浙江大学2016–2017学年秋冬学期

《面向对象程序设计》课程期末考试试卷

课程号： 211C0010 ，开课学院： 计算机学院

考试试卷： √A卷、B卷（请在选定项上打√）

考试形式：√闭、开卷（请在选定项上打√），允许带 无 入场

考试日期： 2017 年 01 月 18 日,考试时间： 120 分钟

诚信考试，沉着应考，杜绝违纪。

考生姓名： 学号： 所属院系：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题序 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 总 分 |
| 得分 |  |  |  |  |  |  |  |  |  |
| 评卷人 |  |  |  |  |  |  |  |  |  |

只能在本页答题，写在其他页无效。

1. [30 marks] [5 mark for each question]

1)

In function Max, 2

In template function Max, 2

In function Max, 2

2)

97 194 297

97 194 33

3)

call A(int ii=0).

call A(int ii=0).

call B(int ii=0).

A::i = 0

A::i = 0

B::i = 2

call A(int ii=0).

call A(int ii=0).

call B(const B&).

A::i = 0

A::i = 0

B::i = 2

4)

A()

Generating an exception object，name is ex\_obj1

copy an exception object，name is ex\_ex\_obj1

destroy an exception object，name is ex\_obj1

~A()

catch exception

destroy an exception object，name is ex\_ex\_obj1

5)

A()1

B()2

C()

6)

Parent(int ii)

Member(int ii)

Child(int ii)

Parent(const Parent&)

Member(const Member&)

Parent: 2

Member: 2

Child: 2

2. [10 marks]

1)

class A{

int i;

public:

A(int ii):i(ii){} //此处改成：A(int ii = 0) : i(ii) {} 或增加一个A():i(0) {}

//增加：virtual ~A(){}

};

class B: public A{

char \*p;

public:

B(char \*p)

{

p = new char[strlen(p)+1]; //此处改成：this->p = new char[strlen(p)+1];

strcpy(p, p); //此处改成：strcpy(this->p, p);

}

~B()

{

delete p; //此处改成：delete[] p;

}

};

int main()

{

B b("hello");

A \*p = new B("world!");

//此处增加：delete p;

}

2)

class Exception {};

class OneException : public Exception {};

void f(int index) throw()

{

if ( index < 0 ) throw new OneException();

}

int main()

{

int k;

cin >> k;

try {

f(k);

}

catch (OneException) {

cout << "caught OneException" << endl;}

catch (Exception) {

cout << "caught Exception" << endl;

}

catch (...) {

cout << "caught ... " << endl;

}

3. [25 marks] 1 mark for each blank

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | int Shape::counter = 0; | 14 | virtual double area()  {  return width\*length;  } |
| 2 | double Shape::error() const  {  cout << "default code for error." << endl;  } | 15 | virtual void error() const  {  cout << "Rectangle code for error." << endl;  } |
| 3 | lax(l),sax(s) | 16 | int Rectangle::counter = 0; |
| 4 | virtual double area()  {  return PI\*lax\*sax;  } | 17 | Rectangle(r,r) |
| 5 | virtual void error() const  {  cout << "Ellipse code for error." << endl;  } | 18 | counter++ |
| 6 | int Ellipse::counter = 0; | 19 |  |
| 7 | Ellipse(r,r) | 20 |  |
| 8 | counter++; | 21 | int Square::counter = 0; |
| 9 |  | 22 | a(a),b(b),c(c) |
| 10 |  | 23 | virtual double area()  {  double l;  l = (a+b+c)/2;  return sqrt(l\*(l-a)\*(l-b)\*(l-c));  } |
| 11 | int Circle::counter = 0; | 24 | virtual void error() const  {  cout << "Triangle code for error." << endl;  } |
| 12 | width(w),length(l) | 25 | int Triangle::counter=0; |
| 13 | if (w!=l) counter++; |  |  |

4. [35 marks]

#ifndef MATRIX\_H

#define MATRIX\_H

#include <iostream>

using namespace std;

template <typename T>

class Matrix {

public:

Matrix(int r,int c);

~Matrix() 3 marks

{

delete[] pv;

}

Matrix(const Matrix& m);

Matrix operator+(const Matrix& m);

T& operator()(int i, int j);

friend ostream& operator<< <>(ostream& os, const Matrix<T>& n);

friend istream& operator>> <>(istream& in, Matrix<T>& n);

private: 2 marks

int row,col; 2 marks

T \*pv; 2 marks

Matrix& operator=(const Matrix& m);

};

The following functions 5 marks

template <typename T>

Matrix<T>::Matrix(int r, int c):row(r),col(c)

{

pv = new T[row\*col];

}

The following functions 5 marks

template <typename T>

Matrix<T>::Matrix(const Matrix& m)

{

row = m.row;

col = m.col;

pv = new T[row\*col];

::memcpy( pv, m.pv, row\*col\*sizeof(T) );

}

The following functions 5 marks

template <typename T>

Matrix<T> Matrix<T>:: operator+(const Matrix& m)

{

if(row == m.row && col == m.col)

{

Matrix temp(row,col);

for (int i=0; i<m.row\*m.col; i++)

\*(temp.pv+i) = \*(pv+i) + \*(m.pv+i);

return temp;

}

}

The following functions 5 marks

template <typename T>

T& Matrix<T>::operator()(int i, int j)

{

if (i>=row) {

std::cout << "行下标越界！" << std::endl;

exit(0);

}

if (j>=col) {

std::cout << "列下标越界！" << std::endl;

exit(0);

}

return \*(pv + i\*col + j);

}

The following functions 4 marks

template <typename T>

ostream& operator <<(ostream& os, const Matrix<T>& n)

{

int total = n.row \* n.col;

for(int i = 0;i < total;i++){

os << \*(n.pv+i) << "\t";

if ((i+1)%n.col == 0)

os << std::endl;

}

return os;

}

The following functions 4 marks

template <typename T>

istream& operator>>(istream& in, Matrix<T>& n)

{

int total = n.row \* n.col;

for(int i = 0; i<total;i++){

in >> \*(n.pv+i);

}

return in;

}