C++

Programming

# 类和对象 I
# Classes and Objects I

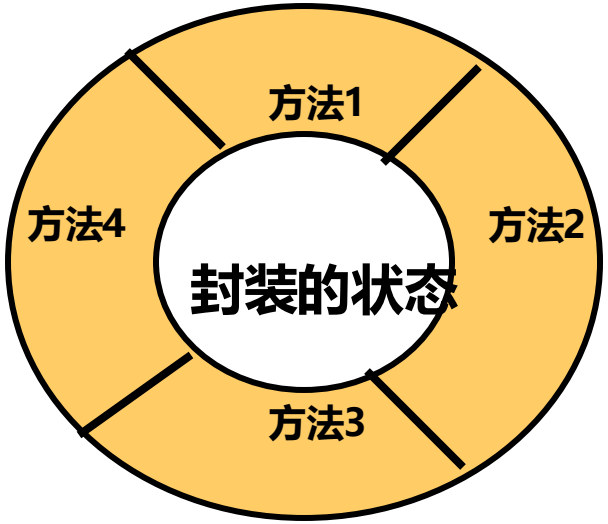
2025年3月3日

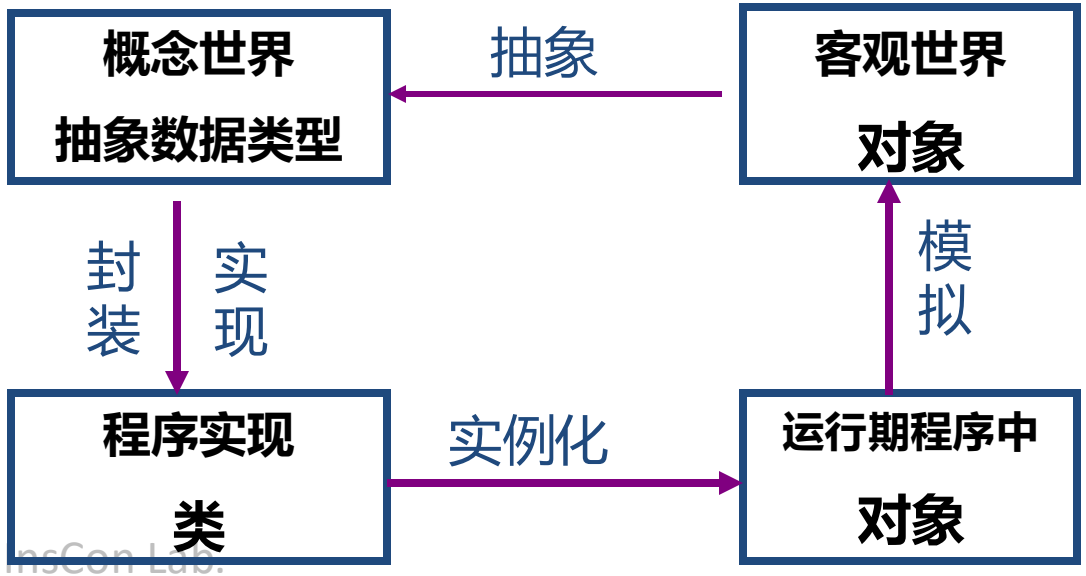
学而不厌 诲人不倦

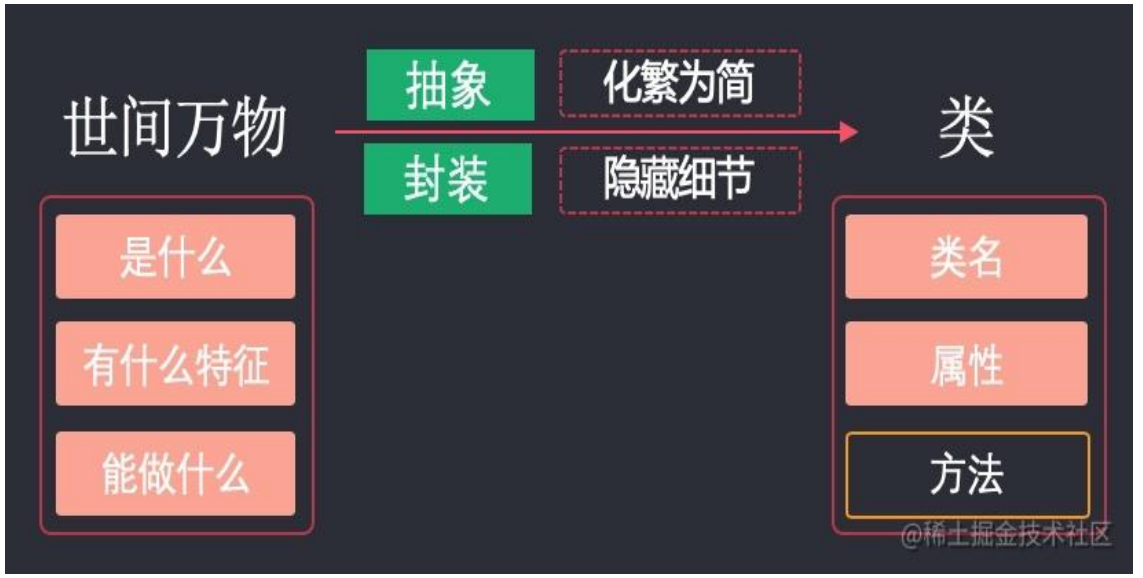
# Chapter 2 类和对象的特性

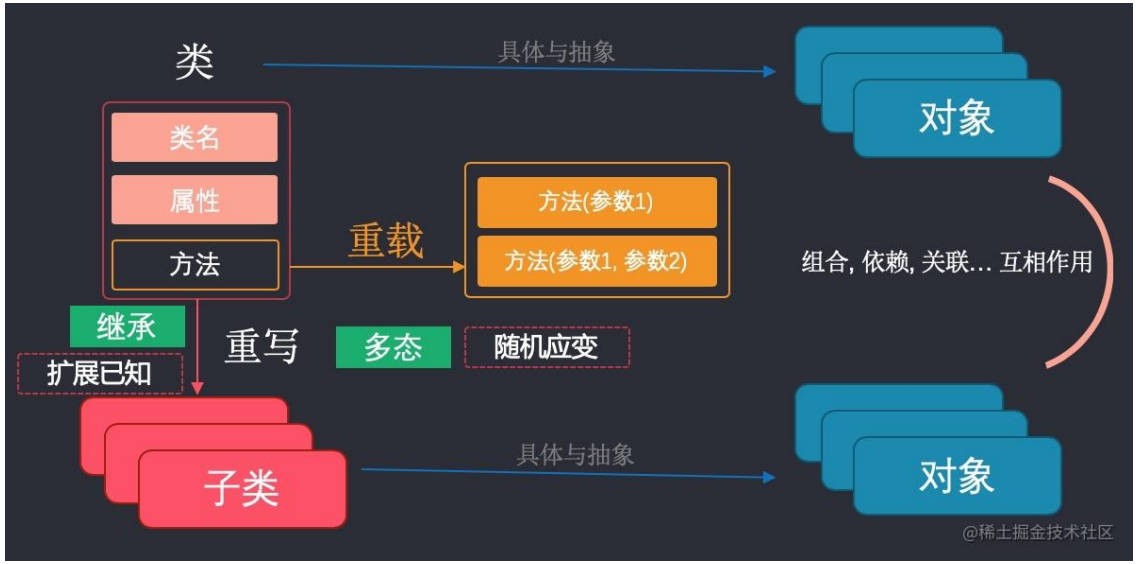
# 2.1 面向对象程序设计方法

## 以对象为中心的设计



# 2.2 类的声明和对象的定义

- **类** **Class**: A programmer defined custom type. An **abstraction** of an object or data type.

```
//student.h
class Student {
    public:
    std::string getName();
    void setName(string
    name);
    int getAge();
    void setAge(int age);

    private:
    std::string name;
    std::string state;
    int age;
};
```

```
//student.cpp
#include student.h
std::string
Student::getName(){
//implementation here!
}
void Student::setName(){
}
int Student::getAge(){
}
 void Student::setAge(int
 age){
}
```

```
//main.cpp
#include student.h
int main(){
    Student sarah;
    sarah.setName("Sarah");
    sarah.setAge(21);
    sarah.setState("CA");
}
```

