C++

- 基础
  - 函数
    - 参数
      - 传值

        传值是将实参的副本传递给被调用函数的形参，这种传递是单向的。
      - 传地址

        传地址是将指向变量的指针作为函数参数，形参是变量指针，实参是变量的地址值。参数传递的不是变量的值，而是变量的地址，所以当在函数中改变形参的值时，改变的就是原来实参的值。
      - 默认参数

        函数标识符 返回类型 函数名（参数=默认值）
    - 返回值
      - 返回引用

        将函数说明为返回一个引用的主要目的是为了将该函数用在赋值运算符的左边。类型标识符 & 函数名（）{参数列表}
      - 返回指针

        函数返回数据的地址，称为指针函数。类型标识符 * 函数名（参数列表）{}
      - 返回对象
    - 内联函数

      通过inline关键字来定义内联函数，放在函数定义类型之前。具有循环语句，switch语句的函数不能说明为内联函数，其余函数皆可。编辑器执行到该函数的调用语句时不产生实际函数调用，而是用该函数体替换调用表达式。使用内联函数能减少调用开销，加快程序执行速度。如果执行的代码太多，会增加程序代码的大小。
    - 重载
    - 函数模板

      函数模板的定义：template <class T> 或者 template <typename T>类型名 函数名（参数列表）{}template <class T>T max(T t1,T t2){ return (t1>t2) ? t1 : t2;}
    - 友元函数

      友元函数不是当前的成员函数，而是独立于当前类的外部函数，但他可以访问该类的成员。友元函数要在类定义里声明，声明时在函数名前加上关键字friend，该声明既可以放在公有部分，也可以放在私有部分。
    - 指针函数

      声明形式：
      返回类型(指向类 :: * 指针 参数类型列表);
    - 输入输出函数

- 数据的简单输入输出格式
  - 操作符
    - dec
      - 设置转换基数为十进制
    - oct
      - 设置转换技术为八进制
    - hex
      - 设置转换技术为十六进制
    - endl
      - 设置一个换行符并刷新流
    - resetiosflags(long flag)
      - 清除flag指定的标志位
    - setiosflags(long flag)
      - 设置flag指定的标志位
    - setfill(char ch)
      - 设置ch为填充字符串
    - setprecision(int n)
      - 设置浮点数输出精度为n
    - setw(int width)
      - 设置输出数据字段宽度为width
  - 常量
    - ios_base::left
      - 输出数据按输出域左边对齐输出
    - ios_base::right
      - 输出数据按输出域右边对齐输出
    - ios_base::showpoint
      - 浮点输出时必须待有一个小数点
    - ios_base::showpos
      - 在正整数前添加一个"+"号
    - ios_base::scientific
      - 使用科学计数法表示浮点数
    - ios_base::fixed
      - 使用定点形式表示浮点数

- 类
  - 类的定义
    - 说明部分

      说明类中包含的数据成员和成员函数。
    - 实现部分

      实现部分是对成员函数的定义。
  - 对象
    - 生命周期
      - 实例化
        - 调用无参构造函数
        - 调用有参构造函数
      - 销毁
        - 调用析构函数

          用来完成对象被删除前的一些清理工作。定义：~类名(){}
      - 对象的创建与销毁顺序
        - 1.调用父类的构造函数
          - 多继承，则按自左向右的顺序
        - 2.调用成员类的构造函数
          - 多类成员，则按自上向下的顺序
        - 3.调用当前类的构造函数
        - 4.调用当前类的析构函数
        - 5.调用成员类的析构函数
        - 6.调用父类析构函数
    - this指针
    - const对象

      在类中使用const关键字可以修饰数据成员和成员函数或对象。分别称为常数据成员，常成员函数和常对象。在一个对象被说明为常对象后，通过这个常对象只能调用它的常成员函数，而不能调用其他的成员函数。
      - 常成员
        - 常数据成员
        - 常引用
        - 静态常数据成员
  - 三大特性
    - 封装

- 使用private修饰数据，提供接口，易于维护跟提高安全性

- 继承
  - 单继承
  - 多继承
  - 访问权限
    - private
      - 保护成员在派生中不可访问
    - protected
      - 保护成员在派生中也是保护的
    - public
      - 公有类成员在派生中也是公有的
  - “is-a”和"has-a"
    - is-a
      - 继承和派生关系
    - has-a
      - 一个类使用另一个类
  - 二义性
    - 多重继承的情况下，对某个成员的访问出现了不唯一的情况，这种访问具有二义性
      - 可以使用作用域分辨运算符"::"，解决二义性的问题

- 多态
  - 联编
    - 静态联编（编译时的多态性）
      - 通过函数重载和模板实现

        利用函数重载机制，在调用同名的函数时，编译系统可根据实参的具体情况确定调用的是同名函数的哪一个。
        1.利用函数模板，编译系统可根据模板实参以及模板实参的具体情况确定所要调用的函数，并生成相应的函数实例。
        2.利用类模板，编译系统可根据模板实参的具体情况确定所要定义的是哪个类的对象，并生成相应的类实例。
        由于有关操作所针对的具体模板的确定都是在编译时完成的，与运行的动态环境无关，因此得名“编译时的多态性”，其实现机制称为静态联编。
    - 动态联编（运行时的多态性）
      - 通过虚函数实现
  - 虚函数
    - 在非静态成员函数前添加virtual关键字

- 实现多态的条件
  - 1.类之间的继承关系满足赋值兼容性规则
  - 2.改写了同名的虚函数
  - 3.根据赋值兼容性规则使用指针或者引用
- 构造函数与析构函数
  - 构造函数不得声明为虚函数
  - 析构函数可以通过virtual修饰为虚函数

    一个类只能有一个虚析构函数。
    只要积累的析构函数被声明为虚函数，那么由它派生的所有派生类的析构函数，无论是否使用virtual进行修饰，都自动的成为虚函数。
- 纯虚函数
  - virtual 函数类型 函数名(参数列表) = 0;

- 类模板
  - 声明

    ```
    template <typename TYPE> //类模板声明
    class c{
        TYPE a,b;
    }
    ```
  - 继承和派生

- 运算符重载
  - 运算符重载的实质就是函数重载
    - 重载为类的成员函数
      - 语法

        ```
        函数类型 operator 运算符(形参列表)
        {
            函数体
        }
        ```
    - 重载为类的友元函数
      - 语法

        ```
        friend 函数类型 operator 运算符(形参列表)
        {
            函数体
        }
        ```
- 流

- istream
- iostream
- streambuf
- ostream

- ios