

742.04

Microsoft Visual C++ 5.0 程序员参考·第4卷

Microsoft Visual C++ 语言参考手册

(美) Microsoft Corporation 著
前导工作室 译

清华大学图书馆
藏书
清华大学出版社

(京)新登字 158 号

内 容 提 要

本书是 Microsoft Visual C++ 程序员参考系列手册(共四册)之四。该系列手册主要面向 C 和 C++ 程序员,为他们提供 MFC 类库、C 语言函数等方面的参考。

本书分为三个部分:第一部分介绍 C 语言,包括术语、概念、程序的结构以及函数等内容;第二部分介绍 C++ 语言,主要介绍运行类型信息(RTTI)和名字空间的有关知识;第三部分介绍预处理的相关知识。

Microsoft Visual C++ 语言参考手册

Microsoft Visual C++ Language Reference

Microsoft Corporation

Copyright © 1997 by Microsoft Corporation.

Original English language Edition Copyright © 1997 by Microsoft Corporation.

Published by arrangement with the original publisher, Microsoft Press,

a division of Microsoft Corporation, Redmond, Washington, U.S.A.

本书中文版由 Microsoft Press 授权清华大学出版社出版。

北京市版权局著作权合同登记号 图字 01-98-2221 号

版权所有,翻印必究。

本书封面贴有 Microsoft Press 激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

Microsoft Visual C++ 语言参考手册/(美)Microsoft Corporation 著;前导工作室译. —北京:清华大学出版社,1998.12

(Microsoft Visual C++ 5.0 程序员参考;第4卷)

ISBN 7-302-03253-X

I. M… II. ①M… ②前… III. C 语言-程序设计-手册 IV. TP312-62

中国版本图书馆 CIP 数据核字 (98) 第 35322 号

出 版 者: 清华大学出版社(北京清华大学校内,邮编 100084)

<http://www.tup.tsinghua.edu.cn>

责任编辑: 胡先福

印 刷 者: 清华大学印刷厂印刷

发 行 者: 新华书店总店北京发行所

开 本: 787×1092 1/16 印张: 37.5 字数: 935 千字

版 次: 1999 年 2 月第 1 版 1999 年 2 月第 1 次印刷

书 号: ISBN 7-302-03253-X/TP·1742

印 数: 0001 ~ 5000

定 价: 69.00 元

译 者 序

C 和 C++ 语言是非常流行的编程语言,它们以高效率、低开销和良好的可移植性而闻名。用 C/C++ 语言编写的程序速度快、结构紧凑,并且易于与系统接口。因此,C/C++ 语言很适合于用来进行系统编程。

Microsoft Visual C++ 程序员参考系列手册共包括四本书,它们是微软公司推出的 Microsoft Visual C++ 5.0 的完整文档。本书是这四册书中的第四卷,主要目的是向程序员介绍 C/C++ 语言。全书分为以下三部分:第一部分介绍 C 语言,包括术语、概念、程序的结构和函数等内容;第二部分介绍 C++ 语言,主要介绍运行类型信息(RTTI)和名字空间的有关知识;第三部分介绍预处理的相关知识。

