第一部分　集成开发环境

第一章　C++ 概述

Borland Intl. 公司开发了一系列 Turbo 编译器——从 Turbo Pascal 到 Turbo C，进而到 Turbo C++及 Borland C++。1990 年 5 月面市的 Turbo C++1.0 版是其中颇具革命意义的成员；Borland C++2.0 版则是前者的更新版本，于 1991 年 2 月问世。不难理解，这二者与以前的系列产品又是一脉相承的。

Turbo C++ 与 Borland C++更像 Turbo C 的支撑版本，它们既支持 AT&T C++2.0 版，又支持实时程序设计语言 C 的面向对象版本。上述两点恰恰体现了 Turbo C++的革命性。为了保证与 Windows 3.0 兼容，BorlandC++ 2.0 版在 Turbo C++ 的基础上提供了强有力的资源开发工具 Whitewater。

考察一种新的算法语言，都要提出这样的问题：为什么要开发这种语言？它比现有语言有何优越之处？我们在分析 C++时，同样要问：C++究竟是什么？它在哪些方面超越了 C 语言？

本章从程序设计发展史的角度引出面向对象程序设计(Object Oriented Programming，简称 OOP)的思想，着重介绍 Borland Intl. 公司在这方面的成功探索，指出 OOP 思想从理论到实践的成熟将导致软件工程学的一场革命。

1.1　概念及发展演变

在实际深入剖析 C++之前，有必要考察 C++与其前身——C 语言及其它程序设计语言的关系。本节旨在介绍 C++的起源、用途以及开创程序设计新思维的重要意义。

众所周知，C++思想的提出是基于 70 年代发明的 C 语言。Matin Richards 最早开发出 BCPL 语言，接着 Ken Thompson 在继承 BCPL 的许多优点的基础上发明了比较实用的 B 语言。B 语言直接促进了 C 语言的诞生，这一任务由 Dennis Ritchie 完成，并在 DEC PDP-11 计算机配备的 UNIX 操作系统中最终实现。C 语言凭借其灵活性和高效性，自 80 年代初以来在程序设计界占领了广泛的市场，程序员运用它成功地开发了许多重要的软件产品。

但是，C 语言并不是万能的。随着软件工程规模的扩大，C 语言的缺陷逐渐显露出来。比如，当程序量超过 50 000 行、开发人员达数十个时，系统维护工作量变得相当大，面且系统的整体性难以保证。正因为如此，1980 年，美国 AT&T 贝尔实验室的研究员 Bjarne Stroustrup 为 C 语言扩充了一系列新的功能，他命名为“C with class”。1982 年，正式定名为“C++”。

Bjarne Stroustrup 对 C 语言的功能扩充是革命性的，这是因为 C++ 支持面向对象的程序设计。由于借鉴了面向对象语言 Simula 67 面向对象的性质，因此，C++实质上是这两种强有力的程序设计语言的有机融合。

程序设计从其发展历史来看，大致经历三个阶段，每个阶段对应一种典型的程序设计范例。

(1) 无序态程序设计范例

早期的程序设计范例称作无序态范例。在这种模式中，程序员的所有注意力直接集中在问题求解本身，很少考虑解决问题的方法。其代表性语言是 BASIC 的低级版本，它甚至没有子例程结构。程序员因考虑结构性或可扩展性等问题而手忙脚乱，不得不把精力放在特殊问题上。因此，在无序态范例中只能编写小型程序(几百行以内)。在这一范例中，由于程序大小不同，代码间参数传递的关系变得复杂，无疑将增加调试与测试的难度。

(2) 过程化程序设计范例

无序态范例经过改进以后发展为过程化程序设计范例，主要以 FORTRAN 和 COBOL 这类语言为代表。在这种范例中，程序员可把一个问题分成许多个函数，理论上每个函数可看作一条语句。函数本身被抽象化了。于是，在源代码和数据之间形成一道屏障。所有注意力集中于代码抽象，而非数据抽象。

(3) 结构化程序设计范例

在这个阶段，代码抽象超出函数的范围。结构化程序设计范例不同于无序态范例的函数控制，而类似于过程化程序设计，按照规则把每种控制结构(如 for ，while ，if 等)当作子功能，以便作深入的数据抽象。例如，可以把一个 for 循环看作独立的程序块，因为全部通道口都以同一种方式进出，不允许跳进循环体去执行一些指令然后又跳出来。因此，支持结构化程序设计范例的算法语言必须像 PASCAL 语言所能定的那样，有一套完整的控制结构。其次，这一范例不再忽视数据部分。函数之间的参数传递有一套规则，且只允许存取以输入参数形式接收的数据。此外，不据倡用全局变量作为通信参数来传递。这些限制反映出过程化范例的弱点，因为如果函数可存取或修改全局变量，就难以抽象成一个实体。

大多数结构化语言鼓励程序员定义新的数据类型以便更加准确地描述客观对象。C 语言用 enum 或 typedef 等语句定义简单的数据类型，而复合类型需要某种结构(structure)来定义。事实上，用"结构"来定义单个实体，可以保证数据的一致性，更加接近模型化之后的客观对象。

