从C++来到Qt中去，从Qt中回来到C++中去、

Switch中结构里面case不能使用字符串，要么使用if else要么使用枚举解决这种情况。

STL中的push\_back是往list<string>的容器中添加string的数据同理其他类型也一样

Vector中push\_back也是在尾部追加数据，然后要获得最后一条的数据只需要 vector[vector.siez()-1]即可

参数返回里面可以使用判断语句? : 根据不同情况返回不同类型

STL与string使用

其中STL的方法有对string的增删，排序，插入

transform()用法

try与catch用法

try 定义异常

catch捕获异常，并处理， catch的数据类型需要跟throw出来的类型相匹配

但是删除throw的话，什么类型都能扔，有throw限定，那么catch对应的只能是throw扔的类型。

例子：

void CrossFactorInterface::on\_pbnokcr\_clicked()

{

if (d->model)

{

try{

d->model->flushStream(); //save,tr定义异常

}

catch (const char\* e) { //catch捕获异常，并且处理异常，提示保存错误.

SelfDestroyMessageBox box(3 \* 1000, QMessageBox::Warning, tr("save error"), e);

box.exec();

}

}

}

try定义异常，若出现异常，那么catch则能捕获异常，

try若没有出现异常，那么正常执行，catch不执行。

读取二进制文件。

二进制写入是保护文件

容器保存的类型是复杂的数据结构

vector<sql\_data> name;  
sql\_data{  
string:name;  
string:type;  
string:long;  
vector<tree> str;  
}  
tree{  
str:child;  
str:parent;  
str:sister;

}

Map用法

<http://blog.csdn.net/zhoujiaxq/article/details/9786551>

template模板类

模板声明：

template<typename T> void swap(T& t1, T& t2);

模板定义：

template<typename T> void swap(T& t1, T& t2) {  
 T tmpT;  
 tmpT = t1;  
 t1 = t2;  
 t2 = tmpT;  
}

使用模板：

#include <stdio.h>  
#include "method.h"  
int main() {  
 //模板方法   
 int num1 = 1, num2 = 2;  
 swap<int>(num1, num2);  
 printf("num1:%d, num2:%d\n", num1, num2);   
 return 0;  
}

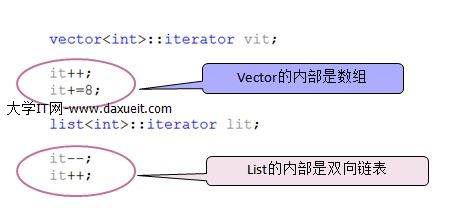
C++ const\_iterator常量迭代器跟iterator 迭代器

容器遍历使用迭代器：【不用迭代器可以使用下标循环去遍历】

<http://www.daxueit.com/article/3101.html>

信息来自上述网站

ni类型是std::vector<name\_index\_t>



std::vector<name\_index\_t>::const\_iterator it = le.ni.begin();

while (it != le.ni.end()) {

lele->SetAttribute(it->first.c\_str(), it->second);

++it;

}

|  |  |  |
| --- | --- | --- |
| 容器 | 支持的迭代器类别 | 说明 |
| vector | 随机访问 | 一种随机访问的数组类型，提供了对数组元素进行快速随机访问以及在序列尾部进行快速的插入和删除操作的功能。可以再需要的时候修改其自身的大小 |
| deque | 随机访问 | 一种随机访问的数组类型，提供了序列两端快速进行插入和删除操作的功能。可以再需要的时候修改其自身的大小 |
| list | 双向 | 一种不支持随机访问的数组类型，插入和删除所花费的时间是固定的，与位置无关。 |
| set | 双向 | 一种随机存取的容器，其关键字和数据元素是同一个值。所有元素都必须具有惟一值。 |
| multiset | 双向 | 一种随机存取的容器，其关键字和数据元素是同一个值。可以包含重复的元素。 |
| map | 双向 | 一种包含成对数值的容器，一个值是实际数据值，另一个是用来寻找数据的关键字。一个特定的关键字只能与一个元素关联。 |
| multimap | 双向 | 一种包含成对数值的容器，一个值是实际数据值，另一个是用来寻找数据的关键字。一个关键字可以与多个数据元素关联。 |
| stack | 不支持 | 适配器容器类型，用vector，deque或list对象创建了一个先进后出容器 |
| queue | 不支持 | 适配器容器类型，用deque或list对象创建了一个先进先出容器 |
| priority\_queue | 不支持 | 适配器容器类型，用vector或deque对象创建了一个排序队列 |

ostringstream是C++的一个字符集操作模板类，定义在sstream.h头文件中

ostringstream类通常用于执行C风格的串流的输出操作，格式化字符串，避免申请大量的缓冲区，替代sprintf

把qmainwindow放到一个dialog里面去然后在放到一个新的qmianwindow，实现成

int sprintf\_s(  
 char \*buffer,  
 size\_t sizeOfBuffer,  
 const char \*format [,  
 argument] ...