Classes provide their users with a **public interface** and separate this from a **private implementation**

# 2.2 类的声明和对象的定义

## 1. 类定义格式

class是保留字，声明类类型

```
class    类名        ← 按标识符取名
{
private:
  //私有成员声明
 public:
  //公有成员声明
 protected:
  //保护成员声明
};                   ← 必须加分号
```

private、public、protected 是保留字，成员访问限定符，其后必须跟冒号。

### 成员访问限定符

- **private**：只能被本类中的成员函数访问，类外（除友元外）不能访问。
- **public**：公有成员可以被本类的成员函数访问，也能在类的作用域范围内的其他函数访问。
- **protected**：受保护成员可由本类的成员函数访问，也能由派生类的成员函数访问。

用class声明的类如果不带成员访问限定符，所有成员默认限定为private

# 2.2 类的声明和对象的定义

## 2. 类与结构体定义方式比较

```
struct  student
{     int    num;
      char  name[20];
};

student  st1,st2;
```

```
Class    Student
{          int   num;
           string  name;

           void  setdata()
           {
                cin >> num;
                cin >> name;
           }
           void display()
           {
                cout<< num<<endl;
                cout<< name<<endl;
           }
};

Student   st1,st2;
```

## 3. 定义对象的方法、对象访问成员的方法

```
//先声明类类型，再像定义变量一样定义对象
class student  st1, st2;
student    st1, st2;
//在声明类类型的同时定义对象
class 类名
{
private:
        ...
public:
        ...
} 对象名表;
```

```
//1. 用对象名和成员运算符访问成员
st1.display();   //  调用成员函数
//2.用指向对象的指针访问成员
Time  t, *p;
p= &t;
//3.用对象的引用访问成员
Time  t1;
Time & t2=t1;
```

在定义对象后，编译程序在编译时会为对象分配内存空间，存放对象的成员。

# 2.3 类的成员函数

## 1. 类成员函数的定义、声明和调用

**类成员函数可以访问本类中的所有成员。**
**对象可以通过public类成员函数访问类的其他成员；**
**对象不能通过private或protected访问类的其他成员。**

```
#include <iostream>
using namespace std;

class CStudent //定义CStudent类
{
private:
int num;//定义成员变量：学号
string name;//定义成员变量：姓名
public:
void Setdata();//声明public类成员函数
void Showdata();//声明public类成员函数
};
```

```
void CStudent::Setdata()
{
cout<<"input num:";
cin>>num;
cout<<"input name:";
cin>>name;
}
```

```
int main()
{
CStudent stu;//定义对象stu
//通过stu对象访问CStudent类的public成员函数
stu.Setdata();
//通过stu对象访问CStudent类的public成员函数
stu.Showdata();
return 0;
}
```

## 2. 类外定义成员函数

```
#include <iostream>
using namespace std;

class CStudent //定义CStudent类
{
private:
int num;//定义成员变量：学号
string name;//定义成员变量：姓名
public:
void Setdata();//声明public类成员函数
void Showdata();//声明public类成员函数
};
```

