**mingw64：git clone https://git.code.sf.net/p/mingw-w64/mingw-w64**

|  |  |  |  |
| --- | --- | --- | --- |
| **msys2：** | | | |
| **换源** | **~/etc/pacman.d/mirrorlist.mingw64**  **Server = https://mirrors.tuna.tsinghua.edu.cn/msys2/mingw/x86\_64** | | |
| **~/etc/pacman.d/mirrorlist.msys**  **Server = https://mirrors.tuna.tsinghua.edu.cn/msys2/msys/$arch/**  **Server = http://mirrors.ustc.edu.cn/msys2/msys/$arch/**  **Server = http://mirror.bit.edu.cn/msys2/msys/$arch/**  **Server = https://mirrors.sjtug.sjtu.edu.cn/msys2/msys/$arch/** | | |
| **命令** | **pacman -Ss <regular>** | | **搜索软件包** |
| **pacman -Qe** | | **查看用户已安装软件包** |
| **pacman -Syu** | | **更新软件包和系统** |
| **pacman -S mingw-w64-x86\_64-<pack>** | | **安装mingw-w64** |
| **软件包** | **toolchain** | **cpp工具链** |

**基础知识：**

|  |  |  |
| --- | --- | --- |
| **内存分区** | **代码区** | **二进制代码** |
| **全局区** | **全局变量，静态变量 (static)，常量 (const)** |
| **栈区** | **函数参数，局部变量** |
| **堆区** | **自由分配、释放 (new，返回指针)** |

**指针声明：int\* p = &a;**

**变量别名：int& ref = a;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **通用** | **const <Type> <Key> = <Value>;** | **定值** | **const int\* p;** | **常量指针 (指向值不变)** |
| **int\* const p;** | **指针常量 (指向地址不变)** |
| **<Type>\* <Key> = new <Type>(args...);** | **堆区指针** | | |
| **extern <Type> <Key>;** | **声明对其它文件的引用** | | |
| **typename <Type> <Key>;** | **声明含模板的类型，并初始化** | | |
| **thread local <Type> <Key>;** | **线程局部变量** | | |
| **static <Type> <Key>;** | **在.cpp时，限定作用域** | | |
| **成员** | **静态变量** | | |
| **mutable <Type> <Key>;** | **可被常函数修改** | | |

|  |  |  |
| --- | --- | --- |
| **注释** | **// String** | **单行** |
| **/\* String \*/** | **多行** |
| **声明** | **typedef <oldType> <newType>;** | **类型重命名** |
| **内存** | **delete[] <Key>;** | **释放数组 (需要检查指针为空)** |
| **sizeof(Type | Key)** | **返回占用字节** |

|  |  |
| --- | --- |
| **数值操作：** | |
| **abs(x)** | **返回数字绝对值** |
| **atof / atoi / atol(string)** | **字符串 → 数值** |

|  |  |
| --- | --- |
| **分支循环：** | |
| **<Bool> ? <true\_out> : <false\_out>** | **条件赋值，返回变量 (三目运算符)** |
| **while、if、else if、else** | **分支常用语句** |
| **do {cmds} while (Bool) {...}** |
| **switch <Exp> {**  **case <Res1>: ...**  **default: ...**  **}** | **满足激活条件时执行余下所有语句** |
| **for (<init>; <cond>; <step>) {...}** | **变量循环获得序列里的值** |
| **for (auto <Key>: <Seq>) {...}** |
| **break;** | **跳出循环体** |
| **continue;** | **返回循环语句** |

|  |  |  |  |
| --- | --- | --- | --- |
| **异常处理：** | | | |
| **throw <etype>(args...);** | | | **抛出异常** |
| **try {...} catch {const <etype>& e} {...}** | | | **尝试** |
| **错误类型** | **exception** | | **异常基类** |
| **runtime\_error** | | **运行异常** |
| **logic\_error** | | **逻辑错误** |
| **invalid\_augument** | | **参数无效** |
| **out\_of\_range** | | **索引越界** |
| **domain\_error** | | **超出定义域** |
| **length\_error** | | **长度错误** |
| **overflow\_error** | | **数值溢出** |
| **实例方法** | **what()** | **获取异常内容** |

|  |  |  |  |
| --- | --- | --- | --- |
| **转义序列：** | | | |
| **振铃字符\a** | **换行字符\n** | **垂直制表符\v** | **退格字符\b** |
| **水平制表符 (以4个字符为一列)\t** | | | **回车字符\r** |

