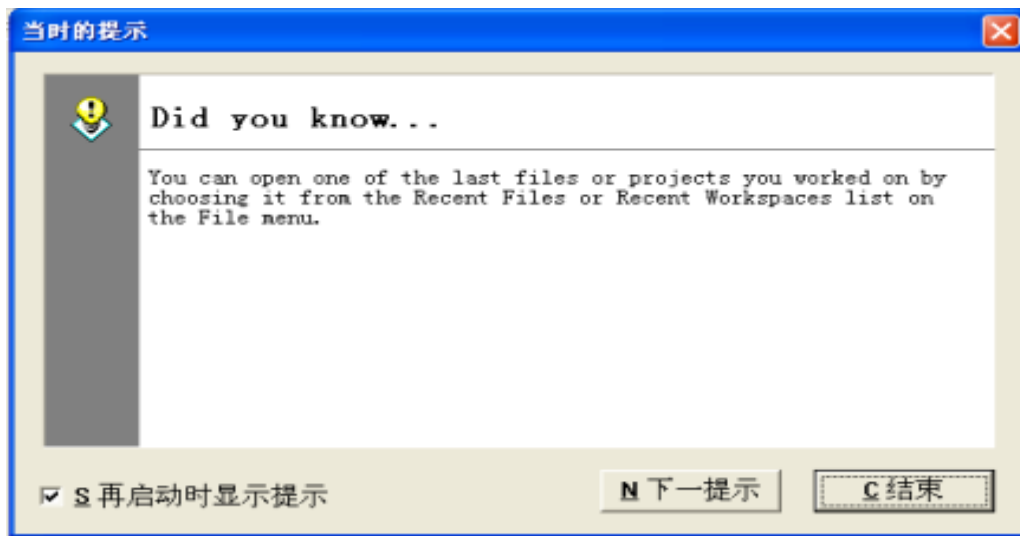


图 1.1 Visual C++ 6.0 启动提示对话框

Visual C++ 6.0 开发环境界面由标题栏、菜单栏、工具栏、项目工作区窗口、文档窗口、输出窗口和状态栏等组成。

进入 Visual C++ 6.0 环境后，我们就可以按照下列步骤进行 C 语言程序的编辑、编译、调试及运行了。

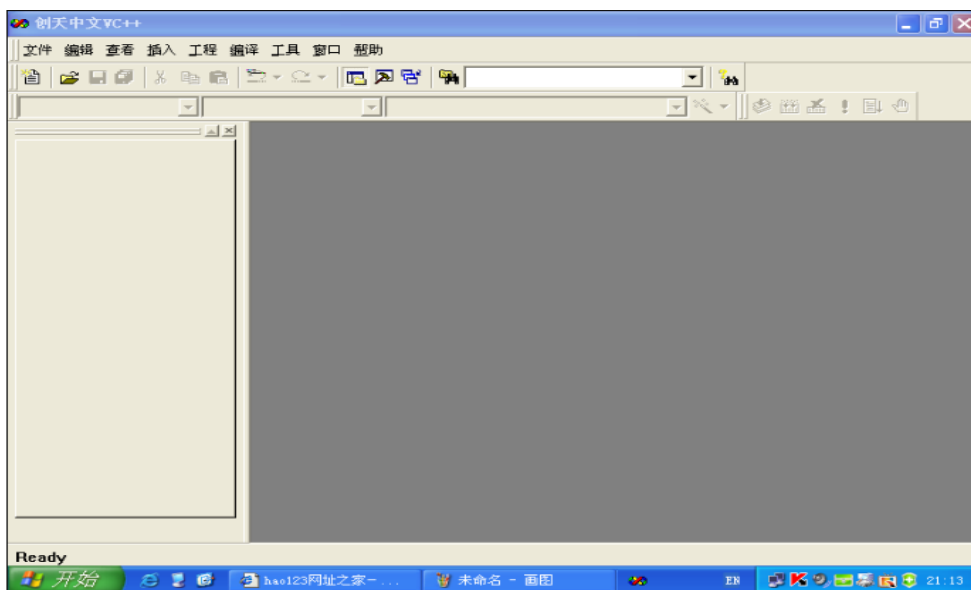


图 1.2 Visual C++ 6.0 开发环境主界面

1、建立新的工程

(1) 进入 Visual C++ 6.0 环境后，选择菜单“文件| 新建”，在弹出的对话框中单击上方的选项卡“工程”，选择“Win32 Console Application”工程类型，在“工程”一栏中填写工程名，在“位置”一栏中填写工程路径（目录），见图 1.3，然后按“确定”继续。



图 1.3 “新建”工程对话框

(2) 弹出如图 1.4 所示对话框，在该对话框中选择“An empty project”，建立空工程。单击“完成”按钮，弹出“新建工程信息”对话框，单击“确定”完成新工程的建立。

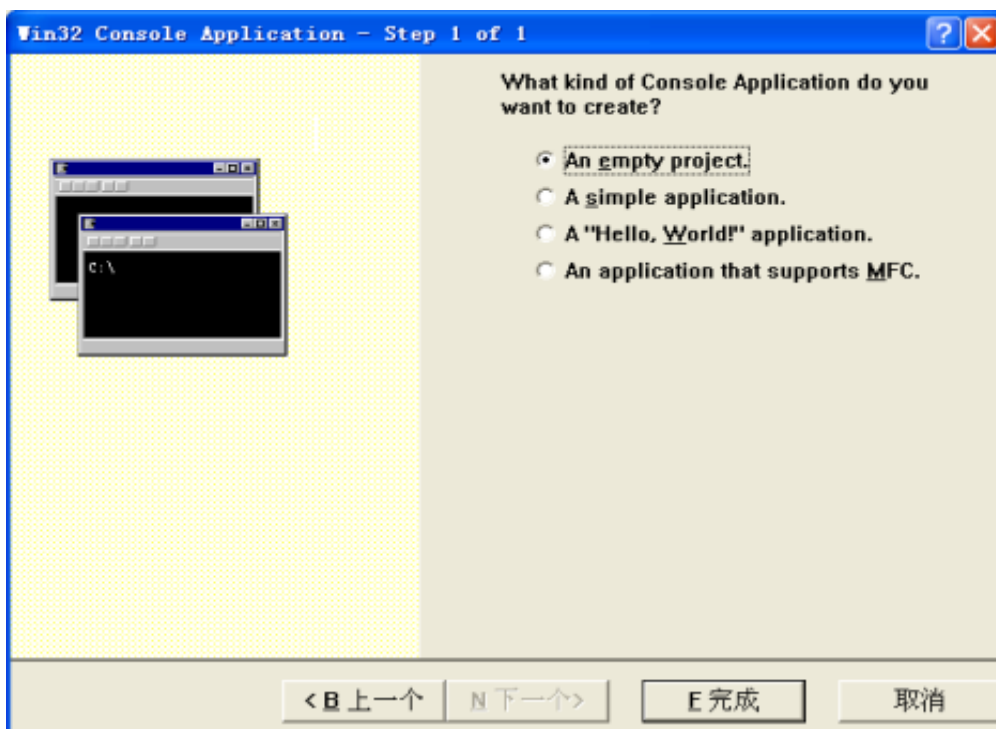
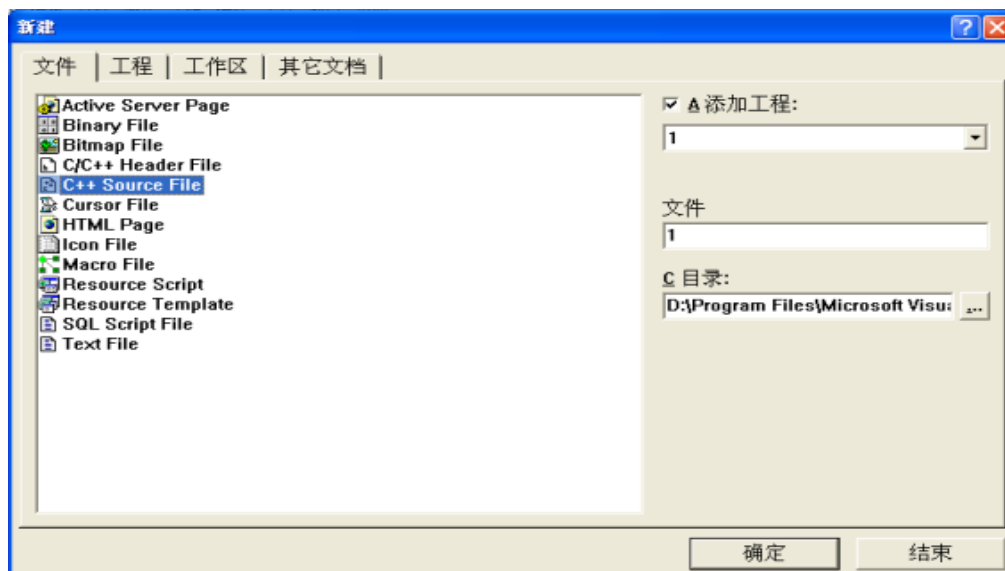


图 1.4 工程类型选择对话框

2、建立源程序

在新建立的空工程中，选择“文件|新建”菜单项，打开“新建”对话框，选择“文件”选项卡，在该选项卡中选择“C++ Source File”选项。在右边的“文件”文本框中输入源文件名，单击“确定”按钮。如图 1.5 所示。



如图 1.5 建立源程序对话框

3、编辑源程序

在文档窗口中，可进行源程序代码的输入或者修改，结束时一定要保存该源程序文件。如图 1.6。

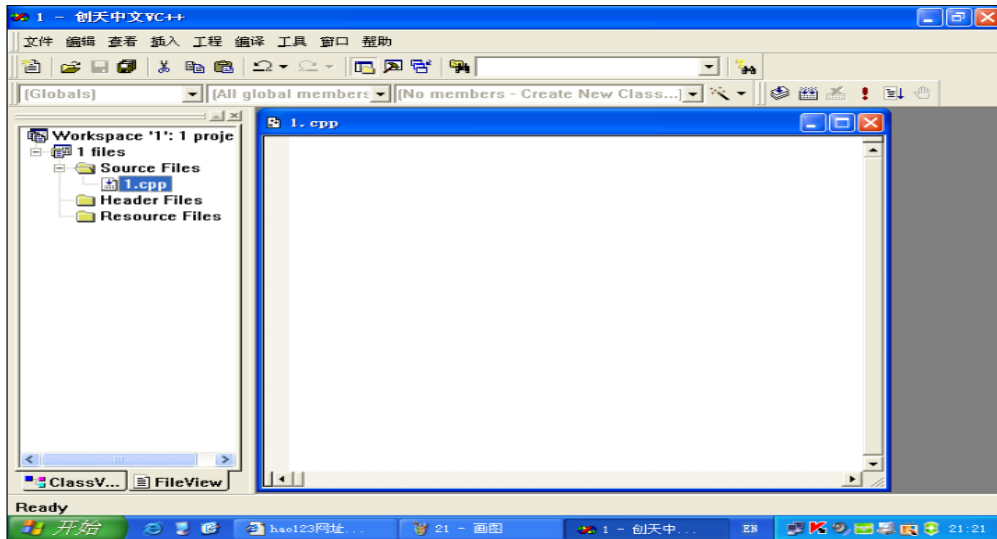


图 1.6 Visual C++ 6.0 编辑窗口

4、编译程序

对源程序进行编译可以检查语法错误，在 Visual C++ 6.0 环境中，通过“编译”菜单、“编译”工具或 Ctrl+F7 对源文件进行编译。如图 1.7 所示。没有语法错误的源程序文件可以编译为目标程序文件。如果有语法错误，将在输出窗口中显示错误提示信息，双击该错误提示信息，查找错误并改正。如图 1.8 所示。

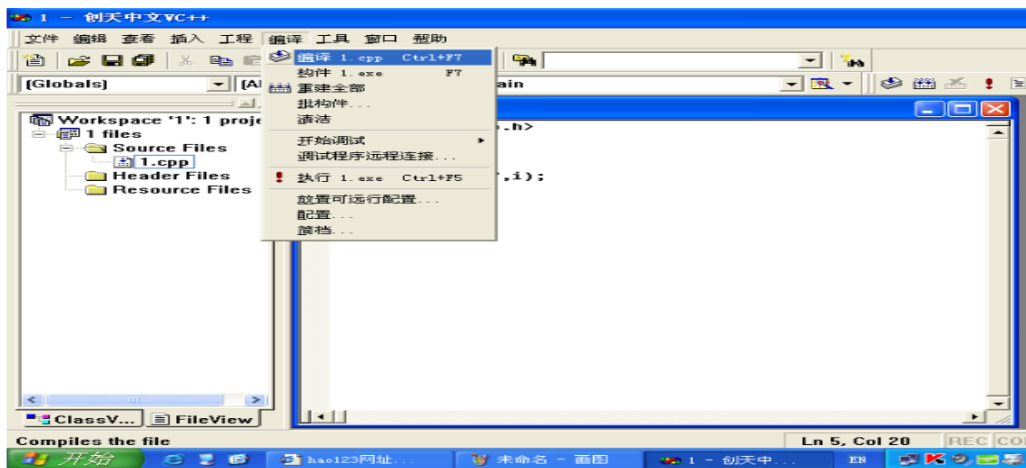


图 1.7 Visual C++ 6.0 编译窗口

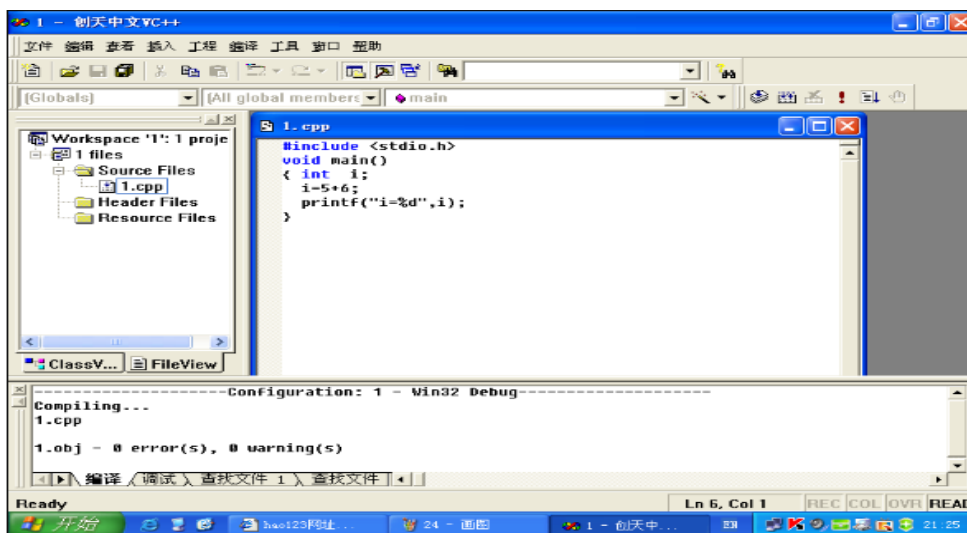


图 1.8 查找错误窗口

5、运行程序

程序编译成功后，通过“编译”菜单中的“运行”工具或 Ctrl+F5 运行程序，并显示运行结果。如图 1.9 所示。

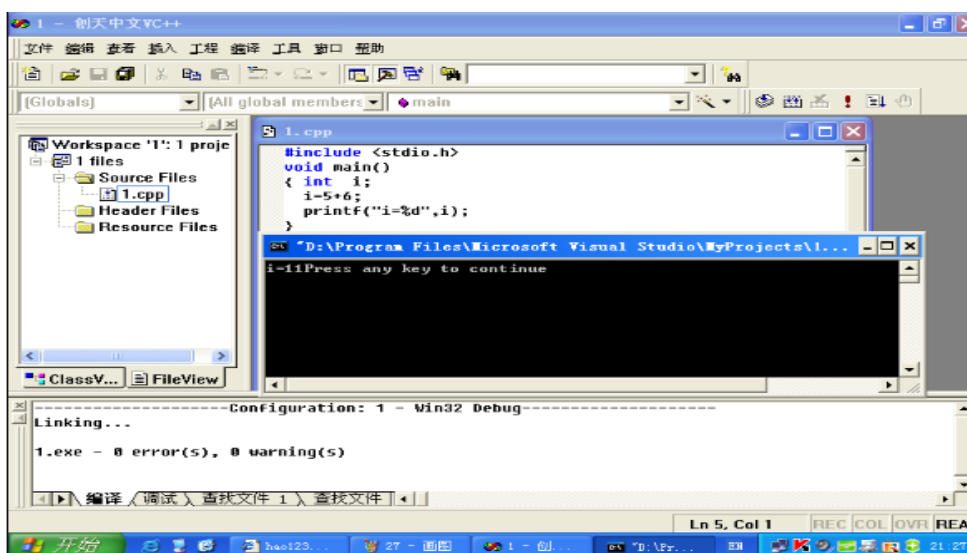


图 1.9 运行结果窗口

另外，可以加载以 .c 为扩展名的 C 源程序。方法是：双击程序文件名，将直接进入开发环境进行调试。

为了保护已完成的程序，注意保存。

1.2 程序调试

程序出错的类型大致可以分为语法错误和逻辑错误。语法错误可以通过编译器的出错信息得到纠正，而逻辑错误则不能。Visual C++ 6.0 提供了 Debug 功能，可以快速找到逻辑错误。

Visual C++ 6.0 的“Debug”菜单如图 1.10 所示，下面对常用的调试命令进行简要介绍：

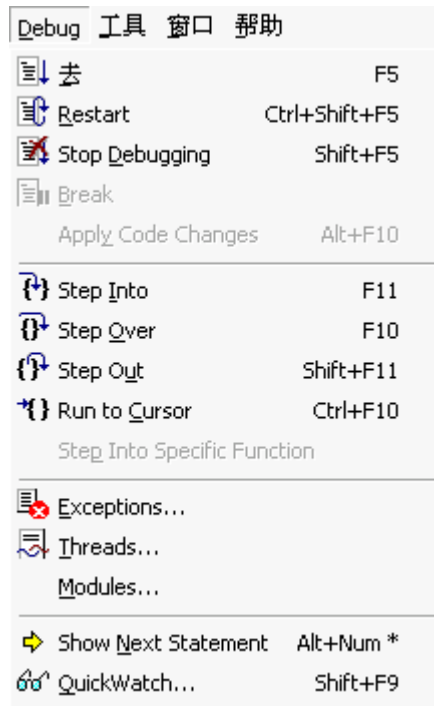


图 1.10 Debug 菜单

1、Go 命令（快捷键：F5）：系统将编译、链接，自动运行程序，但是会在程序设置的断点(breakpoint)处停下。

2、Restart 命令（快捷键：Ctrl+Shift+F5）：重新调试程序。

3、Stop Debugging 命令（快捷键：Shift+F5）：终止(所有)调试,并返回到常规编辑状态。

4、Step Into 命令（快捷键：F11）：单步执行每条语句，在遇到函数的时候，系统将进入函数，单步执行其中的语句。

5、Step Over 命令（快捷键：F10）：单步执行每条语句，但在遇到函数时候，系统将把函数当作“一条语句”来执行，自动执行其中的内容，而不进入函数内部单步执行。

6、Step Out 命令（快捷键：Shift+F11）：结束对所调用函数的调试，跳出函数。

7、Run to cursor 命令（快捷键：Ctrl+F10）：系统将自动执行到用户光标所指的语句前。

Visual C++ 6.0 还提供了一些帮助调试的窗口（可以通过“View”菜单下的“Debug Windows”子菜单中的命令来打开）：

1、观察窗口（操作：Watch 快捷键：Alt+3）：输出变量和表达式的名字和值。

2、调用栈窗口（操作：Call Stack 快捷键：Alt+7）：显示所有未返回的函数调用的堆栈。

3、内存对话框（操作：Memory 快捷键：Alt+6）：显示当前内存的内容。

4、变量窗口（操作：Variables 快捷键：Alt+4）：输出当前和前面的语句中使用的变量信息和函数的返回值信息以及当前函数的局部变量信息。

5、寄存器窗口（操作：Register 快捷键：Alt+5）：显示一般用途寄存器和 CPU 状态寄存器的当前内容。

6、反汇编窗口（操作：Disassembly 快捷键：Alt+8）：显示编译后的程序经反汇编后的汇编语言代码。

以上窗口也可以使用“Debug”工具栏来打开，方法是在环境窗口的菜单栏中单击鼠标右键，选择“Debug”命令。如图 1.11 所示：



图 1.11 Debug 工具栏

图标依次对应于命令：“Restart”、“Stop Debugging”、“Break Execution”、“Apply code change”、“Show next statement”、“Step Over”、“Step Out”、“Run to Cursor”、“Quickwatch”、“Watch”、“Variables”、“Register”、“Memory”、“Call Stack”和“Disassembly”。

设置断点的方法：在程序代码中，移动到需要设置断点的那一行上，单击鼠标右键，在弹出的快捷菜单中选自“Insert/Remove Breakpoints”命令，你可以看到代码行的左端出现了一个红色的圆点——那是 VC++中断点的标志，表示在此行代码上设置了一个断点。以后程序在调试过程中，每次执行到这里，都会停下，方便用户观察 watch 区域中的内容。

去除断点的命令与设置断点的命令相同：在已设置断点的地方，单击鼠标右键，在弹出的快捷菜单中选自“Insert/Remove Breakpoints”命令，左端的红色圆点就消失，断点被去除了。

下面分别用单步调试和断点调试两种方法调试以下程序：

```
#include "stdio.h"
void main()
{int f=1,i;
  for(i=1;i<=20;i++)
    f=f*i;
  printf("f=%d\n",f);
}
```

单步调试方法的特点是程序执行时一次只执行一行，每执行一行，程序就会停止运行，这时，可以通过变量窗口和观察窗口检查有关变量和表达式的值，以此来判断是否正确，从而找到错误的位置。

断点调试方法的特点是在程序中的某行语句位置设置断点，当程序执行到此语句的前一条语句时停止运行，此时在观察窗口中插入必要的表达式，以此来检查错误。

1、使用单步调试方法调试

（1）对上面的程序进行编辑、编译、连接并运行。运行结果为“f=-288522240”，显然结果错误。

（2）选择“Step Into”命令，界面中增加了两个窗口如图 1.12 所示。

增加的左边窗口是变量窗口，右边是观察窗口。在程序的左边有一个黄色箭头。此时在观察窗口中提示错误信息：“CXX0069: Error: variable needs stack frame”表示变量定义类型有错误，改为“long int f=1;”。

(3) 连续按 F10 键，在变量窗口中发现当 i 为 17 时，f 的输出值为负数，把 “printf(“f=%d\n”,f);” 改为 “printf(“f=%ld\n”,f);” ，仍然存在这个问题，说明我们计算的数值太大，已经超过了长整型的取值范围，若把 17 改为 16，则结果正确。

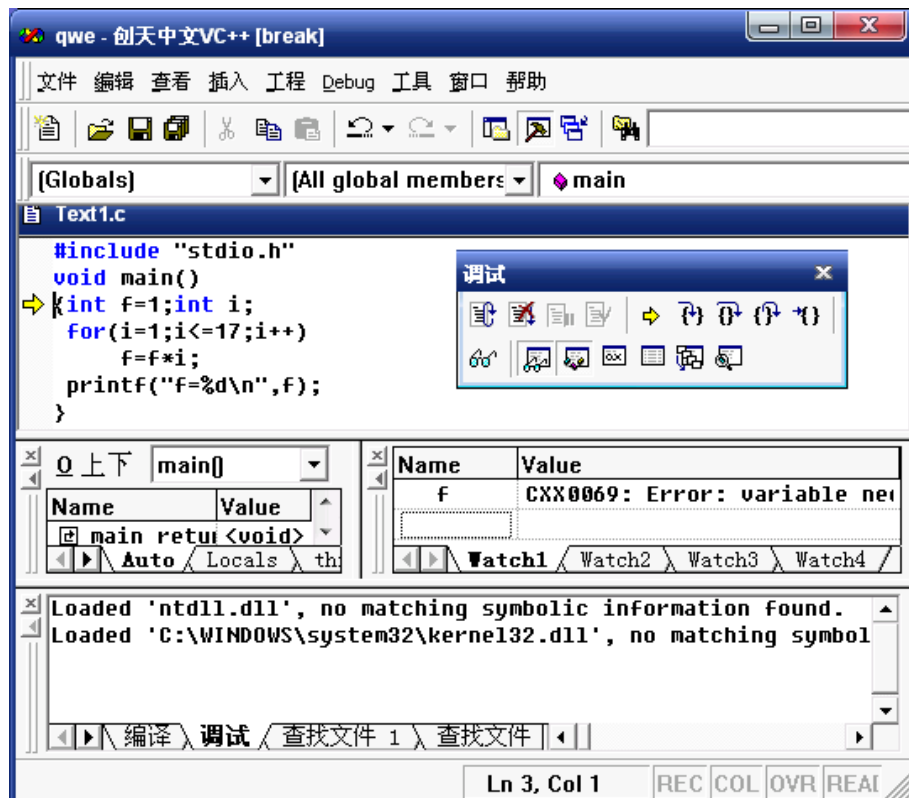


图 1.12 单步调试窗口

2、使用断点调试方法调试

(1) 对上面的程序进行编辑、编译、连接并运行。运行结果为 “f=-288522240”，显然结果错误。

(2) 由于输出结果错误，我们在 “printf(“f=%ld\n”,f);” 处设置断点。

(3) 选择 “go” 命令，进入调试器状态，界面如图 1.13 所示：

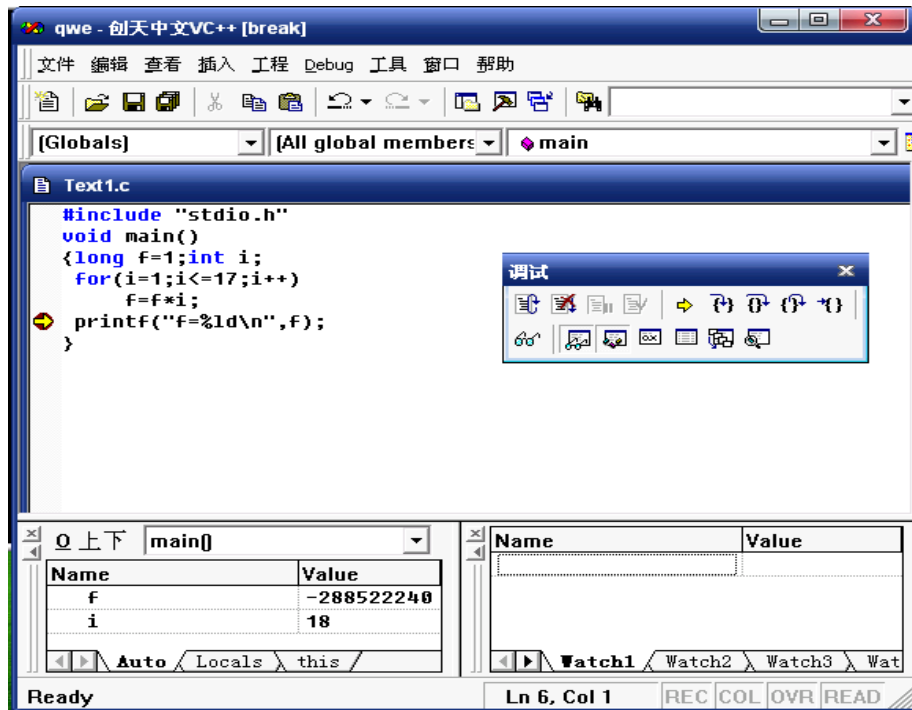


图 1.13 断点调试窗口

程序在断点处停止运行，从变量窗口中显示，可以看出变量 f 的值为负数，超出了表示范围。把“17”改为“16”，重新调试，调试成功。

2 Borland C++V3.1 集成开发环境

2.1 Borland C++V3.1 环境中开发程序的过程

Borland C++V3.1 (以下简称为 BC31) 是 Turbo C++ 3.0 的更新版本, 其配合 Windows 版, 增加多媒体(Multi-Media)、笔(Pen)和 MCI (Media Control Interface)等控制接口。同时新增 Win Spector 工具程序, 此工具程序是当 Windows 程序发生 General Protection Faults 时, 提供发生此错误的地方, 及当时机器的状态。Borland C++ 3.1 可供程序设计人员开发 C++及 C 程序。其中的 C++是依照 AT&T2.1 标准设计的, 而 C 则是 ANSI C。因此开发出来的程序兼容性高. 移植性(Portable)强。

