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. 掌握单继承和多重继承的方式定义派生类的方法；  2·深刻理解在各种继承方式下构造函数和析构函数的执行顺序；  3·理解和掌握公有继承、私有继承和保护继承对基类成员的访问机制；  4·理解虚基类的概念以及引入虚基类的目的和作用 |
| 1. 实验环境   Codeblocks |
| 三、实验内容及结果  1.下面的程序可以输出ASCII字符与所对应的数字的对照表，修改下列程序，使其可以输出字母a到字母z。  #include<iostream.h>  #include<iomanip.h>  class table{  public:  table(int p){ i=p;}  void ascii(void);  protected: int I;  };  void table::ascii(void)  { int k=1;  for(;i<127;i++){ cout<<setw(4)<<i<<” ”<<(char)i;  if ((k)%12==0) cout<<”\n” ; k++; }  cout<<”\n”; }  class der\_table:public table{  public:  der\_table(int p,char\*m):table(p){c=m;}  void print(void);  protected: char\*c; } ;  void der\_table::print(void){ cout<<c<<”\n”; table::ascii();}  void main()  { der\_table ob1(32,”ASCII value-char”); ob1.print();  der\_table ob2(‘a’,”ASCII value-char”);b2.print(); }  提示：修改后的主程序为  void main()  { der\_table ob(‘a’,’z’,”ASCII value-char”); ob.print(); }  2.下面的程序包含了Time类和Date类的声明，要求设计一个Birthtime类，它继承了Time类和Date类，并且还有一项出生孩子的名字Childname，同时设计主程序显示一个小孩的出生时间和名字。  #include<iostream.h>  #include<string.h>  class Time{  public:  Time(int h,int m,int s){ hours=h; minutes=m; seconds=s;}  virtual void display()  { cout<<hours<<”:”<<minutes<<”:”<<seconds<<endl;}  protected: int hours,minutes,seconds;  };  class Date{  public:  Date(int m,int d,int y){ month=m; day=d; year=y;}  virtual void display(){ cout<<month<<”/”<<day<<”/”<<year;}  protected; int month,day,year;  };  3.建立普通的基类building，用来存储一座楼房的层数，房间数以及它的总平方数。建立派生类house，继承building，并存储卧室与浴室的数量，另外，建立派生类office，继承building，并存储灭火器与电话的数目。  4.按照图4所示的类层次图要求编写程序。定义属于类score的对象c1及类teacher的对象t1，分别输入各数据成员的值后再显示出这些数据。  5．递归调用被继承的基类成员函数，实现求素数的功能。  6．递归调用被继承的基类成员函数，求最大公约数。 |

|  |
| --- |
| 1. 实验过程分析与讨论   1.下面的程序可以输出ASCII字符与所对应的数字的对照表，修改下列程序，使其可以输出字母a到字母z。  #include<iostream.h>  #include<iomanip.h>  class table{  public:  table(int p){ i=p;}  void ascii(void);  protected: int I;  };  void table::ascii(void)  { int k=1;  for(;i<127;i++){ cout<<setw(4)<<i<<” ”<<(char)i;  if ((k)%12==0) cout<<”\n” ; k++; }  cout<<”\n”; }  class der\_table:public table{  public:  der\_table(int p,char\*m):table(p){c=m;}  void print(void);  protected: char\*c; } ;  void der\_table::print(void){ cout<<c<<”\n”; table::ascii();}  void main()  { der\_table ob1(32,”ASCII value-char”); ob1.print();  der\_table ob2(‘a’,”ASCII value-char”);b2.print(); }  提示：修改后的主程序为  void main()  { der\_table ob(‘a’,’z’,”ASCII value-char”); ob.print(); }  代码过程：  #include <iostream>  #include <iomanip>  using namespace std;  class Table {  public:  Table(char start, char end) : start(start), end(end) {}  void ascii() const;  protected:  char start, end;  };  void Table::ascii() const {  for (char i = start; i <= end; ++i) {  cout << setw(4) << (int)i << " " << i;  if ((i - start + 1) % 12 == 0) cout << "\n";  }  cout << "\n";  }  class DerTable : public Table {  public:  DerTable(char start, char end, const char\* title) : Table(start, end), title(title) {}  void print() const;  protected:  const char\* title;  };  void DerTable::print() const {  cout << title << "\n";  Table::ascii();  }  int main() {  DerTable ob('a', 'z', "ASCII value-char");  ob.print();  return 0;  }  2.下面的程序包含了Time类和Date类的声明，要求设计一个Birthtime类，它继承了Time类和Date类，并且还有一项出生孩子的名字Childname，同时设计主程序显示一个小孩的出生时间和名字。  #include<iostream.h>  #include<string.h>  class Time{  public:  Time(int h,int m,int s){ hours=h; minutes=m; seconds=s;}  virtual void display()  { cout<<hours<<”:”<<minutes<<”:”<<seconds<<endl;}  protected: int hours,minutes,seconds;  };  class Date{  public:  Date(int m,int d,int y){ month=m; day=d; year=y;}  virtual void display(){ cout<<month<<”/”<<day<<”/”<<year;}  protected; int month,day,year;  };  代码过程：  #include <iostream>  #include <string>  using namespace std;  class Time {  public:  Time(int h, int m, int s) : hours(h), minutes(m), seconds(s) {}  virtual void display() const {  cout << hours << ":" << minutes << ":" << seconds << endl;  }  protected:  int hours, minutes, seconds;  };  class Date {  public:  Date(int m, int d, int y) : month(m), day(d), year(y) {}  virtual void display() const {  cout << month << "/" << day << "/" << year << endl;  }  protected:  int month, day, year;  };  class Birthtime : public Time, public Date {  public:  Birthtime(int h, int m, int s, int mon, int d, int y, const string& name)  : Time(h, m, s), Date(mon, d, y), childName(name) {}  void display() const override {  cout << "Child's Name: " << childName << endl;  cout << "Birth Date: ";  Date::display();  cout << "Birth Time: ";  Time::display();  }  private:  string childName;  };  int main() {  Birthtime birth(12, 30, 45, 5, 10, 2003, "John Doe");  birth.display();  return 0;  }  3.