返回值类型　　:: 作用域限定符

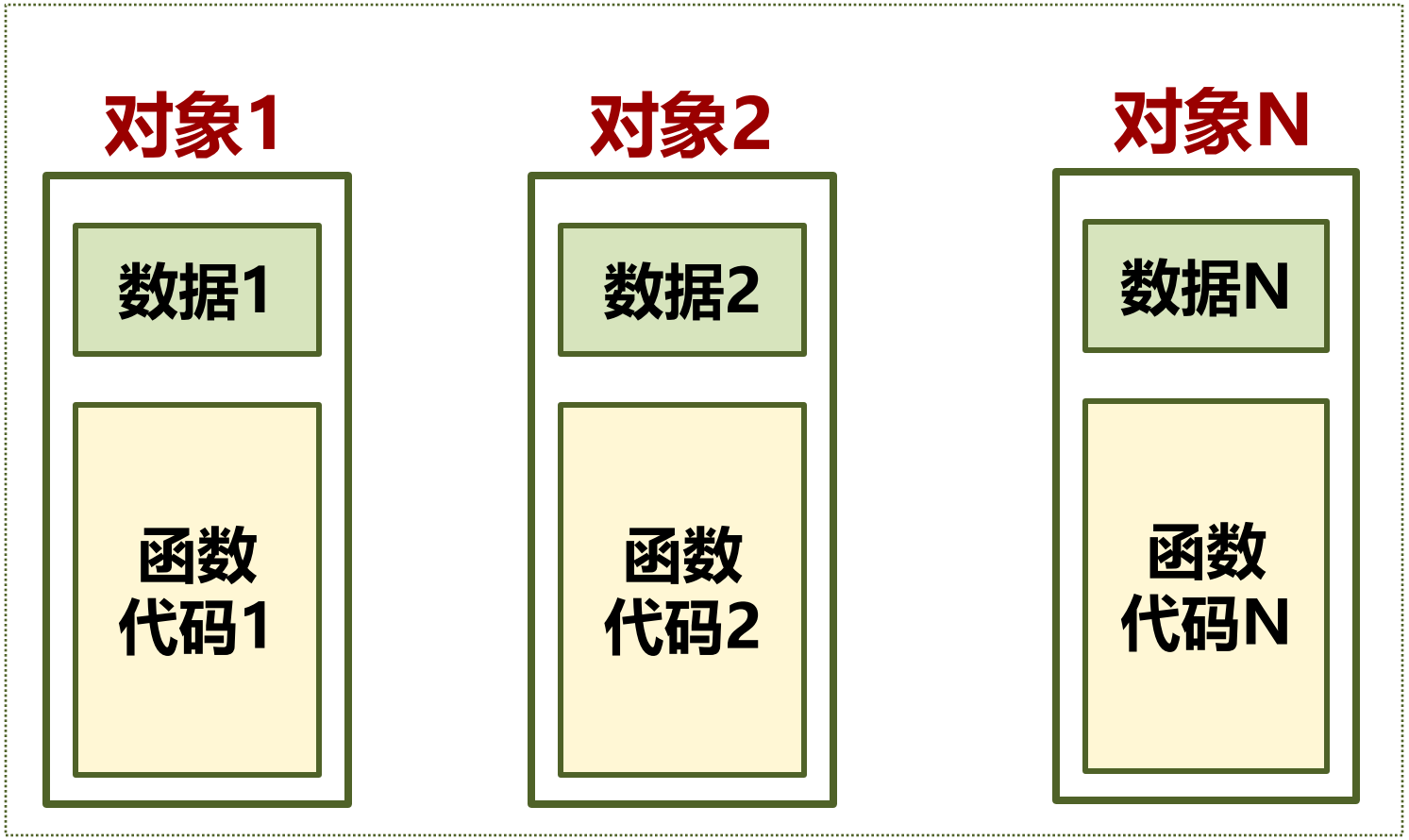
```
void CStudent::Setdata()
{
  cout<<"input num:";
  cin>>num;
  cout<<"input name:";
  cin>>name;
}
```