本书由况军组织翻译,参加翻译工作的还有陈彦海、刘辉、孙议、张志敏、刘国庆、汤朝阳、史章军、刘汇、王建民、萧东等。全书由萧东、李士心进行校对,由陈安安录排。

由于时间仓促,且经验和水平有限,书中难免会有不足之处,恳请读者批评指正。

译 者

1998 年 4 月

目 录

第一部分 C 语言参考手册

引 言	3	2.3.2 作用域与可见性	29
0.1 本手册的组织	3	2.3.3 生命周期和可见性小结	30
0.2 本手册的范围	3	2.3.4 链接	31
0.3 ANSI 约定	3	2.4 名字空间	32
第 1 章 C 的语言成分	4	第 3 章 声明和类型	33
1.1 标记	4	3.1 声明概览	33
1.1.1 空白字符	4	3.2 存储类	35
1.1.2 注释	5	3.2.1 外部声明的存储类	
1.1.3 标记的求值	6	说明符	36
1.2 关键字	6	3.2.2 内部声明的存储类	
1.3 标识符	7	说明符	38
1.3.1 多字节与宽字符	9	3.2.3 函数声明中的存储类	
1.3.2 三连符(Tigraph)	9	说明符	40
1.4 常量	10	3.3 类型说明符	40
1.4.1 浮点常量	10	3.4 类型限定符	42
1.4.2 整数常量	13	3.5 声明符和变量声明	43
1.4.3 字符常量	16	3.5.1 简单变量声明	45
1.5 字符串常量	18	3.5.2 枚举声明	45
1.5.1 字符串常量的类型	19	3.5.3 结构声明	48
1.5.2 字符串常量的存储	19	3.5.4 联合声明	53
1.5.3 字符串常量的连接	19	3.5.5 数组声明	56
1.5.4 最大字符串的长度	20	3.5.6 指针声明	57
1.6 标点符号和特殊字符	20	3.5.7 基指针	60
		3.5.8 抽象声明	61
		3.6 理解更复杂的声明符	61
		3.7 初始化	63
		3.7.1 初始化比例类型	64
		3.7.2 初始化聚合类型	65
		3.7.3 初始化字符串	68
		3.8 基本类型的存储	69
		3.9 不完全类型	73
		3.10 Typedef 定义	73
		3.11 扩展存储类属性	76
		3.11.1 DLL 输入和输出	76
		3.11.2 Naked 函数	77

3.11.3 线程局部存储	77	5.14 try-finally 语句	127
5.15 while 语句	128		
第 4 章 表达式和赋值	79	第 6 章 函数	130
4.1 操作符和表达式	79	6.1 概述	130
4.1.1 基本表达式	79	6.2 函数定义	131
4.1.2 左值和右值表达式	81	6.2.1 函数属性	133
4.1.3 常量表达式	81	6.2.2 DLL 输入和输出函数	134
4.1.4 表达式求值	82	6.2.3 Naked 函数	137
4.2 运算符	83	6.2.4 存储类	140
4.2.1 求值的优先级和顺序	84	6.2.5 返回类型	141
4.2.2 常用的算术转换	86	6.2.6 形参	142
4.2.3 后缀运算符	87	6.2.7 函数体	144
4.2.4 一元运算符	92	6.3 函数原型	144
4.2.5 强制运算符	95	6.4 函数调用	146
4.2.6 乘运算符	95	6.4.1 实参	147
4.2.7 加运算符	97	6.4.2 使用可变个数的 参数调用	149
4.2.8 按位移运算符	98	6.4.3 递归函数	149
4.2.9 关系和相等运算符	99		
4.2.10 按位运算符	101	附录 A C 语言语法概述	151
4.2.11 逻辑运算符	102	A.1 定义和约定	151
4.2.12 条件表达式运算符	103	A.2 词法	151
4.2.13 赋值运算符	104	A.2.1 单词	151
4.2.14 顺序求值运算符	106	A.2.2 关键字	152
4.3 类型转换	106	A.2.3 标识符	153
4.3.1 赋值转换	107	A.2.4 常量	153
4.3.2 类型强制转换	111	A.2.5 串字符	155
4.3.3 函数调用转换	112	A.2.6 操作符	155
		A.2.7 标点符号	155
第 5 章 语句	114	A.3 短语结构语法	156
5.1 概述	114	A.3.1 表达式	156
5.2 break 语句	115	A.3.2 声明	158
5.3 复合语句	115	A.3.3 语句	161
5.4 continue 语句	116	A.3.4 外部定义	162
5.5 do-while 语句	117		
5.6 表达式语句	118	附录 B 由实现所定义的行为	163
5.7 for 语句	118	B.1 翻译; 诊断	163
5.8 goto 和带标号的语句	119	B.2 环境	163
5.9 if 语句	120	B.2.1 main 的参数	163
5.10 空语句	121	B.2.2 交互设备	164
5.11 return 语句	122	B.3 标识符	164
5.12 switch 语句	123		
5.13 try-except 语句	125		

