UMEÅ UNIVERSITET
Institutionen för datavetenskap

2001-06-05

Thomas Johansson
Kalle Prorok

Lösningsförslag till Tentamen i Objektorienterad programmering för ingenjörer (TDBB09) 2001-05-28, kl 09.00 - 15.00

Lokal: Skrivsal 3, Östra Paviljongerna, Ålidbacken 23

Hjälpmedel: inga (förutom penna, suddgummi och matsäck)

Kursutvärdering sker via webben**.**

Tentan beräknas vara **färdigrättad** ca 7 juni och hämtas de närmaste dagarna hos Kalle och sedan på studentexpeditionen.

Börja varje uppgift på ett nytt blad.
Använd bara en sida av papperet.
Skriv namn på varje blad.
Lägg uppgifterna i rätt ordning.
Observera att uppgifterna inte är numrerade efter svårighetsgrad och kom ihåg att även delvis lösta uppgifter kan ge poäng.
Kommentera noga all källkod så att vi kan se hur du har tänkt.
Skriv tydligt.

Maxpoäng är 40, fördelade på 7 frågor.

Betygsgränser:

5	32p
4	26p
3	20p

Fråga om det är något som är oklart ! Vi kommer till skrivsalen ca **11.00**

Lycka till!

Thomas Johansson
tel 12 71 16 / 786 62 59

Kalle Prorok
tel 070-3 33 35 37

Uppgift 1 – STL (9p)
a)

…
```
vector<vector<int>> vi;
```
…

Ovanstående rad i ett program innehåller ett fel på grund av en syntaktisk groda, vilken och vad ska man göra åt den? Följande felmeddelande erhålls : Cannot generate template specialization from vector<T,Allocator>. (2p)

b) Beskriv vilken designstrategi man haft när man organiserat algoritmbiblioteket (STL) (3p)

c) Följande program läser in ett okänt antal tal till en lista m.h.a push_back och skriver sedan ut dem. Ange vad som ska stå istället för XXX (mer än 3 bokstäver behövs)! (2p)

```
#include <list>
#include <iostream>
using namespace std;
int main(int argc, char* argv[])
{
    list<int> li;
    int i;
    do {
         cin >> i;
         li.push_back(i);
    } while(i != 0);
    XXX lii;
    for(lii = li.begin(); lii != li.end(); lii++)
       cout << *lii << " ";
 return 0;
}
```

d) Istället för for-loopen i c) kunde man använt copy. Beskriv och visa hur! (2p)

1. *a) Kompilatorn tolkar >> som shift-right enligt maximal munch-principen. Lägg in t.ex. ett mellanslag.*
   b) Biblioteket är separerat i en 3D rymd med algoritmer, datastrukturer och datatyper. Tack vare templates behövs nästan bara en version av varje.
   c) list<int>::iterator
   d) copy(li.begin(),li.end(),ostream_iterator<int>(cout," "));

Uppgift 2 – OOA/D (6p)

a) Varför blir det ofta förseningar i projekten och hur kan OO bidra till att reducera dessa problem? (4p)

b) Vilka 5 relationer brukar vi prata om? (2p)

2. *a) Komplexa system, Ingen överblick, Ingen riktig förståelse, Big-bang principen. OO bidrar med strukturering som hanterar komplexitet och ger överblick. Modelleringen ger förståelse och Inkrementell systemutveckling undviker BB.*
   b) Arv, aggregat, association,användning, klass-objekt (instansiering)

Uppgift 3 – Exceptions (undantag) (4p)

a) Beskriv (i grova drag) hur standard-exceptions är strukturerade och vad man kan tänkas ha för nytta av denna struktur. (2p)

b) När man bokat upp en resurs (fil, skrivare, minne etc) bör man tänka på hur exceptions används; beskriv (med ett kort exempel) hur man bör göra och varför ! (2p).

