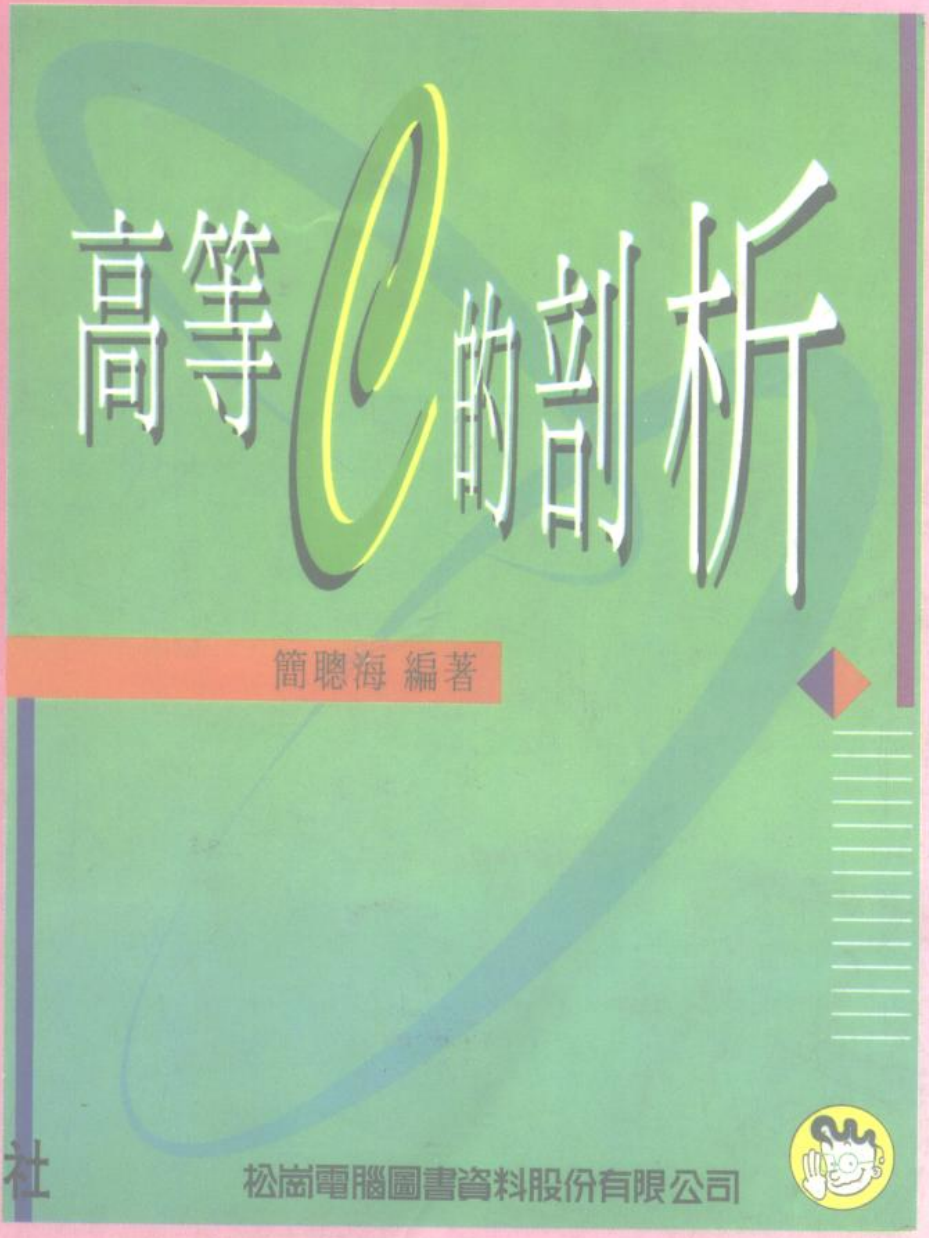
松联
天大松岗系列丛书

高等C的剖析

简聪海 编著



天津大学出版社
松崗電腦圖書資料股份有限公司

高等C的剖析

简聪海　编著

天津大学出版社

内容提要

本书以轻松幽默的语言由浅入深地讲述了C语言的基本结构、输入输出函数、算术运算、if与switch条件结构、循环、数组与字符串、函数、指针、结构、文档与主函数的参数、变量的贮存类别等，最后详述了大程序的编写方法。