模块化程序设计是结构化程序设计风格的延伸，此时函数作为模块被分解成更加独立的可编译源文件。每一模块包含一组唯一被自己存取的数据与函数。一组独立的数据与函数可被所有模块存取。所有相似的函数归在一个模块之中，所有定义的函数并不都对模块外部透明，限制全局定义函数的个数可以减少模块与模块间通信参数的个数，从而降低整个系统的复杂性。C 语言正是便用模块化程序设计范例的典范。首先，单个函数使用的数据相对于其它模块是不可见的，从而实现数据隐藏或封装。其次，用户自定义数据类型与原类型不相同，为了支持数据抽象，必须允许程序员为新类型定义内部操作符。但是定义新数据类型及其相关的全部操作符是一个复杂的过程，加之根据基本内部类型定义新类型掩盖了用户定义类型之间的关系，不及从现有用户定义类型中派生出新类型那么简单明了。由于派生类的出现，继承和多态性等概念相伴产生了。这已不是 C 语言所能完成的任务了。而向对象的语言 C++ 具备上述抽象、封装、继承和多态性等特征，从而取代了 C 语言，在程序设计的发展史上树起一块崭新的里程碑。

程序设计范例的演变过程及其具有代表性的计算机算法语言如表 1-1 所示。

表 1-1 程序设计范例的发展及计算机算法语言

结构化程序设计范例	C++ Smalltalk Simula 67 Ada Object C （具有面向对象的性质） Pascal COBOL FORTRAN BASIC 的高级版本（如 TURE BASIC）
过程化程序设计范例	C FORTH Micro-assembler
无序态程序设计范例	BASIC 的低级版本 Assembler

那么，从用途方面讲，C++ 有哪些突出之处呢？

前面已经指出，发明 C++ 的基本目的是帮助维护大型程序（指超过 50 000 行、开发人员达数十个的大型项目）。不过，随着 C++ 本身的发展、完善和扩充，它的这种面向对象的性质可被有效地应用到任何编程任务中。越来越多的程序员用它开发编辑器、数据库、个人文件系统以及通信与接口程序等。由于保持了的高效率，用它来开发高性能、高质量的系统软件也为时不远。

C++ 之所以能完成如此多方面的编程任务，是由于：一是可方便地构造一个由相关对象组成的层次等级树；二是程序易于维护、移植和扩充。

1.2 面向对象的程序设计

C++ 紧密结合了面向对象程序设计（OOP）的思想，在 C 语言的基础上增加了面向对象的特点。当然，C++ 并非这种结合的唯一方式。例如，目标 C（Object C）语言也具有面向对象的特点，不过它属于与前者不同的结合方式；这二者在原理上比较接近，实现方法却不相同。所以，在理解 C++ 概念之前，必须先掌握 OOP 的基本思想。

人们对于 OOP 概念的理解，大多数集中在“OOP 是如何如何优秀”之类的问题上。也就是说，近几年来在程序设计理论方面的探索中，“面向对象”的思想是最有价值、最热门的话题。从程序设计的角度看，面向对象代表一种通过摹仿人类建立现实世界模型的方法（包括概括、分类、抽象和归纳等）进行软件开发的思想体系。对这一概念的理解，Pinson 与 Wiener 提出：面向对象的计算机语言必须具有四种特殊的对象化属性：“抽象，封装，继承和多态性”（见 Pinson 与 Wiener 著，《An Introduction to Object—Oriented Programming and Smalltalk》，1988）。这一定义叙述简单明了，不过还需要作些解释。

(1) 抽象和封装——对象

对象（Object）是 OOP 最重要的性质之一。从概念上讲，对象就是既含数据又含对数据代码的操作的一个逻辑实体。在对象中，有些代码或数据是为该对象所特有的，亦即不能为对象之外的任何东西直接存取。这样，可以有效地防止程序中其它不相关部分无意修改或不正确使用该对象的私有部分。

实际上，对象是类（Class）的实例。类用于描述对象的群体特性。另一方面，类是进行抽象

和封装的基石。

抽象(Abstraction)是指具体事物一般化的过程，对具有特定属性及行为特征的对象进行概括，从中提炼出这一类对象的共性，并从通用性的角度描述共有的属性及行为特征。抽象包括两方面的内容：一是数据抽象，即描述某类对象的公共属性；一是代码抽象，即描述某类对象共有的行为特征。正是通过类，方便地实现了代码抽象和数据抽象，这种结合的方式就是所谓的封装(Encapsulation)机制。抽象和封装可以提高软件的模块化程度，增强代码的重用性。

需要明确的是：对象只是一个用户定义类型的变量；定义一个对象意味着建立了一个新的用户定义数据类型。

(2) 派生和继承

派生(Deriving)是指由基本类导出子类的过程，这个基本类的子类称作派生类。派生是分层次等级进行的，继承(Inheritance)是指一对象获取另一对象之性质的过程。它与按层次分类的思想是面向对象的主要性质。在使用"类"之前，必须针对每个对象定义其所有性质，由于使用了"类"，就只需定义使对象区别于"类"中其它对象的性质。由于派生类的存在，那些与基本类共享的性质能方便地继承下来，这种继承机制使得对象成为基本类的实例。

(3) 多态性

多态性是面向对象的另一个突出性质。其含义是同一个函数名可用于多个相互之间既有差别又相关联的目的。使用多态性，就是为了让函数名变为说明某种行为的通用类。对于不同的处理数据，相应地执行通用类的某一具体实例。例如，对三种不同类型的数据(如 int，char，float)执行进栈、出栈操作。利用多态性，可建立三组进栈 push( )和出栈 pop( )函数，由编译程序根据不同的参数选择相应的函数。若不用多态性，就必须建立三组不同名称的进栈、出栈函数，比如：

```
pushint( );      popint( );
pushchar( );     popchar( );
pushfloat( );    popfloat( );
```

显然，这样做增加了开销。

面向对象的程序设计是一种崭新的程序设计范例，可将其视为一类方法或一种风格。对程序设计而言，重要的是要抓住其核心部分，为此人们采用一系列方法。在程序设计方法学中，比较而言，"面向对象"是一门新课题，富有挑战性，因而也最有发展潜力。

