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，其中第一位是字母，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，下划线在这里被认为是字母。
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，n为负数则左对齐。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。
8. 关键字typedef允许定义其它类型的同义词。
9. If语句后如果没写大括号，则只有缩进后的第一条语句从属于if。If语句将条件返回的非零值解释为true，将零值解释为false。else总是属于前面最近的、还没有对应else的if。
10. 将小写字母转换为大写的ISO/ANSI C++库函数为toupper()，位于标准库文件<ctype>中。
11. Case语句后如果不写break语句，则程序将继续执行后面那种情形的语句。为标识不同情形而制定的表达式必须是常量，任何两个分情形常量都不能相同，不同的情形不必一定要采取不同的动作。
12. 在for循环中可以采用其他数据类型，例如char、double等，但要注意不应该编写循环的结束取决于浮点循环计数器达到某个精确值这样的for循环，因为许多十进制数不能以二进制浮点形式精确表示，因此积累数值中可能积累误差。
13. do-while循环在循环结束时才检查循环条件，而while循环和for循环则在循环开始时检查循环条件。因此，do-while循环语句总是至少执行一次。
14. 数组指定的初始化数值不能比数组的元素多，但可以比数组的元素个数少。如果少，则列表中的初值被分配给从第一个元素（对应于索引值0）开始的连续元素。那些没有得到初值的数组元素被初始化为0。如果根本没有提供初始化列表，则数组元素包含的将是无用数据。将整个数组初始化为0的便捷方法是仅指定一个初值0。倘若提供初值的话，我们还可以省略数值型数组的长度。数组元素的个数由指定的初值数量决定。
15. 字符串终止符表明字符串已结束，该字符由转义序列’\0’定义，有时被称为空字符，占用一个字节，其中8位全为0。字符串中每个字符占用一个字节，因此算上最后的空字符，字符串需要的字节数比包含的字符数多一个。终止字符’\0’是编译器自动添加的，但是我们在给字符数组定义元素数量时必须考虑到终止字符的存在。wcslen()函数返回宽字符串的长度，strlen()函数返回字符串的长度。
16. 多维数组：数组在内存中的存储方式是使最右边的索引值最快地变化。data[3][4]的数据存储：data[0][0]-> data[0][1]-> data[0][2]-> data[0][3]-> data[1][0]-> data[1][1]-> data[1][2]-> data[1][3]-> data[2][0]-> data[2][1]-> data[2][2]-> data[2][3]。在二维或者多维数组中，最右边的一维必须总是确定的。
17. 取址运算符&又名引用运算符。间接运算符\*又名解除引用运算符，与某个指针一起使用可以存取该指针指向的变量的内容，而该过程被称为解除对指针的引用。
18. char\*类型的指针可以用字符串字面值初始化，例如：

char\* proverb = “A miss is as good as a mile”;

这条语句用引号之间的字符串创建某个以/0终止的字符串字面值（实际上是const char类型的数组）。该字面值的地址将是其第一个字符的地址。

1. 对于指针数组char\* pstr[] = {};，每一个成员都是一个指针，指向一个地址，大小为4；表达式cout << (sizeof pstr)/(sizeof pstr[0]);将整个指针数组占用的字节数量除以数组中第一个元素占用的字节数量。因为该数组中各个元素占用相同的存储空间，所以结果是数组元素的数量。
2. 区分下列三种情况：
3. 指向常量对象的指针；
4. 指向某个对象的常量指针；
5. 指向常量对象的常量指针；
6. 我们不能修改被指向的对象，但可以使指针指向其它对象，例如：const char\* pstring = “Some text”;
7. 我们不能修改指针中存储的地址，但可以修改指针指向的对象，例如：char\* const pstring = “Some text”;
8. 指针和被指向的对象都被定义成常量，因此都不能被修改，例如：const char\* const pstring = “Some text”;
9. 尽管我们可以在表达式中像使用指针一样使用数组名，但数组名不是指针，因为它表示的地址是固定的。
10. double beans[3][4];

double\* pbeans = NULL;

pbeans = &beans[0][0];

pbeans = beans[0];

重新定义pbeans: double (\*pbeans)[4];此处的圆括弧是必需的，否则声明的将是一个指针数组。此时pbeans = beans;合法，但该指针只能用于存储规定大小的数组的地址。

1. delete [] pstr;使用方括号是为了指出要删除的是一个数组。删除后还应该将指针重新设置为0。
2. 可以使用变量来指定new分配的一维数组的大小。对二维或多维数组来说同样如此，但仅限于用变量指定最左边的一维。所有其他维都必须是常量或常量表达式。
3. 引用是另一个变量的别名。这样的别名可以代替原来的变量名。因为是别名而非指针，所以声明引用时必须指出相应的变量，另外与指针不同，我们不能修改引用，使其表示另一个变量。
4. 函数的返回值是局部变量，当函数结束执行时该变量也不复存在，那么如何返回函数的返回值呢？答案是系统将自动生成返回值的副本，该副本可以在程序中的返回点获得。
5. 同名局部变量会屏蔽全局变量，可以用作用域解析运算符：：来访问全局变量。
6. 当定义多维数组作为形参时，我们同样可以省略第一维的大小。