B.3.1	不具有外部链接的 有效字符	164		文件名	171
B.3.2	具有外部链接的 有效字符	164	B.13.3	用引号蕴含的 文件名	171
B.3.3	大写和小写	164	B.13.4	字符序列	171
B.4	字符	164	B.13.5	编译指示	171
B.4.1	ASCII 字符集	164	B.13.6	默认日期和时间	172
B.4.2	多字节字符	164	B.14	库函数	172
B.4.3	每个字符的位数	165	B.14.1	NULL 宏	172
B.4.4	字符集	165	B.14.2	由 assert 函数打印的 诊断信息	172
B.4.5	未表示的字符常量	165	B.14.3	字符检测	172
B.4.6	宽字符	165	B.14.4	值域错	173
B.4.7	转换多字节字符	166	B.14.5	浮点值的下溢	173
B.4.8	char 值的范围	166	B.14.6	fmod 函数	173
B.5	整数	166	B.14.7	signal 函数	173
B.5.1	整数值的范围	166	B.14.8	默认信号	173
B.5.2	整数的降级	166	B.14.9	结束换行字符	173
B.5.3	有符号的按位操作	167	B.14.10	空行	174
B.5.4	余数	167	B.14.11	空字符	174
B.5.5	右移	167	B.14.12	插入模式中的 文件位置	174
B.6	浮点数学	168	B.14.13	文本文件的截断	174
B.6.1	值	168	B.14.14	文件缓冲	174
B.6.2	把整数强制为浮点值	168	B.14.15	零长度的文件	174
B.6.3	浮点值的截断	168	B.14.16	文件名	174
B.7	数组和指针	168	B.14.17	文件访问限制	174
B.7.1	最大数组大小	168	B.14.18	删除打开的文件	174
B.7.2	指针减法	168	B.14.19	用一个已存在的 名字重新命名	175
B.8	寄存器;寄存器的可用性	169	B.14.20	读指针值	175
B.9	结构,联合,枚举和位域	169	B.14.21	读范围	175
B.9.1	对联合的非法访问	169	B.14.22	文件位置错	175
B.9.2	结构成员的填充 和对齐	169	B.14.23	由 perror 函数生成 的信息	175
B.9.3	位域的符号	169	B.14.24	分配零内存	176
B.9.4	位域的存储	169	B.14.25	abort 函数	176
B.9.5	enum 类型	170	B.14.26	atexit 函数	176
B.10	限制符;访问 Volatile 对象	170	B.14.27	环境名	177
B.11	声明符;最大数目	170	B.14.28	system 函数	177
B.12	语句;Switch 语句的限制	170	B.14.29	strerror 函数	177
B.13	预处理命令	171	B.14.30	时区	178
B.13.1	字符常量和条件 蕴含	171	B.14.31	clock 函数	178
B.13.2	用尖括号蕴含的				

第二部分 C++ 语言参考手册

引言	181	2.6.2 对启动附加的考虑	213
0.1 本手册的内容	181	2.6.3 对结束附加的考虑	213
0.2 本手册中的组织	181	2.7 存储类	215
0.3 本手册中的特殊术语	182	2.7.1 自动存储类	215
第 1 章 词法约定	183	2.7.2 静态存储类	215
1.1 概述	183	2.7.3 寄存器存储类	216
1.2 单词	184	2.7.4 外部存储类	216
1.3 注释	185	2.7.5 对象的初始化	216
1.4 标识符	186	2.8 类型	218
1.5 C++ 关键字	187	2.8.1 基本类型	218
1.6 标点符号	188	2.8.2 带长度的整数类型	220
1.7 操作符	188	2.8.3 派生类型	220
1.8 文字	191	2.8.4 类型名	226
1.8.1 整数常数	191	2.9 左值与右值	227
1.8.2 字符常数	193	2.10 数字界限	228
1.8.3 浮点常数	195	2.10.1 整数界限	228
1.8.4 串文字	196	2.10.2 浮点界限	228
第 2 章 基本概念	199	第 3 章 标准转换	231
2.1 术语	199	3.1 整数升级	231
2.2 声明与定义	200	3.2 整数转换	232
2.2.1 声明	200	3.2.1 将带符号的数转换为无 符号的数	232
2.2.2 定义	201	3.2.2 将无符号的数转换为带 符号的数	233
2.3 作用域	201	3.2.3 标准转换	233
2.3.1 声明点	202	3.3 浮点转换	233
2.3.2 隐藏名字	203	3.4 浮点和整数的转换	234
2.3.3 函数形参的作用域	205	3.4.1 浮点到整数	234
2.4 程序和链接	205	3.4.2 整数到浮点	234
2.4.1 链接的种类	205	3.5 算术转换	234
2.4.2 具有文件作用域的名字的 链接	205	3.6 指针转换	235
2.4.3 具有类作用域的名字的 链接	206	3.6.1 空指针	235
2.4.4 具有块作用域的名字的 链接	206	3.6.2 指向 void 类型的指针	235
2.4.5 没有链接的名字	206	3.6.3 指向对象的指针	236
2.4.6 对非 C++ 函数的链接	208	3.6.4 指向函数的指针	236
2.5 启动和结束	209	3.6.5 指向类的指针	236
2.6 定制命令行处理	212	3.6.6 指针表达式	237
2.6.1 程序结束	212	3.6.7 被 const 或 volatile 限定的 指针	237
		3.7 引用转换	238

