## 1、数据结构

### 1.1为什么要学习数据结构

软件设计是计算机各个领域的核心。

软件设计时要考虑的首要问题：数据的表示、组织和处理方法。

数据结构设计和算法设计是软件系统设计的核心。

“数据结构 + 算法 = 程序设计”

### 1.2什么是数据结构

数据（data）：符号集合，处理对象。

数据元素（data element）：由数据项（data item）组成。

关键字（key）：识别元素，主关键字（primary key）唯一识别元素。

数据结构（data structure）：指数组元素之间存在的关系。包含以下三方面：

1）数据的逻辑结构

线性结构：数据元素只有一个前驱数据元素和一个后继数据元素。

树结构：每个数据元素只有一个前驱数据元素，可有零个或若干个后继数据元素。

图结构：每个数据元素可有零个或若干个前驱数据元素，零个或若干个后继数据元素。



2）数据的存储结构

顺序存储结构



链式存储结构



3）数据操作

* 初始化。
* 判断是否空状态。
* 存取，指获得、设置指定元素值。
* 统计数据元素个数。
* 遍历（traverse），指按照某种次序访问一个数据结构中的所有元素，并且每个数据元素只被访问一次。遍历一种数据结构，将得到一个所有数据元素的线性序列。
* 插入（insert）、删除（remove）指定元素。
* 查找（search），指在数据结构中寻找满足给定条件的数据元素。
* 排序（sort），指对数据元素按照指定关键字值的大小递增（或递减）次序重新排列。

### 1.3数据类型与抽象数据类型

1）数据类型（data type）是指一个类型和定义在这个类型上的操作集合。

2）抽象数据类型（Abstract Data Type，ADT）是指一个逻辑概念上的类型和这个类型上的操作集合。即，一种数据结构的抽象数据类型包括：数据的逻辑结构和数据操作。

复数抽象数据类型

ADT Complex //复数抽象数据类型

{

double real, imag; //实部和虚部

Complex(double real, double imag)

Complex add(Complex c) //加法

Complex sub(Complex c) //减法

}

### 1.4集合的表示与实现

ADT Set<T> //集合抽象数据类型

{

数据：集合中的数据元素，数据元素的数据类型为T

操作：

boolean isEmpty(); //判断集合是否为空

int size(); //返回元素个数

T search(T key) //返回查找到的关键字为key元素

boolean contains(T key) //判断是否包含关键字为key元素

boolean add(T x) //增加元素

T remove(T key) //删除关键字为key元素

void clear() //删除所有元素

String toString() //返回所有元素的描述字符串

}

1）集合抽象数据类型

ADT Set<T>

{

boolean equals(Object obj) //比较this与obj是否相等

Object[] toArray() //返回包含所有元素的数组

// 以下方法描述集合运算，参数是另一个集合

boolean containsAll(Set<?> set) //判断是否子集

boolean addAll(Set<? extends T> set)//集合并运算

boolean removeAll(Set<?> set) //集合差

boolean retainAll(Set<?> set)//仅保留那些包含在set的元素，集合差

}

2）实现不同特性的集合

线性表

* 表示可重复的无序集合；
* 元素间具有前驱、后继次序关系；
* 不同元素的关键字可重复，采用序号能够识别关键字重复的数据元素。

排序线性表

* 表示可重复的排序集合；
* 元素按关键字大小次序排序。

散列表

* 表示不可重复的无序集合；
* 元素关键字不重复，元素间没有次序，不排序。

二叉排序树

* 表示不可重复的排序集合；
* 元素关键字不重复，元素按关键字升/降序排序。

### 1.5用Java语言的接口描述抽象数据类型

Java语言的接口（interface）是一组抽象方法、常量和内嵌类型的集合。

1）声明接口：

public interface Set<T> //集合接口，T是泛型参数

2）声明实现接口的类：

public abstract class AbstractSet<T> implements Set<T>

//抽象集合类，没有实现所有抽象方法

public class HashSet<T> implements Set<T> //散列表类

3）接口是引用类型：

Set<T> set = new HashSet<T>(); //接口对象引用实例

set.add(x) //运行时多态性，执行HashSet<T>类实现的add(x)方法

## 2、算法

### 2.1什么是算法

一个算法（Algorithm）是一个有穷规则的集合，其规则确定一个解决某一类特定类型问题的操作序列。

|  |  |
| --- | --- |
| 算法定义 | 算法设计目标 |
| a.有穷性 | a.正确性 |
| b.确定性 | b.可读性 |
| c.输入 | c.健壮性 |
| d.输出 | d.高时间效率 |
| e.可行性 | e.高空间效率 |

