程序设计II期末机考(2014)

事件	时间	
提交开始	2014-06-15	13:0
提交结束	2014-06-15	22:0

1-2 of 2

学号
姓名
班级
保存

1. Write an object function (easy)

Please write an object function in class Triangle (三角形), which compares the perimeters (周长) of two triangles:

- **Function**

  bool Triangle::isLargerThan(const Triangle & tri2) const

  return true if the perimeter of triangle *this* is larger than that of triangle *tri2*.

EXAMPLE INPUT

```
2 6 5
3 4 5
```

EXAMPLE OUTPUT

```
1
```

主程序

```
class Triangle
{
private:
    double side1;
    double side2;
    double side3;

public:
    Triangle(double side1, double side2, double side3) {
        this->side1 = side1;
        this->side2 = side2;
        this->side3 = side3;
    }

#include "source"
};

#include <iostream>
using namespace std;

int main() {
    double side11, side12, side13,
        side21, side22, side23;
    cin >> side11 >> side12 >> side13;
    cin >> side21 >> side22 >> side23;

    cout << Triangle(side11, side12, side13).isLargerThan(
        Triangle(side21, side22, side23));
}
```

答案程序

```
private:
    double getPerimeter() const {
        return side1 + side2 + side3;
```

```
    }

public:
    bool isLargerThan(const Triangle & t) const {
        return this->getPerimeter() > t.getPerimeter();
    }
```

文本编辑框

保存和测试对错

2. Write a static function (easy)

Please write a static function in class Triangle (三角形), which compares the perimeters (周长) of two triangles:

- Function

  `int Triangle::compare(const Triangle & tri1, const Triangle & tri2)`

  return 1 if the perimeter of triangle *tri1* is larger than that of triangle *tri2*, -1 is smaller, and 0 if equals.

EXAMPLE INPUT

```
2 6 5
3 4 5
4 3 5
2 6 5
```

EXAMPLE OUTPUT

```
1
0
-1
```

主程序

```
class Triangle
{
private:
    double side1;
    double side2;
    double side3;

public:
    Triangle(double side1, double side2, double side3) {
        this->side1 = side1;
        this->side2 = side2;
        this->side3 = side3;
    }

#include "source"
};

#include <iostream>
using namespace std;

Triangle readTriangle() {
    double side1, side2, side3;
    cin >> side1 >> side2 >> side3;
    return Triangle(side1, side2, side3);
}

int main() {
    Triangle tri1 = readTriangle();
    for (int i = 0; i < 3; ++ i) {
        const Triangle tri2 = readTriangle();
        cout << Triangle::compare(tri1, tri2) << endl;
        tri1 = tri2;
```

```
    }
}
```

答案程序

```
private:
    double getPerimeter() const {
        return side1 + side2 + side3;
    }

public:
    static int compare(const Triangle & t1, const Triangle & t2) {
        if (t1.getPerimeter() > t2.getPerimeter()) return 1;
        if (t1.getPerimeter() < t2.getPerimeter()) return -1;
        return 0;
    }
```

文本编辑框

保存和测试对错

3. Write a constructor (easy)

Please write a constructor in class TwoTriangle:

- **TwoTriangle(const Triangle & tri1, const Triangle & tri2)**

EXAMPLE INPUT

```
2 6 5
3 4 5
```

EXAMPLE OUTPUT

```
(3,4,5)
(2,6,5)
```

主程序

```
#include <iostream>
using namespace std;

class Triangle
{
private:
    double side1;
    double side2;
    double side3;

public:
    Triangle(double side1, double side2, double side3) {
        this->side1 = side1;
        this->side2 = side2;
        this->side3 = side3;
    }

    void print() const {
        cout << "(" << side1 << "," << side2 << "," << side3 << ")" << endl;
    }
};

class TwoTriangle : public Triangle
{
private:
    Triangle member;

public:
    void print() const {
        Triangle::print();
        member.print();
    }
```

```
33
34 #include "source"
35 };
36
37 Triangle readTriangle() {
38     double side1, side2, side3;
39     cin >> side1 >> side2 >> side3;
40     return Triangle(side1, side2, side3);
41 }
42
43 int main() {
44     Triangle tri1 = readTriangle();
45     Triangle tri2 = readTriangle();
46     TwoTriangle(tri1, tri2).print();
47 }
```

