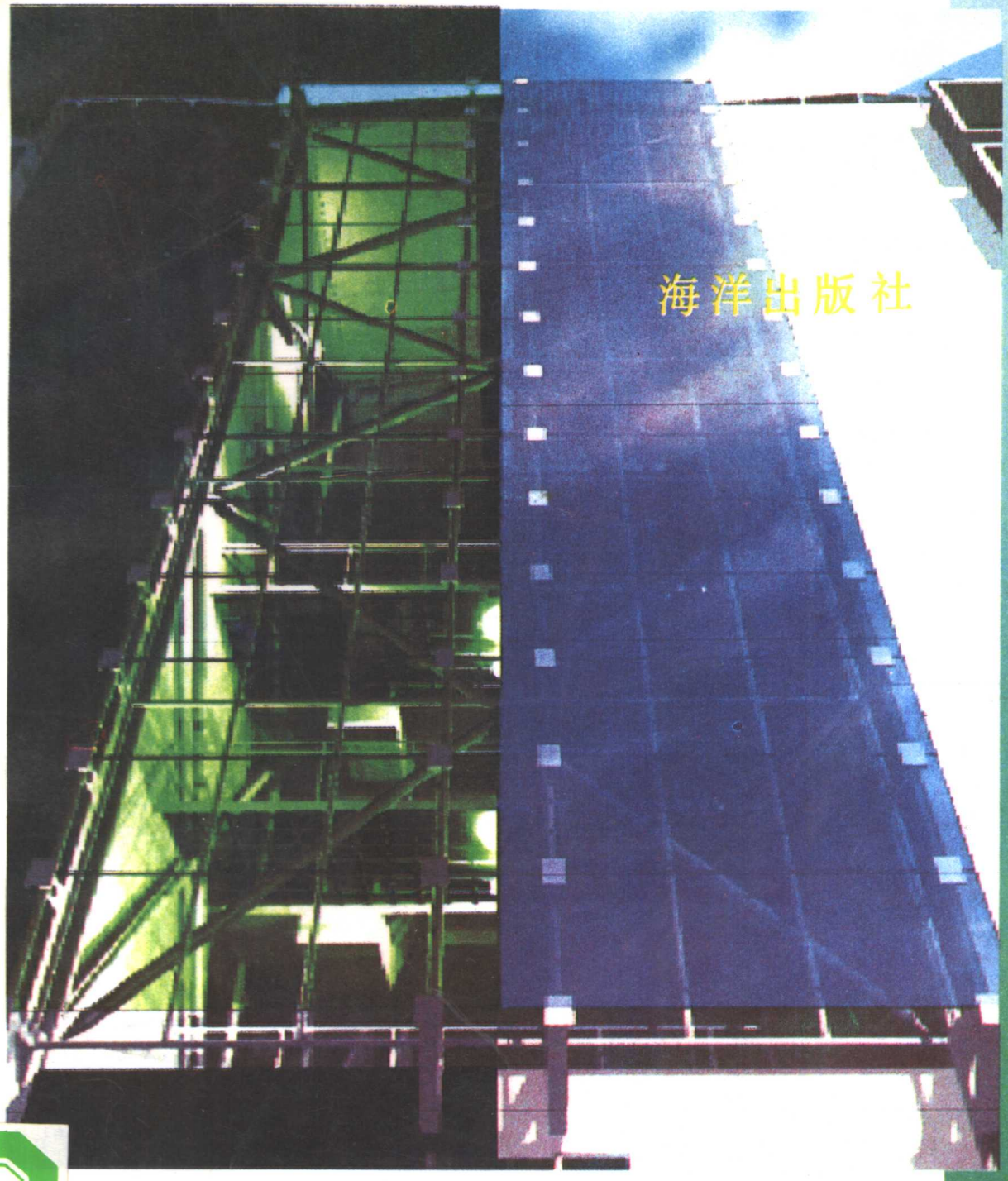

海洋出版社

如何彻底清除微机编译错误及实例分析

盛炎发　等编著
希　望　审　校

北京希望电脑公司计算机技术丛书

如何彻底清除微机编译错误及实例分析

盛炎发　等编著
希　望　　审校

海　洋　出　版　社
1993年·北京

内 容 摘 要

本书介绍的编译程序，是对 Microsoft Quick C Compiler Version 2.02、Microsoft C Optimizing Compiler Version 5.1 和 Microsoft C Professional Development System Version 6.0 中的常见编译错误进行了仔细分析，并提供了消除这些错误的实例。