程序设计范例是人们用于解决程序设计问题的思想方法的总和。程序员从中选取最为优秀的方法编写大型程序的代码。一个程序员的观点只代表一种解决问题的方式，程序员有时需要分析自己选用的范例，根据多年编程实践所积累的经验来判断这些范例是否最理想。C++为程序员考察这些面向对象的程序设计范例提供了有益的启示。

面向对象语言必须支持面向对象的程序设计范例。当然，这里所说的"支持"是一个相对概念。人们曾经讨论过"汇编语言支持结构化程序设计，乃是因为结构化代码又可以写成汇编语言"之类的观点。然而，一般说来，汇编语言并不是一种结构化语言。

C++之所以是一种面向对象的语言，是因为它支持面向对象的程序设计范例。作为 C 语言的超集，C++也支持某些早期的范例。因此，C++属于综合性的算法语言。其它许多语言也有这种性质，如 Apple 公司的 Object Pascal，Borland 公司的 Turbo Pascal 5.5，Neon 公司的 Forth Objects 以及 Logo 语言的变种 Objective Logo。事实上，绝大多数程序设计语言存在或正在使用面向对象的版本(参见 Peter Wenger 著文，《Learning the Language》，1989)。

1.3 从 C 到 C++

C++是 C 的超集。C++保存了 C 的所有组成部分，这样，C 编译的程序可以在 C++环境下运行，基于 K&R C 标准与 ANSI C 标准的程序在 C++环境下产生同样的出错信息和警告信息。经 Turbo 2.0 编译未出错的程序由 Turbo C++或 Borland C++编译时将正常通过。

为实现以上目的，Turbo C++及 Borland C++根据扩展名区分 C 与 C++源程序：当 IDE(集成开发环境)中 Options|Compiler 菜单的 C++ Always 选项无效时，程序源文件扩展名为.C；C++的则是.CPP。

C++与 ANSI C 共享一套通用的规则。ANSI C 的大部分扩展内容来源于 C，如函数说明格式和强类源使用。C++则扩展到包容 ANSI C 的特征以保证最大限度的兼容性。二者的共同点是主要的，差别是次要的，列述如下。

1.3.1 函数原型

原 K&R C 定义通常根据变量大小说明一个变量，根据返回类型说明一个函数。一旦函数说明出错，它将被指派成缺省类型 int(即整型)。这是因为，返回整型值适合于 CPU 寄存器；而舍弃寄存器左侧值不会引起处理器错误。

C++与 ANSI C 都强调高级的类型检查。程序员因而可以避免混淆没有作兼容分配的存储类型(C 中隐含类型转换，同一表达式中可使用字符型、整型与浮点型)，使用起来更加方便。

C++允许在定义函数之前，先说明其类型，这就是所谓的"函数原型"。K&R C 忽视了这一点，ANSI C 和 C++则特别强调原型的使用。其格式是，全部参数名和类型出现在同一行上。例如：

```
unsigned func(signed a,int b);
```

若不想说明函数名和函数类型，可用省略号。例如：

```
void f1(int a,...);
void f2(...);
```

f1()表示函数使用一个整形参数和其它任意类型参数；f2()表示函数接受任意类型的参数。

1.3.2 关键字 void

C++沿用 ANSI C 的关键字 void。最初，函数缺省返回类型为整型，如函数说明 fn()与 int fn()是等价的。

void 的原意指没有返回值的函数。如 void fn()(缺省类型仍为 int)。编译器据此确定函数是否正确调用了。void 的扩展含义指无参数的函数。例如，在 ANSI C 中，int fn(void)表示无参数的函数；int fn()则等价于 int fn(...)，表示接受变长变元参数的函数。在 C++中，int fn()等价于 int fn(void)，不过后者增强了程序的兼容性。

指针变量可定义为 pointer to void(空指针)。它不能作递增、递减或重指运算，但可与其它任意类型作兼容性分配。这是一个"万能"指针，用于程序没有确定指针所指的内容时存储地址。

此外，void 可用作转换符，表示运算结果被有意舍弃。例如：

```
int fn(char x);
fn('z');
```

若忽略函数的返回值，应将第二行改写成：

```
void fn( 'z' );
```

1.3.3 变数与常数

ANSI C 定义了存储类 volatile 与 const。volatile 变量表示该变量可随时改变，因而编译器将不给出优化值。这意味着该变量代表一个内存映像设备或者它是由编译器无法知道的某个特殊任务存取的全局变量。

实际上，每使用一次 volatile 变量必须从内存调入。一般地，一小段 C 程序多次使用同一变量时，编译器将从寄存器而不是内存中读取变量的值。将变量说明为 volatile 将丧失这一功能。

const 变量则刚好相反。其值一经说明将不再改变。编译优化器可对 const 变量设定所希望的值。用户不能更改该变量的值，而且 const 变量既不能出现在赋值号左边，又不能增加或减少。对指针而言，指针本身或指针的值可以使用存储类 volatile 和 const。它们与简单类型有相同的约定。C++遵循 ANSI C 标准的这个约定。

1.3.4 寄存器变量

说明一个寄存器存储类变量要求将该变量存放在寄存器里(而不是 RAM 里)，以便实现快速存取。但如果编译器不能执行，它可以忽视寄存器变量说明。

寄存器变量允许程序员帮助编译器充分使用寄存器。函数频繁用到某个变量，把它说明成寄存器变量可以节省时间。

关键词 register 是原 C 语言的一部分，但很少在基于 DOS 的编译器中实现。Borland C++约定，在 8086/8088 CPU 系列的 SI 和 DI 寄存器中存放两个寄存器变量。因此，函数调用过程中寄存器必须保存起来。任何函数在把 SI 和 DI 用于寄存器变量时，必须做到：在进入函数前保存寄存器值，在退出前恢复寄存器值。

