第3章 C语言上机开发环境介绍

◇ 学习要点

- 掌握 Visual C++ 6.0 环境下 C 程序调试、运行方法。
- 掌握 Visual C++ 2010 Express 环境下程序调试方法。
- 掌握 CodeBlock 17.12 环境下 C 程序调试、运行方法。

3.1 Visual C++ 6.0 开发环境

微软公司于 1998 年出品的 Visual C++ 6.0(以下简称 VC6.0)是当时很长一段时间 Windows 平台上最流行的 C/C++集成开发环境之一,即使到现在仍然被众多 C/C++爱好者 所青睐,本节将介绍在 VC6.0开发环境下如何编辑、编译、调试和运行 C 语言程序。

■3.1.1 启动 Visual C++ 6.0 环境

启动 VC6.0 环境常用的方法有 3 种:双击桌面图标,从桌面左下角的"开始"菜单进入,从桌面左下角的"运行"功能中进入。

1. 通过双击桌面图标直接启动 VC6.0 环境

在桌面上找到 VC6.0 的图标,如图 3-1 所示,使用鼠标左键进行双击,即可打开 VC6.0 环境。

2. 从桌面左下角的"开始"菜单进入 VC6.0 环境

(1) 用鼠标单击桌面左下角的"开始"菜单(见图 3-2)。

- (2) 将鼠标指针单击或上移至"所有程序"处。
- (3) 然后鼠标单击 Microsoft Visual Studio 6.0。
- (4) 最后鼠标单击 Microsoft Visual C++ 6.0。

3. 从桌面左下角的"运行"功能中进入 VC6.0 环境

(1) 用鼠标单击桌面左下角的"开始"菜单。



图 3-1 通过鼠标左键双击桌面图标直接启动 VC6.0 环境



图 3-2 从桌面左下角的"开始"菜单进入 VC6.0 环境

(2)将鼠标指针上移到"运行"处,单击鼠标会出现一个"运行"对话框。

(3) 在弹出的对话框中输入"msdev"字样,如图 3-3 所示,然后单击"确定"按钮即会出 现如图 3-4 所示的 VC6.0 编程环境。



图 3-3 "运行"对话框

A ligrosoft Visual Ctt
File Edit View Jocent Broject Build Teels Window Help
II Die Fair Wew Tuser Diolect Baild Tools William Teh
12 2 2 3 12 2 - 2 - IE 2 14
Build (Debug) Find in Files 1) Find in Files 2) Results]
Beady //

图 3-4 VC6.0 窗口

■3.1.2 建立或打开源程序文件

1. 建立新的源程序文件

进入 VC6.0 环境以后,如果需要创建一个新的源程序文件(.cpp 的文件),可以单击 File 菜单项,然后选择 New 选项(或直接按 Ctrl+N 组合键),就会弹出 New 对话框,初始 出现时显示 Projects 选项卡,用鼠标单击 Files 标签,进入 Files 选项卡(如图 3-5 所示)。

在如图 3-5 所示的对话框中选择文件类型。因为是要建立一个.cpp 的源程序文件,所 以就应当选择 C++ Source File 选项。然后需要在 File 的文本框中输入要创建的源程序文 件名(这里假设是 prg1),再在 Location 文本框中输入或选择文件保存的路径(假设为 C:\cppprg),最后单击 OK 按钮。这样就成功新建了一个名为 prg1.cpp 的源程序文件,如 图 3-6 所示。

在图 3-6 中,标题栏上的[C:\cppprg\prg1.cpp]表示刚刚新建的源程序文件名,窗口的 右面大片空白区域就是程序编辑区,程序员可以在此区域编写程序,编写好的程序可以按 Ctrl+F2 组合键或单击工具栏上的磁盘图标来保存。

fer Files Projects Workspaces Other Documents	_	<u>?</u> ×
 Interpretation Interpretation<!--</th--><th>☐ <u>A</u>dd to project:</th><th><u> </u></th>	☐ <u>A</u> dd to project:	<u> </u>
Cursor File HTML Page Con File Macro File Resource Script Besource Template	File prg1 Location:	
SQL Script File Text File	le-tehhhið	<u></u>
	ОК	Cancel

图 3-5 New 对话框



图 3-6 新建文件 prg1. cpp 后的 VC6.0 环境

2. 打开已保存的源程序文件

进入 VC6.0 环境后,如果想打开以前保存了的源程序文件,可以单击 File 菜单项,然后 选择 Open 选项(或直接按 Ctrl+O 组合键),就会弹出"打开"对话框(如图 3-7 所示),可以 打开以前保存的源程序文件。