|  |  |  |  |
| --- | --- | --- | --- |
| **运算优先级：** | | | |
| **作用域解析** | **::** | | |
| **成员访问 (有序)** | **. (直接)，-> (间接)** | | |
| **地址** | **\* (解引用；先左结合)，& (取地址)** | | |
| **位翻转** | **~** | | |
| **乘除** | **\*，/，% (int)** | | |
| **加减** | **+，-** | **b=++a** | **a自增，返回结果** |
| **b=a++** | **返回结果，a自增** |
| **位移动** | **<<，>>** | | |
| **位运算 (有序)** | **&，^，|** | | |
| **比较操作符** | **>，>=，<，<=，==，!=** | | |
| **逻辑运算符** | **!，&&，||** | | |

**# -：改符号位，并取补码**

**数组 (首地址)：<Type> <Key>[] = {x1, x2, ...};**

**类型转换：<Type> (<Key>)**

|  |  |  |  |
| --- | --- | --- | --- |
| **数据类型：** | | | |
| **修饰** | **signed (带符号)** | **unsigned (无符号)** |  |
| **整型** | **int16\_t (short)** | **int32\_t (int)** | **int64\_t (long long)** |
| **浮点** | **float (4 byte)** | **double (8 byte)** |  |
| **字符** | **char (1 byte)** | **wchar\_t (2 byte)** |  |
| **其它** | **bool (1 byte)** | **void (空类型)** | **指针 (8 byte)** |
| **自动** | **auto (需赋值)** | **decltype(exp)** |  |

**函数func：**

**# define <Func>(x, y) (x+y)**

**# define <Func>(x, y) (x=y)**

|  |  |  |  |
| --- | --- | --- | --- |
| **通用** | **inline <rType> <Func>(args...);** | **内联函数，不调用 直接展开，适用于小函数** | |
| **static <rType> <Func>(args...);** | **在.cpp时，限定作用域** | |
| **成员** | **类方法，只能访问静态成员变量** | |
| **<rType> <Func>(args...) const;** | **常函数，只修改mutable修饰的成员变量** | |
| **virtual <rType> <Func>(args...) = 0;** | **声明** | **虚拟函数，可被基类访问** |
| **赋值** | **抽象方法** |
| **<rType> <Func>(args...) override;** | **覆写虚拟函数** | |
| **friend <rType> <Func>(args...);** | **友元函数** | |
| **explicit <Type>(args...);** | **需显式调用构造函数** | |

**命名namespace：**

|  |  |  |
| --- | --- | --- |
| **定义** | **namespace <ns> {...}** | |
| **调用** | **using namespace <ns>;** | **导入所有变量** |
| **using <ns>::<key>;** | **导入某一变量** |

**枚举enum：**

|  |  |
| --- | --- |
| **定义** | **enum <Type> {\*vars}** |
| **创建** | **<Type> <Key> = var** |

**模板template：**

|  |  |
| --- | --- |
| **定义** | **template<typename T, ...>** |
| **函数** | **fun<T>(args...)** |
| **对象** | **无自动类型推导，可有默认参数** |
| **特化** | **template<>**  **class Type<T> {...};** |

**对象class / struct：**

|  |  |  |
| --- | --- | --- |
| **全局** | **class <Type> final;** | **不可被继承** |
| **成员** | **friend class <Type>;** | **友元类** |

|  |  |  |
| --- | --- | --- |
| **初始化** | **隐式转换** | **<Type> <Key> = {args...};** |
| **显式构造** | **<Type> <Key>(args...);** |
| **<Type> <Key> = <Type>(args...);** |
| **成员方法** | **自身的指针常量this** | |
| **访问权限** | **private** | **私有 (class默认)，不可继承，可被友元访问** |
| **protected** | **保护，可继承，可被友元访问** |
| **public** | **公有 (struct默认)** |
| **继承方式** | **class T: private B** | **私有继承，(protected, public) → private** |
| **class T: protected B** | **保护继承，public → protected** |
| **class T: public B** | **公有继承** |
| **class T: virtual B** | **虚继承，成员变量可重载** |

