山东大学 软件 学院

c++程序设计 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201800301249 | 姓名：王帅 | | 班级：软件4班 |
| 实验题目：实验3 | | | |
| 实验学时：4 | | 实验日期： 2019.4.18 | |
| 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：  IDE： Dev-C++ 5.11  编译器： TDM-GCC 4.9.2 64-bit Release | | | |
| 实验步骤与内容：  实验五.1:  实验代码如下：  --begin—c—code--  #include <iostream>  using namespace std;  void sort1(int x, int y, int z)  {  int t;  if(x>y) {t=x; x=y; y=t;}  if(x>z) {t=x; x=z; z=t;}  if(y>z) {t=y; y=z; z=t;}  cout<<"传值排序后的三个数："<<x<<" "<<y<<" "<<z<<endl;  }  void sort2(int \*px, int \*py, int \*pz)  {  int t;  if(\*px>\*py) {t=\*px; \*px=\*py; \*py=t;}  if(\*px>\*pz) {t=\*px; \*px=\*pz; \*pz=t;}  if(\*py>\*pz) {t=\*py; \*py=\*pz; \*pz=t;}  cout<<"传指针排序后的三个数："<<\*px<<" "<<\*py<<" "<<\*pz<<endl;  }  void sort3(int &x, int &y, int &z)  {  int t;  if(x>y) {t=x; x=y; y=t;}  if(x>z) {t=x; x=z; z=t;}  if(y>z) {t=y; y=z; z=t;}  cout<<"传引用排序后的三个数："<<x<<" "<<y<<" "<<z<<endl;  }  int main()  {  int x,y,z,x0,y0,z0;  cout<<"请输入第一个数x：";  cin>>x;  cout<<"请输入第二个数y：";  cin>>y;  cout<<"请输入第三个数z：";  cin>>z;  cout<<"输入的三个数为: x="<<x<<" "<<"y="<<y<<" "<<"z="<<z<<endl;  cout<<endl;  x0=x; y0=y; z0=z; //保存最初的三个数    sort1(x,y,z);  cout<<"现在的x，y，z的值：x="<<x<<" "<<"y="<<y<<" "<<"z="<<z<<endl;  cout<<endl;    x=x0; y=y0; z=z0; //保证将最初的三个数传给sort2  sort2(&x,&y,&z);  cout<<"现在的x，y，z的值：x="<<x<<" "<<"y="<<y<<" "<<"z="<<z<<endl;  cout<<endl;    x=x0; y=y0; z=z0;  sort3(x,y,z);  cout<<"现在的x，y，z的值：x="<<x<<" "<<"y="<<y<<" "<<"z="<<z<<endl;  }  --end—c—code--  输出结果如下：    可以看出，使用传值的方式排序时，未改变x，y，z原本的值；但使用传指针或传引用的方式排序时，x，y，z的值在排序的过程中改变了，这是因为传入的参数是地址，而传值时传入的是数据的拷贝。  实验五.2：  实验代码如下：  --begin—c—code--  #include <iostream>  using namespace std;  int main()  {  int n;  cout<<"请输入的数组元素个数n"<<" ";  cin>>n;    int \*p = new int[n]; //用new动态创建一个动态数组    cout<<"请输入数组的"<<n<<"个元素："<<endl;  for(int t=0; t<n; t++) {cin>>\*p++;}    cout<<endl;  for(int t=0; t<n; t++) {\*p--;} //是指针p重新指向数组首地址    cout<<"输出数组所有元素的值："<<endl;  for(int t=0; t<n; t++) {cout<<\*p++<<" ";}  }  --end—c—code--  实验结果如下：    实验六1：  修改后的代码如下：  --begin—c—code--  #include <iostream>  using namespace std;  void set\_time(void);  void show\_time(void); //将方法提到外边作为函数    class Time  {  public: //将权限改为public  int hour;  int minute;  int sec;  } ;  Time t;  int main()  {  set\_time();  show\_time();  return 0;  }  void set\_time(void)  {  cin>>t.hour;  cin>>t.minute;  cin>>t.sec;  }  void show\_time(void) //修改返回值类型  {  cout<<t.hour<<":"<<t.minute<<":"<<t.sec<<endl;  }  --end—c—code--  修改后运行该段代码：    实验六2：  改写后的实验代码：  --begin—c—code--  #include <iostream>  using namespace std;  class Time  {  private: //将数据成员改为私有的  int hour;  int minute;  int sec;  public: //将输入和输出的功能改为由成员函数实现  void set\_time(void)  {  cin>>hour;  cin>>minute;  cin>>sec;  }  void show\_time(void) //修改返回值类型  {  cout<<hour<<":"<<minute<<":"<<sec<<endl;  }  } ;  Time t;  int main()  {  t.set\_time();  t.show\_time();  return 0;  }  --end—c—code--  输出结果如下：    实验六3：  实验代码如下：  --begin—c—code--  #include <iostream>  using namespace std;  class cuboid  {  int length;  int width;  int height;  public:  void input(void)  {  cout<<"请输入长方体的长："; cin>>length;  cout<<"请输入长方体的宽："; cin>>width;  cout<<"请输入长方体的高："; cin>>height;  }  int calculate(int length, int width, int height)  {  int v = length\*width\*height;  return v;  }  void output(void)  {  cout<<"长方体的体积为："<<calculate(length, width, height)<<endl;  }  };  int main()  {  cuboid c1,c2,c3;  cout<<"第一个长方形："<<endl;  c1.input();  c1.output();  cout<<"第二个长方形："<<endl;  c2.input();  c2.output();  cout<<"第三个长方形："<<endl;  c3.input();  c3.output();  }  --end—c—code--  实验结果如下： | | | |
| 结论分析与体会：  本次实验使我更加理解了C++语言中指针和引用的本质以及它们的使用方法，也令我熟悉了类与对象的操作与java语言的异同。 | | | |