**注： 函数名前既无类名又无作用域运算符::，表示该函数是全局函数。**

# 2.3 类的成员函数

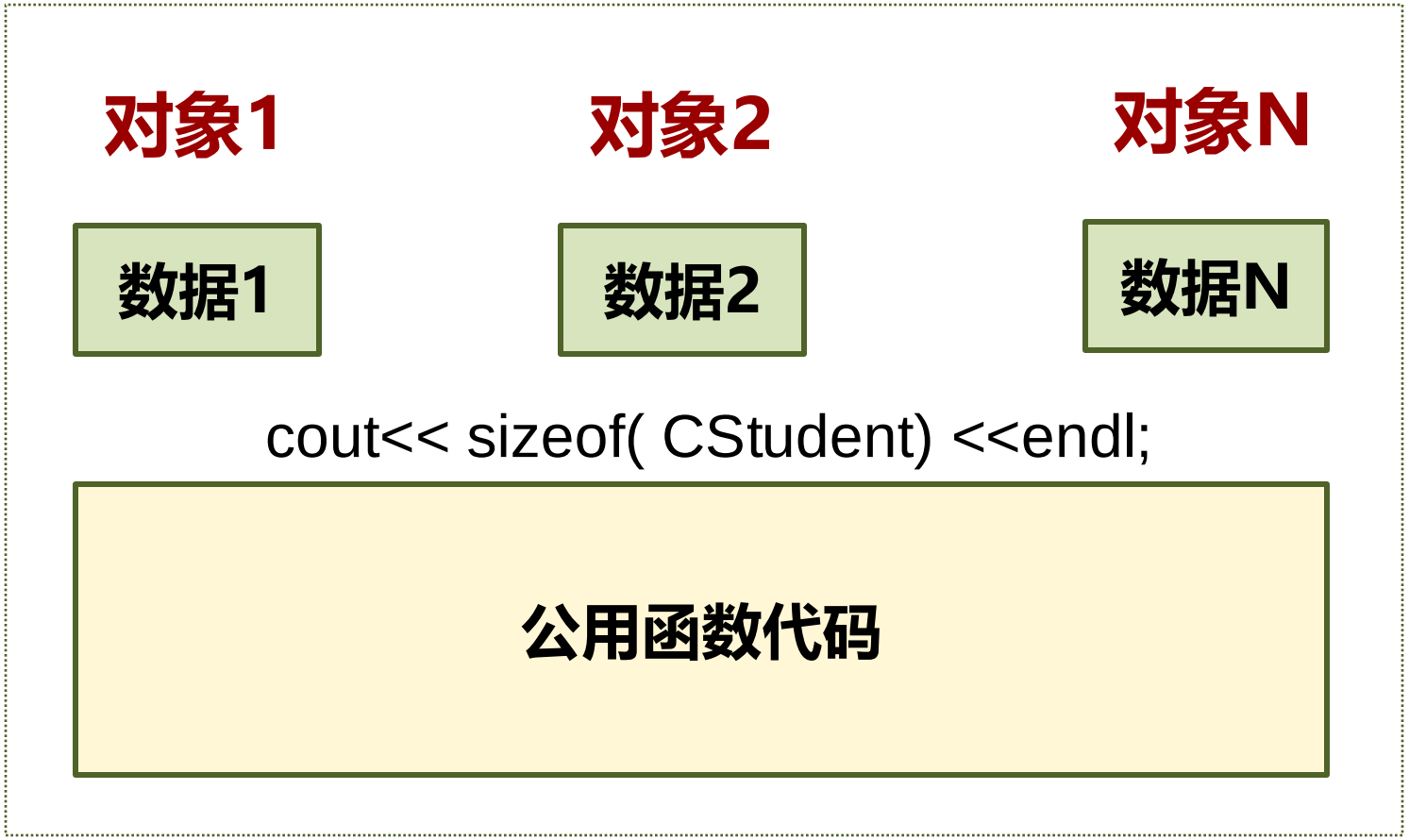
## 3. 成员函数的存储方式

用类定义对象时，系统为每个对象分配内存空间，同一类对象的成员函数是一样的如果每个对象成员函数都分配内存空间，会造成大量浪费。



## 3. 成员函数的存储方式

C++ 处理的方法是，只为对象的数据成员分配内存空间，一个类的所有对象共享一个成员函数空间



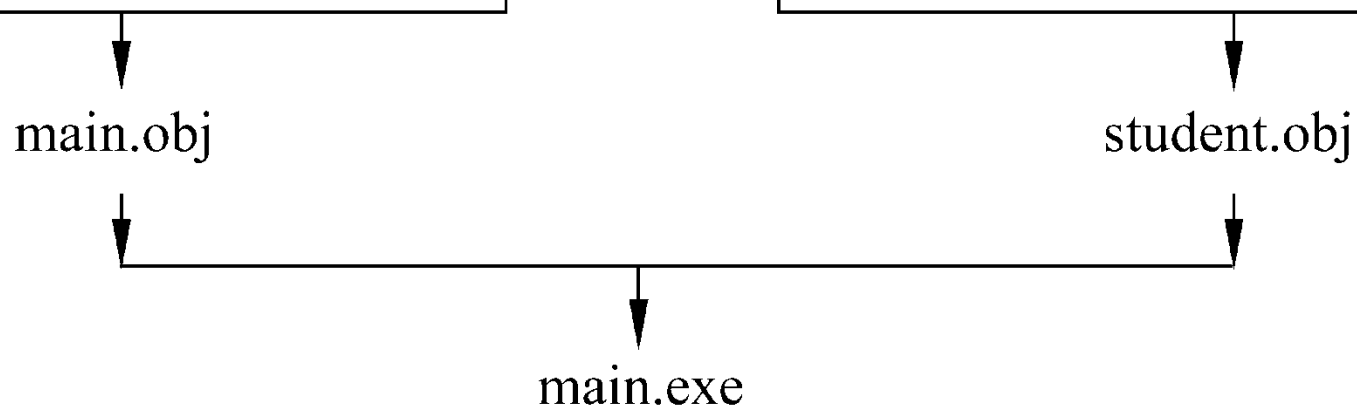
## 类声明和成员函数定义分离

主模块 main.cpp

```
#include <iostream>
#include "student.h"
void main( )
{
   …
}
```

成员函数定义文件 student.cpp

```
#include <iostream>
#include "student.h"
void Student: :display( )
{
   …
}
```



# 2.6 类和对象的简单应用

## ➢ CPoint类　要求：定义二维空间的一个点 (x,y)，并求原点到该点的欧氏距离。

```
#include <iostream>
#include <math.h>
using namespace std;

class CPoint //定义CPoint类
{
private:
float x;//定义成员变量：横坐标x
float y;//定义成员变量：纵坐标y
public:
void Create();//声明public类成员函数,创建默认二维点
void Create(float a, float b);//声明public类成员函数,初始化一个二维点