3.8 指向成员的指针的转换	238	5.7.1 break 语句	303
3.8.1 整常数表达式	238	5.7.2 continue 语句	304
3.8.2 指向基类成员的指针	238	5.7.3 return 语句	305
第 4 章 表达式	240	5.7.4 goto 语句	305
4.1 表达式的种类	240	5.8 声明语句	305
4.1.1 初等表达式	240	5.8.1 自动对象的声明	306
4.1.2 后缀表达式	242	5.8.2 静态对象的声明	307
4.1.3 使用单目操作符 的表达式	251	5.9 异常处理	309
4.1.4 使用双目操作符 的表达式	260	5.9.1 try、catch 及 throw 语句	310
4.1.5 使用条件操作符 的表达式	271	5.9.2 构造的异常处理	315
4.1.6 常数表达式	272	第 6 章 声明	316
4.1.7 使用显式类型转换的 表达式	273	6.1 说明符	316
4.1.8 使用指向成员的指针 操作符的表达式	276	6.1.1 存储类说明符	317
4.2 表达式的语义	279	6.1.2 函数说明符	319
4.2.1 求值的顺序	279	6.1.3 typedef 说明符	322
4.2.2 表达式的注释	281	6.1.4 friend 说明符	325
4.3 Casting	282	6.1.5 类型说明符	325
4.3.1 Casting 操作符	282	6.2 枚举声明	330
4.3.2 运行类型信息	288	6.2.1 枚举符名字	333
第 5 章 语句	292	6.2.2 枚举符常量的定义	333
5.1 概述	292	6.2.3 转换和枚举类型	334
5.2 标号语句	293	6.3 链接规范	335
5.2.1 配合 goto 语句 使用标号	293	6.4 模板规范	337
5.2.2 在 case 语句中 使用标号	294	6.4.1 引用模板	337
5.3 表达式语句	294	6.4.2 函数模板	338
5.4 复合语句(块)	295	6.4.3 成员函数模板	338
5.5 选择语句	296	6.4.4 显式实例化	339
5.5.1 if 语句	296	6.4.5 与其他实现方法的区别	339
5.5.2 switch 语句	297	6.5 名字空间	340
5.6 迭代语句	299	6.5.1 名字空间的声明	341
5.6.1 while 语句	300	6.5.2 名字空间的定义	342
5.6.2 do 语句	301	6.5.3 名字空间成员定义	343
5.6.3 for 语句	301	6.5.4 名字空间的别名	344
5.7 跳转语句	303	6.5.5 Using 声明	344
		6.5.6 Using 指令	349
		6.5.7 显式限定	352
		第 7 章 声明符	353
		7.1 概述	353
		7.2 类型名	355
		7.3 抽象声明符	355
		7.3.1 消除二义性	356