);

template <size\_t size>  
int sprintf\_s(  
 char (&buffer)[size],  
 const char \*format [,  
 argument] ...

); // C++ only

int \_snprintf(  
 char \*buffer,  
 size\_t count,  
 const char \*format [,  
 argument] ...

);

int \_snprintf(  
 char (&buffer)[size],  
 size\_t count,  
 const char \*format [,  
 argument] ...

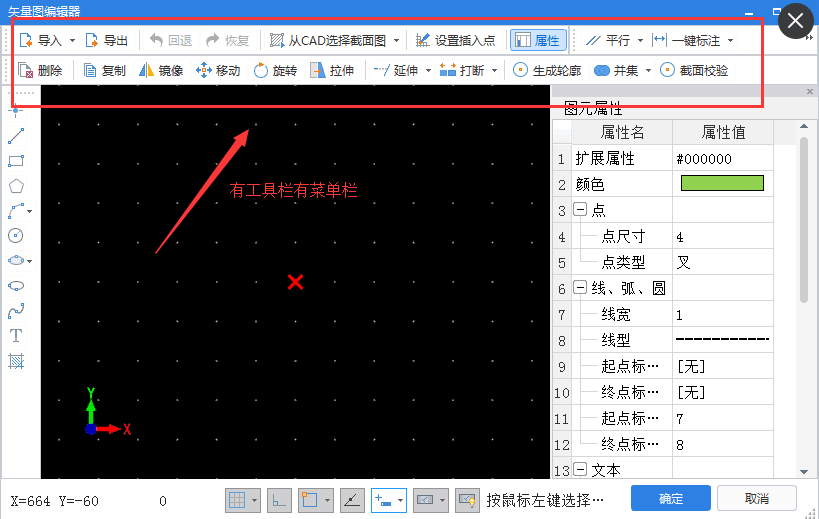
); // C++ only

int \_snprintf\_s(  
 char \*buffer,  
 size\_t sizeOfBuffer,  
 size\_t count,  
 const char \*format [,  
 argument] ...

);

int \_snprintf\_s(  
 char (&buffer)[size],  
 size\_t count,  
 const char \*format [,  
 argument] ...

); // C++ only



itoa函数可以转换任何进制之间的数据

itoa(a,b,2)

a----要转换的数

b----转换以后保存的数

2----为转换的进制

即2可以为4,6,8,10,16进制都可以。

Explicit 不允许隐式调用

Explicit 构造函数是C++用来防止隐式调用

Union 跟struct的区别，

Struct中的成员值改变的时候没有相互影响，并且占用的内存为各个成员所占用的总和，

Union中成员改变的时候，只有改变的值是存在的，其他成员变量不保存，并且内存占用长度是当期联合体中长度最长的成员变量的长度。

std::ostringstream 是C++的一个字符集操作模板类， #include<sstream.h>,通常使用于C风格的串流输出，格式化字符串，避免申请大量的缓冲区，替代sprintf()

派生关系图ios\_base()->ios->ostream->ostringstream；

istringstream类用于执行C++风格的串流的输入操作。   
ostringstream类用于执行C风格的串流的输出操作。   
strstream类同时可以支持C风格的串流的输入输出操作。

sprintf 字符串格式化命令，主要把格式化数据写入某个字符串，sprintf是个变参函数【参数个数可变1~n】，使用sprintf写入buffer的字符数是没有限制的，这就导致buffer存在溢出的可能

ostringstream str;

str<<”s”; //向str插入单个字符

str<<data; //向str插入字符串

结合上面字符串流使用还涉及

std::setw ：需要填充多少个字符,默认填充的字符为' '空格

std::setfill：设置std::setw将填充什么样的字符，如:std::setfill('\*')

std::setbase(n)：将输出数据转换为n进制

std::setprecision()：控制输出流显示浮点数的数字个数，C++默认的流输出数值有效位是6。

Ostringstream buf;

string mydata;

buf<<setw(10)<<setfill(‘0’)<<str;//若str没有10那么长，那么用0字符填充到10长度

对于斜杠不需要转义字符可以写入流/字符串集，然后反斜杠需要转义字符才能使用\\单使用会出错。

C++原始字符串：

C++11语法，

R“data(str)data”