3. *a) I en arvshierarki med* `exception` *som (abstrakt) basklass och* `logic_error` *respektive* `runtime_error` *som delträd med bl.a.* `range_error`*. Används för att ta hand om fel på lämplig nivå genom att ange vilken grupp av fel man vill ta hand om via arvsmekanismen.*

   b) Man bör se till att resursen återlämnas/avbokas i händelse av fel. Exception-hanteringen innebär att man anropar destruktorer på lokala objekt om en exception aktiveras så man bör göra städningen i destruktorn; vilket görs t.ex av ostream-classen (som gör close):

```
void use_file(const char *fn)//felsäker version
{
   ostream f(fn); // destruktorn på f anrop vid ev. fel
   use_vector();
}
```

Lösningsförslag uppgift 4 – 8 (Thomas J)

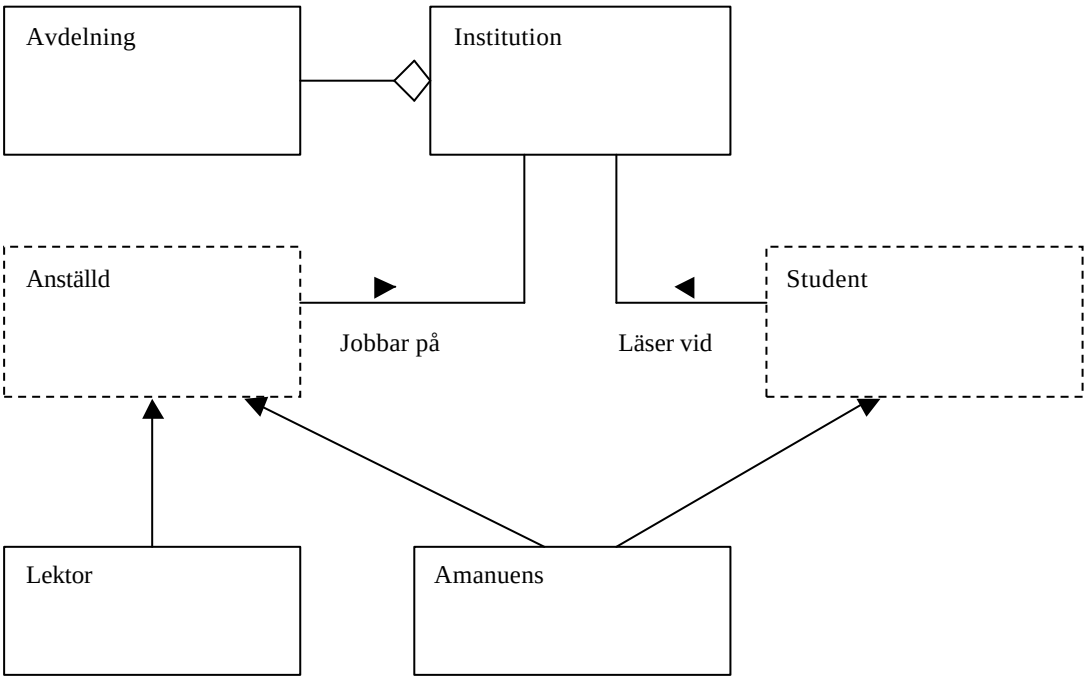
Uppgift 4 – Java (3p)

a) Java har utsatts för en hel del kritik främst i ett avseende, vilket ? (1p)
b) Nämn en fördel med Java jämfört med C++ när det gäller ett av de största problemen med pekarprogrammering. (1p)
c) Java anses vara mer typsäkert än C++, vad menas med det ? (1p)
   1) Bättre typkontroll av parametrar vid anrop av metoder
   2) Bättre kontroll av parametrar till makron
   3) Bättre kontroll av vilka typer som får skriva Java-program

a) Java har åtminstone tidigare varit betydligt långsammare än t.ex. C++
b) Java har inga pekare ! Java har dessutom automatisk skräpsamling, så att frigjort minne kan återanvändas av systemet
c) Alternativ 1 (även om alt 3 kanske får en del att undvika Java…)

Uppgift 5 – Klassdiagram (4p)

Rita ett klassdiagram för institutionen för datavetenskap. Du får använda vilka klasser du vill bara de har en anknytning till institutionen. Du måste ha med minst fyra klasser varav minst en abstrakt. Dessutom måste det finnas minst ett arv, en association och ett aggregat. Observera att du inte skall skriva någon kod i denna uppgift.



