高级语言程序设计

张伯雷
bolei.zhang@njupt.edu.cn
bolei-zhang.github.io
计算机学院，软件教学中心

高级语言程序设计

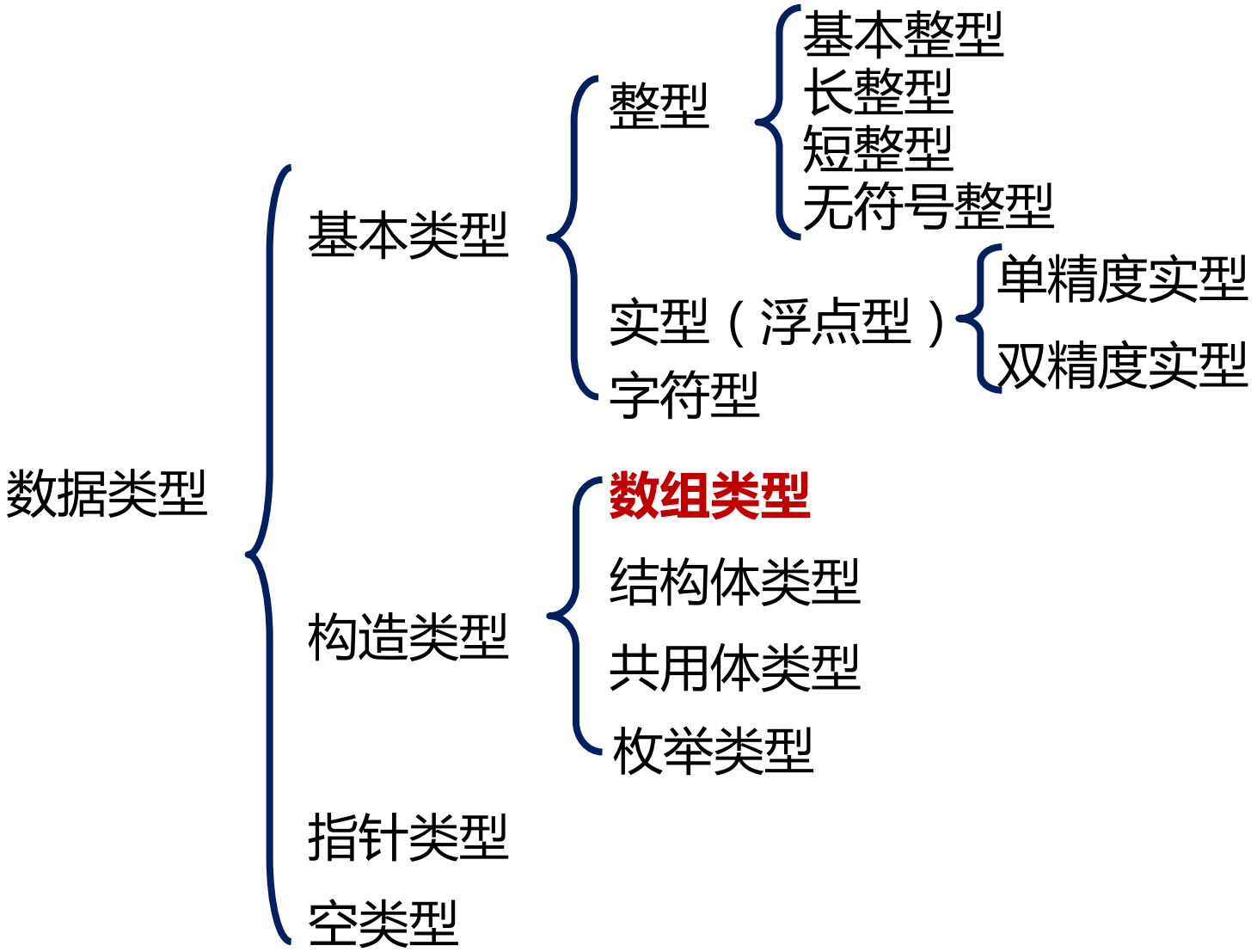
第06章 数组

1. 初识计算机、程序与C语言
2. 初识C源程序及其数据类型
3. 运算符与表达式
4. 程序流程控制
5. 函数
    1. 函数的定义、调用
    2. 变量的作用域和生命周期