答案程序

```
1 public:
2     TwoTriangle(const Triangle & tri1, const Triangle & tri2) :
3         Triangle(tri2), member(tri1)
4     {
5     }
```

文本编辑框

保存和测试对错

4. Polymorphism (easy)

Write three classes (Person, Boy, and Girl) so that you get the desired output.

EXAMPLE OUTPUT

```
Person
Boy
Girl
```

主程序

```
1  #include "source"
2
3  void printClassName(Person * object) {
4      object->print();
5  }
6
7  int main() {
8      Person * person = new Person;
9      printClassName(person);
10
11     person = new Boy;
12     printClassName(person);
13
14     person = new Girl;
15     printClassName(person);
16 }
```

答案程序

```
1  #include <iostream>
2  using namespace std;
3
4  class Person
5  {
6  public:
7      virtual void print() const {
8          cout << "Person" << endl;
9      }
10 };
11
```

```
class Boy : public Person
{
public:
    virtual void print() const {
        cout << "Boy" << endl;
    }
};

class Girl : public Person
{
public:
    virtual void print() const {
        cout << "Girl" << endl;
    }
};
```

文本编辑框

保存和测试对错

5. Exception (medium)

In this exercise, you need to know how and when to throw an exception. Design a class Volume (3维矩阵) that throws the following exception:

- A out_of_range exception, which is thrown whenever a user access an element via an out-of-range index. This exception is thrown by object functions *get* and *set*. This exception is defined by C++ in a library file named <stdexcept>

Note that the elements in the volume are stored in a *vector* of one-dimension.

EXAMPLE INPUT

```
2 3 4

1 2 3 4
2 3 4 5
3 4 5 6

4 5 6 7
5 6 7 8
6 7 8 9

1 1 1
2 3 4
3 3 4
2 4 4
2 3 -2
```

EXAMPLE OUTPUT

```
1
9
caught: out_of_range
caught: out_of_range
caught: out_of_range
```

主程序

```
#include <vector>
#include <stdexcept>
using namespace std;

////////////////////
// VOLUME
//

class Volume
{
private:
     int matrixes; // first dimension
     int rows; // second dimension
```

```
14      int columns; // thrid dimension
15      vector<double> elements;
16
17  public:
18      Volume(int matrixes, int rows, int columns) {
19          this->matrixes = matrixes;
20          this->rows = rows;
21          this->columns = columns;
22          for (int i = 0; i < matrixes * rows * columns;  ++ i) {
23              elements.push_back(0.0);
24          }
25      }
26
27      int size(int dimension) const {
28          switch (dimension) {
29          case 1: return matrixes;
30          case 2: return rows;
31          case 3: return columns;
32          }
33          return 0;
34      }
35
36      double get(int mat, int row, int column) const;
37
38      void set(int mat, int row, int column, double value);
39
40
41  };
42
43  #include "source"
44
45  #include <iostream>
46  using namespace std;
47
48  ////////////////////
49  // READ & PRINT
50  //
51
52  Volume read() {
53      int matrixes, rows, columns;
54      cin >> matrixes >> rows >> columns;
55      Volume volume(matrixes, rows, columns);
56      for (int i = 0; i < matrixes; ++ i) {
57          for (int j = 0; j < rows; ++ j) {
58              for (int k = 0; k < columns; ++ k) {
59                  double value;
60                  cin >> value;
61                  volume.set(i + 1, j + 1, k + 1, value);
62              }
63          }
64      }
65      return volume;
66  }
67
68  void print(const Volume & volume) {
69      int matrixes = volume.size(1);
70      int rows = volume.size(2);
71      int columns = volume.size(3);
72      cout << "(" << matrixes << "," << rows << "," << columns << ")" << endl;
73      for (int i = 0; i < matrixes; ++ i) {
74          for (int j = 0; j < rows; ++ j) {
75              for (int k = 0; k < columns; ++ k) {
76                  cout << " " << volume.get(i + 1, j + 1, k + 1);
77              }
78              cout << endl;
79          }
80          cout << " ----- " << endl;
```

```
    }
}

//////////////////
// MAIN & TEST
//

void test(const Volume & volume) {
    for (int i = 0; i < 5; ++ i) {
        int mat;
        int row;
        int column;
        cin >> mat >> row >> column;
        try {
            double value = volume.get(mat, row, column);
            cout << "value = " << value << endl;
        } catch (out_of_range & ex) {
            cout << "caught: out_of_range" << endl;
        }
    }
}

int main() {
    Volume volume = read();
    test(volume);
}
```