C 编译程序包括命令行编译程序，连接程序和库管理程序等执行程序，这些程序显示不同的出错和警告信息。

这些信息可分类为：(1)编译程序的命令行方式，(2)编译程序，(3)运行时间库(程序的执行)，(4)浮动小数点的例外处理，(5)LINK(覆盖连接程序)，(6)LIB(库管理程序)，(7)NMAKE(程序维护应用)，(8)HELPMAKE(促成文件生成应用)。

本书介绍(1)和(2)部分的错误与警告。

该书是根据作者多年的工作经验编写而成，它可作为从事计算机软、硬件开发人员的参考书，也可供大学本科或研究生的教学参考之用。

需要本书的用户，请直接与北京 8721 信箱联系，电话：2562329，邮政编码：100080。

(京)新登字 087 号

北京希望电脑公司计算机技术丛书

如何彻底清除微机编译错误及实例分析

盛炎发 等编著

希 望 审 校

海洋出版社出版（北京市复兴门外大街 1 号）

海洋出版社发行　　北京双青印刷厂印刷

开本：787×1092 1/16　　印张：13.75 字数：308 千字

1992 年 11 月第一版　　1992 年 11 月第一次印刷

印数 1—4000 册

ISBN7—5027—3363—9/TP・183　　定价：14.00 元

目　　录

第一章　阅读“错误一览”前的必要知识

本章中将解说阅读“错误一览”的方法及编译所必需的知识。在阅读第二章的“错误一览”前，必须先阅读一下本章。

“错误一览”的阅读方法

“错误一览”中，有编译程序和命令行编译程序两种类型。“一览”的形式是统一的，因此无论是阅读还是查阅，都有同样的感觉。但是例子仍分为命令行和源程序。

错误号码由一个字母和四位数构成

C1×××　编译/致命错误

C2×××　编译/错误

C4×××　编译/警告

1.1.1　编译程序“错误一览”的阅读方法

编译程序的“错误一览”由如下几项构成

error C2075(1)

C5 标识符:array initialization needs curly　braces(2)

QC 同上(3)

C6 同上(4)

状况(5)

数组初始化时需花括号{　　}。

消除(6)

　　用{　　}将数组初始值括起来。

例 QC(7)

错误的源程序(8)	消除的例子(10)
/* C2075.C */	/* C2075.C */
int a[2]=1,1;	int a[2]={1,1};
Void main(　)	Void main(　)
{	{
}	}

错误(9)

　　C2075.C(3):error C2075: 'a':array

　　initialization needs curly braces

参照

C2075.C(3):C2059:Syntax error: 'int consarnt'

