Övningar

Dag 3 – Dynamiskt minne och header-filer

Dessa övning går ut på att lära sig de grundläggande principerna med pekare och dynamiskt allokerat minne, samt att dela upp och länka ett projekt.

Övning 1

Öppna projektet som hette FirstClass (Övning 2 under dag 2), där du skapade en klass `Triangle`. Skriv om exemplet så att du skapar minne dynamiskt.

1) Skriv om funktionen `makeTriangle` så att denna dynamiskt skapar en ny struct med `new`, samt returnerar en pekare till structen i stället för en kopia.

2) Skriv om `getArea()`, så att den kan ta emot en pekare till strukten.

3) I `main`-funktion, anropa `makeTriangle` så att du tar emot en pekare som returvärde, samt använd pekaren enligt exemplet så att du kan skriva ut dess värden. Nu ska även anropet till `getArea` fungera. Däremot måste vi ändra till `tstr->base = 20.0;`

4) Gör samma sak, där du skapar ett nytt objekt av `Triangle`

5) Se till att både structen och objektet är avallokerat innan `main`-funktionen avslutas.

Övning 2

I Triangle-klassen, skapa en destruktur som innehåller följande utskrift.

```
cerr << "Triangle object deleted" << endl;
```

Kör programmet och verifiera att du får en utskrift. Testa sedan att lägga till följande kodsnutt och kör programmet.

```
for (int i=0; i < 5; i++)
{
    Triangle tmp(2*i, i);
    cout << tmp.getArea() << endl;
}
```

Ändra till följande.

```
for (int i=0; i< 5; i++)
{
    Triangle* tmp = new Triangle(2*i, i);
    cout << tmp->getArea() << endl;
    delete tmp;
}
```

Och följande.

```
Triangle* tmp = new Triangle(0, 0);
for (int i=0; i< 10; i++)
{
    tmp->setBase(i);
    tmp->setHeight(2*i);
    cout << tmp.getArea() << endl;
}
delete tmp;
```

Vad är för- och nackdelar med det senaste exemplet jämfört med de tidigare exemplen. Vilket är att föredra?

Några saker kan vi alltså notera om **destruktorn**

1) Destruktorn har samma namn som klassen med ett inledande ~.

2) Destruktorn anropas automatiskt när objektet avallokeras.

3) Destruktorn returnerar inget

Övning 3

Öppna projektet Calendar där du skapade en klass som hette `Timestamp`. Om ursprungsfilen hette `timestamp.cpp` döp om den till `main.cpp`. Skapa sedan två nya filer i src-mappen, en som heter `timestamp.h` och en som heter `timestamp.cpp`. Lyft över de nödvändiga delarna där klassen är definierad, men lämna `main`-funktionen i ursprungsfilen.

Övning 4

I Calendar-projektet skapa nu följande klass i filerna date.h och date.cpp och implementera metoderna.

```
class Date
{
private:
    int year;
    int month;
    int day;
public:
    Date(int, int, int);
    int   daysInMonth();
    bool  isLeapYear();
    Date* nextDate();
}
```

Kod till `daysInMonth` gjorde vi under dag 1. Ett skottår är ett år som är jämt delbart med 4 eller 400, men inte med 100. Alltså, år 2000 och 2004 var ett skottår, men inte år 1900.