答案程序

```
double Volume::get(int mat, int row, int column) const {
    if (mat <= 0 || mat > matrixes) {
        throw out_of_range("mat");
    }
    if (row <= 0 || row > rows) {
        throw out_of_range("row");
    }
    if (column <= 0 || column > columns) {
        throw out_of_range("column");
    }
    int index = (mat - 1) * rows * columns + (row - 1) * columns + (column - 1);
    return elements[index];
}

void Volume::set(int mat, int row, int column, double value) {
    if (mat <= 0 || mat > matrixes) {
        throw out_of_range("mat");
    }
    if (row <= 0 || row > rows) {
        throw out_of_range("row");
    }
    if (column <= 0 || column > columns) {
        throw out_of_range("column");
    }
    int index = (mat - 1) * rows * columns + (row - 1) * columns + (column - 1);
    elements[index] = value;
}
```

文本编辑框

保存和测试对错

6. Composition (medium)

In this exercise, class *Volume* is implemented by using classes *vector* and *Matrix*, class *Matrix* is implemented by using classes *vector* and *Vector*, and class *Vector* is implemented by using class *vector*.
Please implement a class Volume (a volume is a 3-dimensional matrix) by completing the following object functions:

- **double get(int mat, int row, int col)**
- **void set(int mat, int row, int col, double value)**