1）算法描述

// 在当前数据结构中，顺序查找与key相等的元素（数据类型为T）；key提供查找条件的关键字元素

search(T key) {

for(elem:数据结构中的每个元素) { //遍历

if(key与elem元素相等) {

//由T类型约定两个元素相等的比较规则

查找成功，返回元素或元素位置；

}

若查找不成功，返回查找不成功标记；

}

}

2）算法与数据结构（线性表插入操作）





### 2.2算法分析

1）度量算法的时间效率

算法的时间效率：指算法执行时间随问题规模的增长而增长的趋势，通常采用时间复杂度来度量算法的时间效率。

T(n) = O(f(n))

2）度量算法的空间效率

空间复杂度：指算在执行时为解决问题时需要的额外内存空间（不包括输入数据所占用的存储空间）。

S(n) = O(f(n))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时间复杂度 | n=8(23) | n=10 | n=100 | n=1000 |
| O(1) | 1 | 1 | 1 | 1 |
| O(log2n) | 3 | 3.322 | 6.644 | 9.966 |
| O(n) | 8 | 10 | 100 | 1000 |
| O(nlog2n) | 24 | 33.22 | 664.4 | 9966 |
| O(n2) | 64 | 100 | 10000 | 106 |

3）时间复杂度随n变化情况的比较

4）算法时间复杂度分析

一条简单语句的时间复杂度是O(1)。

int count = 0;

一个循环的时间复杂度是O(n)。

int n = 8, count = 0;

for(int i = 1; i <= n; i++)

count++; //循环体执行n次

循环语句的时间复杂度是O(log2n)。

for(int i = 1; i <= n; i\*=2) //按2的幂（1,2,4,8）递增

count++; //循环体执行1+log2n次

二重循环的时间复杂度为O(n2)。

for (int i=1; i<=n; i++)

for (int j=1; j<=n; j++）

二重循环的时间复杂度是O(n\*log2n)。

for (int i=1; i<=n; i\*=2) //循环log2n次

for (int j=1; j<=n; j++) //循环n次

二重循环的时间复杂度是O(n)。

for (int i=1; i<=n; i\*=2) //循环log2n次

for (int j=1; j<=i; j++) //循环i次

//循环次数2n - 1



### 2.3算法设计

1）求两个整数的最大公约数。

目的：说明算法的必要性。

质因数分解法（26460,12375）

26460 = 22 \* 33 \* 55\* 72 12375 = 32 \* 53 \* 11

最大公约数（26460,12375） = 32 \* 5 = 45

更相减损术（91,49）

“以少减多，更相减损，求其等也，以等数约之。等数约之，即除也，其所以相减者皆等数之重叠，故以等数约之。”：(91,49)=(42,49)=(42,7)=7。

2）欧几里得（Euclid）的辗转相除法

gcb(a,b) = gcb(b,a)

gcb(a,b) = gcb(-a,b)

gcb(a,0) = |a|

gcb(a,b) = gcb(b,a%b), 0 <= a%b < b

gcb(91,49) = gcb(49,42) = gcb(42,7) = gcb(7,0) = 7

3）返回a与b的最大公约数

int gcb(int a, int b) {

while(b!=0)

{

int temp = a % b;

a = b

b = temp;

}

return a;

}

4）线性存储的随机数集合

本题目的：

* 使用Java的一维数组，对象数组作为方法的参数和返回值；
* 输出对象的通用方法；
* 声明类，包含对数组操作的通用方法；
* 随机数集合，线性存储。

Java的数组：

Java数组的特点是动态数组、具有长度属性length、引用数据类型。

for语句的逐元循环（for each loop），作用于数组的语法格式如下：

for(类型 变量:数组)

// <变量> 获得 <数组> 的每个元素

声明Array1类

public class Array1 {

public static void print(Object[] value)

public static Integer[] randomInteger(int n, int size)

}

注：java.lang.Integer是int整数类型的包装类

调用时默认将int整数与Integer对象相互转换。例如：

Integer key = new Integer(100);

Integer key = 100; //与上句等价，Java自动将int整数封装成Integer对象

int i = key.initValue();

int i = key; //Java自动调用Integer的initValue()方法，将Integer对象转换成int整数。

Integer[] randomDifferent(int n, int size)

// 返回n个互异的随机数，范围是0~size-1

Integer[] randomSorted(int n, int size)

// 返回n个排序的随机数

Integer[] randomDifferentSorted(int n, int size)

// 返回n个互异的排序的随机数