Borland C++V3.1 可以在 DOS 的环境下运行, 也可以在 win2000/winxp/vista/win7 下完美使用, 接下来介绍的 Borland C/C++ 3.1 系统是在虚拟的 DOS 环境下使用的, 同时包括了图形程序。该系统中所有环境均已经配置好, 不用修改。双击执行“Borland C++ 3.1”即可。如图 1.1 所示。

名称	修改日期	类型	大小
DISK_C	2016/6/17 11:06	文件夹	
dosbox	2016/6/19 16:07	文件夹	
Borland C++ 3.1.vbs	2008/5/2 10:17	VBScript Script ...	1 KB
说明.txt	2010/6/26 22:59	文本文档	1 KB

图 1.1 BC31 系统下的文件目录

1. BC31 的使用

该 Borland C++ 3.1 系统虚拟了 DOS 环境, 文件夹 DISK_C 就是虚拟的 DOS 环境的 C 盘; 文件夹 DISK_C 下的文件夹 BORLANDC 表示 Borland C/C++ 3.1 程序文件, 编辑的程序都要放在该文件夹下面; 文件夹 dosbox 下是虚拟 DOS 环境的软件 DosBox 0.72。使用文件路径的时候需要注意目录路径。

例如, 图形程序中初始化图形环境的代码:

```
int gdriver=VGA, gmode=VGAHI;  
initgraph(&gdriver, &gmode, "C:\\BORLANDC\\BGI");
```

其中: initgraph 是函数名, 功能为初始化图形系统, gdriver 表示图形驱动器, gmode 表示图形模式, "C:\\BORLANDC\\BGI" (也就是 path) 是指图形驱动程序所在的目录路径, 此处的"C:\\"实际上是指 DISK_C 文件夹。

启动系统“Borland C++ 3.1”后, 将出现 BC31 的开发环境页面, 如图 1.2 所示。其中, 同时按下 Alt + Enter 可以切换全屏/窗口。



图 1.2 BC31 编程界面

2. BC31 的菜单结构和导航操作

1) BC31 的菜单结构

BC31 是由主菜单和下拉子菜单组成的多层菜单结构。启动系统后，在 File 中打开一个程序，如图 1.3 所示的主屏幕，由主菜单条（亮灰底面背景）、编辑窗口（亮蓝色底面背景）、信息窗口（亮绿底面背景）和操作提示行（亮灰底面背景）等 4 部分组成。主菜单条位于屏幕的最上方排列成一个横行并总是显示在屏幕上，其中包含有 File（文件管理）、Edit（编辑）、Search（搜索）、Run（运行）、Compile（编译）、Debug（调试）、Project（工程项目）、Options（选择）、Window（窗口管理）和 Help（帮助）等主菜单项，且每个英文单词的第 1 个字母用英文大写红色字母以便于操作者记忆快捷操作方式的按钮。

2) BC31 的导航操作

所谓“导航操作”就是能够把握住方向的基本操作，也是初学者首先必须掌握的基本操作方法。BC31 既可以用键盘进行操作，也支持鼠标操作，它具有如下 3 种导航操作：

① 不管是哪个窗口指定为当前窗口，也不管操作进行到哪一步，按功能键 F10 则启动主菜单，主菜单一启动通常选中第 1 个菜单项 File（文件管理），即一个亮绿色的方框覆盖该菜单项。



图 1.3 BC31 的主窗口

② 用←和→光标键可移动该亮绿色方框，即按主菜单的排列顺序向左或向右移动选择项，每按一次→光标键则绿色方框向右移动一个主菜单项，如从“File（文件管理）”移动到“Edit（编辑）”，再按一次→光标键移动到“Search（搜索）”。每按一次←光标键则绿色方框向左移动一个主菜单项，且主菜单条的左右边缘是连接在一起的，即当选中第1个菜单项“File（文件管理）”时，再按两次←光标键则移动到“Help（帮助）”。若选中了某个主菜单项后再按回车键，则进入到该主菜单立即弹出它的下拉子菜单，如图1.3所示，当选中“File（文件管理）”后按压回车键则弹出下拉子菜单，其中包含有New（新建）、Open（打开，F3）、Save（保存，F2）、Save as（另存）、Save all（全部保存）、Chang dir（更改路径）、Print（打印）、Dos Shell（DOS 界面）和Quit（退出，“Alt+X”）等子菜单项。在BC31集成开发环境中也可以用鼠标选择菜单项，将鼠标箭头移动到所要选择的主菜单项如Options，按压鼠标左键则进入到Options主菜单弹出下拉子菜单，接着再用鼠标左键点击某个子菜单项，若它是最终的菜单项命令则立即执行该菜单项命令，或者采用鼠标拖放操作，即按住鼠标左键不放，沿着下拉子菜单上下拖动到所要选择的子菜单项再松开鼠标左键则执行该子菜单命令，若操作者又临时改变主意，可将鼠标箭头移动到下拉子菜单的方框范围以外，松开鼠标左键则不会执行任何命令。

③ 选择主菜单项的操作还有快捷方式，用Alt键加上每个主菜单项的第1个英文字母（红色），即Alt+F，Alt+E，...，Alt+H等。在BC31集成开发环境中也可以用鼠标左键方便地直接点击要选择的主菜单项，不仅启动了主菜单，且直接进入到该主菜单项弹出了它的下拉子菜单。下拉子菜单也有快捷操作方式，当打开某个下拉子菜单后，再按每个子菜单项中红色字母所指定的按键，如图1.3所示，当File主菜单的下拉子菜单打开时，按O键则执行“Open（打开文件）”子菜单命令，按S键则执行“Save（保存）”子菜单命令，按a键则执行“Save as（另存）”子菜单命令...如此类推。

3. BC31 集成开发环境设置

1) 六种编译模式

为了适应不同的应用场合，BC31提供了6种内存编译模式，即微型(tiny)、小型(small)、中型(medium)、紧凑型(compact)、大型(large)和巨型(huge)等模式，各种编译模式以不同的方式管理计算机的内存(memory)并控制程序代码和数据区的大小，同时还决定程序的执行速度。微型模式(tiny model)的所有代码、数据和堆栈都在同一个64KB

(kilobytes, 千字节)内存段内，这种编译方式所产生的代码容量(占用内存空间的字节数)最小，执行速度最快，适用于内存空间不到1MB的各种微小型系统。小型模式(small

model) 是 BC31 的缺省编译模式, 即操作者没有指定编译模式时系统自动选用的编译模式, 它所生成的代码在同一个 64KB 内存段内, 而数据、堆栈和附加段占用另一个 64K 内存段, 因此, 所生成代码的执行速度与微型模式相同, 而代码容量却增大一倍左右, 这种模式适用多种编程任务, 也是应用最多的一种编程模式。中型模式 (medium model) 用于大代码程序, 所生成的代码超过 64K 内存段可达到 1MB (megabytes, 兆字节), 而数据、堆栈和附加段最多为 64KB, 这种模式适合于数据量较小的大代码程序, 其程序执行速度比小模式要慢得多, 但读写数据的速度与小模式一样快。紧凑型模式 (compact model) 的代码容量和数据量与中型模式正好相反, 即所生成的代码容量限制在 64KB 内存段内, 而数据量超过 64KB, 适合于代码较短但数据量大的程序, 程序执行速度与小型模式相同, 而数据读写时速度较慢。大型模式 (large model) 的代码容量和数据量都超过 64KB, 两者均可达 1MB, 而全部静态数据如一个数组存放的数据不超过 64KB, 它适合于需要处理大量数据的大程序, 但它的运行速度也就远慢于上述几种模式。巨型模式 (huge model) 与大型模式基本相同, 代码容量和数据量均超过 64KB 内存段, 只是静态数据也超过 64KB 内存段, 其运行速度比大型模式还要慢。用户可根据所使用的目标系统 (所开发好的程序最终移植到实际运行的系统) 具体情况来选择编译模式。

2) BC31 环境的设置

在第 1 次进行程序开发时, 应检查一下 BC31 集成开发环境的主要设置, 如编译模式、编译路径、输出目录路径和链接方式等。这是通过执行主菜单 Options 的子菜单命令进行的, 其主要应检查的设置项目如下:

① 设置编译模式: 操作者沿着 “Options (主菜单) / Compiler (第 1 级子菜单项) / Code Generation (第 2 级子菜单项)” 的操作路径用键盘操作或用鼠标点击, 将弹出 “Code Generation” 对话框如图 1.4 所示, 其设置是指定集成编译器以何种方式生成扩展名为 “.OBJ” 的目标代码文件。其左上边为 “Model” 选择框, 其内包含 6 个 “单选” 按钮。“单选” 按钮就像收音机的波段组合开关, 这组开关只能选中一个选择项 (圆括号内有●点的为被选中的), 如图 1.4 所示为选中 Small 编译模式。用户可根据实际需要改变编译模式, 可用 ↑ 或 ↓ 光标键上下移动●点选择其他项, 或者用鼠标左键直接点击所要选择的编译模式再点击 “OK” 按钮即可。初学者通常选择 Small 编译模式, 因此它是系统的缺省编译模式, 即系统原始设置的编译模式。

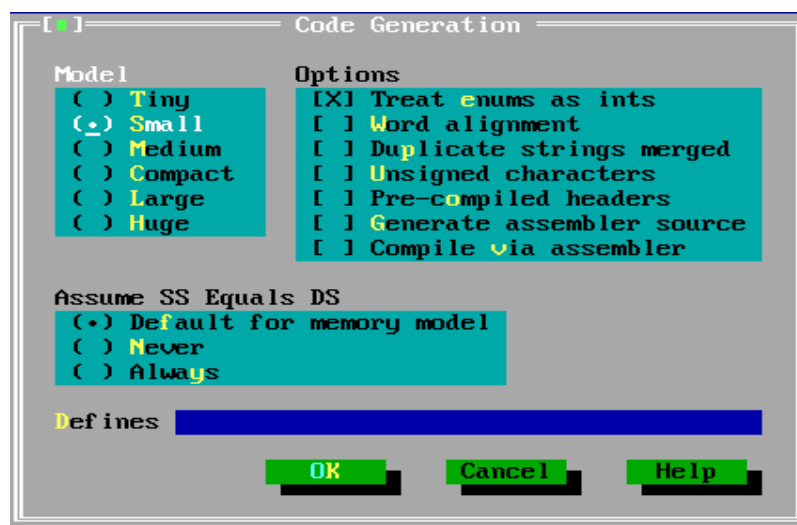


图 1.4 Code Generation 对话框

② 检查编译路径：操作者沿着“Options（主菜单）/Directories（第1级子菜单项）”的操作路径用键盘操作或用鼠标点击，将弹出“Directories”对话框如图1.5所示，其内包含4个文本框。第1个文本框的标题为“Include Directories”，即指定系统标准头文件所存放的磁盘驱动器号和目录路径名为“C:\BORLANDC\INCLUDE”，C语言标准函数库的所有头文件均放在此目录下，在源程序中的#include语句中都是用一对尖角号“< >”括起来的头文件名，系统都是在该目录下查找所需要的标准头文件。第2个文本框的标题是“Library Directories”，它指定C语言标准函数库所存放的磁盘驱动器号和目录路径名为“C:\BORLANDC\LIB”，它和“C:\BORLANDC\INCLUDE”通常称为“标准查找路径”。第3个文本框的标题是“Output Directories”，由它指定在编译和链接过程中由系统自动生成的目标代码文件(filename.obj)、可执行文件(filename.exe)和图形文件(filename.MAP)存放的目录路径，通常，把它们与其源文件放在同一个目录下。若此项设置为空时则把这些文件存放在“当前目录”下，所谓“当前目录”就是显示在编辑窗口上、已打开的源文件所在的目录。

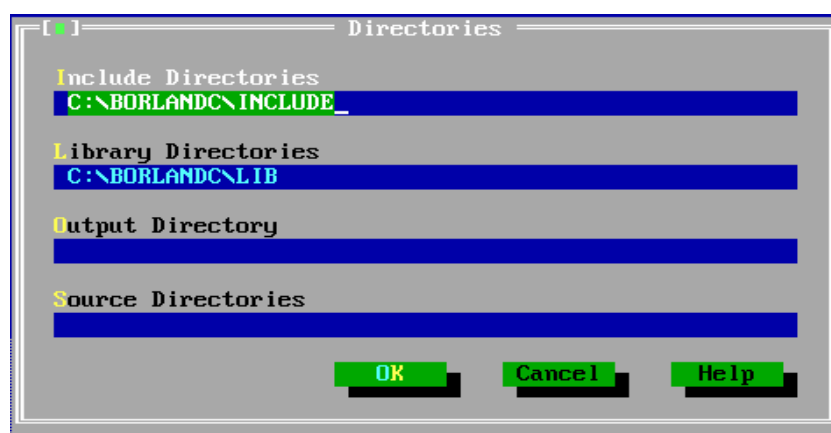


图 1.5 Directories 对话框

③ 设置绘图方式：BC31 系统具有很强的绘图功能，利用其功能强大、内容丰富的 BGI 图形库编程简便、直观，且便于结合定量计算来绘制图形，特别适于初学者编写绘图程序。但是，必须进行“设置绘图方式”的操作，其操作方法是沿着“Options（主菜单）/Linker（第1级子菜单项）/Libraries（第2级子菜单项）”的操作路径用键盘操作或用鼠标点击，将弹出“Libraries”对话框如图1.6所示，其内包含 Libraries、Object Windows Library（BC31 的标准类库 OWL）、Container Class Library（标准容器类库）和 Standard Run-time Libraries 等4个选择框，每个选择框包含有多个“单选按钮”。用鼠标左键点击做如图所示选择即可。

然后，再用鼠标左键点击“OK”按钮则设置被系统确认，在链接操作时将与 BGI 图形库（GRAPHICS.LIB）进行链接，即使源程序中可能没有调用标准绘图函数。



图 1.6 Libraries 对话框

4. 常用热键

绝大多数菜单项命令都有对应的热键，这些热键有的是单个键如 F1 ~ F10 单个功能键，有的是两个键的组合，如上述选择主菜单项的热键 Alt+F、Alt+E、...、Alt+H 和 Alt+X 等。经常使用的热键以及对应的菜单项命令和功能如表 1.1 所示。

表 1.1 常用热键对应的菜单项命令和功能

键	对应的菜单项命令	功 能
F1	Help / Contents	启动帮助系统，显示帮助窗口
F2	File / Save	保存活动（被激活）编辑窗口中的当前文件
F3	File / Open	显示“Open（打开）”对话框，用以打开一个文件
F4	Run / Go to Cursor	程序运行到光标所在行
F5	Window / Zoom	按压一次放大活动窗口，再按一次恢复
F6	Window / Next	在所有已打开窗口间循环切换
F7	Run / Trace Into	调试程序时跟踪到被调用函数体内
F8	Run / Step Over	调试程序时单步执行函数调用不跟踪到被调用函数体内
F9	Compile / Make EXE	对活动编辑窗口中的文件或工程项目进行编译、链接生成扩展名为 .EXE 的可执行文件
F10		转回到主菜单
Alt+F9	Compile（主）/Compile	编译生成扩展名为 .OBJ 的目标代码文件
Ctrl+F9	Run（主）/ Run	编译、链接和运行程序
Ctrl+Del	Edit / Clear	删除被激活的编辑窗口或剪贴板窗口的选择块且不存入剪贴板
Ctrl+Ins	Edit / Copy	把任意窗口中的选择块复制到剪贴板
Shift+Del	Edit / Cut	把任意窗口中的选择块复制到剪贴板并从窗口中删除掉
Shift+Ins	Edit / Paste	把剪贴板中的选择块插入到任意活动窗口内的光标所在位置
Alt+O	Window / List	弹出一个对话框，列出所有被打开的窗口
Alt+F3	Window / Close	关闭活动窗口
Alt+F4	Debug / Inspect	打开一个观察对话框窗口
Alt+F5	Window / User Screen	切换到用户（显示输出结果的）屏幕，按任意键返回
Ctrl+F1	Topic / search	显示在当前编辑窗口内光标所在处的编程语言单词的帮助信息
Ctrl+F5	Window/Size/Move	改变活动窗口的大小和位置

5. 源程序的编辑和输入

按照“File（文件管理）/New（新建）”的操作路径用鼠标左键点击，立即进入到编辑窗口，并弹出一个新窗口其标题行中间由系统自动提供的文件名为“NONAME00.CPP”，光标停留在第 1 行第 1 列。在“编辑窗口”内，键入用户编写的 FileName.cpp（或 FileName.c）的具体内容，所采用的编辑操作方法与 Windows 有些不同之处。现在选择如下几个特别有用的加以介绍：

1) Edit 主菜单中的大多数命令如 Cut（剪切）、Copy（复制）和 Paste（粘贴）等命令都是对选择块和剪贴板进行操作的。选中要复制的内容，如图 1.7 出现的白色块即为选择块。在按下 Ctrl 键的同时按 Ins 键，或者直接点击 Edit 菜单中的 Copy。然后用光标键将光标移动到要粘贴的起始处，按下 Shift 键的同时按 Ins 键，或者直接点击 Edit 菜单中的 Paste，则出现如图 1.7 所示的图形。即将选择块复制到新的一行。

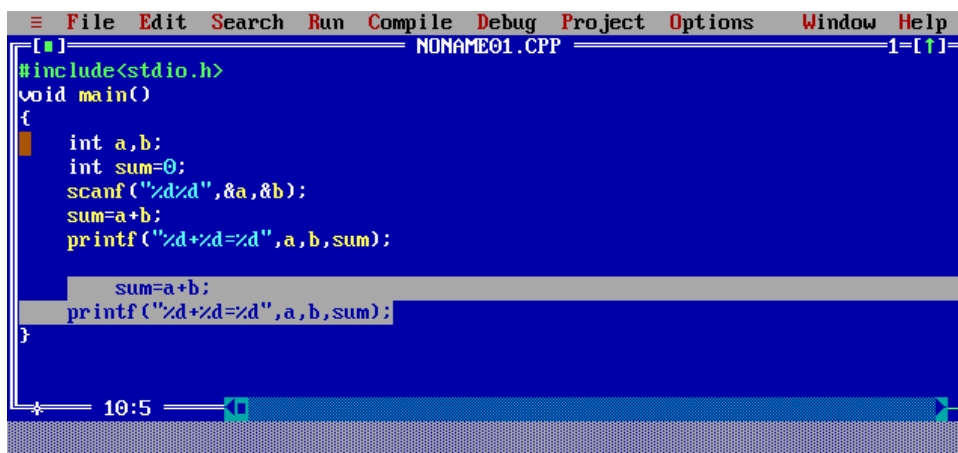


图 1.7 选择块的设置和复制

2) 用 Cut (剪切) 或 Copy (复制) 命令将选择块的信息复制到“剪贴板”上。BC31 新提供了一个“剪贴板窗口”，所谓剪贴板 (clipboard) 是操作系统开辟的一个存放文本信息的专用内存空间，而选择块就是编辑窗口或剪贴板窗口内一段白色高亮度显示的文本。操作者沿着“Edit (编辑) / Show Clipboard (显示剪贴板窗口)”的操作路径用鼠标左键点击，即可打开如图 1.8 所示的剪贴板窗口。



图 1.8 剪贴板窗口

6. BC31 集成开发环境的专用窗口

集成开发环境 IDE 内部专用窗口：BC31 集成开发环境还预先定义有几个专用窗口，在此结合 Window 主菜单下的窗口管理命令加以介绍。

1) IDE 预先定义了如下内部专用窗口：

- Message (信息) 窗口：当对编辑窗口上的当前文件进行编译、链接时所产生的错误和警告信息将在该窗口内显示。
- Output (输出) 窗口：显示程序的输出结果。
- User Screen (用户屏幕)：用“Window/User Screen”菜单命令或者按“Alt+F5”键即可切换到用户屏幕，再按任意键就返回到主屏幕。
- Watch (监视) 窗口：该窗口显示指定的监视表达式和它们的当前值。
- Project (工程项目) 窗口：显示当前工程项目文件。
- Project notes (项目笔记) 窗口：显示工程项目的笔记。
- Register (寄存器) 窗口：显示 CPU 所有寄存器的当前值，只能查看这些寄存器的值而不能修改，该窗口如图 1.9 所示。

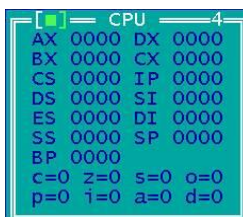


图 1.9 寄存器窗口

2) Window 主菜单还包含如下的

窗口管理命令：用鼠标左键点击“Window 主菜单”，则弹出一个下拉子菜单项由用户选择如下菜单命令：

- Size/Move: 该菜单命令一旦执行将当前窗口变成可变状态, 即边框线从白色双实线变成亮绿色单实线, 接着用 ←、↑、→ 和 ↓ 光标键可在主屏幕上向左、向上、向右和向下移动该窗口到操作者所指定的位置, 再按回车键则该窗口恢复到原来固定不变的状态。

- Zoom: 该菜单命令可缩放被激活的当前窗口。它与当前窗口右上角的“缩放块”具有相同的功能, 若用鼠标左键点击一次则放大, 再点击一次则缩小。

- Tile: 该菜单命令把所有已打开的窗口并列显示在屏幕上, 即如图 1.10 所示, 所有窗口互不重叠按照预先定义好的格式在屏幕上同时显示出来, 可以用鼠标左键点击指定窗口或者用热键“Alt+窗口序号”进行窗口间的切换。

- Cascade: 该菜单命令把所有已打开的窗口堆叠式显示在屏幕上, 即新打开窗口将覆盖原来显示的窗口, 它是窗口缺省设置的显示方式。

- Next: 集成开发环境能记录所有已打开窗口的激活顺序并将它们排列成一个循环, 每执行一次该菜单命令或者按一次热键 F6 将立即激活循环中相对于当前窗口的下一个窗口。若一直按下 F6 不放, 则按循环中的所有窗口排列次序顺序地切换。

- Close: 执行该菜单命令将关闭被激活的当前窗口, 并激活排列在上述循环中的前一个窗口。

- Close all: 关闭所有活动的窗口。

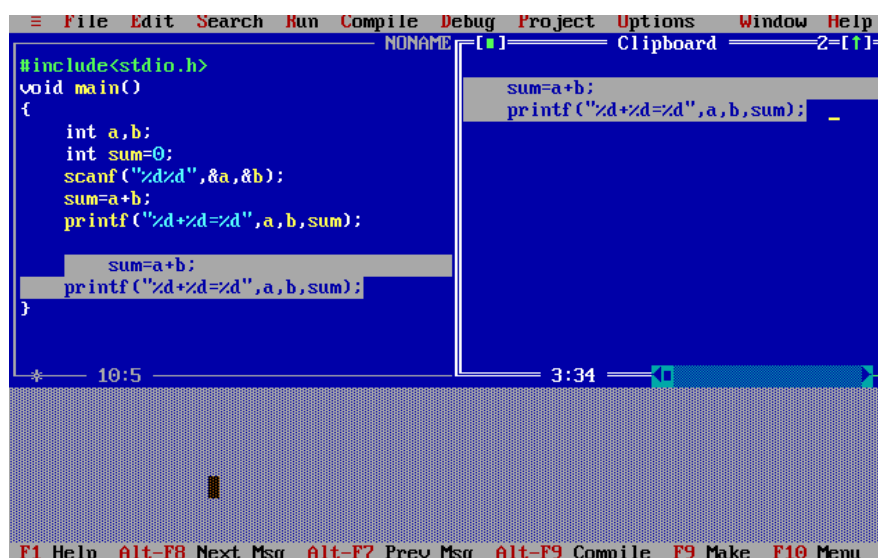


图 1.10 执行 Tile 菜单命令所有已打开的窗口并列显示

7. 存储和重新打开源程序

保存源程序的操作方法是沿着操作路径“File/Save”用鼠标左键点击或按热键 F2。对于新建的源程序第 1 次进行保存时, 在显示器屏幕上将弹出一个标题为“Save File As”的对话框如图 1.11 所示, 在该对话框中的“Save File As”文本框内系统将自动提供一个存储路径名和文件名, 其存储路径名是采用标准查找路径名“C:\BORLANDC\BIN\”, 而文件名是采用通用文件名“NONAME00.CPP”(第 1 次提供的文件名, 第 2 次为“NONAME01.CPP”, 第 3 次为“NONAME02.CPP”, 如此类推)。

