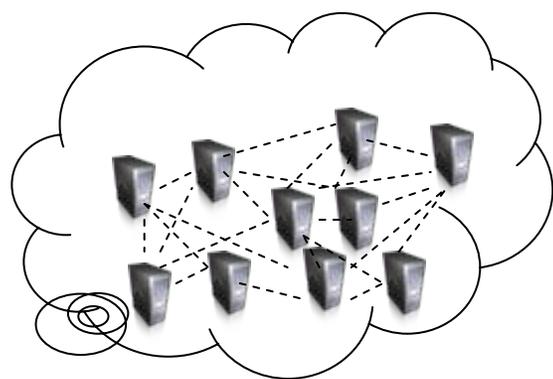


# 厦门大学非计算机专业本科生公共课 (2012-2013第2学期)



## 《C语言程序设计》

### 第8章 数组

林子雨

厦门大学计算机科学系

E-mail: [ziyulin@xmu.edu.cn](mailto:ziyulin@xmu.edu.cn)

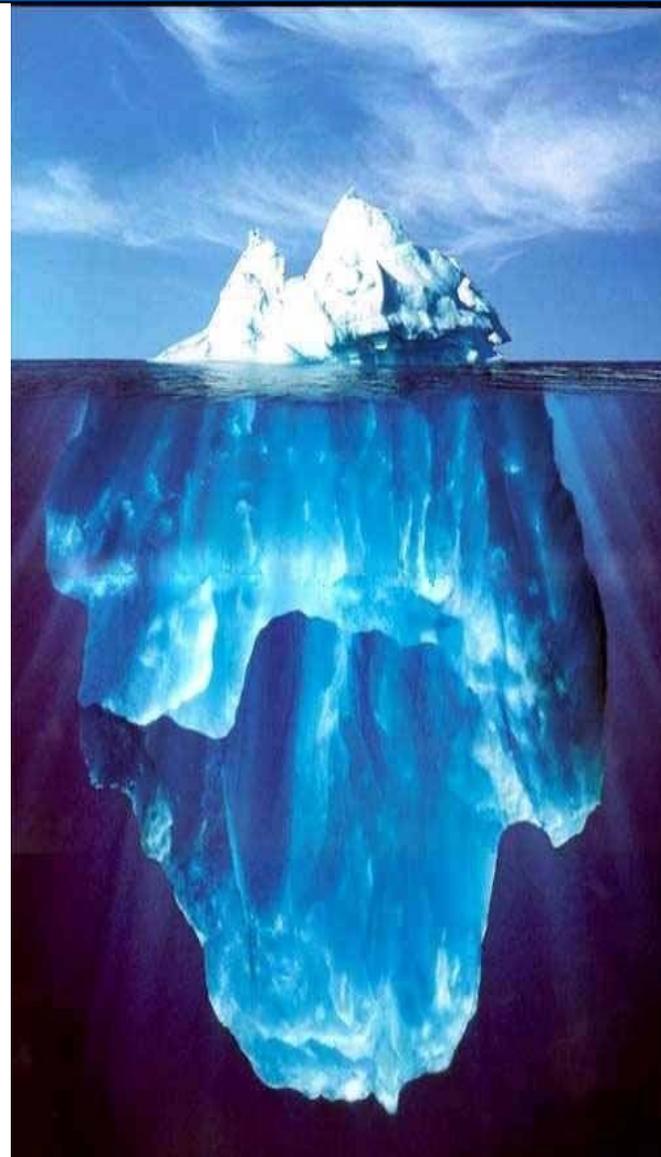
个人主页: <http://www.cs.xmu.edu.cn/linziyu> ▶▶





# 课程提要

- 第一章 绪论
- 第二章 C语言基础
- 第三章 结构化程序设计
- 第四章 选择结构
- 第五章 循环结构程序设计
- 第六章 函数
- 第七章 编译预处理
- **第八章 数组**
- 第九章 结构体、共用体和枚举类型
- 第十章 指针