(1)表示错误的种类和号码

(2)Microsoft C Optimizing Compiler Version5.1 的出错信息

(3)Microsoft Quick C Compiler Version 2.02 的出错信息。

此为试读，需要完整PDF请访问：www.ertongbook.com

(4)Microsoft C Professional Development System
Version6. 0 的出错信息。
(5)解释错误或警告的内容。
(6)提出改正错误或警告的方法。
(7)编译例子中错误源程序的编译程序的种类。
C5 Microsoft C Optimizing Compiler Version5. 1
QC Microsoft Quick C Compiler Version 2. 02
C6 Microsoft C Professional development system
version6. 0
(8)显示错误或警告的源程序实例。
(9)编译(8)时显示的错误或警告信息。
C2075. C (3) :error C2075: 'a' :array initialization
needs curly braces
信息
错误的种类和号码
表示错误的行号
错误的源文件名
(10)消除错误或警告的源程序实例
(11)表示可参照的"出错"信息
参照方法
参照出错的信息,根据信息头的→记号的种类进行
语义分类。→记号有无→ →
→ C5
→ QC
→ C6 5 种,区别如下。
(1)无→
即使(3)的源程序里只有一个错误,有时也会显示多个信息。但是(9)中的错误信息,只用(1)的错误号码信息,其余的信息作为参考显示出来。无→号的就是错误信息。(上面的例子就是这个信息)关于由一个错误而显示多个出错信息这一点,在 1—5—4(最初出错的改正)一项中也有说明。
例
参照
C2075. C(3) :error C2059 :Syntax error: 'int constant
参照 fatal error C1020 错误的说明
(2)→
同(9)的错误内容意思相近,作为参考的错误信息。
例
参照
→C1070. C(13) :fatal error C1070 :mismatched
#if/#endif pair in file 'C1070. C'

参照 fatal error C1070
的错误说明

(3)→ C5

用 C5 编译(8)的源程序出现的错误信息。即使是同一源程式,由于编译式的种类和版本的不同,有时也会显示不同的信息,所以有必要加以注意。

例

参照

→ C5 C2041.C(5):error2020:bad Octal number '9'

参照 error 2020 的错误说明

(5)→ C6

用 C6 编译(8)的源程式时出现的错误信息。即使是同一源程式,由于编译程式的种类和版本的不同,有时会显示不同的信息,所以有必要加以注意。

例

参照

→ C6 C2036(7):error C2223:left of '→a' must
point to struct/union

参照 C2223 的错误说明

(4)→ QC

用 QC 编译(8)源程式时出现的错误信息。即使是同一源程式,由于编译程式的种类和版本的不同,有时会显示不同的信息,所以有必要加以注意。

例

参照

→ QC C1071.C(9):fatal error C1004:unexpected EOF

参照 fatal error C1004 的错误说明

其他((2) (3) (4))

同上

表示与上述显示内容相同

无

表示无信息

1.2 初学者应该从什么着手

将 C 语言的初学者进行分类,有以下几种类型。

(1)初次接触计算机

(2)盲目的接触(无意识的接触)

(3)能使用(文)字处理机

(4)能进行程序编辑

(5)能使用计算机

(6)能使用数据库

(7)能用 Basic 进行程序编制

(8)能用 BASIC 以外的语言进行程序编制

(9)具备 MS—DOS 的知识,学过 C 语言的语法

下面一一说明不同的初学者应该具备哪些必要的知识。

(4)初次接触计算机

初次接触计算机者,学习 C 语言是个错误,至少得先学会包括无意识的接触学习的编辑程序。

→参照(3)

(2)无意识接触

虽然无意识接触也是重要的学习方法,但一定得学会计算机的编辑程序。

参照(4)

(3)能使用(文)字处理机

(文)字处理机和计算机有区别。所以一定要学会计算机的编辑程序。

参照(4)

(4)能使用编辑程序

是学习 MS—DOS 和 C 语言的手段。

(5)能使用计算机

考虑程序的算法和思考表软件的宏( )是一样的,但是,有时语法和差异也会产生障碍,所以有必要掌握 MS—DOS 和 C 语言的语法。

参照(9)

(6)能使用数据库

参照程序的算法,和用数据库来考虑是一样的,但是,有时语法的差异也会产生障碍,所以有必要掌握 MS—DOS 和 C 语言的语法。

→参照(9)

(7)能用 BASIC 进行程序编制

考虑程式的算法,和用 BASIC 语言进行考虑一样。但是,有时语法的差异也会产生障碍,所以有必要掌握 MS—DOS 和 C 语言的语法。

参照(9)

(8)能用 BASIC 以外的语言来编制程序

考虑程序的算法,和用 BASIC 以外的语言来考虑是同样的,但是有时语法的差异也会产生障碍,所以有必要掌握 MS—DOS 和 C 语言的语法。

→参照(9)