内容提要

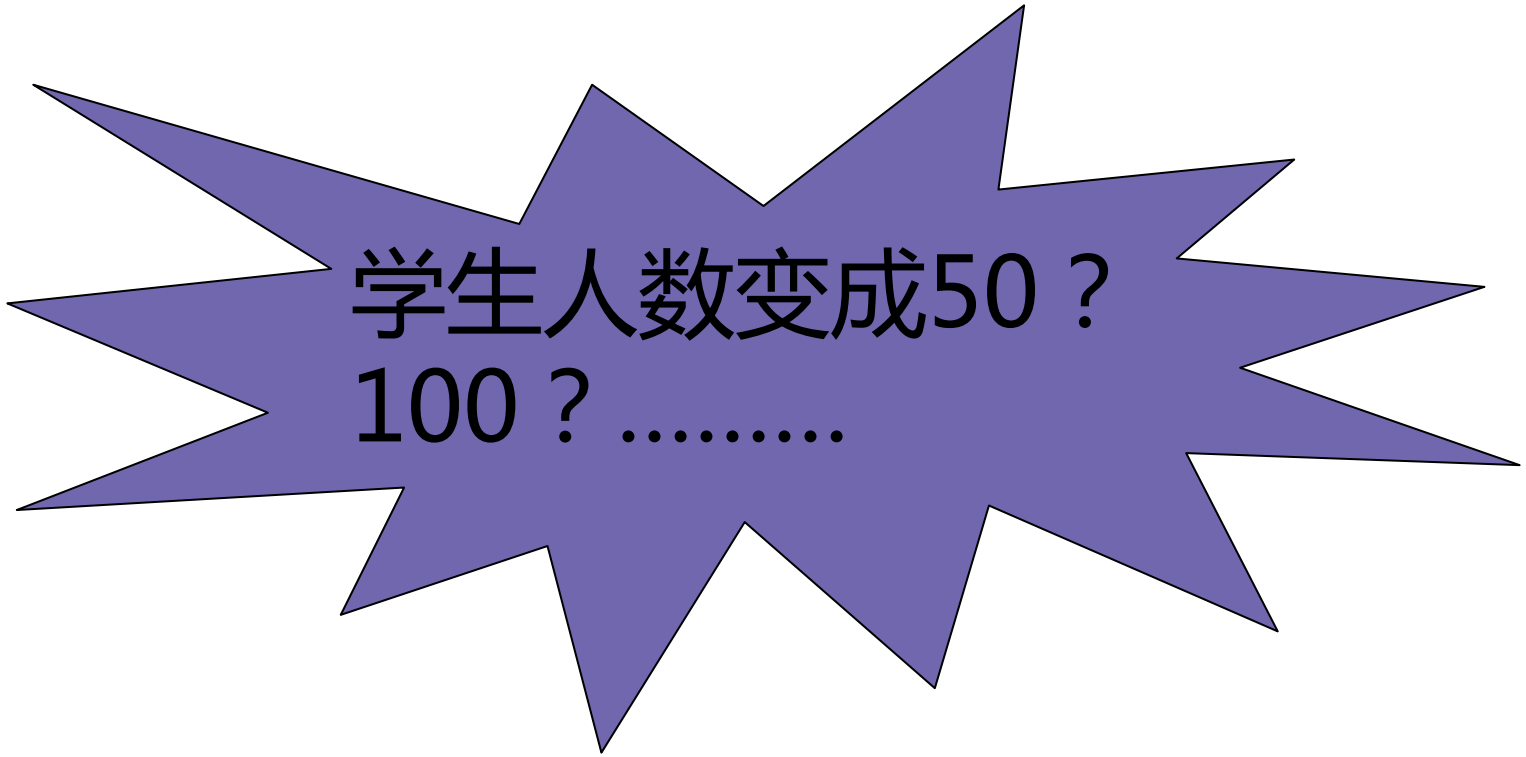
- 一维数组
- 二维数组
- 数组常用算法
  - 查询
  - 插入
  - 删除
  - 排序

C语言中的数据类型


数据类型
基本类型
整型
基本整型
长整型
短整型
无符号整型
实型（浮点型）
单精度实型
双精度实型
字符型
构造类型
数组类型
结构体类型
共用体类型
枚举类型
指针类型
空类型

数组的作用

- 问题：
  - 输入3个学生的c语言成绩，计算平均分。



数组

- 数组用于保存大量同类型的相关数据
- 数组是具有一定顺序关系的若干相同类型变量的集合体。
- 组成数组的变量称为该数组的元素。
- 例6_1:50个学生成绩的存储与处理

6.1 一维数组

- 定义格式：

  数据类型 数组名[常量表达式]

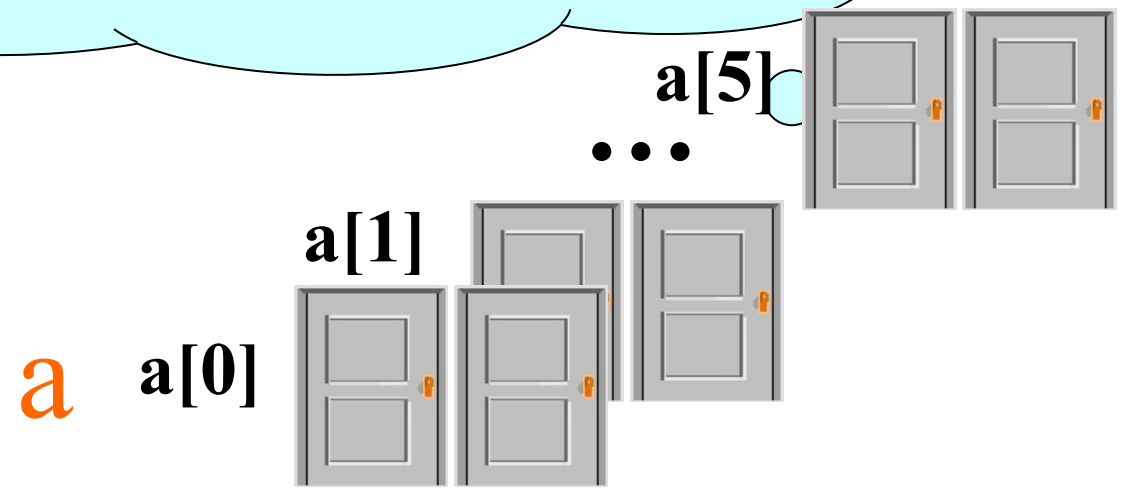
  - 例： int score[5];
  - 定义一个有5个元素的数组
  - 规定下标从0开始，可使用score[0]、score[1]、score[2] 、score[3]、score[4]访问每个元素
  - 每个元素都是类型为int的变量，像使用普通int型变量一样使用
  - 定义后，数组长度不能改变

整型，表示数组的元素个数

一维数组的存储

- 数组元素在内存中顺次存放，它们的地址是连续的。
  - 例 int a[6];
  - 系统会在内存分配连续的6个int空间给此数组

一维数组所占的字节数
= 数组长度×sizeof(元素类型)