建立普通的基类building，用来存储一座楼房的层数，房间数以及它的总平方数。建立派生类house，继承building，并存储卧室与浴室的数量，另外，建立派生类office，继承building，并存储灭火器与电话的数目。  #include <iostream>  using namespace std;  class Building {  public:  Building(int floors, int rooms, double area) : floors(floors), rooms(rooms), area(area) {}  virtual void display() const {  cout << "Floors: " << floors << ", Rooms: " << rooms << ", Area: " << area << endl;  }  protected:  int floors, rooms;  double area;  };  class House : public Building {  public:  House(int floors, int rooms, double area, int bedrooms, int bathrooms)  : Building(floors, rooms, area), bedrooms(bedrooms), bathrooms(bathrooms) {}  void display() const override {  Building::display();  cout << "Bedrooms: " << bedrooms << ", Bathrooms: " << bathrooms << endl;  }  private:  int bedrooms, bathrooms;  };  class Office : public Building {  public:  Office(int floors, int rooms, double area, int extinguishers, int phones)  : Building(floors, rooms, area), extinguishers(extinguishers), phones(phones) {}  void display() const override {  Building::display();  cout << "Extinguishers: " << extinguishers << ", Phones: " << phones << endl;  }  private:  int extinguishers, phones;  };  int main() {  House h(2, 5, 150.0, 3, 2);  h.display();  Office o(3, 10, 300.0, 5, 20);  o.display();  return 0;  }  4.按照图4所示的类层次图要求编写程序。定义属于类score的对象c1及类teacher的对象t1，分别输入各数据成员的值后再显示出这些数据。  #include <iostream>  #include <string>  class Person {  public:  Person(const std::string& name) : name(name) {}  virtual void display() const {  std::cout << "Name: " << name << std::endl;  }  protected:  std::string name;  };  class Score : public Person {  public:  Score(const std::string& name, int score) : Person(name), score(score) {}  void display() const override {  Person::display();  std::cout << "Score: " << score << std::endl;  }  private:  int score;  };  class Teacher : public Person {  public:  Teacher(const std::string& name, const std::string& subject)  : Person(name), subject(subject) {}  void display() const override {  Person::display();  std::cout << "Subject: " << subject << std::endl;  }  private:  std::string subject;  };  int main() {  Score c1("Student1", 95);  Teacher t1("Teacher1", "Math");  std::cout << "Score object:\n";  c1.display();  std::cout << "\nTeacher object:\n";  t1.display();  return 0;  }  5．递归调用被继承的基类成员函数，实现求素数的功能。  #include <iostream>  using namespace std;  class PrimeChecker {  public:  virtual bool isPrime(int n, int i = 2) const {  if (n <= 2) return (n == 2);  if (n % i == 0) return false;  if (i \* i > n) return true;  return isPrime(n, i + 1);  }  };  class DerivedPrimeChecker : public PrimeChecker {  public:  bool checkPrime(int n) const {  return isPrime(n);  }  };  int main() {  DerivedPrimeChecker checker;  int num;  cout << "Enter a number: ";  cin >> num;  if (checker.checkPrime(num)) {  cout << num << " is a prime number." << endl;  } else {  cout << num << " is not a prime number." << endl;  }  return 0;  }  6．递归调用被继承的基类成员函数，求最大公约数。  #include <iostream>  using namespace std;  class GCDCalculator {  public:  virtual int gcd(int a, int b) const {  if (b == 0) return a;  return gcd(b, a % b);  }  };  class DerivedGCDCalculator : public GCDCalculator {  public:  int calculateGCD(int a, int b) const {  return gcd(a, b);  }  };  int main() {  DerivedGCDCalculator calculator;  int num1, num2;  cout << "Enter two numbers: ";  cin >> num1 >> num2;  int result = calculator.calculateGCD(num1, num2);  cout << "The GCD of " << num1 << " and " << num2 << " is " << result << "." << endl;  return 0;  } |
|  |
| 五、指导教师意见  指导教师签字：  年 月 日 |