函数变元也可定义成 register 型。例如，下列说明是合法的：

```
unsigned fn(register int a);
```

Turbo C++和 Borland C++环境下寄存器在函数调用之初被压进栈，在函数开始时读入寄存器。上面的语句等价于：

```
unsigned fn(int atemp){
    register int a=atemp;
}
```

然而，C++不主张使用寄存器变量说明。编译优化器偏向于用高速缓存器存取常用变量，把寄存器用作特殊定义的寄存器变元取消了编译器的优化功能。在正常情况下，运行结果与采用优化器时是相近的；反之，若没有作优化选择，效果将很差。

1.3.5 静变

ANSI C 标准包含少量"静变"(Quiet Changes)。所谓"静变"，是指用于表现特定构成变化的方式；也就是说，同样的源代码在原版 C、ANSI C、Turbo C++、Borland C++的编译器中产生不同的目标码，但两者都没有出错信息或提示信息。

静变产生的原因乃是不同的编译器解释这些构成的方式是不一样的。在 ANSI 标准委员会协商之前，编译器之间没有统一约定，从一个 C 编译器转到另一个 C(或 C++)编译器时产生了大量静变。

例如，原版 K&R C 以双精度方式执行浮点运算。这是因为最初用 C 语言编程的 PDP—11 机按双精度方式运算，其速度与单精度方式一样快。

因此，许多工程项目程序员偏向于以双精度方式运算。ANSI 标准规定，单精度数之间的算术运算以单精度方式进行。这样，如果运算过程中用到双精度数，在环境下编译时将不产生任何警告信息。

第二类静变是指 16 进制返回值的解释性差异。Turbo C 2.0 之前，一个 16 进制字符限于 3 位数。与 ANSI C 兼容的 Turbo C 2.0 则没有这类限制。例如，"\0x0071"在 Turbo C 2.0 中当作单一字符"\0x71"，但在 Turbo C1.5 及其它先于 ANSI C 的编译器中当作双字符"\0x007"和"1"。

这些静变很少导致危险，其危害只限于不产生出错信息。由于继承了 ANSI C 标准，Borland C++ 将会遇到同样的静变问题。

1.4 从 Turbo C 到 Turbo C++ 1.0

Turbo C++1.0 是跟 Turbo C 2.0 相对独立的新成员，但二者具有不可分割的联系。Turbo C++沿用 Turbo C 编译器的命令行版本以及交互式集成开发环境（Integrated Development Environment，简称 IDE），但功能上有较大的扩充。其次，Turbo C++除了支持键盘输入以外，还支持鼠标器操作。下面介绍 Turbo C++在非语言方面的扩充功能。

1.4.1 交互式集成开发环境(IDE)

IDE 支持在同一环境下直接进行编辑、编译、连接、运行与调试，而不必经过从编辑器到编译器以及调试器的切换。在编辑与调试程序时可以一次装入多个文件，由多重覆盖窗口显示。窗口切换既可通过功能键，又可用鼠标器操作。

程序员可通过键选菜单调整编译与连接过程，故不必记住命令行编译程序的开关参数(编译时仍可直接运行 TCC.EXE 文件)。菜单项的变换更适合于鼠标操作。

Turbo C++在边框右侧和下侧增加了滑动杆。边角上的编放盒可将窗口切换成全屏幕方式或整屏关闭。屏幕顶端的主菜单既可由"Alt+首字母"作为按键选取，又可由鼠标器选取。

添加用于数据入口域的命令栈表进一步强化了 IDE 的功能。选择 File|Open 项时，出现一根亮条线，供用户输入文件及其路径。此时键入下行键，屏幕将显示最近编辑过的十个文件的下拉式栈表。这个功能与 Turbo C 的 Pick 选项差不多，只不过扩展到接受全部输入入口域。

Turbo C++的交互式调试器类似于 Turbo C。唯一增加的命令选项是 Inspect，命令行 Evaluate 仍然保留下来，但二者的作用不一样。Inspect 同时显示变量的值与类型，可以有效地用于检查一个结构或含有多个成员的类对象。这时，每个成员的名称、类型及值都显示出来，程序员可以迅速掌握混合的甚至复杂的数据元素的总体情况。

此外，Transfer 菜单项帮助程序员方便地存取其它的程序。该项与 DOS Shell 项相似，允许暂时脱离运行其它的程序。不过它比 DOS Shell 项方便得多，可用热键直接进入程序体。其次，程序运行时，可标明全路径和参数；而且，切换宏允许指明被编辑文件名与生成的可执行文件名。这样，Transfer 功能就可以全自动运行了。

1.4.2 VROOMM

Turbo C++ 支持"虚拟实时的面向对象的存储管理"(Virtual Rum time Object Oriented Memory Manager，简称 VROOMM)，这点也是它相对于 Turbo C 的非语言方面的优越之处。VROOMM 是一种成熟的多重覆盖管理技术，允许建立超过 640K 的大型程序。

多重覆盖的原理很简单。一段程序代码，如一个函数，实际运行只需在 RAM 中进行。程序的大部分经常用到，故在整个运行时间驻留内存，从而避免运行出错。其次，装载覆盖部分的程

序段在程序运行时必须驻留内存。由于覆盖部分程序并不经常用到，可将它放在磁盘里，只有当需要时，才调进内存。由于该部分在装载时重写内存，故称为"覆盖"。

VROOMM 不仅大大简化程序员安排覆盖程序的工作量，而且利用小型覆盖可以增强程序的运行效果。这种增强的程序"颗粒化"(Granularity)技术使得应用程序开辟 50K 的覆盖区执行 25 行语句的功能成为现实。Turbo C++因采用 VROOMM 技术而具备相当可观的处理能力，并为用户程序保留了内存空间。

