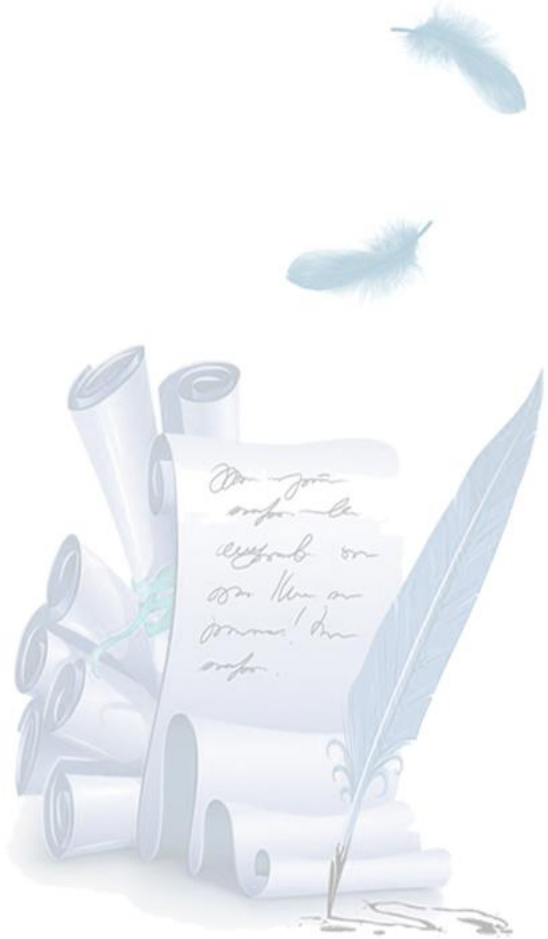
面向对象方法与C++程序设计

第1章　面向对象概述

大连理工大学
主讲人-赵小薇

数据结构与数据访问





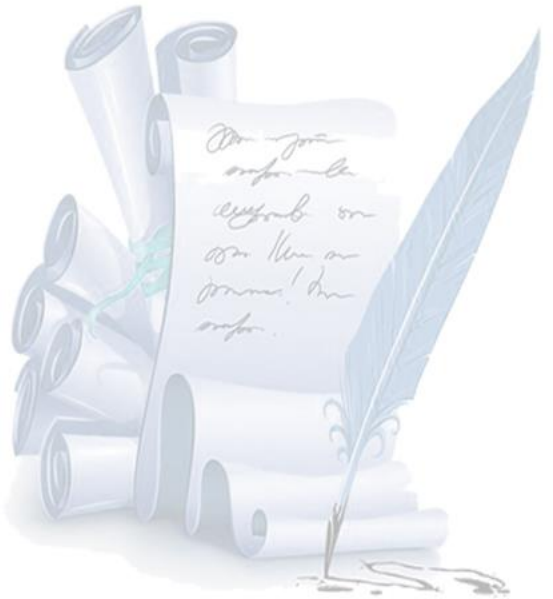
指针变量定义



- 指针变量是用来存放地址的，指针变量定义的一般形式为：

  类型说明符 *变量名;

- 例如： int *p1;
- 对指针变量的定义包括三个内容：
  - 指针类型说明符，*表示这是一个指针变量；
  - 指针变量名，p1为指针变量名；
  - 指针所指向的变量的数据类型，int为指针变量所指向的变量的数据类型，说明p1只能储存整型变量的地址。



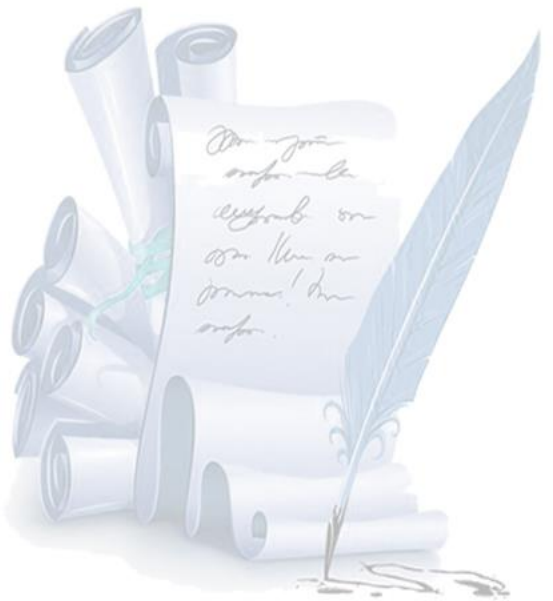
指针变量引用



- 指针变量使用之前必须赋予具体的值。否则将造成系统混乱，甚至死机。
  - & 取地址运算符。
  - * 指针运算符（或称“间接访问” 运算符）

```
int i;
*p1= & i;    //此处*是类型说明符
int i2=*p1+1;  // 此处*代表间接访问运算符
```

