C&CPP Introduction

V1.0

2022/08/29

修订历史（Revision History）

日期	版本	修改	作者	Reviewer
2022/08/29	Version 1.0	初稿	Tianyu Ao	

目 录

表格目录

未找到图形项目表。

图表目录

未找到图形项目表。

词汇表

缩写	含义
BT	Bluetooth
FW	Firmware

1 C&CPP

1.1 C&CPP 简介

C&CPP 可按照以下几部分进行简单划分：

1. 基础语法：

   数据类型、运算符、作用域（命名空间（namespace））、语言结构（顺序、选择、循环）、结构体、共用体、数组、引用、指针（智能指针、函数指针、其他指针）、函数（自定义函数、回调函数 callback）、文件操作。

2. 类与面向对象：

   封装（类（构造函数、拷贝构造函数、析构函数））、继承（单继承、多继承）、多态（函数重载、运算符重载、虚函数、纯虚函数、派生）。

3. 常用数据结构及算法（标准模板库（Standard Template Library，STL））：

   算法、容器、迭代器。

4. 泛化数据类型：

   模板。

5. 其他特性：

   异常处理及其他 CPP 最新特性。

1.2 C&CPP 实例

STL 算法常用头文件	STL 数据结构常用头文件
#include <algorithm> #include <functional> #include <iterator> #include <memory> #include <numeric> #include <utility>	#include <vector> #include <list> #include <deque> #include <map> #include <queue> #include <set> #include <stack>
STL 部分简单了解，代码中用到的时候在去具体看。	

```
class BT
{
public:
BT() {}//构造函数
BT(int para){inqiry_parameter = para;} //构造函数，函数重载
BT(const BT &obj){ inqiry_parameter = obj.inqiry_parameter;
page_parameter = obj.page_parameter;
a2dp_parameter = obj.a2dp_parameter;} //拷贝构造函数
~BT() {} //析构函数
void inquiry(int inqiry_para) {inqiry_parameter=inqiry_para;}//自定义函数及实现
void page(string page_para){this->page_parameter=page_para;}
void set_adv_parameter (int &adv_para) {adv_parameter=adv_para;}
void a2dp(uint16_t a2dp_para); //自定义函数声明
int get_inqiry_parameter();
virtual string set_advv_parameter() = 0;//纯虚函数
protected://有时会用
private:
int inqiry_parameter;
string page_parameter;
uint16_t a2dp_parameter;
int adv_parameter;
};
void BT::a2dp(uint16_t a2dp_para) {a2dp_parameter = a2dp_para;} //自定义函数实现
int BT::get_inqiry_parameter() {return inqiry_parameter;}
```

1. 了解 public、protected、private 作用及在类继承时的访问权限。
2. 构造函数是与类名相同的函数，可以省略不写，系统会自动生成。但如果写了构造函数，定义对象时必须定义符合构造函数参数的对象。定义该类对象时会自动调用构造函数。
3. 同名函数为函数重载。
4. 拷贝构造函数和析构函数简单了解，拷贝构造函数、析构函数系统也会自动生成。
5. 自定义函数可以在类内实现也可在类外实现。
6. 有纯虚函数的类为抽象类。抽象类无法直接定义对象，需要有其他类继承该抽象类，并对纯虚函数进行实现，之后继承的类可正常定义对象。抽象类一般可作为函数接口，类似 java interface。

```
class BT1:public BT
{
public:
void advertising(){}
string set_advv_parameter() {set_adv_parameter(new_adv_parameter);}
private:
int new_adv_parameter;
};
```

BT1 以 public 方式继承 BT。BT1 中有自己定义的变量与函数，同时实现了 BT 中纯虚函数 set_advv_parameter()。BT1 类可正常定义对象，同时 BT1 类对象也可直接调用 BT 中 public 下变量或函数。

类多继承、运算符重载等概念可之后遇到在学习。

```
template <class T>
class BT2 {
private:
vector<T> elems;
public:
void push(T const& elem) {elems.push_back(elem);}
T pop(){return elems.pop_back();}
bool empty() const{ return elems.empty();}
};
```

模板（template）类似于定义一个通用数据类型。在定义模板类对象时，该对象可使用任意数据类型替换通用类型，例如 int，string，BT1 等。除了类模板，还有函数模板。

```
typedef struct BT3
{
    int inqiry_parameter;
    string page_parameter;
    uint16_t a2dp_parameter;
    int (*enable)();
    int (*create_bond)(const string bd_addr, int transport);
}BT3;
```

与类不同，结构体（struct）中只能定义变量，不能直接定义函数。然而可以通过定义函数指针变量，实现类似于类中定义函数的功能。之后在结构体外部对函数进行实现。

```
static int enable() {
    printf("bluetooth enable");
    return true;
}
static int create_bond(const string bd_addr, int transport) {
        string addr = bd_addr;
        int tp = transport;
        printf("%s,%d",addr,tp);
}
int main()
{
        BT3 BT3Interface = { 5, "5", 5, 5, enable, create_bond};//结构体变量初始化。
        BT2<int> int_bt;
        //定义类对象时会自动调用该类构造函数，类中没有显式定义构造函数时，调用系统默认分配的构造函数。
        BT2<string> string_bt;//类模板。
        BT1 bt1;
        //可以观察父类构造函数是否会被调用，以及该定义是否正确。
        bt1.advertising();//调用自己类定义的函数。
        bt1.inquiry();//调用父类 public 函数。
        bt1.set_advv_parameter();//调用实现的纯虚函数。
        BT1 *pbt1 = &bt1;//pbt1 为指针。
        BT1 &abt1 = bt1;//abt1 为引用（引用通常使用于运算符重载）。
        BT bt;//BT 为抽象类，不能定义对象，该定义错误。
        return 0;
}
```