假设这里是打开源程序文件 C:\cppprg\prg1.cpp(即鼠标双击 prg1.cpp 文件名或单击 prg1.cpp,再单击"打开"按钮),系统将把源程序 prg1.cpp 的内容显示在 VC6.0 环境的 编辑区。此时可以对 prg1.cpp 文件进行重新编辑了。

如果想把 prg1. cpp 的程序内容保存到另外的文件中,此时可以单击 File 菜单项,然后 选择 Save As,就会弹出"保存为"对话框(如图 3-8 所示),可以选择要保存到的文件路径,输

入要保存到的文件名(假设保存到 C:\cppprg\prg2. cpp 文件中),然后单击"保存"按钮即可。此时 VC6.0 环境将变成 prg2. cpp 的编辑环境了。

打开		? ×
查找范围(<u>t</u>):	🔁 cppprg 💌 🗲 🖻	
Debug		
prg1.cpp		
Ct prg2.cpp		
I		
文件名(M):	prgl	打开(0)
文件类型(1):	C++ Files (.c;.cpp;.cxx;.tli;.h;.tlh;	取消
	🔲 以只读方式打开 🗷	
Open <u>a</u> s:	Auto	,
		//

图 3-7 "打开"对话框

🍪 Microsoft Visual C++ - [prg1.cpp]		- II X		
Eile Edit View Insert Project Build To	ols <u>W</u> indow <u>H</u> elp		_ 8 ×		
]12 2 - 2 0 % 6 6 2 - 2	E a 6' 4				
	Y	<u>×</u> ₩ & <u></u>	: 1		
<pre>## #include void mai { int a, printf scanf c = a printf }</pre>	<pre><stdio.h> n () 保存方 保存在 (): cppprg Debug Cpp1 Cpp1.BAK CPp1.OBJ edCpp1.cpp Cpp1.opt Cpp1.dsp Cpp1.dsp Cpp1.dsp Cpp1 医prg1.dsp </stdio.h></pre>		K		
	文件名 (M): prg2. cpp	保存 (S)			
保存类型(I): Text file (*.*) 取消 取消 ↓ Build (Debug) Find in Files 1) Find in Files 2) Results] ↓ ↓					
Ready		Ln 7, Col 8 REC COL OVR	READ		

图 3-8 将 prg1. cpp 另存为 prg2. cpp

■3.1.3 程序的编辑、编译、链接和运行

VC6.0编辑器的编辑功能和 Windows 的记事本很相像,并有许多专门为编写代码而 开发的功能,如关键字加亮、自动完成、自动调整格式等。鼠标和键盘配合使用,可以大大加 快编程速度。

程序编写完毕,单击 Build 菜单项,然后选择 Build 命令或直接按 F7 键,开始编译和连接。但在正式编译之前,VC6.0 会先弹出如图 3-9 所示的对话框,询问是否建立一个默认的项目工作区。VC6.0 必须有项目才能编译,所以这里必须回答"是"。然后在.cpp 文件的目录中会生成与 cpp 源程序文件同名的.dsw 和.dsp 等文件。以后可以直接打开这些文件继续编写程序,不必再重复上面的过程。

∎i croso	ft Visual C++
?	This build command requires an active project workspace. Would you like to create a default project workspace?
	<u>是(Y)</u> <u>否(W)</u>
-	

如果修改完代码后没有保存,这时还会提示是否保存。保存后,随着硬盘清脆的响声, VC6.0下方的白色消息区会显示如图 3-10 所示的内容。它表明编译和连接过程中没有任 何错误信息。

×	Configuration: prg1 - Win32 Debug	7
	Compiling	1
	prg1.cpp	
Ш	L10K10g	
	prg1.exe - 0 error(s), 0 warning(s)	
	Build Debug & Find in Files 1 & Find in Files 2 & Results	

图 3-10 编译信息

如果没有错误,VC6.0将生成与 cpp 源程序文件同名的执行文件(.exe 文件),该执行 文件存放在与源程序文件同一文件夹下的 Debug 文件夹下。此时就可以运行了,按 Ctrl+ F5 组合键或单击工具栏上的"!"图标,程序将在一个新的 DOS 窗口中运行。窗口的最下面 会显示一行"Press any key to continue",这是 VC6.0加上的提示,并不是程序的输出。看 到此条提示时,说明程序已经运行完毕,按照提示按任意键关闭窗口。