一维数组的初始化

- 一维数组的初始化：在定义数组时为其全部或部分元素指定初值。
- 正确的初始化示例：

```
int a[5]={1,2,3,4,5};
int a[ ]={1,2,3,4,5};
int a[5]={1,2*4};
int a[5]={0};
```

- 错误的初始化示例：

```
int a[5]={,2,3 };
int a[5]={1,2,3,4,5,6,7};
```

相当于a[5]={1,2,3,4,5};数组长度省

相当于a[5]={1,8,0,0,0};部分赋值注意！当不做初始化时所有数组元素的值均为随机数而不是0

对数组元素初始化时只能从左到右依次，只能缺省最右边的元素值

初值个数不能超过数组元素的个数

应用举例---数列

例6_2：Fibonacci数列

时间（月）	小兔子（对）	大兔子（对）	兔子总数（对）
1	1	0	1
2	0	1	1
3	1	1	2
4	1	2	3
5	2	3	5
6	3	5	8
7	5	8	13
8	8	13	21
…	…	…	…

- 算法分析：
- 1)求出n个月后兔子的数目
- 2)问要几个月后兔子的数目超过1000对？

应用举例—数位统计

- 例6_3：统计一个整数中各个数字出现的次数。
- 算法分析：
  - 数位分离：do-while
  - 统计方法

应用举例：最值求解

- 例6_4：用函数完成数组中最大值的求解
- 数据类型：一维数组存储待比较的数据
- 算法：
  - 最值的求解
  - 函数之间批量数据的共享

```
int main()
{
   int array[N],n;
   int max;
 .....
 max=max_num(array,n);
......}
```

```
int max_num(int a[],int n)
{
   int i,max;
   max=a[0] ;
   for (i=1;i<n; i++)
       if (a[i]>max)
              max=a[i];
   return max;
}
```

课堂练习

扩展例6.4：

1）用函数完成数组最小值的求解。

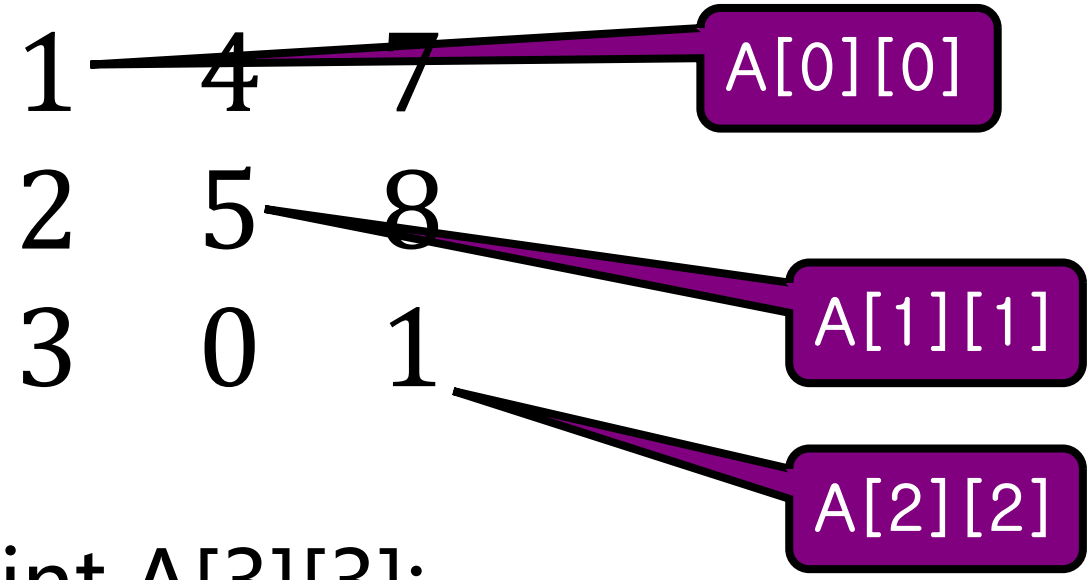
2）用函数完成数组的求和。

- 1. 找出100到999之间的水仙花数：$153 = 1^3+5^3+3^3$

- 2.通过辗转相除法计算两个非负整数a，b的最大公约数。

6.2 二维数组

- 例6_5 存储矩阵，并输出对角线



- 二维数组定义：int A[3][3];

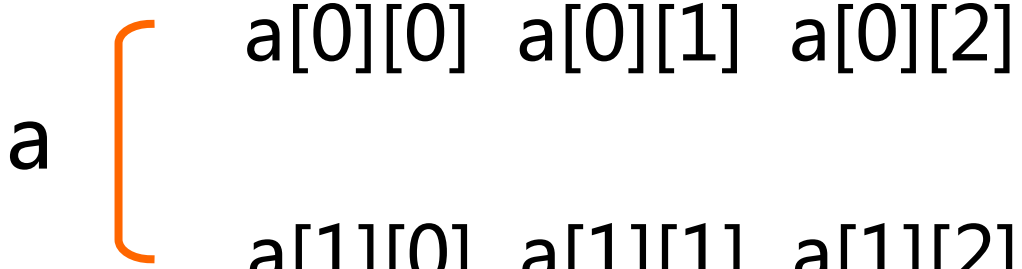
行数　列数

二维数组的定义

定义格式：

类型标识符 数组名[整型常量表达式1] [整型常量表达式2];

例 int a[2][3];