|  |  |  |
| --- | --- | --- |
| **基本方法** | **<Type>(args...): <attr>(value), { } = default;** | **构造函数** |
| **<Type>(const <Type>& <Key>){ } = delete;** | **拷贝函数，默认浅拷贝** |
| **~<Type>(args...){ };** | **析构函数，在此释放堆区数据** |
| **<rType> operator()(args...);** | **仿函数** |
| **friend ostream& operator<<**  **(ostream& os, const <Type>& <Key>);** | **流左移函数** |
| **比较方法** | **bool operator==(<Type>& other);** | **==, !=, <, <=, >, >=** |
| **算术方法** | **<Type> operator+=(<dType>& other);** | **+, -, \*, /, %, <<, >>** |
| **<Type> operator++(int);** | **自增 (x++)，return <new>** |
| **<Type>& operator++();** | **自增 (++x)，return \*this** |
| **<Type>& operator=(<Type> const& other);** | **赋值，return \*this** |
| **键值操作** | **<rType>& operator[](<dType> key);** | **<object>[key]** |

**头文件hpp：**

|  |  |
| --- | --- |
| **宏常量：** | |
| **\_\_cplusplus** | **编译器的C++标准年份** |

|  |  |
| --- | --- |
| **宏定义：** | |
| **#include "header"** | **包含头文件** |
| **#pragma once** | **只包含一次** |
| **#define <Key> <Value>** | **定义宏常量 (只进行字符替换)** |
| **#if / #ifdef / #ifndef / #else / #endif** | **宏定义检查** |

**算法algorithm：**

|  |  |  |
| --- | --- | --- |
| **通用** | **swap(x, y)** | **交换值** |
| **copy(beg\_iter, end\_iter, out\_iter)** | **拷贝** |
| **random\_shuffle(beg\_iter, end\_iter)** | **随机打乱** |
| **有序** | **sort(beg\_iter, end\_iter, key=NULL)** | **原地排序** |
| **merge(beg\_iter1, beg\_iter2, end\_iter1, end\_iter2,**  **out\_iter, op=less<>() )** | **有序合并** |
| **binary\_search(beg\_iter, end\_iter, x)** | **返回二分查找真值** |
| **set\_intersection / set\_union / set\_difference**  **(beg\_iter1, end\_iter1, beg\_iter2, end\_iter2, out\_iter)** | **集合运算** |
| **查找** | **find(beg\_iter, end\_iter, x)** | **返回首个满足条件值的迭代器** |
| **find\_if(beg\_iter, end\_iter, op)** |
| **adjacent\_find(beg\_iter, end\_iter, op=equal\_to<>() )** | **返回首个二元运算为真的迭代器** |
| **统计** | **count(beg\_iter, end\_iter, x)** | **返回元素出现次数** |
| **count\_if(beg\_iter, end\_iter, op)** | **返回满足条件的元素数目** |
| **替换** | **replace(beg\_iter, end\_iter, oldv, newv)** | **替换相等值** |
| **replace\_if(beg\_iter, end\_iter, op, newv)** | **替换满足条件值** |
| **翻转** | **reverse(beg\_iter, end\_iter)** | **原地翻转** |
| **reverse\_copy(beg\_iter, end\_iter, out\_iter)** | **拷贝翻转** |
| **迭代** | **for\_each(beg\_iter, end\_iter, op)** | **执行映射** |
| **transform(beg\_iter, end\_iter, out\_iter, op)** | **一元映射** |
| **transform(beg\_iter1, end\_iter1, beg\_iter2, out\_iter, op)** | **二元映射** |

**数学cmath：**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **数值操作：** | | | | | | | |
| **M\_PI** | **п** | **isnan(x)** | **判断nan** | **isinf(x)** | **判断inf** | **isfinite(x)** | **是否有限** |
| **M\_E** | **e** | **NAN** | **nan** | **INFINITY** | **∞** |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **运算函数：** | | | |
| **sqrt(x)** | **x ^ 0.5** | **pow(x, a)** | **x ^ a** |
| **sin(x) / cos(x) / tan(x)** | **三角函数** | **正运算** | |
| **asin(x) / acos(x) / atan(x)** | **逆运算** | |
| **exp(x) / log(x, base=e)** | **指数运算** | | |
| **ceil(x) / floor(x) / round(x)** | **取整** | | |
| **hypot(x, y)** | **返回向量模** | | |

**复数complex：**

|  |  |
| --- | --- |
| **属性访问：** | |
| **conj(z)** | **对应共轭复数** |
| **abs(z)** | **复数的模** |
| **arg(z)** | **复数相角 (-п, п]** |
| **polar(r, phi)** | **极坐标 → 复数** |

|  |  |  |
| --- | --- | --- |
| **complex<T>(real, imag)** | | **实例化复数** |
| **实例方法** | **real / imag()** | **实部 / 虚部** |

