第一讲

课程设置

- 秋季 C++
  - w15 笔试
- 小学期 python&实践
  - 两周选一周参加
  - 两个大作业 多选2
  - 24/9/？？提交

成绩评定 `总成绩100% = 平时作业30% + 期末笔试40% + 暑假小学期大作业30%` **重点** 类与对象、多态性与虚函数

从C语言到C++：面向对象的程序设计方法

数据抽象 把数据和操作分开；模板（template）：少量代码是相对各类数据的兼容 生成严格控制存取权限的“黑盒子”：object

面向对象的程序设计方法

面向过程 程序=算法+数据结构，函数与数据分离，可以任意互相存取调用，以main函数为重心，通过函数间调用串成程序 *问题* 重用性差，数据安全弱 **面向对象 对象**：一个具象的事物 程序可以通过对象调用函数进行操作 对象通过**类**`class`封装，类中定义*数据*和*函数* 面向对象的编程分为两步：

1. **类的设计和封装**：决定类的属性和可执行的操作
2. **函数调用**：实现对象之间的交互

设计方法：程序=对象+消息（交互） 对象有*属性（数据）和方法（函数）*

继承和派生

利用不同类的共性提高代码的重用性（后面会介绍）

C++的几个基本问题

数据类型

增加了布尔类型`bool`,取值只有`true`和`false`

输入输出

利用`cin`和`cout`两个流类对象进行输入输出

```
#include<iostream>
using namespace std;
int main(){
    int i;
    float f;
```

```
    cin>>i>>f;
    cout<<i<<f<<endl;
    return 0;
}
```

cin可以直接读取空格分隔的字符串（不读空格）

```
#include<iostream>
using namespace std;
int main(){
    char s1[10],s2[10];
    cin>>s1>>s2;
    cout<<s1<<s2<<endl;
    return 0;
}
```

cin.get(char* s1,int N,结束符):一次连续读入多个（包括空格字符），直到读满N-1个或遇到结束符，默认"\n",不读也不存结束符 cin.getline(char* s2,int N,结束符)：会读入结束符但不存储

内联函数

调用函数的过程：

1. 断点，压栈
2. 转移执行权
3. 出栈，断点恢复 内联：与#define类似，inline会把主程序内对应代码进行替换，执行时不需要压栈出栈操作，提高效率 demo：

```
inline int add(int a, int b){
    return a+b;
}
```

与#define区别：会自动带括号、适用于各种类型 可以定义变量、使用简单循环语句、分支语句等，但**不能直接递归和间接递归**和其他不能把函数调用*在线化*的情形 **inline只是给编译器的一种建议**，他不一定会执行

动态内存分配：new&delete

分配内存单元并赋值：int *p = new int(10); 释放：delete p; **数组** 分配int *pt = new int[10]; 释放delete []pt; 二维分配int (*pt)[10]= new int[8][10]; 释放delete []pt; **指针数组** int **p = new int*[10]; delete [p]

指针/地址传递

C语言中只能通过显式的指针方式传递地址 C++提供了新的方式：在函数头中加上&即可

```
#include<iostream>
using namespace std;
void swap(int &a, int &b){//交换a和b的值
    int tmp = b;
    b=a;
    a=tmp;
    return;
}
int main(){
    int a=5,b=3;
    swap(a,b);
    cout<<a<<b<<endl;
    return 0;
}
```

引用（reference） 对变量取别名

```
int i,j;
int &r = i;//r是i的别名，公用一个内存单元
int *p = &i;
i = 10;
j = r;
*p = 20;
```

意义：简化书写、提高效率（引用对象的时候）

类型修饰符const

`const int a = 1` 经常用于函数参数说明，防止函数中无意破坏了参数的值/所指内容 常指针：`int* const p` 指向常对象的指着：`const int* p` 指向常对象的常指针：`const int* const p`

作用域与可见性

`::`全局作用域符/类作用域符

缺省参数

可以在声明函数的时候为一个或多个参数指定缺省值，调用时可以从右边开始省略一个或多个参数