书中通过400多个例题详细分析了C语言的指令及语法规则的应用要领，并通过例题对编程时易犯的错误做了画龙点睛的剖析。

本书是优秀的C语言的自学用书，也可作为各类学校的C语言教材。

高等C的剖析

简聪海 编著

*

天津大学出版社出版

（天津大学内）

邮编：300072

高等教育出版社印刷厂印刷

新华书店天津发行所发行

*

开本：787×1092毫米 1/16　印张：34¼　字数：855千

1996年9月第1版　1996年9月第1次印刷

印数：1—5000

ISBN 7-5618-0867-4

TP·80　定价：60元

自　　序

C语言在高级语言(如FORTRAN、PASCAL、COBOL、PL-1)中算是资浅,它的年纪也比低级的组合语言轻。把高低级语言贯通之后,今日它的C++如日中天。没有C自然不会有C++,正如没有旧约就不会有新约圣经一样。新的以旧的为骨架撑开面皮走在旧的前面。君不见耶稣走在基督前面合称耶稣基督。C++的++领着C向右走,所以C是C++的子集合。再者,C++的++恰好是C的递增运算符。如果把C++的C小写上分号";","C++;"这条指令就是"C=C;C=C+1;"的两条指令(statememt)的总和。C提出++就是为节省程序的执行时间,正如C提出malloc()函数与union结构为了节省内存的空间。节省时间与空间是C语言的精神所在。这是软件工程师挖空心思的题目。这部分的基础在C语言的范围内。

事本身固然有繁简之分,并没有所谓难易之别。当您觉得C很难时,不妨读这本《高级C的剖析》。因为它的基础构想是领着难于C的读者走向C++的门口,而成为一个不觉得C难的读者;因为本书是从一个最简单的C语言程序谈起,一路走向C的中心也是重心的指针,然后是结构,OS系统的命令的模拟,最后完成于团队team work写大程序的基本架构。因此,本书的范例共有四百多题加上重点剖析。

先有C才有C++,常见人问先有鸡呢,还是先有蛋?是鸡生蛋还是蛋生鸡?有人说答案在耶和华先生身上。为了解开这迷惑众生无数的错误命题,答案应不在耶和华先生那里。三岁孩提都知道先有老母鸡才会有小鸡。怎么可以用孵小鸡的"蛋"来冒充"老母鸡"去颠倒众生?想到个人力量薄弱,才疏学浅加上老眼昏花,书中难免有错,望社会贤达不吝指正。