**迭代numeric：**

|  |  |
| --- | --- |
| **fill(beg\_iter, end\_iter, x) / fill\_n(beg\_iter, n, x)** | **填充值** |
| **accumulate(beg\_iter, end\_iter, init, op=plus<>() )** | **返回累积运算结果** |

**时间ctime：**

|  |  |  |
| --- | --- | --- |
| **时间元组\*tm：** | | |
| **tm\_sec** | **%S** | **0~61** |
| **tm\_min** | **%M** | **0~59** |
| **tm\_hour** | **%H** | **0~23** |
| **tm\_mday** | **%d** | **1~31** |
| **tm\_mon** | **%m** | **01~12** |
| **%b** | **Jan ~ Dec** |
| **%B** | **January ~ December** |
| **tm\_year** | **%Y** | **2001** |
| **tm\_wday** | **%a** | **Mon ~ Sun (0 ~ 6)** |
| **%A** | **Monday ~ Sunday** |
| **tm\_yday** | **年中第几天** | **1 ~ 366** |
| **tm\_isdst** | **夏令时真值** | **0,1,-1 (代表夏令时)** |

|  |  |  |  |
| --- | --- | --- | --- |
| **时间转换：** | | | |
| **秒数**  **int64\_t** | **time(seconds=nullptr)** | **获得当前秒数** | |
| **ctime(\*seconds)** | **秒数 → 字符串** | **%a %b %d %H:%M:%S %Y** |
| **gmtime(\*seconds)** | **秒数 → 时间元组** | **格林威治时间** |
| **localtime(\*seconds)** | **当地时间** |
| **时间元组**  **\*tm** | **mktime(\*t\_tuple)** | **时间元组 → 秒数** | |
| **strftime(char[], size, format, \*t\_tuple)** | **时间元组 → 字符串** | |

**计时chrono：**

**using namespace chrono;**

|  |  |  |
| --- | --- | --- |
| **system\_clock** | | **系统时钟** |
| **静态属性** | **time\_point** | **时间点类** |
| **静态方法** | **now()** | **获得当前time\_point** |
| **to\_time\_t(tp)** | **time\_point → int64\_t** |

|  |  |  |
| --- | --- | --- |
| **duration / milliseconds / seconds / minutes / hours** | | **持续时间** |
| **实例方法** | **count()** | **获得持续时间** |

**函数functional：**

|  |  |
| --- | --- |
| **plus / minus / multiplies / divides / modulus / negate<T>()** | **实例化算术仿函数** |
| **equal\_to / not\_equal\_to / greater / greater\_equal / less / less\_equal<T>()** | **实例化比较仿函数** |
| **logical\_not / logical\_and / logical\_or<T>()** | **实例化逻辑仿函数** |

**文件fstream：**

|  |  |  |
| --- | --- | --- |
| **ios** | **默认模式** | **in 只读模式** |
| **写入模式**  **(若文件存在)** | **out 覆盖** |
| **app 追加写入** |
| **特殊模式** | **binary 二进制模式** |

|  |  |  |
| --- | --- | --- |
| **fstream** | | |
| **形参** | **file, mode** | **以写入模式打开文件** |
| **实例方法** | **<< String** | **写入数据** |
| **write(string, size)** |
| **>> String** | **读出单行** |
| **read(string, size)** | **读出定长数据** |
| **is\_open()** | **打开状态** |
| **close()** | **关闭文件** |

**IO流iostream：**

|  |  |  |
| --- | --- | --- |
| **system(cmd)** | | **输入命令，输出响应** |
| **常量** | **endl** | **换行符** |
| **IO** | **cout << String** | **输出字符串** |
| **cerr << String** | **输出报错信息** |
| **cin >> String** | **读取输入** |

**随机random：**

|  |  |
| --- | --- |
| **srand(id)** | **指定随机种子** |

**线程thread：**

|  |  |  |
| --- | --- | --- |
| **this\_thread** | **sleep\_for(chrono::duration)** | **当前线程休眠指定时长** |
| **sleep\_until(chrono::time\_point)** | **当前线程休眠到某时刻** |

|  |  |  |
| --- | --- | --- |
| **thread(func, args...)** | | **实例化线程** |
| **实例方法** | **join()** | **阻塞当前线程** |

