数组、指针与递归

2012.11.16

同类元素的聚会

数组

数组(Array)

- 声明方法

Type arrayName[arraySize]

- 占有一块连续的、可以存放arraySize个相同类型为Type的元素的内存空间。这些元素的编号为0-arraySize-1
- 对数组元素的访问和赋值
  - a[2]，a的第3个元素
  - a[exp]，A的第exp-1个元素，要求：exp在0到arraySize-1之间
  - 赋值：a[i] = exp，要求：i在0到arraySize-1之间

例子

- **int a[9]**

i	0	1	2	3	4	5	6	7	8
a[i]	-45	6	0	72	1543	-89	0	62	-3

已知0<=i<9，a[i] = 3；

a[0] 等于多少?

如果执行赋值语句a[3] = 100，那么a中的内容如何变化？

此时，a[i]等于多少？

数组名字

- int a[9]

- a
- 数组名字数组首元素的内存地址
- 数组名是一个常量，不能被赋值

数组初始化

- **初始化方法一：**
  - 通过循环语句进行初始化
  - for(i=0; i<arraySize; i++)

    a[i] = 0;
- **初始化方法二：**
  - 对部分元素赋值
  - int a[10] = {1, 2, 3, 4}
- **初始化方法三：**
  - int a[] = {1,2,3,4,5,6,7}
    - (What is the size of a?)

示例

- **用数组来求Fibonacci数列问题**

```
#include<iostream>
using namespace std;
int main(){
  int i;
  int f[20]={1,1};            //初始化第0、1个数
  for(i=2;i<20;i++)
      f[i]=f[i-2]+f[i-1];     //求第2～19个数
  for(i=0;i<20;i++){          //输出，每行5个数
       if(i%5==0)
            cout<<endl;
      cout<<f[i];
  }
  return 0;
}
```

多维数组

- int a[2][2]; //a为2行2列的数组；
- 可以把a看作是一维数组，而每个元素又是一个一位数组。
- C/C++语言中多维数组的存放方式：

a[0][0]	a[0][1]	a[1][0]	a[1][1]

- 多维数组的存取
- a[i][j] //数组a的第i行第j列上的元素

二维数组的初始化

按顺序赋值

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

分行赋值

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

部分赋值

- int a[3][4] = {{1},{5},{9,10}};

数组作为函数参数

- 数组元素作为实参，与单个变量一样
- 数组名做参数，形参、实参都应是数组名，类型要一样，传送的是数组**首地址**。对形参数组的改变会**直接影响**到实参数组。

示例

```
#include <iostream>
using namespace std;
void RowSum(int A[][4], int nrow){
    int sum;
    for (int i = 0; i < nrow; i++){
      sum = 0;
      for(int j = 0; j < 4; j++)
        sum += A[i][j];
      A[i][0]=sum;
    }
}

int main(){
    int a[3][4] = {1,2,3,4,2,3,4,5,3,4,5,6}
    print(a);     //a function to show a
    RowSum(a, 3);
    print(a);     //What will a be?
    return 0;
}
```

练习（1）

- 首先读入一个正整数n（n小于100），再读入n个整数并存放到数组中。然后：
  - 求出这n个整数中的最大值和最小值并输出，每个数一行；
  - 将这n个整数倒过来输出。
  - 求出这n个整数中的从大到小排列的第n/2个数。假设我们不修改数组中的整数，也不增加新的数组，如何做？

练习（2）

- 一元多项式的处理：
  - 首先读入一个正整数m（m<10），然后读入m个正整数c_0，c_1，…，c_{m-1}，表示多项式$c_0+c_1x+c_2x^2…+c_{m-1}x^{m-1}$。
  - 再读入一个正整数n（n<10），然后读入n个正整数d_0，d_1，…，d_{n-1}，表示多项式$d_0+d_1x+d_2x^2…+d_{n-1}x^{n-1}$。
- 要求
  - 求这两个多项式的和，从小到大输出这个多项式的系数；
  - 要求同上，但是输出方式改成：按照多项式的各个项的次数，从低到高输出其系数和指数，系数为0时不输出
  - 求这两个多项式的积，从小到大输出积的系数。
  - 完成求积运算后，分别求出当x=1,2,3,4时，多项式的值。
  - 求出第一个多项式除以第二个多项式的商和余式。