# 第8章 数组

## 8.1 一维数组

## 8.2 多维数组(\*)

## 8.3 字符串





# 8.1 一维数组

- 8.1.1 一维数组的定义
- 8.1.2 一维数组的引用
- 8.1.3 一维数组的初始化
- 8.1.4 一维数组应用举例





## 8.1.1 一维数组的定义

- 定义一个数组应指出该数组的名字叫什么、包含几个下标、包含几个元素和各元素的数据类型。
- 只包含一个下标的数组称为“一维数组”，定义格式如下：  
**类型说明符 数组名[常量表达式];**  
例如：`int array[5];`
- 说明：
  - (1) “数组名”是标识符，应该遵循标识符构成规则和作用域规则，即在同一个作用域内，两个数组不能同名，数组和简单变量也不能同名。
  - (2) “类型说明符”是数组中各元素的类型，可以是简单类型，也可以是已经定义的构造类型。
  - (3) “数组名”后的方括号个数表示数组的维数，一个方括号表示定义的是一维数组。
  - (4) 方括号中的“常量表达式”表示数组的元素个数，即数组长度。表达式只可包含常量，不能包含变量。**C语言不允许动态定义数组大小。**
  - (5) 数组也是变量，在一个语句中可以同时定义多个数组，数组也可以和简单变量一同定义。例如：`int a,array[5];`





## 8.1.2 一维数组的引用

- 数组必须先定义后引用。数组定义之后，系统为该数组在内存分配一块连续的存储空间，数组中各元素在内存中依次存放。即数组中每个元素有一个序号（下标）与存储空间的起始位置相关联，序号从零开始，序号最大值为数组长度减1。
- 若要引用数组中的某个元素应使用表达式：

**数组名[下标]**

其中“下标”是一个整型表达式，其值表示某个元素在数组中的序号。





## 8.1.2 一维数组的引用

- **例8.1.1**用于计算某学生三门课程的平均成绩。程序中用数组 `grade` 存放学生各课程成绩，用简单变量 `avg` 存储平均成绩。

```
#include <stdio.h>
void main()
{
    float grade[3], avg;
    grade[0]=90; grade[1]=75; grade[2]=85;
    avg=(grade[0]+grade[1]+grade[2])/3;
    printf("avg=\"%f\n", avg );
}
```





## 8.1.2 一维数组的引用

- **例8.1.2:** 从键盘输入三个整数，然后按相反的顺序输出。程序清单如下。

```
#include<stdio.h>
void main()
{
    int i,a[3];
    printf("请输入3个整数: \n");
    for(i=0;i<3;i++)
        scanf("%d",&a[i]);
    for(i=2;i>=0;i--)
        printf("%d\t",a[i]);
    printf("\n");
}
```





## 8.1.2 一维数组的引用

- **引用数组时应注意:**
- (1)C系统规定不能直接引用整个数组，只能引用数组元素。例如数组输出时，不能写成“`printf(“%d\t”,a)`”，应分别输出数组中各元素。因此，数组的引用通常和循环结构结合在一起。
- (2)引用数组元素时，下标应该是整型表达式，表达式中可包含变量。如果下标值非整型，系统将自动转换为整型。
- (3)C编译系统没有对数组下标进行越界检查。当引用数组元素时，下标值不在数组定义的下标范围内，系统并没有产生编译错误，但运行时可能会引用到不属于本数组的其他存储空间，这就可能会破坏其他变量的数据，或破坏目标代码甚至破坏系统程序，从而引起运行错误。
- (4)数组定义和数组元素引用的形式相似，但意义不同。在数组定义中，方括号中的常量表达式的值代表数组长度；而数组元素引用时，方括号中的表达式可以包含变量，表达式的值是下标，代表该元素在数组中的位置。





## 8.1.3 一维数组的初始化

- 在定义数组时，如果数组元素的初值已经确定，可以对数组进行初始化，以提高程序的执行效率。
- 初始化数组时，数组元素的初始值用花括号括起来，各常量之间用逗号分隔，形成常量表。常量列表的每一项和数组中各元素依次对应。例如：

```
float grade[3] = {90.0,75.0,85.0};
```

### 初始化数组应注意：

- (1)如果数组没有初始化，系统会用默认值对它初始化，其规则和简单变量相同，即外部变量或静态变量赋值为0（数组各元素均为0），自动变量赋值随机。
- (2)初始化数组时，初始值的个数可以比数组元素的个数少，这时，未提供初始值的元素被置为0.例如：

```
int a[10] = {1,2,3,4,5};
```

数组a在内存的存储状态如下：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
1	2	3	4	5	0	0	0	0	0





## 8.1.3 一维数组的初始化

- **初始化数组应注意:**
- (3)当对全部数组元素赋初值时, 可以不指定数组长度, 但数组名后面的方括号不能省略。例如:  

