想了解全部内容请移步至

<http://www.cplusplus.com/>

<http://zh.cppreference.com/w/%E9%A6%96%E9%A1%B5>

<https://www.tutorialspoint.com/c_standard_library/>

<https://msdn.microsoft.com/en-us/library/hh875057.aspx>

<http://pubs.opengroup.org/onlinepubs/9699919799/>

<https://www-s.acm.illinois.edu/webmonkeys/book/c_guide/>

|  |  |
| --- | --- |
| c++98(withdrawn) | <http://www.iso.org/iso/catalogue_detail.htm?csnumber=25845> |
| c++03(withdrawn) | <http://www.iso.org/iso/catalogue_detail.htm?csnumber=38110> |
| TR1 | <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1745.pdf>  <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1836.pdf> |
| c++11(withdrawn) | <http://www.iso.org/iso/catalogue_detail.htm?csnumber=50372> |
| c++14 | <http://www.iso.org/iso/catalogue_detail.htm?csnumber=64029>  <http://download.csdn.net/download/sirenxiaohuayuan/9410281> |

更新地址：<https://github.com/jhcarl0814/notes>

若发现文档内容与实际不符，请向GitHub上的作者反馈。

红色的是正则表达式算子<https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F>

蓝色的单元格是未完成（目前无法完成）的内容。

运算符用法、功能、优先级和结合顺序（！！！运算符执行时至少求1个表达式的值。表达式的求值过程可能有side-effect，例如：调用函数进而变量的值被修改。因此是否去求表达式的值以及求多少次也都需要被关注。）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 编译器自动连接相邻的字符串。 |  |  |
| ::一元作用域运算符 | ::全局变量名 | 访问全局变量。 |  |  |
| ::二元作用域运算符 | :: |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

（**某值**的）引用

|  |  |  |
| --- | --- | --- |
| 声明引用  在某数据类型的变量定义语句中将**变量名**换为&**别名**。  若变量定义语句不是必须初始化的，必须用左值初始化。  若变量定义语句是必须初始化的（（没有\*时有const）或最后1个\*后有const），必须用左值或右值初始化。 | 1、引用的类型是在**某值的数据类型**最后1个\*后（如果没有\*则是第1个[前）插入1个&。  2、**某值的数据类型**不是数组。  3、**初始化表达式**不是引用。  4、若用左值初始化，**初始化表达式**的数据类型中所有有const的位置，引用的数据类型中的所有相应位置都有const。   |  | | --- | | int \* \*const \* a=0;  const int \*const \* \* &b=a;错误  const int \* \*const \* &c=a;正确  const int \* \* \*const&d=a;错误  int \*const \*const \* &e=a;正确  int \*const \* \*const&f=a;错误  int \* \*const \*const&g=a;正确 |   5、可以用typedef给引用的类型创建别名。 |
| 函数的形参是（**某值**的）引用  在函数原型和函数定义中在**形参名**前插入1个&。 | 1、引用的传参过程和返回值过程的规则都与引用的初始化操作的规则相同。 |
| 函数的返回值是（**某值**的）引用  在函数原型和函数定义中在**返回值数据类型名**后插入1个&。 |

函数

|  |
| --- |
| 定义、声明  1、函数名后()或(void)表示不接收实参。  2、在返回值类型前插入inline表示建议内联该函数。除最小的函数外，编译器可能忽略inline。 |
| 带默认实参的形参   |  | | --- | | 定义、声明  1、只在定义或1个声明中指定所有要指定的默认实参。  2、若1个形参指定了默认实参，则它右侧的所有形参都指定默认实参。（若1个形参没有指定默认实参，则它左侧的所有形参都不指定默认实参。） | | 调用  1、若省略1个实参，则它右侧的所有实参都省略。（若1个实参没有省略，则它左侧的所有实参都不省略。） | |
| 函数重载  1、main函数不能重载。  2、任意两个版本的函数的形参类型列表不相同。  3、若调用时因为省略实参导致无法确定使用哪个版本，错误。  4、函数签名：编译器由函数签名区分重载函数。函数签名由函数名、每个形参的数据类型、每个形参是否是引用、（如果是引用）引用的访问限定、函数的访问限定唯一确定。   |  |  | | --- | --- | | void f(const int a){}  void f(int a){} | 错误 | | void f(const int&a){}  void f(int a){} | 正确 | | void f(const int a){}  void f(int&a){} | | void f(const int&a){}  void f(int&a){} | |
| 函数模板   |  | | --- | | 定义、声明  在函数头前插入template<typename **形参名**(,typename**形参名**)\*>。  形式类型形参可用于函数形参数据类型、返回值数据类型、变量数据类型。 | | 调用  **函数名**(<**数据类型名**(,**数据类型名**)\*>)?(**形参名**(,**形参名**)\*)  1、形式类型实参个数可以少于任何1个版本的函数的模板形参个数。 | |