(9)具备 MS—DOS 的知识,学过 C 语言的语法

如果有 MS—DOS 的知识和 C 语言使用的经验,那么最低的条件就成立了。

可以从短的程序编制开始演习。

1.3 调试程序始于错误和警告消除后

程序(源文件)完成后,为了生成执行文件,要进行源文件的编译。这时命令行或源文件若有差错,编译程序显示错误或警告信息。修改命令行或源程式,再进行编译,将全部的错误和警告消除。但是,这样还没有完成。

即使执行生成的程序，也不一定会如自己所想的那样运转。这是由于在程序设计中的算法程序有问题。出错和警告对算法程序的差错不显示任何信息，只不过指出语法错误罢了。消除出错和警告，并不等于调试，正式的调试，始于出错和警告消除后。

1.4 警告级的功能和区分方法

1.4.1 将警告级提到最大

编译程序时，如果程序中出现重大问题，错误信息就显示出来并中断编译。非重大问题时，显示警告信息并继续编译。

警告分成好几个级别。处于第1级状态时，即使错误和警告全部消除，提高级就会出现迄今没有显示过的警告，不修改这个警告，常会出现奇怪的程序运转结果。

象这样如果在警告级时编译，即使没有警告显示，也有潜在的可能性。必须将警告级提到最大后进行编译。

有关编译程式其它的警告级及其功能如下表所示

选择项	功　能
/WO 或/W	不显示任何警告信息
/W1	显示几乎所有警告信息(　　)
/W2	除显示/W1 的警告信息外，还显示函数和数据精度的警告信息
/W3	除显示/W2 的警告信息外，还显示不能用 ANSI 标准下定义的功能微软件规格的警告信息
/W4	支援更详细的(象 LINT 一样的)警告和 ANSI 的重合检验
/WX	将警告作为错误处理，警告一发生，OBJ 文件就不能产生

1.4.2 警告级的区分方法

警告级的区分有命令行区分和环境变量 CJ 区分两种方法。

(1)MS—C Ver5.1 命令行

.命令行 CL/M3 TEST.C

.环境变量 CL SETCL=W3 CL TEST.C

(2)MS—C Ver6.0 命令行

.命令行 CL /W4 TEST.C

.环境变量 CL

SETCL=/W4 CL TEST.C

(3)Quick C ver2.0 命令行

·命令行

```
QCL/W3 TEST.C
```

·环境变量 CL

```
SET CL=/W3
QCL TEST.C
```

(4)MS—C Ver6.0 工作台(Work bench)

下面说明将警告级提到最大的步骤。

(1)同时按 GRPH 和 O 健

显示程序单的<Options>菜单

(2)按 C 键

选择菜单的<C Compiler Options>,就会显示编译选择的长方形框。

(3)按 4 键

警告级<4>的左边显示 *

(4)按回车键

不按回车键而终结时,不会变换警告级。这确认变换与否,从最初开始按步骤执行,确定警告级有无达到<4>。

(5)Quick C Ver2.0 集成环境

下面说明将警告级提到最大的步骤。

(1)同时按 GRPH 和 O 键

显示程序单的<O/环境>菜单

(2)按 M 键

选择菜单中的<M/编译、连接…>,就会显示出编译、连接的长方形框。

(3)按 C 键

选择<C/编译程序选择>,就会显示出它的长方形框。

(4)按 3 键

警告级<3>的左边显示 * 。

(5)按回车键

回到编译、连接的长方形框。

(6)用

TAB 将黄色的箭头移动到<确认>后，按回车键。

没有移到<确认>而终结时，警告级不变化。为确认变换与否，从最初的步骤开始执行，确认警告级达到<3>与否。

1.5 消除出错和警告的决窍

1.5.1 利用调试程序寻找异常

执行程式的运转一发生异常，马上起动调试程序进行检查。因此，有必要指定/zi 选择进行编译。但是发生编译错误的时候，若一一指定/zi 选择修改编译的话，要花费很多时间。当然，光凭看源程式来寻找异常是没有用的，必须通过调试程式来寻找。要做到无论什么时候都能执行源码调试，并指定/zi 选择执行编译。