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

可以写成:  

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

但不能写成:  

```
int a={1,2,3,4,5};
```
- (4)初始化数组时, 初始值的个数不能多于数组元素个数, 否则错误。  
例如: 

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

//错误
- (5)初始化是在编译时完成的, 而赋值运算是在程序运行时进行的, 要注意二者的区别。例如**不能**把:  

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

写成  

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

或写成  

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





## 8.1.4 一维数组应用举例

- **例8.1.3:** 用数组求Fibonacci数列的前20项 (1, 1, 2, 3, 5, 8.....), 程序清单如下。

```
#include<stdio.h>
void main()
{
    int i,f[20]={1,1};
    //定义并初始化数组f, 使f[0]=1, f[1]=1, 其余元素为0
    for(i=2;i<20;i++)
        f[i]=f[i-1]+f[i-2];    //数列的第i项等于第i-1项和第i-2项之和
    for(i=0;i<20;i++)
    {
        printf("%d\t",f[i]);
        if((i+1)%5==0)    //每输出5个数据后换行
            printf("\n");
    }
}
```





## 8.1.4 一维数组应用举例

- **例8.1.4:** 由键盘输入一个班级的学生考试成绩，统计出本班的人数和各分数段的人数。以10分为一个分数段，即0~9,10~19,...,90~99,100共11个分数段，程序清单如下。

```
#include<stdio.h>
void main()
{
    int i,score ,n=0; //score为某学生成绩；n为班级人数，初始化为0
    int num[11]={0}; //数组num为各分数段人数，初始化为0
    printf("输入各学生的考试成绩，当学生成绩输入结束时请输入-1\n");
    while(1)
    {
        scanf("%d",&score);
        if(score== -1) break ;
        //-1称为“结束标志”，当输入-1时，表示全部数据输入完毕，结束循环
        n++; //统计班级人数
        num[score/10]++; //统计各分数段人数
    }
    printf("本班考试人数为： %d\n",n);
    printf("分数段:\t人数\n");
    for(i=0;i<10;i++)
        printf("%d~%d:\t%d\n",i*10,i*10+9,num[i]);
    printf("100:\t%d\n",num[10]);
}
```





## 8.1.4 一维数组应用举例

- **例8.1.5:** 用“筛法”求500以内的所有素数，程序清单如下。

```
#include <stdio.h>
#include<math.h>
#define N 500
void main()
{
    int i,j,n0,a[N];
    for(i=2;i<N;i++) a[i]=1;//a各元素赋初值 1，不能使用int a[N]={1};为什么？
    n0=sqrt(N);//为什么开根号？
    for(j=2;j<=n0;j++)
    {
        if(a[j]==0) continue;
        for(i=j+1;i<N;i++)
            if(i%j==0) //如果 j能除尽 i, 则i就不是素数
                a[i]=0; //置i不是素数标志 (a[i]=0)
    }
    for(i=2;i<N;i++)
        if(a[i])//如果a[i]的值仍然是1，则i为素数，输出i
            printf("%d\t",i);
    printf("\n");
}
```





## 8.1.4 一维数组应用举例

- **例8.1.6:** 输入10个整数，输出其中的最大者和最小者，程序清单如下。

```
#include <stdio.h>
#define N 10
void main()
{
    int max,min,a[N],i;
    printf("Input numbers:\n");
    for(i=0;i<N;i++)
        scanf("%d",&a[i]);
    max=min=a[0];
    for(i=1;i<N;i++)
    {
        if(a[i]>max) max=a[i];
        if(a[i]<min) min=a[i];
    }
    printf("max=%d\n",max);
    printf("min=%d\n",min);
}
```





## 8.1.4 一维数组应用举例

- **例8.1.7:** 用冒泡法对10个整数排序，程序清单如下。

```
#include <stdio.h>
void main()
{
    int i,j,t,n,a[100];
    printf("n=");
    scanf("%d",&n);
    printf("Input numbers:\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=1;i<n;i++) //比较和交换要反复多次
        for(j=0;j<n-i;j++) //从左到右依次比较两个相邻的元素
            if(a[j]>a[j+1]) //如果左大右小，则彼此的值交换
                {t=a[j];a[j]=a[j+1];a[j+1]=t;}
    printf("The sorted numbers:\n");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
    printf("\n");
}
```