数组所占的字节数

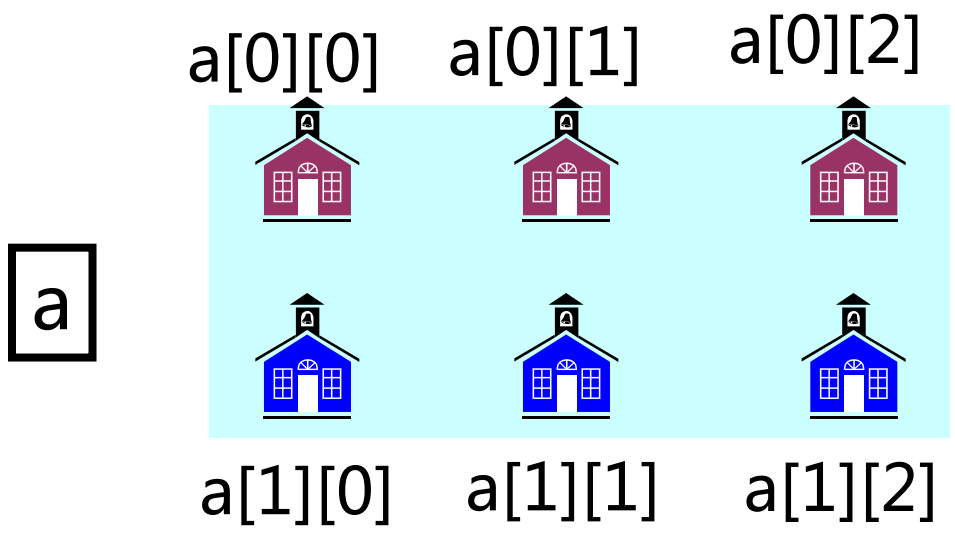
=第一维长度*第二维长度 *sizeof(数据类型)

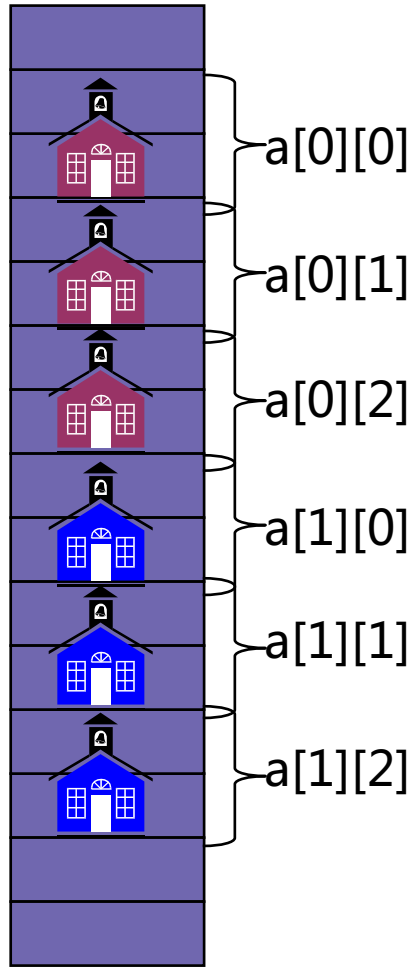
二维数组的存储

原则：行优先

先顺序存放第0行的元素，再存放第1行的元素……

例如：int a[2][3];





二维数组的初始化

- 原则：按行从左到右依次赋值
  - 逐行初始化：

    int a[4][3]={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
  - 不分行，用类似一维数组的方式初始化：

    int a[4][3]={1,2,3,4,5,6,7,8,9,10,11,12};
  - 全部赋值时，行数可以缺省，列数不能省 ：

    int a[ ][3]={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
  - 部分赋值

    int a[4][3]={{1,2},{4,5},{7,8,9},{10,11,12}};

    int a[3][4]={1,0,6,0,0,11};
  - 最简单的初始化：

    int a[4][3]={0};

二维数组的访问

- 例6_6 矩阵转置运算
  - 二维数组元素的访问需用两层循环

例：int a[3][4],n=1;

```
for(int i=0;i<4;i++)
    for(int j=0;j<3;j++)
        a[i][j]=n++;
```

  - 数据类型:由于矩阵行、列不同，定义两个二维数组分别存放转置前、后的矩阵
  - 算法提示:只需要在控制循环时先控制列下标再控制行下标就能实现矩阵的转置

应用举例：模拟发牌

- 例6_7 模拟扑克牌游戏中的发牌过程，随机将52张扑克发给两个玩家。
- 关键问题：
  - 如何表示某张扑克牌？
  - 发牌过程
    - 随机抽出一张扑克牌
    - 标注这张扑克牌属于谁
  - 显示发牌结果

课外知识：计算机里的随机数

- 在宏观物理中，没有真正的随机数
- 量子力学
- 计算机中的随机数
  - 统计意义上的随机数

```
int __cdecl rand (
        void
        )
{
        _ptiddata ptd = _getptd();

        return( ((ptd->_holdrand = ptd->_holdrand * 214013L
            + 2531011L) >> 16) & 0x7fff );
}
```

1. 假设我们手头已经有一个数字，我们称它们为**种子**。
2. 对这些种子用递归法（如线性同余），生成**一连串0到某个自然数N之间的自然数**。
3. 利用这串随机数，把它们用某些算法来**转换成其他分布的随机数**。