若操作者要打开一个已存在的源文件, 可采用“File/Open”菜单命令, 将会弹出一个如图 1.12 所示的“Open a File (打开文件)”对话框。若此时操作者改变主意想选择其他目录路径下的文件, 沿着“File/Open”路径点击则弹出如图 1.12 所示的对话框, 点击 Name 文本框则光标停留在该框内即可敲入目录路径名。

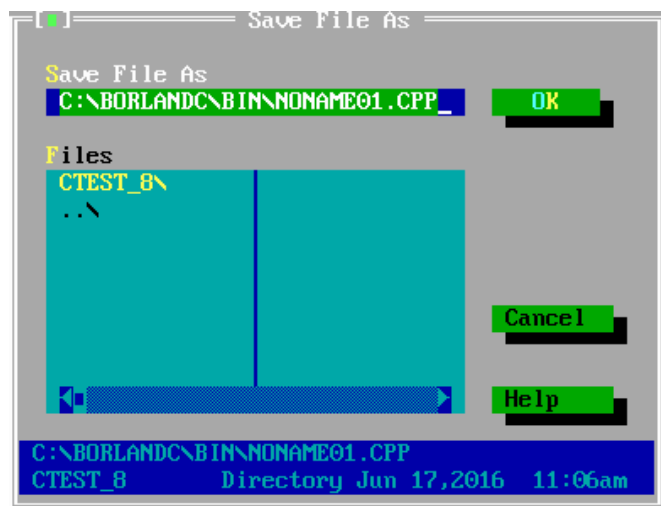


图 1.11 “Save File As (存储文件)”对话框

另外,在“Open a File(打开文件)”对话框中,可以用两种方式打开文件,其一是“Open”按钮,即打开一个新的编辑窗口并将所选择的新文件装入到该编辑窗口。其二是“Replace”按钮,在被激活的当前编辑窗口中,用新选择的文件替换其内的旧文件并不打开一个新编辑窗口。如果系统找不到选择文件则创建一个新文件。

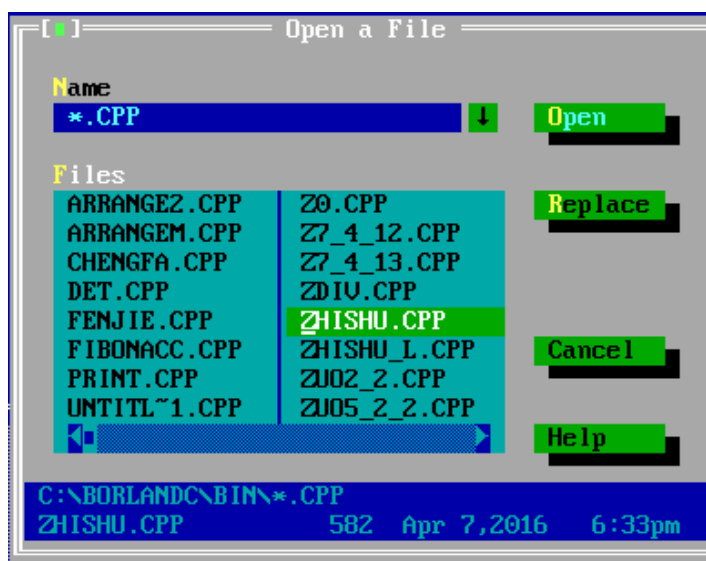


图 1.12 “Open (打开一个已存在的文件)”对话框

8. 编译、链接单个源文件的程序

选择 **Compile** 子菜单命令再按回车键,系统立即对当前编辑窗口内的源文件进行编译,在主屏幕中间将弹出标题为“Compiling (正在编译)”的窗口如图 1.13 所示,其内显示出在编译过程中所产生的编译错误(Compiler Error)和警告信息的个数,并提示按任意键即可进入 Message 窗口,如图 1.14 所示,其内将列出每个错误所在的文件目录路径和所在行号以及错误性质的简洁提示,例如,若将 ARRANGE2.CPP 文件中的第 1 条语句“#include <stdio.h>”去掉,则按 ISO/ANSI C++新标准,那么标准函数 printf()和 scanf()就将没有原型声明成为没有定义的标识符,在对应的编辑窗口内还用高亮度亮绿色框表明该错误所在行。往往一个错误会引起多行错误信息,因此一般是针对第 1 行错误来修改源程序再重新编译,如果原来的第 1 行错误消除了,再针对新的第 1 行错误来修改,再重新编译,直到没有编译错误为止,例如“C:\BORLANDC\BIN\”目录下的源程序 ARRANGE2.CPP,将“#include <stdio.h>”语句添加在文件的开头处即可消除掉上述三个编译错误,如图 1.15 所示。这说明编译成功,

并生成以源程序名 filename 为名字的可重定位文件 filename.obj（即采用源文件加扩展名“.obj”作为全文件名）。在此须提醒初学者，在每次修改源程序后，一定要存盘一次以确保修改后的内容保存下来，然后再进行新一轮的编译操作，也避免了编译系统是用没有修改前的内容在进行编译。通常，操作者都采用热键操作方式，即按“Alt+F9”键进行编译操作。

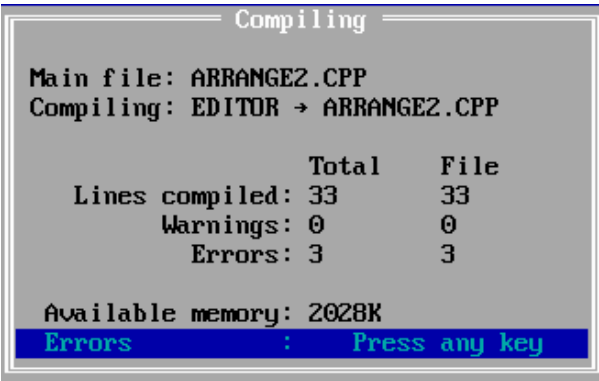


图 1.13 “Compiling（正在编译）” 窗口

接着进行链接操作，采用“Compile（主菜单）/Make（子菜单）”或“Compile（主菜单）/Link（子菜单）”菜单命令即可实现链接操作。通常，都采用热键 F9。在链接过程中的错误处理与编译时类同，也是在 Message 窗口内列出一个个链接错误以及所在的文件目录路径和错误性质提示信息。操作者首先要当链接成功后，在 Message 窗口上将出现如下信息：

Linking ..\..\BORLANDC\BIN\ARRANGE2.EXE

则说明已生成了可执行文件 filename.exe。

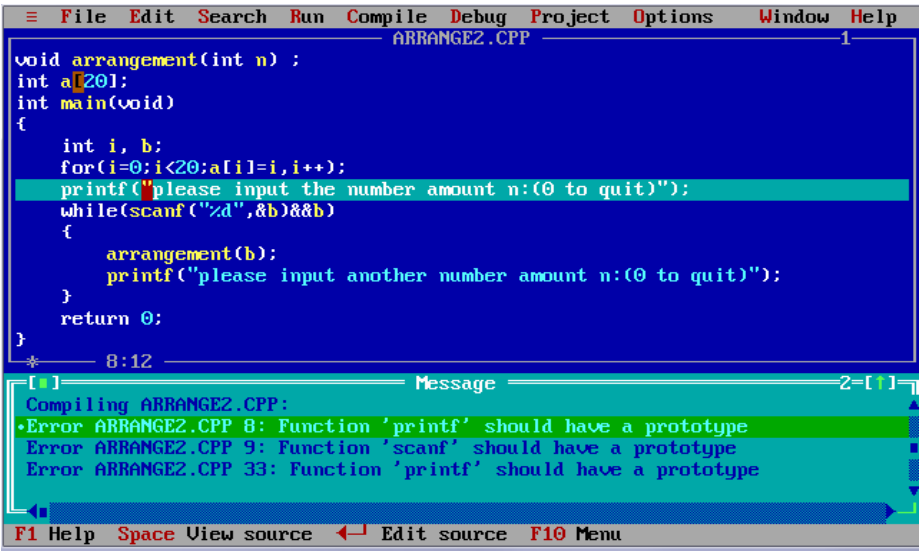


图 1.14 编译完成后的主屏幕

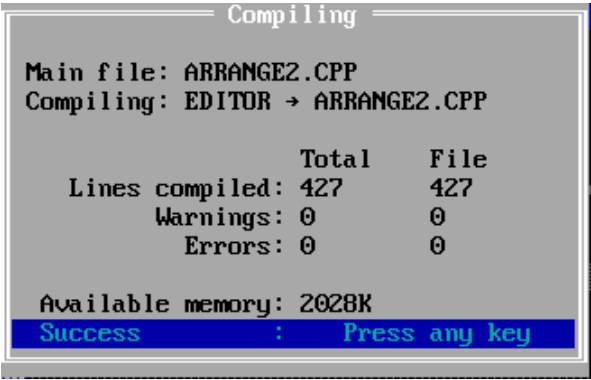
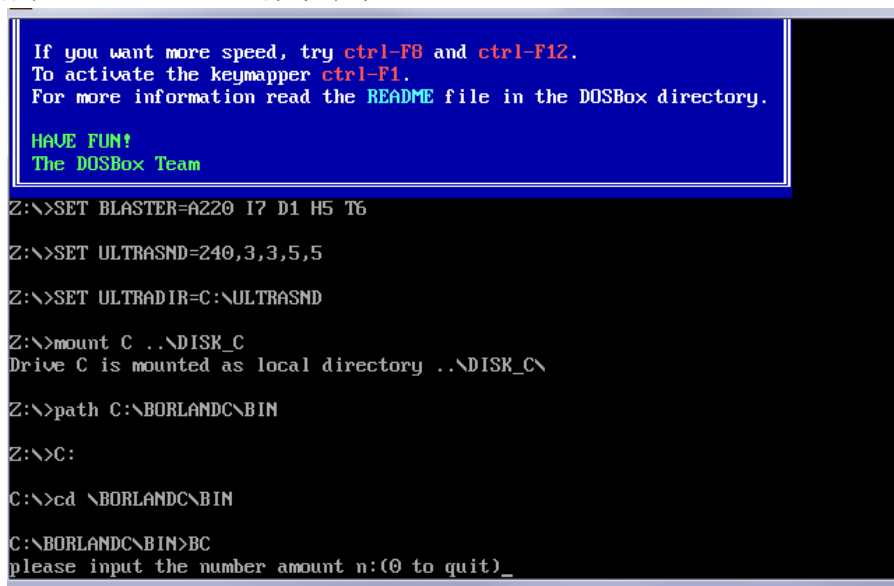


图 1.15 编译成功的窗口

9. 运行可执行文件

采用“Run（主菜单）/Run（子菜单）”菜单命令，则执行 filename.exe。源程序中凡是调用标准函数 printf() 的语句所得到的输出结果都将显示在如图 1.16 所示的“Output(输出)”窗口上。用“Window/Output”菜单命令可以打开该输出窗口，选择 Output 子菜单命令再按回车键，则弹出“Output（输出）”窗口如图 1.16 所示。通常是按热键“Alt+F5”切换到用户屏幕，再按任意键则切换回到主屏幕。对于含单个源文件的程序，每当调试完一个后若不再使用它所占用的编辑窗口，先把该窗口指定为当前窗口，再按热键“Alt+F3”关闭它，或者用“Window/Close”菜单命令。



```
If you want more speed, try ctrl-F8 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>SET ULTRASND=240,3,3,5,5

Z:\>SET ULTRADIR=C:\ULTRASND

Z:\>mount C ..\DISK_C
Drive C is mounted as local directory ..\DISK_C\

Z:\>path C:\BORLANDC\BIN

Z:\>C:

C:\>cd \BORLANDC\BIN

C:\BORLANDC\BIN>BC
please input the number amount n:(0 to quit)_
```

图 1.16 “Output（输出）”窗口

10. 帮助系统的使用

Help 主菜单项链接着 BC31 的帮助系统，在 BC31 集成开发环境中用如下操作即可方便地利用帮助系统。

1) 打开一个文件，在 BC31 集成开发环境的任何位置，按 F1 则弹出如图 1.17 所示的窗口。

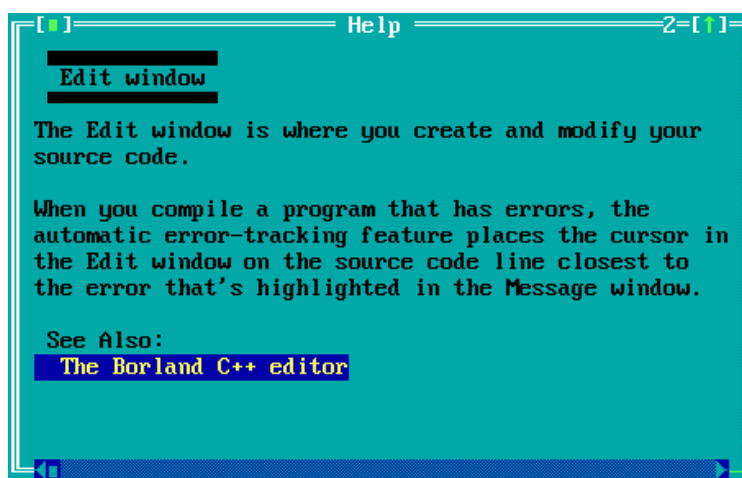


图 1.17 按 F1 弹出的窗口

上图中显示 BC31 集成编辑器使用方法的帮助信息，用光标键 ↑ 和 ↓ 移动到最下面的“**The Borland C++ editor**”项，该项将被一个亮蓝色方框覆盖表明选中它，再按回车键

即可得到更详细的帮助信息。

2) 在一个被激活的编辑窗口内，如果把光标停留在某个程序单词例如 C:\BORLANDC\BIN\ARRANFE2.CPP 文件中第 8 行“printf”下面，再按热键“Ctrl+F1”或者用“Help/Topic search”菜单命令（详见表 1.1），即按 F10 启动主菜单，用←和→光标键选择 Help 主菜单项再按回车键则弹出下拉子菜单，在其内用↑和↓光标键选择“Topic search”子菜单命令再按回车键，则弹出标题为 Help 的窗口如图 1.18 所示，其内显示有标准函数 printf() 的详细信息。

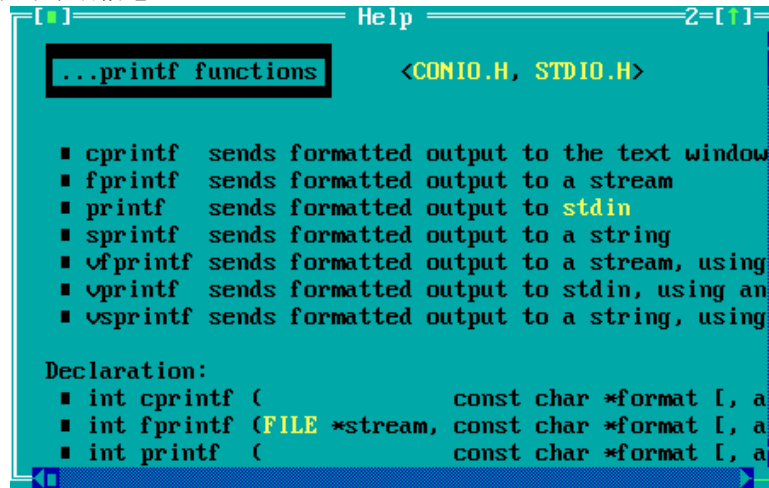


图 1.18 按 Ctrl+F1 弹出的窗口

3) 当弹出一个对话框时，用 Tab 键可把控制焦点（拥有光标的控件，像按钮、文本框和列表框等都称为控件，当光标停留在某个控件中如文本框则可以用键盘修改文本内容，列表框拥有光标就可以用←、→、↑和↓光标键选择指定项）在对话框内的所有控件间循环移动，每按一次移向下一个控件直到“Help”按钮再按回车键，则将弹出 Help 窗口提供相关的帮助信息，如图 1.18 所示的黄色字体，当光标在上面时，按下回车键，即可出来相应的 Help 提示。

4) 按热键“Alt+F1”或者用“Help/Previous topic”菜单命令，即按 F10 启动主菜单，用←和→光标键选择 Help 主菜单项再按回车键则弹出下拉子菜单，在其内用↑和↓光标键选择“Previous topic”子菜单命令再按回车键，则弹出上一次打开的帮助窗口。

5) 按热键“Shift+F1”或者用“Help/Index”菜单命令，即按 F10 启动主菜单，用←和→光标键选择 Help 主菜单项再按回车键则弹出下拉子菜单，在其内用↑和↓光标键选择“Index”子菜单命令再按回车键，则打开 Help Index 窗口，其内按英文字母排列顺序列出了整个帮助系统中的所有关键字和系统使用的标识符的帮助信息。例如，若编程者要查找一个开平方标准函数 sqrt()，可打开 Help Index 窗口，用键盘敲入函数名或前面几个字符甚至第 1 个字符 S。如图 1.20 所示，用光标键把光标移动到 sqrt 再按回车键则显示出如图 1.21 所示的 Help 窗口，其内显示有标准函数 sqrt() 调用它的详细帮助信息，如函数功能、形式参数、返回值类型和包含原型声明的头文件。

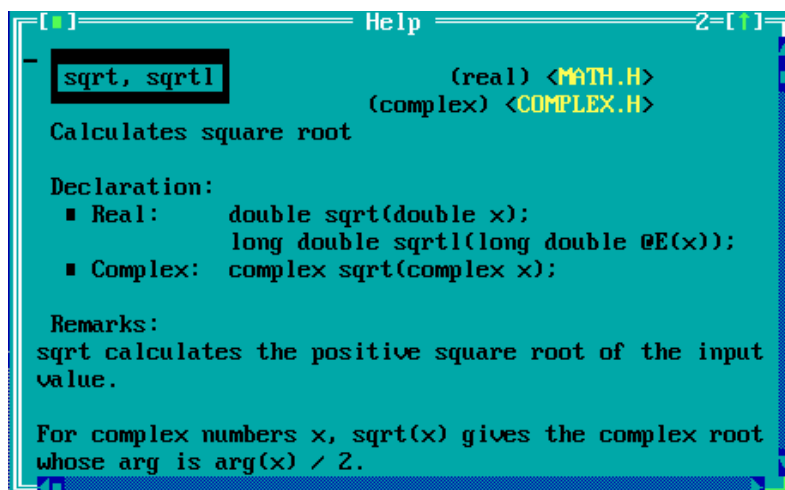


图 1.20 光标移动在 sqrt 上

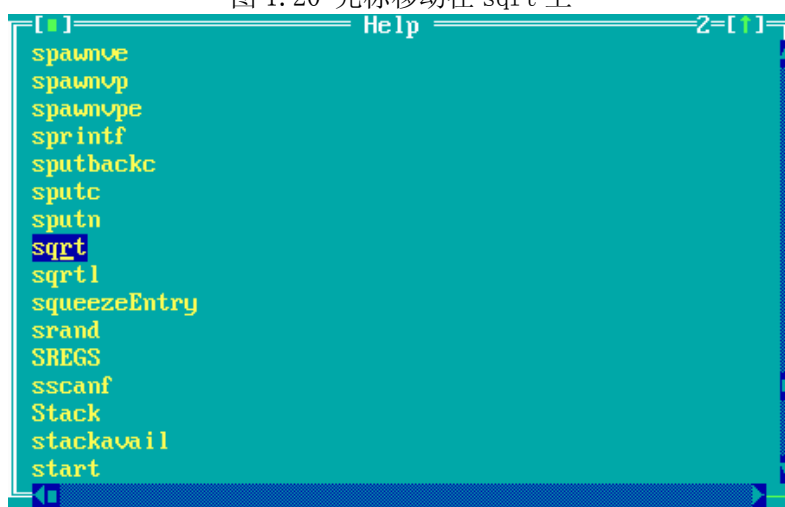


图 1.21 sqrt 函数的详细信息

6) BC31 备有一整套“用户编程指南”的帮助信息，当编程者需要查找时用“Help/Contents”菜单命令，即按 F10 启动主菜单，用←和→ 光标键选择 Help 主菜单项再按回车键则弹出下拉子菜单，在其内用↑ 和 ↓ 光标键选择“Contents”子菜单命令再按回车键，弹出如图 1.22 所示的 Help 窗口，其内列出了 BC31 帮助系统的主题目录，操作者可用 Tab 键或者光标键把光标移动到某个主题词上，图 1.22 中是移动到“Borland C++ Language”上，该主题词立即被高蓝色框覆盖，接着按回车键（或者用鼠标左键双击该主题词）将弹出该主题项的子主题列表如图 1.23 所示，如果操作者选中子主题“Keywords”再按回车键（或者用鼠标左键双击该子主题词）将弹出如图 1.24 所示的窗口，列出了 BC31 系统所预先定义的关键字，操作者可进一步选择其中某个关键字，再按回车键（或者用鼠标左键双击该关键字）即可得到它的详细信息。由此可见，凡是黄色标题字符串均具有超文本链接功能，可进一步查找更详细信息。

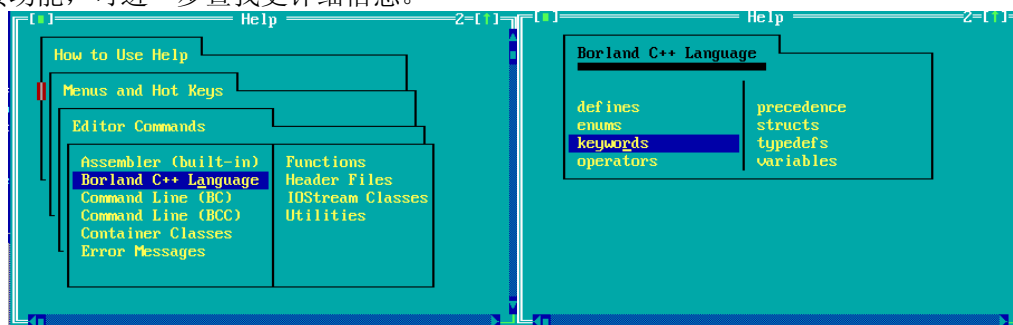


图 1.22 帮助系统的主题目录

图 1.23 子主题目录之一

7) 编程者在当前编辑窗口上编写程序时, 随时可以查找到所要的帮助信息, 只需把光标移动到某个单词上, 该单词既可以是 BC31 系统预先定义的关键字, 也可以是标准函数名、结构名和运算符等, 然后用 “Help/Topic search” 菜单命令 (参见表 1.1), 即按 F10 启动主菜单, 用←和→ 光标键选择 Help 主菜单项再按回车键则弹出下拉子菜单, 在其内用↑和 ↓ 光标键选择 “Topic search” 子菜单命令再按回车键, 将弹出一个窗口, 其内显示出光标所在单词的语言帮助信息。

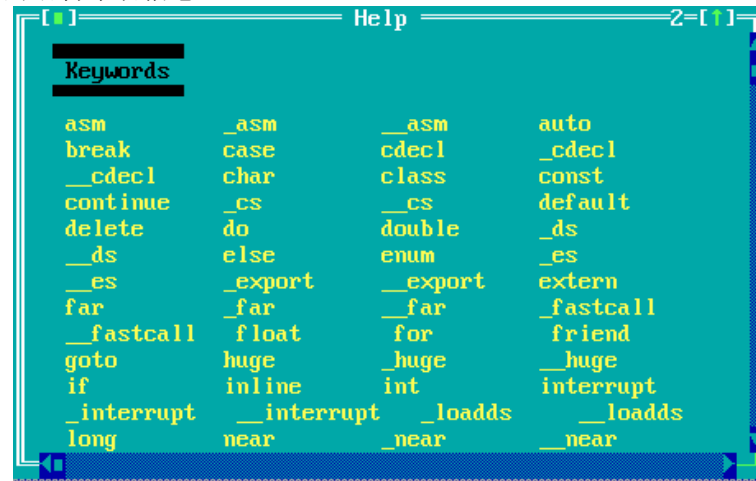


图 1.24 Keywords 子主题

2.2 程序调试

掌握调试器 Debugger 的使用方法是开发程序的最重要的基本技能。当编译、链接顺利完成（没有通过编译、链接操作的程序不要使用调试器）后，若发现某项的结果有误，可启动调试器 Debugger，利用调试器“单步（热键 F8）”或“跟踪（热键 F7）”的功能，按顺序逐条执行语句，在“监视(Watch)”窗口上观察所监视对象的变化，以确定究竟在执行哪条语句时发生了错误。

1) 在使用集成调试器前，必须进行如下检查和设置使系统进入具有调试信息的状态：

①用“Options/Compiler/Advanced code generation”菜单命令，将弹出一个窗口如图 1.25 所示，在右边标题为 Options 的复选框内必须选中“Debug Info in OBJs”项。