其中原始字符串必须用()括起来，括号前后可以增加其他字符串，所加的字符串会被忽略，并且加的字符串必须在括号两边同时出现。

CString:;format用法

CString str;

str.Format(\_T("%s"), \_T("Hello world!")); //Hello world!

str.Format(\_T("Integer - %d"), 123); //Integer - 123

char temp1 = 'a';

str.Format(\_T("Character - %c"), temp1); //Character - a

float temp2 = 1.234;

str.Format(\_T("Folat - %f"), temp2); //Folat - 1.234

ULONGLONG temp3 = 123456789;

str.Format(\_T("ULONGLONG - %I64u"), temp3); //ULONGLONG - 123456789

SIZE\_T temp4 = 12344;

str.Format(\_T("SIZE\_T - %lu"), temp4);//SIZE\_T - 12344

float temp5 = 1.234567;

str.Format(\_T("Float 3 decimal places - %.3f"), temp5);//Float 3 decimal places - 1.235

str.Format(\_T("Float 3 decimal places, right justified 5 chars - %5.3f"), temp5);//Float 3 decimal places, right justified 5 chars - 00001.234

str.Format(\_T("Float 3 decimal places, left justified 5 chars - %-5.3f"), temp5);//Float 3 decimal places, left justified 5 chars - 1.235

int temp6 = 12;

str.Format(\_T("Int Left aligned 5 character - %.5d"), temp6);//Int Left aligned 5 character - 00012

str.Format(\_T("Int Right aligned 5 character, fill 0 - %05d"), temp6);//Int Right aligned 5 character, fill 0 - 00123

dll(调试的过程中)工程A远程调试的时候可以附加工程B的DLL

C风格的强制类型转换(Type Cast)很简单，不管什么类型的转换统统是：  
TYPE b = (TYPE)a。  
C++风格的类型转换提供了4种类型转换操作符来应对不同场合的应用。

const\_cast，字面上理解就是去const属性。  
static\_cast，命名上理解是静态类型转换。如int转换成char。  
dynamic\_cast，命名上理解是动态类型转换。如子类和父类之间的多态类型转换。  
reinterpret\_cast，仅仅重新解释类型，但没有进行二进制的转换。

Try throw catch的使用：

catch(..)能匹配成功所有的数据类型的异常对象，包括C++语言提 供所有的原生数据类型的异常对象，如int、double，还有char\*、int\*这样的指针类型，另外还有数组类型的异常对象。同时也包括所有自定义 的抽象数据类型

//http://www.runoob.com/cplusplus/cpp-exceptions-handling.html

try{

可能出现异常的代码：

throw data；//(这里的data类型可以为自定义或者常见str,int等)

}

Catch(data) //data类型对应throw抛出的类型

{

}

Catch(…) //…为以上所有不匹配的类型外的异常类型

{

}

#ifndef跟#define的用法，注意一下<标识>这个概念

#ifndef <标识>   
#define <标识>   
......   
......   
#endif   
<标识>在理论上来说可以是自由命名的，但每个头文件的这个“标识”都应该是唯一的。标识的命名规则一般是头文件名全大写，前后加下划线，并把文件名中的“.”也变成下划线，如：[stdio.h](https://www.baidu.com/s?wd=stdio.h&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLmWn4PHm3n1I-rHIbnWn10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWcdPHbvrHm)   
#ifndef \_STDIO\_H\_   
#define \_STDIO\_H\_   
......   
#endif

l = 0x11223344 调用Swap32IfLE(l)之后， l = 0x44332211

广电ctrl+F5 F6 F7 F8对应的 开启 关闭 重启 显示/隐藏 程序功能是使用注册系统热键

也就是将快捷键组合方式写到代码里面，程序起来的时候这些快捷键操作就能对程序起作用，然后就能使用自定义的快捷键；

其实还有另外一种方法，可以使用钩子，在侦听键盘然后获取按键，【这种骚操作一般死在某杀毒软件·····】但是对于这种专用的设备在专门的工控机上是完全不需要钩子的。

广电师傅写的iniwrite.exe机制，用了带参数的main(char argv,char\*argc)，在bat执行的时候讲参数传递给这个iniwirte.exe,然后根据解析接受到的命令，执行对应的操作,详情出门右转google带参数main的用法

程序升级的机制，替换旧的文件【DLL，ini,txt等】