7.3.2	指针	357	8.9	类作用域中的类型名	412
7.3.3	引用	359	第 9 章 派生类		413
7.3.4	成员指针	364	9.1	概述	413
7.3.5	数组	369	9.1.1	单继承	413
7.3.6	函数声明	372	9.1.2	多重继承	417
7.3.7	默认参数	377	9.1.3	虚基类层次结构	417
7.4	函数定义	380	9.1.4	类协议实现	418
7.5	初始化	382	9.1.5	基类	418
7.5.1	初始化 const 对象的指针	383	9.2	多重基类	419
7.5.2	非初始化对象	383	9.2.1	虚基类	420
7.5.3	初始化静态成员	384	9.2.2	名字的二义性	421
7.5.4	初始化聚集	384	9.3	虚函数	424
7.5.5	初始化字符数组	387	9.4	抽象类	428
7.5.6	初始化引用	387	9.5	作用域规则总结	430
第 8 章 类		389	9.5.1	二义性	430
8.1	概述	389	9.5.2	全局名	430
8.1.1	定义类类型	390	9.5.3	名字和受限名	430
8.1.2	类类型对象	392	9.5.4	函数参数名	431
8.2	类名	393	9.5.5	构造函数初始化	431
8.2.1	声明和访问类名	394	第 10 章 成员访问控制		432
8.2.2	typedef 语句和类	395	10.1	控制对类成员的访问	432
8.3	类成员	395	10.2	访问说明符	432
8.3.1	类成员声明语法	397	10.3	基类的访问说明符	433
8.3.2	在成员中声明未定义长度的数组	398	10.4	友元	436
8.3.3	类成员数据的存储	399	10.4.1	友元函数	436
8.3.4	成员命名限制	399	10.4.2	作为友元的类成员函数和类	438
8.4	成员函数	399	10.4.3	友元声明	439
8.4.1	成员函数综述	401	10.4.4	在类声明中定义友元函数	440
8.4.2	this 指针	402	10.5	保护成员访问	440
8.5	静态数据成员	404	10.6	对虚函数的访问	440
8.6	联合	405	10.7	多重访问	441
8.6.1	联合中的成员函数	406	第 11 章 特殊成员函数		442
8.6.2	作为类类型的联合	406	11.1	构造函数	443
8.6.3	联合的成员数据	406	11.1.1	构造函数做什么	443
8.6.4	无名联合	407	11.1.2	声明构造函数的规则	443
8.7	位域	408	11.1.3	构造函数和数组	446
8.8	嵌套类声明	409	11.1.4	构造的次序	446
8.8.1	访问特权和嵌套类	410	11.2	析构函数	446
8.8.2	嵌套类中的成员函数	410			
8.8.3	友元函数和嵌套类	411			

11.2.1	声明析构函数	447	A.2	表达式	493
11.2.2	使用析构函数	448	A.3	声明	498
11.2.3	析构的顺序	448	A.4	声明符	502
11.2.4	显式析构函数调用	449	A.5	类	504
11.3	临时对象	450	A.6	语句	506
11.4	转换	451	A.7	Microsoft 扩展特性	507
11.4.1	转换构造函数	451	附录 B	Microsoft 特定的修饰符	509
11.4.2	转换函数	453	B.1	基址寻址	509
11.5	new 和 delete 操作符	455	B.2	调用和命名约定修饰符	510
11.5.1	operator new 函数	455	B.3	扩展存储类属性	510
11.5.2	处理内存不够的情况	456	B.3.1	扩展属性语法	511
11.5.3	operator delete 函数	460	B.3.2	线程属性	511
11.6	使用特殊成员函数进行		B.3.3	naked 属性	513
初始化		462	B.3.4	dllexport 和 dllimport	
11.6.1	显式初始化	462	属性		515
11.6.2	初始化数组	464	B.3.5	在 C++ 中使用 dllimport 和	
11.6.3	初始化静态对象	465	dllexport		518
11.6.4	初始化基和成员	465	B.4	嵌入汇编器	520
11.7	拷贝类对象	467	附录 C	Microsoft 特定编译器的 COM	
11.7.1	编译器生成的拷贝	468	支持类		521
11.7.2	按成员赋值和初始化	469	C.1	_com_error	521
第 12 章	重载	471	C.1.1	成员函数	522
12.1	概述	471	C.1.2	操作符	525
12.1.1	参数类型的区别	471	C.2	_com_ptr_t	525
12.1.2	重载函数的限制	472	C.2.1	成员函数	526
12.2	声明匹配	473	C.2.2	操作符	529
12.3	参数匹配	475	C.3	_bstr_t	532
12.3.1	参数匹配和 this 指针	476	C.3.1	成员函数	532
12.3.2	参数匹配和转换	477	C.3.2	操作符	533
12.4	重载函数的地址	480	C.4	_variant_t	535
12.5	重载操作符	480	C.4.1	成员函数	535
12.5.1	操作符重载的通用		C.4.2	操作符	538
规则		482	附录 D	表格	541
12.5.2	一元操作符	483	D.1	ASCII 字符代码表 1	542
12.5.3	二元操作符	486	D.2	ASCII 字符代码表 2	543
12.5.4	赋值	488	D.3	ASCII 多语种代码表	544
12.5.5	函数调用	488	D.4	ANSI 字符代码表	545
12.5.6	下标	489	D.5	键盘代码表 1	545
12.5.7	类成员访问	490	D.6	键盘代码表 2	546
附录 A	文法小结	492			
A.1	关键字	492			

