# C++关键整理

目录

[C++关键整理 1](#_Toc21586)

[一. 库函数/头文件 1](#_Toc27202)

[二. 标识符 3](#_Toc4605)

[三. 关键字 3](#_Toc10970)

[四. 空格 3](#_Toc2569)

[五. 文件路径 3](#_Toc8536)

[六. 转义字符 3](#_Toc26184)

[七. 标识符 4](#_Toc14991)

[八. 基本输入输出 4](#_Toc18785)

[九. 运算符 5](#_Toc23335)

[十. 循环 6](#_Toc1937)

[十一. 数据类型 7](#_Toc18385)

[十二. 类型限定符 8](#_Toc13956)

[十三. 类型转换 8](#_Toc7236)

[十四. 定义（初始化）变量 9](#_Toc21737)

[十五. 运算符等级 9](#_Toc12029)

[十六. 指针 9](#_Toc4011)

[十七. 引用 9](#_Toc28689)

[十八. const 10](#_Toc6651)

[十九. static 11](#_Toc924)

[二十. 函数 11](#_Toc30343)

[二十一. 文件操作 12](#_Toc7310)

[二十二. 宏定义 13](#_Toc17300)

[二十三. 类&对象 13](#_Toc734)

[二十四. 重载 16](#_Toc26847)

[二十五. 多态 16](#_Toc18546)

[二十六. 高级数据类型 17](#_Toc15715)

[二十七. 动态内存 17](#_Toc2628)

[二十八. 命名空间 17](#_Toc11386)

[二十九. 模板 17](#_Toc18299)

[三十. STL 18](#_Toc23805)

[三十一. 预处理 20](#_Toc26268)

[三十二. 信号处理 20](#_Toc12863)

[三十三. 多线程 20](#_Toc258)

[三十四. web编程 20](#_Toc32291)

[三十五. 其他 20](#_Toc3890)

1. **库函数/头文件**
   1. C语言继承

cstdio

cstdlib

cstdarg

cstring

* 1. C++标准库

iostream

* 1. 标准模板库(STL)
     1. 容器 —— 见下
        1. 序列式容器
           1. 向量 —— vecter
           2. 数组 —— array
           3. 双端队列 —— deque
           4. 单向列表 —— queue
           5. 双向列表 —— list
           6. 字符串 —— string
           7. 数组 —— array
           8. 元组 —— tuple
           9. 位集合 —— bitset
        2. 有序关联式容器
           1. 集合 —— set
           2. 映射 —— map
           3. 组合(pair) —— utility
        3. 容器适配器
           1. 栈 —— stack
           2. 队列 —— queue
     2. 算法 —— algorithm
     3. 迭代器 —— 、、
  2. 工具库(Boost)

1. **标识符**

标识变量、函数、类、模块，或任何其他用户自定义项目的名称。

标识符以字母 A-Z 或 a-z 或下划线 \_ 开始，后跟零个或多个字母、下划线和数字（0-9）。

标识符内不允许出现标点字符，比如 @、& 和 %。

1. **关键字**

这些保留字不能作为常量名、变量名或其他标识符名称。

具体名称查阅链接

<https://www.runoob.com/w3cnote/cpp-keyword-intro.html>

1. **空格**

空格用于描述空白符、制表符、换行符和注释

1. **文件路径**

绝对路径：（Z:\\Clion\\date.txt）

相对路径：（/Clion/date.txt）

包含头文件时可使用以上两种方式到指定文件夹搜索

1. **转义字符**

转义字符 意义 ASCII码值（十进制）

\a 响铃（BEL） 007

\b 退格（BS） 008 将当前位置移到前一列

\f 换页（FF） 012 将当前位置移到下页开头

\n 换行（LF） 010 将当前位置移到下行开头

\r 回车（CR） 013 将当前位置移到本行开头

\t 水平制表（HT） 009 跳到下一个TAB位置

\v 垂直制表（VT） 011

\\ 代表一个反斜线字符 092

\' 代表一个单引号字符 039

\" 代表一个双引号字符 034

\？ 代表一个问号 063

\0 空字符（NULL） 000

\ddd 1到3位八进制数所代表的任意字符

\xhh 1到2位十六进制所代表的任意字符

—— d的范围 0~7 、h的范围0~9 a~f( A~F )