如果编译出错,在 VC6.0 环境的下面窗口中会列出错误的位置与内容,并统计错误和警告的个数。在错误信息窗口中,利用鼠标双击错误信息,该错误信息所对应的程序语句所在行将会出现一箭头标识(如图 3-11 所示)。这时,根据错误提示信息修改程序。通常情况下,必须先改掉第一个错误,才能准确修改后续错误。有时仅仅是一个地方错了,就会引起多条错误信息。



图 3-11 错误信息提示窗口

图 3-9 询问是否建立项目工作区

■3.1.4 程序调试方法

初学 C 语言程序设计,往往一看到自己编的程序出现错误就不知所措了。有些读者上 机时,只要程序能够顺利运行,就认为大功告成,根本没想到程序还存在某些隐患。要想不 犯或少犯错误,就需要了解 C 语言程序设计的错误类型和纠正方法。C 语言程序设计的错 误可分为语法错误、连接错误、逻辑错误和运行错误。

语法错误:在编写程序时违反了 C 语言的语法规定。语法不正确、关键词拼错、标点漏 写、数据运算类型不匹配、括号不配对等都属于语法错误。在进入程序编译阶段,编译系统 会给出出错行和相应"出错信息"。可以双击错误提示行,将光标快速定位到出错代码所在 的出错行上。根据错误提示修改源程序,排除错误。

连接错误:如果使用了错误的函数调用,比如书写了错误的函数名或不存在的函数名, 编译系统在对其进行连接时便会发现这一错误。纠正方法同上。

逻辑错误:虽然程序不存在上述两种错误,但程序运行结果就是与预期效果不符。逻辑错误往往是因为程序采用的算法有问题,或编写的程序逻辑与算法不完全吻合。逻辑错误比语法错误更难排除,需要程序员对程序逐步调试,检测循环、分支调用是否正确,变量值是否按照预期产生变化。

运行错误:程序不存在上述错误,但运行结果时对时错。运行错误往往是由于程序的容错性不高,可能在设计时仅考虑了一部分数据的情况,对于其他数据就不能适用了。例如,打开文件时没有检测打开是否成功就开始对文件进行读写,结果程序运行时,如果文件能够顺利打开,程序运行正确,反之则程序运行出错。要避免这种类型的错误,需要对程序反复测试,完备算法,使程序能够适应各种情况的数据。

为了方便程序员排除程序中的逻辑错误,VC6.0提供了强大的调试功能。每当创建一 个新的 VC6.0工程项目时,默认状态就是 Debug(调试)版本。调试版本会执行编译命令 _D_DEBUG,将头文件的调试语句 ifdef 分支代码添加到可执行文件中;同时加入的调试信 息可以让开发人员观察变量,单步执行程序。由于调试版本包含大量信息,所以生成的 Debug 版本可执行文件容量会远远大于 Release(发行)版本。

1. 设置断点

VC6.0可以在程序中设置断点,跟踪程序实际执行流程。设置断点后,可以按 F5 功能 键启动 Debug 模式,程序会在断点处停止。此时可以接着单步执行程序,观察各变量的值 如何变化,确认程序是否按照设想的方式运行。设置断点的方法是:将光标停在要被暂停 的那一行,单击 Build MiniBar 工具栏中的 Insert/Remove Breakpoint (F9)按钮添加断点, 如图 3-12 所示,断点所在代码行的最左边出现了一个深红色的实心圆点,这表示断点设置 成功。如果该行已经设置了断点,那么再次按 F9 键会清除该断点。

2. 调试命令

VC6.0的调试功能极其强大,熟练使用后将如虎添翼。

要进入程序调试,可首先将程序进行编译、连接,当没有错误后,单击 Build 菜单项,然

后选择 Start Debug 中的 Step Into 选项,或直接按 F11 键就进入调试状态,此时会弹出一个 Debug 的窗口,窗口中有许多基本调试命令和各个调试窗口开关。表 3-1 是基本的调试 命令及其图标和快捷键对照表。



图 3-12 设置断点

命令	图标	快捷键	说 明
Go	Ĩ⊒↓	F5	开始或继续在调试状态下运行程序
Run to Cursor	₹}	Ctrl+F10	运行到光标所在行
Stop Debugging	34	Shift+F5	停止调试程序
Insert/Remove Breakpoint	-@	F9	插入或删除断点
Step Into	{ }	F11	进入函数内部单步执行
Step Over	$\mathbf{D}_{\mathbf{f}}$	F10	执行下一条语句,不进入函数
Step Out	ዮ	Shift+F11	跳出当前函数