练习（3）

- 假设两个$n*n$矩阵A和B，那么A和B的乘积C的定义如下：C的第i行第j列的元素等于：

  注意，矩阵的首行/首列称为第一行/第一列

- 矩阵相乘：
  - 首先读入正整数n（n小于10），表示矩阵是$n*n$的，然后分别读入两个矩阵，每个矩阵n^2个整数；分别存放到两个二维数组中。
  - 求出两个矩阵的乘积，并输出。输出格式如下：
    - 共输出n行，矩阵的每行作为一行输出；
    - 每个整数之间相隔一个空格，但是每一行的末尾没有空格；

指向它的来源

指针

指针

- 声明方式

```
type   *varName
```

- 变量varName的类型为type *，这个变量可以指向一个内存位置，该位置上的值的类型为type。
- 例子
  - `int     *pi = null, iVar;`
  - `int     *pia[100];`
  - `int     (*api)[100];`

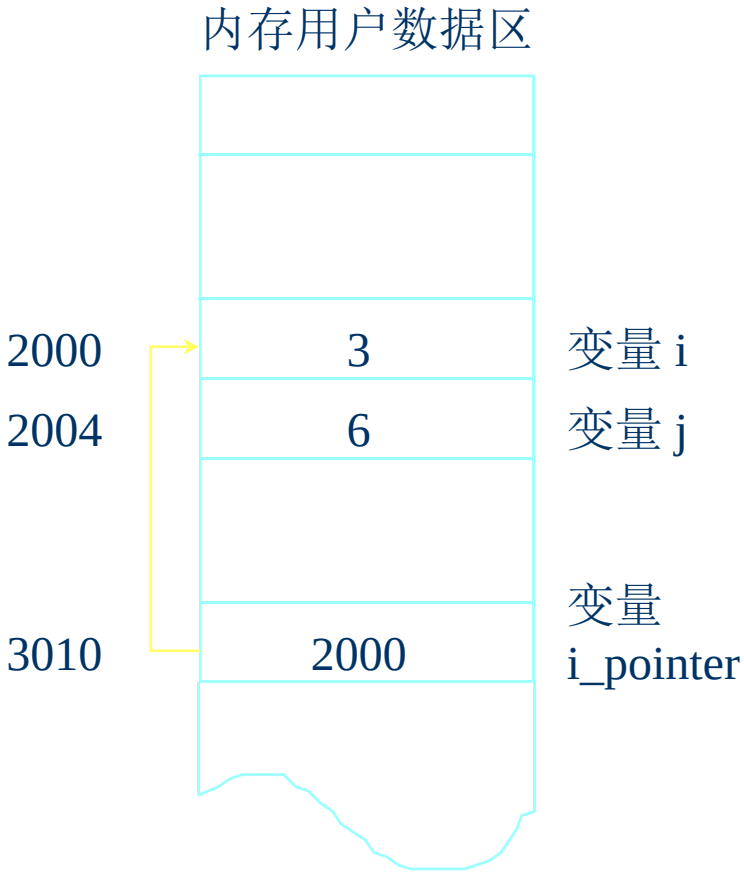
内存地址

- 内存空间的访问方式
  - 通过变量名访问
  - 通过地址访问
- 地址运算符 &
  - int var;
  - 则&var表示变量var在内存中的开始地址

指针变量的概念

概念

- 指针：内存地址，用于直接访问内存单元
- 指针变量：用于存放地址的变量



指针运算符

运算符	含义	例子
*	获取指针所指向的值	int *p; …… y = *p + 3; 必须保证p指向有效的地址
&	获得表达式的左值，即指向这个表达式的值的地址。	&v, &a[i+j], &(i+j)//错误
->	如果p指向的类型为struct，p->m1表示p所指向的结构型值的成员m1。	p->m1
++, --	令指针指向后一个元素/上一个元素	p++; p-- p应该指向数组中的某个元素
>,<,>=,<=,==	比较指针的大小	

数组与指针

- 假设有元素类型为T的数组类型的变量a，a在表达式中看作T *类型的值，但是没有左值。
- 例子：

```
int a[100], *p;
p = a;
x = *p;
p++;
y = *p;
```

指针的使用（1）

- 指针可以指向某个变量或者其内部的值。
  - int *p, v, a[100];
  - p = &v;… *p = 3;//改变了v中的值；
  - p = a;　　//p指向a的第0个元素；
  - *p = 5;　　//将a的第0个元素赋值为5；
  - p = a + i;　　//p指向a的第*i*个元素；
  - p ++;　　//p指向第*i*+1个元素。
- 指针的别名
  - int *p,*q;
  - … ……
  - *p = 3; *q = 5;
  - 此时*p等于多少？