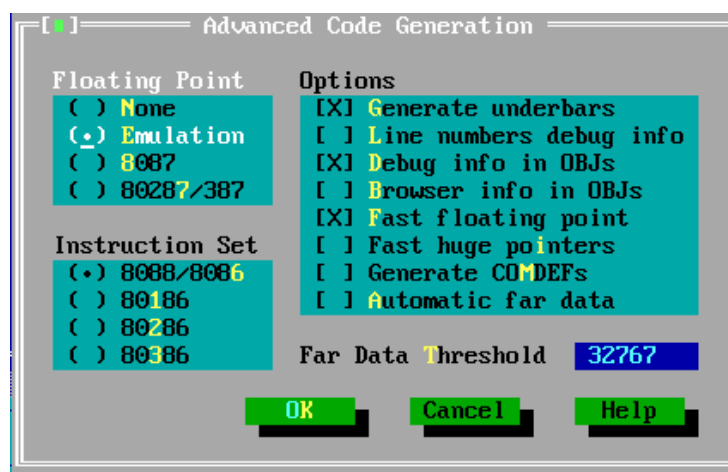


图 1.25 选中“Debug Info in OBJs”项

②用“Options/Debugger”菜单命令，将弹出一个窗口如图 1.26 所示，必须把左边标题为“Source Debugger”的单选框设置成 on。

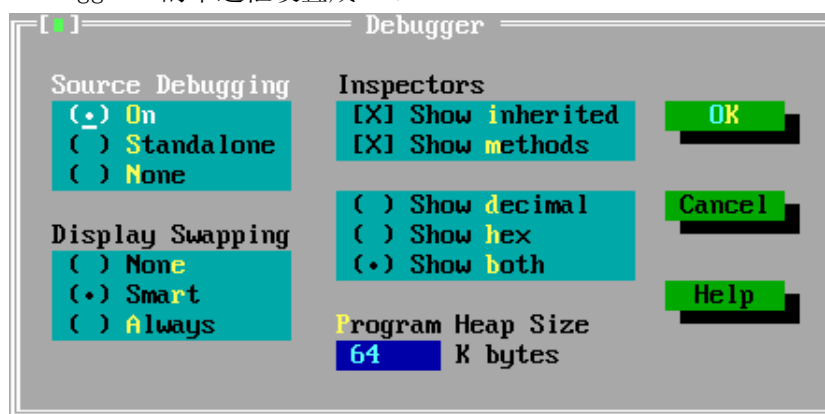


图 1.26 “Source Debugging”设置为 on

2) 进入调试状态

选用 Run 主菜单中的“Go to Cursor（或者按热键 F4）”、“Trace Into（或者按热键 F7）”和“Step Over（或者按热键 F8）”等 3 个子菜单命令之一，都可以进入调试状态。初学者最好选用第 1 个命令，现在以 ARRANGE2.CPP 为例来说明调试器的使用。当编译、链接顺利完成后，应先将光标移动到一个函数头或者函数体内的某个执行语句。如图 1.27 所示，当系统进入到调试状态后，可以用“Debug/Inspect...”菜单命令，将弹出一个 Data Inspect 对话框，在其标题为 Inspect 文本框内敲入想要查看的变量名或表达式，按回车键或者用鼠标左键点击“OK”按钮即可打开一个观察窗口。例如图 1.27 中打开了 3 个观察窗

口查看变量 i, b 和 a[] 的值。另外还可以使用快捷操作方式, 即把光标移动到某个要查看的变量名下面, 按热键 “Alt+F4” 立即打开一个观察窗口查看该变量的信息。顺便指出, 按热键 “Alt+F3” 即可关闭被激活的观察窗口。

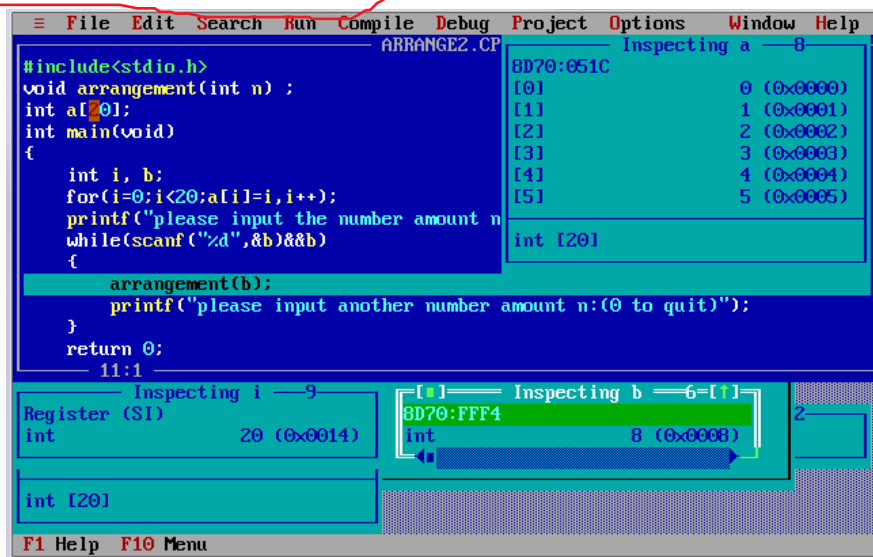


图 1.27 BC31 集成开发环境的调试状态

3) 单步执行

使用 Run 主菜单中的 “Trace Into (或者按热键 F7)” 和 “Step Over (或者按热键 F8)” 这两个子菜单命令可以控制编辑窗口上的源程序一条一条语句顺序执行, 通常使用热键 F7 或 F8, 即每按一次热键 F7 或 F8 执行一条 “执行语句”, 从各个观察窗口查看这些变量或表达式的变化情况。热键 F7 和 F8 的区别是前者可一直跟踪到被调用函数体内称为 “跟踪” 操作, 后者不跟踪到被调用函数体内即只执行函数调用跳过被调用函数体称为 “单步” 操作。初学者应特别注意, 对于系统提供的标准函数调用语句由于它没有函数体内一条条执行语句的源代码无法跟踪而不能按热键 F7 只能按热键 F8, 否则将跟踪到以反汇编形式显示的机器码指令而难于继续调试; 对于编程者自行定义的函数若需要检查函数体内的情况才按热键 F7。如图 1.27 所示, 一般观察窗口其第 1 行显示变量在内存中的存放地址, 该行以下的行显示观察项的信息, 左边显示数据类型名、数组下标或结构类型的成员名, 右边显示变量所具有的数值。序号为 8 的观察窗口是数组观察窗口 (Array Inspector Window) 用来监视数组 a[], 其第 1 行显示数组在内存中的存放地址, 以下行用相同的格式显示该数组的所有元素, 即左边显示数组下标如 [0]、[1]、[2]、[3] 等, 右边显示对应数组元素值分别为 0、1、2 和 3 等。窗口序号为 6 和 9 的观察窗口上显示的是变量 i 和 b 的值。

用 “Debug/Watches /Add watch (添加监视项)” 菜单命令, 在屏幕中间将弹出一个如图 1.28 所示、标题为 “Add watch” 的对话框, 并提供一个输入监视表达式 (Watch Expression) 的文本框, 操作者可用键盘直接输入要监视的表达式。若在 watch (监视) 窗口内, 可直接按 Ins 键即可弹出一个 “Add watch” 对话框, 在其内的文本框中输入新的监视表达式插入到 watch (监视) 窗口内。

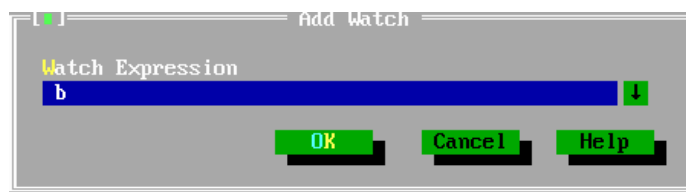


图 1.28 “Add watch” 窗口

当 watch 窗口被激活时, 采用 “Debug/Watches/ Delete watch (删除监视项)” 菜单命令, 可删除被一个高亮度绿色框覆盖的当前监视表达式。采用 “Debug/Watches/Remove all watch (删除所有监视项)” 菜单命令, 可删除所有的监视表达式。采用 “Debug/Watches/Edit watch (编辑监视项)” 菜单命令, 操作者可用键盘修改该表达式再按回车键, 则调试器以修改后的表达式替代原来的表达式。

4) 设置断点

对于大型程序的调试经常需要设置断点, 所谓断点就是程序启动后运行到源程序的一条语句暂时停止的位置点, 断点可以让操作者利用上述观察和监视手段查看程序运行情况以及变量和对象的变化情况。用 “Debug/Breakpoint” 菜单命令可设置/清除一个无条件或有条件的断点, 其内的断点列表框 (Breakpoint List) 内列出了所有已设置的断点, 包括每个断点的行号、中断条件和执行次数等信息, 其中被亮绿色框覆盖的横行称为 “当前行”。该对话框内共包含 OK、Edit、Delete、View、At、Cancel 和 Help 等 7 个按钮, 每个按钮上的字符串中都有一个高亮度的黄色字母, 它表示快捷操作方式的字母键, 如 OK 按钮的 K、Edit 按钮的 E、Delete 按钮的 D、View 按钮的 V 和 At 按钮的 A 等。例如, 操作者若要设置一个程序断点可用 Edit 按钮的 E 键, 即按 E 键就弹出如图 1.29 所示、标题是 “Breakpoint Modify/New (修改和添加断点)” 的对话框, 操作者根据其标题提示输入如下 4 种信息:

- 在标题为 “Condition” 的文本框内输入一个表达式指定中断条件, 若无约束条件可跳过此项设置。
- 在标题为 “Pass Count” 的文本框内输入一个正整数值, 指定该断点在中断前被执行的次数, 不输入该值时系统自动设置为 0, 则每次遇到该断点都将中断。可跳过此项设置。
- 在标题为 “File Name” 的文本框内输入断点所在的、带磁盘号和目录路径的文件名, 如图 1.29 中的 “C:\BORLANDC\BIN\ARRANGE2.CPP”, 这是操作者必须要指定的设置项。
- 在标题为 “Line Number” 的文本框内输入断点所在的行号, 如图 1.29 中输入 11, 这也是操作者必须要指定的设置项。

当以上设置做完后, 按 OK 键加以确认则系统根据断点列表框的内容 (如图 1.30 所示的断点列表框) 来设置断点, 并在源程序中用高亮度红色方框覆盖每个断点。

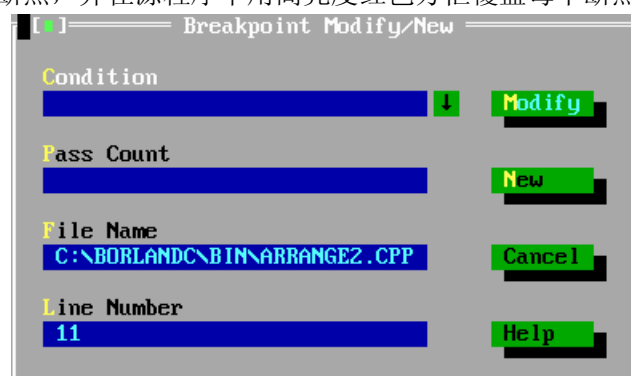


图 1.29 “修改和添加断点” 对话框

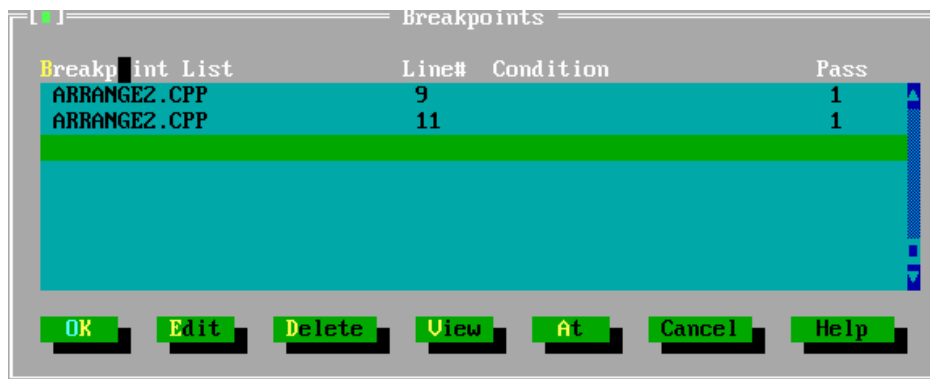


图 1.30 “Breakpoint” 对话框

当操作者设置了第 9 和 11 行 2 个断点后再按回车键或者用鼠标左键点击 OK 键，则将得到如图 1.31 所示的画面，每个断点都被一个高亮度红色方框覆盖，并把光标停留在 main() 函数头，按热键“Ctrl+F9”启动程序运行到第 1 个断点即第 7 行中断停止，该行被亮绿色方框覆盖以表明程序运行的中断点。此时，操作者可以使用上述观察和监视方法查看被监视变量或对象的信息，确定程序运行时的错误以便对症下药地修改。

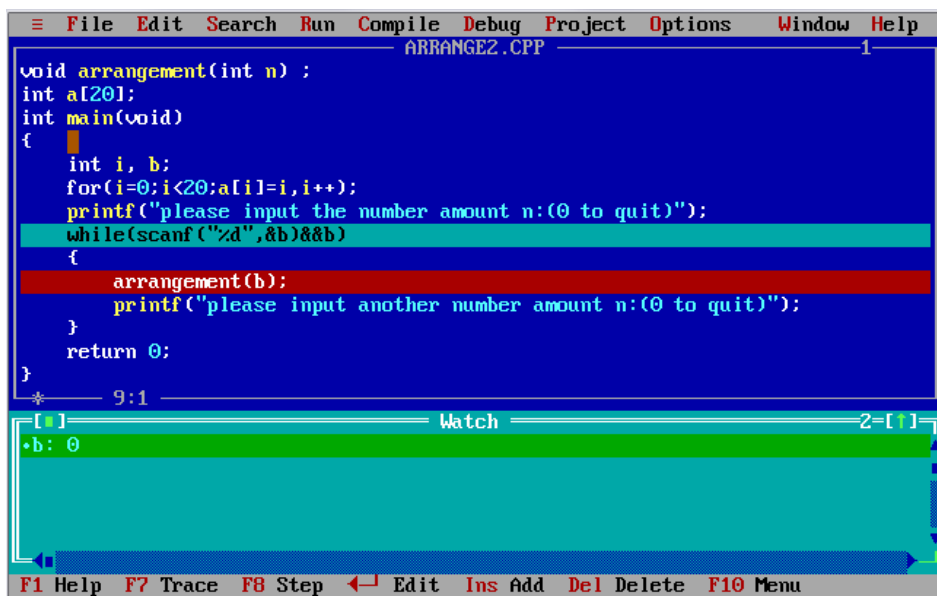


图 1.31 设置 2 个断点后并运行到源程序的第 9 行

在如图 1.29 所示的“Breakpoint”对话框中，按 Delete 按钮或热键 D 则删除断点列表框中的当前行断点（被亮绿色方框覆盖的行）；按 View 按钮或热键 V 则系统切换到编辑窗口，光标停留在断点列表框中的当前行断点所对应的、在编辑窗口中所对应的行提供给编程者查看。只有在程序运行时 At 按钮才被激活，按 At 按钮或热键 A 则弹出如图 1.32 所示的“Breakpoint At Function”对话框，在标题为“Symbol Name”的文本框内用键盘输入一个函数名（不带一对圆括号）再按回车键或者用鼠标左键点击 OK 键，则将该函数头设置为一个新断点，如图 1.33。

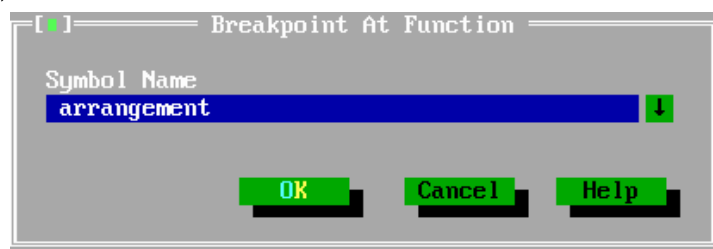
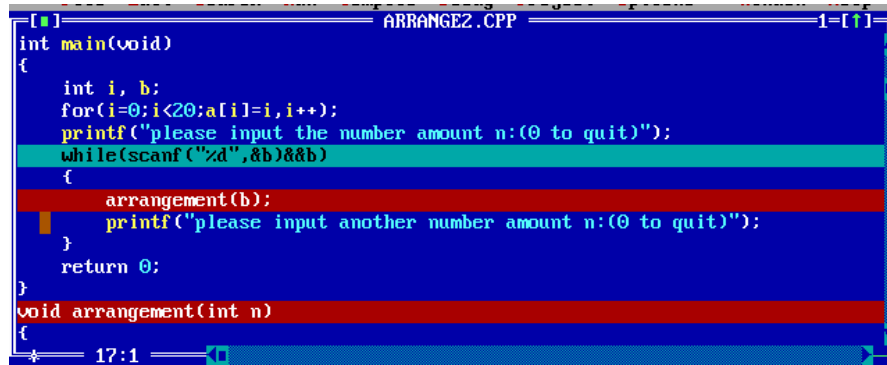


图 1.32 “Breakpoint At Function” 对话框

用“Debug / Breakpoint”菜单命令可设置/清除一个无约束条件断点，则将光标所在行设置成一个无约束条件断点。如果再做一次这样的操作则将光标所在行已设置的断点清除掉。用“Run（主菜单）/Run（子菜单）”菜单命令，或者按热键 F4（经常使用的操作方式）则程序继续往下执行，直到下一个断点处停止，该位置的一条语句被亮绿色方框所覆盖。若此位置以后没有断点，再按 F4 程序就一直运行到终点结束。



```
int main(void)
{
    int i, b;
    for(i=0; i<20; a[i]=i, i++);
    printf("please input the number amount n:(0 to quit)");
    while(scanf("%d", &b) && b)
    {
        arrangement(b);
        printf("please input another number amount n:(0 to quit)");
    }
    return 0;
}
void arrangement(int n)
{
    ...
}
```

图 1.33 通过 At 按钮设置新的断点

2.3 建立工程文件

对于由多个源文件组成的源程序,可采用建立工程文件的方法编辑、编译、链接和调试,这也是作为一个软件开发小组的负责人必须掌握的基本操作方法,当他从各个小组成员手上收集到已开发成功的、以源文件形式保存的程序模块后,建立一个工程项目文件,把这些源文件都添加到该工程项目中,经编译、链接操作最终生成一个扩展名为“.exe”可执行文件,即通常所称的应用程序。

1. 建立一个工程项目

在 BC31 集成开发环境 IDE (Integrated Development Environment) 中, Project 主菜单包含工程项目管理的所有命令。首先采用“Project/Open Project”菜单命令在 IDE 上建立一个工程项目,则弹出如图 1.34 所示、标题为“Open Project File”的对话框,对于新创建的工程项目应在标题为“Open Project File”的文本框内键入所启用工程项目名(包括磁盘号和目录路径名),例如 C:\BORLANDC\BIN\NEWPRO.PRJ,再按回车键,则打开如图 1.35 所示(已经添加了一个程序进去),新建的标题为“Project:NEWPRO”的工程项目窗口,其内目前是空的不包含任何文件。

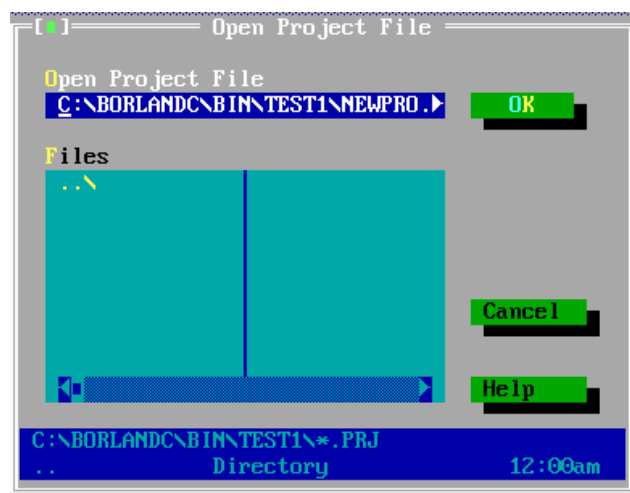


图 1.34 “Open Project File”对话框

2. 向工程文件项目中添加文件

接着,应向这个内容为空的工程项目中添加文件,既可以添加扩展名为“.cpp”的源文件,不仅可添加源文件,也可以是扩展名为“.h”的头文件,或者是 filename.obj 文件等。用“Project (主菜单) /Add item... (子菜单)”菜单命令,将弹出一个标题为“Add to Project List”的对话框如图 1.28 所示,它与图 1.12 所示的“Open a File (打开文件)”对话框的外观和使用方法都十分相似。如果所添加文件是过去已经开发好的文件,用光标键把亮绿色方框移动到要选择的文件如“NONAME10.CPP”上,再按回车键,则系统就把该源文件如“NONAME10.CPP”添加到工程项目中,并关闭“Add to Project List”的对话框,激活标题为“Project:NEWPRO”的工程项目窗口,其内显示出刚添加的“NONAME10.CPP”文件如图 1.35 所示。如果要添加的是已经存在的头文件,则在标题为 Name 的文本框内键入“磁盘号:\目录路径名*.H”即可找到该目录路径下的所有头文件,再用同样的操作方法添加到该工程项目中。

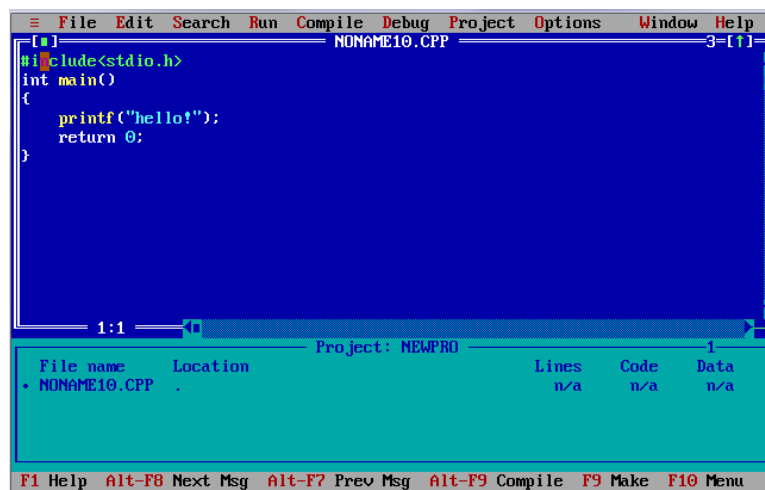


图 1.35 “Project:NEWPRO” 工程项目窗口

若要添加的是一个新建的文件，只需在标题为 Name 的文本框内键入该新文件所存放的磁盘号、目录路径名和带扩展名的新文件名，再按回车键，则系统将创建一个新文件存放在指定的位置上，与此同时，系统将打开一个新的编辑窗口作为当前窗口，操作者可在其内键入新文件的具体内容，完成后按热键 F2 保存该新文件。顺便指出，图 1.366 中的 Done 按钮是用来关闭该窗口的，因对话框被激活时是无法切换到其他窗口，只有用鼠标左键点击 Done 按钮关闭该对话框后才能激活其他窗口。

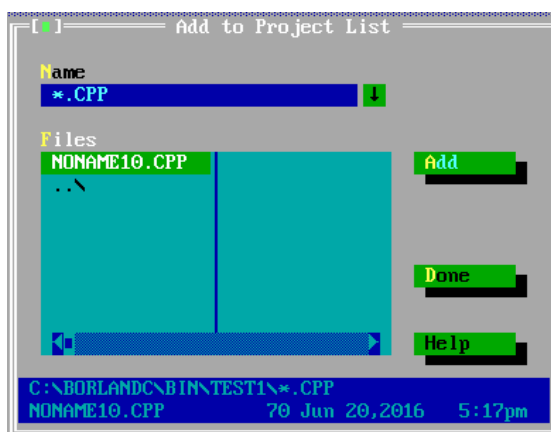


图 1.36 “Add to Project List” 对话框

3. 编译、链接和运行工程项目

1) 当打开一个工程项目文件如 NEWPRO.PRJ，且标题为“Project:NEWPRO”的工程项目窗口处于激活状态如图 1.35 所示，用“Compile/Build all”菜单命令，则系统首先对该工程项目中所包含的所有文件进行编译，然后再把生成的所有目标代码文件进行链接。若有错误，将显示在错误信息窗口上，便于编程者修改程序时阅读，直到没有错误信息为止。这时便生成了一个可执行文件 NEWPRO.exe，其文件名就是项目名。

2) 运行工程文件的方法与运行单个源程序基本类似。首先应激活标题为“Project:NEWPRO”的工程项目窗口，然后用“Run（主菜单）/Run（子菜单）”菜单命令，或者按热键“Ctrl+F9”，则执行 NEWPRO.exe 应用程序。

3) 接着可使用调试器对整个工程项目的源文件进行调试，其操作方法与单个源文件的基本类同，只是在操作者使用热键 F7 跟踪到另一个源文件的被调用函数体内时，系统会自动切换到该文件所在的窗口，若该文件还没有打开，系统会自动打开该源文件并显示在

编辑窗口上，跟踪指示位置也自动地停在该函数的第 1 条语句，操作者即可按热键 F7 或 F8 在该函数体内跟踪操作。

4) 在调试工程项目的程序时，可能会打开数量较多的窗口。但被打开的窗口序号只有 1~9 共计 9 个，当被打开窗口数量超过 9 个时将出现无序号窗口。因此无法采用按压“Alt+窗口序号”的操作方式，此时可按“Alt+0”键则打开标题为“Window List (窗口列表框)”如图 1.37 所示，其内列出了所有被打开的窗口及其序号，且还能列出曾经被打开现已关闭窗口的历史记录。

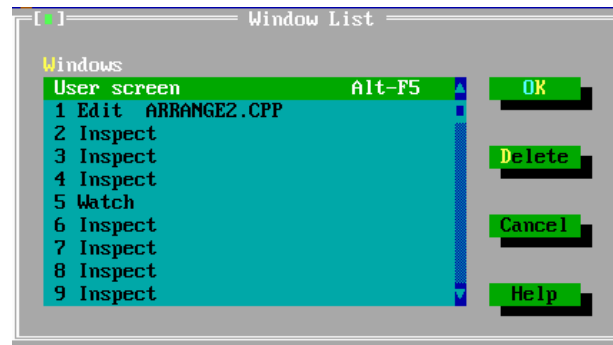


图 1.37 “Window List” 列表框

第二部分 C 语言实验指导

实验一 简单程序设计实验

1 实验目的

- 1) 掌握 C 语言的基本语法单位;
- 2) 掌握 C 语言的数据类型, 运算符和表达式, 以及各类数值类型数据间的混合运算;
- 3) 学会使用格式化输入/输出函数;
- 4) 熟悉 C 程序的编辑、编译、连接和运行的过程, 理解 C 语言程序的顺序结构。

2 实验相关知识

1) 数据类型

- a) 常量分为整型常量、浮点常量、字符常量和字符串常量。
- b) 变量使用时需要遵守命名规则, 必须是以英文字母或下划线开头的, 由字母、数字和下划线组成的字符序列; 不能与 C 语言的的关键字重名; C 语言对变量名的大小写敏感。
- c) 变量的基本数据类型有字符型、整型、单精度浮点类型、双精度浮点类型及指针型, 由基本数据类型的数据长度不同及有无符号又可以派生出多种类型。

2) 运算符和表达式

- a) C 语言的运算符按其所在表达式中参与运算的操作数的数目来分, 可分为单目运算符、双目运算符和三目运算符; 按照其功能来又可分为算术运算符、赋值运算符、关系运算符、逻辑运算符、自增自减运算符、条件运算符、逗号运算符等等。
- b) C 语言中的表达式根据运算符的种类可以分为: 算术表达式、关系表达式、逻辑表达式、赋值表达式、条件表达式、逗号表达式以及混合表达式等, C 程序中任何一个表达式表示的都是具有某个数据类型的一个值。

3) 输入/输出函数

- a) 输出函数 `printf()` 的一般使用形式:

`printf("输出格式",输出项系列);`

输出格式: 由两部分组成, 一个是格式说明, 由“%”和格式字符组成的, 如 `%d`, `%c`, `%f`, 它的作用是把输出数据转换为指定格式输出, 格式的说明总是由“%”字符开始的。另一个是普通字符, 需要原样输出的字符, 或者一些有特殊含义的字符, 如 `\n`, `\t`。

eg: `printf("a=%d,b=%f,c=%c",a,b,c);`

- b) 输入函数 `scanf()` 的一般使用形式:

`scanf("输入格式",输入项系列);`

此处的输入格式与输出函数的格式相差无几, 主要是要注意输入项系列必须为地址量, 变量的地址用地址符 (&) 加变量名构成。

eg: `scanf("%d",&a);`

3 实验范例

1) 新建一个“Hello World!”的程序, 输入如下代码, 然后编译、连接、运行程序, 并查看结果, 将程序中 3 个 `printf` 函数中最后的“`\n`”去掉后, 再编译、连接、运行程序, 看看结果如何。

- ① 参考源程序:

`#include<stdio.h>`

```

void main()
{
    printf("*****\n");
    printf("Hello  World!\n");
    printf("*****\n");
}

```

② 运行结果:



```

*****
Hello  World!
*****
请按任意键继续. . .

