

C语言电脑投影 教学实践

王兴波 黄国立 编著

国防科技大学出版社

2

3/1

内 容 简 介

根据电脑辅助教学以及投影教学的规律,本书系统地介绍了用电脑投影进行教学的C语言知识体系与实践原则。通过对C语言程序设计的每个知识点以及要领的详细分析与综合和对大量实用例程的详细解释,本书展现了C语言程序设计的知识网络和特征。

全书共分两篇15章。第一篇为认识篇,介绍C语言程序设计的基础,包含C语言的基本数据I/O、C语言语句规范、C语言函数特征、C语言数组与指针等内容;第二篇为实践与提高篇,介绍C语言程序设计方法和技巧,包含构造数据类型、图形开发、低级I/O、文件操作、界面制作等内容。为便于复习巩固,每节都列有教学重点。本书还附有Turbo C库函数简介。书后习题由计算机随机排列组合,适合于综合复习和巩固C语言知识点。

本书所附软盘是按照电脑投影教学规范制作的CAI课件,适用于课堂投影教学与复习辅导。

本书适合于大学工科学习计算机语言课程的学生使用,也可作为广大C语言爱好者自学C语言程序设计的参考书目。

33 335260/13

前 言

随着计算机应用技术和集成化教学环境的推广和发展,CAI(计算机辅助教学)的方式已经迅速得到推广和普及,并且CAI已逐步由单一的习题教学方式发展成为综合播放指导方式。综合播放的CAI能够更好地应用现代教学手段,除了计算机外,它还能够与投影教学紧密结合。这种结合使传统的固定数据的幻灯播放、图片讲解、录像示教等投影教学具有新的活力,将计算机处理的画面动态实时地反映在大屏幕或者电视屏幕上,既能展示CAI的魅力,又能发挥投影教学集中管理的特长。

虽然如此,这样的教学环境在我国并不多见。这倒不是我们的硬件环境不具备,而是软件环境不能满足要求。

现实分析一下我国CAI的形式,可以发现,目前的CAI大多偏重于课外教学与自学,而忽视了作为教学主体的课堂教学;偶尔有些以课堂教学为基点的,但是由于过分渲染CAI课件本身的色彩而使人感到课本成为多余。这些与教学规律相去甚远。

作为教师,我感觉需要这样的教学环境,那就是它能够使我的课堂教学发挥更大的效能,也能使我的学生在这种课堂教学中轻松地获取更多的知识。

《C语言电脑投影教学实践》试图实现这一点。

C语言作为高级语言中的低级语言,无论在学校教学还是自学中,都存在一定的难度。这就要求教师的教和学生的学都需要根据教学中出现的实际情况,实时地调整教学内容(基本原理、理论、例子和实验)。作者在多年的C语言教学过程中,摸索和积累了一些教学数据。我把这些积累和收集写成这部系统教学的教材,并把这些内容制作成一个CAI课件。这个课件完全结合教材内容,非常适合于课堂教学。考虑到教学实践过程中的多态性,课件里所有的教学内容、例程和重点难点,是可以由教师根据教学需求修改的。这就改变了传统的固化教学数据,使得教学理论能够真正地与教学实践相结合。

本书内容分为两篇。第一篇为认识篇,主要介绍C语言程序设计的基础知识(包括C语言的基本语法、词法、数据构造规则、程序中数据流的流动规则及I/O规则等)、C语言程序设计的技巧两大部分。第二篇为实践与提高篇,主要列举了几个实际开发的例子,以期使读者进一步提高使用C语言编程的能力。内容包括:界面制作技术、图形开发技术、汉字技术及其他一些程序设计中经常需要用到的技术。

书末附录 A 中给出了大量的测试题及参考答案, 适合于读者结合复习和巩固 C 语言程序设计知识。附录 B 中对 Turbo C 的常用库函数作了简单的介绍。本书所附软盘是按照电脑投影教学规范制作的 CAI 课件, 适用于课堂教学与复习辅导。

