Mälardalens Högskola
Thomas Larsson
Tel. 021-101514

2008-12-03

Tentamen

CDT202
Objektorienterad programutveckling med C++

7 januari 2009 kl. 14.30–19.30

Denna tentamen omfattar 6 uppgifter och totalt 36 poäng.
För godkänt krävs cirka 50 procent av maximal poäng.

Alla svar skall motiveras
och om förutsättningar saknas
skall rimliga antaganden göras.

Inga hjälpmedel är tillåtna.

Lycka till!

Thomas

1. KLASSER OCH OBJEKT (6p)

   Definiera följande begrepp inom objektorienterad programmering

   (a) klass

   (b) objekt

   (c) överlagring

   (d) arv

   (e) interface

   (f) polymorfism

2. FILHANTERING (6p)

   Skriv en funktion som räknar antalet förekomster av de olika bokstäverna i det engelska alfabetet (A - Z, a - z) i en given textfil. Namnet på textfilen skall ges som en inparameter. Resultatet av operationen skall sparas i en array (som också är en parameter till funktionen). Funktionsprototypen är redan given enligt följande:

   ```
   int countLetters(string & filename, int result[26]);
   ```

   Låt returvärdet visa om operationen lyckades eller ej. Skriv även ett huvudprogram som anropar funktionen och sedan skriver ut resultatet på ett överskådligt sätt. Nedan visas ett exempel på hur resultatet skulle kunna presenteras.

```
a:  7        n:  3
b:  3        o:  6
c:  1        p:  5
d:  2        q:  9
e: 11        r: 12
f:  0        s:  8
g:  1        t:  9
h:  3        u:  7
i:  7        v:  2
j:  1        w:  8
k:  3        x:  0
l:  5        y:  1
m:  2        z:  1
```

3. OPERATORÖVERLAGRING (6p)

Klassen `Date` används för att representera datum. Nedan visas en del av denna klass.

```
class Date {
private:
    int year, month, day;

    bool leapYear(void) { // ger true om året är ett skottår
        return ((year%4==0 && year%100!=0) || year%400==0);
    }
public:
    Date(int y=1, int m=1, int d=1) : year(y),month(m),day(d) { }

    void setDate(int y, int m, int d) {
        year = y;  month = m;  day = d;
    }

    Date get(void) {
        return Date(year, month, day);
    }
};
```

Utöka klassen `Date` genom att du implementerar en algoritm för att beräkna skillnaden eller differensen i antal dagar mellan två givna datumobjekt. Använd dig av operatoröverlagring. Följande huvudprogram visar ett exempel på hur det kan se ut när man använder den operator som du har överlagrat:

```
void main(void) {
    Date d1(2007, 07, 31), d2(2009, 03, 04);
    int days = d2 - d1;
    cout << days << endl; // ger utskriften: 582
}
```

4. ARV (6p)

Ange den utskrift som erhålls då följande program körs:

```
class Date {
public:
    Date(void) { cout << "Constructor: Date" << endl; }
    ~Date(void) { cout << "Destructor: Date" << endl; }
};

class Person {
private:
    Date birth;
public:
    Person(void) { cout << "Constructor: Person" << endl; }
    ~Person(void) { cout << "Destructor: Person" << endl; }
};

class Employee : public Person {
private:
    Date employment;
public:
    Employee(void) { cout << "Constructor: Employee" << endl; }
    ~Employee(void) { cout << "Destructor: Employee" << endl; }
};

void foo(void) {
    Employee e[2];
}

void main(void) {
    foo();
}
```

5. DYNAMISK BINDNING (6p)

Studera nedanstående program:

```
#define PI 3.141592265

class Shape {
public:
    double area(void) {return 0.0; }
    double volume(void) {return 0.0; }
};

class Sphere : public Shape {
protected:
    double radius;
public:
    Sphere(double r) : radius(r) { }
    double area(void) { return 4.0 * PI * radius * radius; }
    double volume(void) { return 4.0 * PI * radius*radius*radius / 3.0; }
};

class Cylinder : public Shape {
protected:
    double radius, height;
public:
    Cylinder(double r, double h) : radius(r), height(h)  { }
    double area(void) { return 2.0 * PI * radius * (radius + height); }
    double volume(void) { return PI * radius * radius * height; }
};

class ShapeContainer {
    vector<Shape> vec;
public:
    void addShape(Shape s) { vec.push_back(s); }

    void calculate(double & totalArea, double & totalVolume) {
        totalArea = totalVolume = 0.0;
        for (unsigned int i = 0; i < vec.size(); i++) {
            totalArea += vec[i].area();
            totalVolume += vec[i].volume();
        }
    }
};
```

```
void main(void) {
    ShapeContainer shapes;
    double area, volume;
    shapes.addShape(Sphere(2.0));
    shapes.addShape(Cylinder(5.0, 3.0));
    shapes.addShape(Sphere(4.5));
    shapes.addShape(Cylinder(1.5, 9.0));
    shapes.calculate(area, volume);
    cout << "Total area: " << area << endl;
    cout << "Total volume: " << volume << endl;
}
```

Ett antal fel har dessvärre smugit sig in i koden som gör att den dynamiska bindningen inte fungerar då medlemsfunktionerna `area` och `volume` anropas inuti loopen i medlemsfunktionen `calculate`. Visa vilka ändringar som måste göras i programmet för att den dynamiska bindningen skall fungera.

6. TEMPLATES (6p)

Betrakta följande enkla funktion för att finna det största talet av två givna heltal:

```
int max(int a, int b) { return (a > b) ? a : b; }
```

Visa hur man kan skriva om denna funktion med hjälp av *templates* så att koden blir oberoende av datatypen på talen. Följande program skall sedan fungera på förväntat sätt:

```
void main(void) {
    int a = 5, b = 7, c;
    float r = 0.53f, s = 0.98f, t;
    double u = 10.23, v = 7.77, x;
    char * s1 = "abba", * s2 = "bruce", *s3;

    c = getMax<int>(a, b);
    t = getMax<float>(r, s);
    x = getMax<double>(u, v);
    s3 = getMax<char *>(s1, s2);
}
```