float GetX();//声明public类成员函数
float GetY();//声明public类成员函数
float GetDistance();
void ShowPoint();
};
```

```
int main()
{
CPoint p1,*p2;//定义CPoint类的普通对象p1和指针对象p2
p1.Create();
p1.ShowPoint();
p1.Create(3,3);
p1.ShowPoint();

//p2初始化
p2 = new CPoint;
p2->Create(5,6);
p2->ShowPoint();

float d = p1.GetDistance();
cout<<"Distance from [0,0] to
"<<"["<<p1.GetX()<<","<<p1.GetY()<<"] is "<<d<<endl;
return 0;
}
```

# 小结

- 2.1 面向对象程序设计方法概述
- 2.2 类的声明和对象的定义
- 2.3 类的成员函数
- 2.4 对象成员的引用
- 2.5 类的封闭性和信息隐藏
- 2.6 类和对象的简单应用

# 实验作业二 简单排序方法测试与性能比较

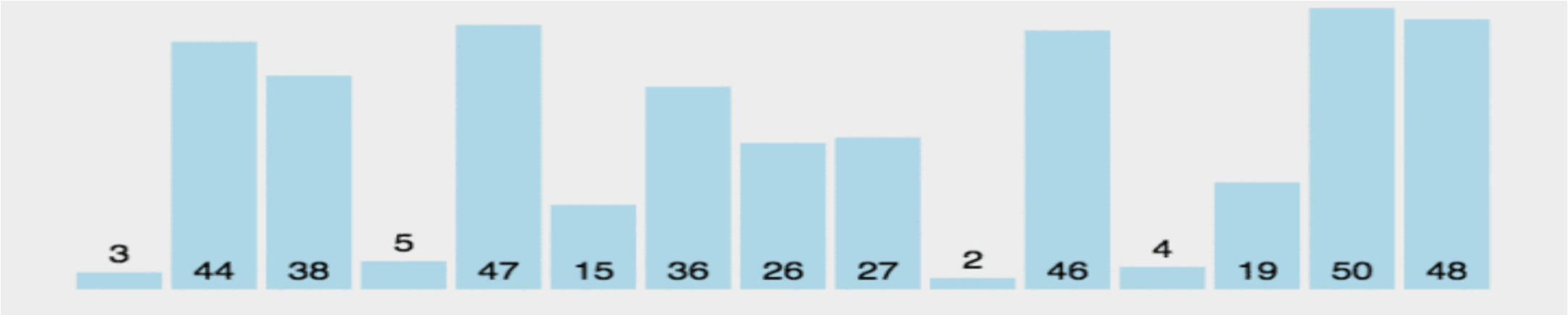
1. 查阅资料，利用随机数生成函数，产生待排序序列，序列长度可任意指定。
2. 设计CSort类，完成**冒泡排序**、**选择排序**和**插入排序**三种排序方法设计。
3. 查阅资料，调用系统函数，实现算法运行时间统计。
4. 仔细设计程序界面功能。
5. 认真按格式撰写实验作业报告，补充目的、原理等各部分内容。
6. 扩展功能：查阅资料，实现希尔排序、二路归并排序等任一种排序方法。