## 8.2 多维数组(\*)

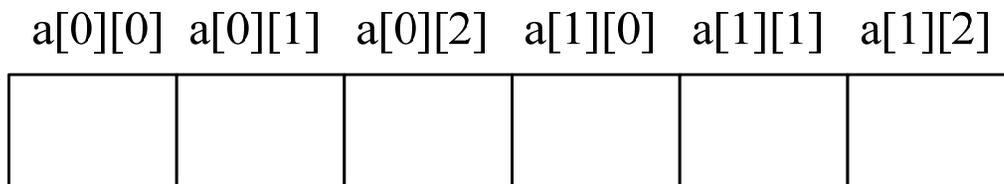
- 8.2.1 二维数组的定义和引用
- 8.2.2 二维数组的初始化
- 8.2.3 二维数组应用举例





## 8.2.1 二维数组的定义和引用

- 二维数组的定义形式：  
类型说明符 数组名[常量表达式1][常量表达式2];
- [常量表达式1]是第一维的长度， [常量表达式2]是第二维的长度。  
例如： `int a[2][3];`  
该二维数组包含2行3列，6个元素分别是：  
`a[0][0]`、`a[0][1]`、`a[0][2]`、`a[1][0]`、`a[1][1]`、`a[1][2]`
- 二维数组逻辑上是二维的，而内存单元是按一维方式组织，连续编址的。



- 二维数组的引用形式为：  
数组名[下标1][下标2]
- 注意事项：
  - (1) 两个下标必须分别加方括号。不能把`a[0][1]`写成`a[0,1]`
  - (2) 行下标和列下标不能交换。`a[0][1]`和`a[1][0]`是两个不同数组元素
  - (3) `a[0]`虽然是二维数组`a`的元素，但`a[0]`是一维数组，不能直接引用





## 8.2.2 二维数组的初始化

- 二维数组初始化时，初始值按行的顺序排列，每行一组，每组加花括号。例如：  
`int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};`
- 这种初始化的方式，把一个二维数组看成是一个一维数组，其数组元素又是一个一维数组。
- 和对一维数组赋初值一样，也可以只对二维数组的部分元素赋初值，没有提供初值的元素，其初值为0.例如：  
`int a[3][4]={{1},{2,3},{4,5,6}};`
- 初始化后，数组a各元素的值分别为：  
`a[0][0]=1 a[0][1]=0 a[0][2]=0 a[0][3]=0`  
`a[1][0]=2 a[1][1]=3 a[1][2]=0 a[1][3]=0`  
`a[2][0]=4 a[2][1]=5 a[2][2]=6 a[2][3]=0`
- 如果为全部元素提供初值，内层花括号可以省略。即把所有初始数据按行为序写在一个花括号内；如果为全部元素都提供初值，定义数组时可以省略第一维的长度。下面几种初始化方法等价：  
`int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};`  
`int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};`  
`int a[][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};`  
`int a[][4]={1,2,3,4,5,6,7,8,9,10,11,12};`
- **注意**：如果只对二维数组的部分元素赋初值，内层花括号不能省略，数组第一维的长度也不能省略。





## 8.2.3 二维数组应用举例

例8.2.1: 定义一个5行5列的二维数组a (矩阵), 数组元素为二位数整数 (10~99), 其值由系统提供的随机函数rand产生, 输出该矩阵; 把矩阵a转置 (行列对换), 然后输出转置后的矩阵a。程序清单如下。

```

#include <stdio.h>
#include<stdlib.h>
#include<time.h>
#define Max 99
#define Min 10
void main()
{
    int i,j,a[5][5],temp;    srand(time(0)); //初始化随机函数
    for (i=0;i<5;i++)      //数组元素的值由随机函数rand()产生
        for(j=0;j<5;j++)
            a[i][j]=Min+rand()/32768.0*(Max-Min+1);
    printf("a矩阵的值如下: \n");
    for (i=0;i<5;i++)
        {
            for(j=0;j<5;j++) printf("%d\t",a[i][j]);
            printf("\n");
        }
    for (i=0;i<5;i++)      //矩阵转置 (行列对换)
        for(j=i+1;j<5;j++) //注意循环变量j的初值
            {temp=a[i][j]; a[i][j]=a[j][i]; a[j][i]=temp;}
    printf("a矩阵转置后的值如下: \n");
    for (i=0;i<5;i++)
        {
            for(j=0;j<5;j++) printf("%d\t",a[i][j]);
            printf("\n");
        }
}