指针的使用（2）

- 数据存放的区域
  - 静态区：全局变量
  - 栈区：局部变量
  - 堆区：动态申请的数据对象
- 动态申请/释放：malloc/free
  - Malloc在堆区中申请一块空间，返回指向这块空间的指针；
  - 这块空间一值有效，直到被释放为止
  - Free释放相应的空间（必须使用malloc返回的指针）

空间的动态申请释放

动态申请/释放的要求

- 申请得到的空间在被释放之前一直有效；
- 不需要使用的空间必须被释放
  - 否则内存泄露（当有持续运行的程序时，未释放的空间可能会逐渐占用大量的资源）
- 尚需使用的空间不能释放
  - 否则出现悬空指针
- free的参数必须是前面malloc返回的指针。

例子

- 读入一个整数n；申请一块空间来存放接下来读入的n个整数；计算这些整数的和；输出。

```
int i,n,*p;
cin >> n;
int *p = (int *)malloc(sizeof(int) * n)
      //(int *)为类型转换，malloc返回void *，
      //sizeof()返回该类型的内存大小；
for(i=0; i<n; i++)
      cin >> p[i];
      //注意p[i]这个用法，它和数组的用法一致
for(i=0; i<n; i++)
      sum += p[i];
cout << sum;
free(p);//释放空间！
```

练习（5）

- 首先读入两个整数m、n，然后读入$m*n$个整数并保存起来，作为矩阵M。要求：
  - 首先申请一个长度为m，元素类型为int *的数组；
  - 再为这个数组中的每个元素申请一个长度为n，元素类型为int的动态数组。
- 接着读入n个整数，作为一个向量p；同样存放在一个动态数组中。
- 计算M乘以p得到的向量并输出。
  - 结果向量包括m个分量，其值为
- 最后释放申请到的所有空间！

数组名、指针

关系

- 数组名是数组的第一个元素在内存中的首地址
- 数组名在表达式中被自动转换为一个指向数组第一个元素的**指针常量**。数组名所放的地址是不可改变的。
- C++语言的下标运算符是以指针为操作数的，因此a[i]会被编译系统解释为*(a+i)，即为a所指元素向后的第i个元素

自己调用自己?

递归

递归

- 如果一个函数（直接或间接）调用了自身，则该函数是递归的。
- 如果一个问题可以分成一个或多个子问题，而某些子问题和原来问题类似，只是规模不同，那么就使用递归函数。
- 递归函数通常比较容易理解和书写：
  - 对于一些数学问题，递归函数通常和问题的定义一致
  - 但是效率较差。
- 注意：递归函数的每次被调用都有一个独立的空间存放局部变量！

递归程序

递归的定义

- f(n) = g(n, f(n-1)) n>0
- f(0) = a
- 其中，f(0)=a称为递归边界

一般适用的场合

- 问题的定义是递归的，如Fibonacci数列
- 数据之间的关系是递归的，如树，图等

例子（1）

- 阶乘的定义（要求x>=0）

$$x! = \begin{cases} x \times (x-1)! & x > 1 \\ 1 & x = 0 \end{cases}$$

- 相应的程序：

```
int F(int x)
{
        if(x==0)
                return 1;
        return x*F(x-1);
}
```