1. **标识符**
   1. 用作变量名的一组字符
   2. 由英文字母、下划线开头，中间只能出现英文字母、下划线和数字
   3. 关键字不能用作标识符
2. **基本输入输出**
   1. 输入
      1. cin
         1. 状态条件(condition state)
            1. 标识符

goodbit —— 无错误

eofbit —— 已到达文件尾

failbit —— 非致命的I/O错误，可挽回

badbit —— 致命的I/O错误,无法挽回

* + - * 1. 成员函数（s为流名称）

s.eof()

若流s的eofbit置位，则返回true

s.fail()

若流s的failbit置位，则返回true

s.bad()

若流s的badbit置位，则返回true

s.good()

若流s的goodbit置位，则返回true

s.clear(flags)

清空状态标志位，并将给定的标志位flags置为1，返回void

s.setstate(flags)

根据给定的flags条件状态标志位，将条件状态位置为1，返回void

s.rdstate()

返回流s的当前条件状态，返回值类型为strm::iostate

* + 1. get
    2. getline
  1. 输出
     1. cout

1. **运算符**
   1. 算术运算符
   2. 关系运算符
   3. 逻辑运算符
   4. 位运算符
   5. 赋值运算符
   6. 杂项运算符
      1. sizeof
      2. … ? … : …
      3. ,
      4. (var). / (\*var)->
      5. &
   7. 范围解析运算符 ::
   8. 空间管理
      1. new(new[])
      2. delete(delete[])
2. **循环**
   1. 循环类型
      1. for —— 特定次数

for ( init; condition; increment )

—— 基于范围(C++11)

for ( type &var\_name : array\_name )

* + 1. while
    2. do…while
  1. 循环控制
     1. break
     2. continue
     3. goto —— 语法规则如下（语句演示，可向前跳转）

{

goto label;

..

label: statement;

}

1. **数据类型**
   1. 基础类型
      1. 无类型 —— Void
      2. 空指针 —— （std::nullptr\_t ）
      3. 算术类型
         1. 整数类型 —— int
         2. 布尔类型 —— bool
         3. 字符类型 —— char
         4. 浮点类型 —— float , double

—— 指数：’e’或者‘E’

* + - 1. 枚举类型 —— enum
      2. 指针类型 —— \*(…)
      3. 常量类型 —— const ()
      4. 符号修饰 —— unsigned , signed（默认）
      5. 常量修饰 —— 无符号U(unsigned)，长整数L(long)
      6. 位宽修饰 —— short , long , long long

（注意搭配使用）

—— 定宽整数类型（普通整数类型别名）

* 1. 高级类型
     1. 数组 —— array
     2. 结构体 —— struct
     3. 联合体 —— union
     4. 类 —— class
  2. 迭代器
     1. 概念阐述：
        1. 遍历容器(STL)元素的数据类型
        2. 迭代器也是变量
        3. 相当于容器和操纵容器的算法之间的中介
     2. 包含种类：
        1. 按容器分类(STL)：
           1. vector —— 随机访问
           2. queue —— 不支持迭代器
           3. priority\_queue —— 不支持迭代器
           4. deque —— 随机访问
           5. stack —— 不支持迭代器
           6. list —— 双向访问
           7. set/multiset —— 双向访问
           8. map/multimap —— 双向访问
        2. 按功能分类：
           1. 随机访问
           2. 双向访问
           3. 正向访问
        3. 操作应用：
           1. 正向迭代器定义格式

STL\_type<type\_name>::iterator iter\_name ;

* + - * 1. 反向迭代器定义格式

STL\_type<type\_name>::reverse\_iterator iter\_name ;

* + - * 1. 成员函数( #include<algorithm> )