本书适合于大学工科计算机编程课程的教学。本书宜配合谭浩强先生的《C 语言程序设计教程》(高等教育出版社, 1995)使用。

王兴波

1998.8.1

目 录

第一篇 认识篇

1 C 语言的特点与程序结构	3
1.1 C 语言的特点	3
1.1.1 C 语言的发展历史	3
1.1.2 C 语言的编程特点	3
1.2 C 语言程序的基本结构	6
1.2.1 程序头部	7
1.2.2 main 函数	8
1.2.3 其他部分	8
1.2.4 示例	9
1.3 程序设计中的几个名词术语	9
1.3.1 源程序	9
1.3.2 编译/解释, 编译/解释型语言系统	10
1.3.3 编译程序、目标程序	10
1.3.4 链接程序与链接	10
1.3.5 解释程序与解释	11
1.3.6 编译型程序系统与解释型程序系统	11
2 C 语言的基本数据类型与简单 I/O	12
2.1 C 语言的基本数据类型	12
2.2 C 语言的常量与变量	14
2.2.1 常量	14
2.2.2 变量	15
2.2.3 标识符	17
2.3 C 语言的基本 I/O 操作	18
2.3.1 printf	18
2.3.2 scanf	19
2.3.3 getch, getche, gets	21
2.3.4 putchar, puts	21
2.3.5 /*...*/C 语言的注释	21

目 录

3 C 语言的语句特征	23
3.1 语句结构概述	23
3.1.1 语句	23
3.1.2 流程及流程图	25
3.1.3 表达式	26
3.2 C 程序的三种基本结构及其语句	29
3.2.1 顺序结构	29
3.2.2 选择结构	30
3.2.3 循环结构	33
4 函 数	40
4.1 库函数	40
4.2 自定义函数	41
4.2.1 定义函数	41
4.2.2 声明原型	42
4.2.3 函数结构的要素	42
4.3 函数的参数调用	46
4.3.1 调用条件与方式	46
4.3.2 形参与实参	47
5 变量的存储属性	52
5.1 变量性质概述	52
5.1.1 变量的分类	52
5.1.2 数据在内存的存储	52
5.2 各类变量的存储属性分析	53
5.2.1 局部变量	53
5.2.2 全局变量	57
5.2.3 动态、静态、全局、局部的关系	58
6 宏与工程	62
6.1 宏及其特点	62
6.1.1 宏定义的类型	63
6.1.2 宏定义的注意事项	64
6.1.3 宏与函数的比较	64
6.2 工程与工程文件	66
6.2.1 工程的概念	66
6.2.2 建立工程文件的方法	66
6.2.3 开发工程时的注意事项	67

7 C 语言的一维数组及其应用	68
7.1 数组的概念与基本属性	68
7.1.1 数组的概念	68
7.1.2 一维数组的定义	68
7.1.3 一维数组的初始化	69
7.1.4 一维数组的操作	69
7.1.5 一维数组的下标运算	70
7.1.6 数组的存储属性	71
7.2 数组的引用	71
7.2.1 一维数组作为函数的参数的特点	72
7.2.2 使用数组的注意事项	74
7.3 字符数组与字符串	76
7.3.1 字符数组的定义	76
7.3.2 字符数组的初始化	76
7.3.3 字符'\0'的特殊意义	77
8 指针及其属性	79
8.1 C 语言的指针及其性质	79
8.1.1 指针的概念	79
8.1.2 首地址	80
8.1.3 指针变量	80
8.1.4 指针变量容易产生的误区	80
8.2 指针变量的引用规律	82
8.2.1 一般引用	82
8.2.2 指针变量作为函数的参数	84
8.2.3 用指针作函数的参数的优点	84
8.3 指针与数组	86
8.3.1 指针访问数组元素	86
8.3.2 指针与字符串	88
8.3.3 指针、数组、字符串的关系	91
8.4 指针的其他问题	94
8.4.1 返回指针的函数	94
8.4.2 指针数组	94
8.4.3 指向函数指针的应用	97
9 结构体、共用体数据及其应用	100
9.1 结构数据类型	100
9.1.1 结构体的构造与特点	100

9.1.2 结构体类型变量的定义.....	101
9.1.3 类型与变量	102
9.1.4 不同定义方法的比较	102
9.1.5 结构体变量的存储属性.....	103
9.2 结构变量的引用规律.....	103
9.2.1 一般引用	103
9.2.2 作为函数的参数的引用.....	105
9.3 结构数组、指针及函数.....	107
9.3.1 结构数组的定义与调用.....	107
9.3.2 结构指针的定义与引用.....	107
9.3.3 结构体类型的函数	111
9.4 链表及其应用	113
9.5 共用体	116
9.5.1 共用体的概念与特点	116
9.5.2 共用体变量的引用规律.....	118
10 C 语言的文件 I/O 及其应用	122
10.1 C 语言文件及其属性.....	122
10.2 缓冲文件的操作	123
10.2.1 缓冲文件的打开与关闭.....	124
10.2.2 文件的顺序读写	126
10.2.3 文件的随机读写	129
10.2.4 文件结束符	132
10.2.5 缓冲文件的其他操作	133
10.3 非缓冲文件及其操作	133
10.3.1 非缓冲文件的基本操作.....	133
10.3.2 非缓冲文件的应用	134
11 C 语言的位运算与低级操作	136
11.1 位运算	136
11.1.1 位运算的概念	136
11.1.2 C 语言的位操作运算.....	137
11.1.3 按位操作的一些简单应用技巧.....	137
11.2 位运算的应用	138
11.3 C 语言的低级操作.....	142
11.3.1 int86	143
11.3.2 bdos.....	146
11.3.3 intdos	148