1.5.2 有效利用指示字检验选择

编译时若能预先指定/zi 选择，能确认程式中的 NULL 指示字和范围外的 FAR 指示字。根据这个可以发现程式的异常。

如果指示字检验如下使用 #Progma，则源程序也可指定。

#Pragma Check Pointer([{on|off}])

on 指示字检验有效

off 指示字检验无效

1.5.3 不生成复句

请看下面的表：

```
/* LIST1.C */
#define A 1
#define C 2
void main( )
{
int a,b,c;
a=A; b=B; c=C;
printf("a=%d b=%d c=%d",a, b, c);
```

用 Quick C Ver2.0 编译这表，就会显示以下出错信息：

```
Microsoft(R)QuickC compiler Version 2.00.000
Copyright(C)Microsoft Corp 1987—1989. All rights
reserved.
LIST1.C
LIST1.C(12):error c2065: 'B':undefined
```

出错信息由文件名：出错所在行的行号和出错内容等构成。但是出错显示的行（第 12 行）里有 3 个句子，这 3 句中哪一句是错误的呢？这个例子因为简单，所以很容易发现。

但是，在复杂的程式中，不可能那么简单地发现，怎么办才好呢？在一行不写双数句子，出错只显示以行号为单位产生的错误部分。即使是一个句子也不要使它复杂化，能分就分成

此为试读，需要完整PDF请访问：www.ertongbook.com

多句。修改 LIST1. C,则程式变为 LIST2. C。

```
/* LIST2.C */
#include <stdio.h>
#define A 1
#define B 2
#define C 3
void main( )
    {
    int a,b,c;
    a=A;
b=B;
    c=C;
    Printf("a=d% b=%d c=%d",a, b, c);
    }
```

1.5.4 最初修改出错

请看下表。

用 Quick C Ver2. 0 编译 LIST3. C，会显示如下出错和警告信息。

LIST3. C

```
/* LIST3.C */
void main( )
    {
    a( );
    }
void a( )
    {
    }
```

出错：

```
Microsoft (R)QuickC Compiler Version 2.00.000
Copyright Microsoft Corp 1987—1989. all rights reserved.
LIST3.C
LIST3.C(5):Warning c4016: 'a' :no function return type, using 'int'as default
LIST3.C(5):Warning c4071: 'a' :no function prototype given
LIST3.C(9):error c2086: 'a' :redefinition
```

错误信息有 3 个，在源程式的最先追加一行

```
Void a(void)
```

则所有的信息消失。LIST4.C 是修改后和情况：

LIST4. C

```
/*LIST4.C */
void a(void);
void main( )
```

```
{
a( );
}
void a( )
{
}
```

这样稍微的变换会引起多个出错和警告的消失，所以即使在显示很多出错和警告信息的时候，也不必惊讶。在出错和警告中要尽量从最初的出错开始修改。如果源程序较长，会屡次发生由最初的出错引起后面的出错和警告。即使出错信息很多，只要修改最初的出错，也可以解决所有的问题。

第二章　错误一览

2.1 出错一览

2.1.1 编译程序/致命出错(fatal error)

fatal error C1000

C5 UNNKNOWN FATAL ERROR

Contact Microsoft Technical Support

QC 同上

C6 UNNKNOWN FATAL ERROR

Cortact Microsoft Technical Support

状况

检查出编译程序中不能识别的重大出错

消除

这时,发生了编译程序的说明书上所没有的情况。因此,在明确原因以前,大概需要很多时间。若程序开发中没有空裕时间,这大概要以自身过去的经验为基础加以消除吧。

例子

这种出错一旦存在就会非常棘手,所以只有祈求它不发生。

fatal error C1001

C5 Internal Compiler Error

(Compiler file '文件名',line '行号')

Contact Microsoft Technical Support

QC 同上

C6 Internal Compiler Error

(Compiler file '文件名', line '行号')