返回指向首元素的迭代器地址

iter\_name.begin()

返回尾元素后一个（STL容器外）的迭代器地址

iter\_name.end()

使迭代器iter移动 n 个元素（n>0向后，n<0反之）

advance(iter, n)

计算两个迭代器之间的距离（由其实现原理，iter1要在iter2前面）

distance(iter1, iter2)

交换两个迭代器iter1、iter2指向的值

iter\_swap(iter1, iter2)

* + - * 1. 运算符（不同功能类别提供不同操作）：

通用（全部支持）：

解引用1

\*iter

解引用2（结构体、类成员？）

iter->member\_name

移动指向（单个间距）

iter++(++iter)

比较

==, !=

赋值（复合赋值运算亦可）

\*iter = value (+=, -=, \*=, /=, %=)

逆序遍历（双向、随机）：

反向移动（单个间距）

iter--(--iter)

访问（随机）

移动指向（多间距）

iter += i

iter -= i

随机访问

# 返回 p 后面第 i 个元素的迭代器

iter + i

iter - i

解引用3

# 返回 p 后面第 i 个元素的引用

iter[i]

比较

<, <=, >, >=

距离计算

n(int) = iter1 - iter2（iter1,iter2间元素个数减一）

* 1. 数据类型范围
     1. char类型 —— 1字节

有符号范围为-128到127

无符号范围为0到255。

* + 1. unsigned char类型 —— 1字节

范围为0到255。

* + 1. short类型 —— 2字节

有符号范围为-32,768到32,767

无符号范围为0到65,535

* + 1. unsigned short类型 —— 2字节

范围为0到65,535。

* + 1. int类型 —— 通常为4字节

有符号范围为-2,147,483,648到2,147,483,647

无符号范围为0到4,294,967,295

* + 1. unsigned int类型 —— 通常为4字节

范围为0到4,294,967,295。

* + 1. long类型 —— 通常为4字节

有符号范围为-2,147,483,648到2,147,483,647

无符号范围为0到4,294,967,295

* + 1. unsigned long类型 —— 通常为4字节

范围为0到4,294,967,295。

* + 1. long long类型 —— 8字节

有符号范围为-9,223,372,036,854,775,808到9,223,372,036,854,775,807

无符号范围为0到18,446,744,073,709,551,615。

* + 1. unsigned long long类型 —— 8字节

范围为0到18,446,744,073,709,551,615。

* + 1. float类型 —— 4字节，通常具有6位有效数字

范围为1.2E-38到3.4E+38。

* + 1. double类型 —— 8字节，通常具有15位有效数字

范围为2.3E-308到1.7E+308。

1. 类型限定符
   1. Auto —— 自动判定数据类型（存储类）
   2. const —— 定义常量，表示该变量的值不能被修改
   3. volatile —— 该变量的值可能会被程序以外的因素改变（硬件等）
   4. restrict —— 所修饰指针变量的唯一访问所指对象的方式
   5. mutable —— 所修饰类中的成员变量能被修改（const函数）
   6. static —— 定义静态变量，作用域仅限于当前文件或当前函数
   7. register —— 定义寄存器变量（存储在CPU的寄存器中）
2. **类型转换**

——数值提升优先于数值转换

* 1. 数值提升

数值提升过程中，值本身保持不变

小整数类型的纯右值可转换成较大整数类型的纯右值

* 1. 数值转换

数值转换过程中，值可能会发生改变。

* 1. 格式

C风格 —— (dest\_type) source\_type

—— (dest\_type) expression

C++ —— 前往下面查阅

<https://www.runoob.com/cplusplus/cpp-casting-operators.html>

1. **定义（初始化）变量**

全局变量无初始值会被初始化为 0

局部变量需要手动赋初始值

1. **运算符等级**

<https://zh.cppreference.com/w/cpp/language/operator_precedence>

1. **指针**
   1. 定义格式
      1. type (\*p[a])[b]

定义了a个指向集合（集合中有b个元素）的指针

* + 1. type (\*p[a])( arg\_type1 , … )