课外知识：计算机里的随机数

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main(){
    printf("%d\n", rand());
    printf("%d\n", rand());
}
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main(){
    srand(time(NULL));
    printf("%ld\n", time(NULL));
    printf("%d\n", rand());
    printf("%d\n", rand());
}
```

第一个rand()和第二个的输出一致吗？
把这个程序运行多次，出来的结果一样吗？

应用举例：九九乘法表

- 例6_8 实现九九乘法表并输出用户要求的格式。
- 关键问题：
  - 九九乘法表的存储：二维数组
  - 显示方式：
    - 全部
    - 下三角
    - .....

    提供的菜单选项
- 关注：二维数组作为函数的形参

数组常用算法：查找

- 例6_9 在a数组中查找x，如果存在输出它的下标，否则提示：“Not present!”。
- 算法：
  - 顺序查找法，对数组排列无要求.
  - 查找的过程:从第一个元素开始依次与待查找的元素进行比较，如果相等就查找成功，输出元素及对应下标；如果与所有元素都比较结束仍没有相等元素，则输出元素不存在的提示信息
- 数据结构：
  - 数组a、变量x用于存储要查找的数据
  - 循环控制变量i，循环结束条件：i>=n或者x==a[i]

查找函数的实现

- 接口确定：
  - 功能
  - 入口：数组名、数组长度、待查找的数
  - 返回

```
#define SIZE 10
int main()
 {……
     //读入array的长度、值和x
    pos=find(array,n,x);
    if(pos<n)
          printf( “value=%d, …);
   else
          printf("Not present!\n");
…….
}
```

```
int find(int a[],int n,int x)
{
    int i=0;
    while(i<n)
   {
          if (x==a[i])
              break;
           i++;
    }
    return i;
}
```

数组常用算法：元素插入

- 例6_10 将元素x插入数组a中保持数组非递减有序
- 算法：有序数组中插入一个数据元素，并保持数组有序。
  - （1）确定待插入位置—循环
  - （2）元素后移，腾出相应位置---后移方法
  - （3）在“空”位置上插入新元素
- 数据结构：
  - 数组a、变量x用于输入要插入的数据
  - 循环控制变量i，两次循环，查找位置以及移动

insert函数的实现

- 接口确定：
  - 功能：有序插入,升序
  - 入口：数组名、数组长度、待插入的数
  - 返回

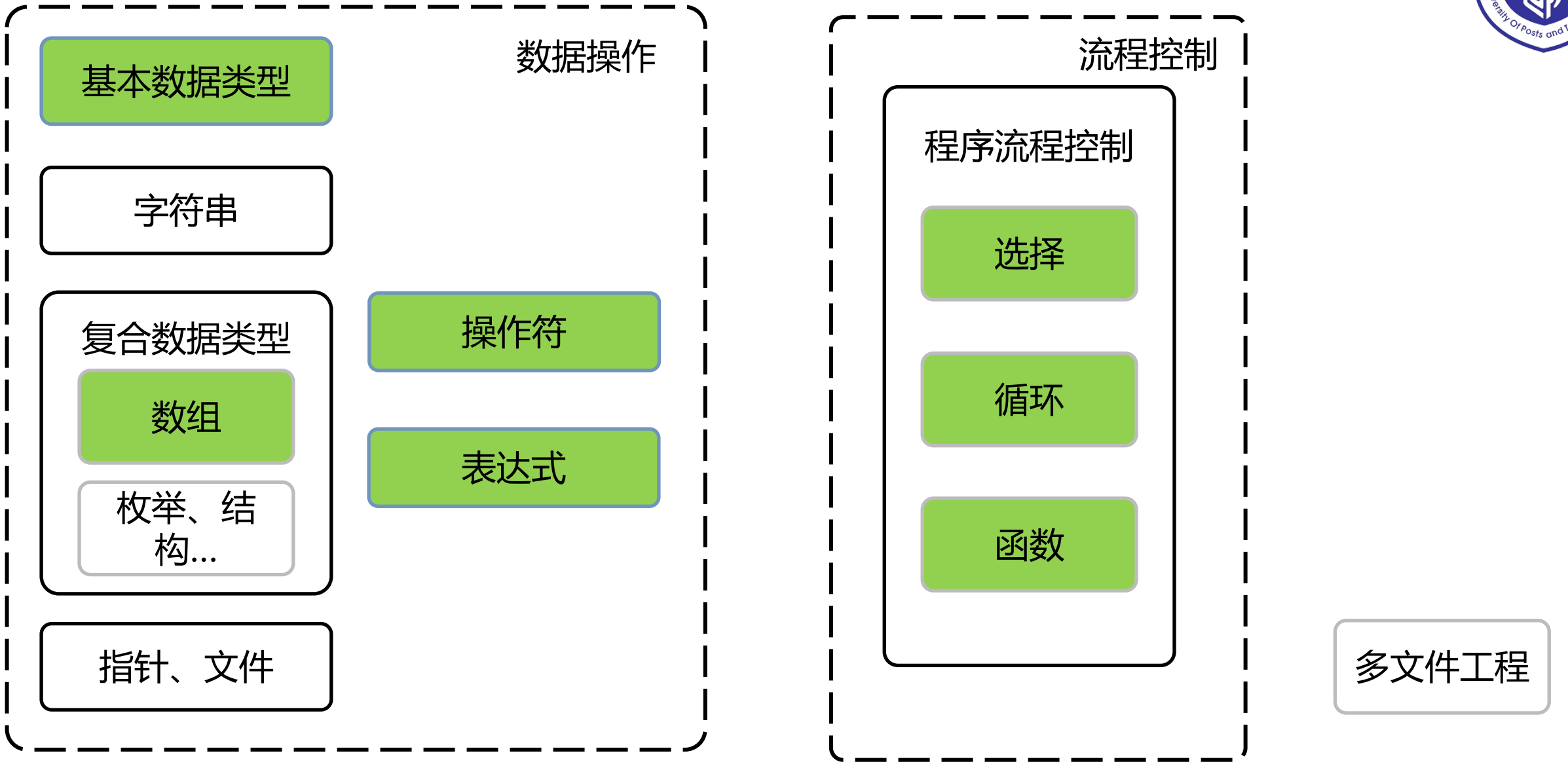
```
#define SIZE 7
int main()
 {.....
     //初始化array值、读入x
    insert(array,SIZE-1,x);
     .......
}
```