**互斥锁mutex：**

|  |  |  |
| --- | --- | --- |
| **mutex()** | | **实例化互斥锁** |
| **实例方法** | **lock()** | **锁定** |
| **unlock()** | **解锁** |

**类型typeinfo：**

|  |  |
| --- | --- |
| **typeid(x).name()** | **类型名称** |

**通用utility：**

|  |  |
| --- | --- |
| **make\_pair(v1, v2)** | **返回二元组** |

**数组vector：**

**长度变化：容量为**

|  |  |  |
| --- | --- | --- |
| **vector<dType>** | | |
| **形参** | **n, x** | **元素拷贝** |
| **beg\_iter, end\_iter** | **迭代器转化** |
| **静态属性** | **iterator** | **迭代器类** |
| **动态属性** | **size()** | **返回元素个数** |
| **front() / back()** | **首尾元素** |
| **begin() / end() / rbegin() / rend()** | **指定位置迭代器** |
| **元素操作** | **assign(beg\_iter, end\_iter) / assign(n, x)** | **赋值** |
| **resize(length, pad\_value)** | **填充 (指定 capacity)**  **截断 (不改变 capacity)** |
| **push\_back(x) / pop\_back()** | **尾部添加/删除元素** |
| **insert(pos\_iter, x) / insert(pos\_iter, n, x)** | **插入元素** |
| **erase(beg\_iter, end\_iter=(beg\_iter+1) )** | **删除元素** |
| **内存操作** | **capacity()** | **返回容量** |
| **shrink\_to\_fit()** | **收缩容量** |
| **reserve(capacity)** | **预留空间，指定容量** |

**双队列deque：**

|  |  |  |
| --- | --- | --- |
| **deque<dType> (类似[vector](#vector))** | | |
| **元素操作** | **push\_front(x) / pop\_front()** | **头部添加/删除元素** |

**双链表list：**

|  |  |  |
| --- | --- | --- |
| **list<dType> (类似[deque](#deque)，无内存操作)** | | |
| **元素操作** | **remove(x)** | **删除相等元素 (all)** |
| **reverse()** | **元素翻转** |
| **sort(key=NULL)** | **原地排序** |

**字典map：**

|  |  |  |
| --- | --- | --- |
| **map / multimap<kType, vType, comp=less<>()>** | | |
| **形参** | **beg\_iter, end\_iter** | **迭代器转化** |
| **静态属性** | **iterator** | **迭代器类** |
| **动态属性** | **size()** | **返回元素个数** |
| **begin() / end() / rbegin() / rend()** | **指定位置迭代器** |
| **元素操作** | **find(k)** | **返回元素所在的迭代器** |
| **count(k)** | **返回元素出现次数** |
| **pair insert( {k, v} ) / insert(beg\_iter, end\_iter)** | **插入元素** |
| **erase(k) / erase(beg\_iter, end\_iter=(beg\_iter+1) )** | **删除元素** |

**队列queue：**

|  |  |  |
| --- | --- | --- |
| **queue<dType>** | | |
| **实例方法** | **size()** | **返回元素个数** |
| **front() / back()** | **首尾元素** |
| **push(x)** | **元素入栈** |
| **pop()** | **元素出栈** |

**集合set：**

**set / multiset<dType, comp=less> (类似[map](#map))**

**栈stack：**

|  |  |  |
| --- | --- | --- |
| **stack<dType>** | | |
| **实例方法** | **size()** | **返回元素个数** |
| **top()** | **栈顶元素** |
| **push(x)** | **元素入栈** |
| **pop()** | **元素出栈** |

**字符串string：**

|  |  |
| --- | --- |
| **getline(istream, string)** | **读出单行 (无数据时返回false)** |
| **to\_string(x)** | **强制类型转换** |

|  |  |  |
| --- | --- | --- |
| **string (类似[vector](#vector))** | | |
| **动态属性** | **substr(pos, n)** | **切片** |
| **元素操作** | **insert(pos, string) / insert(pos, n, char) /**  **insert(pos\_iter, sub\_beg\_iter, sub\_end\_iter)** | **插入元素** |
| **erase(pos, n=this->size() )** | **删除元素** |
| **assign(string, n) / assign(string, pos, n)** | **赋值** |
| **匹配操作** | **find / rfind(sub, pos, n=sub.size())** | **查找子字符串的位置** |
| **replace(pos, n, new)** | **删除指定子串，替换为新子串** |