1.4.3 支持汇编语言

Turbo C++大大减轻了汇编程序员的工作。首先，它与 Turbo C 一样，支持伪寄存器，每一个 8086 寄存器对应一个内部变量名。其次，Turbo C++包含中断类型函数——由 int(而不是 call)指令调用。它可用于编制起停驻留程序。上述两个关键字减少了汇编语言程序量。

Turbo C++ 的内部汇编语言要求用 Turbo Assembler 工具生成目标文件(.OBJ)以及一个不需独立汇编器支持的类 PASCAL 风格的内部转换机构。这样，程序员可以将 C 或 C++模块与汇编语言模块连接起来。

1.5 Borland C++ 2.0 简介

Borland Intl. 公司一直致力于倡导程序设计领域的新潮流。1990 年 5 月推出该公司第一份面向对象的程序设计软件包——Turbo C++1.0，继承并发展了 Turbo C 集成开发环境的优良特性，扩充了面向对象的新思想和设计方法，因而成为一时非常受欢迎的面向对象程序设计软件包。

Microsoft 公司不甘示弱。在同一时间推出了具有面向对象特性的 Windows3.0，在全球微机软件界引起一场革命。许多新的适用于 Windows3.0 的软件纷纷面世。众所周知，软件界的竞争是异常激烈的，落后、过时就意味着被淘汰　基于兼容性和新颖性方面的考虑，Borland Intl. 公司推出了适合于开发 Windows 应用软件的 C++版本——Borland C++。

Borland C++2.0 向上完全兼容于 Turbo C 及 Turbo C++1.0，具有以下新特色：

① C++：提供 C++ 编程的全部功能(AT&T C++2.0 版的实现)，支持 C++1.2 版流，并包含一个类库。

② 实现最新的 ANSI C 标准。

③ 支持 Microsoft Windows 3.0 应用程序开发。扩充了包括资源编译器和资源开发工具在内的新功能。

④ 预编译的头文件：可缩短编译时间。

⑤ 实式和保护模式的编译器。

⑥ 类库包含了集合、数组等类型。

⑦ 联机求助系统扩充了 Windows API。

⑧ 新的 Borland IDE：提供对计算机上全范围内程序和工具的访问。包括：

- 鼠标器支持
- 多重覆盖窗口
- 多重文件编译
- 支持内部汇编码
- 内部汇编器
- 集成调试器

· 大缓冲区的恢复(undo)和重复(redo)功能。

⑨ VROOMM:简单方便地实现代码覆盖。

⑩ 新 IDE 的联机访问。

⑪ 联机快速文本求助:用模拟程序测试每一个函数。

⑫ 独立的库函数:包括堆检测函数和完整的复数与数学函数集。

⑬ 对 -S 选择的扩展:C 源码作为注释可加到结果汇编码上。

⑭ 远程对象和大容量数组。

⑮ 替补. CFG 文件。

⑯ 命令行编译的响应文件。

⑰ 快速完成复杂的算术运算。

第二章　集成开发环境简介

Borland C++ 2.0 是一个内容丰富的 C++ 开发软件包，提供了命令行编译器 集成开发环境软件 侦错软件 性能测试软件和汇编程序。对于 C++ 程序员，最重要的，也是 Borland C++ 2.0 中功能最强的软件，称为集成开发环境软件。集成开发环境译自英文的 Integrated Development Environment，简称 IDE，是集编辑、编译、连接、调试于一体的全屏幕、多窗口、具有菜单结构的用户应用程序开发环境。这个环境在语言上完全兼容命令行方式对 C++ 的支持，但使用起来比命令行方式容易得多、效率高得多，目前大多数 Borland C++ 2.0 的用户都是在这个环境下开发软件的，本章将向用户介绍集成开发环境的使用方法。

这一章将分几个方面讨论 Borland C++ 2.0 的集成开发环境。2.1 节是集成开发环境的简单介绍，2.2 节列出了菜单的结构。2.3 节列出了用户可以使用的热键。2.4 节重点介绍编辑窗口和编辑命令。

对初次接触 Borland C++ 2.0 的用户，应该首先阅读本章，并争取在计算机上进行操作；有经验的用户和已使用过 Borland C++ 2.0 集成开发环境的用户，可跳过本章直接进入下面章节。

2.1　集成开发环境入门

本节介绍集成开发环境的启动、退出和集成开发环境的各个组成部分。

2.1.1　启动集成开发环境

启动集成开发环境即启动 Borland C++ 2.0 软件包中的 BC.EXE 软件。如使用 Borland C++ 2.0 的自动装载程序机，BC.EXE 被安装在\BORLANDC\BIN 目录中。用户应修改 DOS 环境变量 PATH，使 DOS 能够找到 BC.EXE。

启动集成开发环境最简单的方法是在 DOS 提示符下键入

C：BC

用户就可以在显示器上看到一个全屏幕窗口环境。

BC.EXE 对机器的硬件和软件环境都有一定的要求，用户很少碰到这方面的问题，如果碰到了可以参考 Borland C++ 2.0 的用户手册或阅读软件包中的 README 文件。

BC.EXE 可以有若干选择项，这些选择项对提高 BC.EXE 的性能是有效的、但并不是必要的。一个带选择项的启动命令是

C：BC [option [option...]][filename]

option 有以下几种：/B，/D，/E，/H，/L，/M，/P，/RX，/S，和 /X，option 在命令行中可以没有、有一个或多个；filename 有两种，可以是后缀为 .PRJ 的工程文件，也可以是一个或多个源程序文件名。

