山东大学 软件 学院

c++程序设计 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201800301108 | 姓名： 牛翔宇 | | 班级： 18.4 |
| 实验题目：实验三 | | | |
| 实验学时：4 | | 实验日期： 4.25 | |
| 实验目的：  熟悉c++的指针、引用，运算符new和delete。  熟悉c++的类和对象。 | | | |
| 硬件环境：  PC机 | | | |
| 软件环境：  Dev-c++ | | | |
| 实验步骤与内容：  1.实现排序方法sort1(int x, int y, int z),传入参数为整形值，利用冒泡排序比较三个数并按照从小到大排列。  void sort1(int x, int y, int z){  int t;  if(x > y){  t = x;  x = y;  y = t;  }  if(y > z){  t = y;  y = z;  z = t;  }  if(x > y){  t = x;  x = y;  y = t;  }  }  实现排序方法sort2(int \*px, int \*py, int \* pz)，传入参数为指针，还是利用冒泡排序比较三个数并按照从小到大排序。  void sort2(int \*x, int \*y, int \*z){  int t;  if(\*x > \*y){  t = \*x;  \*x = \*y;  \*y = t;  }  if(\*y > \*z){  t = \*y;  \*y = \*z;  \*z = t;  }  if(\*x > \*y){  t = \*x;  \*x = \*y;  \*y = t;  }  }  实现排序方法sort3(int &x, int &y, int &z)，传入参数为指引用，还是利用冒泡排序比较三个数并按照从小到大排序。  void sort3(int &x, int &y, int &z){  int t;  if(x > y){  t = x;  x = y;  y = t;  }  if(y > z){  t = y;  y = z;  z = t;  }  if(x > y){  t = x;  x = y;  y = t;  }  }  在main函数中接受从键盘输入的3个整数，调用以上三个函数进行排序，然后输出排序后的结果。  void sort1(int x, int y, int z);  void sort2(int \*x, int \*y, int \*z);  void sort3(int &x, int &y, int &z);  int main(){  int a, b, c;    cout << "请输入三个不同大小的整数：" << endl;  cin >> a >> b >> c;    int a1 = a, b1 = b, c1 = c;  int \*a2 = &a, \*b2 = &b, \*c2 = &c;  int &a3 = a, &b3 = b, &c3 = c;    sort1(a1, b1, c1);  cout << a1 << "\t" << b1 << "\t" << c1 << "\n" << endl;    sort2(a2, b2, c2);  cout << \*a2 << "\t" << \*b2 << "\t" << \*c2 << "\n" << endl;    sort3(a3, b3, c3);  cout << a3 << "\t" << b3 << "\t" << c3 << "\n" << endl;    return 0;  }  运行结果，以2、3、1为例：    分析：结果发现三种排序函数只有传值的方法没有实现排序，其他两种方法（传指针，传引用）均排序成功，原因是传值的方法传入的参数实际上传入的是原来整型值的复制，所以排序的对象是值的复制而不是原值，所以没有排序成功。而第二第三种排序方法传入的实参为地址，两种方法都把地址指向的数做了改变，因此能排序成功。  完整代码：  --begin--cpp--code--  #include<iostream>  using namespace std;  void sort1(int x, int y, int z);  void sort2(int \*x, int \*y, int \*z);  void sort3(int &x, int &y, int &z);  int main(){  int a, b, c;    cout << "请输入三个不同大小的整数：" << endl;  cin >> a >> b >> c;    int a1 = a, b1 = b, c1 = c;  int \*a2 = &a, \*b2 = &b, \*c2 = &c;  int &a3 = a, &b3 = b, &c3 = c;    sort1(a1, b1, c1);  cout << a1 << "\t" << b1 << "\t" << c1 << "\n" << endl;    sort2(a2, b2, c2);  cout << \*a2 << "\t" << \*b2 << "\t" << \*c2 << "\n" << endl;    sort3(a3, b3, c3);  cout << a3 << "\t" << b3 << "\t" << c3 << "\n" << endl;    return 0;  }  void sort1(int x, int y, int z){  int t;  if(x > y){  t = x;  x = y;  y = t;  }  if(y > z){  t = y;  y = z;  z = t;  }  if(x > y){  t = x;  x = y;  y = t;  }  }  void sort2(int \*x, int \*y, int \*z){  int t;  if(\*x > \*y){  t = \*x;  \*x = \*y;  \*y = t;  }  if(\*y > \*z){  t = \*y;  \*y = \*z;  \*z = t;  }  if(\*x > \*y){  t = \*x;  \*x = \*y;  \*y = t;  }  }  void sort3(int &x, int &y, int &z){  int t;  if(x > y){  t = x;  x = y;  y = t;  }  if(y > z){  t = y;  y = z;  z = t;  }  if(x > y){  t = x;  x = y;  y = t;  }  }  --end--cpp--code--  2. 在main函数中，先接受从键盘输入的数组元素个数n，然后，用new动态创建包含n个元素的一维数组，再从键盘输入数组元素值，最后输出数组所有元素的值。  --begin--cpp--code--  #include<iostream>  using namespace std;  int main(){  int n;  int \*p;  cout << "输入n：" << endl;  cin >> n;  p = new int[n];  cout << "输入n个数组元素：" << endl;  for(int i = 0; i < n; i++){  cin >> p[i];  }  for(int i = 0; i < n; i++){  cout << p[i] << "\t";  }  if (p) {  delete []p;  p = NULL;  }  return 0;  }  --end--cpp--code--  注意利用new运算符开辟空间时，空间的地址必须用指针保存下来，且不能在分配空间时进行初始化。另外，程序结束前用delete运算符将动态分配到的内存空间归还给系统。  运行结果：    3. 检查下面的程序，找出其中的错误并改正。  }-🡪成员函数声明为public  }-🡪成员变量声明为private  }🡪调用函数时改为t.set\_time()，t.show\_time()  🡪函数返回值类型改为void  }🡪去掉t.  🡪函数返回值类型改为void  }🡪去掉t.  改正后的完整代码：  --begin--cpp--code--  #include <iostream>  using namespace std;  class Time{  public:  void set\_time(void);  void show\_time(void);  private:  int hour;  int minute;  int sec;  };  Time t;  int main(){  t.set\_time();  t.show\_time();  return 0;  }  void Time::set\_time(void){  cin >> hour;  cin >> minute;  cin >> sec;  }  void Time::show\_time(void){  cout << hour << ":" << minute << ":" << sec << endl;  }  --end—cpp--code--  运行结果：    4. 改写1中程序，将数据成员改为私有的，将输入和输出的功能改为由成员函数实现，在类体内定义成员函数。  为了使输入和输出的功能改为由成员函数实现，在类体内定义私有成员函数in()和out()，  分别实现输入和输出：  void in(){  cin >> hour;  cin >> minute;  cin >> sec;  }  void out(){  cout << hour << ":" << minute << ":" << sec << endl;  }  在公开函数set\_time()和show\_time()中分别调用in()和out()：  void set\_time(){  in();  }  void show\_time(){  out();  }  完整代码：  --begin--cpp--code--  #include <iostream>  using namespace std;  class Time{  public:  void set\_time(){  in();  }  void show\_time(){  out();  }  private:  int hour;  int minute;  int sec;  void in(){  cin >> hour;  cin >> minute;  cin >> sec;  }  void out(){  cout << hour << ":" << minute << ":" << sec << endl;  }  };  Time t;  int main(){  t.set\_time();  t.show\_time();  return 0;  }  --end—cpp--code--  运行结果：    5.编写类Cuboid，包括数据成员：length(长)、width(宽)、 height(高)、v(体积)定义三个成员函数：  (1) 由键盘分别输入长方体的长、宽、高。  void set\_cuboid(){  cout << "请输入长方体的长、宽、高：" << endl;  cin >> length >> width >> height;  }  (2) 计算长方体的体积；  void get\_V(){  v = length \* width \* height;  }  (3) 输出长方体的体积。  void print\_V(){  cout << "长方体的体积为：" << v << endl;  }  完整程序：  --begin—cpp--code--  #include<iostream>  using namespace std;  class Cuboid{  public:  void set\_cuboid(){  cout << "请输入长方体的长、宽、高：" << endl;  cin >> length >> width >> height;  }  void get\_V(){  v = length \* width \* height;  }  void print\_V(){  cout << "长方体的体积为：" << v << endl;  }  private:  int length, width, height, v;  };  int main(){  Cuboid c1, c2, c3;    c1.set\_cuboid();  c2.set\_cuboid();  c3.set\_cuboid();    c1.get\_V();  c2.get\_V();  c3.get\_V();    c1.print\_V();  c2.print\_V();  c3.print\_V();  }  --end—cpp--code--  运行结果： | | | |
| 结论分析与体会：   1. 函数传入的参数不同，其实际效果可能也不同，例如参数为值，则传入的实际上是值的复制，而对于指针实参，函数调用不能改变实参指针变量的值（即指针指向的地址），但可以改变实参指针变量所指向的变量的值。 2. 用 new 运算符分配的空间，不能在分配空间时进行初始化。 用new开辟的内存单元如果程序不“主动”收回，那么这段空间就一直存在，直到重新开机为止。所以要用delete运算符来将动态分配到的内存空间归还给系统。 3. 在类定义后的}后要加“;”，在定义类对象的时候不用new，编译器自动调用构造方法。注意类成员可见性的声明方式，public:和private:与java区分。 | | | |