20220616-C++

1.学习内容

1.1 C++

模板

模板并非实实在在的函数或类，而仅仅是一个函数或类的描述，是参数化的函数和类。

模板可以分为函数模板和类模板。

函数模板

```cpp
template<class T1,class T2,...>返回类型 函数名(参数表){...}
/*
class表示其后的参数可以是任意类型。模板参数常称为类型参数，在模板实例化时需要传递的实参是一种数据类型，如int和double等。
在定义模板时，不允许template语句和函数模板定义之间有其它任何语句；
函数模板可以有多个类型参数，但每个类型参数都必须用class或typename限定。此外，模板参数中还可以出现确定类型参数，称为非类型参数。
*/
template<class T1,class T2,class T3,int T4>
T1 fx(T1 a,T2 b,T3 c){...}
/*
在传递实参时，T4只能使用常量。
这里的class表示T是一个类型参数，可以是任何数据类型，如int，或者用户定义的struct，enum，class等。为了区别于类的声明关键字class，C++鼓励使用typename，但也支持使用class
*/
```

实例化的方式

```cpp
template<typename T>
T max(T,T);
隐式实例化
int i=max(1,2);
float f=max(1.0,2.0);
显式实例化
int i=max<int>(1,2);
```

类模板

```cpp
template<>返回类型 类模板名<特化的数据类型>::特化成员函数名(参数表){...}
```

C++ | 复制代码

```cpp
template<>
void Array<char*>::Sort()
{
    for(int i=0;i<Size-1;i++)
    {
        int p=i;
        for(int j=i+1;j<Size;j++)
            if(strcmp(a[p],a[j])<0)
                p=j;
        char* t=a[p];
        a[p]=a[i];
        a[i]=t;
    }
}
```

2.