Contact Microsoft Product Support Services

状况

编译程序检查出内部的出错。

消除

这时,发生了编译程序所没有说明的情况,因此,在查明原因前,大概需要很多的时间。若程序开发中没有富裕的时间。大概就要以自身过去的经验为基础加以消除。

例子

出错一旦存在就会非常棘手,所以只要祈求它不发生。

fatal error C1002

C5 Out of heap space

QC 同上

C6 Compiler is out of heap space in Pass 2

状况

编译程序用完了动态存储器的空间。

消除

(1)原文件较大,把它分成几个小文件。

(2)将式分成小的子式

(3)在 OS/2 环境下,使用/B2 选择调动编译程序第二通路的大型模式版本。

(4)存储器空间不足。

(1)找出用 CONFIG.SYS 或 ADDDRV 设定的设备驱动程序中不必要的文件,并加以删除。

(2)找出常驻程序中不必要的文件,并加以消除。

(3)停止与编译程序同时执行的其它程序的子过程处理。

例子 C6

命令行

CL C1002.C

出错

fatal error C1002:Compiler is out of heap space inPass 2

消除的例子

源文件 C1002.C 很大的时候,可分成 C1002—1.C 和 C1002—2.C 两个程序。

CL C1002—1.C C1002—2.C

fatal error C1003

C5 error count exceeds;stopping compilation

QC 同上

C6 同上

状况

由于程序内出错过多或过于严重,不能恢复。编译中止。

消除

从最先的出错开始依次修改,减少出错的数量。

fatal error C1004

C5 unexpected end—of—file found

QC 同上

C6 unexpected end—of—file found

状况

(1)错误的磁盘机上,没有编译程序处理临时文件的空间。必要的空间大小约为源文件

的二倍。

(2)不能发现与#if 对应的#endif。

(3)注释由于缺少 * /而没有终结。

(4)没有相应的函数。

消除

(1)删除不必要的驱动器程序,或者把指定临时磁盘的环境变数 TMP 设定到别的磁盘里去。

(2)追加#endif 语句

(3)追加 * /到源程序注释的最后。

(4)追加相应的函数。

例子 QC

出错的源程序	消除的例子
/* C1004.C */	/* C1004.C */
#define add(x,y)(x+y)	#define add(x,y)(x+y)
void main( )	void main( )
{	{
int C;	int C;
c=add(1,2;	c=add(1,2);←漏了 add
}	} 函数的括号,因此要追加上去。

出错:

C1004.C(12):fatal error C1004:unexpected EOF

参照

→C1057.C(9):fatal error C1057:unexpected
end—of—file in macro expansion
(missing ')'?)

→C6 C1004.C(9):fatal error C1057:unexpected
end—of—file in macro expansionmissing ')'?)

fatal error C1005

C5 string too big for buffer

QC 同上

C6 同上

状况

编译的中间文件的字符串超出了间隔的距离。

消除

将编译的中间文件的字符串修改成不超过间隔距离的限度。

fatal error C1006

C5 write error on compiler intermediate file

QC 同上

C6 write error on compiler—generated file

状况

不能生成编译工作中使用的中间文件

消除

原因是磁盘没有空间，所以须消除磁盘中不必要的文件。

fatal error C1007

C5 unrecognized flag '选择' in '程序'

QC 同上

C6 同上

状况

程序命令行编译的选择无效。

消除

确认选择可以使用后，再进行订正。

例子 C6

命令行

CL /FPa C1007.C

出错

fatal error C1007: unrecognized flag '—FPa' in 'qccom'

消除的例子

QC 中浮动小数点演算只有两种类型，指定/FPi87 和/FPi 两者中的一个。

参照

→Command line error D2011: only one floating point model allowed

fatal error C1008

C5 no input file specified

QC 同上

C6 同上

状况

没有指定编译的文件。

消除

指定编译文件名。

fatal error C1009

C5 Compiler limit: macros too deeply nested

QC 同上

C6 Compiler limit: macros nested too deeply

状况

同时展开了很多宏命令

消除

此为试读，需要完整PDF请访问：www.ertongbook.com