**实验作业的扩展功能不做要求，学有余力同学可以试一下！**

2025年2月24日-2025年3月7日

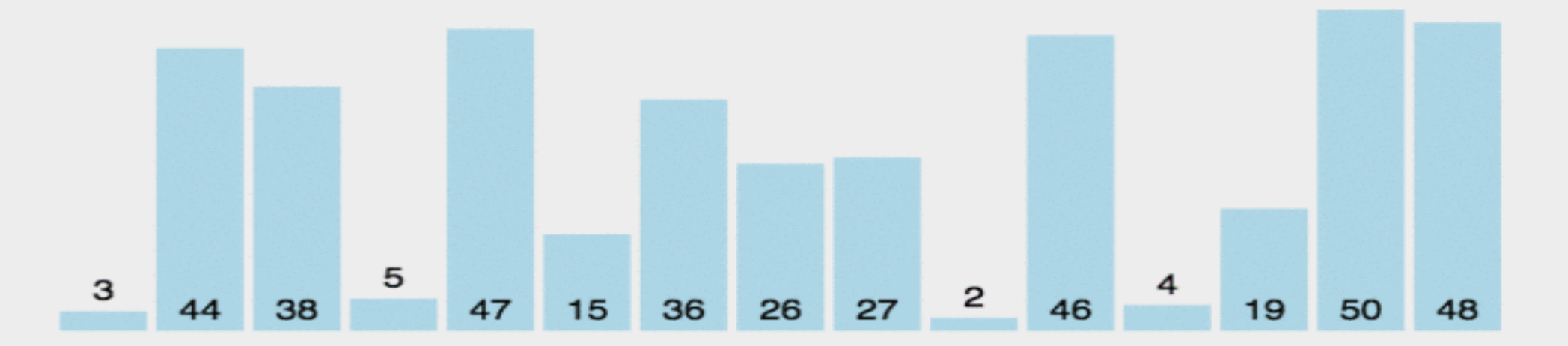
# 扩充1. 冒泡排序

**冒泡排序的基本思想：两两比较相邻记录，如果反序则交换，直到没有反序的记录为止。**



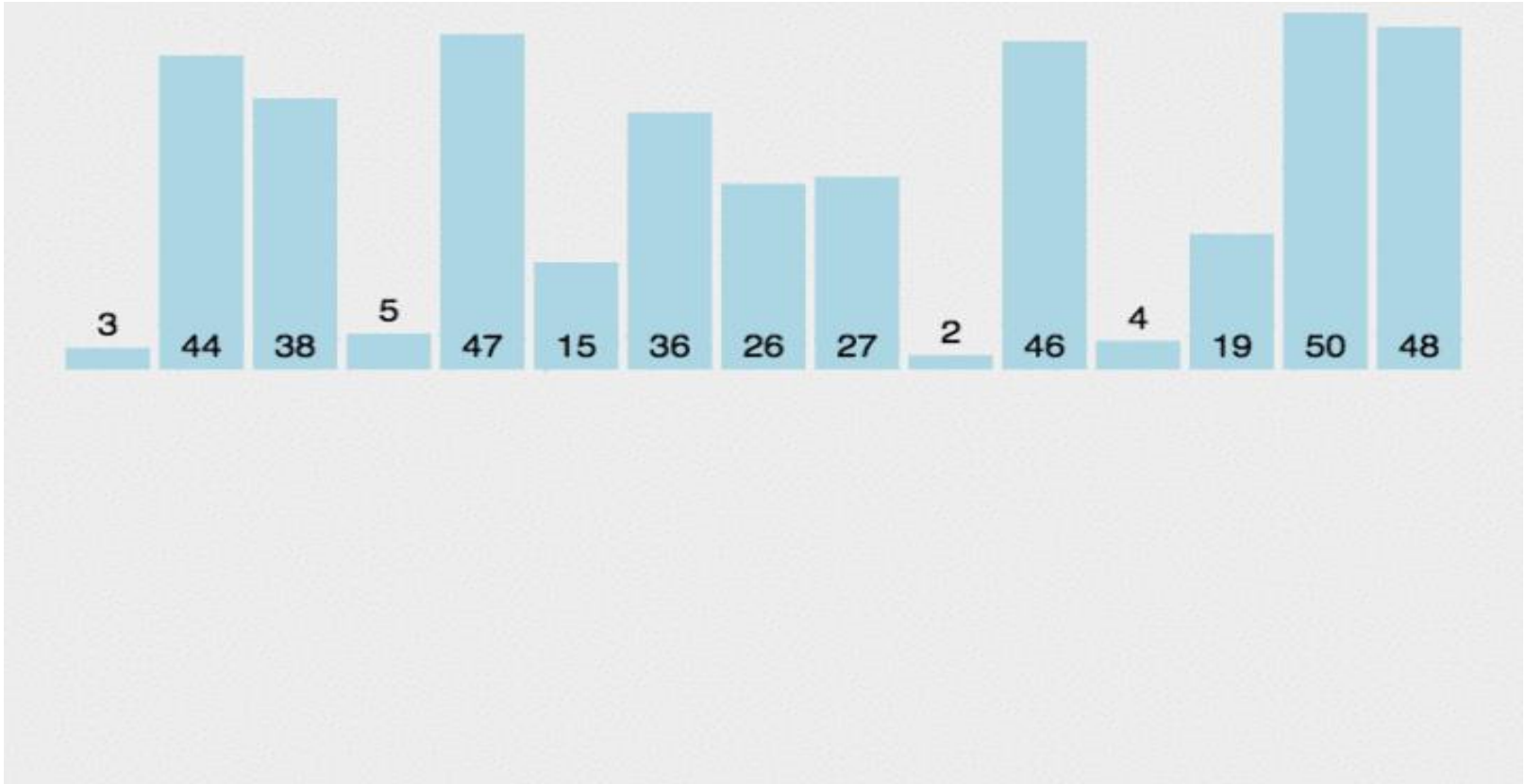