第三部分 预处理器参考手册

引言	549	2.1.2 pointers_to_members	573
第1章 预处理器	550	2.1.3 vtordisp	574
1.1 编译的阶段	550	2.2 C和C++编译器的编译指令	574
1.2 预处理命令	551	2.2.1 alloc_text	574
1.2.1 #define 指令	551	2.2.2 auto_inline	575
1.2.2 #error 指令	553	2.2.3 bss_seg	575
1.2.3 #if, #elif, #else 和 #endif 指令	553	2.2.4 check_stack	575
1.2.4 #ifdef 和 #ifndef 指令	557	2.2.5 code_seg	576
1.2.5 #import 指令	557	2.2.6 const_seg	576
1.2.6 #include 指令	562	2.2.7 comment	576
1.2.7 #line 指令	564	2.2.8 component	577
1.2.8 空(NULL)指令	565	2.2.9 data_seg	578
1.2.9 #undef 指令	565	2.2.10 function	578
1.3 预处理操作符	566	2.2.11 hdrstop	578
1.3.1 字符串化符(#)	566	2.2.12 include_alias	579
1.3.2 字符化符(# @)	567	2.2.13 inline_depth	580
1.3.3 符号粘贴符(##)	567	2.2.14 inline_recursion	581
1.4 宏	568	2.2.15 intrinsic	581
1.4.1 宏和C++	568	2.2.16 message	582
1.4.2 预定义宏	569	2.2.17 once	582
第2章 Pragma 指令	572	2.2.18 optimize	582
2.1 C++编译器专有的编译指令	572	2.2.19 pack	583
2.1.1 init-seg	572	2.2.20 setlocale	584
		2.2.21 warning	584
		附录 语法总结	586

第一部分

C 语言参考手册

引 言

0.1 本手册的组织

- C 的语言成分
- 程序的结构
- 声明和类型
- 表达式和赋值
- 语句
- 函数
- C 语言语法总结
- 实现定义的行为

0.2 本手册的范围

C 是一种十分灵活的编程语言,它将编程中的许多决定交给程序员去做。由于具有这种特性,C 语言中诸如数据类型转换这样的限制比较少。虽然这种特性可以使程序员的编程工作更容易,但程序员必须十分了解该语言才能理解程序将怎样运行。本部分提供 C 语言成份的信息和 Microsoft 实现的特性。C 语言语法源于 ANSI X3.159-1989, American National Standard for Information Systems-Programming Language-C(以下称为 ANSI C 标准),尽管这不是 ANSI C 标准的一部分。附录 A“C 语言语法总结”将介绍该语法,并说明如何阅读和使用该语法的定义。

