1. 名称cout表示在默认情况下对应于命令行的标准输出流；cin为来自键盘的互补输入流；名称endl表示换行字符。
2. 变量名可以包含字母A-z（大写或小写）、数字0~9和下划线字符。其它字符则不允许使用。变量名长度最好限定在31个字符。最好避免采用以下划线开头的变量名，因为这与具有相同形式的标准系统变量会发生潜在的冲突；同样的原因，避免使用以双下划线开头的名称；变量名中不允许有空格；变量名是区分大小写的；空白字符一般不能出现在变量名内；C++中通常把以大写字母开头的名称用于类的命名，把以小写字母开头的名称用于变量。
3. C++中的关键字也是区分大小写的。
4. C++基本数据类型分为3类：存储整数的类型、存储非整数的类型（浮点型）、以及指定空的值集或者不指定任何类型的void类型。
5. TRUE和FALSE不是C++中的关键字，只是在MFC内部定义的符号；TRUE和FALSE不是合法的bool值，所以不能混淆true和TRUE。
6. ISO/ANSI C++中的基本类型

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 字节数 | 值域（VC2005） | 字面值 |
| bool | 1 | true或false | true和false |
| char | 1 | 默认情况下和signed char一样 | ‘A’ |
| signed char | 1 | -128到127 | ‘Z’ |
| unsigned char | 1 | 0到255 | ‘8’ |
| wchar\_t | 2 | 0到65535 | L’A’ |
| short | 2 | -32768到32767 | -77 |
| unsigned short | 2 | 0到65535 | 65 |
| int | 4 | -2147483648到2147483647 | -77 |
| unsigned int | 4 | 0到4294967295 | 65 |
| long | 4 | -2147483648到2147483647 | -77L |
| unsigned long | 4 | 0到4294967295 | 65L |
| float | 4 | ±3.4x10±38,精度大约为7个数位 | 3.14f |
| double | 8 | ±1.7x10±308,精度大约为15个数位 | 1.414 |
| long double | 8 | ±1.7x10±308,精度大约为15个数位 | 1.414L |

1. 枚举类型：

例如 enum Week{Mon, Tues, Wed, Thurs, Fri, Sat, Sun} thisWeek;

可以把一个枚举常量作为变量thisWeek的值赋给它，例如：

thisWeek = Tues;

可以定义枚举类型为Week的变量，例如：

Week nextWeek;

枚举类型默认情况下为int类型，可以把枚举类型指定为任何signed或unsigned整数类型（short/int/long/char）或者bool类型，也可以将枚举类型定义为：

enum Week : char{Monday = ‘M’, Tuesday = ‘T’, Wednesday = ‘W’, Thursday = ‘T’, Friday = ‘F’, Saturday = ‘S’, Sunday = ‘S’};

enum State : bool{On = true, Off = false};

枚举中赋予的值不一定遵循升序方式，如果没有显式地指定所有值，那么将按照从最后一个指定值开始递增1的方式给每个计数器赋值。

1. 默认的流输出只把输出值中的数字输出，它没有规定恰当地把连续的输出值分隔开以便于区分。可以使用操作符setw(n)，它输出的值在n个空格宽的字段中遵循右对齐。如果只插入了一个setw()，那么它只作用于插入它之后要输出的第一个数值。后面的所有数值都将以默认的方式输出。
2. 转义序列

|  |  |
| --- | --- |
| 转义序列 | 作用 |
| \a | 发出蜂鸣声 |
| \n | 换行 |
| \’ | 单引号 |
| \\ | 反斜杠 |
| \b | 退格 |
| \t | 制表符 |
| \” | 双引号 |
| \? | 问号 |

1. const变量类型用于定义程序中使用的常量，不能合法地把它放在赋值运算符的左侧。
2. a / = b + c;与a = a / (b + c);相同

逗号运算符：num4 = (num1 = 10, num2 = 20, num3 = 30);变量num4接受赋值语句中最后一个赋值语句的值，此时num4 = 30;

如果没有括号，则num4 = num1 = 10;num4的值为10；

1. 运算符优先顺序按照优先级从上到下按照关联性决定执行方向

|  |  |
| --- | --- |
| 运算符 | 关联性 |
| :: | 从左向右 |
| () [] -> . | 从左向右 |
| ! ~ +(一元) –(一元) ++ -- &(一元) \*(一元) (typecast) static\_cast const\_cast dynamic\_cast reinterpret\_cast sizeof new delete…typeid | 从右向左 |
| .\*(一元) ->\* | 从左向右 |
| \* / % | 从左向右 |
| + - | 从左向右 |
| << >> | 从左向右 |
| < <= > >= | 从左向右 |
| == != | 从左向右 |
| & | 从左向右 |
| ^ | 从左向右 |
| | | 从左向右 |
| && | 从左向右 |
| >> | 从左向右 |
| ?:(条件运算符) | 从右向左 |
| = \*= /= %= += -= &= ^= |= <<= >>= | 从右向左 |
| , | 从左向右 |

1. 显式类型强制转换语句static\_cast<要转换的类型>(表达式)，static\_cast表明将静态地检查类型强制转换，也就是说，在编译程序时进行检查；dynamic\_cast对转换进行动态地检查，也就是说，在执行程序时进行检查；const\_cast用于删除表达式中的.const属性；reinterpret\_cast是一种无条件的强制转换。旧的写法采用的是(要转换的类型)表达式。推荐采用前一种写法。
2. 按位运算符：&按位与；|按位或；^按位异或；~取反；>>右移；<<左移
3. 作用域解析运算符（::）：如果作用域解析运算符的左操作数指定了一个名称，那么在包含这个名称的命名空间中，编译器将搜索指定为右操作数的名称。如果在运算符的前面没有指定一个命名空间名，那么编译器将在全局命名空间中搜索运算符后面的名称。
4. 静态变量（static）的默认初值始终是0，它将被转换成适合于该变量的类型。如果没有对自动变量进行初始化，那么它们包含的数据将是最近使用它们所占用内存的程序留下的无用值。
5. 命名空间：using namespace std;这个语句是using指令；using std::vector;这个语句是using声明；推荐使用using声明而不使用using指令。
6. 不能在函数内声明命名空间，命名空间包含函数、全局变量和其他有名的实体，如程序中的类。对于一个具有给定名称的命名空间，可以有多个声明，所有具有给定名称的命名空间块的内容都可以在同一个命名空间内。
7. #include<>包含的文件通常为C++库文件，系统将按照INCLUDE环境变量指定的路径搜索C++库文件；#include””包含的文件系统将搜索文件夹，如果没有找到则到包含当前文件的目录中进行搜索，如果没有找到则搜索库目录。