- 完全等价于： int i2=i+1;



指针类型



- 指向一维数组的指针

  类型说明符 (*指针变量名)[数组长度];

- 指针数组

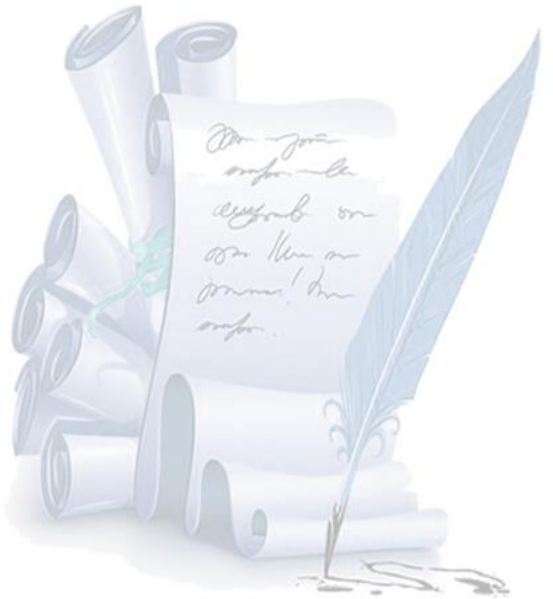
  类型说明符 *数组名[数组长度];

- 指向指针的指针

  类型说明符 **变量名;

- 指向函数的指针

  类型说明符 (*指针变量名)(参数表);



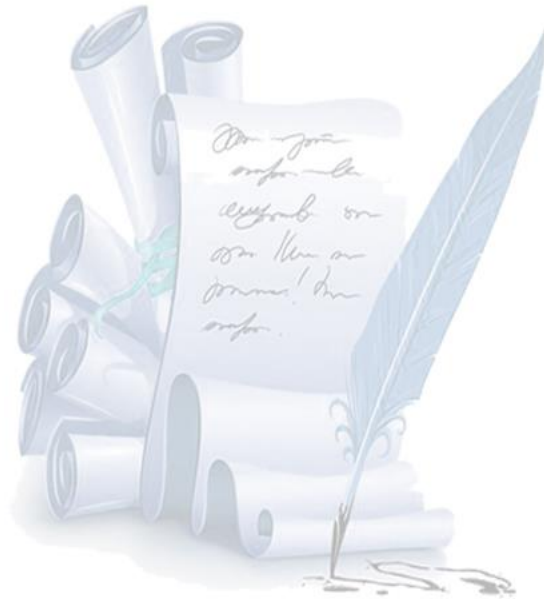
引用



- C++引用是C++引入的新语言特性，是C++常用的一个重要内容之一，正确、灵活地使用引用，可以使程序简洁、高效。
- 引用就是某一变量（或目标）的一个别名，对引用的操作与对变量直接操作完全一样。
- **引用的声明格式：**