EXAMPLE INPUT

```
2 3 4

1 2 3 4
2 3 4 5
3 4 5 6

4 5 6 7
5 6 7 8
6 7 8 9
```

EXAMPLE OUTPUT

```
 1 2 3 4
 2 3 4 5
 3 4 5 6
 ----------
 4 5 6 7
 5 6 7 8
 6 7 8 9
 ----------
```

主程序

```
#include <vector>

using namespace std;


//////////////////
// VECTOR
//

class Vector
{
private:
    vector<double> data;

public:
    Vector(int length) : data(length) {
        for (int i = 0; i < length; ++ i) {
            data[i] = 0;
        }
    }

    double get(int index) {
        return data[index - 1];
    }

    void set(int index, double value) {
        data[index - 1] = value;
    }

    int size() {
        return data.size();
    }
};

//////////////////
// MATRIX
//

class Matrix
{
private:
    vector<Vector> rows;

public:
```

```
43      Matrix(int rowCount, int columnCount) {
44          for (int i = 0; i < rowCount; ++ i) {
45              rows.push_back(Vector(columnCount));
46          }
47      }
48
49      Vector & getRow(int row) {
50          return rows[row - 1];
51      }
52
53      double get(int row, int col) {
54          return getRow(row).get(col);
55      }
56
57      void set(int row, int col, double value) {
58          getRow(row).set(col, value);
59      }
60
61      int size(int dimension) {
62          if (dimension == 1) {
63              return rows.size(); // number of rows
64          }
65          else {
66              return getRow(1).size(); // number of columns
67          }
68      }
69  };
70
71  //////////////////
72  // VOLUME
73  //
74
75  class Volume
76  {
77  private:
78      vector<Matrix> mats;
79
80  public:
81      Volume(int matCount, int rowCount, int columnCount) {
82          for (int i = 0; i < matCount; ++ i) {
83              mats.push_back(Matrix(rowCount, columnCount));
84          }
85      }
86
87      Matrix & getMatrix(int matCount) {
88          return mats[matCount - 1];
89      }
90
91      int size(int dimension) {
92          if (dimension == 1) {
93              return mats.size(); // number of rows
94          }
95          else {
96              return getMatrix(1).size(dimension - 1); // number of columns
97          }
98      }
99
100 #include "source"
101
102 };
103
104 #include <iostream>
105 using namespace std;
106
107 //////////////////
108 // PRINT
```

```
//

void printVector(Vector & v) {
    for (int i = 0; i < v.size(); ++ i) {
        cout << " " << v.get(i + 1);
    }
}

void printMatrix(Matrix & m) {
    for (int i = 0; i < m.size(1); ++ i) {
        printVector(m.getRow(i + 1));
        cout << endl;
    }
}

void printVolume(Volume & vl) {
    for (int i = 0; i < vl.size(1); ++ i) {
        printMatrix(vl.getMatrix(i + 1));
        cout << " ---------- " << endl;
    }
}

//////////////////
// READ
//

void readVector(Vector & v) {
    for (int i = 0; i < v.size(); ++ i) {
        double value;
        cin >> value;
        v.set(i + 1, value);
    }
}

void readMatrix(Matrix & m) {
    for (int i = 0; i < m.size(1); ++ i) {
        readVector(m.getRow(i + 1));
    }
}

void readVolume(Volume & vl) {
    for (int i = 0; i < vl.size(1); ++ i) {
        readMatrix(vl.getMatrix(i + 1));
    }
}

//////////////////
// MAIN
//

int main() {
    int mat, row, col;
    cin >> mat >> row >> col;

    Volume vl(mat, row, col);
    readVolume(vl);

    printVolume(vl);
}
```

答案程序

```
public:
    double get(int mat, int row, int col) {
        return getMatrix(mat).get(row, col);
    }

```

```
    void set(int mat, int row, int col, double value) {
        getMatrix(mat).set(row, col, value);
    }
```

文本编辑框

保存和测试对错

7. Encapsulation of dynamic memory allocation (medium)

In this exercise, you need to know how and when to implement class with member variables of type address. Design a class Volume (3维矩阵) that has the following object functions:

- Volume(int mats, int rows, int cols)
- int size(int dimension) const
- double get(int mat, int row, int col) const
- void set(int mat, int row, int col, double value)

Note that you also need to implement the following object functions to ensure the correctness of your code:

- ~Volume()
- Volume(const Volume &)
- Volume & operator = (const Volume &)