简聪海写于永和

1995.元.12

目　录

第一章

从一个简单程序说起

简　介

今日属于C语言的软件发展速度很快。尤其在台湾这个宝岛的科技、人文等专业人才密度在全世界的排行榜上有名的情况下，用C来发展的软件到处可见。据说台湾的外汇主体是电脑软硬件的产品，相信这种说法离事实很近。拿C语言来说，它的历史是大约在23年前，美国大电话公司AT & T的贝尔实验室两位高级电脑专家Brian W. Kernighan与Dennis M. Ritchie发展出最早期的C语言。到了1978年，这两位专家合写了一本The C Programming language把他们创造的C正式发表，然后在1988年美国国家标准局(National Standards Institute)简称ANSI正式把第二版的The C Programming language鉴定为标准C之后，大公司如Microsoft、Borland等等纷纷以ANSI C为标准发展自己系统上的C语言。虽然每一家厂商为赚钱之故发展一些属于自己的C编译程序(compiler)，但是像今日的Borland C以及C++都接受ANSI C的保留字(Reserved Word)及函数库的函数等等规定。因此，本书所采用的C保留字与函数以ANSI C为标准，其目的是没有必要受厂牌因素的限制，让使用不同厂牌的C的读者方便。本书内的所有程序全部在Turbo C之下编写的，相信持有Microsoft C、Boland C、Turbo C++、Boland C++等等属于C语言系统的编译程序的读者也可以使用这本书来学习高等C。所谓高等C是C+的说法，因为从C到C++的发展过程中间原本有一段是C+。等到C++站稳脚步之后C+很快在C的领域内摇摆于C与C++之间。比方说，有些C++把指向函数的指针，从系统的命令行(command line)把命令行的字串当参数从系统传给主函数的void main(int argc,char *argv[])以及返回指针的函数(function returns a pointer)等等当做C++的一部分。在美国有些C+的作家就认定，凡是深入C的指针(pointer)、结构

(structure)，以及上面所提及的从系统的命令行把参数传给主函数，加上编写大程序(larger program)的架构的安排以利(team work)合作写大程序的题目，那就是高等C(C+)的定义，也是这本书的书名——高等C的由来。加上本书内以百为单位的范例，详细说明C的语法及应用，以利初学C的读者。因此，本书的全名叫做《高等C的剖析》(The dissection of C+)。

1-1 一个简单的C程序

如果读者要用C语言在电脑的屏幕上印出一句英文字，C语言的程序是：

(A)
```
main()
{
    printf("Learn C")
}
```

经过编译、执行之后，屏幕便出现一句英文

```
Learn C
```

当然这里假设读者已知道如何用C的软件，如Turbo C，已知道如何在Turbo C的IDE(Integrated Development Environment)之下编辑(editing)上面这个C程序，并且使用IDE所规定的功能键执行编译(compiling)以及运行程序(Run)。如果连这种最基本的上机学C的简单步骤都有困难的读者，一定要请教会的人，大约只要花20分钟的学习时间。

C的程序是由所谓函数(function)所组成，像前述的程序

```
main()
{
    printf("Learn C")
}
```

主函数的范围在一双大括号之内

main是主函数的名称。C语言规定凡是一个名称后面紧接一双小括号如main()，这就是函数的结构

这种主函数的结构大约是1986以前流行的C。自从ANSI C成立后，具备功能的"原型函数"(function prototype)已取代传统的C函数的结构，就以上面(A)式来说，文的"原型"(prototype)是：

(B)
```
void main(void)
{
    printf("Learn C")
}
```

比较(A)式与(B)式，(B)式多了两个void字。void的字面意思是"空虚"。电脑语言是无知觉的无情物，当然不会感觉空虚。那是说：

```
void main (void)
```

这个void表示从main的外面没有把参数传给主函数main()。

这个void代表main()函数没有转出任何数据

经过主函数的结构分析后，C的函数语法明白告诉学它的人：函数不是一个老死不与其他函数或OS系统(操作系统)往来的东西。从外面传进来的东西，C语言把它叫做"参数"(argument)，送出去的东西叫做数据或资料(data)。再者，主函数是C语言程序的主体，程序内任何函数都以主函数为总部或总连络站。因此，大括号内的

```
printf("Learn C");
```

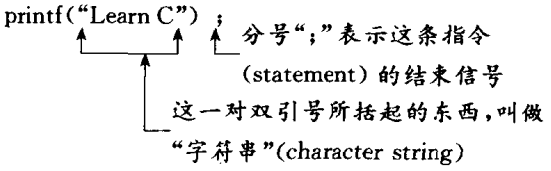
printf()函数才能把“Learn C”这句英语文字显示在屏幕上。函数的原型也就是原型函数原本只规定一个函数的名称前面一定要加上返回数据的数据型式(data type),比方说,void 就是一种名叫“空”的数据类型,也就是什么都不转回(转出)的数据类型,所以