Uppgift 6 – C++ (8p)
Implementera (skriv kod för) en matris-klass. Klassen skall representera matriser av storleken 3x3 och följande operatorer skall finnas:

Addition matris + matris (exempel M + M) ”motsvarande element adderas”
Addition matris + skalär (exempel M + 7.4) ”skaläre adderas till samtliga element””
Addition skalär + matris (exempel 7.4 + M) – ” -
Utskrift med <<
Defaultkonstruktor som skapar en identitetsmatris (alla element är 0 utom de på diagonalen som är 1:or)
Initieringskonstruktor
Tilldelningsoperator

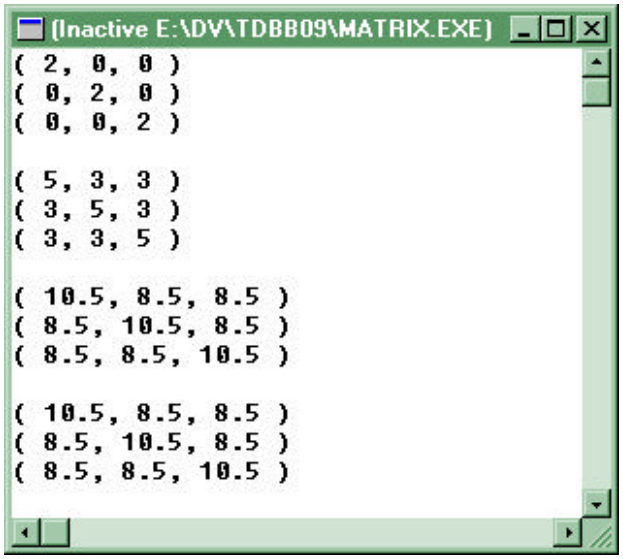
Det ska gå att kedja additioner. Du ska använda dynamisk minneshantering med new.

```
#include <iostream.h>

class Matrix
{
	private:

	double ** data;

	public:

	// default constructor,
	// initializes to an identity matrix
	Matrix()
	{
		data = new double*[3];
		for (int i = 0; i < 3; i++)
		{
			data[i] = new double[3];
			for (int j = 0; j < 3; j++)
			{
				if (i == j)
					data[i][j] = 1.0;
				else
					data[i][j] = 0.0;
			}
		}
	}

	// Copy constructor
	Matrix(const Matrix &m)
	{
```

```
	data = new double*[3];
	for (int i = 0; i < 3; i++)
	{
		data[i] = new double[3];
		for (int j = 0; j < 3; j++)
		{
			data[i][j] = m.data[i][j];
		}
	}
}

// output operator has to be a friend
// (we should have had index operators !)
friend ostream& operator<<(ostream &os, Matrix &m);

// Assignment, with chaining
Matrix & operator=(Matrix &m)
{
	if (this != &m)
	{
		for (int i = 0; i < 3; i++)
		{
			delete [] data[i];
		}
		delete [] data;

		data = new double*[3];
		for (i = 0; i < 3; i++)
		{
			data[i] = new double[3];
			for (int j = 0; j < 3; j++)
			{
				data[i][j] = m.data[i][j];
			}
		}
	}

	return *this;
}

// Addition of matrices, eg. A + B
Matrix operator+(Matrix &m)
{
	Matrix temp;

	for (int i = 0; i < 3; i++)
		for (int j = 0; j < 3; j++)
```

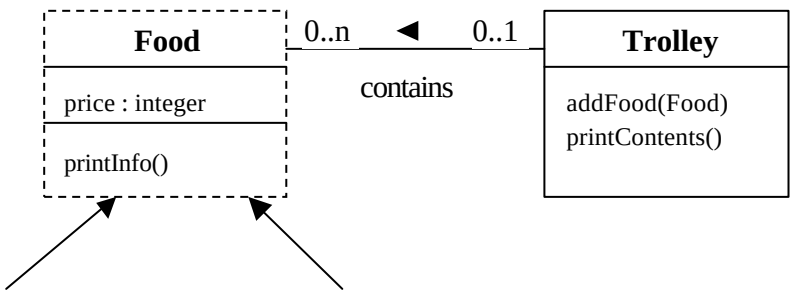
```
                    temp.data[i][j] = this->data[i][j] +
                                      m.data[i][j];

          return temp;
     }

     // Matrix plus scalar
     Matrix operator+(double d)
     {
          Matrix temp;

          for (int i = 0; i < 3; i++)
               for (int j = 0; j < 3; j++)
                    temp.data[i][j] = this->data[i][j] + d;

          return temp;
     }

};


ostream& operator<<(ostream &os, Matrix &m)
{
     for (int i = 0; i < 3; i++)
     {
          os << "( " << m.data[i][0];
          for (int j = 1; j < 3; j++)
          {
               os << ", " << m.data[i][j];
          }
          os << " )" << endl;
     }

     return os;
}

// Addition of scalar plus matrix,
// uses M + s to compute s + M
// Doesn't have to be friend
Matrix operator+(double d, Matrix &m)
{
     return m + d;
}

int main(void)
{
```