类

|  |  |
| --- | --- |
| 定义  class 类名  {  (访问限定标记|函数成员原型|函数成员定义|数据成员定义|友元函数的函数原型  )\*  }; | |
| 访问限定标记  **访问限定符**:  访问限定符   |  |  |  | | --- | --- | --- | |  | 不能访问 | 能访问 | | public | / | 所有函数 | | protect | 不是当前类的友元函数或某个派生类的友元函数的全局函数  不是当前类的友元函数或某个派生类的友元函数的其它类的函数成员 | 当前类的友元函数、派生类的友元函数  当前类的函数成员、派生类的函数成员 | | private | 不是当前类的友元函数的全局函数  不是当前类的友元函数的其它类的函数成员 | 当前类的友元函数  当前类的函数成员 |   1、类定义体中默认访问限定是private。  2、作用域：从访问限定标记到下一个访问限定标记或类定义体结束。 | |
| 函数成员  1、编译器尝试内联在类定义体内定义的函数成员。  2、在类定义体外定义函数成员，在函数名前插入**类名**::。  3、按函数名分类   |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 构造函数  1、函数名是**类名**，不返回值，不指定返回值类型。  2、对象成员的构造函数在宿主对象的构造函数执行之前执行，顺序为类定义中的声明顺序。  3、成员初始化列表   |  | | --- | | 在(**形参列表**)后插入:**成员名**(**实参列表**)(,**成员名**(**实参列表**))\*。  1、不能初始化static数据成员。  2、const数据成员和引用数据成员必须用成员初始化列表初始化。  3、成员初始化列表在函数体执行之前执行。  4、若对象成员没有初始化，隐式调用默认构造函数（若没有定义或没权限调用则出错）。 |   4、按形参列表分类   |  |  |  | | --- | --- | --- | | 默认构造函数：没有[没有默认实参的形参]。  1、当且仅当没有定义或声明任何构造函数时，编译器定义1个默认构造函数。内联，访问限定是public，内置数据类型的数据成员不赋初值，隐式调用每个类类型的数据成员的默认构造函数（若没有定义或没权限调用则出错）。  2、调用   |  | | --- | | 通过句柄调用。  定义变量时不初始化。  定义变量时用括号初始化，调用默认构造函数。  包含该类对象的类的构造函数没有在初始化列表中初始化该类对象。  包含该类对象的类的构造函数在初始化列表中调用该类的默认构造函数。 | | | | 复制构造函数（拷贝构造函数）：[没有默认实参的形参]个数等于1且其数据类型是(const |volatile )?**当前类名**&。  1、当且仅当没有定义或声明复制构造函数时，编译器定义1个复制构造函数。内联，访问限定是public，内置数据类型的数据成员赋值，隐式调用每个类类型的数据成员的复制构造函数（若没有定义或没权限调用则出错）（在初始化列表中进行），数组成员的每个元素用相应的方法复制。  2、调用   |  | | --- | | 通过句柄调用。  定义变量时用等号初始化为相同类类型的值。  定义变量时用括号初始化，调用复制构造函数。  函数传参且不是引用。  函数返回值且不是引用。（函数返回值时制造的对象是临时的。）  包含该类对象的类的构造函数在初始化列表中调用该类的复制构造函数。  对象数组初始化。  （数据）类型强制转换运算符 | | 1、防用户复制对象：将访问限定设置为private。  2、防用户、函数成员、友元函数复制对象：将访问限定设置为private，声明，不定义。 | | 转换构造函数：[没有默认实参的形参]个数等于1且其数据类型不是(const |volatile )?**当前类名**(&)?。 | | | 其它构造函数  1、不能出现：[没有默认实参的形参]个数等于1且其数据类型是(const |volatile )?**当前类名**。 | | | | 析构函数  1、函数名是~**类名**，访问限定是public，不接收实参，不返回值，不指定返回值类型。（不能重载。）  2、若至少2个对象同时被删除，析构函数调用顺序与构造函数执行顺序相反。  3、exit不执行自动对象的析构函数。abort不执行任何对象的析构函数。 | | 其他函数成员 |   4、访问限定：（非const、）const   |  | | --- | | 定义、声明  在定义和声明时在(**形参列表**)后插入const。  1、构造函数和析构函数的访问限定不设置const。  2、修改对象的函数成员的访问限定不设置const。  3、调用非const函数成员的函数成员的访问限定不设置const。 | | 调用  1、const对象不能调用非const函数成员。  （2、const函数成员不能调用非const函数成员。）  3、若非const对象调用函数成员时匹配至少1个非const函数成员和至少1个const函数成员，先选用非const版本。   |  |  |  | | --- | --- | --- | | class c  {  public:  void f(void)const{}  void f(void){}  };  int main()  {  c a;  ((const c)a).f();  } | class c  {  public:  void f(void)const{}  void f(void){}  };  int main()  {  c a;  ((const c&)a).f();  } | class c  {  public:  void f(void)const{}  void f(void){}  };  int main()  {  c a;  ((const c\*)&a)->f();  } | | 非const版本 | const版本 | | |   5、this指针   |  |  | | --- | --- | | 1、非const函数中this数据类型是**类名**\*const；const函数中this数据类型是const **类名**\*const。 | 1、用this指针判断和处理自赋值。 |   6、存储类型：（非static、）static   |  | | --- | | 1、static函数成员不指定访问限定。  2、static函数成员没有this指针。（可以访问static数据成员，不能访问非static数据成员。）  3、static函数成员不能调用非static函数成员。 | | |
| 数据成员  1、存储类型：（非static、）static   |  | | --- | | 1、没有初始化相当于初始化为0。  2、作用域是类域。  3、可以通过**类名**::**变量名**访问static数据成员；不能通过**类名**::**变量名**访问非static数据成员。  4、static const int数据成员和static const enum数据成员可以在类定义体内（必须是常量表达式）或文件作用域内初始化；其他static数据成员在文件作用域内初始化。 | | |
| 友元函数 |  |
|  |  |
|  |  |

操纵标准流的函数

#include<iostream>

流操纵符

|  |  |  |  |
| --- | --- | --- | --- |
| endl | 输出缓冲区中数据，输出1个新行。 |  |  |
| setfill(**表达式**) |  | 1、**表达式**的数据类型为char。 | 黏性 |
| setw(**表达式**) |  | 1、**表达式**的数据类型为整型。 | 非黏性 |

操纵流的函数

#include<string>

getline