第十五讲

Data Abstraction, Class

薛浩

2023 年 5 月 25 日

www.stickmind.com

今日话题

话题 1：编程基础 初学编程的新手，一般应该熟练使用函数和库处理字符串相关的编程任务。

话题 2：抽象数据类型的使用 在尝试实现抽象数据类型之前，应该先熟练使用这些工具解决问题。

话题 3：递归和算法分析 递归是一种强有力的思想，一旦掌握就可以解决很多看起来非常难的问题。

话题 4：类和内存管理 使用 C++ 实现数据抽象之前，应先学习 C++ 的内存机制。

话题 5：常见数据结构 在熟练使用抽象数据类型解决常见问题之后，学习如何实现它们是一件很自然的事情。

话题 4：类和内存管理

学习使用 C++ 实现数据抽象之前，应先了解 C++ 的内存机制。

- 指针和数组
- 动态内存管理
- 数据封装和类

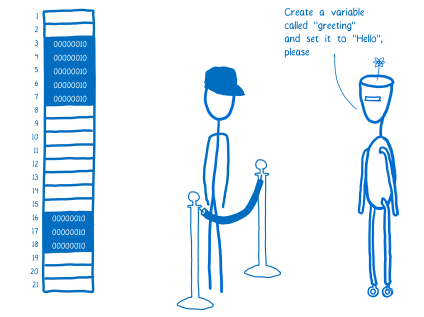

Figure 1: 数据封装和内存管理

话题 5：常见数据结构

在熟练使用抽象数据类型解决常见问题之后，学习如何实现它们是一件很自然的事情。

- 链表
- 动态数组
- 二叉堆
- 二叉搜索树

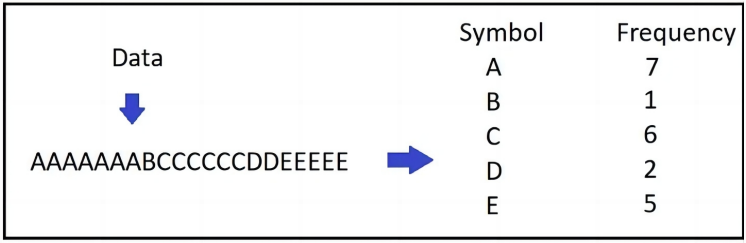

Figure 2: 数据结构和算法

C++ 是如何实现数据抽象的?

目录

复习：结构体

结构体 Structure

所有的高级编程语言都提供了类似**结构体**（Structure）的工具，用于组合不同类型的值，并且每个值都有指定的名称。C++ 是 C 语言的超集，C 语言中传统的结构体在 C++ 中有一些改进：

```
struct Point {
  int x;
  int y;
};
```

以上定义只创建了一个 Point 类型，而不是变量！如果想声明一个 Point 变量，只需要参照其他类型声明即可。注意，C++ 中声明结构体变量不需要像 C 那样追加 struct 关键字。

```
Point pt;
```

有了变量之后，可以通过点操作符访问每个成员，例如 pt.x 和 pt.y。

类和对象

类和对象

面向对象语言的特点是将数据结构表示为**对象**（object），对象将数据结构的表示和行为封装在单个实体中。

在 C 语言中，结构体用于组合不同类型的值，而行为用函数额外定义。而 C++ 将两者集成在一起。

与 Java 一样，C++ 对象模型基于**类**（class）的思想，类是描述特定类型的所有对象的模板。类定义通过**成员**（field）来指定对象的表示，并提供一组**方法**（method）来指定对象的行为。

创建的每个对象都称为该类的**实例**（instance）。

类的定义

在 C++ 中，类的一般定义形式如下：

```
class typename {
public:
    prototypes of public methods

private:
    declarations of private instance variables
    prototypes of private methods
};
```

类的定义分为两个部分：

public 该部分供类的客户端访问

private 该部分仅限于类的实现者访问

案例：Point 类

```
class Point {
public:
    Point();
    Point(int xc, int yc);
    int getX();
    int getY();
    string toString();
private:
    int x;
    int y;
};
```

类方法的实现

类定义通常以 .h 文件的形式出现，该文件定义了该类的接口。类定义没有指定类的方法实现，只有原型。在编译和执行包含类定义的程序之前，必须提供其每个方法的实现。

从风格上讲，最好定义一个单独的 .cpp 文件来隐藏这些细节。但是，由于历史原因，这种接口和实现分离的思想并不能被编译器很好的支持。为了避免不必要的麻烦，我们将类的定义和其方法的实现放在同一个 .h 文件中。

方法与函数的实现形式类似。唯一的区别是，类的名称必须写在方法名称之前，用双冒号分隔。例如，如果类 Point 的 toString 方法，实现时需要将方法名改为 Point::toString。

```
string Point::toString() {
  ...
}
```

构造器 Constructor

除了方法原型之外，类定义通常还包括一个或多个**构造函数**（constructor），用于初始化对象。

构造函数的原型没有返回类型，与类具有相同的名称。它可以接受也可以不接受参数，只要构造函数具有不同的参数列表，类就可以有多个构造函数。

不带参数的构造函数被称为**默认构造函数**（default constructor）。如果没有定义任何构造函数，C++ 将自动生成一个空的默认构造函数。

当创建一个类的实例时，即使只声明了一个变量，也总会调用该类的构造函数。

RAII-Compliant

练习：链表的类封装

```
class List {
public:
    List();
    ~List();

    void insert(char ch);
    void clear();
    void print();

private:
    Node* m_head;
};
```

析构函数 Destructor

在 C++ 中，类定义通常包含**析构函数**（destructor），用于指定如何释放该类实例的内存分配。

析构函数的原型和默认构造函数的区别是，类名的前面有波浪号。析构函数不能接受任何参数。

离开当前作用域时，C++ 都会自动调用析构函数，释放内存。对于堆上分配的对象，这个规则的效果是，C++ 程序将自动回收这些动态分配的内存。

如果使用 new 为堆中的对象分配空间，则必须通过调用 delete 显式释放该对象。调用 delete 会自动调用析构函数。

C++ 是如何实现数据抽象的?