Chapter 01

输出运算符（<<）输入运算符（>>）返回其左侧运算对象作为其计算结果。

不断读取数据直至没有新的输入为止：

|  |
| --- |
| 1. **int** value = 0; 2. **while** (std::cin >> value) |

当使用一个istream对象作为条件时，其效果是检测流的状态。如果流是有效的，即流未遇到错误，那么检测成功。当遇到文件结束符（end-of-file），或遇到一个无效输入时（例如读入的值不是一个整数），istream对象的状态会变为无效。处于无效状态的istream对象会使条件变为假。

从键盘向程序输入数据时，Windows系统中输入文件结束符的方法是敲Ctrl+Z，然后按Enter或Return键。

Chapter 02

1字节 = 8比特，1个字 = 32或64比特。

在实际应用中，short常常显得太小而long一般和int有一样的大小。如果你的数值超过了int的表示范围，选用long long。

在算术表达式中不要使用char或bool。如果你需要使用一个不大的整数，那么明确指定它的类型是signed char或者unsigned char。

当一个算术表达式中既有无符号数又有int值时，那个int值就会转换成无符号数。

当从无符号数中减去一个值时，不管这个值是不是无符号数，都必须确保结果不能是一个负值。

如果两个字符串字面值位置紧邻且仅由空格、缩进和换行符分隔，则它们实际上时一个整体。

nullptr是指针字面值。

如果使用列表初始化且初始值存在丢失信息的风险，则编译器将报错。

|  |
| --- |
| 1. **long** **double** ld = 3.1415926536; 2. **int** a{ ld };  // 错误 |

如果是内置类型的变量未被显式初始化，1）定义在函数体内部的内置类型变量将不被初始化；2）定义于任何函数体之外的变量被初始化为0。

|  |
| --- |
| 1. **extern** **int** i;  // 声明i而非定义i 2. **int** j;   // 声明并定义j |

变量能且只能被定义一次，但是可以被多次声明。

引用必须被初始化，且引用的类型必须与其所引用对象的类型一致。

一旦初始化完成，引用将和它的初始值对象一直绑定在一起。因为无法令引用重新绑定到另外一个对象。

引用不是对象，没有实际地址。

|  |
| --- |
| 1. **int** ival = 1024; 2. **int** &refVal = ival; |

建议初始化所有的指针。

生成空指针的方法：

|  |
| --- |
| 1. **int**\* p1 = nullptr; 2. **int**\* p2 = 0; 3. **int**\* p3 = NULL; |

任何非0指针对应的条件都是true。

void\*：可用于存放任意对象的地址。但不能直接操作void\*指针所指的对象，因为我们并不知道这个对象到底是什么类型的。

默认情况下，const对象被设定为仅在文件内有效。

若想让const对象只在一个文件中定义const，而在其他多个文件中声明并使用它。解决的方法是：对于const变量不管是声明还是定义都添加extern关键字。

|  |
| --- |
| 1. // file\_1.cpp 2. **extern** **const** **int** bufSize = 512; 4. // file\_1.h 5. **extern** **const** **int** bufSize; |

|  |
| --- |
| 1. **const** **int** ci = 1024; 2. **const** **int**& r1 = ci; 3. **int**& r2 = ci;   // 错误，试图让一个非常量引用指向一个常量对象 |

（例外）在初始化常量引用时允许用任意表达式作为初始值：

|  |
| --- |
| 1. **int** i = 42; 2. **const** **int**& r1 = i; 3. **const** **int**& r2 = 42; 4. **const** **int**& r3 = r1 \* 2; |

常量引用仅对引用可参与的操作做出了限定，对于引用的对象本身是不是一个常量未作限定，所以允许通过其他途径改变它的值。

指向常量的指针不能用于改变其所指对象的值。

|  |
| --- |
| 1. **const** **double** pi = 3.14; 2. **const** **double**\* cptr = pi; |

允许令一个指向常量的指针指向一个非常量对象。

常量指针必须初始化，而且一旦初始化完成，则它的值（也就是存放在指针中的那个地址）就不能再改变了。

|  |
| --- |
| 1. **int** errNumb = 0; 2. **int**\* **const** curErr = &errNumb; |

非常量可以转换成常量，反之则不行。

声明为constexpr的变量一定是一个常量，而且必须用常量表达式初始化。

|  |
| --- |
| 1. constexpr **int** mf = 20; 2. constexpr **int** limit = mf + 1; 3. constexpr **int**\* q = nullptr;  // q是一个指向整数的常量指针 |

|  |
| --- |
| 1. // 类型别名 2. **typedef** **double** wages; 3. **using** DI = **double**; 4. wages hourly, weekly; 5. DI monthly; 6. **typedef** **char**\* pstring; 7. **const** pstring cstr1 = 0;  // 指向char的常量指针 8. **const** **char**\* cstr2 = 0;  // 指向const char的指针 |

auto让编译器通过初始值来推算变量的类型。auto定义的变量必须有初始值。

decltype：选择并返回操作数的数据类型。

|  |
| --- |
| 1. **const** **int** ci = 0, &cj = ci; 2. decltype(ci) x = 0;  // const int类型 3. decltype(cj) y = x;  // const int&类型 4. **int** i = 42; 5. decltype((i)) d = i;  // d是int&类型，必须初始化 |

设置头文件保护符：

|  |
| --- |
| 1. #ifndef SALES\_DATA\_H 2. #define SALES\_DATA\_H 4. **struct** Sales\_data { 6. }; 8. #endif |

Chapter 03

位于头文件的代码，一般来说不应该使用using声明。

string

初始化string对象的方式：

|  |
| --- |
| 1. string s1; 2. string s2(s1); 3. string s3 = s1; 4. string s4("value"); 5. string s5 = "value"; 6. string s6(10, 'c');  // “cccccccccc” |

getline()：从给定的输入流中读入内容，直到遇到换行符为止，然后把所读的内容存入到string对象中去（注意不存换行符）。

|  |
| --- |
| 1. string word; 2. **while** (cin >> word)  // 反复读取，直至到达文件末尾 3. cout << word << endl; 4. string line; 5. **while** (getline(cin, line))  // 每次读取一整行，直至到达文件末尾 6. cout << line << endl; 7. line.empty(); // line为空返回true 8. line.size(); // 返回line中字符的个数，类型为无符号整型数string::size\_type 9. line[n]; // 返回line中第n个字符的引用 |