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| c++98(withdrawn) | 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| c++03(withdrawn) | 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| c++11(withdrawn) | 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。）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ::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 | :: |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

（**某数据类型的值**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。 |
| 函数的形参的值是（**某数据类型的值**的）引用  在函数原型和函数定义中在**形参名**前插入1个&。 | 1、若**实参**是右值，**形参的变量定义语句**（没有\*时有const）或最后1个\*后有const。  2、若**实参**是左值，**实参**的数据类型中所有有const的位置，**形参**的数据类型中的所有相应位置都有const。 |
| 函数的返回的值是（**某数据类型的值**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&。 | 1、若**返回的值**是右值，**返回值数据类型**（没有\*时有const）或最后1个\*后有const。  2、若**返回的值**是左值，**返回的值**的数据类型中所有有const的位置，**返回值数据类型**中的所有相应位置都有const。 |
|  |  |

函数

|  |
| --- |
| 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   |  | | --- | | 指定  1、1次性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。） | | 调用函数  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。 |
| 函数模板   |  | | --- | | 定义函数时使用函数模板  在函数头前插入template<typename **形参名**(,typename**形参名**)\*>。  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。 | | 调用函数时使用函数模板  **函数名**(<**数据类型名**(,**数据类型名**)\*>)?(**形参名**(,**形参名**)\*)  1、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 | | |
| 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**::。   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 构造函数  1、函数名是**类名**，不返回值，不指定返回值类型。  2、定义变量时隐式调用构造函数。  类名 变量名;调用默认构造函数  类名 变量名(**表达式**(,**表达式**)\*);   |  |  | | --- | --- | | 默认构造函数：没有[没有默认实参的形参]。  1、当且仅当没有定义任何构造函数时，编译器生成1个默认构造函数。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（若没有或没权限调用则出错）。  2、 |  | | 复制构造函数：[没有默认实参的形参]个数等于1且其数据类型是(const )?**当前类名**(&)?。 |  | | 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 )?**当前类名**(&)?。 |  | | 其它构造函数 |  | | | 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。） | |  | | |
| 数据成员 |  |
|  |  |
|  |  |
|  |  |

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