表 3-1 VC6.0 基本调试命令

在调试模式下,程序停止在某条语句,该条语句左边就会出现一个黄色的小箭头。可以随时中断程序、单步执行、查看变量、检查调用情况。例如,按 F5 键进入调试模式,程序运行到断点处暂停;不断按 F10 功能键,接着一行一行地执行程序,直到程序运行结束。

需要说明的是,如果希望能一句一句地单步调试程序,在编写程序时就必须一行只写一 条语句。

3. 查看变量

单步调试程序的过程中,可以在下方的 Variables(变量)子窗口和 Watch(监视) 子窗口

第3章 C语言上机开发环境介绍

中动态地查看变量的值,如图 3-13 所示。Variables 子窗口中自动显示当前运行上下文中的各个变量的值变量,而 Watch 子窗口内只显示在此 Watch 子窗口输入的变量或表达式的值。随着程序的逐步运行,也可以直接用鼠标指向程序中变量查看其值。例如,在图 3-13 中可以清楚地看到,程序已经为自动型变量 a、b、max 分配了内存,但它们的初始值是随机的。



图 3-13 查看变量

Variables 子窗口有 3 个选项卡: Auto、Locals 和 this。

Auto 选项卡:显示出当前语句和上一条语句使用的变量,它还显示使用 Step over 或 Step out 命令后函数的返回值。

Locals 选项卡:显示出当前函数使用的局部变量。

this 选项卡:显示出由 this 所指向的对象(C语言不用 this)。

如果变量较多,自动显示的 Variables 窗口难以查看时,还可以在右边的 Watch 子窗口 中添加想要监控的变量名。例如,图 3-13 在 Watch1 子窗口中添加了变量"max"。还可以 直接将变量拖动到 Watch 子窗口的空白 Name 框中。添加结束后,该变量的值会被显示出 来。并且随着单步调试的进行,会看到变量 max 的值逐渐变化。如果各变量的值按照设想 的方式逐渐变化,程序运行结果无误,本次开发就顺利结束了。如果发现各变量值的变化和 设想的不一致,说明程序存在逻辑错误,那就需要停止调试,返回编辑窗口,查错并修改 程序。

4. 查看内存

数组和指针指向了一段连续的内存中的若干个数据。可以使用 Memory 功能显示数 组和指针指向的连续内存中的内容。在 Debug 工具条上单击 Memory 按钮,弹出一个对话 框,在其中输入数组或指针的地址,就可以显示该地址指向的内存的内容,如图 3-14 所示。