```
void insert(int a[],int n,int x)
{
   for (i=0;i<n&&a[i]<x;i++);
                                  /*定位*/

    for (j=n-1;j>=i;j--)    /*移位*/
         a[j+1]=a[j];

     a[i]=x;                        /*插入*/
}
```

C语言


数据操作
基本数据类型
字符串
复合数据类型
数组
枚举、结构...
指针、文件
操作符
表达式
流程控制
程序流程控制
选择
循环
函数
多文件工程

29

高级语言程序设计

张伯雷
bolei.zhang@njupt.edu.cn
bolei-zhang.github.io
计算机学院，软件教学中心

高级语言程序设计

第06章 数组

1. 初识计算机、程序与C语言
2. 初识C源程序及其数据类型
3. 运算符与表达式
4. 程序流程控制
5. 函数
    1. 函数的定义、调用
    2. 变量的作用域和生命周期

图灵奖得主

年份	中文译名	姓名	贡献领域/获奖理由
1966年	艾伦·佩利	Alan J. Perlis	高级程序设计技巧，编译器构造
1967年	莫里斯·威尔克斯	Maurice V. Wilkes	存储程序式计算机EDSAC，程序库
1968年	理查德·卫斯里·汉明	Richard Hamming	数值方法，自动编码系统，错误检测和纠错码
1969年	马文·明斯基	Marvin Minsky	人工智能
1970年	詹姆斯·维尔金森	James H. Wilkinson	数值分析，线性代数，倒退错误分析
1971年	约翰·麦卡锡	John McCarthy	人工智能
1972年	艾兹格·迪科斯彻	Edsger Dijkstra	程序设计语言的科学与艺术
1973年	查理士·巴赫曼	Charles W. Bachman	数据库技术
1974年	高德纳	Donald E. Knuth	算法分析、程序设计语言的设计、程序设计
1975年	艾伦·纽厄尔	Allen Newell	人工智能，人类认知心理学和列表处理（list processing）
	赫伯特·西蒙	Herbert A. Simon	
1976年	迈克尔·拉宾	Michael O. Rabin	非确定性自动机
	达纳·斯科特	Dana S. Scott	
1977年	约翰·巴克斯	John Backus	高级编程系统，程序设计语言规范的形式化定义
1978年	罗伯特·弗洛伊德	Robert W. Floyd	设计高效可靠软件的方法学
1979年	肯尼斯·艾佛森	Kenneth E. Iverson	程序设计语言和数学符号，互动系统的设计，运用 APL进行教学，程序设计语言的理论与实践
1980年	东尼·霍尔	C. Antony R. Hoare	程序设计语言的定义与设计
1981年	埃德加·科德	Edgar F. Codd	数据库系统，尤其是关系型数据库

图灵奖得主

1982年	史提芬·古克	Stephen A. Cook	计算复杂度
1983年	肯·汤普逊	Ken Thompson	UNIX操作系统和C语言
	丹尼斯·里奇	Dennis M. Ritchie	
1984年	尼古拉斯·沃斯	Niklaus Wirth	程序设计语言设计、程序设计
1985年	理查德·卡普	Richard M. Karp	算法理论，尤其是NP-完全性理论
1986年	约翰·霍普克罗夫特	John Hopcroft	算法和数据结构的设计与分析
	罗伯特·塔扬	Robert Tarjan	
1987年	约翰·科克	John Cocke	编译理论，大型系统的体系结构，及精简指令集（RISC）计算机的开发
1988年	伊凡·苏泽兰	Ivan Sutherland	计算机图形学
1989年	威廉·卡亨	William Morton Kahan	数值分析
1990年	费尔南多·考巴脱	Fernando J. Corbató	CTSS和Multics
1991年	罗宾·米尔纳	Robin Milner	LCF，ML语言，CCS
1992年	巴特勒·兰普森	Butler W. Lampson	分布式，个人计算环境
1993年	尤里斯·哈特马尼斯	Juris Hartmanis	计算复杂度理论
	理查德·斯特恩斯	Richard E. Stearns	