类型标识符 &引用名=目标变量名;


```
int a;
int &ra=a; //定义引用ra，它是变量a的引用
```

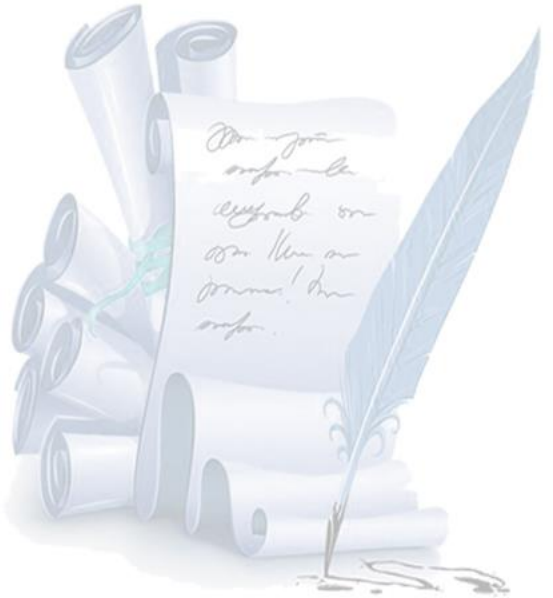


说明



- 说明
  - &在此不是求地址运算，而是起标识作用。
  - 类型标识符是指目标变量的类型。
  - 声明引用时，必须同时对其进行初始化。

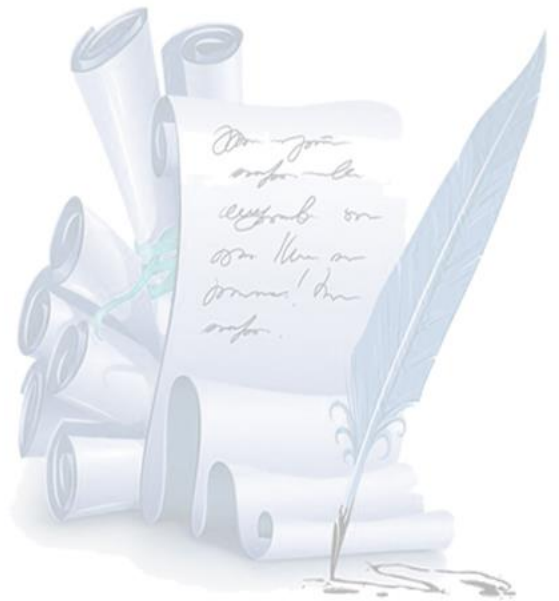
- 使用引用变量的时候有一些限制：
  - 不能引用一个引用变量。
  - 不能创建一个指向引用的指针。
  - 不能建立数组的引用。



引用作为参数



```
#include<iostream>
using namespace std;
void swap(int &p1, int &p2)
{ int p; p=p1; p1=p2; p2=p; }
int main( )
{
    int a,b;
    cin>>a>>b; //输入a,b两变量的值
    swap(a,b);
     cout<<a<<' '<<b; //输出结果
     return 0;
}
```



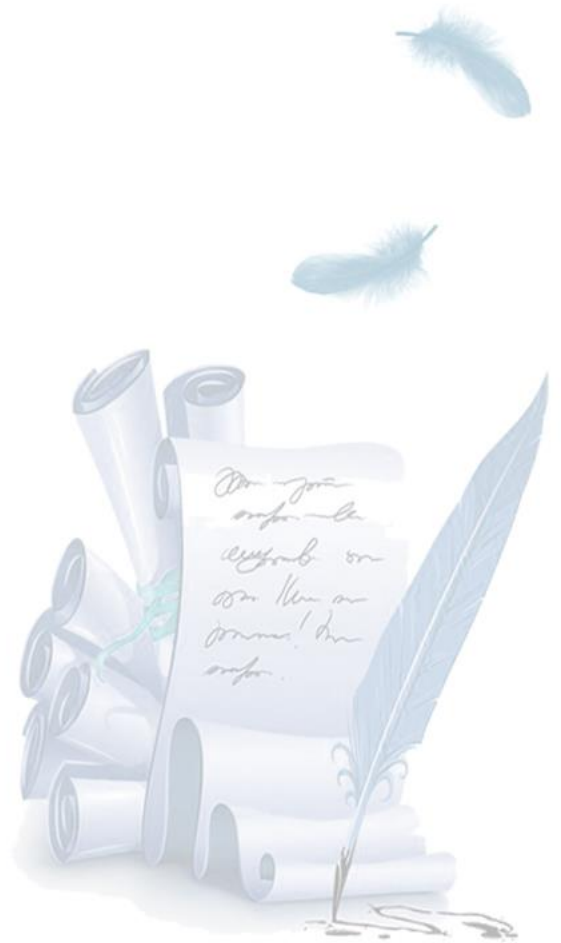
常引用



- 常引用声明一般形式：
  const 类型标识符 &引用名=目标变量名;
- 用这种方式不能通过引用对目标变量的值进行修改。

```
int a ;
const int &ra=a;
ra=1;    ×
a=1;     √
```

