Laboration 4

Mål: Du ska träna på arv, virtuella funktioner och aggregat. Du ska också generera och fånga exceptionella händelser samt filhantering i C++.

Läsanvisningar

Läs kap 9–11 i läroboken. Läs också föreläsningsbilder från de föreläsningar, som behandlar motsvarande avsnitt. Dessa finns på kursens hemsida.

Förberedelser

Läs igenom den inledande texten under rubriken ”Uppgifter” nedan och lös uppgift U1-U3 och U5-U8. Läs igenom och sätt dig in i uppgift U4. Minst en av uppgifterna U5, U7 och U8 skall lösas. Uppgift U1 är frivillig.

Uppgifter

I denna uppgift ska du

U1. En cylinder kan uppfattas som en cirkel med höjd, dvs man kan låta den ärva från en cirkel. Skapa en sådan klass, Cylinder, utgående från klassen Circle nedan.

```
class Circle {
public:
  Circle(double r=0) : radius(r) {}
  double getArea() const;
  double getRadius() const { return radius; }
private:
  double radius;
};
// Implementering av medlemsfunktioner:
double Circle::getArea() const {
  return 3.1416*radius*radius;
}
```

Förutom en konstruktor, som har cirkelns radie och cylinderns höjd som parametrar, ska klassen ha följande medlemsfunktioner: `getVolume()` som returnerar cylinderns volym samt `getHeight()` som returnerar cylinderns höjd. Skriv en `main`-funktion som testar `Cylinder`-klassen.

U2. Skapa en abstrakt basklass, `Pet`, som förutom standardkonstruktor och virtuell destruktor, har den rena virtuella medlemsfunktionen `speak()`. De tre subklasserna `Dog`, `Cat` och `Bird` har egna `speak`-versioner som ger ifrån sig lämpliga djurläten.
Skriv en `main`-funktion som först skapar ett objekt av vardera `Dog`, `Cat` och `Bird`.
Utnyttja sedan dessa objekt för att initiera ett fält av pekare till basklassen `Pet`.
Inför även en nytt pekarfält, där ingående djur (ett av vardera slaget) är dynamiskt allokerade.
Använd upprepningssatser för att låta djuren i de bägge fälten tala (hitta på lämpliga läten).
Vad händer om man försöker skapa ett objekt av typen `Pet`?

U3. Denna uppgift går ut på att skapa en bostad (bestående av ett antal rum) genom att kombinera olika rumstyper. Utgå från basklassen `Rum` (se nedan) och skapa subklasserna `Kok`, `Badrum`, `Vardagsrum` och `Sovrum`, med datamedlemmar enligt:

```
Kok             harDiskmaskin och harFrys
Badrum          harDusch
Vardagsrum      harBalkong
Sovrum          antalGarderober
```

Konstruera sedan klassen `Bostad` (alla bostäder består av ett kök, ett badrum, ett vardagsrum och ett variabelt antal sovrum).
Klassen Rum:

```
class Rum {
private:
   int yta;
public:
   Rum(int y) : yta(y) {}
   virtual ~Rum() {}
   virtual void skriv() const { cout << "Yta: " << yta; }   };
```

Skriv klasserna `Bostad`, `Kok`, `Badrum`, `Sovrum` och `Vardagsrum`. Förutom konstruktorer och destruktorer ska samtliga klasser ha medlemsfunktionen `skriv()` som ger rumsinformation eller info om en hel `Bostad` (ett antal rum).

Följande testprogram skall vara körbart:

```
int main() {
  int sovrumsstorlek[] = {16, 14}; // 2 sovrum
  int garderober[] = {4, 3};
  Bostad minLya(12, false, true, // kök
                4, true, // badrum
                24, false, // vardagsrum
                sovrumsstorlek,
                garderober,
                sizeof(sovrumsstorlek)/sizeof(int));
  minLya.skriv();
}
```

och då ge utskriften:

```
Kök: Yta: 12, Diskmaskin: Nej, Frys: Ja
Badrum: Yta: 4, Dusch: Ja
Vardagsrum: Yta: 24, Balkong: Nej
Sovrum: Yta: 16, 4 garderober
Sovrum: Yta: 14, 3 garderober
```

U4. a) Välj ut någon av medlemsfunktionerna eller operatorerna i klassen `Vektor` (från föreläsningarna) där `assert` används och skriv om felhanteringen så att undantag genereras istället. Välj själv lämplig undantagstyp (ev. någon ny).

b) Fånga genererade undantag i ett huvudprogram

U5. Skriv ett program som avgör om innehållet i två filer är lika. Dialog:

```
Ange första filens namn: prog1.cpp
Ange andra filens namn: prog2.cpp
Filerna är inte lika
```

Begreppet likhet skiljer sig mellan textfiler (lika många rader med alla raderna parvis lika) och binära filer (lika många bytes med alla bytes parvis lika). Implementera den binära varianten.

U6. Vi har en textfil med namn på studerande i en kurs vid Grönköpings högskola (Internetdomän gronkoping.se). Varje rad i filen innehåller för- och efternamn för en person. Förnamnet skiljs från efternamnet med ett blanktecken, inga andra blanktecken finns i filen. Nu vill vi utifrån denna fil producera en ny fil med studenternas epostadresser.

Vi ska alltså läsa en person i taget från den gamla filen (som heter `namnlista.txt`) och utifrån personens namn generera en epostadress som sparas i en ny fil (med namnet `email.txt`). Exemplet nedan visar hur epostadresserna ska se ut. Om `namnlista.txt` innehåller

```
Arne Adamsson
Lena Evander
Orvar Persson
Petra Olsson
```

ska `email.txt` bli

```
Arne.Adamsson@gronkoping.se
Lena.Evander@gronkoping.se
Orvar.Persson@gronkoping.se
Petra.Olsson@gronkoping.se
```

OBS: Omdirigering av `cin` och `cout` är ej tillåten.

U7. Olika typer av filer har olika fördelning av tecken. Detta kan användas för att ta reda på av vilken typ en viss fil är, speciellt om filnamnet är vilseledande. Visserligen kan man då identifiera speciella teckenkombinationer som ska finnas i speciella positioner i filen (signaturer vilka lagras i en tabell). Ett alternativ till detta är att göra en enkel karakterisering av teckenfördelningen (fileprint analysis). En noggrannare analys används för att komplettera traditionell virus- och maskdetektering. I denna uppgift delar vi (lite grovt) in intervallet 0-255 (ASCII-kod) i 6 olika kategorier: 0, 1-31 (kontrolltecken förutom 0), 32 (blanktecken), 33-127 ”vanliga” tecken) , 128-254 respektive 255. Skriv ett program som läser igenom en fil och sedan skriver ut procentuell fördelning av de 6 olika kategorierna. *Exempeldialog*:

```
Filnamn: d:\winnt\notepad.exe
   Kod      %
---------------------
      0:  33.91
   1-31:  12.43
     32:   0.93
 33-127:  27.40
128-254:  20.22
    255:   5.10
```

U8. Som föregående uppgift, men med ett grafiskt användargränssnitt (wxWidgets). Använd det grafiska användargränssnittet för att välja fil att analysera. Komplettera den grafiska rapporten med ett horisontellt stapeldiagram som visar den procentuella fördelningen av de olika kategorierna. Workspace/solution inklusive ett projekt som innehåller ett skelett till lösning finns att ladda ner från kursens hemsida.