实验报告

东北林业大学

信息与计算机科学技术实验中心

|  |
| --- |
| 1. 实验目的   1·理解运算符重载的概念和实质；  2·掌握运算符重载函数的定义方法；  3·掌握虚函数的定义方法及其在实现多态性中的作用；  4·理解和掌握实现动态多态性的前提条件；  5·理解静态多态性和动态多态性的区别。 |
| 1. 实验环境   Codeblocks |
| 三、实验内容及结果  1． 编一个程序，用成员函数重载运算符“+”和“-”，将两个二维数组相加和相减，要求第一个二维数组的值由构造函数设置，另一个二维数组的值由键盘输入。  2． 修改上题，用友员函数重载运算符“+”和“-”将两个二维数组相加和相减。  3． 编写程序，计算汽车运行的时间，首先建立基类car，其中含有数据成员distance存储两点间的距离。假定距离以英里计算，速度为每小时80英里，使用虚函数travel\_time()计算并显示通过这段距离的时间。在派生类kilometre中，假定距离以千米计算，速度为每小时120千米，使用函数travel\_time()计算并显示通过这段距离的时间。 |

|  |
| --- |
| 四、实验过程分析与讨论  1． 编一个程序，用成员函数重载运算符“+”和“-”，将两个二维数组相加和相减，要求第一个二维数组的值由构造函数设置，另一个二维数组的值由键盘输入。  #include <iostream>  using namespace std;  class TwoDArray {  private:  int rows, cols;  int \*\*data;  public:  // 构造函数  TwoDArray(int r, int c) : rows(r), cols(c) {  data = new int\*[rows];  for (int i = 0; i < rows; ++i) {  data[i] = new int[cols];  for (int j = 0; j < cols; ++j) {  data[i][j] = rand() % 10; // 随机初始化数组  }  }  }    // 从键盘输入数组的值  void input() {  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  cout << "Enter value for element [" << i << "][" << j << "]: ";  cin >> data[i][j];  }  }  }    // 运算符+重载  TwoDArray operator+(const TwoDArray &other) {  TwoDArray result(rows, cols);  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  result.data[i][j] = data[i][j] + other.data[i][j];  }  }  return result;  }    // 运算符-重载  TwoDArray operator-(const TwoDArray &other) {  TwoDArray result(rows, cols);  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  result.data[i][j] = data[i][j] - other.data[i][j];  }  }  return result;  }    // 打印数组  void print() {  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  cout << data[i][j] << " ";  }  cout << endl;  }  }    // 析构函数  ~TwoDArray() {  for (int i = 0; i < rows; ++i) {  delete[] data[i];  }  delete[] data;  }  };  int main() {  int rows = 2, cols = 3;  TwoDArray array1(rows, cols); // 通过构造函数设置第一个二维数组  TwoDArray array2(rows, cols);    cout << "Input values for the second array:" << endl;  array2.input(); // 从键盘输入第二个二维数组的值    cout << "Array 1:" << endl;  array1.print();  cout << "Array 2:" << endl;  array2.print();    TwoDArray sum = array1 + array2;  TwoDArray diff = array1 - array2;    cout << "Sum of arrays:" << endl;  sum.print();  cout << "Difference of arrays:" << endl;  diff.print();    return 0;  }  2． 修改上题，用友员函数重载运算符“+”和“-”将两个二维数组相加和相减。  #include <iostream>  using namespace std;  class TwoDArray {  private:  int rows, cols;  int \*\*data;  public:  // 构造函数  TwoDArray(int r, int c) : rows(r), cols(c) {  data = new int\*[rows];  for (int i = 0; i < rows; ++i) {  data[i] = new int[cols];  for (int j = 0; j < cols; ++j) {  data[i][j] = rand() % 10; // 随机初始化数组  }  }  }    // 从键盘输入数组的值  void input() {  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  cout << "Enter value for element [" << i << "][" << j << "]: ";  cin >> data[i][j];  }  }  }    // 友元函数重载运算符+  friend TwoDArray operator+(const TwoDArray &a, const TwoDArray &b) {  TwoDArray result(a.rows, a.cols);  for (int i = 0; i < a.rows; ++i) {  for (int j = 0; j < a.cols; ++j) {  result.data[i][j] = a.data[i][j] + b.data[i][j];  }  }  return result;  }    // 友元函数重载运算符-  friend TwoDArray operator-(const TwoDArray &a, const TwoDArray &b) {  TwoDArray result(a.rows, a.cols);  for (int i = 0; i < a.rows; ++i) {  for (int j = 0; j < a.cols; ++j) {  result.data[i][j] = a.data[i][j] - b.data[i][j];  }  }  return result;  }    // 打印数组  void print() {  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  cout << data[i][j] << " ";  }  cout << endl;  }  }    // 析构函数  ~TwoDArray() {  for (int i = 0; i < rows; ++i) {  delete[] data[i];  }  delete[] data;  }  };  int main() {  int rows = 2, cols = 3;  TwoDArray array1(rows, cols); // 通过构造函数设置第一个二维数组  TwoDArray array2(rows, cols);    cout << "Input values for the second array:" << endl;  array2.input(); // 从键盘输入第二个二维数组的值    cout << "Array 1:" << endl;  array1.print();  cout << "Array 2:" << endl;  array2.print();    TwoDArray sum = array1 + array2;  TwoDArray diff = array1 - array2;    cout << "Sum of arrays:" << endl;  sum.print();  cout << "Difference of arrays:" << endl;  diff.print();    return 0;  }  3． 编写程序，计算汽车运行的时间，首先建立基类car，其中含有数据成员distance存储两点间的距离。假定距离以英里计算，速度为每小时80英里，使用虚函数travel\_time()计算并显示通过这段距离的时间。在派生类kilometre中，假定距离以千米计算，速度为每小时120千米，使用函数travel\_time()计算并显示通过这段距离的时间。  #include <iostream>  using namespace std;  // 基类Car  class Car {  protected:  double distance; // 距离，单位：英里  public:  Car(double d) : distance(d) {}    // 虚函数，计算并显示通过这段距离的时间  virtual void travel\_time() {  double speed = 80.0; // 每小时80英里  double time = distance / speed;  cout << "Travel time in miles: " << time << " hours" << endl;  }  };  // 派生类Kilometre  class Kilometre : public Car {  public:  Kilometre(double d) : Car(d) {}    // 覆盖基类的travel\_time函数  void travel\_time() override {  double speed = 120.0; // 每小时120千米  double time = distance / speed;  cout << "Travel time in kilometers: " << time << " hours" << endl;  }  };  int main() {  Car car1(160); // 160英里  Kilometre car2(160); // 160千米    car1.travel\_time();  car2.travel\_time();    return 0;  } |
|  |
| 五、指导教师意见  指导教师签字：  年 月 日 |