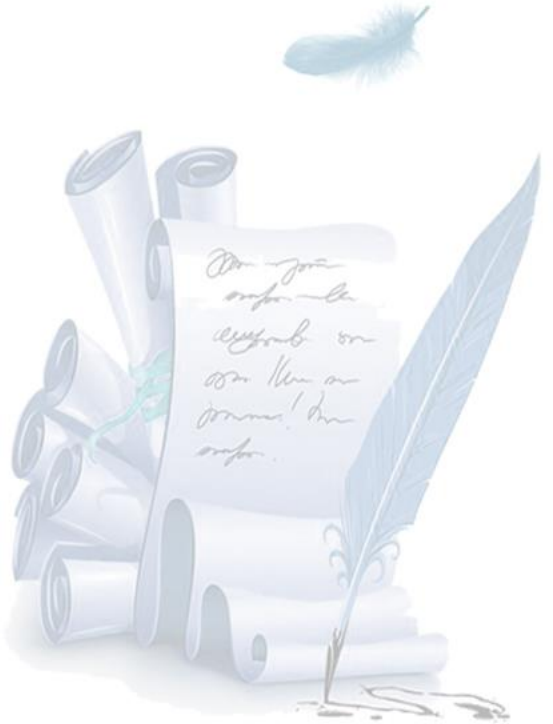
- 若函数中不需要改变形参数据的情形，引用型参数应该尽量定义为const类型。如果既要利用引用提高程序的效率，又要保护传递给函数的数据不在函数中被改变，就应使用常引用，这也符合软件工程的最小权限原则。



引用作为返回值



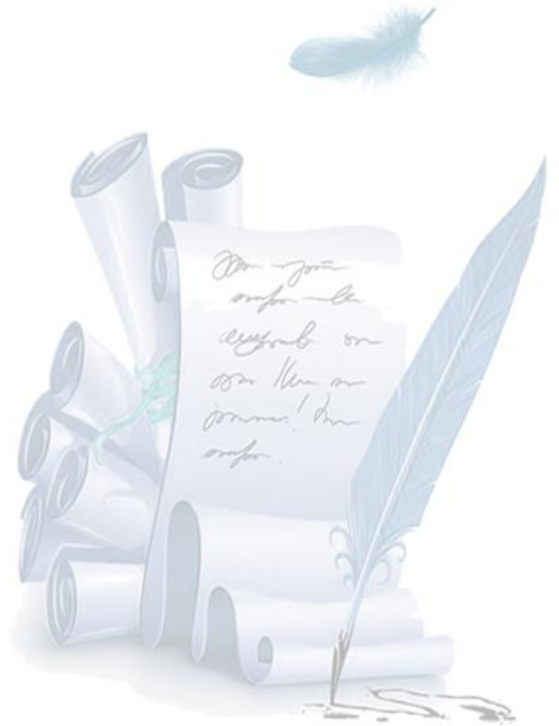
- 要以引用返回函数值，则函数定义时要按以下格式：
  类型标识符 &函数名（形参列表及类型说明）
  {函数体}
- 说明：
  - 以引用返回函数值，定义函数时需要在函数名前加&。
  - 用引用返回一个函数值的最大好处是，在内存中不产生被返回值的副本。
  - 引用作为返回值，不能返回局部变量的引用。主要原因是局部变量会在函数返回后被销毁，因此被返回的引用就成为了“无所指”的引用，程序会进入未知状态。



引用作为返回值程序例子



```
#include<iostream>
using namespace std;
int &put(int n);   int vals[10];     int error=-1;
void main(){
    put(0)=10; //等价于vals[0]=10;
    put(9)=20; //等价于vals[9]=20;
    cout<<vals[0];    cout<<vals[9];
}
int & put(int n){
    if (n>=0 && n<=9 )  return vals[n];
    else {cout<<"subscript error"; return error;          }
}
```



new运算符



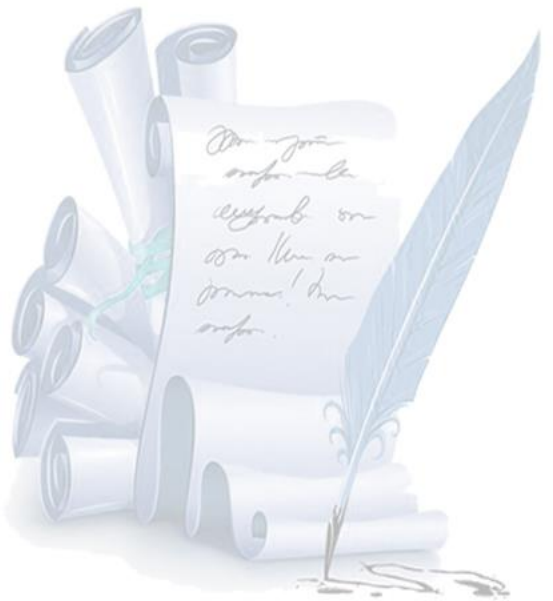
- new运算符：开辟指定大小的存储空间，并返回该存储区的起始地址。
- new的一般格式：