```

矩阵的值:

10 60 27 82 62

53 41 90 84 77

25 87 73 56 37

11 18 42 23 24

98 50 20 10 10

矩阵转置后的值:

10 53 25 11 98

60 41 87 18 50

27 90 73 42 20

82 84 56 23 10

62 77 37 24 10





## 8.2.3 二维数组应用举例

例8.2.2: 求二维数组中最大元素的值及在数组的位置, 程序清单如下。

```
#include <stdio.h>
#include<stdlib.h>
void main()
{
    int i,j,a[5][5],max,maxi,maxj;
    srand(100);
    for (i=0;i<5;i++)
        for(j=0;j<5;j++)
            a[i][j]=Min+rand()/32768.0*(Max-Min+1);

    printf("a矩阵的值如下: \n");
    for (i=0;i<5;i++)
        {for(j=0;j<5;j++)
            printf("%d\t",a[i][j]);
        printf("\n");
        }
    max=a[0][0];maxi=0;maxj=0;
    for (i=0;i<5;i++)
        for(j=0;j<5;j++)
            if(max<a[i][j])
                {max=a[i][j];maxi=i;maxj=j;}
    printf("a矩阵的最大元素是: a[%d][%d]=%d\n",maxi,maxj,max);
}
```





## 8.2.3 二维数组应用举例

例8.2.3: 打印扬辉三角。扬辉三角形满足以下规则：第一行有一个元素，第n行有n个元素；第一行元素值为1，以后各行首尾元素值为1，中间的第k元素值等于上一行第k-1元素和第k元素之和。

```
#include <stdio.h>
#define N 6
void main()
{
    int i,j,a[N][N]={0};           //二维数组初始化为0
    for (i=0;i<N;i++)             //二维数组的第0列元素和对角线元素置1
        a[i][0]=a[i][i]=1;
    for(i=1;i<N;i++)              //其他元素的值等于其左上元素与正上元素之和
        for(j=1;j<N;j++)
            a[i][j]=a[i-1][j-1]+a[i-1][j];
    printf("a矩阵的值如下: \n");
    for (i=0;i<N;i++)            //输出矩阵的下三角元素
        {for(j=0;j<=i;j++)
            printf("%d\t",a[i][j]);
        printf("\n");
    }
}
```

	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1





## 8.3 字符串

- 8.3.1 字符型数组
- 8.3.2 字符串
- 8.3.3 字符串处理函数(\*)





## 8.3.1 字符型数组

- 数组元素类型为整型的数组，称为“整型数组”。
- 数组元素类型为字符型的数组称为“字符型数组”，简称“字符数组”。
- 定义字符数组并对字符数组初始化，用户提供的初始值列表必须是字符常量表。例如：

```
#include<stdio.h>
void main()
{
    char i,c[5]={'H','e','l','l','o'};
    for(i=0;i<5;i++) printf("%c",c[i]);
    printf("\n");
}
```





## 8.3.1 字符型数组

- 定义字符数组c也可用省略数组长度的方式:

```
char c[] = {'H','e','l','l','o'};
```

- 定义字符数组并初始化, 当提供的字符常量个数少于数组元素个数时, 其余元素定为空字符'\0', 空字符即ASCII码为0的字符。例如:
- `char c[10] = {'H','e','l','l','o'};`
- 则数组c的值如下:

c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]
H	e	l	l	o	\0	\0	\0	\0	\0

**注意:** 空字符不同于空格字符, 空格字符的ASCII码为32, 输出时是一个空格; 而空字符没有任何输出。





## 8.3.1 字符型数组

- 和一般数组一样，可以引用字符数组元素，如`c[5]='!`，但不能引用字符数组，例如`c="Hello!"`是错误的。
- 在C语言中，字符型和整型是相通的，所以上面的定义也可以改成：

```
int c[5]={'H','e','l','l','o'};
```

但是，用整型空间存储字符型数据会造成空间的浪费。





## 8.3.2 字符串

- 字符串就是一串字符，是程序设计中经常使用的数据。在C语言中，并没有专门设置“字符串类型”，而是将字符串作为字符数组来处理。
- 字符数组是一种构造型数据，不能整体处理。而在实际应用中，经常需要把字符串当作基本类型的数据，需要对字符串进行整体处理。为此，C语言对字符数组进行了补充定义。