定义了a个指向函数（函数中有参数如括号所示）的指针

* 1. 指针操作
     1. 空指针 —— NULL , nullptr
     2. 算术运算
     3. 指针VS数组（指针常量）
     4. 指针数组
     5. 多级指针

1. **引用**
   1. 含义解释：引用变量是一个别名（已存在变量的另一个名字）
   2. 对比区别（引用VS指针）
      1. 不存在空引用（引用必须连接到一块合法的内存）
      2. 一旦引用被初始化为一个对象，就不能被指向到另一个对象。
      3. 引用必须在创建时被初始化。
      4. 不存在多级应用
   3. 定义格式

type (&var\_name)[n] = obj\_name ;

#定义一个指向集合（集合中有n个元素，n=1时则省略即单个变量）引用

* 1. 操作应用

1. const
   1. 定义
      1. 关键字
      2. 指定一个语义约束，编译器会强制实施（某值不变）
   2. 应用操作
      1. 普通变量

(const) type (const) var\_name = value ;

#前后二者等价，后者注意在一次定义多变量时的格式应用

* + 1. 指针变量

(const) type (const) var\_name = value ;

#前者限制指针所指内容为常量，地址可变；

#后者限制指针地址（指针指向）为常量，所指内容（值）可变；

* + 1. 函数参数
       1. 普通变量（同’a)’）
       2. 指针变量（同’b)’）
       3. 自定义类型参数

type func\_name( const type agr1, … )

* + 1. 函数返回值

const type func\_name(arg\_list) ;

const type var\_name = func\_name(arg\_list) ;

#返回值不可变，接收变量也必须const修饰

* + 1. 成员函数

type func\_name(arg\_list) const (;)

{ …… }

#在函数参数列表后面加const

* + - 1. const 对象只能访问const 成员函数，非const 的对象无限制
      2. 非const成员函数无限制
      3. const对象调用const成员函数，非const对象默认调用非const的成员函数
      4. const成员函数可以访问所有成员变量，但是只能访问const的成员函数
      5. const成员函数不能修改任何的成员变量，除非变量用mutable修饰。
    1. 类成员函数

同“成员函数”

1. static
   1. \\
2. **函数**
   1. 定义
      1. 若干语句的集合
      2. 也可以将其称作子过程（subroutine）
   2. 要素
      1. 返回类型
      2. 函数名称
      3. 参数列表
      4. 函数主体
   3. 参数传递（形式参数，实际参数）

——形式参数在进入函数时被创建，退出函数时被销毁

* + 1. 传值调用
    2. 指针调用 —— function\_name(type \*)
    3. 引用调用 —— function\_name(type &)
    4. 参数默认

调用函数时若未传递值（留空）则使用默认值，若指定了值，则忽略默认值，使用传递的值。

—— function\_name(type var = val )

* 1. 格式演示

return\_type function\_name( parameter list )

{

body of the function

}

* 1. 匿名函数(lambda)
     1. ?

1. **文件操作**

、、

1. **宏定义**
   1. 标准I/O流(stream)

cin —— 标准输入流

cout —— 标准输出流

cerr —— 标准错误流

clop —— 标准日志流

1. **类&对象**
   1. 定义阐述：
      1. 一种模板，可以用来创建具有相同属性和行为的多个对象
      2. 封装了数据和函数的组合
      3. 用户自定义的数据类型
   2. 要素：
      1. 成员变量（数据）
      2. 成员函数（函数）
      3. 访问修饰符
   3. 定义格式：

class class\_name //定义类

{

access\_specifiers :

type data\_member ;

type func\_member() //成员函数定义&声明方式一

{…}

type func\_member() ; //成员函数声明方式二

friend type func\_name() ; //友元函数声明

friend class class\_name ;//友元类声明

class\_name() //构造函数

{…}

~class\_name() //析构函数

{…}

};

type func\_member() //成员函数定义方式二

{…}

friend type func\_name() //友元函数定义

{…}

class\_name var\_name ; //定义对象

* 1. 操作应用
     1. 访问引用：’.’（访问运算符）

var\_name.data\_menber(func\_member)

* + 1. 成员函数
    2. 访问修饰符（访问继承权限）
       1. 公有成员 —— public
       2. 私有成员 —— private
       3. 受保护成员 —— protected
    3. 构造函数&析构函数
       1. 每次创建类的新对象时执行
       2. 构造函数的名称与类的名称是完全相同的
       3. 不会返回任何类型，也不会返回 void
    4. 拷贝构造函数

?