(C)
```
void main()
{
    printf("Learn C");
}
```

也是正式的原型函数(function prototype)。

从(A)型到(B)、(C)两型,笔者认为(A)型已成过去,(B)型虽然很周全,但嫌多了一个 void。(C)型是中庸之道的主函数,因为它合乎原型函数的结构。因此,本书的程序一律采用(C)型的结构,望读者留意。再者,目前市面上 C 的软件大部分都接受上面(A)、(B)、(C)三种不同类型的主函数。

主函数内的 printf()是一种专管把数据显示在屏幕上的函数。若把 f 从 printf 拿掉,相信大家都认识这个“print”的意思是“印出来”。顾名思义,printf()函数是 C 语言做数据输出(output)的主角。所以主函数要它把“Learn C”印在屏幕上:



因此,“Learn C”是一条字符串。再者,任何函数都有它自己的定义。比方说,C 语言有两种函数:

(1)现成函数库中的函数;

(2)读者自己编写的函数。

像 printf()函数是 C 语言的现成函数库中的函数。C 语言有许多函数库,printf()函数是〈stdio.h〉库的函数。因此,一个养成好习惯的程序设计者遵照 C 语言的规定,一定会注明 printf()是从那一个函数库出来的。职是之故

```
#include <stdio.h>
void main()
{
    printf("Learn C")
}
```

才是一个正式的最简单的 C 语言程序,大多数的 C 编译程序(compiler)都会接受它。

第二章

C的基本结构及输出输入函数

简　介

任何电脑语言都有其独特的程序结构。C语言并不例外而且更具特色，因为C具备结构化程序(structured program)之条件以模块(module)的形式，由块状指令码(code blocks)所组成。所谓模块，指其程序结构乃是独立函数所组成，其内部之结构以及函数之自身都可谓之由模块的语码所组成。我们似乎可以说C语言程序结构的主体就是函数(function)，而函数自身的结构基因乃是模块。因此，所谓C语言的程序乃是由独立的主函数(main function)调用(call)另一函数，如此这般去完成程序设计员所托付之任务。学过其他高级语言的读者也许会觉得本书一开始马上涉及函数论有违由浅入深之学习阶梯。然而，因为连最基本且重要之输出与输入(I/O)设备都谓之输出函数与输入函数，浓汤亦可浅尝。知道如何调用现成函数与自己编写所需的函数，前者属浅尝，后者是做浓汤。

本章之目的：

(一)介绍C语言程序之基本结构：主函数 main()的原型(function prototype)；

(二)如何说明(declare)变量(variable)去代表数值(number)、字符(character)以及字符串(character string)；

(三)如何调用基本输出函数(output function)以及输入函数(input function)。

2-1　C语言程序的基本结构：主函数 main()

学过高级语言的读者往往因习性之故而喜做语言比较。有人说C的结构像PASCAL，然

而两者固有雷同之处，但差异也大。比方说，C 的指针(pointer)、函数(function)等等实在有异于 PASCAL 的指针(pointer)、函数(function)或过程(procedure)。因此，依笔者之浅见何妨放弃先见，用崭新的观点去学 C 语言，效果应该会更好。现在从一个完整且可执行的 C 语言程序谈起。

例 2-1　如果我们希望电脑会为我们印出一个英文句子“Ten years is long time for human's life.”一篇完整的程序如下：

```
/* ex2-1.c:To print a character string */
#include <stdio.h>
void main()
{
  printf("Ten years is a long time for human's life.");
}
```