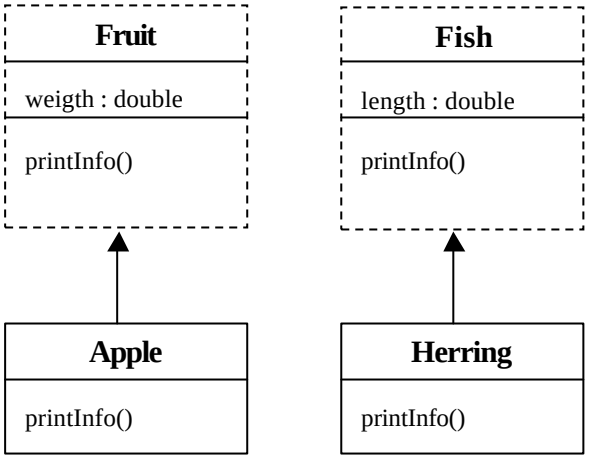
```
        Matrix a;
        Matrix b;

        Matrix c = a + b + 3;
        Matrix d;
        Matrix e = d = 5.5 + c;

        cout << (a + b) << endl;
        cout << c << endl;
        cout << d << endl;
        cout << e << endl;

}
```



Uppgift 7 – Mer C++ (6p)

Implementera (dvs skriv kod för) nedanstående klassdiagram. Bifoga dessutom ett testprogram som lägger in två äpplen och en sill i kundvagnen och sedan skriver ut hela innehållet i den.





Streckade rutor indikerar att klassen skall vara abstrakt. Skriv lämplig(a) konstruktor(er). Kom ihåg att använda dynamisk bindning så att rätt information skrivs ut! Pilen ovanför ordet ’contains’ betyder i vilken riktning denna relation skall läsas, dvs ’Trolley contains Food’.

```
#include <iostream.h>

class Food
{
	private:
		int price;

	protected:
		Food(int price)
		{
			this->price = price;
		}

	public:
		virtual void printInfo()
		{
			cout << "Price : " << price << endl;
		}
};

class Fruit : public Food
{
	private:
		int weight;

	protected:
		Fruit(int price, int weight) : Food(price)
		{
			this->weight = weight;
		}

		virtual void printInfo()
```

```
            {
                  Food::printInfo();
                  cout << "weight : " << weight << endl;
            }
};

class Fish : public Food
{
      private:
            int length;

      protected:
            Fish(int price, int length) : Food(price)
            {
                  this->length = length;
            }

            virtual void printInfo()
            {
                  Food::printInfo();
                  cout << "weight : " << length << endl;
            }
};

class Apple : public Fruit
{
      public:
            Apple(int price, int weight) : Fruit(price, weight)
            {
            }

            virtual void printInfo()
            {
                  cout << "An apple : " << endl;
                  Fruit::printInfo();
            }
};

class Herring : public Fish
{
      public:
            Herring(int price, int length) : Fish(price, length)
            {
            }

            virtual void printInfo()
            {
                  cout << "A herring : " << endl;
                  Fish::printInfo();
            }
};


class Trolley
{
      private:
            Food* foods[100];
```

```
            int nFoods;

      public:
            Trolley()
            {
                  nFoods = 0;
            }

            void addFood(Food* f)
            {
                  if (nFoods < 100)
                  {
                        foods[nFoods] = f;
                        nFoods++;
                  }
            }

            void printContents()
            {
                  for (int i = 0; i < nFoods; i++)
                  {
                        foods[i]->printInfo();
                  }
            }
};

int main(void)
{
      Trolley t;
      Apple * a1 = new Apple(2, 6);
      Apple * a2 = new Apple(3, 7);
      Herring * h = new Herring(12, 65);

      t.addFood(a1);
      t.addFood(a2);
      t.addFood(h);

      t.printContents();

   delete a1, a2, h;

}
```