```

2) 将如下代码键入程序，比较下列格式化输入与输出，看看有什么不同之处。

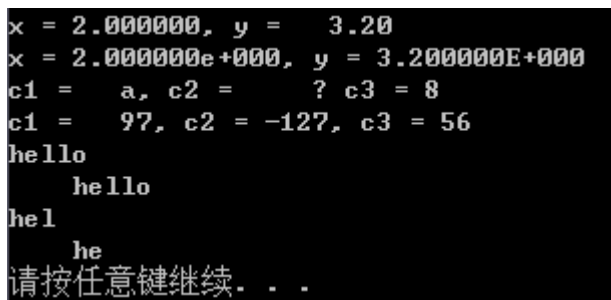
① 参考源程序:

```

#include<stdio.h>
int main()
{
    float x, y;
    char c1, c2, c3;
    x=2;
    y = 1.2 + 5 / 2;
    printf("x = %f, y = %6.2f\n", x, y);
    printf("x = %e, y = %E\n", x, y);
    c1='a';
    c2 = c1 + 32;
    c3 = '0' + 8;
    printf("c1 = % 3c, c2 = % 5c, c3 = %-5c \n", c1, c2, c3);
    printf("c1 = % 4d, c2 = %-4d, c3 = %d \n", c1, c2, c3);
    printf("%s\n % 8s\n%.3s\n%6.2s\n", "hello", "hello", "hello", "hello");
    return 0;
}

```

② 运行结果:



```

x = 2.000000, y = 3.20
x = 2.000000e+000, y = 3.200000E+000
c1 =  a, c2 =  ? c3 = 8
c1 =  97, c2 = -127, c3 = 56
hello
    hello
hel
    he
请按任意键继续. . .

```

3) 输入圆的半径，输出圆的周长和面积。

① 参考源程序:

```

#include<stdio.h>
void main()
{

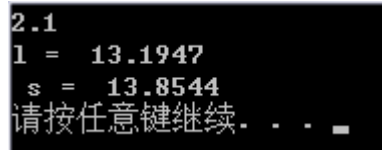
```

```

float r, l, s;
scanf ("%f", &r);
l = 2 * 3.14159*r;
s = 3.14159*r*r;
printf("l = %8.4f\n", l);//注意输出的格式
printf("s = %8.4f\n", s);
}

```

② 运行结果：



```

2.1
l = 13.1947
s = 13.8544
请按任意键继续...

```

4) 关于运算符和表达式的程序。

① 参考源程序：

```

#include<stdio.h>
void main()
{
    int a = 4, b = 5, c = 0, d;
    d = !a&&!b || !c;//逻辑与“&&”优先级高于逻辑或“||”
    printf("%d\n", d);
    printf("%d\n", a++);
    printf("%d\n", ++a);
}

```

② 运行结果：



```

1
4
6
请按任意键继续...

```

4 实验内容

1) 程序分析题。（写出下列程序的输出结果，然后上机确认结果）

①程序 1：

```

#include <stdio.h>
void main( )
{
    int a = 1, b = 1;
    a += b += 1;
    {
        int a = 10, b = 10;
        a += b += 10;
        printf("b = %d\t", b);
    }
    a *= a *= b * 10;
    printf ("a = %2d\n", a);
}

```

②程序 2：

```

#include <stdio.h>

```

```

void main( )
{
    int a = 1, b = 2, c = 3;
    ++a;
    b += ++c;
    {
        int b = 4, c = 5;
        c = b * c;
        a += b += c;
        printf("a1 = %d , b1 = %d\n", a, b, c);
    }
    printf("a2 = %d , b2 = %d\n", a, b, c);
}

```

③程序 3:

```

#include <stdio.h>
void main( )
{
    char c;
    printf("Input print_char : ");
    scanf("%c", &c);
    printf("%4c\t%c\n", c, c);
    printf("%2c\t%c\t%3c\t%c\n", c, c, c, c);
    printf("%c\t%c\t%5c\t%c\n", c, c, c, c);
    printf("%c\t%c\t%5c\t%c\n", c, c, c, c);
    printf("%2c\t%5c\n", c, c);
    printf("%3c\t%c\n", c, c);
    printf("%2c\t%5c\n", c, c);
    printf("%c\t%c\t%5c\t%c\n", c, c, c, c);
    printf("%c\t%c\t%5c\t%c\n", c, c, c, c);
    printf("%2c\t%c\t%3c\t%c\n", c, c, c, c);
    printf("%4c\t%c\n", c, c);
}

```

2) 先计算出表达式的值，然后编写程序验证。(各表达式之间相互独立)

a) 已知 unsigned int x=015,y=0x2b; 求解: x|y; x^y; x&y; ~x+~y; x<<3; y>>4;

b) 已知 int i=10,i=5; 求解: ++i-j--; i=i*=j; i=3/2*(j=3-2); ~i^j; i&j|1; i+i&0xff;

c) 已知 int a=5,b=3; 求解下列各表达式的值以及 a 和 b 的值:

①!a && b ++; ②a || b + 4 && a * b; ③a = 1,b = 2,(a > b) ? ++a: ++b;

④++b, a = 10, a + 5; ⑤a += b %= a + b; ⑥a != b > 2 <= a + 1;

d) 计算下列表达式的值，并指出结果值的类型，以及变量 x、y 最后的值;

①3+7%4-1;

②已知 int x=24, y=3;

x++/--y x&y x&&y x|y x||y

x>>=y-1 y<<=3 x^y ~x+~y

③已知 int x=0, y=1;

x!=y<=2<x (x=y)?x++:y-- (x==y)?x++:y--

x-=y*=x+3 x=2,y=x*++y

3) 编写一个完整的可运行源程序，要求从键盘输入 a, b, c, d 这四个整数值，计算表达式 (a+b-c) * d, 并显示计算结果。

4) 编写一个程序：从键盘输入整型变量 a 的值，计算 b=a++和 c=++a 这两个表达式，并显示结果，观察自增运算符的使用。

5) 输入一个正整数，将其逆序输出。例如，输入 1234，输出 4321。

6) 输入 2 个整数，求两数的平方和并输出。

- 7) 任意输入一个大写字母，输出其小写字母。
- 8) 用*输出字母 E 的图案。

实验二 选择程序设计实验

1 实验目的

- 1) 学会 C 语言程序设计的结构框架以及一些编程规律方法;
- 2) 了解结构化程序设计的基本思想, 加深对顺序结构, 选择结构和循环结构的设计方法的理解;
- 3) 掌握 `if...else...` 的嵌套和 `switch...case...` 多分支语句控制流程语句, 并能通过嵌套编写出各种复杂的程序。

2 实验相关知识

1) C 语言程序设计语句

- a) 说明语句是对程序中所使用的各种类型变量及属性进行说明, 其格式为:

`<存储类型> 数据类型 变量名列表;`

- b) 执行语句包含表达式语句、复合语句、流程控制语句、辅助控制语句四大类。

2) 流程控制语句

- a) `if...else` 语句

`if` 语句根据给定的条件表达式的值 (0 或非 0) 进行判断, 决定执行两条分支中的哪一条, 有时候会包含多层嵌套的分支。

`if` 语句的一般形式为:

```
if(表达式)
{
    语句 1;
}
else
{
    语句 2;
}
```

- b) `switch` 语句

从表达式值等于某个 `case` 语句后的值开始, 它下方的所有语句都会一直运行, 直到遇到一个 `break` 为止。假如任何一个 `case` 语句的值都不等于表达式的值, 就运行可选标签 `default` 之下的语句。

`switch` 语句的一般形式为:

```
switch(表达式)
{
    case 判断值 1:语句组 1;
    break;
    case 判断值 2:语句组 2;
    break;
    .....
    case 判断值 n:语句组 n;
    break;
    default:语句组;
    break;
```


}

3 实验范例

1) 求出含有两个实根 ($b^2-4ac \geq 0$) 的一元二次方程 $ax^2+bx+c=0$ 的解。

① 程序分析:

定义变量: a,b,c,x1,x2,delta

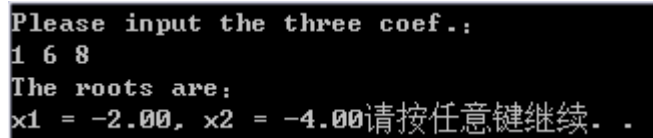
输入 a,b,c

计算 $\text{delta} = b^2 - 4ac$ (设 $\text{delta} \geq 0$), $x1 = (-b + \sqrt{\text{delta}}) / (2 * a)$, $x2 = (-b - \sqrt{\text{delta}}) / (2 * a)$

② 参考源程序:

```
#include "stdio.h"
#include "math.h"
void main()
{
    float a, b, c, x1, x2;
    float delta;
    printf("Please input the three coef.: \n");
    scanf ("%f%f%f", &a, &b, &c);
    delta = b*b - 4 * a*c;
    x1 = (-b + sqrt(delta)) / (2 * a);
    x2 = (-b - sqrt(delta)) / (2 * a);
    printf("The roots are: \nx1 = %4.2f, x2 = %4.2f", x1, x2);
}
```

③ 运行结果:



```
Please input the three coef.:
1 6 8
The roots are:
x1 = -2.00, x2 = -4.00请按任意键继续. . .
```

2) 给出下列分段函数的计算程序代码。根据输入的 x 的值, 计算出 Y 的值, 并输出。

$$Y = \begin{cases} (x+5)^2 + 3x, & x > 0 \\ 0, & x = 0 \\ (x-5)^2 - 3x, & x < 0 \end{cases}$$

① 程序分析:

对于分段函数, 运用 if-else 语句, 即可解决。

② 参考源程序:

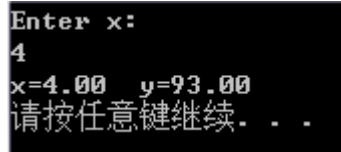
```
#include "stdio.h"
void main()
{
    float x,y;
    printf("Enter x:\n");
    scanf ("%f",&x);
    if(x>0)
        y=(x+5)*(x+5)+3*x;
    else if(x==0)
        y=0;
}
```

```

else
    y=(x-5)*(x-5)-3*x;
printf("x=%.2f  y=%.2f\n",x,y);
}

```

③ 运行结果:



```

Enter x:
4
x=4.00 y=93.00
请按任意键继续. . .

```

3) 求解简单表达式。输入一个形式如“操作数 运算符 操作数”的四则运算表达式，输出运算结果。如：输入：2+45；显示：2+45=47。

① 程序分析:

由于运算符是不确定的，故有+，-，*，/四种选择，可采用 switch... case...语句和 if...else if..else 语句写程序

② 参考源程序:

a.第一种方法:

```

#include "stdio.h"
void main()
{
    char oper;
    float operand1, operand2;
    printf("Type in an expression : \n");
    scanf("%f%c%f", &operand1, &oper, &operand2);
    switch(oper)
    {
        case '+':
            printf("%.2f + %.2f = %.2f\n", operand1, operand2, operand1 + operand2);
            break;
        case '-':
            printf("%.2f - %.2f = %.2f\n", operand1, operand2, operand1 - operand2);
            break;
        case '*':
            printf("%.2f * %.2f = %.2f\n", operand1, operand2, operand1 * operand2);
            break;
        case '/':
            printf("%.2f / %.2f = %.2f\n", operand1, operand2, operand1 / operand2);
            break;
    }
}

```

b.第二种方法:

```

#include "stdio.h"
void main()
{
    char oper;

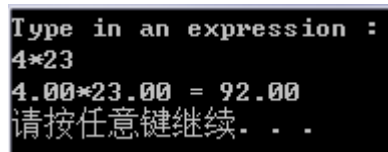
```

```

float operand1, operand2;
printf("Type in an expression : \n");
scanf("%f%c%f", &operand1, &oper, &operand2);
if (oper == '+')
    printf("%.2f + %.2f = %.2f\n", operand1, operand2, operand1 + operand2);
else if (oper == '-')
    printf("%.2f - %.2f = %.2f\n", operand1, operand2, operand1 - operand2);
else if (oper == '*')
    printf("%.2f * %.2f = %.2f\n", operand1, operand2, operand1 * operand2);
else if (oper == '/')
    printf("%.2f / %.2f = %.2f\n", operand1, operand2, operand1 / operand2);
else
    printf("input right opreation!\n");
}

```

③ 运行结果：



```

Type in an expression :
4*23
4.00*23.00 = 92.00
请按任意键继续. . .

```

4) 征税的办法如下：收入在 800 元以下（含 800 元）的不征税；收入在 800 元以上，1200 元以下者，超过 800 元的部分按 5% 的税率收税；收入在 1200 元以上，2000 元以下者，超出 1200 元部分按 8% 的税率收税；收入在 2000 元以上者，2000 元以上部分按 20% 的税率收税，试编写按收入计算税费的程序。

① 程序分析：

使用 switch 语句时，由于输入的数字是个随机的，并不能满足整数 800,1200 等等，故在用 switch 之前先对数字求解，用一个整数替代一部分数，分为 0,1,2,3,4,5，其他共七个部分，即可求出。

② 参考源程序：

```

#include<stdio.h>
void main()
{
    int s, p;
    float tax;
    printf("Enter income:");
    scanf("%d", &s);
    printf("\n");
    if (s > 800)
    {
        if ((s - 800) % 100 != 0)
            p = (s - 800) / 100 + 1;
        else
            p = (s - 800) / 100;
        switch (p)
        {

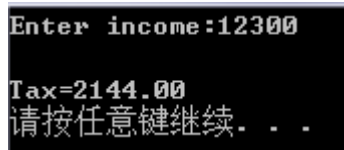
```

```

        case 0:
        case 1:
        case 2:
        case 3:
        case 4: tax = (s - 800)*0.05;
                break;
        case 5: tax = 400 * 0.05 + (s - 1200)*0.08;
                break;
        default: tax = 400 * 0.05 + 800 * 0.08 + (s - 2000)*0.20;
                break;
    }
}
else
    tax = 0.0;
printf("Tax=%.2f\n", tax);
}

```

③ 运行结果:



```

Enter income:12300
Tax=2144.00
请按任意键继续. . .

```

4 实验内容

1) 程序分析题, 深刻理解 switch...case 结构的执行流程。(写出下列程序的输出结果, 然后上机确认结果)

① 程序 1:

```

#include<stdio.h>
void main ( )
{
    int s=2,k;
    for(k=7;k>=4;k--)
    {
        switch(k)
        {
            case 1:
            case 4:
            case 7:
                s++;
                break;
            case 2:
            case 3:
            case 6:
                break;
            case 0:
            case 5:
                s+=2;
                break;
        }
    }
}

```

```
    printf("s=%d\n",s);
}
```

- 2) 输入两个整数，输出较大者。
- 3) 有三个整数 a , b , c ，由键盘输入，输出其中最大的数。
- 4) 从键盘输入一个字符型数据，若输入一个数字字符 ('0'-'9')，则将其转换成相应的整数显示出来；若输入其它字符，则显示出错信息。（根据字符型数据的 ASCII 码值是否在 '0' 字符与 '9' 字符之间进行判断。）
- 5) 输入一个自然数，判断它是奇数还是偶数。
- 6) 有一个函数：
$$Y = \begin{cases} x, & x < 1 \\ 2x - 1, & 1 \leq x < 10 \\ 3x - 11, & x \geq 10 \end{cases}$$
，写一程序，输入 x ，输出 Y 的值。
- 7) 先输入一个整数，再输出与该整数对应的星期几的英文，其中星期日到星期六一次对应于整数 0~6。
- 8) 某产品生产成本 $c=c_1+mc_2$ ，其中 c_1 为固定成本， c_2 为单位产品可变成本。当生产数量 $m < 10000$ 时， $c_1=20000$ 元， $c_2=10$ 元；当生产数量 $m \geq 10000$ 时， $c_1=40000$ 元， $c_2=5$ 元；编写一个程序，其功能为：分别计算出生产数量为 6000 以及 25000 时，总生产成本及单位生产成本。

实验三 循环程序设计实验

1 实验目的

- 1) 熟练的掌握 `while`, `for`, `do-while` 这三种语句实现循环的方法;
- 2) 了解这三种循环语句的异同、各自的适应性、循环嵌套的使用;
- 3) 掌握 `break` 语句、`continue` 语句以及 `goto` 语句在循环体中的运用, 改变程序流程;
- 4) 能够灵活的运用循环的方法实现各种复杂的程序。

2 实验相关知识

1) 循环控制语句

循环结构是在给定条件时, 反复执行某个程序段, 反复执行的程序叫循环体。C 语言有三种循环流程控制, 分别是 `while` 循环, `for` 循环, `do-while` 循环。

a) `while` 语句的程序表达式:

```
while(表达式)
{
    循环体语句;
}
```

执行 `while` 语句的过程如下:

- ①计算 `while` 之后的表达式的值;
- ②测试表达式的值, 当值为非 0 时, 转步骤 3, 值为 0 时, 则结束 `while` 语句;
- ③执行 `while` 语句的循环体, 并转为步骤 1, 从而构成了循环。

b) `for` 语句的一般形式:

```
for(表达式 1;表达式 2;表达式 3)
{
    循环体语句;
}
```

`for` 语句的执行过程如下:

- ①计算表达式 1;
- ②计算表达式 2 的值。并测试其值为 0 或非 0。若值为非 0, 转步骤 3; 否则结束 `for` 语句;
- ③执行循环体;
- ④计算表达式 3;
- ⑤转向步骤 2。

c) `do-while` 语句的一般形式:

```
do
{
    循环体语句;
}while(表达式);
```

`do-while` 语句执行过程如下:

- ①执行 `do-while` 语句的循环体;
- ②求 `while` 之后的表达式的值;
- ③测试表达式 的值, 当值为非 0 时, 转步骤 1, 值为 0 时, 则结束循环。

2) 辅助控制语句

a) break 语句

break 语句的使用范围:

①在 switch 语句中可以使流程跳出 switch 语句, 继续执行 switch 下面的语句;

②在循环体语句中可以用来从循环体内跳出循环体, 即结束当前循环, 执行循环下面的语句 (break 语句只能跳出一层循环);

③break 不能用于循环体语句和 switch 语句之外的任何其他语句。

b) continue 语句

结束本次循环, 即跳过循环体尚未执行的语句, 接着进行下一次是否执行循环的判定。

c) goto 语句和标号

一般形式:

goto 标号;

标号:语句;

作用是把程序控制转移到标号指定的语句处。

3 实验范例

1) 求下列表达式的值。

$1/3+3/5+5/7+\dots+n/(n+2)$ (n 为任意一个奇数)

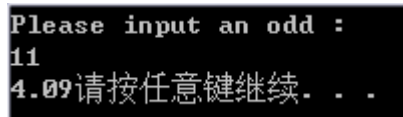
① 程序分析:

从上述表达式的形式看, 只有一个变量 n , 当 n 确定下来之后, 我们就从 $1/3$ 开始, 按照式子中给出的规律, 一直累加到 $n/(n+2)$, 该题采用 for 循环来实现。

② 参考源程序:

```
#include "stdio.h"
void main()
{
    float i, n, sum = 0.0;
    printf("Please input an odd : \n");
    scanf("%f", &n);
    for (i = 1; i <= n; i = i + 2)
        sum = sum + i / (i + 2);
    printf("%.2f", sum);
}
```

③ 运行结果:



```
Please input an odd :
11
4.09请按任意键继续. . .
```

2) 打印倒等腰三角, 输入一个底边长为 n , 例如: 输入 $n=7$, 输出



```
*****
 ***
  ***
   *
```

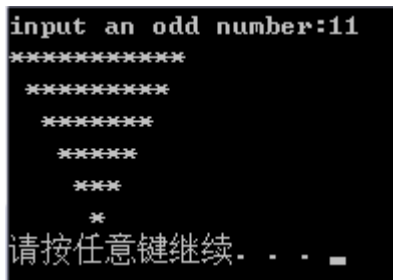
① 程序分析:

使用循环的方式输出，三种循环方式都可以，该程序使用 for 循环。

② 参考源程序：

```
#include <stdio.h>
int main(void)
{
    int i = 0;
    int j = 0;
    int n = 0;
    int counter = 0;
    printf("input an odd number:");
    scanf("%d", &n);
    for (i = n; i > 0; i -= 2)
    {
        for (j = 0; j < counter; j++)
        {
            printf(" ");
        }
        for (j = 0; j < i; j++)
        {
            printf("*");
        }
        counter++;
        printf("\n");
    }
    return 0;
}
```

③ 运行结果：



3) 输入两个分数，输出两分数之和(要求约分)。

例如： 输入 3/5 7/8 输出： 59/40；

输入:2/8 5/12 输出: 2/3。

① 程序分析：

首先需要寻找分母之间的最小公倍数，然后相加，再看看相加的结果有没有共同的公约数，再约分，即转变为求最大公约数和最小公倍数的问题了，本题用了 while 循环。

② 参考源程序：

```
#include <stdio.h>
int main(void)
{
```

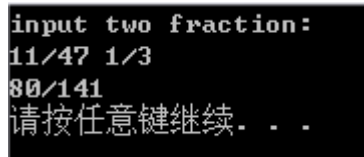


```

int a1, b1, a2, b2;
char ch1, ch2;
int s1, s2;
int m, n, r, temp;
printf("input two fraction:\n");
scanf("%d%c%d %d%c%d", &a1, &ch1, &b1, &a2, &ch2, &b2);
s1 = a1 * b2 + b1 * a2;
s2 = b1 * b2;
n = s1;
m = s2;
if (n < m)
{
    temp = m;
    m = n;
    n = temp;
}
while (m != 0)
{
    r = n % m;
    n = m;
    m = r;
}
s1 /= n;
s2 /= n;
if (s1 == s2)
    printf("%d\n", s1);
else
    printf("%d/%d\n", s1, s2);
return 0;
}

```

③ 运行结果：



```

input two fraction:
11/47 1/3
80/141
请按任意键继续. . .

```

4) 打印出所有的“水仙花数”，所谓“水仙花数”是指一个三位数，其各位数字立方和等于该数本身。例如： $153 = 1^3 + 5^3 + 3^3$ 。

① 程序分析：

求出每个数的个位、十位、百位，并与原来的数进行比较，本题运用了循环嵌套。

② 参考源程序：

```

#include <stdio.h>
int main(void)
{

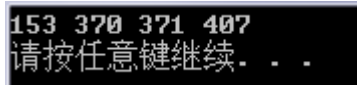
```

```

int i = 0;
int sum = 0;
int j = 0;
for (i = 100; i < 1000; i++)
{
    sum = 0;
    int temp = i;
    while (temp)
    {
        j = temp % 10;
        temp /= 10;
        sum += j * j * j;
    }
    if (sum == i)
    {
        printf("%d ", i);
    }
}
printf("\n");
return 0;
}

```

③ 运行结果：



```

153 370 371 407
请按任意键继续。 . .