输出结果：

```
Ten years is a long time for human's life.
```

以此例谈 C 语言程序之基本结构。

(一)C 语言中共有 5 种基本的数据类型(data tyep)：

(1)字符(character)：以 char 为代表；

(2)整型(integer)：以 int 为代表；

(3)实型(real number)：以 float 为代表；

(4)双精度实型(double)：以 double 为代表；

(5)空(void)：以 void 为代表。

从字符到双精度实数之定义易懂，后面章节会详列其范围。由于原型函数之基本定义(function prototype)一定要返回某一种数据类型(data type)。若某一函数什么都不返回，则该函数之本身的数据类型就是空无(void)。换言之，若遇到什么都不必返回的函数，在函数名称之前空一格，然后加上 void。如本例所示：

```
void main()
```

(二)主函数之定义(The definition of main function)：

主函数之基本结构如下：

```
void main()
{
        statement 1
        statement 2
            ⋮
        statement n-1
}
```

(1)首先解释

```
        void main()
```

(A)任何原型函数(function prototype)在标准 C 的规定下一定要返回某一种数据类型(data type)，如实型(float)、整型(int)、字符型(char)以及什么都没有的空无 void。

(B)main 是函数名称，其后不能有空格，紧接一小括号“()”，两者合起来对编译程序(com-

piler)而言，它代表函数。换言之，一个名称紧接一个小括号就是C语言的函数题头的写法。

(2)定界符号(delimiter)：一双大括号“{}”。

```
void main()
{
    statement 1
    statement 2
        ⋮
    statement n-1
}
```

void main()的下面第一行及最后一行合成一双大括号。第一个“{”代表函数main()所涵括范围的起头位置，而最后一行“}”表示main()函数的影响位置止于彼处。大括号内部的指令(statement)统统表示在main()函数的影响范围内。上面例2-1的主函数内只有一条指令(statement)：调用printf(…)而已。

(3)指令终结符号(statement terminator)：一个分号“;”。

C语言用分号“;”终结指令，表示此条指令之作用范围到“;”所在之位置为止。因此，“printf(…);”这条指令是以分号“;”殿后，以示该指令之终结处。若忘了加上分号，或在不应该加上分号之地方加上分号。比方说，把分号加在函数题头之后如下所示：

```
void main();    不可加分号
```

编辑程序巡察到它，一定给错误信息。

(4)空无(void)：以void为代表。

(三)关于#include <stdio.h>：预处理程序(preprocessor)

```
/* ex2-1.c:To print a character string */
#include <stdio.h>
void main()
{
    printf("Ten years is a long time for human's life.");
}
```

现在解释放在void main()上面的#include <stdio.h>。凡是在C语言内，用“#”号所引领的指令叫预处理程序(preprocessor)。include译成中文是“包括”。望文解义，其作用在把标准输出输入的文件<stdio.h>拷贝在#include <stdio.h>所在的位置，其影响范围遍及整个程序。这由于C语言的创立者把许多输出输入函数(如printf()、getchar()、scanf()等I/O函数)写好之后，存放在<stdio.h>文件内，因为我们调用printf()函数印出“Ten years is a long time for human's life.”这句英文。我们用了<stdio.h>内的现成函数printf()，所以用#include <stdio.h>把<stdio.h>整个文件拷贝在程序的最上层，让编译程序(compiler)巡察到printf()这条指令后，知道printf()函数被定义(编写)在<stdio.h>文件内后才能把printf()之定义引用出来做输出工作。

<stdio.h>是一种文件名。std表示standard(标准)；io代表输入输出(input and output)；h代表header起头之意。因此，<stdio.h>合称为“标准输出输入资料库”。任何一篇C语言凡有输出或输入或两者皆有的地方，一定要把#include <stdio.h>这条预处理程序打在第一行。

以上拉杂解释一大堆，重注如下：

```
#include <stdio.h>   (把标准输出输入库的数据拷贝在此处)
void main()          (定义一个返回空无的主函数)
```

```
{                    (属于 main 的指令由此处起)
    printf("Ten years is a long time for human's life.")
}    (用函数 printf()印出一句英文)
```