EXAMPLE INPUT

```
2 3 4

1 2 3 4
2 3 4 5
3 4 5 6

4 5 6 7
5 6 7 8
6 7 8 9

1 1 1
2 3 4
2 2 4
2 3 1
1 3 4
```

EXAMPLE OUTPUT

```
value = 1
value = 9
value = 8
value = 6
value = 6
```

主程序

```
////////////////////
// VOLUME
//

class Volume
{
private:
    int matrixes; // first dimension
    int rows; // second dimension
    int columns; // thrid dimension
    double * elements;

public:
    Volume(int matrixes, int rows, int columns) {
        this->matrixes = matrixes;
```

```
        this->rows = rows;
        this->columns = columns;
        int elemCount = matrixes * rows * columns;
        elements = new double[elemCount];
        for (int i = 0; i < elemCount; ++ i) {
            elements[i] = 0;
        }
    }

#include "source"

};

#include <iostream>
using namespace std;

////////////////////
// READ & PRINT
//

Volume read() {
    int matrixes, rows, columns;
    cin >> matrixes >> rows >> columns;
    Volume volume(matrixes, rows, columns);
    for (int i = 0; i < matrixes; ++ i) {
        for (int j = 0; j < rows; ++ j) {
            for (int k = 0; k < columns; ++ k) {
                double value;
                cin >> value;
                volume.set(i + 1, j + 1, k + 1, value);
            }
        }
    }
    return volume;
}

void print(const Volume & volume) {
    int matrixes = volume.size(1);
    int rows = volume.size(2);
    int columns = volume.size(3);
    cout << "(" << matrixes << "," << rows << "," << columns << ")" << endl;
    for (int i = 0; i < matrixes; ++ i) {
        for (int j = 0; j < rows; ++ j) {
            for (int k = 0; k < columns; ++ k) {
                cout << " " << volume.get(i + 1, j + 1, k + 1);
            }
            cout << endl;
        }
        cout << " ----- " << endl;
    }
}

////////////////////
// MAIN & TEST
//

void test(Volume volume) {
    for (int i = 0; i < 5; ++ i) {
        int mat;
        int row;
        int column;
        cin >> mat >> row >> column;
        double value = volume.get(mat, row, column);
        cout << "value = " << value << endl;
    }
}

```

```
int main() {
    Volume volume(1,1,1);
    volume = read();
    test(volume);
}
```

答案程序

```
private:
    int getIndex0(int mat, int row, int col) const {
        return (mat - 1) * rows * columns + (row - 1) * columns + (col - 1);
    }

    void assign0(const Volume & vol) {
        matrixes = vol.matrixes;
        rows = vol.rows;
        columns = vol.columns;
        int elemCount = matrixes * rows * columns;
        elements = new double[elemCount];
        for (int i = 0; i < elemCount; ++ i) {
            elements[i] = vol.elements[i];
        }
    }

public:

    int size(int dimension) const {
        switch (dimension) {
        case 1: return matrixes;
        case 2: return rows;
        case 3: return columns;
        }
        return 0;
    }

    double get(int mat, int row, int col) const {
        return elements[getIndex0(mat, row, col)];
    }

    void set(int mat, int row, int col, double value) {
        elements[getIndex0(mat, row, col)] = value;
    }

    ~Volume() {
        delete [] elements;
    }

    Volume(const Volume & vol) {
        assign0(vol);
    }

    Volume & operator = (const Volume & vol) {
        delete [] elements;
        assign0(vol);
        return *this;
    }
```

文本编辑框

保存和测试对错

8. Input/Output (medium)

In this exercise, class *Matrix* is implemented using class *Volume*, and class *Vector* is implemented using class *Matrix*. Please implement class *Matrix* and write the input and output operator functions for the class.

You need to implement the following object functions:

- **int Matrix::size(dimension) const**
- **double Matrix::get(int row, int col) const**
- **void Matrix::set(int row, int col, double value)**

You also need to implement the following input and output operator functions:

- **istream & operator >> (istream &, Matrix &)**
- **ostream & operator << (ostream &, const Matrix &)**