12 C 语言混合编程	151
12.1 C 语言与汇编语言接口	151
12.1.1 ANSI C 里使用 #asm 和 #endasm 指令	152
12.1.2 TURBO C 和 BORLAND C 里使用 asm 语句	152
12.2 C 语言与其它高级语言接口	155

第二篇 实践与提高篇

13 界面制作技术	161
13.1 基本制作	161
13.1.1 菜单制作技术	161
13.1.2 综合界面制作	170
13.2 一个完整的界面例子	173
14 图形开发技术	188
14.1 图形显示与视频初始化	188
14.2 图形开发的几个技术问题	192
14.2.1 图形菜单的制作	192
14.2.2 汉字的显示	197
15 汉字开发技术	200
15.1 汉字显示的问题及解决方法	200
15.1.1 西文 DOS 下显示汉字的方法	200
15.1.2 西文 DOS 下汉字显示技术开发	202
15.1.3 汉字显示的放大技术	204
15.2 自建汉字库与汉化软件	207
15.3 其他开发技术实践	210
15.3.1 PASCAL 语言到 C 语言的转换工具	211
15.3.2 分页打印程序的设计	218
附录 A 测试题	225
第一部分 单向选择题	225
第二部分 阅读理解题	240
第三部分 测试题参考答案	268
附录 B TURBO C 常用库函数简介	272

第一篇

认识篇

1 C 语言的特点与程序结构

计算机语言是学习和了解计算机知识的工具,已经是毋庸置疑的公理。C 语言作为高级语言中的低级语言,在软件设计和开发中起着重要的作用,其学习和掌握有其独特的规律。本章介绍一些学习和掌握 C 语言的入门知识。

1.1 C 语言的特点

学习 C 语言,首先应该了解 C 语言的特点。这可以从以下两个方面入手。

1.1.1 C 语言的发展历史

C 语言不是历史最悠久的高级语言,却是生命力最旺盛的高级语言之一。

早期 C 语言的发展是围绕能否与硬件打交道的问题的。人们希望能找到一种取代汇编语言、能够编写操作系统的语言。C 语言是在当时一些已经使用的高级语言 ALGOL 等的基础上发展起来的。其历史可简单描述如下:

1960: ALGOL-60->面向问题的语言,远离硬件。

1963: CPL->在 ALGOL-60 基础上,向硬件接近,但规模庞大难以实现。

1967: BCPL->简化了 CPL。

1970: B 语言->简化 BCPL,基本能够与硬件打交道,产生了简单的 UNIX 操作系统。

1973: C 语言->完善了 B 语言,但移植性较差。

1977: 可移植的 C 语言问世。

从此, C 语言开始风靡全球。

1.1.2 C 语言的编程特点

从编程设计来讲, C 语言具有如下特点:

1.1.2.1 符合结构化、模块化程序设计规范

结构化模块化是 20 世纪 60 年代中期提出的一种程序设计规则,它要求程序按照模块进行结构设计,每个模块具有相对独立性。按照这种要求设计的程序可读性好,移

植性好。由于 C 语言是在 60 年代中后期发展起来的。因此，它一问世就必然满足这种规则。

1.1.2.2 基本数据简单, 数据处理内容丰富, 范围广泛

一种语言的数据处理能力确定了这种语言的应用范围。虽然 C 语言是同 PASCAL、BASIC 等高级语言一样的高级语言，但是 C 语言具有很强的数据处理能力。需要说明的是，C 语言直接处理的基本数据很简单(只有整型、实型、字符型、指针型和 void 类型)，但是 C 语言通过一些规则将这些数据扩充成许多丰富的类型。这与 PASCAL、BASIC 等其他语言不一样。PASCAL、BASIC 的基本数据类型多、而扩充处理的灵活性较差。