```

4 实验内容

1) 程序分析题。（写出下列程序的输出结果，然后上机确认结果）

① 程序 1:

```

#include<stdio.h>
void main()
{
    int i,j,p,s;
    s=0;
    for(i=1;i<=4;i++)
    {
        p=1;
        for(j=1;j<=4;j++)
            p=p*j;
        s=s+p;
    }
    printf("s=%d\n",s);
}

```

② 程序 2: 将输入的小写字母转换为大写字母输出，当输入为 ‘\$’ 字符时，则停止转换，请在空白处填上合适的语句，以使程序正确运行。

```

#include<stdio.h>
void main( )
{

```

```

char c;
do{
    printf("enter a char:");

    if('a'<=c&& c<='z')
        printf("%c\n",c);
    }while(c!='$');
}

```

- 2) 从 1 累加到 100，用 while 语句。
- 3) 已知 $a_1 = 10$, $a_2 = -3$, $a_i = 3a_{i-1} + a_{i-2}$ ，求 $\{a_i\}$ 的前十项。
- 4) 给一个不多于 5 位的正整数，要求：求出它是几位数，分别打印出每一位数字，最后按照逆序打印各位数字，例如原数为 321，应输出为 123。
- 5) 输入两个整数，求它们的最大公约数和最小公倍数。
- 6) 编写一猜数游戏程序，随机产生某个整数，从键盘反复输入整数进行猜数，当未猜中时，提示输入过大或过小；猜中时，指出猜的次数，最多允许猜 20 次。
- 7) 计算 1~999 中能被 3 整除的数，且至少有一位是 5 的所有整数。
- 8) 计算 $1!$ ， $2!$ ， $3!$ ， $4!$ ，...， $10!$ 。
- 9) 猴子吃桃问题：第一天吃掉总数的一半多一个，第二天又将剩下的桃子吃掉一半多一个，以后每天吃掉前一天剩下的一半多一个，，到第十天准备吃的时候只剩下一个桃子，求第一天开始吃的时候的桃子的总数。
- 10) 输入一个整数，求它包含有多少个 2 的因子。例如，8 含有 3 个 2 的因子，10 含有一个 2 的因子，15 不含有 2 的因子。
- 11) 求 $S_n = a + aa + aaa + \dots + \underbrace{aaa \dots a}_n$ (n 个 a)，其中 a 是一个数字，例如：2+22+222+2222+22222。
(此时 n=5，n 由键盘输入)

实验四 函数与变量的存储类型实验

1 实验目的

- 1) 掌握 C 语言程序结构开发的基本要点，即自顶向下，逐步求精和模块化设计；
- 2) 掌握函数的定义方法和调用规则；
- 3) 熟悉 C 语言中的变量存储类型，掌握静态型变量、自动型变量、寄存器型变量和外部参照型变量的作用域以及存在性和可见性；
- 4) 掌握在 C 语言程序中主调函数和被调函数之间进行数据传递的规则，以及常见数据结构作为函数参数的实参形参的对应关系。

2 实验相关知识

在 C 语言的程序设计中，无论多么复杂、规模多么大的程序，最终都落实到一个小型简单的函数编写工作上，因此，C 语言程序设计的基础工作是函数的设计和编制。一个 C 语言程序清单是由一个或多个函数定义组成的，包括系统提供的标准库函数和用户自定义的函数。

1) 函数的定义

函数定义的一般格式为：

```
<存储类型> <数据类型> 函数名 (<形式参数及说明>)
{
    说明语句;
    执行语句;
}
```

2) 函数的调用

a) 函数的原型（也称函数声明）

函数声明是指在主调函数中，对在本函数中将要被调用的函数提前做必要的声明。函数原型的一般格式为：

函数数据类型 函数名（参数类型 1，参数类型 2，.....）

（或 函数数据类型 函数名（参数类型 1 参数名 1，参数类型 2 参数名 2，.....））

b) 函数的调用一般格式：

函数名（实参表）；

3) 变量的存储类型

a) 变量定义的一般形式为：

<存储类型> 数据类型 变量名表；

b) 存储类型包括：自动型变量[auto]、寄存器型变量[register]、外部参照型变量[extern]、静态型变量[static]。

4) 函数间的数据传递

C 语言在函数间传递数据有 4 中方式，值传递方式、地址传递方式、返回值方式、全局变量传递方式。其中：

- a) 值传递方式所传递的是参数值，其特点是“参数值的单向传递”。
- b) 地址传递方式所传递的是地址，其特点是“参数值的双向传递”。
- c) 返回值方式不是在形式参数和实际参数之间传递数据，而是通过函数调用后直接返回一个值到主调函数中。该函数的数据类型不能是 void 类型，且函数体中应有“return<表达式>”语句。

d) 全局变量传递方式不是在形式参数和实际参数之间传递数据，而是利用在主调函数和被调函数中均有有效的全局变量，在主调函数和被调函数之间任意传递数据。

5) 递归函数

一个函数直接或间接地调用其自身，便构成了函数的递归调用。这种函数成为递归函数。

3 实验范例

1) 编写两个函数，分别求最大公约数 gcd (greatest common divisor) 和最小公倍数 lcm (least common multiple) 。

① 程序分析：

最大公约数函数：int gcd(int a,int b);

最小公倍数函数：int lcm(int a,int b,int g)，其中 int g 为两个数的最大公约数。

② 参考源程序：

```
#include<stdio.h>
int gcd(int a, int b)
{
    int t, r;
    if (a < b)
    {
        t = a;
        a = b;
        b = t;
    }
    while (b != 0)
    {
        r = a%b;
        a = b;
        b = r;
    }
    return(a);
}

int lcm(int a, int b, int g)
{
    return(a*b / g);
}

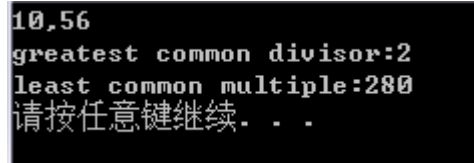
void main()
{
    int m, n, g, l;
    scanf("%d,%d", &m, &n);
    g = gcd(m, n);
    l = lcm(m, n, g);
```

```

printf("greatest common divisor:%d\n", g);
printf("least common multiple:%d\n", l);
}

```

③ 运行结果:



```

10,56
greatest common divisor:2
least common multiple:280
请按任意键继续. . .

```

2) 求方程 $ax^2+bx+c=0$ 的根, 用三个函数分别求 b^2-4ac 大于零、等于零和小于零时的根。

① 程序分析: 分三种情况来写这个程序, 所以有三个对应的子函数:

b^2-4ac 大于零函数: void f1(float a,float b);

b^2-4ac 等于零函数: void f1(float a,float b);

b^2-4ac 小于零函数: void f3(float a,float b)。

并且此题采用全局变量, 简化了很多变量, 如下为全局变量的说明:

float X1 方程第一个根;

float X2 方程第二个根;

float Disc = b^2-4ac ;

float Re 实部 (real part) ;

float Im 虚部 (imaginary part) 。

② 参考源程序:

```
#include<stdio.h>
```

```
#include<math.h>
```

```
float X1, X2, Disc, Re, Im;
```

```
void f1(float a, float b)
```

```
{
    X1 = (-b + sqrt(Disc)) / (2 * a);
    X2 = (-b - sqrt(Disc)) / (2 * a);
}
```

```
void f2(float a, float b)
```

```
{
    X1 = (-b) / (2 * a);
}
```

```
void f3(float a, float b)
```

```
{
    Re = (-b) / (2 * a);
    Im = sqrt(-Disc) / (2 * a);
}
```

```
void main()
```

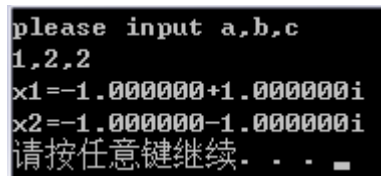
```
{
```

```

float a, b, c;
printf("please input a,b,c\n");
scanf("%f,%f,%f", &a, &b, &c);
Disc = b*b - 4 * a*c;
if (Disc > 0)
{
    f1(a, b);
    printf("x1=%f\nx2=%f\n", X1, X2);
}
else if (fabs(Disc) < 1e-6)
{
    f2(a, b);
    printf("x1=x2=%f\n", X1);
}
else
{
    f3(a, b);
    printf("x1=%f+%fi\nx2=%f-%fi\n", Re, Im, Re, Im);
}
}

```

③ 运行结果：



```

please input a,b,c
1,2,2
x1=-1.000000+1.000000i
x2=-1.000000-1.000000i
请按任意键继续. . .

```

3) 编写一个判断素数（prime number）的函数。

① 程序分析：

用一个子函数来说明素数函数：int prime (int n)，有返回值，当返回值为 1 则为素数，当返回值为 0 则不是素数。

② 参考源程序：

```

#include<stdio.h>
#include<math.h>
int prime(int n)
{
    int i, flag = 1;
    for (i = 2; i <= (int)sqrt(n/1.0); i++)
        if (n%i == 0)
            flag = 0;
    return(flag);
}

void main()
{

```

```

int n;
printf("please input an integer:\n");
scanf("%d", &n);
if (prime(n))
    printf("%d is a prime.\n", n);
else
    printf("%d is not a prime.\n", n);
}

```

③ 运行结果：

```

please input an integer:
153
153 is not a prime.
请按任意键继续. . .

```

4) 有一对雌雄兔子，假定两个月便可以繁殖雌雄各一对兔子。问 n 各月后共有多少对兔子？

① 程序分析：

用递归的方法来解决该程序，首先需要满足递归的三要素：

递归的形式： $T[n]=T[n-1]+T[n-2]$;

基本推导： $T[1]=1$, $T[2]=1$, $T[3]=2$, $T[4]=3$, $T[5]=5$, ;

结束条件： n 个月。

② 参考源程序：

```

#include<stdio.h>
int sum(int n)
{
    int rabbit=1;
    if (n <= 2)
        return rabbit;
    else if (n >= 3)
    {
        rabbit = sum(n - 1) + sum(n - 2);
    }
    return rabbit;
}
int main()
{
    int month;
    scanf("%d", &month);
    printf("%d\n", sum(month));
}

```

③ 运行结果：

```

12
总共产生兔子对数数目: 144
请按任意键继续. . .

```


4 实验内容

1) 程序分析题。(单步执行, 认真分析各种类型变量的生存周期和作用域, 并写出程序运行结果和各个变量的作用域和周期)

①程序 1:

```
#include <stdio.h>
void func( );
int a ;
void func( )
{
    printf("no 1 a=%d",a);
}
void main()
{
    int a = 1;
    printf("no 1 a=%d",a);
    func( );
    {
        int a = 1;
        printf("no 1 a=%d",a);
        func( );
    }
}
```

②程序 2:

```
#include <stdio.h>
void func()
{
    static int a=0;
    register int b=0;
    auto int c=0;
    printf("a=%d\tb=%d\tc=%d\n",a++,b++,c++);
}
void main()
{
    func();
    func();
    func();
}
```

③程序 3:

```
#include <stdio.h>
int n = 1;
void func()
{
    static int x=4;
    int y=10;
    x=x+2;
    n=n+10;
    y=y+n;
    printf("func:x=%d,y=%d,n=%d\n",x,y,n);
}
void main( )
{
    static int x = 5;
    int y;
    y = n;
```

```

printf("main:x=%d,y=%d,n=%d\n",x,y,n);
func();
printf("main:x=%d,y=%d,n=%d\n",x,y,n);
func();
}

```

- 2) 输入两个整数，调用函数 `squSum()` 求两数平方和，返回主函数显示结果。
- 3) 角谷定理。输入一个自然数，若为偶数，则把它除以 2，若为奇数，则把它乘以 3 加 1。经过如此有限次运算后，总可以得到自然数值 1。求经过多少次可得到自然数 1。
- 4) 一个人赶着鸭子去每个村庄卖，每经过一个村子卖去所赶鸭子的一半又一只。这样他经过了七个村子后还剩两只鸭子，问他出发时共赶多少只鸭子？经过每个村子卖出多少只鸭子？
- 5) 输入 n 个整数 ($n < 10$)，排序后输出。排序的原则由函数的一个参数决定，参数值为 1，按递减顺序排序，否则按递增顺序排序。
- 6) 求方程 $ax^2 + \sin x = 0$ 在附近的一个实根， a 和 b 由键盘输入。
- 7) 编写计算定积分：

$$\int_0^1 (x^2 + e^x \sin x) dx$$

的近似值和函数及主函数。

实验五 数组实验

1 实验目的

- 1) 掌握一维数组和多维数组的定义和应用;
- 2) 熟练运用一维数组和多维数组的初始化方法;
- 3) 熟悉数组在函数间的传递;
- 4) 掌握选择排序法和冒泡排序法这两种数据排序法, 能够独立编程完成排序功能。

2 实验相关知识

1) 一维数组

- a) 一维数组的定义格式:

<存储类型> 数据类型 数组名[常量表达式];

- b) 一维数组初始化

在定义数组时对各元素指定初始值, 称为数组的初始化。其规则如下:

① 当对数组中全体元素赋值时, 可以不必指明数组的元素个数。在编译时会根据花括号中的初值个数确定数组的实际长度;

② 在定义数组时也可以只对一部分元素赋值。

- c) 一维数组的引用:

数组名[下标];

2) 二维数组

- a) 二维数组的定义格式:

<存储类型> 数据类型 数组名[下标][下标];

- b) 二维数组的初始化规则:

① 可以分行对各元素赋值;

② 可以将各初始值全部展开地写在一堆花括号内;

③ 可以只对部分元素赋值;

④ 可以在分行赋初值时, 只对该行中一部分元素赋初值;

⑤ 如果在定义数组时, 给出了全部数组元素的初值, 则数组的第一维下标可以省略。

- c) 二维数组的引用

数组名[下标 1][下标 2];

3 实验范例

- 1) 编写函数, 使得给定的一个二维数组 (3*3) 转置。

① 程序分析:

二维数组转置函数: `void trans(int a[][3])`, 该函数没有返回值, 通过传递地址实现转置。

② 参考源程序:

```
#include<stdio.h>
#include<stdlib.h>
void trans(int a[][3])
{
    int t, i, j;
```

```

    for (i = 0; i < 3; i++)
        for (j = i + 1; j < 3; j++)
        {
            t = a[i][j];
            a[i][j] = a[j][i];
            a[j][i] = t;
        }
}

void main()
{
    int a[3][3], i, j;
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            a[i][j] = rand() % 100;
    printf("array a:\n");
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
    trans(a);
    printf("transposed array a:\n");
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
}

```

③ 运行结果：

```

array a:
 41  67  34
  0  69  24
 78  58  62
transposed array a:
 41  0  78
 67  69  58
 34  24  62
请按任意键继续. . .

```

2) 编写一个函数，使得输入的一个字符串反序存放。

① 程序分析：

反序函数：void inverse(char c[])，函数没有返回值，通过传递地址实现反序。

② 参考源程序：

```
#include<stdio.h>
```

```
#include<string.h>
```

```

void inverse(char c[])
{
    int n, i, j;    char t;
    n = strlen(c);
    for (i = 0, j = n - 1; i < n / 2; i++, j--)
    {
        t = c[i];
        c[i] = c[j];
        c[j] = t;
    }
}

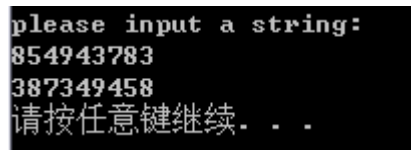
```

```

void main()
{
    void inverse(char c[]);
    char s[100];
    printf("please input a string:\n");
    gets(s);
    inverse(s);
    puts(s);
}

```

③ 运行结果:



```

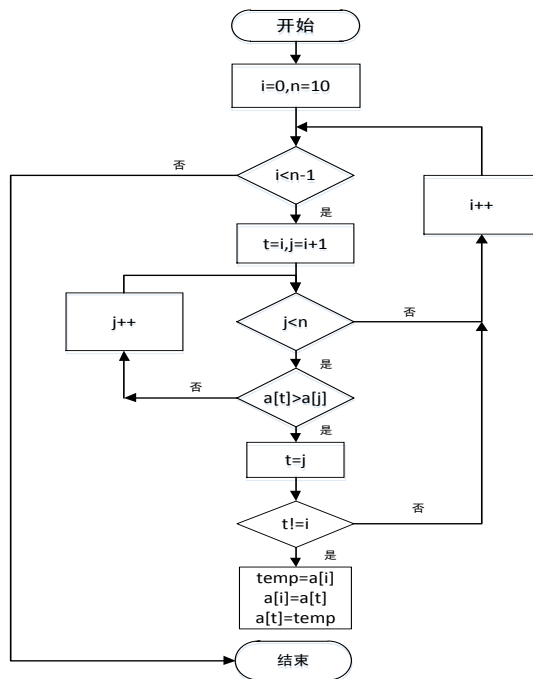
please input a string:
854943783
387349458
请按任意键继续. . .

```

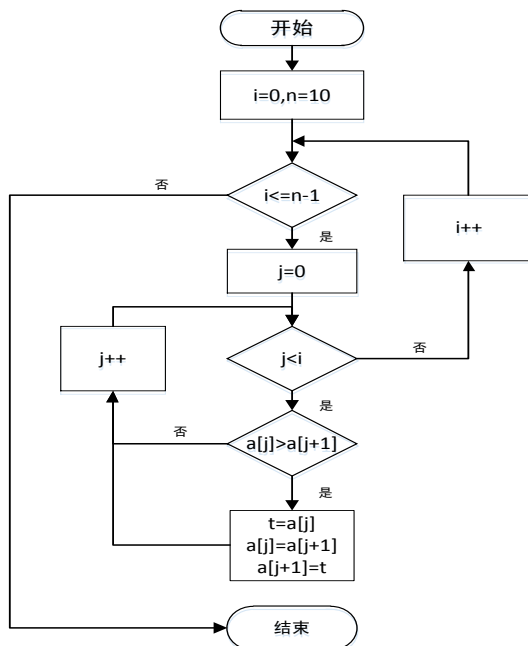
3) 使用选择法、冒泡法对 10 个数进行排序，并输出排序前后的数列。

① 程序分析:

选择排序流程图:



冒泡排序流程图：



② 参考源程序：

第一种方法：选择排序法

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
    int a[10] = { 12, 45, 7, 8, 96, 4, 10, 48, 2, 46 }, n = 10, i, j, t, temp;
```

```
    printf("Before sort:");
```

```
    for (i = 0; i < 10; i++)
```

```
    {
```

```
        printf("%4d", a[i]);
```

```

    }
    printf("\n");
    for (i = 0; i < 9; i++)
    {
        t = i;
        for (j = i + 1; j < 10; j++)
        {
            if (a[t] > a[j])
            {
                t = j;
            }
        }
        if (t != i)
        {
            temp = a[i];
            a[i] = a[t];
            a[t] = temp;
        }
    }
    printf("After sorted:");
    for (i = 0; i < 10; i++)
    {
        printf("%4d", a[i]);
    }
    printf("\n");
}

```

第二种方法：冒泡排序法

```
#include<stdio.h>
```

```
void main()
```

```

{
    int a[10] = { 12, 45, 7, 8, 96, 4, 10, 48, 2, 46 }, n = 10, i, j, t;
    printf("Before sort : ");
    for (i = 0; i < 10; i++)
    {
        printf("%4d", a[i]);
    }
    printf("\n");
    for (i = 0; i <= n - 1; i++)
    {
        for (j = 0; j < i; j++)
            if (a[j] > a[j + 1])
            {
                t = a[j];
                a[j] = a[j + 1];

```

```

        a[j + 1] = t;
    }
}
printf("After sorted : ");
for (i = 0; i < 10; i++)
{
    printf("%4d", a[i]);
}
printf("\n");
}

```

③ 运行结果：

```

Before sort: 12 45 7 8 96 4 10 48 2 46
After sorted: 2 4 7 8 10 12 45 46 48 96
请按任意键继续. . .

```

4) 编写程序，把下面的数据输入一个二维数组中。

25	36	78	13
12	26	88	93
75	18	22	32
56	44	36	58

然后执行以下操作：

- 输出矩阵两个对角线上的数；
- 分别输出各行和各列的和；
- 交换第一行和第三行的位置；
- 交换第二列和第四列的位置；
- 输出处理后的数组。

① 程序分析：

运用二维数组输出数据，即写两个循环，主要是了解所需要输出的数据在数组的哪个位置。

② 参考源程序：

```

#include <stdio.h>
#define SIZE 4
void main()
{
    int a[SIZE][SIZE] = { { 25, 36, 78, 13 }, { 12, 26, 88, 93 }, { 75, 18, 22, 32 }, { 56, 44, 36, 58 } };
    int i, j, t, sum;
    printf("二维数组:\n");
    for (i = 0; i < SIZE; i++)
    {
        for (j = 0; j < SIZE; j++)
        {
            printf("%5d", a[i][j]);
        }
        printf("\n");
    }
}

```



```

printf("\n");
printf("主对角线上的数:");
for (i = 0; i < SIZE; i++)
{
    printf("%4d", a[i][i]);
}
printf("\n");
printf("副对角线上的数:");
for (i = 0; i < SIZE; i++)
{
    printf("%4d", a[i][SIZE - 1 - i]);
}
printf("\n\n");
for (j = 0; j < SIZE; j++)
{
    sum = 0;
    for (i = 0; i < SIZE; i++)
    {
        sum += a[i][j];
    }
    printf("第%d列的和=%d\n", j + 1, sum);
}
printf("\n");
for (j = 0; j < SIZE; j++)
{
    t = a[0][j];
    a[0][j] = a[2][j];
    a[2][j] = t;
}
printf("交换第一行和第三行后的二维数组:\n");
for (i = 0; i < SIZE; i++)
{
    for (j = 0; j < SIZE; j++)
    {
        printf("%5d", a[i][j]);
    }
    printf("\n");
}
printf("\n");
for (i = 0; i < SIZE; i++)
{
    t = a[i][1];
    a[i][1] = a[i][3];
    a[i][3] = t;
}

```

```

    }
    printf("交换第2列和第4列后的二维数组:\n");
    for (i = 0; i < SIZE; i++)
    {
        for (j = 0; j < SIZE; j++)
        {
            printf("%5d", a[i][j]);
        }
        printf("\n");
    }
}

```

③ 运行结果：

```

二维数组:
25  36  78  13
12  26  88  93
75  18  22  32
56  44  36  58

主对角线上的数: 25  26  22  58
副对角线上的数: 13  88  18  56

第1列的和=168
第2列的和=124
第3列的和=224
第4列的和=196

交换第一行和第三行后的二维数组:
75  18  22  32
12  26  88  93
25  36  78  13
56  44  36  58

交换第2列和第4列后的二维数组:
75  32  22  18
12  93  88  26
25  13  78  36
56  58  36  44

请按任意键继续. . .

```

4 实验内容

- 1) 统计一维整型数组 $a[N]$ 中正数、负数和零的个数。
- 2) 求一维整型数组 $a[N]$ 中所有数的和。
- 3) 求二维整型数组 $a[N][N]$ 中所有正数的和。
- 4) 找出二维整形数组 $a[N][N]$ 中最大数以及其所在的行和列。
- 5) 求 $N*N$ 的行列式中主对角线上的数的和。
- 6) 将二维整型数组 $a[N][N]$ 中每一行的数按从小到大进行排序。
- 7) 求二维整型数组 $a[N][N]$ 中每一行的数中的最大值，放在另一个一维数组 $b[N]$ 中，最后对 $b[N]$ 中的数按从小到大的进行排序。

- 8) 计算矩阵 $A_{4 \times 3}$ 的转置矩阵 A^T ，例如： $A = \begin{bmatrix} 1 & 9 & 8 \\ 5 & 3 & 2 \\ 8 & 4 & 3 \\ 2 & 6 & 0 \end{bmatrix}$ ， $A^T = \begin{bmatrix} 1 & 5 & 8 & 2 \\ 9 & 3 & 4 & 6 \\ 8 & 2 & 3 & 0 \end{bmatrix}$ 。

9) 某班有 N 个学生，每个学生有三门课程，试编程用数组实现以下功能：

- ① 从键盘输入所有学生的成绩；
- ② 求每个学生的平均分；
- ③ 按平均分排序。

实验六 指针实验

1 实验目的

- 1) 掌握指针的概念和定义方法;
- 2) 掌握指针的操作符和指针的运算;
- 3) 掌握指针与数组的关系。

2 实验相关知识

1) 指针变量的定义和初始化

指针变量是用来存储地址的变量。其定义格式如下:

<存储类型> 数据类型 *指针变量名 1[=初值], ...

指针定义时的初值可以是 NULL, 表示一个空地址, 其值是 0。注意, 当定义一个指针变量而未指定初值时, 并不表示它指向 NULL。

当需要对一个指针变量进行操作时, 必须使指针变量指向一个特定地址, 可以通过以下两种方法实现:

- a) 将指针指向一个变量、一个数组或一个具体的地址。
- b) 使用 malloc 或 calloc 函数进行动态地址分配。

2) &运算符和*运算符

- a) &运算符称为“取地址运算符”。
- b) *运算符称为“指针运算符”, 也称为“间接运算符”。

3) 使用指针运算符应注意的问题

- a) 指针变量定义中的“*”与“&”运算符的区别

指针变量中定义的“*”与“&”不是运算符, 它只是表示其中的变量是一个指针类型的变量。对语句“*p=5;”, 其中的“*”是指针运算符“*”, “*p”代表 p 指向的变量。

- b) &运算符与*运算符是互逆的。如 y=x; y=&*x; 两个语句是等效的。

4) 对指针变量的操作

在定义了一个指针变量之后, 如 int *p, a; , 就可以对该指针进行各种操作。

- a) 给一个指针变量赋一个地址, 如 p=&a; 。
- b) 指针变量中存储的是它所指向的变量的地址值, 可以输出这个地址, 格式如下:

```
printf("%lu", (unsigned int p));    /* 按无符号整型输出 */
printf("%p", p);                  /* 按十六进制地址格式输出 */
printf("%o", p);                  /* 按八进制整型输出 */
```

5) 指向数组的指针变量的使用

数组名本身可以看成是该数组的指针, 但它的位置是固定的。可以定义一个指针变量, 并把这个指针指向该数组的起始地址, 那么通过对指针的运算, 就可以完成对数组的访问。注意数组名所代表的地址是一个常量, 因而不能改变其值, 但指针是一个变量, 其值可以改变。

例如, 引用一维数组元素 int a[10], *p=a;的方法有以下两种:

- a) 下标法, 如 a[i], p[i]。
- b) 地址法, 如*(a+i), *(p+i)。

对指向数组、字符串的指针变量可以进行相加减运算, 如 p+n、p-n、p++、p--等。

对指向同一数组的两个指针变量可以相减, 但不可以相加。对指向不同类型的指针变量作加减运算是无意义的。

3 实验范例

1) 写一函数，实现两个字符串的比较。即自己写一个 strcmp 函数，函数原型为：

```
int strcmp(char *p1, char *p2)
```

设 p1 指向字符串 s1，p2 指向字符串 s2。要求：当 s1=s2 时，返回值为 0。当 s1 不等于 s2 时，返回它们二者的第一个不同字符的 ASCII 码差值（如“BOY”与“BAD”，第二字母不同，“O”与“A”之差为 79-65=14）；如果 s1>s2，则输出正值；如果 s1<s2，则输出负值。

① 程序分析：

定义指针变量，比较时两个指针进行自加就可逐个字符比较。

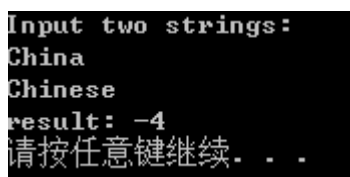
② 参考源程序：