EXAMPLE INPUT

```
1 2 3 4
2 3 4 5
3 4 5 6

4 5 6 7 8
5 6 7 8 9
6 7 8 9 10
7 8 9 10 11
```

EXAMPLE OUTPUT

```
1 2 3 4
2 3 4 5
3 4 5 6

4 5 6 7 8
5 6 7 8 9
6 7 8 9 10
7 8 9 10 11
```

主程序

```
1  ////////////////////
2  // VOLUME
3  //
4
5  class ExceedingCapacityException {};
6
7  class Volume
8  {
9  private:
10     int mats;
11     int rows;
12     int cols;
13     double elements[10][10][10];
14
15 public:
16     Volume(int mats, int rows, int cols) {
17         if (mats > 10) throw ExceedingCapacityException();
18         if (rows > 10) throw ExceedingCapacityException();
19         if (cols > 10) throw ExceedingCapacityException();
20
21         this->mats = mats;
22         this->rows = rows;
23         this->cols = cols;
24
25         for (int i = 0; i < mats; ++ i) {
26             for (int j = 0; j < rows; ++ j) {
27                 for (int k = 0; k < cols; ++ k) {
28                     elements[i][j][k] = 0;
29                 }
30             }
31         }
32     }
33
34     double get(int mat, int row, int col) const {
35         return elements[mat - 1][row - 1][col - 1];
36     }
```

```
37
38      void set(int mat, int row, int col, double value) {
39          elements[mat - 1][row - 1][col - 1] = value;
40      }
41
42      int size(int dimension) const {
43          switch (dimension) {
44          case 1: return mats;
45          case 2: return rows;
46          case 3: return cols;
47          }
48          return 0;
49      }
50
51  };
52
53  //////////////////
54  // MATRIX
55  //
56
57  class Matrix
58  {
59  private:
60      Volume volume;
61
62  public:
63      Matrix(int rows, int cols) : volume(1, rows, cols) {
64      }
65
66      int size(int dimension) const;
67      double get(int row, int col) const;
68      void set(int row, int col, double value);
69  };
70
71  #include "source"
72
73  //////////////////
74  // VECTOR
75  //
76
77  class Vector
78  {
79  private:
80      Matrix matrix;
81
82  public:
83      Vector(int size) : matrix(1, size) {
84      }
85
86      int size() {
87          return matrix.size(2);
88      }
89
90      double get(int index) const {
91          return matrix.get(1, index);
92      }
93
94      void set(int index, double value) {
95          matrix.set(1, index, value);
96      }
97
98  };
99
100 //////////////////
101 // TEST
102 //
103
```

```
int main() {
    Matrix mat1(3, 4);
    Matrix mat2(4, 5);
    cin >> mat1 >> mat2;
    cout << mat1 << endl << mat2;
}
```

答案程序

```
int Matrix::size(int dimension) const {
    return volume.size(dimension + 1);
}

double Matrix::get(int row, int col) const {
    return volume.get(1, row, col);
}

void Matrix::set(int row, int col, double value) {
    volume.set(1, row, col, value);
}

#include <iostream>
using namespace std;

istream & operator >> (istream & in, Matrix & matrix) {
    int rows = matrix.size(1);
    int cols = matrix.size(2);
    for (int i = 0; i < rows; ++ i) {
        for (int j = 0; j < cols; ++ j) {
            double value;
            in >> value;
            matrix.set(i + 1, j + 1, value);
        }
    }
    return in;
}

ostream & operator << (ostream & out, const Matrix & matrix) {
    int rows = matrix.size(1);

    int cols = matrix.size(2);
    for (int i = 0; i < rows; ++ i) {
        for (int j = 0; j < cols; ++ j) {
            out << matrix.get(i + 1, j + 1) << " ";

        }
        out << endl;
    }
    return out;
}
```

文本编辑框

保存和测试对错

9. People (hard)

Please add the necessary functions.

EXAMPLE OUTPUT

```
 --- add people to list ---
object allocated:Boy
object allocated:Girl
object allocated:Boy
object allocated:Girl
 --- copy constructor ---
object allocated:Boy
object allocated:Girl
object allocated:Boy
object allocated:Girl
```

```
 --- add people to list ---
object allocated:Boy
object allocated:Girl
object allocated:Boy
object allocated:Girl
 --- print ---
Boy+Girl+Boy+Girl+
 --- assignment operator ---
object released
object released
object released
object released
object allocated:Boy
object allocated:Girl
object allocated:Boy
object allocated:Girl
object allocated:Boy
object allocated:Girl
object allocated:Boy
object allocated:Girl
 --- print ---
Boy+Girl+Boy+Girl+Boy+Girl+Boy+Girl+
object released
object released
object released
object released
object released
object released
object released
object released
object released
object released
object released
object released
object released
object released
object released
object released
object released
```