1.1.2.3 运算丰富, 支持简单明了的运算符

与其他语言相比, C 语言的运算符和表达式要简单多了。表 1-1 是 C 语言与 PASCAL 的简单比较。

表 1-1	
C 语言的表达式	PASCAL 的表达式
$x \approx y + z$	$x := y + z$
$x++$	$x := x + 1$
$x \& \& y$	$(x > 0) \text{ and } (y > 0)$

虽然表达式简单，但是 C 语言支持的运算有 34 种之多。

1.1.2.4 功能强大, 能进行多种低级操作

C 语言是除了汇编语言以外功能最强大的语言。它能实现许多低级操作如对磁盘、打印机、I/O 端口、鼠标等直接操作。利用 C 语言能编写控制硬件的程序对一些硬件进行操作。例如，用 C 语言编制一些对磁盘操作的程序如创建磁盘目录，删除磁盘文件，修改文件属性，读写磁盘物理扇区等的程序是非常方便的。而像 FORTRAN，FOXPRO 等高级语言则基本难以做到。

1.1.2.5 目标码质量高, 可移植性好

C 语言的编译程序是高效编译系统, 生成的目标代码质量高。经过实验统计，C 语言编译的目标码只比汇编目标码的效率略低而高于其他所有高级语言的目标码。C 语言生成的目标码可以与其他语言链接生成可执行文件，其移植性能极好。

1.1.2.6 丰富多彩的系统工具、库函数

所有的 C 语言系统都有一个共通的特点: 提供大量的库函数。现在的 C 系统都拥有几百个库函数。利用这些库函数，程序设计者能够开发丰富多彩的软件。

1.1.2.7 能培养具有良好素质的程序员

不像 PASCAL 那样,编译程序对源程序进行严格的语法检查,在编译时就把一些可能潜在的错误全部排除。C 语言的编译程序对一些语法不作严格的检查。例如,如果我们定义了一个长度为 10 的数组,用这个数组作一个函数的参数,则下面的语句:

```
void myfunc(int array[]);  
void myfunc(int array[10]);  
void myfunc(int array[100]);
```

在 C 语言里全部能够通过编译,而在 PASCAL 里,只有第二个句子能够通过。

因此像这样的问题,需要编程的人在设计时作好总体规划工作,在实时编程时要心中有数。长期下去,自然会养成良好的习惯,提高素质。

本节要点:

- C 语言不是历史最悠久的高级语言,确是生命力最旺盛的高级语言之一。
- 符合结构化、模块化程序设计规范。
- 基本数据简单,数据处理范围广泛丰富。
- 运算丰富,支持简单明了的运算符。
- 功能强大,能进行多种低级操作。
- 目标码质量高,可移植性好。
- 丰富多彩的系统工具、库函数。
- 能培养具有良好素质的程序员。

小知识——C 语言的家族

C 语言的发展历史虽然短,C 语言的家族却成为计算机高级语言中最庞大的。针对不同的任务和对象,现在有数种 C 系统在被广泛地使用。常见的有:

Turbo C 系列(简称 TC):美国 Borland 公司的产品,它有一个集成环境 IDE;使用方便,现在几乎所有的大学采用它作为教学工具。但是它编译出 DOS 或 WINDOWS 实模式的 .EXE 文件,因此适用于一般教学、科研。

Borland C/C++ 系列(简称 BC):美国 Borland 公司的产品,它有一个集成环境 IDE,使用方便,功能比 TC 更强大。2.0 以上的版本支持面向对象的程序设计思想。4.0 以下版本只能编译出 DOS 或 WINDOWS 实模式的 .EXE 文件,4.0 以上版本需要 32 位操作系统的支持。适用于一般教学、科研。

Microsoft C/C++ 系列(简称 MSC):美国 Microsoft 公司的产品,6.0 以下的版本没有集成环境;7.0 以上版本有一个集成环境 IDE。MSC 功能强大,尤其是图形处理方面比 BC 和 TC 提供更多的库函数。7.0 以上版本能开发 WINDOWS 程序。但是由于它早期版本没有集成环境,另外由于 MSC 通常用于专业性较强的开发领域,现在国内使用它的人不多。

WATCOM C/C++ 系列(简称 WC):加拿大 WATCOM 公司的产品,9.0 以前版本没有集成环境;10.0 有一个 WINDOWS 集成环境 IDE。WC 的功能强大,许多专业开发需要用到它。

如 NOVELL NETWARE 系统开发, AutoCAD-ADS 开发等。它不仅能够编译出 DOS 或 WINDOWS 实模式的 EXE 文件,也能编译出保护模式下的程序。有些专业开发方面,几乎是它莫属。但是由于它没有 DOS 下集成环境,国内用户极少。