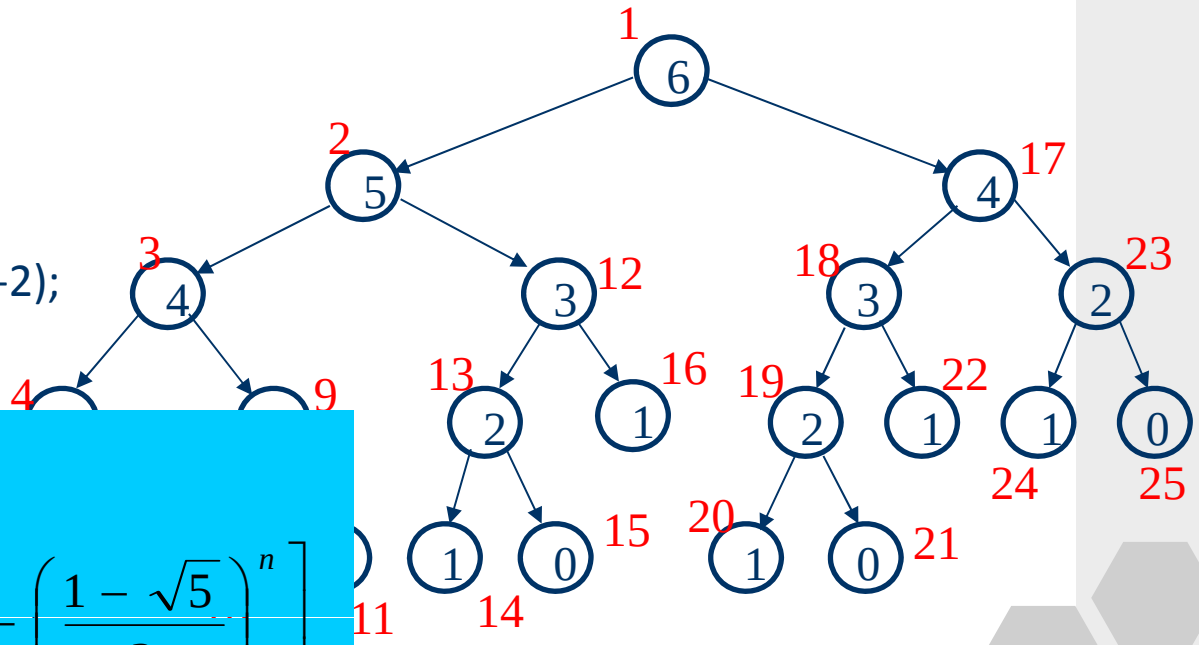
例子（2）

❖ 菲波那契数列的定义

$$Fib(x) = \begin{cases} 1 & x = 1, 2 \\ Fib(x-1) + Fib(x-2) & \text{otherwise} \end{cases}$$

❖ 相应的程序

```
int Fib(int x)
{
    if(x<=1)  return 1;
    return Fib(x-1)+Fib(x-2);
}
```



For your reference

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

例子（3）

- 前缀表达式
  - 单个整数是前缀表达式
  - 二目运算符 前缀表达式 前缀表达式
  - 单目运算符 前缀表达式
- 例如：+ 3 * 4 5　　等于　3+(4*5)
- 递归处理程序：

```
int Exp()
{
    cin >> ch;
    if(ch>='0' && ch <= '9')
            return ch – '0';
     else if(ch == '+')
            return Exp() + Exp();
     else if(ch == '*')
            return Exp() * Exp();
            …
}
```

用递归函数帮助思维

- 假设我们需要求出某个问题Q的解，相应的参数为X。
  - 首先考虑X等于很简单的值时，Q的解法
  - 然后考虑Q(X)的解能不能分解成为更加简单的Q(X’),Q(X’’)…的解。
- 例子
  - 假设有一百层楼，2个完全一致的坚固的鸡蛋。
  - 求一种方法确定最低从哪一层楼扔下就会被摔碎。要求最坏情况下的实验次数最低。

用递归输出全排列

```
void Swap(int& a, int& b) {                    // 交换a和b
    int temp = a;
    a = b;
    b = temp;
}
void Perm(int list[], int k, int m) {          //生成list [k：m ]的所有排列方式
    int i;
    if (k == m) {                              //输出一个排列方式
        for (i = 0; i <= m; i++)
            cout<<list[i];
        cout<<'\n';
    }
    else                                        // list[k：m ]有多个排列方式
        for (i=k; i <= m; i++) {
            Swap (list[k], list[i]);
            Perm (list, k+1, m);
            Swap (list [k], list [i]);
        }
}
```

练习

- 带变量的表达式的求值
- 输入
  - 第一行是一个字符串，该字符串是一个前缀表达式，包含+，-，*，/四个运算符（/表示整数除法）；它的运算分量可能是0-9的整数，也可能是A，B，C，D之一，他们表示变量。
  - 接下来，每行四个整数，分别表示A，B，C，D的值。
  - 四个-1表示结束，不处理。
- 输出：
  - 对于第二行开始的每一行a b c d，输出当A，B，C，D分别取值a，b，c，d时的值。
- 例子输入：
  - -+*4A/BCD
  - 1 1 1 1
  - 1 2 3 4
  - -1 -1 -1 -1
- 输出
  - 4
  - 0

Q&A

Thank You !