主程序

```
#include <iostream>
#include <string>
using namespace std;

///////////////
// PEOPLE
//

class People
{
public:
    People() {
        cout << "object allocated";
    }

    virtual ~People() {
        cout << "object released\n";
    }

    virtual string toString() = 0;
    virtual People * getNewInstance() = 0;
};

class Boy : public People
{
public:
    Boy() {
        cout << ":Boy\n";
    }

    string toString() {
        return string("Boy");
    }
```

```
34
35      People * getNewInstance() {
36          return new Boy();
37      }
38  };
39
40  class Girl : public People
41  {
42  public:
43      Girl() {
44          cout << ":Girl\n";
45      }
46
47      string toString() {
48          return string("Girl");
49      }
50
51      People * getNewInstance() {
52          return new Girl();
53      }
54  };
55
56  ////////////////
57  // PEOPLE LIST
58  //
59
60  class PeopleList
61  {
62  private:
63      int peopleCount;
64      int listCapacity;
65      People * * list;
66
67  public:
68      PeopleList() {
69          peopleCount = 0;
70          listCapacity = 2;
71          list = new People * [listCapacity];
72      }
73
74      void print() const {
75          for (int i = 0; i < peopleCount; ++ i) {
76              cout << (*this)[i] << "+";
77          }
78          cout << endl;
79      }
80
81      void addBoy() {
82          this->addToLast0(new Boy());
83      }
84
85      void addGirl() {
86          this->addToLast0(new Girl());
87      }
88
89  #include "source"
90
91  };
92
93  ////////////////
94  // TEST
95  //
96
97  void addPeopleToList(PeopleList & list, int pairs) {
98      cout << " --- add people to list --- " << endl;
99      for (int i = 0; i < pairs; ++ i) {
100         list.addBoy();
```

```
        list.addGirl();
    }
}

int main() {
    PeopleList list1;
    addPeopleToList(list1, 2);

    cout << " --- copy constructor --- " << endl;
    PeopleList list2 = list1;

    addPeopleToList(list2, 2);

    cout << " --- print --- " << endl;
    list1.print();

    cout << " --- assignment operator --- " << endl;
    list1 = list2;

    cout << " --- print --- " << endl;
    list1.print();
}
```

答案程序

```
public:
    string operator [] (int index) const {
        return list[index]->toString();
    }

    ~PeopleList() {
        cleanUp0();
    }

    PeopleList(const PeopleList & pList) {
        assign0(pList);
    }

    PeopleList & operator = (const PeopleList & pList) {
        cleanUp0();
        assign0(pList);
        return *this;
    }

private:
    void addToLast0(People * people) {
        ensureCapacity0(peopleCount + 1);
        list[peopleCount] = people;
        ++ peopleCount;
    }

    void cleanUp0() {
        for (int i = peopleCount - 1; i >= 0; -- i) {
            delete list[i];
        }
        delete [] list;
    }

    void ensureCapacity0(int capacity) {
        while (listCapacity < capacity) {
            listCapacity *= 2;
        }
        People * * old = list;
        list = new People * [listCapacity];
        for (int i = 0; i < peopleCount; ++ i) {
            list[i] = old[i];
        }
        delete [] old;
```

```
    }

    void assign0(const PeopleList & pList) {
        listCapacity = pList.listCapacity;
        list = new People * [listCapacity];
        peopleCount = 0;
        for (int i = 0; i < pList.peopleCount; ++ i) {
            addToLast0(pList.list[i]->getNewInstance());
        }
    }
```

文本编辑框

保存和测试对错

测试

请使用以下快捷键盘操作在程序框之间拷贝程序

- Control + c 复制
- Control + v 粘贴

主程序

```
#include "source"
```

答案程序

```

```

文本编辑框

测试数据

保存和测试

测试输出