- 8.3.2.1 字符串结束符
- 8.3.2.2 字符数组初始化
- 8.3.2.3 字符数组的输入和输出





## 8.3.2.1 字符串结束符

- 系统在存储字符串时，自动在字符串后面加入空字符（'\0'），作为字符串结束符。例如：字符串“Hello”，系统在内存中的实际存储为：





## 8.3.2.2 字符数组初始化

- 可以用字符串对字符数组初始化，例如下面几种字符数组的定义方式和初始化方式是等价的：

```
char c[6]={'H','e','l','l','o','\0'};
```

```
char c[]={ 'H','e','l','l','o','\0'}; //数组长度可以省略
```

```
char c[6]={'H','e','l','l','o'}; //初始值个数小于数组长度，c[5]初始化为'\0'
```

```
char c[6]="Hello";
```

```
char c[6]="Hello"; //花括号可以省略，问：char c[5]="Hello"是否可以？
```

```
char c[]="Hello"; //数组长度可以省略
```

- **说明：**

(1) 存储字符串的字符数组的长度必须大于字符串长度。

(2) `char c[]="Hello"`和`char c[]={ 'H','e','l','l','o','\0'}`等价，但是：

`char c[]="Hello"`和`char c[]={ 'H','e','l','l','o'}`不等价。

(3) 如果不把字符数组看作是字符串，字符数组中不一定要包含值为'\0'的元素。





## 8.3.2.3 字符数组的输入和输出

- 可以使用两种方式对字符数组进行输入输出：
  - 使用%c逐个字符输入或输出
  - 使用格式符%s对字符数组整体输入或输出

- **字符数组的输出：**

下面是两种等价的输出方法：

```
//以字符方式输出
```

```
i=0;
```

```
while(c[i]) printf(“%c”,c[i++]);
```

```
//以字符串方式输出
```

```
printf(“%s”,c);
```

可以看出，第二种方式比较简练。执行”printf(“%s”,c)”时，输出从c[0]到结束符'\0'前的所有字符。当字符数组c包含多个结束符时，第一个结束符后的字符不输出；当字符数组c不包括结束符时，输出c的全部字符及数组c之后的字符，知道遇到一个'\0'时停止。





## 8.3.2.3 字符数组的输入和输出

- 字符数组的输入
- 执行scanf(“%s”,c)时，字符数组c从键盘读取字符串，直到遇到空格或回车键为止，同时系统会自动在接受到的字符后面加一个‘\0’，即输入时字符串不能包含空格字符。例如：

```
#include <stdio.h>
void main()
{
    char c[5];
    scanf(“%s”,c); //字符串变量c之前可加&运算符或不加
    printf(“%s\n”,c);
}
```

- 键盘输入“abc 123”，屏幕输出“abc”；键盘输入“abcdefg”，屏幕输出“abcdefg”。
- 当数组c接受的字符串长度超过数组长度时，不会产生错误，但字符串会占用其他存储空间，肯能会引起运行错误，应特别注意。
- 使用scanf函数可以一次输入多个字符串，各个输入串之间可以空格或回车符分隔。
- 在scanf(“%s”,c)中，输入项是字符数组名。输入项前可以不加运算符&（也可以加），因为数组名c本身就是字符数组的起始地址。





# 附件：课程教材（2012-2013第2学期）

- 《C语言程序设计（第2版）》
- 清华大学出版社，黄保和，江弋 编著
- 版次：2011年10月第2版
- ISBN:978-7-302-26972-4
- 定价：35元



