Mälardalens Högskola
Thomas Larsson
Tel. 021-101514

2009-05-20

Tentamen

CDT202
Objektorienterad programutveckling med C++

2 juni 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. ALLMÄNT (6p)

   Besvara följande frågor utförligt:

   (a) Vilka fördelar har objektoriented programmering jämfört med andra typer av programmeringsspråk?

   (b) C++ är ett programmeringspråk som stöder ett flertal olika slags programmeringsparadigmer. Vilka då och hur kan man påstå detta?

2. FIL- OCH STRÄNGHANTERING (6p)

   Lös följande uppgifter där indata är namnet på en textfil:

   (a) Skriv en funktion som räknar antalet ord som finns i texten.

   (b) Implementera en funktion som räknar antalet förekomster av varje unikt ord i textfilen. Utdata skall vara en vektor som innehåller denna information.

   (c) Skriv en funktion som skriver ut en sorterad lista över alla de unika orden som förekommer i texten. Sorteringen skall göras med avseende på ordens användningsfrekvens i texten.

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 som räknar ut vilket datum som ligger ett givet antal dagar längre fram i tiden. 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(2009, 9, 30), d2;
    d2 = d1 + 42;
    // Raden nedan ger i detta fall utskriften: 2009-11-11
    d2.print();
}
```

4. ARV (6p)

   Förklara i detalj skillnaderna mellan privat arv, skyddat arv samt publikt arv.

5. 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);
}
```

6. 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.