简单说:例 2-1 是主函数 main()调用 printf()函数印出一句英文而已。初学者若一时无法全部了解上面的解释,请不必心烦加失望,这是正常现象。到此为止,只要知道 C 语言的一篇最简单的程序之结构如例 2-1 即可。

(注一)C 语言程序的编辑程度在乎英文字之大小写(这是其他高级语言不分别的地方)。比方说,如果读者把 main()打成大写的 MAIN()一定行不通。盖小写 main 才是 C 语言的函数保留字。同理,把小写的 printf()打成大写的 PRINTF()也行不通。简言之,凡是程序内的指令(statements)一律要用小写字母,如例 2-1 内的 main、printf 和 include <stdio.h>,统统是小写。此规则亦有例外,后面章节遇到时再随时指出。

2-2 结构化的 C 语言

所谓"结构化的程序语言"(structured language)指该语言的程序具备:

(1)其程序的演算法(algorithm)之流程控制"由上而下"(Top to down);

(2)其程序的演算法之流程控制"一进一出"(one way in one way out);

(3)以及阶梯式的程序结构。

C 语言程序具备结构化之条件,故其程序易读、易修改,以及层次分明的逻辑结构。比方说,例 2-1

```
#include <stdio.h>
void main()
{
    printf("Ten years is a long time for human's life.");
}
```

符合程序流程控制由上而下、一进一出以及阶梯式的结构。从此以后,本书内任何程序之结构安排以及演算法(algorithm)的流程控制均采用阶梯式书写,由上而下以及一进一出的方式,以利程序之易读、易修改、易除错以及层次分明的逻辑结构。

2-2-1 奇形怪状的 C 语言程序结构

由于 C 的编译程序做语法(syntax)巡察时,凡是遇到程序内有

(1)换行(new line);

(2)空格字符(white space characters);

(3)按一下 tab 即跳过 8 个空格。

以上三种,C 的编译程序统统视而不见不予理会。由于这种原因,因此,C 程序的结构可写成千奇百怪之状。也因此,C 语言的程序才有结构化的可能。请参考下面数例。

例 2-2

```
/* ex2-2.c:To show one of the structures of C */
#include <stdio.h>
void main(){
```

```
    printf("One of the structures of C.");
}
```

输出结果：

```
One of the structures of C.
```

例 2-3

```
/* ex2-3.c:To show one of the structures of C. */
#include <stdio.h>
void main()
{
printf("Strange structure !");}
```

输出结果：

```
Strange structure!
```

例 2-4

```
/* ex2-4.c:To show one of the structures of C. */
#include <stdio.h>
void main()
{printf("Funny structure !");}
```

输出结果：

```
Funny structure!
```

例 2-5

```
/* ex2-5.c:To show one of the structures of C. */
#include <stdio.h>
void main(){printf("Ridiculous structure !!");}
```

输出结果：

```
Ridiculous structure !!
```

（注一） 以上五个例题以第一个例题最正式。因为符合易读性和层次分明之故。例二尚有价值。例三、四、五不足为法。一者，破坏了阶梯书写；再者指令(statement)之段落不分以致于破坏程序的易读性及不易除错，他人也不易读懂你所写的程序。甚至过一阵子，连写程序的自己也难读自己的程序。职是之故，本书内的程序一律以第一例之结构为样本，表面看去略为多占篇幅，但是结构清楚明白。

（注二） 记得主函数 main 与其后的小括号()中间一定不可有空格。

2-3 程序的文字注释(comment)

任何电脑语言为了注明该篇程序之用途或者某一条指令(statement)之意义都有所谓注释或说明的指令。C 语言也不例外，它是用/*…*/以/*为左端，以*/为右端，把所要说明的一段文字括起来，其位置可任意放在程序内任何地方，以利程序的易读性，其形式如下所示数种。

例 2-6 程序内之文字注明行的样式

```
(A) /* This is a single comment line */
(B) /* This case is
       * also suitable for
```