本节主题：

抽象数据类型

以数据为核心的思维

共性？

数据！

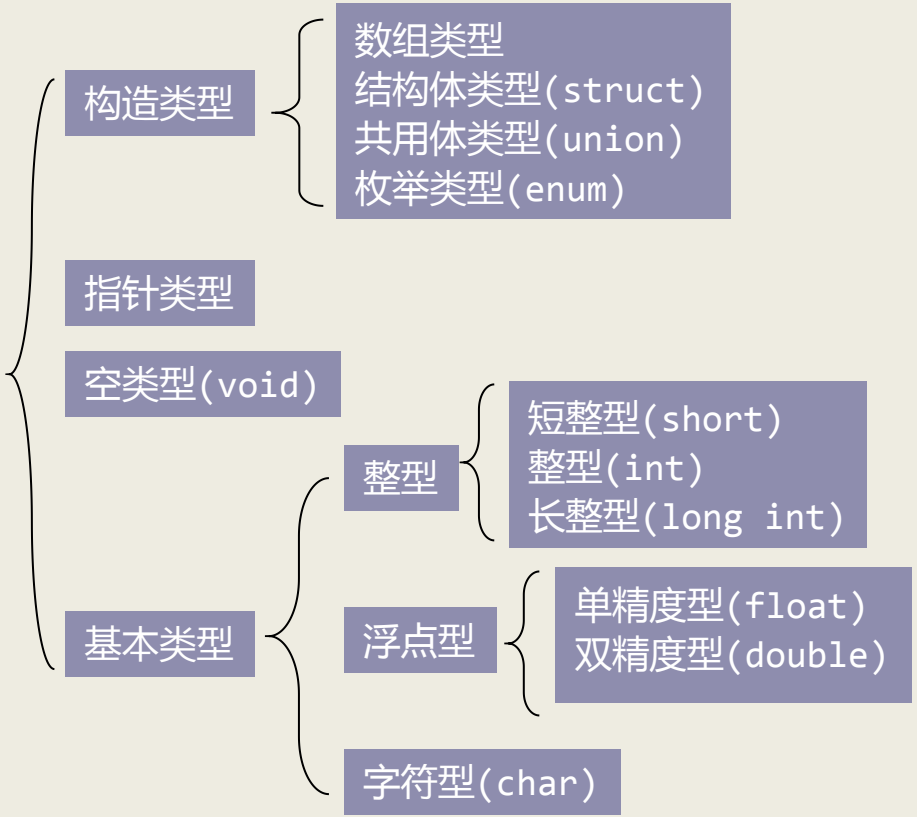
如何以数据为中心思考、设计？

忽略具体系统的形态，抓本质！

数据类型

- 高级程序语言中，一般变量、常量或表达式，都有明确的所属数据类型
- C语言中的数据类型
- 不同类型的变量，其所能取的值的范围不同，所能进行的操作不同。

数据类型是一个值的集合和定义在此集合上的一组操作的总称。



抽象数据类型(Abstract Data Type, ADT)

- 抽象数据类型 = 逻辑结构 + 抽象运算
- 抽象数据类型暂不考虑计算机的具体存储结构和运算的具体实现。
- 抽象数据类型实质上，就是在描述问题本身（与计算机无关）。
- 目标：在不涉及具体的，和计算机系统相关的细节情况下，优先理解问题本身，在此基础上，实现用计算机求解问题的过程。

```
ADT <抽象数据类型名>
{
    数据对象：<数据对象的定义>
    数据关系：<数据关系的定义>
    基本操作：<基本操作的定义>
}
```

例：抽象数据类型复数(e_1+e_2i)

```
ADT Complex
{
数据对象：
        D={e1,e2 |e1,e2均为实数}
数据关系：
        R={<e1,e2>|e1是实部，e2是虚部}
基本操作：
        AssignComplex(&z,v1,v2)：构造复数Z
        DestroyComplex(&z)：复数z被销毁
        GetReal(z,&real)：返回复数z的实部值
        GetImag(z,&Imag)：返回复数z的虚部值
        Add(z1,z2,&sum)：返回两个复数z1、z2的和
}
```


```
class Complex
{
public:
   Complex( );
   Complex(double r,double i);
   Complex add(Complex &c2);
   void display( );
private:
   double real;
   double imag;
};
```

体会：数据抽象、数据封闭

- 抽象数据类型中对数据对象和数据运算的声明，将对数据对象的表示和数据运算的实现分离。
- 抽象数据类型的两个重要特征
  - 数据抽象

    用ADT描述程序处理的实体时，强调的是其本质的特征、其所能完成的功能，以及它和外部用户的接口。

  - 数据封装

    将实体的外部特征和内部实现细节分离，并且对外部用户隐藏其内部实现细节。

```
ADT <抽象数据类型名>
{
    数据对象：<数据对象的定义>
    数据关系：<数据关系的定义>
    基本操作：<基本操作的定义>
}
```

思考题

- 数据类型和抽象数据类型有什么不同？