下面对 option 的几个选择项逐一介绍：

(1) /B 选择项和 /M 选择项

/B 选择项使 Borland C++ 2.0 的集成开发环境启动后自动装入工程文件或命令行中的源文件，然后，编译、连接所有的文件生成可执行程序，最后自动退出集成开发环境返回操作系统。编译、连接信息，包括错误信息，送到标准设备输出。例如：

BC /B hello.prj

BC /b hello.c

工程文件是在集成开发环境中生成的，本章后面予以介绍。

/M 选择项非常类似 /B 选择项，不同之处在于它仅编译 连接修改过的源程序文件。

(2) /E 选择项和 /X 选择项

集成开发环境在编译、连接等过程中，会出现低端 640 KB 内存不够用的情况。此时集成开发环境会把部分数据交换到磁盘上。使用 /E 选择项，可以让集成开发环境把数据交换到扩充内存（Expanded Memory），使用 /X 选择项，数据交换到扩展内存（Extended Memory）。由于存储器的存取速度远高于磁盘存取速度，使用这两个选择项都可以大大提高 BC.EXE 的性能。

使用 /E 选择项的句法为

BC /E[=N]

N 是 BC.EXE 使用的扩充内存页数，每页为 16 KB。

使用 /X 选择项的句法为

BC /X[=[R][,N]]

R 是集成开发环境留给其它应用程序的扩展内存千字节(KB)数，N 是用于数据交换的千字节(KB)数。

(3) /RX 选择项

用户机器如果设置了虚拟盘且打算把虚拟盘用于集成开发环境的数据交换，可使用/RX 选择项，其中 X 为虚拟盘盘号。例如，用虚拟盘 D 作为数据交换，可键入

BC /RD

(4) /D 选择项

如果用户有双显示器硬件，用 /D 选择项可以使用户程序的输出出现在活动显示器上，集成开发环境出现在非活动显示器上。活动显示器和非活动显示器的设置可参考 DOS 和有关的硬件手册。

(5) /L 选择项

用户使用液晶显示器时使用 /L 选择项。

(6) /P 选择项

如果用户程序修改 EGA 调色板，需要用 /P 选择项，否则集成开发环境屏幕和用户程序输出屏幕无法正常切换，/P 选择项在每次屏幕切换时都保留或恢复相应的 EGA 调色板值。

(7) /H 选择项

显示 BC.EXE 命令行帮助信息。

2.1.2 退出集成开发环境

选择 File 下拉菜单中的 Quit 项。具体方法为：按 ALT-F 键，再按 Q 键。按热键 ALT-X 也可以退出集成开发环境。

2.1.3 集成开发环境的组成部分

集成开发环境主要由以下几个部门组成：菜单，热键，对话盒，窗口和状态行。

(1) 菜单

Borland C++ 2.0 集成开发环境的菜单分三种：主菜单，下拉菜单和弹出式菜单，如图 2-1 所示。

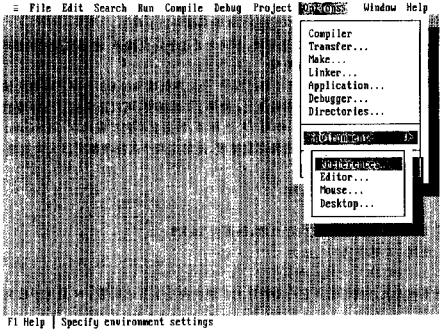

图 2-1　菜单的例子

屏幕最上面的一行是主菜单，选中主菜单后新出现的菜单是下拉菜单，下拉菜单项被选中后，又弹出下一级菜单，称为弹出式菜单。

把鼠标移到菜单项上，按鼠标左键可以选中这个菜单项，如果移到菜单区域外按鼠标左键可以关闭这个菜单。用键盘也可以选择菜单项，一个菜单被激活后，有一个高亮度条，用光标键可以移动它，移到要选择的菜单项后，回车即选中。

主菜单可以用 F10 键激活，也可以联合按 ALT 和主菜单项的头一个字母，直接打开相应的下拉菜单。主菜单中，最左边的菜单项称为系统菜单项，可以用 ALT-SPACE 打开。

用键盘的 ESC 键可以依次退出各级菜单。

(2) 热键

许多菜单项都可以借助于一个或几个按键的组合打开，这些键称为热键。对于熟悉 Borland C++ 2.0 集成开发环境的用户，利用热键输入命令更加简捷有效。有热键的菜单项都在该项的右端显示出了热键的提示。例如 File 菜单中的 Quit 项，热键为 ALT-X。使用热键等效于用菜单输入命令。

(3) 对话盒

后面有省略号"..."的菜单项被选中后会打开一个对话盒。对话盒用于接收用户的多项选择和字符串的输入。对话盒由单选按钮、复选方框、操作按钮、输入盒、列标盒等组成。图 2-2 是一个典型的对话盒。

对话盒一般都有 OK，CANCEL 和 HELP 三个操作按键，OK 代表确认该对话盒中的内容，CANCEL 表示对当前对话盒的修改无效并关闭对话盒，选中 HELP 则显示关于该对话盒的帮助信息。