# 扩充2. 选择排序

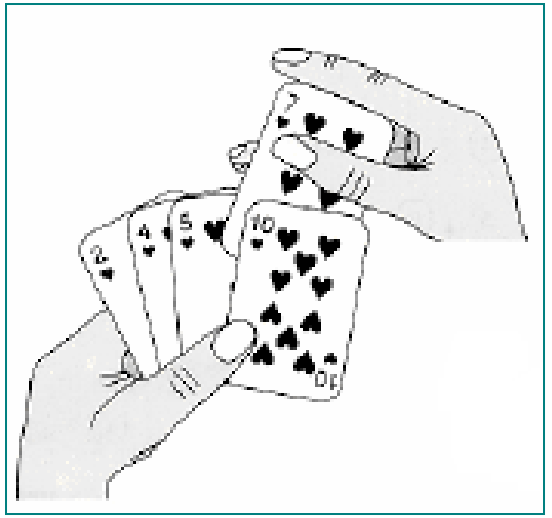
**选择排序的基本思想**：两两比较**相邻**记录，如果**反序**则记录最小值（选择无序区最小值），一趟比较后完成一次交换。



# 扩充3. 直接插入排序

**直接插入排序的基本思想：依次将待排序序列中的每一个记录插入到已排好序的序列中，直到全部记录都排好序。**




# Thank You !

## Q & A