Примерна тема за второ контролно по ООП

Задача 1. Да се реализира шаблон за елемент с етикет **Labelled**, който съдържа член-данна от произволен тип T и етикет – низ с до 100 символа. За класа да се реализират:

- подходящи конструктори, селектори и мутатори;
- функции за вход и изход;
- оператор за сравнение (==), като два елемента с етикет съвпадат ако съвпадат самите елементи и етикетите им.

Задача 2.

а) Да се дефинира абстрактен клас **Container**, дефиниращ интерфейс на контейнер, съдържащ числа с плаваща запетая (double). Класът да дефинира следните методи:
   - *bool member (double x),* проверяващ дали дадено число е елемент на структурата
   - *double operator [] (int i)*, намиращ *i*-тия елемент на структурата. Номерацията на елементите зависи от специфичната структура
   - *int count ()*, намиращ броя на елементите в структурата
   - *void write ()*, извеждащ на конзолата елементите на структурата.

б) Да се дефинира наследник **Vector** на клас **Container**, реализиращ вектор от числа с плаваща запетая. Размерът и елементите на вектора се задават при конструиране на обекта. Номерацията на елементите е естествена. Печатането на елементите е последователно на едни ред, разделени със запетая и оградени със скоби (например: (1,2,3)).

в) Да се дефинира наследник **Matrix** на клас **Container**, реализиращ квадратна матрица от числа с плаваща запетая. Размерът и елементите на матрицата се задават при конструиране на обекта. Номерацията на елементите на матрицата е следната: Ако големината на матрицата е N, то елементът (a_{ij}), $0<=i<N$, $0<=j<N$ има пореден номер $i*N+j$.

г) Да се реализира функция *bool intersection (Container *arr[], int n)*, която по масив от указатели към *n* контейнера проверява дали има такова число *x*, което да се съдържа във всяка една от структурите.

Второ контролно по ООП

I поток, спец. Информатика, 29.05.2010

Вариант Б

Задача 1. Да се реализира шаблон за номериран елемент **Numbered**, който съдържа член-данна от произволен тип *T* и число – пореден номер на елемента. Да се реализират:

- подходящи селектори и мутатори;
- функции за вход и изход;
- оператор за сравнение (<), който сравнява елементите по номер;
- функция, която по масив от номерирани елементи извежда всички елементи, чиито номера са в даден интервал;
- функция, която по масив от *n* номерирани елемента ги подрежда по номер и след това ги преномерира с числата от 1 до *n*.

Задача 2.

д) Да се дефинира абстрактен клас **Container**, дефиниращ интерфейс на контейнер, съдържащ числа с плаваща запетая (double). Класът да дефинира следните методи:
   - *bool member (double x),* проверяващ дали дадено число е елемент на структурата
   - *double operator [] (int i)*, намиращ *i*-тия елемент на структурата. Номерацията на елементите зависи от специфичната структура
   - *int count ()*, намиращ броя на елементите в структурата
   - *void read ()* и *void write ()*, прочитащ от клавиатурата и съответно извеждащ на конзолата елементите на структурата.

е) Да се дефинира наследник **Vector** на клас **Container**, реализиращ вектор от числа с плаваща запетая. Размерът на вектора се задава при конструиране на обекта. Номерацията на елементите е естествена. Печатането на елементите е последователно на едни ред, разделени със запетая и оградени със скоби (например: (1,2,3)).

ж) Да се дефинира наследник **Matrix** на клас **Container**, реализиращ квадратна матрица от числа с плаваща запетая. Размерът на матрицата се задава при конструиране на обекта. Номерацията на елементите на матрицата е следната: Ако големината на матрицата е N, то елементът (a_{ij}), *0<=i<N, 0<=j<N* има пореден номер *i*N+j*.

з) Да се реализира функция *bool intersection (Container *arr[], int n)*, която по масив от указатели към нескаларни СД с дължина *n* елемента проверява дали има такова число *x*, което да се съдържа във всяка една от структурите.

Второ контролно по ООП

I поток, спец. Информатика, 29.05.2010

Вариант А

Задача. Напишете какво ще отпечата на екрана следната програма:

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

class Base {
     protected:
          int a;
     public:
          Base(int _a=1) : a(_a)
          { cout << "Base()\n"; }
          Base(Base const& base) : a(base.a)
          { cout << "Base(Base const&)\n"; }
     protected:
          void increase()
          { a += 3; }
     public:
          virtual void print()
          { cout << "a = " << a << endl; }
};

class Derived1 : public Base {
     private:
          double d;
     public:
          Derived1(double _d=2) : d(_d)
          { cout << "Derived1()\n"; }
          Derived1(Derived1 const& derived1) : d(derived1.a +
derived1.d)
          { cout << "Derived1(Derived1 const&)\n"; }
     protected:
          virtual void increase()
          { Base::increase();
            d *= 2;
          }
     public:
          void print()
          { cout << "d = " << d << endl; }
};

class Derived2 : public Derived1 {
     public:
          Derived2(int _a, double _d) : Derived1(_d)
          { a = _a; cout << "Derived2(int,double)\n"; }
          Derived2(Derived2 const& derived2)
          { cout << "Derived2(Derived2 const&)\n";
            increase();
         }
```

```
     public:
          void increase()
          { Derived1::increase(); a++; }
          void print() {
            Base::print();
            increase();
            Derived1::print(); }
};

void f(Derived2 d2) {
     d2.print();
     d2.increase();
}

void g(Base& b) {
     b.print();
}

int main() {
     Derived1* d1 = new Derived2(2,3.5);
     Base* bb = new Derived1(*d1);
     Derived2 d2(4,1.5);
     d1->print();
     bb->print();
     d2.print();
     f(d2);
     g(*bb);
     g(*d1);
     g(d2);
     return 0;
}
```

Второ контролно по ООП

I поток, спец. Информатика, 29.05.2010

Вариант Б

Задача. Напишете какво ще отпечата на екрана следната програма:

```
#include <iostream>

using namespace std;

class Base {
     protected:
          double a;
     public:
          Base(double _a=1) : a(_a)
          { cout << "Base()\n"; }
          Base(Base const& base) : a(base.a)
          { cout << "Base(Base const&)\n"; }
     protected:
          void increase()
          { a *= 3; }
     public:
          virtual void print()
          { cout << "a = " << a << endl; }
};

class Derived1 : public Base {
     private:
          int d;
     public:
          Derived1(int _d=2) : d(_d)
          { cout << "Derived1()\n"; }
          Derived1(Derived1 const& derived1) : d(derived1.a +
derived1.d)
          { cout << "Derived1(Derived1 const&)\n"; }
     public:
          virtual void increase()
          { Base::increase();
            d += 2;
          }
     public:
          void print()
          { cout << "d = " << d << endl; }
};

class Derived2 : public Derived1 {
     public:
          Derived2(double _a, int _d) : Derived1(_d)
          { a = _a; cout << "Derived2(double,int)\n"; }
          Derived2(Derived2 const& derived2)
          { cout << "Derived2(Derived2 const&)\n";
            increase();
         }
```

```
     public:
          void increase()
          { Derived1::increase(); a+=0.5; }
          void print()
          { Base::print();
            increase();
            Derived1::print(); }
};

void f(Derived1 d1) {
     d1.print();
     d1.increase();
}

void g(Base* b) {
     b->print();
}

int main() {
     Derived2 d2(2.5,6);
     Derived1* d1 = new Derived2(1.5,3);
     Base* bb = new Derived2(d2);

     d1->print();
     bb->print();
     d2.print();

     f(*d1);
     g(bb);
     g(d1);
     g(&d2);

     return 0;
}
```