选中操作按钮有三种方法：

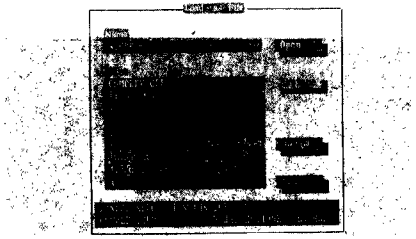

图 2-2　典型的对话盒

① 移动鼠标到操作按钮上，按鼠标左键。

② ALT 加上操作按钮的高亮度字母。如 OK 键可以用 ALT-K 选中。

③ 用 TAB 键或 shift-TAB 使操作按钮加亮，再按回车键。

复选方框是一些开关量的选择。[X] 表示该项被选中，[ ] 代表没有选中。选择或关闭一个复选方框有三种方法：

① 用鼠标直接在方框中按鼠标左键。

② ALT 加复选方框说明的高亮度字母。

③ 用 TAB 或 shift-TAB 移动光标到对应的复选方框后按 SPACE 键。有时复选方框是成组的，这时需用 TAB，shift-TAB 移动光标到所希望的组，然后再用光标键在组内移动。

单选按钮都是成组出现的，一组单选按钮有且仅有一个被选中。选择单选按钮有三种方法：

① 用鼠标移到单选按钮的（）中按左键。

② ALT 加单选按键说明中的高亮度字母。

③ 用 TAB，shift-TAB 移动光标到本组单选按钮中，再用光标键移动光标到要选择的单选按钮项上，按 SPACE 键。

对话盒的输入盒是用来输入正文的，用鼠标，或 ALT 加输入盒说明中的高亮度字母，或用 TAB，shift-TAB 都可以激活输入盒。某些输入盒右边有一个下箭头标志，用鼠标在该标志上按左键可以打开一个历史输入列表，鼠标可以挑选某个历史输入作为当前输入。

列表盒可以用鼠标或键盘激活，用户可以选取列表盒中的某一项。最典型的列表是文件名列表，盒中列出了所有的文件名，用户可选取某一文件进行操作。

（4）窗口

集成开发环境中许多工作都是用窗口完成的，例如编辑、工程管理。窗口是屏幕的一个区域，可以打开、关闭、移动、放大、缩小和覆盖。

集成开发环境中可以有多个窗口同时打开，但仅有一个窗口是活动的，用户的操作是针对活动窗口的。活动窗口的边框由双线组成，其它窗口采用单线边框。活动窗口不会被覆盖。

图 2-3 是一个典型的窗口，它有边框，四个角，标题，窗口关闭图标，放大图标，窗口号，滚动条，修改标志，行号和列号。有些窗口相对简单一些，例如除了编辑窗口，都没有修改标志和

行列号。

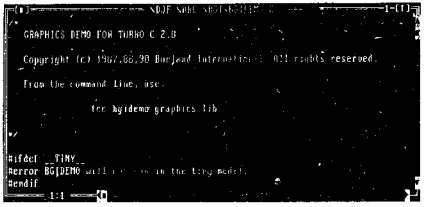

图 2-3　窗口的典型例子

窗口的创立有几个途径：

① 编辑一个新文件或打开一个已有的文件。

② 在 Window 菜单中，利用菜单项激活一个窗口，如 Message 窗口，Watch 窗口。

③ 集成开发环境在编辑、连接过程完成后自动激活 Message 窗口。

④ 求助时集成开发环境打开 Help 窗口。

所有的窗口都是可以关闭。用鼠标在关窗口团图标上按左键，或用菜单命令 Window close，还可用热键 ALT-F3，都可以关闭当前被激活的窗口。求助窗口可以用 ESC 键关闭。

