1. **填空题（每空2分，共34分）**

1. 继承 、 多态

2. 参数个数 、 参数类型

3. 参数类型 、 返回值类型

4. this指针

5. static（静态）

6. friend（友元）

7. private（私有）

8. 抽象类 、 纯虚

9. 先执行基类构造函数，再执行成员对象的构造函数，最后执行本类构造函数

10. inline（内联）

11. a的数据类型应该是指针Person\*； Person\* a = new Person("noah");

12. 函数 、 类

1. **程序综合题(每空2分，第1、6、9空4分，共28分)**

1. （1） 4分

double Area()

{

double half = (\_line1 + \_line2 + \_line3)/2.0;

double s = sqrt(half \* (half - \_line1) \* (half - \_line2) \* (half - \_line3));

return s;

}

（2） 2分

double Perimeter()

{

return \_line1 + \_line2 + \_line3;

}

2. （3） 2分

static int \_count;

(4) 2分

int Student::\_count = 0;

3. (5) 2分

template <typename T>

(6) 4分

int i, j;

T t;

for (i = 1; i < n; i++)

{

t = a[i];

j = i;

while (j > 0 && t < a[j - 1])

{

a[j] = a[j - 1];

j--;

}

a[j] = t;

}

(7) 2分

template <typename T>

4. (8) 2分

sal["Bob"] = 2000; 或者： sal.insert(make\_pair("Bob",2000));

sal["Tom"] = 3000; sal.insert(make\_pair("Tom",3000));

5. (9) 4分

if (arr.\_sz <= 0)

exit(1);

else

{ \_sz = arr.\_sz;

\_data = new int[arr.\_sz];

for (int i = 0; i < arr.\_sz; i++)

\_data[i] = arr.\_data[i];

}

6. (10) 2分

friend float Total(StuScore s);

(11) 2分

return s.\_phy + s.\_che + s.\_bio;

**三、程序设计题(第1小题16分，第2小题22分，共38分)**

1.

class Son: public Father //4分

{public:

Son(string name = "XXX", int age = 0);

void Ability();

void Show();

~Son() {}

};

Son::Son(string name, int age) :Father(name, age) //4分

{

}

void Son::Ability() //4分

{

Father::Ability(); //或者cout << "He plays the piano well." << endl;

cout << "He sings very well." << endl;

}

void Son::Show() //4分

{

cout << "Son: " << \_name << " " << \_age << endl;

Ability();

cout << endl;

}

2.

Matrix Matrix::operator+(const Matrix& m) const //4分

{

Matrix result;

int i, j;

if (\_row == m.\_row && \_line == m.\_line)

{

result.\_row = \_row;

result.\_line = \_line;

for (i = 0; i < \_row; i++)

for (j = 0; j < \_line; j++)

result.\_data[i][j] = \_data[i][j] + m.\_data[i][j];

}

else

cout << "Can not Matrix(" << \_row << "," << \_line << ")+Matrix(" << m.\_row << "," << m.\_line << ")!" << endl;

return result;

}

Matrix Matrix::operator-(const Matrix& m) const //4分

{

Matrix result;

int i, j;

if (\_row == m.\_row && \_line == m.\_line)

{

result.\_row = \_row;

result.\_line = \_line;

for (i = 0; i < \_row; i++)

for (j = 0; j < \_line; j++)

result.\_data[i][j] = \_data[i][j] - m.\_data[i][j];

}

else

cout << "Can not Matrix(" << \_row << "," << \_line << ")-Matrix(" << m.\_row << "," << m.\_line << ")!" << endl;

return result;

}

Matrix Matrix::operator\*(const Matrix& m) const //6分

{

Matrix result;

int i, j,k;

if (\_line == m.\_row)

{

result.\_row = \_row;

result.\_line = m.\_line;

for (i = 0; i < result.\_row; i++)

for (j = 0; j < result.\_line; j++)

{

result.\_data[i][j] = 0;

for (k = 0; k < \_line; k++)

result.\_data[i][j] += \_data[i][k] \* m.\_data[k][j];

}

}

else

cout << "Can not Matrix(" << \_row << "," << \_line << ")\*Matrix(" << m.\_row << "," << m.\_line << ")!" << endl;

return result;

}

ostream& operator<<(ostream& out, const Matrix& m) //4分

{

int i, j;

if (m.\_row == 0 || m.\_line == 0)

cout << "No data" << endl;

else

for(i=0;i<m.\_row;i++)

{

for (j = 0; j < m.\_line; j++)

cout << m.\_data[i][j] << " ";

cout << endl;

}

return out;

}

istream& operator>>(istream& in, Matrix& m) //4分

{

int i, j,row,line;

do

{

cout << "Row(int, >=2,<=100): ";

cin >> row;

cout << "Line(int, >=2,<=100): ";

cin >> line;

} while (row<2||line<2||row>100||line>100); //row、line的输入判定可以不写

m.\_row = row;

m.\_line = line;

for (i = 0; i < m.\_row; i++)

for (j = 0; j < m.\_line; j++)

cin >> m.\_data[i][j];

return in;

}