```
#include <stdio.h>

int main()
{
    int strcmp(char *p1, char *p2); /*在函数内部声明，适用范围为main函数中，若有其他函数，
    则不能使用*/
    int m;
    char str1[20], str2[20], *p1, *p2;
    printf("Input two strings:\n");
    scanf("%s", str1);
    scanf("%s", str2);
    p1 = &str1[0];
    p2 = &str2[0];
    m = strcmp(p1, p2);
    printf("result: %d\n", m);
}

int strcmp(char *p1, char *p2) /*两个字符串比较的函数*/
{
    int i;
    i = 0;
    while (*(p1 + i) == *(p2 + i))
        if (*(p1 + i++) == '\0')
            return(0); /*相等时返回结果0*/
    return(*(p1 + i) - *(p2 + i)); /*不等时返回结果为第一个不等字符ASCII码的差值*/
}
```

③ 运行结果：



2) C 语言用指针方法输入 3 个字符串按由小到大顺序输出。

① 程序分析：

字符串的比较需要用到库函数 strcmp，其原型为：int strcmp(char *s1, const char *s2);

当 $s1 < s2$ 时，返回为负数；

当 $s1 = s2$ 时，返回值 = 0；

当 $s1 > s2$ 时，返回正数。

即：两个字符串自左向右逐个字符相比（按 ASCII 值大小相比较），直到出现不同的字符或遇 '\0' 为止。

② 参考源程序：

```
#include "stdio.h"
#include "string.h"
int main()
{
    char *t;
    char *p1, *p2, *p3;
    char ch1[20], ch2[20], ch3[20];
    p1 = ch1;
    p2 = ch2;
    p3 = ch3;
    printf("No1:");
    scanf("%s", p1);
    printf("No2:");
    scanf("%s", p2);
    printf("No3:");
    scanf("%s", p3);
    if (strcmp(p1, p2) > 0)
    {
        t = p1;
        p1 = p2;
        p2 = t;
    }
    if (strcmp(p1, p3) > 0)
    {
        t = p1;
        p1 = p3;
        p3 = t;
    }
    if (strcmp(p2, p3) > 0)
    {
        t = p2;
        p2 = p3;
        p3 = t;
    }
    printf("%s\n%s\n%s\n", p1, p2, p3);
    return 0;
}
```

③ 运行结果：

```
No1:hello
No2:world
No3:welcome
hello
welcome
world
请按任意键继续. . .
```

3) 编程输入一行文字，找出其中的大写字母，小写字母，空格，数字，及其他字符的个数。

① 程序分析：

定义指针变量，对输入的字符串进行遍历，用 `while` 循环，结束条件时字符串的结束符 `'\0'`，循环过程就是指针往后移动，逐步判断每一个字符是属于哪个区间，若属于那个区间，则对应的数字分别加 1，以此类推。

② 参考源程序：

```
#include<stdio.h>
void main()
{
    int a = 0, b = 0, c = 0, d = 0, e = 0, i = 0;
    char *p, s[20];
    while ((s[i] = getchar()) != '\n')
        i++;
    p = s;
    while (*p != '\n')
    {
        if (*p >= 'A' && *p <= 'Z')
            a++;
        else if (*p >= 'a' && *p <= 'z')
            b++;
        else if (*p == ' ')
            c++;
        else if (*p >= '0' && *p <= '9')
            d++;
        else
            e++;
        p++;
    }
    printf("大写字母 %d 小写字母 %d\n", a, b);
    printf("空格 %d 数字 %d 非字符 %d\n", c, d, e);
}
```

③ 运行结果：

```
hus fDS09-= .;f
大写字母 2 小写字母 5
空格 2 数字 2 非字符 4
请按任意键继续. . .
```

4) 写一个函数，将 3*3 矩阵转置。

① 程序分析：

使用指针的偏移，一个二维数组相当于一个**指针，即 `a[i][j]=*(*(p+i)+j)`；调用子函数进行转置功能。

② 参考源程序：

```
#include "stdio.h"
void Transpose(int(*matrix)[3])
{
    int temp;
    int i, j;
    for (i = 1; i < 3; i++)/*转置*/
    {
        for (j = 0; j < i; j++)
        {
            temp = (*(matrix + j) + i);
            (*(matrix + j) + i) = (*(matrix + i) + j);
            (*(matrix + i) + j) = temp;
        }
    }
}

void main()
{
    int a[3][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
    printf("original matrix:\n");
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
    Transpose(a);
    printf("transposed matrix:\n");
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
}
```


③.运行结果:

```
original matrix:
1 2 3
4 5 6
7 8 9
transposed matrix:
1 4 7
2 5 8
3 6 9
请按任意键继续. . .
```

4 实验内容

1) 程序分析题。(先写出答案, 然后上机验证)

① 程序 1:

```
#include <stdio.h>
void reverse(int a[],int n)
{
    int *p;
    for(p=a+n-1;p>=a;p--)
        printf("%4d",*p);
    printf("\n");
}
void main( )
{
    int a[20], n;
    int i;
    printf("Input the length of array:");
    scanf("%d",&n);
    printf("Input the number of array:");
    for( i=0; i<n; i++ )
        scanf("%d",&a[i]);
    reverse(a,n);
}
```

该程序的输出结果如下 (请补充完整):

```
Input the length of array:5
Input the number of array:1 4 7 3
9 3 4
```

② 程序 2:

```
#include <stdio.h>
void main()
{
    int a,b,k=4,m=6,*p=&k,*q=&m;
    a=p==&m;
    b=(*p)/(*q)+7;
    printf("a=%d\n",a);
    printf("b=%d\n",b);
}
```

该程序执行之后, 输出的 a 的值为____, b 的值为____。

③ 程序 3:

```
void main()
{
    int a[5]={1,2,3,4,5};
}
```

```

    int *ptr=(int *)(&a+1);
    printf("%d,%d",*(a+1),*(ptr-1));
}

```

输出结果是什么：_____。

④ 程序 4:

```

#include<stdio.h>
void fun(int (*p)[N], int *q, int n);
#define N 2
void main()
{
    int a[N][N]={ {1,2},{3,4}};
    int i,b[N*N]={5,6,7,8};
    fun(a,b,N);
    for(i=0; i<N*N; i++)
        printf("%3d", b[i]);
}
void fun(int (*p)[N], int *q, int n)
{
    int i,j,k=0;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            q[k++]=p[i][j];
}

```

运行输出结果是：_____。

2) 若有如下图所示五个连续的 `int` 类型的存储单元并赋值, `a[0]` 的地址小于 `a[4]` 的地址。
`p` 和 `s` 是基类型为 `int` 的指针变量。

<code>a[0]</code>	<code>a[1]</code>	<code>a[2]</code>	<code>a[3]</code>	<code>a[4]</code>
22	33	44	55	66

回答下列问题:

- ①若 `p` 已指向存储单元 `a[1]`, 则 `a[4]` 用指针 `p` 表示为_____;
 - ②若指针 `s` 指向存储单元 `a[2]`, `p` 指向存储单元 `a[0]`, 表达式 `s-p` 的值是_____;
 - ③若指针 `s` 指向存储单元 `a[2]`, `p` 指向存储单元 `a[1]`, 表达式 `*(s+1)-*p++` 的值是_____。
- 3) 输入三个整数, 按从小到大的顺序输出, 用三种不同方式实现。
- 4) 有 `n` 个整数, 使其前面各数顺序向后移 `m` 个位置, 最后 `M` 个数变成最前面 `m` 个数。
- 5) 利用数组和指针, 将一个 `4*4` 的矩阵转置, 并输出矩阵中的最大值及其位置。
- 6) 将一个 `5×5` 的矩阵中最大的元素放在中心, 4 个角分别放 4 个最小的元素 (顺序为从左到右, 从上到下顺序依次从小到大存放), 编写一个函数, 并用 `main` 函数调用来实现。

实验七 指针的综合应用

1 实验目的

- 1) 进一步了解数组的指针和指向数组的指针变量;
- 2) 熟悉指针作为函数的参数以及返回指针的函数;
- 3) 了解函数指针。

2 实验相关知识

1) 数组指针与指针数组

a) 数组指针

数组指针是指数组首元素地址的指针，也是指向数组的指针。例：`int (*p)[10]`; `p` 即为指向数组的指针。

b) 指针数组

一个数组，如果每个元素都是指针类型的，则它是指针数组。指针数组用来存放一系列地址。例：`int* a[4]`; `a` 为指针数组，表示数组 `a` 中的元素都为 `int` 型指针。（`*a[i]` 与 `*(a[i])` 是一样的，因为 `[]` 优先级高于 `*`）

2) 多级指针

当定义的某个指针变量专门用来存放其他指针变量的地址时，这样的指针变量就称为指针的指针，也称为二级指针。

3) 指针作为函数的参数

指针变量可以作为函数的参数，所代表的意义是将实参的地址传递给形参，因此可以实现参数值的双向传递，如：

```
void main()
{
    ...
    a=5;
    fun(&a);
    ...
}
void fun(int *a)
{
    *a=1;
}
```

执行结果，`a` 的值将被改变为 8。

3) 指针型函数及函数指针

a) 指针型函数，是指函数的返回值是指针型的，即这类函数的返回值是地址数据。

指针型函数调用与一般函数的调用方法完全相同，唯一需要注意的是只能用指针变量或指针型数组元素来接收指针型函数的返回值。

b) 指向函数的指针称为函数指针，当把函数名赋给指针变量时，该指针变量的内容就是函数的存储地址。

函数指针的作用主要是把函数作为参数传送到其他函数。如果使指针变量指向不同的函数，将它的值传给被调用函数中的形参时，能调用不同的函数。

3 实验范例

1) 利用指向行的指针变量求 5×3 数组各行元素之和。

① 程序分析：

对二维数组的灵活运用，了解二维数组和指针的多种表达形式。

② 参考源程序：

```
#include <stdio.h>
void fun(int(*p)[3]);
int main()
{
    int a[5][3], i, j;
    printf("请输入5*3矩阵a:\n");
    for (i = 0; i < 5; i++)
        for (j = 0; j < 3; j++)
            scanf("%d", &a[i][j]);

    fun(a);
    return 0;
}
void fun(int(*p)[3])
{
    int i, j, sum = 0;
    for (i = 0; i < 5; i++)
    {
        printf("第%d行元素之和为: ", i + 1);
        for (j = 0; j < 3; j++)
            sum += p[i][j];
        printf("%d\n", sum);
        sum = 0;
    }
}
```

③ 运行结果：



```
请输入5*3矩阵a:
22 3 34
5 67 0
9 8 7
56 8 44
9 22 55
第1行元素之和为: 59
第2行元素之和为: 72
第3行元素之和为: 24
第4行元素之和为: 108
第5行元素之和为: 86
请按任意键继续. . .
```

2) 从键盘上输入 10 个整数存放于一维数组中，用函数实现将 10 个整数按输入时的顺序逆序排列，函数中对数据的处理要用指针方法实现。

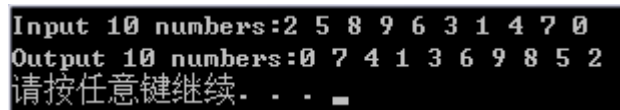
① 程序分析：

和之前使用的冒泡排序思想大同小异，主要是这里运用指针地址传参改变序列。

② 参考源程序：

```
#include <stdio.h>
void fun(int *pa, int t);
int main()
{
    int a[10], i;
    printf("Input 10 numbers:");
    for (i = 0; i < 10; i++)
        scanf("%d", &a[i]);
    fun(a, 10);
    printf("\n");
    return 0;
}
void fun(int *pa, int t)
{
    int i;
    printf("Output 10 numbers:");
    for (i = t - 1; i >= 0; i--)
        printf("%d ", pa[i]);
}
```

③ 运行结果：



```
Input 10 numbers:2 5 8 9 6 3 1 4 7 0
Output 10 numbers:0 7 4 1 3 6 9 8 5 2
请按任意键继续. . .
```

3) 从键盘上输入 10 个数据到一维数组中，然后找出数组中的最大值和该值所在的元素下标。

要求调用子函数 search(int *pa,int n,int *pmax,int *pflag)完成，数组名作为实参，指针作为形参，最大值和下标在形参中以指针的形式返回。

① 程序分析：

使用如下的编程素材：

```
printf("Input 10 numbers:");
printf("Max is:%d\n",...);
printf("Max position is:%d\n",...);
```

② 参考源程序：

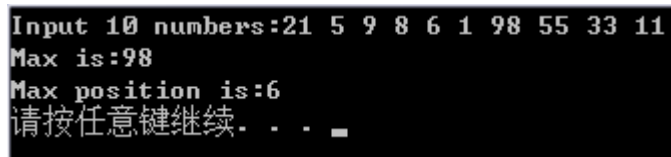
```
#include <stdio.h>
int search(int *pa, int n, int *pmax, int *pflag);
int main()
{
    int a[10], i, max, flag, pmax;
    printf("Input 10 numbers:");
    for (i = 0; i < 10; i++)
        scanf("%d", &a[i]);
    pmax = search(a, 10, &max, &flag);
```

```

    printf("Max is:%d\n", max);
    printf("Max position is:%d\n", flag);
    return 0;
}
int search(int *pa, int n, int *pmax, int *pflag)
{
    int i, *max;
    max = pmax;
    *pmax = pa[0];
    for (i = 1; i < n; i++)
    {
        if (*pmax < pa[i])
        {
            *pmax = pa[i];
            *pflag = i;
        }
    }
    return *max;
}

```

③ 运行结果:



```

Input 10 numbers:21 5 9 8 6 1 98 55 33 11
Max is:98
Max position is:6
请按任意键继续. . .

```

4) 找出一个 2×3 的矩阵中的最大值及其行下标和列下标, 要求调用子函数 FindMax(int p[][3], int m, int n, int *pRow, int *pCol)实现, 最大值以函数的返回值得到, 行下标和列下标在形参中以指针的形式返回。

① 程序分析:

使用如下的编程素材:

```

printf("Input 6 datas:");printf("\n****array****\n");
printf("%4d",...);
printf("\nMax=%d, Row=%d, Col=%d\n",...);

```

输入内容为: 12,23,10,11,19,17.

② 参考源程序:

```

#include <stdio.h>
int FindMax(int p[][3], int m, int n, int *pRow, int *pCol);
int main()
{
    int a[2][3], i, j, row, col, max;
    printf("Input 6 datas:");
    for (i = 0; i < 2; i++)
        for (j = 0; j < 3; j++)
            scanf("%d", &a[i][j]);
    printf("\n****array****\n");
}

```

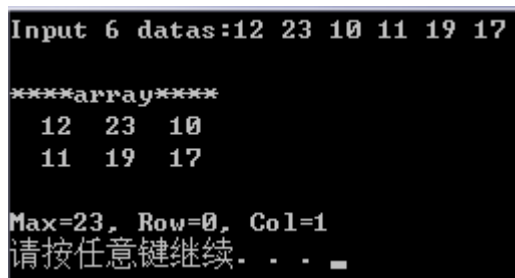
```

    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
    max = FindMax(a, 2, 3, &row, &col);
    printf("\nMax=%d, Row=%d, Col=%d\n", max, row, col);
    return 0;
}

int FindMax(int p[][3], int m, int n, int *pRow, int *pCol)
{
    int *max, i, j;
    max = p[0];
    *pRow = 0;
    *pCol = 0;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
        {
            if (*max < p[i][j])
            {
                *max = p[i][j];
                *pRow = i;
                *pCol = j;
            }
        }
    return (*max);
}

```

③ 运行结果:



```

Input 6 datas:12 23 10 11 19 17

****array****
 12  23  10
 11  19  17

Max=23, Row=0, Col=1
请按任意键继续. . .

```

4 实验内容

- 1) 利用指针函数编写程序，输入 n 为偶数时，调用函数求 $1/2+1/4+1/6+\dots+1/n$ 的值，当输入 n 为奇数时，调用函数求 $1/1+1/3+1/5+\dots+1/n$ 的值。
- 2) 用指向指针的指针的方法对 n 个整数排序并输出。要求将排序单独写成一个函数。 n 个整数在主函数中输入，最后在主函数中输出。
- 3) 输入两个整数，然后让用户选择 1 或 2，选 1 时调用 `max` 函数，输出二者中的大数，

选 2 时调用 `min` 函数，输出二者中的小数，要求用指向函数的指针来实现。

4) 有一个班的 4 个学生，有 5 门课程，分别编写 3 个函数实现下面三个要求：

①求第一门课程的平均分；

②找出有两门以上课程不及格的学生，输出他们的学号和全部课程成绩及平均分；

③找出平均分在 90 分以上或全部课程成绩在 85 分以上的学生。

5) 利用数组和指针，将一个 4×4 的矩阵转置，将数组名作为函数实参，在执行函数的过程中实现矩阵转置，函数调用结束后在主函数中输出已转置的矩阵和矩阵中的最大值及其位置。

6) 编写程序实现任意阶的奇阶幻方的生成，算法自定（可以采用 Merzirac 法或 Loubere 法），有关奇阶幻方的内容由同学门查资料获取。

7) 有两个数组 `a`、`b`，各有 10 个元素，分别统计出两个数组对应元素大于($a[i] > b[i]$)、等于($a[i] = b[i]$)和小于($a[i] < b[i]$)的次数。要求：通过函数调用方式，并分别用数组元素、数组名和指针变量作函数的参数。

实验八 字符串实验

1 实验目的

- 1) 掌握字符串的基本概念;
- 2) 掌握字符数组与指针的关系;
- 3) 掌握字符串的处理函数;
- 4) 进一步熟悉指针作为函数的参数以及返回指针的函数。

2 实验相关知识

1) 字符串的基本概念

a) 字符与字符串

字符是指计算机中使用的字母、数字、符号。每个程序适用很多指令组成的，每个指令由一连串的字符组成。

字符型数字与数值型数字之间的关系：一个字符常量可以看成是一个整数，字符是按其代码形式存储的，因此可以把字符型数据作为整数类型的一种。字符中除了数字还有字母、符号等，而数值型数组范围仅限于数字。

字符串常量是由一对双引号括起来的字符序列，字符串中可以包括字母、数字及各种字符。字符串可以看作是一个指针，它和数组很相似，字符常量可以看作是一个字符数组。

b) 字符数组

存放字符的数组是字符数组，字符数组的定义与其他数组的定义类似，例如：

```
char string[10];
```

字符数组的初始化可以与一般的一维数组的初始化形式相同，例如：

```
char string[10]={'s', 't', 'r', 'i', 'n', 'g', '\0'};
```

字符数组的引用、赋值和其它运算与普通数组相同，必须逐个元素进行。但字符数组可以通过 `scanf` 和 `printf` 函数用 `%s` 输入和输出整个字符串。

字符串数组是特殊的二维字符数组，它的每一行元素都含有字符串结束符 `'\0'`，因此它的一行和前面的一维字符数组一样。

c) 字符指针

字符指针的初始化及赋值：

①在字符指针初始化时，可以直接用字符串常量作为初始值。

如：`char *p="we are family"`。

②在程序中也可以直接把一个字符串常量赋给一个指针。

如：`char *p; p="a program"`。

在给字符指针赋初值后，如果要对该常量字符串进行处理，要保证处理后的字符串长度不能超过初始的字符串长度，否则会修改初始字符串所在区域后面的数据。

2) 字符串的相关函数

`gets()` 字符串输入;

`puts()` 字符串输出;

`strcpy()` 字符串复制;

`strcmp()` 字符串比较;

`strlen()` 求字符串的长度;

`strupr()` 字符串大小写转换;

strlwr() 字符串转换为小写形式;
strstr() 查找指定字符串在给定的字符串中第一次出现的位置。

3 实验范例

1) 设计函数 `char *insert(s1,s2,n)`, 用指针实现在字符串 `s1` 中的指定位置 `n` 处插入字符串 `s2`。

① 参考源程序:

```
#include <stdio.h>
char* insert(char* s1, char* s2, int n)
{
    int j = 0;
    char* ss = new char[100];
    char* tsptr = ss;
    for (int i = 0; i < n; i++)
        *ss++ = *s1++;
    while (*s2 != '\0')
        *ss++ = *s2++;
    while (*s1 != '\0')
    {
        *ss++ = *s1++;
    }
    *ss = '\0';
    return tsptr;
}
void main()
{
    char s1[] = "123456789";
    char s2[] = "1234";
    char* ss = insert(s1, s2, 4);
    printf("%s\n", ss);
}
```

② 运行结果:



2) 用指针法编程, 从键盘上输入多个字符串 (每个串不超过 5 个字符且没有空格), 用 “*****” 作为输入结束的标记。从所输入的若干字符串中, 找出一个最大的串, 并输出该串。要求串的输入以及最大串的查找通过调用编写的函数实现。

① 程序分析:

使用如下的编程素材:

```
printf("****Input strings****\n");
printf("max=%s\n",...).
```

输入内容为 hello apple zone world *****。

② 参考源程序:

```
#include <stdio.h>
#include <string.h>
void find(char *name[], int n, int *p)
{
    int i, m = 0;
    char *pmax = name[0];
    for (i = 1; i < n; i++)
    {
        if (strcmp(pmax, name[i]) < 0)
        {
            strcpy(pmax, name[i]);
            m = i;
        }
    }
    *p = m;
}
int main(void)
{
    char *str[100];
    char a[100][6];
    int i = 0, max;
    printf("****Input strings****\n");
    do
    {
        str[i] = a[i];
        scanf("%s", str[i]);
        i++;
    } while (strcmp(str[i - 1], "*****") != 0);
    find(str, i, &max);
    printf("max=%s\n", str[max]);
    return 0;
}
```

③ 运行结果:



```
****Input strings****
hello apple zone world ****
max=zone
请按任意键继续. . .
```

3) 输入一个长度不大于 30 的字符串, 将此字符串中从第 m 个字符开始的剩余全部字符复制成为另一个字符串, 并将这个新字符串输出。要求用指针方法处理字符串。

① 程序分析:

使用如下的编程素材:

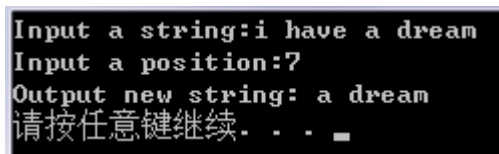
```
printf("Input a string:");
printf("Output new string:");
```

printf("Input a position:").

② 参考源程序:

```
#include <stdio.h>
#include <string.h>
void copystr(char *p1, char *p2, int m)
{
    int n = 0;
    while (n < m - 1)
    {
        n++;
        p1++;
    }
    while (*p1 != '\0')
    {
        *p2 = *p1;
        p1++;
        p2++;
    }
    *p2 = '\0';
}
void main()
{
    int m = 0;
    char str1[30], str2[30];
    printf("Input a string:");
    gets(str1);
    printf("Input a position:");
    scanf("%d", &m);
    if ((strlen(str1)) < m)
        printf("input error!");
    else
    {
        copystr(str1, str2, m);
        printf("Output new string:%s\n", str2);
    }
}
```

③ 运行结果:



```
Input a string:i have a dream
Input a position:?
Output new string: a dream
请按任意键继续. . .
```

4 实验内容

1) 程序分析题。(写出下列程序的输出结果, 然后上机确认结果)

①程序 1:

```
#include<stdio.h>
#include<string.h>
void main()
{
    char *name[]={“Java”,“Basical”,“windows”,“TurboC++”};
    int a,b,n=4;
    char *temp;
    for(a=0;a<n-1;a++)
        for(b=a+1;b<n;b++)
        {
            if(strcmp(name[a],name[b])<0)
            {
                temp=name[a];
                name[a]=name[b];
                name[b]=temp;
            }
        }
    for(a=1;a<n;a++)
        printf(“%s\n”,name[a]);
}
```

②程序 2:

```
#include<stdio.h>
#include<string.h>
void main()
{
    char ch;
    unsigned int i,j,bit,dit,n;
    long int a[20];
    char *str=”a123x456_789”;
    for(i=0,j=0,a[0]=0,bit=0,dit=0;i<strlen(str);i++)
    {
        ch=(str+i);
        if(ch>=’0’&&ch<=’9’)
        {
            a[j]*=bit;
            a[j]+=(ch-’0’);
            bit=10;    dit=1;
            n=j;
        }
        else
    }
```

```

        {
            if(dit==1)
            {
                j++; a[j]=0;
            }
            dit=0; bit=1;
        }
    }
    for(i=0;i<=n;i++)
        printf("a[%d]=%d",i,*(a+i));
}

```

2) 编写一函数，由实参传来一个字符串，统计此字符串中字母、数字、空格和其他字符的个数，在主函数中输入字符串以及输出上述统计的结果。

3) 编写一个函数，在字符串中的指定位置插入一个子串，如在字符串 "abcghi" 中第三个字符后插入子串 "def" 为 "abcdefghi"，如插入位置不合法，原字符串不作任何处理；主函数完成字符串、插入位置、子串的输入，调用所编函数得到插入后的字符串，并输出。

4) 输入一个八进制数的字符串，将他转换成等价的十进制字符串，用 printf 的 %s 格式输出转换结果以检验转换的正确性。例如：输入字符串 "1732"，转换成十进制数的字符串为 "986"。

5) 编写一个程序，将两个字符串连起来，不要用 strcat 函数，并返回连接后字符串的长度。

6) 输入 10 个字符串，然后排序输出。排序的原则由键盘输入的数来决定，为 0，将输入的字符串按整数值大小由小到大排序，否则按字典顺序排序。要求：输入、输出、排序分别用函数实现，主函数只是调用这些函数。

7) 编写一个程序，主程序中输入一行字符串，内有数字字符和非数字字符，调用函数（自己定义及实现的函数），求该字符串中数字子串中最大的数字，并在主程序中显示最大的数字（限定该字符串中数字子串最多不超过 10 个）。如字符串 "a123b345.6x876.1y76t"，该字符串中含有数字子串最大的数字是 876.1。

8) 输入一行包含若干单词的字符串，单词之间用空格分开，要求按单词长短从小到大的次序排序后形成新的字符串输出。（假定字符串中单词个数不包括 10 个，字符串输入并形成单词序列、单词排序、排序后的单词形成新串并输出要求用不同的函数实现，编写主函数完成上述函数的调用。）

实验九 结构和联合实验

1 实验目的

- 1) 掌握结构体类型说明和结构体类型变量、数组、指针的定义方法及使用;
- 2) 学会引用结构体中的成员;
- 3) 掌握利用指向结构体的指针成员构成链表的基本算法;
- 4) 了解联合体类型和枚举类型的说明、变量的定义及赋值方法;
- 5) 了解联合类型变量中各成员的存储结构, 学会引用个成员中的数据;
- 6) 学会正确引用枚举类型常量, 了解如何对枚举类型变量进行操作。

2 实验相关知识

1) 结构型变量的定义和引用

构造类型由相同或不同的数据类型组合而成。用户自己定义的一种用来存放不同类型数据的数据类型结构称为结构型。

a) 结构型的定义:

```
struct <结构体名>
{
    类型 1  成员名 1;
    类型 2  成员名 2;
    ...
    类型 n  成员名 n;
}
```

结构型是一种数据类型, 其中的成员不是变量, 系统不会给成员分配内存。已经定义的某种结构型可以作为一种数据类型, 用来定义变量、数组、指针, 这时才会给定义的变量、数组、指针分配内存。

b) 结构型变量的定义

结构型变量的定义有 3 中方法: 先定义结构型, 然后定义变量、数组、指针; 同时定义结构型变量、数组; 定义无名称的结构型同时定义变量、数组。

c) 结构型变量的引用

当某种结构型的变量、数组被定义以后, 对其只能使用其中的成员, 常见的引用方法如下:

①结构型变量、数组元素成员的引用:

结构类型变量名 · 成员名

或

结构类型数组名[下标] · 成员名

②结构型变量、数组元素成员地址的引用:

&结构类型变量名 · 成员名

或

&结构类型数组名[下标] · 成员名

③结构型变量、数组元素地址的引用:

&结构类型变量名

或

&结构类型数组名[下标]

2) 联合型

联合型数据中所有成员占用相同的内存单元,设置这种数据类型的主要目的是节省内存。

a) 联合型的定义

```
union <共用体名>
{
    类型 1  成员名 1;
    类型 2  成员名 2;
    ...
    类型 n  成员名 n;
}
```

联合型中的每个成员所占的内存单元是连续的,而且都是从分配的连续内存单元中第一个内存单元开始存放,联合体所占的内存长度等于最长的成员的长度。所以,对联合型数据来说,所有成员的首地址都是相同的。

b) 联合型变量的定义

联合型变量的定义有 3 种方法:先定义联合型,然后定义变量、数组;同时定义联合型和变量、数组;定义无名称的联合型同时定义变量、数组。

c) 联合型变量的引用

对联合型变量、数组的引用与对结构型变量、数组的引用方法和要求相同。

C 语言中还有一个重要的规定,联合型数据不能作为函数的参数在函数间传递,也不可以定义某函数返回联合型数据值。但是运行使用指向联合型数据的指针变量在函数间传递联合型数据。

3) 枚举型

将一个有限的变量值一一列举出来称为枚举。

a) 枚举型的定义

```
enum <枚举类型名> {<枚举常量 1>, <枚举常量 1>, ...}[<枚举变量 1>, ...];
```

b) 枚举型变量、数组的定义

枚举变量、数组的定义有 3 种方法:先定义枚举型,然后定义变量、数组;同时定义枚举型和变量、数组;定义无名称的枚举型同时定义变量、数组。

c) 枚举型变量的引用

给变量或数组元素赋值:

枚举型变量或数组元素=同一枚举产量名;

用比较运算符对两个枚举型变量或数组元素进行“大小”比较,可以按照变量或数组元素的枚举常量值(整数)的大小进行。

枚举型变量或数组元素可以进行“自增”和“自减”运算。

3 实验范例

1) 利用结构体编写程序,求任意两复数的和。具体要求如下:

a) 定义表示复数的结构体类型;

b) 定义函数 Add,求两复数的和;

c) 任意两复数及其和在主函数中输入、输出。

① 参考源程序:

```
#include <stdio.h>
struct fushu
```

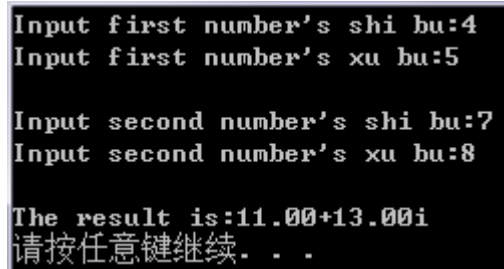


```

{
    float a;
    float b;
};
void Add(struct fushu p1, struct fushu p2, struct fushu *p3)
{
    p3->a = p1.a + p2.a;
    p3->b = p1.b + p2.b;
}
int main(void)
{
    struct fushu f1, f2, s;
    printf("Input first number's shi bu:");
    scanf("%f", &f1.a);
    printf("Input first number's xu bu:");
    scanf("%f", &f1.b);
    printf("\nInput second number's shi bu:");
    scanf("%f", &f2.a);
    printf("Input second number's xu bu:");
    scanf("%f", &f2.b);
    Add(f1, f2, &s);
    printf("\nThe result is: %.2f+%.2fi\n", s.a, s.b);
    return 0;
}

```

② 运行结果:



```

Input first number's shi bu:4
Input first number's xu bu:5

Input second number's shi bu:7
Input second number's xu bu:8

The result is:11.00+13.00i
请按任意键继续. . .

```

2) 已知今天的日期, 编程求出明天的日期

① 参考源程序:

```

#include <stdio.h>
struct Date
{
    int year, month, day;
};
int judgey(struct Date *p){//判断是否是闰年
    int year_r = 0;
    if ((p->year % 4 == 0 && p->year % 100 != 0) || p->year % 400 == 0)
        year_r = 1;
    return year_r;
}

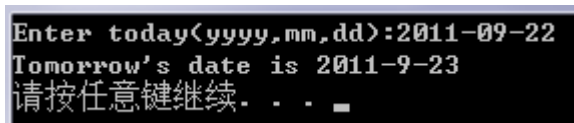
```

```

}
int judgem(struct Date *p){//判断月份
    int day_y;
    int month_h[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    if (judgey(p) && (p->month == 2))
        day_y = 29;
    else
        day_y = month_h[p->month];
    return day_y;
}
void main(){
    struct Date today, tomorrow;
    int judgey(struct Date*), judgem(struct Date*);
    printf("Enter today(yyyy,mm,dd):");
    scanf("%d-%d-%d", &today.year, &today.month, &today.day);
    if (today.day != judgem(&today)){
        tomorrow.day = today.day + 1;
        tomorrow.month = today.month;
        tomorrow.year = today.year;
    }
    else if (today.month == 12)
    {
        tomorrow.day = 1;
        tomorrow.month = 1;
        tomorrow.year = today.year + 1;
    }
    else{
        tomorrow.day = 1;
        tomorrow.month = today.month + 1;
        tomorrow.year = today.year;
    }
    printf("Tomorrow's date is %d-%d-%d\n", tomorrow.year, tomorrow.month, tomorrow.day);
}

```

② 运行结果：



```

Enter today(yyyy,mm,dd):2011-09-22
Tomorrow's date is 2011-9-23
请按任意键继续. . . _

```

3) 请定义枚举类型 `score`，用枚举元素代表成绩的等级，如：90 分以上为优 (excellent)，80—89 分之间为良 (good)，60-79 分之间为中 (general)，60 分以下为差 (fail)，通过键盘输入一个学生的成绩，然后输出该生成成绩的等级。

① 参考源程序：

```

#include <stdio.h>
enum score { excellent, good, general, fail };

```

```

void main()
{
    int n, ss;
    printf("input score:");
    scanf("%d", &ss);
    if (ss >= 90)
        n = 0;
    else if (ss >= 80)
        n = 1;
    else if (ss >= 60)
        n = 2;
    else
        n = 3;
    switch (n){
    case excellent:
        printf("excellent!");
        break;
    case good:
        printf("good!");
        break;
    case general:
        printf("general!");
        break;
    case fail:
        printf("fail!");
        break;
    }
}

```

② 运行结果:

```

input score:88
good!请按任意键继续. . .

```

4) 已知一个无符号的整数占用了 4 个字节的内存空间, 现欲从低位存储地址开始, 将其每个字节作为单独的一个 ASCII 码字符输出, 试用共同体实现交换。

① 参考源程序:

```

#include <stdio.h>
void main()
{
    union zifu{
        int i;
        unsigned char ch[4];
    };
    union zifu x = { 0x12345678 };
}

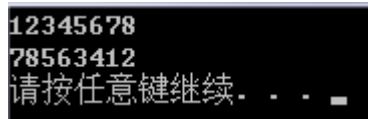
```

```

printf("%lx\n", x.i);
printf("%x%x%x%x\n", x.ch[0], x.ch[1], x.ch[2], x.ch[3]);
}

```

② 运行结果:



```

12345678
78563412
请按任意键继续. . .

```

5) 建立一结构体, 其中包括学生的姓名、性别、年龄和一门课程的成绩。建立的结构体数组通过输入存放若干个学生的信息, 输出考分最高的同学的姓名、性别、年龄和课程的成绩。

① 参考源程序:

```

#include<stdio.h>
void main()
{
    int i, b, n;
    float a;
    printf("请输入学生人数:");
    scanf("%d", &n);
    getchar();
    struct person
    {
        char name[20];
        char sex[10];
        int year;
        float score;
    }stu[20];
    for (i = 0; i < n; i++)
    {
        printf("请输入第%d个学生的名字、性别、年龄及成绩\n", i + 1);
        gets(stu[i].name);
        gets(stu[i].sex);
        scanf("%d", &stu[i].year);
        scanf("%f", &stu[i].score);
        getchar();
    }
    for (b = 0, a = stu[0].score, i = 0; i < n; i++)
        if (a < stu[i].score)
        {
            a = stu[i].score;
            b = i;
        }
    printf("成绩最优秀的是第%d个学生\n", b + 1);
    printf("名字: %s 性别: %s 年龄: %d 成绩: %f\n", stu[b].name, stu[b].sex, stu[b].year,
stu[b].score);

```

}

② 运行结果：

```
请输入学生人数:3
请输入第1个学生的名字、性别、年龄及成绩
张三
男
19
88
请输入第2个学生的名字、性别、年龄及成绩
李四
男
18
90
请输入第3个学生的名字、性别、年龄及成绩
小红
女
17
95
成绩最优秀的是第3个学生
名字: 小红 性别: 女 年龄: 17 成绩: 95.000000
请按任意键继续. . .
```

4 实验内容

1) 程序分析题。（写出下列程序的输出结果，然后上机确认结果）

①程序 1:

```
#include <stdio.h>
struct std {
    int id;
    char * name;
    float sf1;
    float sf2;
};
void main( )
{
    int i;
    char * s;
    float f1, f2;
    struct std a;

    i = a.id = 1998;
    s = a.name = "Windows 98";
    f1 = a.sf1 = 1.18f;
    f2 = a.sf2 = 6.0;
    printf("%d is %s\n", i, s);
    printf("%.2f\t%.2f\n", f1, f2);
}
```

②程序 2:

```
#include <stdio.h>
struct bicycle
{
    long num;
    char color;
    int type;
```

```

};
void main()
{
    static struct bicycle bye[ ] = { {200012, 'B', 18},
                                      {970101, 'R', 12},
                                      {960005, 'G', 30},
                                      {981168, 'Y', 20},
                                      {991688, 'W', 18} };

    int i;
    printf("number color type\n");
    printf("-----\n");
    for(i = 0; i < 4; i++)
        printf("%-9ld%-6c%d\n",bye[i].num,bye[i].color,bye[i].type);
}

```

③程序 3:

```

#include<stdio.h>
struct Key
{
    char *keyword;
    int keyno;
};
void main()
{
    struct Key kd[3]={{"are",123},{"your",456},{"my",789}};
    struct Key *p;
    int a;
    char *str;

    p=kd;
    str=p++->keyword;
    printf("str=%s\n",str+1);

    a=++p->keyno;
    printf("a=%d\n",a);

    p=kd;
    a=p->keyno;
    printf("a=%d\n",a);
}

```

2) 定义一个结构体变量，其成员项包括工作证号、姓名、工龄、职务、工资。然后通过键盘输入所需的具体数据，再进行打印输出。

3) 按上题的结构体类型定义一个有 N 名职工的结构体数组。编一程序，计算这 N 名职工的总工资和平均工资。

4) 设有 N 名考生，每个考生的数据包括考生号、姓名、性别和成绩。编一程序，要求用指针方法找出女性考生中成绩最高的考生并输出。

5) 建立一个学生班的成绩登记表，包括的信息有班号（全班一个），总人数（设为 10 人），制表日期（整个表一个）和每个学生的信息。每个学生包含下列信息：学号、姓名、四门课程的成绩。输入每个人的上述所有信息，统计每个学生四门课程的平均成绩，然后按平均成绩从高到低的顺序输出每个学生的学号、姓名、平均成绩和名次一览表。要求平均成绩保留两位小数；成绩相等者名次应相同。

6) 在题 5 的基础上，增加下列功能：

①找出平均成绩的最高分、输出最高分学生的学号、姓名、平均成绩；

②统计平均成绩在 60 分以下学生的人数，输出这些学生的学号、姓名和平均成

绩。

要求将功能①和②分别定义成函数。

7) 编写一程序计算 2000 年 ~ 2001 学年度第 2 学期 (从 2001 年 2 月 12 日开始至 2001 年 7 月 6 日结束) 有多少天? 要求应用如下结构类型:

```
struct Date {  
    int  day;  
    int  month;  
    int  year;  
    int  yearday;  
    char month_name[4];  
};
```

8) 编写一函数 day_of_year(struct Date *pd) 计算某日在本年中是第几天, 注意闰年问题:

- ① 闰年年号必须能被 4 整除;
- ② 是 100 整数倍的年号不是闰年;
- ③ 是 400 整数倍的年号又是闰年。

调用该函数计算 2001 年 2 月 12 日是该年的第几天, 再计算 2001 年 7 月 6 日是该年的第几天, 这两天数之差即为该学期的天数, 试写成一个完整的可运行源程序。

9) 一个公司, 有若干名员工, 每名员工有姓名、性别、工龄、工资等信息。编程输入员工档案信息和便于工资发放的各种钞票数 (工资为整数, 发放的工资各种钞票限定为 100 元、50 元、20 元、10 元、5 元、1 元, 发放的钞票数张数要求为最少), 要求输出工龄大于 20 年, 工资高于 5000 元的所有男员工信息和工资发放的各种钞票数。(要求输入和输出功能用不同的函数实现, 编写主函数完成上述函数的调用)

实验十 文件操作与图形实验

1 实验目的

- 1) 掌握 C 语言中文件和文件指针的概念;
- 2) 掌握 C 语言中文件的打开、读写与关闭方式;
- 3) 掌握各种文件函数的使用方法;
- 4) 掌握 C 语言中图形设计的主要函数。

2 实验相关知识

1) 文件概述

a) 文件的组织形式

C 语言把文件看作是字符的序列,即由一个字符顺序组成的。根据数据的组织形式,可以将文件分成 ASCII 文件和二进制文件。

①ASCII 文件又称文本文件,它的每一个字节存放一个 ASCII 代码(代表一个字符)。因而便于对字符进行逐个处理,也便于输出字符。但一般占存储空间较多,而且花费转换时间。

②二进制文件是把内存中的数据按其内存中的存储形式原样输出到磁盘上存放。但一个字节并不对应一个字符。不能直接输出字符形式。

b) 文件的处理方法

缓冲文件系统和非缓冲文件系统:

①用缓冲文件系统进行的输入/输出称为高级(高层)磁盘输入/输出。系统自动地在内存区为每一个正在使用的文件开辟一个缓冲区。从内存向磁盘输出数据必须先送到内存中的缓冲区,装满缓冲区后才一起送到磁盘去。如果从磁盘向内存读入数据,则从磁盘文件中将一批数据输入到内存缓冲区,然后再从缓冲区逐个地将数据送到程序数据区(程序变量)。

②用非缓冲文件系统进行的输入/输出称为低级(低层)磁盘输入/输出系统。系统不自动地开辟缓冲区,系统直接对磁盘读写数据。

c) 文件的存取方式

C 语言对文件的操作都是通过文件处理函数实现的。用文件处理函数存取文件存取文件的方式有两种:一种是顺序存取,另一种是随机存取。

d) 设备文件

由于计算机中的输入/输出设备的作用是输入/输出数据,所有把输入/输出设备也看成文件,称为设备文件。

计算机上配备的常用输入设备是键盘,称为标准输入设备;常用输出设备是显示器,称为标准输出设备;还有一个专用于输出错误信息的标准错误输出设备,即显示器。

2) 操作文件的常用函数

C 语言中操作文件的常用函数如下:

`fopen()` 打开文件函数;

`fclose()` 关闭文件函数;

`feof()` 文件尾测试函数;

`fgetc()/fputc()` 字符读/写函数;

`fgets()/fputs()` 字符串读/写函数;

`fread()/fwrite()` 数据读/写函数;

fscanf()/fprintf() 格式读/写函数;
rewind() 文件头定位函数;
fseek() 文件随机定位函数;
ferror() 错误测试函数。

3) C 语言中的常用图形函数如下:

initgraph() 初始化图形系统函数;
closegraph() 关闭图形系统函数;
setcolor() 设置画笔当前颜色及屏幕背景色;
putpixel() 画点及获取屏幕点的颜色;
setlinestyle() 设置线型及画直线;
circle() 画圆;
ellipse() 画椭圆;
rectangle() 画矩形;
drawpoly() 画多边形;
setfillpattern() 设置填充模式和填充颜色;
setfillstyle() 设置填充模式和填充颜色。

4) 图形方式下的文本常见操作函数

a) 视口操作函数

视口是指用于图形输出的屏幕矩形区域, 初始化图形系统时, 视口默认为整个屏幕区域, 视口内的左上角坐标为 (0, 0)。

视口函数有:

- ① setviewport() 设置视口在屏幕中的位置及视口区的大小;
- ② getviewport() 返回当前视口区的信息;
- ③ clearviewport() 清除当前视口区。

b) 图形方式下的文字输出

- ① 输出字符串的字体、大小和方向由函数 settextstyle() 来指定;
- ② 文件操作函数 outtext() 和 outtextxy() 用于输出一个指定的字符串。

c) 屏幕图形的保存和恢复使用函数

- ① getimage() 将屏幕上的某一个矩形区的图像保存到指定的内存中。
- ② imagesize() 计算保存图形所需要的存储空间大小;
- ③ putimage() 恢复函数 getimage() 保存的图像, 并将其显示到屏幕。

3 实验范例

1) 从键盘输入一个字符串, 以感叹号! 作为结束标志, 将其中的小写字母全部转换成大写字母, 然后输出到一个磁盘文件 test 中保存。

① 参考源程序:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(){
    FILE *fp;
    int i = 0;
    char str[100];
    printf("请输入字符串, 并且以感叹号! 结束: \n");
```

```

if ((fp = fopen("test.txt", "w")) == NULL)//打开输出文件并使fp指向此文件
{
    printf("无法打开此文件! \n");
    exit(0);
}
gets(str);
while (str[i] != '!'){
    if (str[i] >= 'a' && str[i] <= 'z')
        str[i] -= 32;
    fputc(str[i], fp);//向磁盘输出字符,将str所指向的字符串输出到fp指向的文件中
    i++;
}
fclose(fp);
if ((fp = fopen("test.txt", "r")) == NULL)//打开输出文件并使fp指向此文件
{
    printf("无法打开此文件! \n");
    exit(0);
}
printf("转换后的字符串是: \n");
puts(fgets(str, strlen(str) + 1, fp));
fclose(fp);
return 0;
}

```

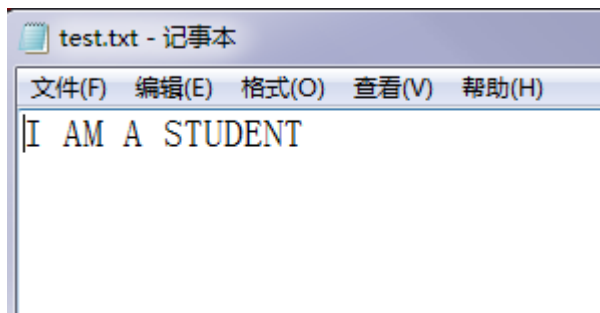
② 运行结果:



```

请输入字符串, 并且以感叹号! 结束:
i am a student!
转换后的字符串是:
I AM A STUDENT
请按任意键继续. . .

```



2) 有两个磁盘文件 A 和 B,各存放一行字母,要求把这两个文件中的信息合并(按字母顺序 排列), 输出到一个新文件 C 中。

① 参考源程序:

```

#include <stdio.h>
#include <stdlib.h>
int comp(const void *a, const void *b)
{

```

```

        return *(char *)a - *(char *)b;
    }
int main()
{
    FILE *fp;
    int i, n;
    char ch;
    char c[512] = { 0 };
    if ((fp = fopen("A.txt", "r")) == NULL){
        return -1;
    }
    else{
        printf("open A.txt:\n");
    }
    for (i = 0; (ch = fgetc(fp)) != EOF; i++){
        c[i] = ch;
        putchar(c[i]);
    }
    n = i;
    fclose(fp);
    putchar('\n');
    if ((fp = fopen("B.txt", "r")) == NULL){
        return -1;
    }
    else{
        printf("open B.txt:\n");
    }
    for (i = 0; (ch = fgetc(fp)) != EOF; i++){
        c[i + n] = ch;
        putchar(c[i + n]);
    }
    fclose(fp);
    putchar('\n');
    n = n + i;
    qsort(c, n, sizeof(char), comp);
    fp = fopen("C.txt", "w");
    for (i = 0; i < n; i++){
        putc(c[i], fp);
        putchar(c[i]);
    }
    fclose(fp);
    putchar('\n');
    return 0;
}

```

② 运行结果:

```
open A.txt
hello world

open B.txt
my first procedure

cddeefhilllmooprrrrstuw
请按任意键继续. . .
```

3) 输入 3 个学生的信息 (含学号、姓名、年龄、3 科成绩、总分), 统计所有学生的总分, 并存入磁盘二进制数据文件 `student.dat` 中。然后在读取该文件, 寻找总分最高的学生并输出该生的所有信息。

①程序分析:

用一个结构体 `student` 来表示学生信息。先创建二进制文件 `student.dat`, 在输入学生信息后就计算总分, 然后写入文件。设置整型变量 `m` 存放最高总分值, 整型变量 `n` 记录最高总分者的编号。寻找结束后, 用库函数 `fseek` 定位至最高总分的学生所在的位置, 读出后显示。

② 参考源程序:

```
#include <stdio.h>
#include <stdlib.h>

struct student
{
    long num;
    char name[20];
    int age;
    int subj[3];
    int score;
}stu;

int main()
{
    FILE *f1;
    int m, n, j;
    if ((f1 = fopen("student.dat", "wb")) == NULL){
        printf("Can't open the file! \n");
        exit(0);
    }
    for (m = 1; m <= 3; m++){
        printf("\nNumber:");
        scanf("%ld", &stu.num);
        printf("\nName:");
        scanf("%s", stu.name);
```

```

    printf("\nAge:");
    scanf("%d", &stu.age);
    printf("\nThree Subject Score:");
    stu.score = 0;
    for (j = 0; j < 3; j++){
        scanf("%d", &stu.subj[j]);
        stu.score += stu.subj[j];
    }
    fwrite(&stu, sizeof(struct student), 1, f1);
}
fclose(f1);
if ((f1 = fopen("student.dat", "rb")) == NULL){
    printf("Can't open the file! \n");
    exit(0);
}
m = 0;
for (j = 0; j < 3; j++){
    fread(&stu, sizeof(struct student), 1, f1);
    if (stu.score > m){
        m = stu.score;
        n = j;
    }
}
fseek(f1, (long)n*sizeof(struct student), 0);
fread(&stu, sizeof(struct student), 1, f1);
printf("\nTop student Number:%ld", stu.num);
printf("\nName:%s", stu.name);
printf("\nThree Subject Score:");
for (j = 0; j < 3; j++)
    printf("%d ", stu.subj[j]);
printf("\ntotal score:%d", stu.score);
fclose(f1);
}

```

③ 运行结果:

```

Number:001

Name:li

Age:18

Three Subject Score:99 80 88

Number:002

Name:liu

Age:20

Three Subject Score:89 86 82

Number:003

Name:wang

Age:19

Three Subject Score:79 89 81

Top student Number:1
Name:li
Three Subject Score:99 80 88
total score:267请按任意键继续. . .

```

4 实验内容

1) 程序分析题。（写出下列程序的输出结果，然后上机确认结果）

①程序 1:

```

#include <stdio.h>
void main()
{
    char ch1, ch2;
    while((ch1 = getchar()) != EOF)
    {
        if(ch1 >= 'a' && ch1 <= 'z')
        {
            ch2 = ch1 - 32;
            putchar(ch2);
        }
        else putchar(ch1);
    }
}

```

②程序 2:

```

#include <stdio.h>
void main()
{
    FILE * point1, * point2;
    point1 = fopen("file1.cpp", "r");
}

```

```

        point2 = fopen("file2.cpp", "w");
        while(!feof(point1))
            fputc(fgetc(point1), point2);
        fclose(point1);
        fclose(point2);
    }

```

③程序 3:

```

#include<graphics.h>
#include<stdlib.h>
#include<conio.h>
void main()
{
    int driver=VGA;
    int mode=VGAHI;
    initgraph(&driver,&mode,"c:\\bc31\\bgi");
    cleardevice();
    setviewport(20,20,570,450,1);
    setcolor(2);
    setbkcolor(3);
    getch();
    closegraph();
}

```

- 2) 用文件的字符串输入函数 `fgets()`，读取磁盘文件中的字符串，并在屏幕上输出。
- 3) 编一程序，读取磁盘上某一 C 源程序文件，要求加上行号后再存回磁盘中。
- 4) 从键盘输入一个字符串，并逐个将字符串的每个字符传送到磁盘文件"test.txt"中，字符串的结束标志为"#"。
- 5) 编一程序，把文本文件 w1.txt 中的数字字符复制到文本文件 w2.txt 中。
- 6) 编一程序，统计一字符文件中字符和数字的个数。
- 7) 建立 5 名学生的信息表，其中包括学号、姓名、年龄、性别、民族、电话号码、Email、以及 8 门课的成绩。要求从键盘输入数据，并将这些数据写入到磁盘文件 studata.dat 中。用 `fwrite()` 函数完成。
- 8) 用 C 语言图形函数画出奥运五环。