激活一个窗口可以有以下几种方法：

① 在要激活的窗口内按鼠标键。

② 联合按 ALT 键和欲激活窗口的窗口号，此方法仅能打开前 9 个窗口。

③ 用菜单命令 Window|List 或 ALT-0 打开窗口列表，在列表中选取一个窗口为活动窗口。

④ 用 Window|Size/Move 或热键 F6 依次按顺序激活窗口。

所有的窗口都可以移动：

① 用鼠标拖曳窗口的边框。如果要移动的窗口已被激活，不要拖曳窗口的四个角。

② 用 Window|Size/Move 或热键 F5 可以移动激活的窗口，用光标键移到合适位置后按回车。

大多数窗口可以放大或缩小，方法是：

① 用鼠标拖曳被激活窗口的四个角。

② 鼠标在窗口的标题横杠上快速连按两次。

③ 在窗口的放大图标上按左键。

④ 选择 Window|Size/Move 或热键 F5 后，用 Shift 和光标键的组合放大和缩小活动窗口，最后按回车键确认。

⑤ 选择 Window|Cascade 或 Window|Title 可以重新安排编辑窗口的大小和位置。

窗口的滚动条是给鼠标用的，鼠标在滚动条两端的箭头上按左键，可以逐行逐列移动窗口中的内容，在滚动条中间按鼠标左键可以多行多列地移动窗口中的内容。

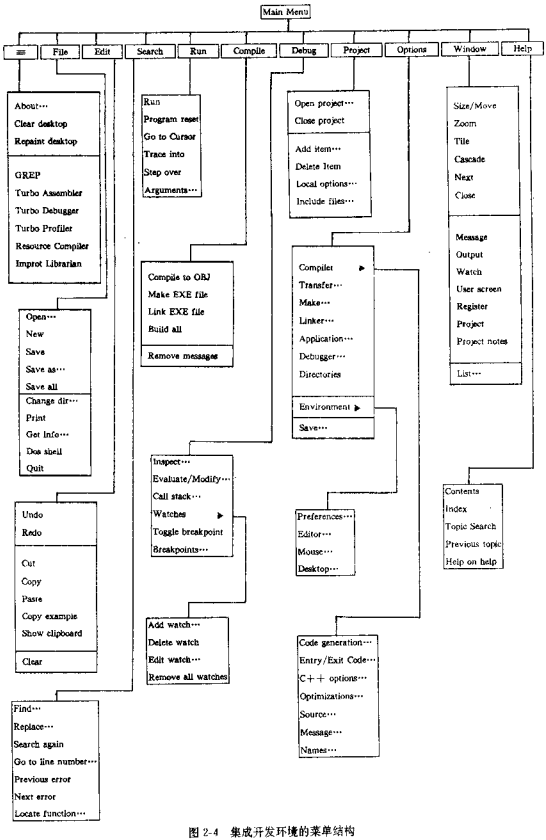

图 2-4　集成开发环境的菜单结构

只有编辑窗口才有修改标志的行、列号，当被编辑的文件修改且没有存盘的情况下，修改志才出现。行、列号是当前光标在被编辑文件中的行号和列号。

在编辑窗口中，窗口的标题是被编辑文件的文件名，在其它窗口中，是窗口的名字。

(5) 状态行

状态行位于屏幕的底部，用于提示用户的常用热键和菜单项的功能，有时也显示集成开发环境的当前操作过程。状态行在用户执行不同的操作时，内容是变化的。对于有鼠标的用户，状态行的各项提示还可以作为鼠标的输入。例如，状态行中有一项 F9 Make，用鼠标选择该项，等效于选择 Compile|Make。

2.2 集成开发环境的菜单

图 2-4 列出了集成开发环境的树状菜单结构。每个菜单由若干菜单项组成，如果菜单项的最右端有键的组合提示，表示该键的组合等效于选择这个菜单项，称这种键的组合为热键；如果菜单项右端有省略号"..."，说明选中此菜单项后会弹出一个对话盒；有的菜单项在某些情况下是无效的，如 project 窗口为当前活动窗口时，Edit|Copy 就无效，菜单项无效时变成浅灰色，对它的选择无效；还有的下拉菜单中菜单项中有 > 符号，表示它还有一级弹出式菜单。

在下一章中，将详细解释所有的菜单项。

2.3 热键

表 2-1 是所有热键的列表，每个热键都对应一个菜单命令。

表 2-1 热键列表

热键	菜单命令	功能
F1	Help|Contents	打开求助窗口
ALT -F1	Help|Previous TTopic	显示上一屏求助信息
Ctrl-F1	Help|Topic Search	打开光标处字符串的求助信息
Shift-F1	Help|Index	打开求助信息索引窗口
ALT-Space	= 菜单	打开系统菜单
ALT-F	File 菜单	打开 File 菜单
ALT-E	Edit 菜单	打开 Edit 菜单
ALT-S	Search 菜单	打开 Search 菜单
ALT-R	Run 菜单	打开 Run 菜单
ALT-C	Compile 菜单	打开 Compile 菜单
ALT-D	Debug 菜单	打开 Debug 菜单
ALT-P	Project 菜单	打开 Project 菜单
ALT-O	Options 菜单	打开 Options 菜单
ALT-W	Window 菜单	打开 Window 菜单
ALT-H	Help 菜单	打开 Help 菜单
F2	File|Save	保存活动编辑窗口的文件
F3	File|Open	打开一个编辑窗口并装入文件文件
ALT-X	File|Quit	退出集成开发环境
ALT-0	Window|List	显示窗口的列表
ALT-N		激活窗口号为 N 的窗口 1<=N<=9

续表

热键	菜单命令	功能
ALT-F5	Window|User Screen	打开用户输出屏幕窗口
ALT-F3	Window|Close	关闭当前活动窗口
F6	Window|Next	依次激活各个窗口
F5	Window|Zoom	放大/还原当前活动窗口
Ctrl-F5	Window|Size/Move	改变当前活动窗口的位置和大小
Ctrl-Del	Edit|Clear	删除当前活动窗口中被选取的正文块
Ctrl-Ins	Edit|Copy	将当前活动窗口中被选取的正文块拷贝到剪裁板
Shift-Del	Edit|Cut	将当前活动窗口中被选取的正文块拷贝到剪裁板，并删除活动窗口中的这个正文块
Shift-Ins	Edit|Paster	把剪裁板上的正文块拷贝到当前活动窗口的光标处
Shift-Backspace	Edit|Undo	取消编辑窗口的上一次编辑操作
Ctrl-L	Search|search again	重复上一次搜索或替换的操作
ALT-F9	Compile|Compile to OBJ	编译当前活动编辑窗口中的源程序
F9	Compile|Make EXE File	构造当前工程(编译、连接)
ALT-F7	Search|Previous Error	在编辑窗口中搜索上一个语法错误
ALT-F8	Search|Next Error	在编辑窗口中搜索下一个语法错误
Ctrl-F2	Run|Program Reset	复位当前可执行程序
Ctrl-F9	Run|Run	从程序运行指针处执行程序
F4	Run|Go to Cursor	执行程序到当前编辑窗口光标行
F7	Run|Trace Into	单步执行程序，如当前行有函数调用，跟踪进函数中
F8	Run|Step Over	单步执行程序，如当前行有函数调用，不跟踪进函数中
Ctrl-F3	Debug|Call Stack	显示函数调用堆栈
Ctrl-F4	Debug|Evaluate/Modify	计算和修改变量的值
Ctrl-F7	Debug|Watch	增加一个观察变量到观察窗口
Ctrl-F8	Debug|Toggle Breakpoint	设置当前编辑窗口光标所在行为断点行或清除该行断点
ALT-F4	Debug|Inspect	打开一个变量的检察窗口

2.4 编辑窗口

Borland C++ 2.0集成开发环境的编辑器可以同时编辑多个源程序，每个源程序都有一个窗口。集成开发环境还设置了一个剪裁板，通过剪裁板，各个编辑窗口之间可以方便地进行