类型说明符 *指针变量名 = new 类型说明符;

类型说明符 *指针变量名 = new 类型说明符[整型表达式];

- 例如：

```
float *p1=new float;
//在堆空间中开辟一个float类型空间，并把其地址赋值给p1。
float *p2=new float[10];
//在堆空间中开辟10个float类型空间，并把其地址赋值给p2。
```



delete运算符



- delete运算符：释放new开辟的存储空间。
- delete的一般格式：

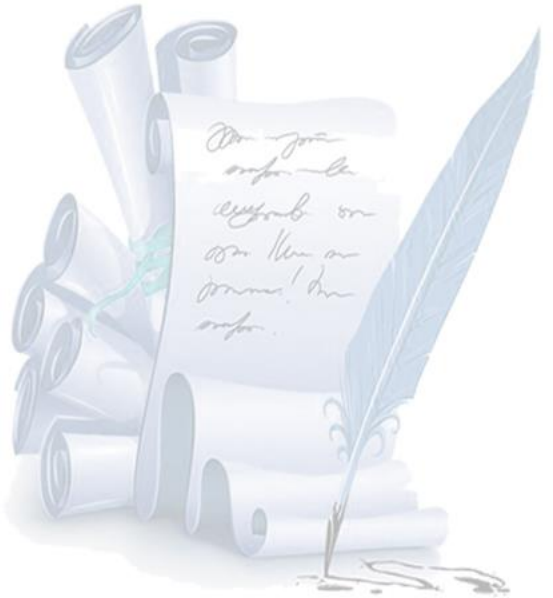
  delete 指针变量名;

  delete [] 指针变量名;
- 能够使用delete释放的空间必须由new申请

- 例如：

  delete p1;

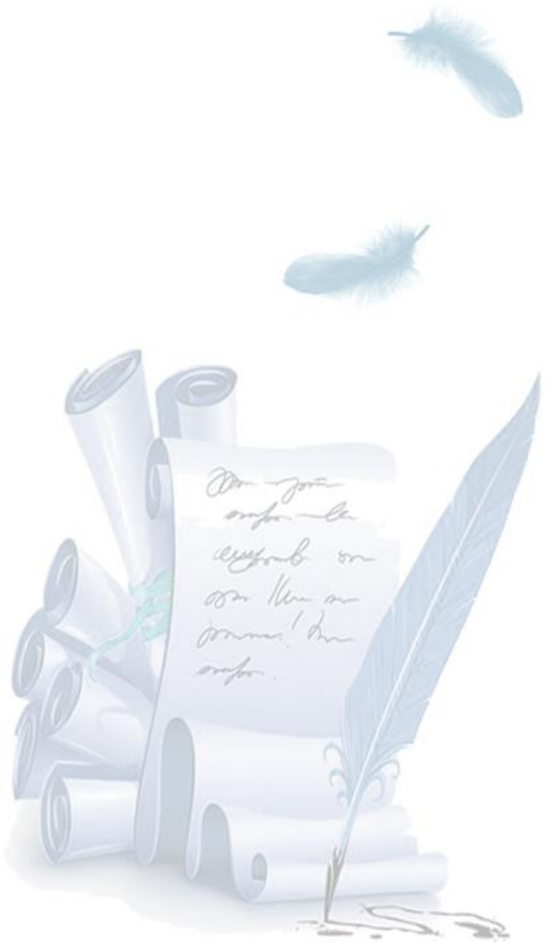
  delete [] p2;



new与delete举例



```
int num,i;
int *p;
 cin>>num;
 p=new  int[num];
 for(i=0;i<inum;i++)
 cin>> p[i];
        …
 delete [] p;
```



- 使用new申请来的空间完全可以像普通数组一样使用，而且它的大小取决于用户输入num值。