High C/C++ 系列(简称 HC):MetaWare 的产品。基于 intel80386 及 intel80486 处理器以及 MS-DOS Extender 的 C 语言系统。HC 能够编写 DOS 实模式及保护模式的程序以及 WINDOWS 程序。HC 到现在为止,没有集成开发环境。也不能单独使用,它需要 Phar Lap 公司的链接器 386LINK 才能生成可执行文件。HC 的功能较强大,但是由于它的使用条件较苛刻,因此国内用户较少。HC 常用于开发一些专业软件如 AutoCAD 的二次开发等。

Visual C/C++系列(简称 VC):美国 Microsoft 公司的产品。基于 WINDOWS 环境的 C 语言系统。具有集成开发环境 IDE。VC 是新型的 C 语言系统。由于它基于 WINDOWS 环境,因此特别适用于开发 WINDOWS 应用程序。2.0 以下版本可以开发 DOS 及 16 位 WINDOWS 程序。2.0 以上版本需要 32 位操作系统的支持,常用于开发 WIN95 及 WIN-NT 应用程序。

1.2C 语言程序的基本结构

在一个源文件里,C 语言程序的结构基本上是有规律的。一般地,我们可以把一个源文件分成三个大的部分:

```
/*程序头部*/
#include <xyz.h>          /*当.H 文件在系统路径时*/
#include "xyz.h"          /*当.H 文件在当前目录时*/
...
#define A B               /*程序中定义的各种宏 */
...
#define A B               /*程序中定义的各种宏 */
...                       /*其他包含的头文件 */
void MyFunc();            /*自定义函数的原型 */
...
int a,b;                  /*全局变量的定义 */
...

/*main()函数*/
void main()
{
    int x,y;              /*main 的局部变量定义区*/
```

```

...

x=2;y=0;          /*main 的运算处理区域  */
y+=x;
...
...
return;           /*main 的返回值   */
}

/*自定义函数区域*/

MyFunc1()
{...}

MyFunc2()
{...}

...

```

1.2.1 程序头部

程序头部是由一些预处理包含命令、宏定义组成。

在一个程序里，我们不可避免地要调用一些系统的库函数或者使用一些系统定义的常数、变量等。这些库函数的调用规范、常数、变量的定义都是描述在一些后缀为.H 的文本文件里。这些文件称为“头文件”。

C 语言程序的第一部分要通过包含头文件来表明该程序调用那些库函数，定义了那些常数、变量。其格式为：

```
#include <C 的头文件名>
```

或：

```
#include "C 语言的头文件名"
```

尖括号<>表示引用的头文件在系统路径里；

双引号" "表示引用的头文件在当前目录里。

例如：

```
#include "stdio.h"
```

 表示 `stdio.h` 这个文件在当前工作目录下，而 `stdio` 是由 `standard input and output` 几个单词而来的；因此，上面的包含句子说明程序中要用到标准输入输出的操作。

常见的一些包含命令有：

```

#include <conio.h>      /*程序里用到控制台 I/O*/
#include <stdlib.h>     /*程序里用到标准库*/
#include <string.h>     /*程序里用到字符串操作函数*/
#include <bios.h>       /*程序里用到 BIOS 硬件操作的函数*/

```

1.2.2 main 函数

所有 C 语言程序都必须有一个 main 函数。例如,下面是一个最简单的 C 语言程序。

```
#include <stdio.h>
void main()
{
    printf("Hello,World!\n");
    return;
}
```

C 语言的 main 函数称为主函数,程序运行时总是从 main 函数开始。main 函数基本上是调用其他函数(也可以递归地调用它本身)或者进行一些基本运算。其基本结构如下:

```
void main()          /*主函数名*/
{
    XXXXXXXXXX      /*定义 main 函数的变量*/
    YYYYYYYYYY
    FunctionI();      /*调用函数 FunctionI*/
    .....
    .....
    FunctionN();      /*调用函数 FunctionN*/
    return;          /*main 函数结束返回*/
}
```

1.2.3 其他部分

程序的其他部分基本上是由用户定义的各种函数组成。

除 main 函数外, C 语言源文件的其他部分基本上是由程序设计者自己定义的函数或数据组成。一个源程序里,可能会有成千上万个用户自己定义的函数。程序设计者的主要工作就是设计这些函数。因此,一个 C 语言源文件的结构为:

```
#include <XXXXXX.H>
.....      /* 头文件包含区*/
void main  /*    main 函数*/
{
    .....
}
```