Cpp1 - Microsoft Visual C++ [break] - [Cpp1.cpp]						
Eile Edit View Insert Project Debug Iools Window Help						
12 🚅 🖬 🕼 2 × 2 × 12 🔊 🕾 🙀 int main() 💽 14						
[Globals] 🔹 [All global members 🖌 🌢 main 🔹 🗟 👻 🎽 🛃 🕛						
#include <st< th=""><th>:dio.h></th><th>-</th><th></th><th></th><th></th><th>-</th></st<>	:dio.h>	-				-
int main()		Me	mory		×	
{			ddraaat 0x0	1018#34		
short a[10]	= {15,22,13,4,50,26 mini	5,17,8,30,12};	18FF34 0F 0	00 16 00 00 01	9	
Short Max,	MIN, I,	00	18FF3A 04 0	00 32 00 1A 0	02	
🛑 max = min =	· a[0];	00	18FF40 11 0	00 08 00 1E 00	0	
⇒ for(i = 1; i ≤ 10; i++) 0018FF46 0C 00 88 FF 18 00 .			18FF4C 89 1	00 88 FF 18 00 13 40 00 01 00	0@	
max = max < a[i] ? a[i] : max: 0018FF52 00 00 A0 0E 43 00C.						
{ max = m	nax < a[i] ? a[i] : m	nax; 00	18FF52 00 0	00 A0 0E 43 0I	0	
{ max = m min = m	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m	nax; nin;	18FF52 00 0	00 A0 0E 43 0I	8	J
<pre>{ max = m min = m min = m</pre>	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m	hax; hin;	18FF52 00 0	00 A0 0E 43 01	ðt.	
{ max = r min = m ↓ ↓ Context: main()	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m	nax; nin;	18FF52 00 0	00 A0 0E 43 00 Value		
{ max = m min = m ∡ Context: main() Name	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value	nax; nin; ▼ ▲ Name □ □ □ □	18FF52 00 0	00 A0 0E 43 00 Value 0×0018ff3	14	
{ max = r min = m ✓ Context: main() Name a[0]	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value 15	hax;	18FF52 00 0	00 A0 0E 43 01 Value 0x0018ff3 15 22	9t.	
<pre>{ max = r min = r </pre> <pre> Context: main() Name a[0] i may </pre>	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value 15 -13108	hax; tin; ▲ Name ↓ ▲ Name	18FF52 00 0	00 A0 0E 43 01 Value 0x0018ff3 15 22 13	14	
<pre>{ max = r min = r</pre>	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value 15 -13108 15 15	hax; tin;	18FF52 00 0 [0] [1] [2] [3]	00 A0 0E 43 01 Value 0x0018ff3 15 22 13 4	4	
{ max = r min = r i Context: main() Name a[0] i max min	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value 15 -13108 15 15 15	hax; in;	18FF52 00 0 [0] [1] [2] [3] [4]	00 A0 0E 43 01 Value 0x0018ff3 15 22 13 4 50	<u>1</u> 14	
{ max = r min = r { Mame a[0] i max min	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value 15 -13108 15 15 15	hax; in;	18FF52 00 0	00 A0 0E 43 01 Value 0×0018ff3 15 22 13 4 50 26 43	<u>1</u>	
{ max = r min = r { Mame a[0] i max min	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value 15 -13108 15 15	hax; in;	18FF52 00 0	00 A0 0E 43 01 Value 0×0018ff3 15 22 13 4 50 26 17 9	<u>14</u>	
{ max = r min = r i Context: main() Name a[0] i max min	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value 15 -13108 15 15	hax; iin;	18FF52 00 0 [0] [1] [2] [3] [4] [5] [6] [7] [8]	Value 0x0018ff3 15 22 13 4 50 26 17 8 30	14	
{ max = r min = r i Context: main() Name a[0] i max min	nax < a[i] ? a[i] : m nin > a[i] ? a[i] : m Value 15 -13108 15 15	hax; iin;	18FF52 00 0 [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]	Value 0x0018ff3 15 22 13 4 50 26 17 8 30 12	<u>14</u>	
<pre>{ max = r min = r ✓ Context: main() Name a[0] i max min for the second seco</pre>	<pre>hax < a[i] ? a[i] : m hin > a[i] ? a[i] ?</pre>		18FF52 00 0 [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] Watch1 / Watch	Value 0x0018ff3 15 22 13 4 50 26 17 8 30 12 Value Value 0x0018ff3 15 22 13 4 50 26 17 8 30 12 Value Va	Watch4 /	

图 3-14 查看内存

总之, VC6.0的调试功能非常强大, 很多高级功能是 VC6.0 特有的, 如图 3-15 所示。

🕫 prg1 - Microsoft Visual C++ [break] - [prg1.cpp]	
Eile Edit View Insert Project Debug Iools Window Help	×
🛛 省 🖬 🕼 🕺 🗞 📭 🖻 🗠 - 오 🖪 🗖 🗟 🦄 gettin	me 🔽 🙀
(Globals) (All global members 🌒 🔶 max	U 🗐 🖬 🛃 🐨 🔽 🚽
#include <stdio.h></stdio.h>	Memory
int max (int a, int b);	<u>A</u> ddress: a
void main() Debug 🗵	
{ inta_b_c: ፲፻፻፪ ፲ ≥ ንንንንንን	0012FF2E 00 00 00 F0 FD 7F CC痕
60 💭 🖾 🖾 🛱 💭	9912FF35 CC CC CC CC CC CC 烫烫烫。
a = 20; b = 30:	8612FF43 CC CC CC CC CC CC 交烫烫.
c = max (a, b);	9612FF4A CC CC CC CC CC CC 烫烫烫.▼
printf ("c = %d\n", c);	Call Stack
1 调试窗口开关	→ max(int 20, int 30) line 17
int max (int a, int b)	main() line 11 + 13 bytes
<pre>c>if (a > b)</pre>	KERNEL32! 77e7ca90()
return (a); 当前语句标志	
return (b);	
}	
	<u>►</u>
Context: max(int, int)	× Name Value
Name Value	⊞ &a 0x0012ff20 ⊞ &b 0x0012ff20
a 20	
	Watch窗口
Auto / Locals > this /	▼ Tatch1 (Watch2) Watch3) Watch4 /
Ready	

图 3-15 VC6.0 调试界面