# 附件：课程和班级网站（2012-2013第2学期）

- 课程介绍网站：

<http://dbllab.xmu.edu.cn/node/124>

- 班级网站：

<http://dbllab.xmu.edu.cn/node/347>



# 附件：课程教师和助教（2012-2013第2学期）

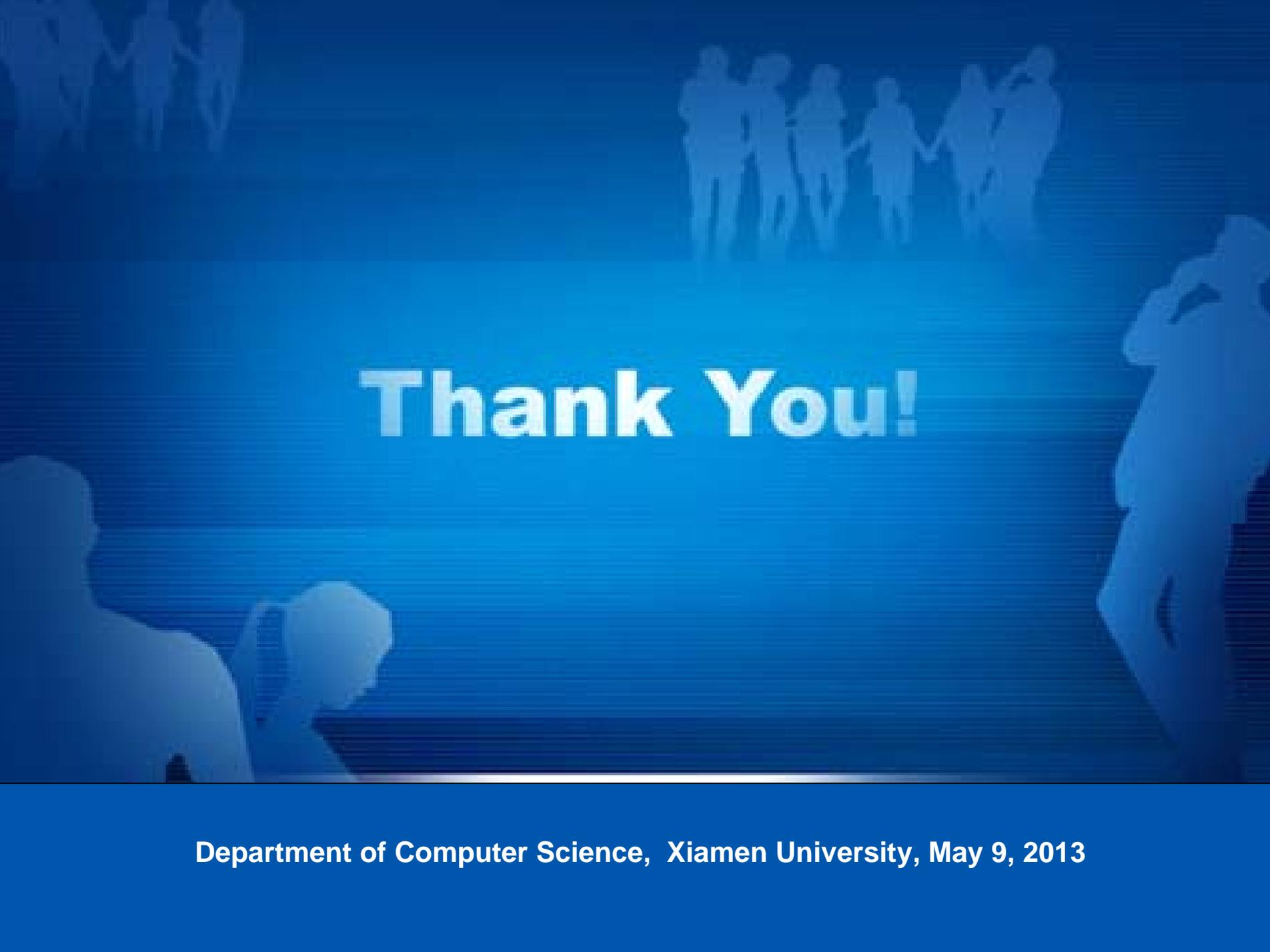


## 主讲教师：林子雨

单位：厦门大学信息科学与技术学院计算机科学系  
办公地点：福建省厦门市思明区厦门大学海韵园  
E-mail: ziyulin@xmu.edu.cn  
个人主页: <http://www.cs.xmu.edu.cn/linziyu>

## 助教：刘颖杰

单位：厦门大学计算机科学系2012级硕士研究生  
E-mail: 376339705@qq.com  
手机：18020761782

The background is a solid blue color with a gradient. It features several faint, light-blue silhouettes of people. At the top, there are two groups of people: one on the left and one on the right, both appearing to be in conversation or holding hands. On the right side, there is a larger silhouette of a person standing and talking on a mobile phone. In the bottom left corner, there are two more silhouettes of people, one larger and one smaller, also appearing to be in conversation.

# Thank You!

Department of Computer Science, Xiamen University, May 9, 2013