1994年	爱德华·费根鲍姆	Edward Feigenbaum	大规模人工智能系统
	拉吉·瑞迪	Raj Reddy	
1995年	曼纽尔·布卢姆	Manuel Blum	计算复杂度理论，及其在密码学和程序校验上的应用
1996年	阿米尔·伯努利	Amir Pnueli	时序逻辑，程序与系统验证
1997年	道格拉斯·恩格尔巴特	Douglas Engelbart	互动计算
1998年	詹姆斯·尼古拉·格雷	James Gray	数据库与事务处理
1999年	弗雷德里克·布鲁克斯	Frederick Phillips Brooks，Jr.	计算机体系结构，操作系统，软件工程
2000年	姚期智	Andrew Chi-Chih Yao	计算理论，包括伪随机数生成，密码学与通信复杂度
2001年	奥利-约翰·达尔	Ole-Johan Dahl	面向对象编程
	克利斯登·奈加特	Kristen Nygaard	

图灵奖得主

2003年	艾伦·凯	Alan Kay	面向对象编程
2004年	文特·瑟夫	Vinton G. Cerf	TCP/IP协议
	罗伯特·卡恩	Robert E. Kahn	
2005年	彼得·诺尔	Peter Naur	Algol 60语言
2006年	法兰西斯·艾伦	Frances E. Allen	优化编译器
2007年	爱德蒙·克拉克	Edmund M. Clarke	开发自动化方法检测计算机硬件和软件中的设计错误
	艾伦·爱默生	Allen Emerson	
	约瑟夫·斯发基斯	Joseph Sifakis	
2008年	芭芭拉·利斯科夫	Barbara Liskov	编程语言和系统设计的实践与理论
2009年	查尔斯·萨克尔	Charles Thacker	帮助设计、制造第一款现代PC
2010年	莱斯利·瓦伦特	Leslie Valiant	对众多计算理论所做的变革性的贡献
2011年	犹大·伯尔	Judea Pearl	人工智能
2012年	莎菲·戈德瓦塞尔	Shafi Goldwasser	在密码学和复杂理论领域做出创举性工作
	希尔维奥·米卡利	Silvio Micali	
2013年	莱斯利·兰伯特	Leslie Lamport	在提升计算机系统的可靠性及稳定性领域的杰出贡献
2014年	迈克尔·斯通布雷克	Michael Stonebraker	对现代数据库系统底层的概念与实践所做出的基础性贡献
2015年	惠特菲尔德·迪菲	Whitfield Diffie	非对称加密的创始人
	马丁·赫尔曼	Martin Hellman	
2016年	蒂姆·伯纳斯·李	Tim Berners-Lee	万维网的发明者
2017年	约翰·轩尼诗	John Hennessy	开发了RISC微处理器并且让这一概念流行起来的工程
	大卫·帕特森	David Patterson	
2018年	约舒亚·本希奥	Yoshua Bengio	在人工智能深度学习方面的贡献
	杰弗里·欣顿	Geoffrey Hinton	
	扬·莱坎	Yann LeCun	
2019年	帕特里克· 汉拉汗	Patrick M. Hanrahan	对3D计算机图形学的贡献，以及这些技术对电影制作和计算机生成图像（CGI）等应用的革命性影响
	艾德文·卡特姆	Edwin E. Catmull	
2020年	杰弗里·戴维·乌尔曼	Jeffrey David Ullman	创造了全球数百万编程人员使用的工具和教材，推进编程语言实现的基础算法和理论，并在极具影响力的书籍中综述了这些研究成果
	阿尔佛雷德·艾侯	Alfred Vaino Aho	

内容提要

- 一维数组
- 二维数组
- 数组常用算法
  - 查询
  - 插入
  - 删除
  - 排序

数组常用算法：删除

- 例6_11 从整型数组a中删除第一个等于x的元素，如果x不是数组中的元素，则显示：“can not delete x!”。
- 算法：数组空间中的数据只能修改，不能“擦除”。
  - 确定待删除元素的位置
  - 元素从删除位置开始依次前移，覆盖待删除元素
  - 有效元素个数减一。
  - 注意：数组的最后一个元素实际上有两份拷贝
- 数据结构：

  数组a、变量x用于输入要删除的数据

  循环控制变量i，两次循环，查找位置以及移动

删除函数的实现

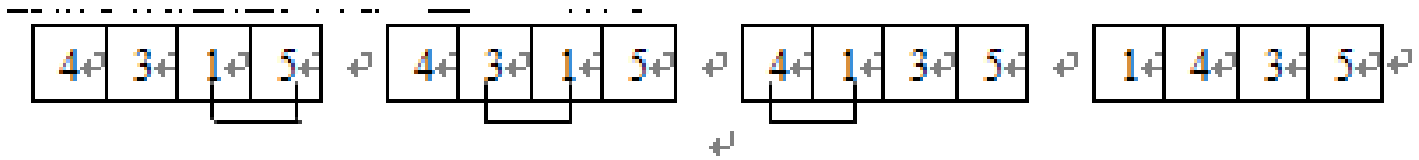
- 接口确定：
  - 功能
  - 入口：数组名、数组长度、待删除的数
  - 返回