本部分不讨论 C++ 语言的编程。有关 C++ 的内容请参考《C++ 语言参考手册》部分。

注意 有关 Microsoft 产品的支持,参见文件 PSS.HLP。

0.3 ANSI 约定

Microsoft C 符合 ANSI C 标准所述的 C 语言标准。和联机参考一样,本书中 Microsoft 对 ANSI C 标准的扩充将用文字和语法注明。由于这些扩充不是 ANSI C 标准的一部分,所以它们的使用可能会限制程序在不同系统中的可移植性。在默认情况下允许使用 Microsoft 所做的扩充。若不想使用扩充,应指定编译选项/Za。使用/Za 时,所有非 ANSI 代码都将产生错误或警告。

第 1 章 C 的语言成分

本章描述 C 编程语言的成分,包括组成 C 程序的名字、数字和字符。ANSI C 语法将这些组成部分记作“标记(token)”。本章介绍如何定义标记以及编译器如何解释这些标记。

本章将讨论如下的内容:

- 标记
- 注释
- 关键字
- 标识符
- 常量
- 字符串
- 标点符号与特殊字符

本章还包括三连符(trigraph)、浮点常数、整常数和转义符序列的参照表。

“运算符”是指定如何运算的符号(包括单一字符和字符组合)。每一个符号都被编译为一个单一的单位,称为标记。要了解详细的信息,请参见第 4 章中的“运算符”。

1.1 标 记

在 C 源程序中,被编译器识别的最基本的成分是“标记(token)”。标记是源程序文本中编译器不可再分割的基本成分。

语法

token :

keyword

identifier

constant

string-literal

operator

punctuator

注意 有关 ANSI 语法规则的解释,参见附录 A 中的“C 语言语法概述”。

本章所描述的关键字、标识符、常量、字符串和运算符都是标记的实例。标点字符,如方括号([])、花括号({ })、圆括号(())和逗号(,)也都是标记。

1.1.1 空白字符

空格、制表符、换行、回车、换页、垂直制表符和新行字符(newline character)被称为“空白

字符”，因为这些字符都用于同一目的即在字间和行间插入空间以便于阅读。标记由空白字符或其他标记，如运算符和标点进行分隔。当编译器分析代码时，忽略空白字符，除非它们作为分隔符或者作为字符常量或字符串常量的组成部分。使用空白字符可使程序更易于阅读。

注意 编译器把注释当作空白。

1.1.2 注释

“注释”是以斜线和星号的组合(/*)开始的字符序列，被编译器解释为单个空白字符，且注释被编译器所忽略。注释包括可表示字符集的任何字符组合，包括换行符，但注释结束符(/*)除外。注释可占多于一行，但不能嵌套。

注释可以出现在任何允许空白字符出现的地方。因为编译器把注释当作单个空白字符，所以标记中不能包括注释。编译器忽略注释中的所有字符。

使用注释可以为源代码增加文档。下面的例子是编译器可以接受的注释：

```
/* Comments can contain keywords such as
   for and while without generating errors. */
```

注释可以和源代码语句出现在同一行：

```
printf("hello \n");/* comments can go here */
```

可以在函数或程序模块前使用注释作描述说明：

```
/* MATHERR.C illustrates writing an error routine
   * for math functions .
   */
```

因为注释不能嵌套，所以下面的例子将导致出错：

```
/* Comment out this routine for testing
   /* Open file */
   fh = _open("myfile. c", _o_ RDONLY)
   .
   .
   .
*/
```

错误的发生是因为编译器在识别出 Open file 之后的第一个 /* 之后，就把它当作注释的结束了。在处理其后的文字中，当注释之外出现 /* 就会产生错误。

在测试时，可以使用注释把几行代码设置为非活动状态。预处理指令 # if 和 # endif 与条件编译指令是另一种实现这种方法的功能。有关的内容参见“预处理参考”中的“预处理指令”。

Microsoft 特性→

Microsoft 的编译器还支持由两条斜线(//)开始的单行注释。但如果编译时使用了 /Za (ANSI 标准)选项，那么这类注释会导致出错。这种类型的注释不能延续到下一行。

```
// This is valid comment
```

由两条斜线(//)引出的注释以回车符作为结束,但该回车符前不能有转义字符。下例中的回车符前有一条反斜线(\),因而构成“转义序列”。这一转义序列使编译器把回车符当作注释行的一部分(详细的内容参见“转义序列”)。

```
//my comment \  
i++;
```