* + 1. 友元函数（类）
       1. 友元函数（类）不是成员函数
       2. 友元类应在被调用类中声明
       3. 两个定义声明格式与成员函数相同（句首添加friend）
    2. 内联函数
    3. inline
    4. this指针（特殊指针）
       1. 指向当前对象的实例的指针
       2. 对象访问调用自身地址
       3. ’->’ 访问运算符

this->var\_name/this->func\_name

* + 1. 指针结合

’->’ 访问运算符

* + 1. 静态成员
       1. 静态成员只有一个副本
       2. 定义

static type var\_name (= value) ;

* + - 1. 定义时若无人为初始化，则自动数据为零
      2. 可在类外用范围解析运算符’::’再次声明并初始化

type class\_name :: var\_name = value ;

* + 1. 类继承
       1. 概念
          1. 基类（已有）
          2. 派生类
          3. 继承方式
       2. 继承格式
          1. 单继承

class derived\_class: access\_specifier base\_class

* + - * 1. 多继承

class derived\_class: acs\_spcf base1 , acs\_spcf base2 , …

* + - 1. 继承性质

1. **重载**
   1. 函数重载
   2. 运算符重载
      1. operate
      2. 可重载
      3. 不**可重**载
2. **多态**
   1. 概念阐述
      1. virtual
   2. 链接种类
   3. 操作应用
      1. 虚函数
      2. 纯虚函数
3. **高级数据类型**
   1. 数据抽象
      1. 访问标签
   2. 数据封装
      1. 数据隐藏
   3. 数据接口（抽象类）
      1. 实例化
4. **动态内存**
   1. 内存分配

new

* 1. 释放内存

delete

（单元素）{ delete \*point }

（多元素）{ delete [] \*point }

1. **命名空间**
   1. namespace
   2. using
   3. 访问运算符’::’
2. **模板**
   1. 函数模板
      1. 定义格式

template <type\_name T, … >

type func\_name(T (&)agr1 , … )

{ …… }

#注意参数列表中引用和其他类型变量的区别

* 1. 类模板
     1. 定义格式

template <type\_name T, … >

class class\_name

{

access\_specifiers :

type func\_name (T agr1, … ) ;

}

template <type\_name T, … >

T class\_name<T>::func\_name(T agr1, … )

{ …… }

#可类比类成员函数定义格式一

1. STL

###

* 1. 容器 —— 见下
     1. 序列式容器
        1. 向量 —— vecter
        2. 数组 —— array
        3. 双端队列 —— deque
        4. 单向列表 —— forward\_list
        5. 双向列表 —— list
        6. 字符串 —— string
        7. 数组 —— array
        8. 元组 —— tuple
        9. 位集合 —— bitset
     2. 有序关联式容器
        1. 集合 —— set
        2. 多重集合 —— multiset
        3. 映射 —— map
        4. 多重映射 —— multimap
        5. 组合 —— pair
     3. 容器适配器

###

* + - 1. 栈 —— stack
      2. 队列 —— queue
      3. 优先队列 —— priority\_queue
  1. 算法 —— algorithm
     1. 排序

sort

* + 1. 二分查找

lower\_bound

upper\_bound

binary\_search

* 1. 迭代器 —— 、、
  2. 函数传参调用

1. **预处理**
   1. define
   2. # , ##
2. **信号处理**
3. **多线程**
4. web**编程**
5. **其他**

C++ STL库函数总结

<https://wyqz.top/p/870124582.html>