```
#define SIZE 5
int main()
 {……
     //初始化array、删除的x
     if(delArray(array,SIZE,x))
         print(array,SIZE-1);
     else
        printf("can not delete x!\n");
    ……
}
```

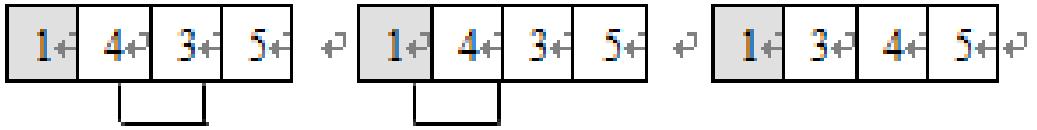
```
int delArray(int a[],int n,int x)
{
    int flag=1; /*标志位*/
   for (i=0;i<n &&a[i]!=x;i++) ;
                                  /*查找x*/
   if (i==n)
      flag=0;
   else
   {
      for (j=i;j<n-1 ;j++)
          a[j]=a[j+1];    /*前移覆盖*/
   }
   return flag;
}
```

数组常用算法：排序

- 例6_12 从键盘上输入n(1≤n≤10)个整数，用冒泡法将元素按从小到大的顺序排序，然后输出排序后元素。
- 冒泡排序的算法思想：在排序过程中对元素进行两两比较，越小的元素会经由交换慢慢“浮”到数组的前面（低下标处），像气泡一样慢慢浮起，由此得名。



(a) 一趟冒泡



(b) 二趟冒泡

冒泡排序算法

- 假设对长度为n的数组进行冒泡排序，算法可以描述如下：
  - 第1趟冒泡：从数组n-1下标的元素到0下标元素遍历，比较相邻元素对，如果后一个元素小于前一个元素，则交换。第一趟结束时，最小元素“浮起”到达0下标位置。
  - 第2趟冒泡：从数组n-1下标的元素到1下标元素遍历（因为0下标的已是最小元素，已经到位，无需再参加比较），比较相邻元素对，如果后一个元素小于前一个元素，则交换。第二趟结束时，本趟最小元素到达1下标位置。
  - 依此类推，最多n-1趟冒泡（n是元素个数），便可以完成排序。

冒泡排序函数的实现

- 接口确定：
  - 功能：小到大排序
  - 入口：数组名、数组长度
  - 返回

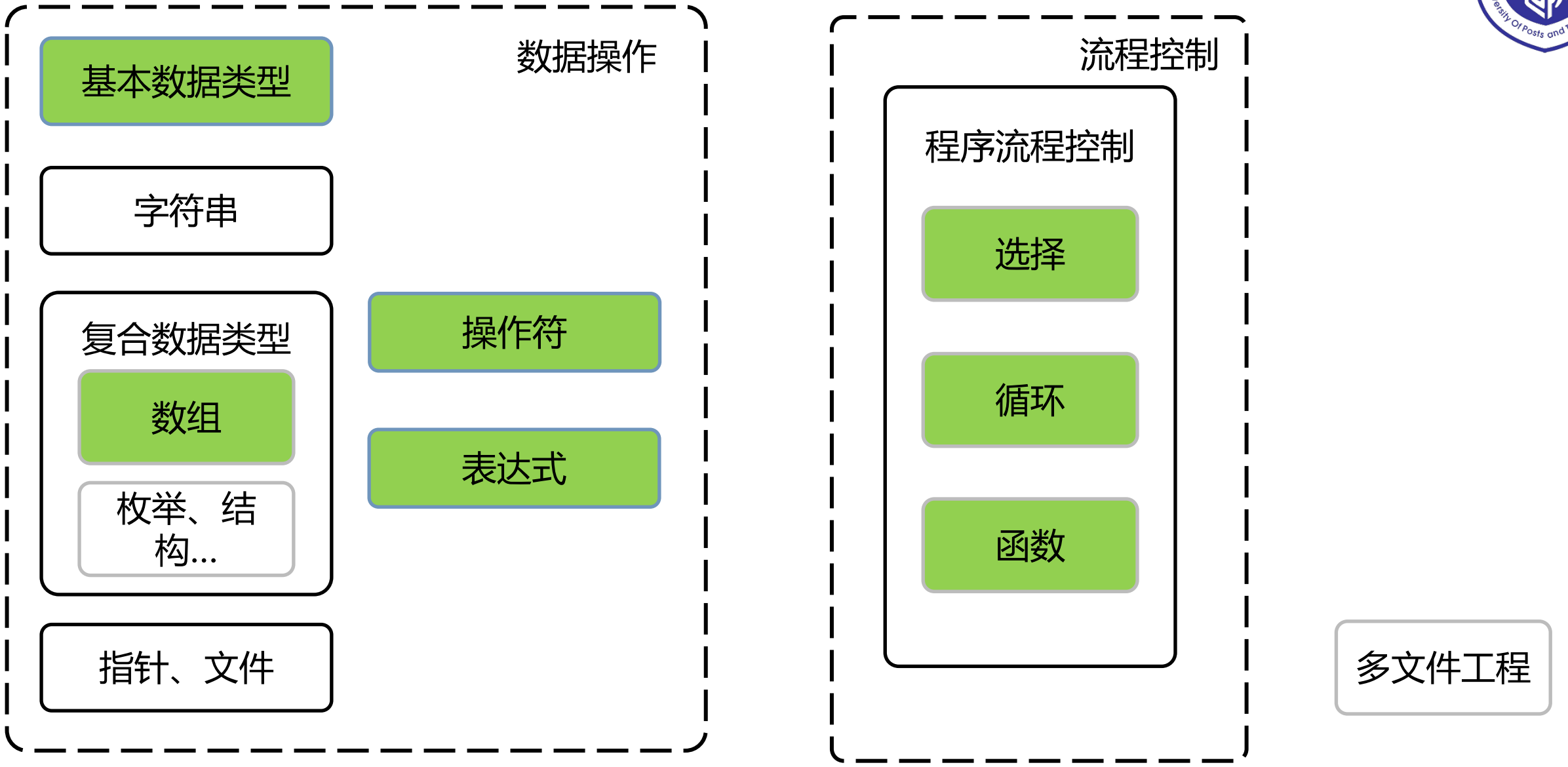
```
#define SIZE 7
int main()
 {……
     //输入array的长度和值
    BubbleSort(array, n);
     …….
}
```

```
void BubbleSort(int a[],int n)
{
    int temp;
    for (i = 0; i < n-1; i++){  /*共进行n-1趟排序*/
       for (j =n-1; j>i ; j--){   /*递减循环，从后往前比较*/
          if (a[j ] < a[j-1]){
             temp=a[j-1];
             a[j-1]=a[j];
             a[j]=temp;
          }
       }
    }
}
```

本章小结

- 一维数组的定义、初始化、元素的存储及访问
- 二维数组的定义、初始化、元素的存储及访问
- 一维数组中的经典算法：查找、插入、删除、排序等
- 二维数组的典型应用 ：矩阵、图形打印
- 一维、二维数组作为形参和实参的基本使用方法

C语言


数据操作
基本数据类型
字符串
复合数据类型
数组
枚举、结构...
指针、文件
操作符
表达式
流程控制
程序流程控制
选择
循环
函数
多文件工程

43