语句 `i++;` 被认为是注释而忽略。

默认时 Microsoft C 扩展有效。用 `/Za` 可禁止该扩展。

Microsoft 特性结束

1.1.3 标记的求值

当编译器解释标记时,会在一个标记中包括尽可能多的字符,然后再扫描下一个标记。由于这一特性,如果标记没有用空白进行适当的分隔,那么编译器可能不会像你想象的那样解释标记。考虑如下表达式:

```
i+++j
```

在这个例子中,编译器从三个加号中先找出尽可能长的运算符(`++`),然后再把剩下的加号作为加法运算符。于是这个表达式就被编译为 `(i++)+(j)`,而不是 `(i)+(++j)`。在这个例子以及类似的情况下,应该使用空白和圆括号来避免意义不明确,并保证表达式正确地求值。

Microsoft 特性→

C 编译器把 `Ctrl + Z` 字符当作文件结束符。在 `Ctrl + Z` 之后的文字均被忽略。

Microsoft 特性结束

1.2 关 键 字

“关键字”是对 C 编译器具有特殊意义的词。编译器在第 7 和第 8 遍扫描中,标识符不得与 C 关键字有相同的拼法和大小写(参见《预处理器参考》中的有关“编译扫描(translation phase)”的描述),C 语言使用如下关键字:

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

关键词不能重新定义,但可以通过在编译前使用 C 预处理指令说明替代关键字的

文本。

Microsoft 特性→

ANSI C 标准允许以两个下划线开始的标识符作为编译器保留的关键字。Microsoft 约定 Microsoft 特有的关键字均以双下划线开始,它们也不能作为标识符名。要了解 ANSI 关于命名标识符的规则,包括双下划线的使用,请参见 1.3 节的“标识符”。

Microsoft C 编译器识别以下的关键字和特殊标识符:

<code>_asm</code>	<code>_dllimport²</code>	<code>_int8</code>	<code>_naked²</code>
<code>_based¹</code>	<code>_except</code>	<code>_int16</code>	<code>_stdcall</code>
<code>_cdecl</code>	<code>_fastcall</code>	<code>_int32</code>	<code>_thread²</code>
<code>_declspec</code>	<code>_finally</code>	<code>_int64</code>	<code>_try</code>
<code>_dllexport²</code>	<code>_inline</code>	<code>_leave</code>	

- ¹. 关键字 `_based` 仅限于 32 位目标的编译。
- ². 当与 `_declspec` 一起使用时是特殊标识符;在其他环境中使用时没有限制。

Microsoft 扩展在默认时是允许的。如果要保证程序的可移植性,那么可以在编译时使用 `/Za` 选项禁止 Microsoft 扩展(按照与 ANSI 兼容的方式编译)。当你这样做时,Microsoft 的关键字被禁止使用。

当允许使用 Microsoft 扩展时,程序可以使用上面列出的关键字。为服从 ANSI 的规则,这些关键字多数是以双下划线开头的,只有四个例外: `_dllexport`, `_dllimport`, `_naked` 和 `_thread`。而这四个例外关键字只能与 `_declspec` 共同使用,因此无需以双下划线开头。为了向后兼容,其余的单下划线关键字仍然被支持。

Microsoft 特性结束

1.3 标识符

“标识符”或“符号”是程序员为程序中变量、类型、函数和标号提供的名字。标识符不能与关键字在拼法和大小写上相同。关键字(不论是标准 C 或 Microsoft C)不能用作标识符,它们被保留用于特殊的目的。通过在变量、类型、函数的声明中进行说明可以创建标识符。在这个例子中, `result` 是一个整型变量标识符, `main` 和 `printf` 是函数的标识符。

```
void main()  
{  
    int result;  
    if ( result! = 0 )  
        printf( "Bad file handle \n" )  
}
```

标识符一旦被声明就可以在其后的程序语句中引用。

有一种特殊的标识符,称为语句标号,可以用在 `goto` 语句中(声明描述见第 3 章“声明和类型”。语句标号见第 5 章 5.8 节“`goto` 和带标号的语句”)。