# Enkel listhantering

*I denna uppgift kommer du att implementera en dynamisk datastruktur, som till skillnad från fält (array) och poster (record) kan växa vid behov, och som till skillnad från filer lagras i interminnet och kan växa i flera dimensioner.*

## Mål

Du ska efter denna uppgift:

- förstå pekare
- kunna använda pekarstrukturer

Tonvikt läggs på:

- struktureringen av problemet
- val av underprogram och parametrar samt namn till dessa
- att göra små elementära operationer istället för komplexa
- abstraktion

## Uppgift

Du ska skriva ett paket som hanterar lagringen av en enkel form av dynamiska listor, en såkallad *enkellänkad lista*. De data som ska lagras i listan ska lagras sorterade i stigande ordning, d.v.s. det minsta datat först. Alla procedurer och funktioner som kräver genomgång av listor ska göras **rekursiva**.

På kurshemsidan finns ett givet huvudprogram i filen `test_list.adb` som ska gå att kompilera och köra med ditt paket. **Se till att skicka med detta huvudprogram och paketets filer när du lämnar in denna uppgift så vi kan testa din kod! OBS! Ändra inte på vad huvudprogrammet gör!**

Följ instruktionerna noga så att du inte ställer till det för dig.

## Instruktioner

Du ska skapa ett paket för att hantera en enkellänkad lista av heltal. Paketet ska innehålla en privat datatyp `List_Type` som motsvarar en lista och en del operationer för att hantera en sådan. Det paket som du ska skapa ska ligga i filerna ”`sorted_list.ads`” och ”`sorted_list.adb`”. När det gäller de procedurer och funktioner som ska implementeras vill vi bara säga att ni ska tänka efter innan ni börjar koda så slipper ni en massa besvär. **RITA FIGURER!**

Den privata datatypen ska implementeras som en pekare till en post, där posten ska innehålla dels själva datat (heltalet) och dels en pekare till nästa post i listan. Hela paketet ska ha förutsättningen att de data som lagras i listan ska vara kopior av det som stoppas in.

De uppgifter som här följer är de underprogram (metoder) som utgör gränssnittet mellan paketet och ett annat program (som använder paketet). I vissa fall kan det vara bra att införa hjälpoperationer för att underlätta hanteringen av de problem som uppstår. Dessa hjälpoperationer göms då lämpligen undan i filen ”`sorted_list.adb`” (syns då inte i "`sorted_list.ads`").

Du bör läsa igenom hela uppgiften innan du börjar programmera och lösa problemen. Det kan hända att du i tidigare uppgifter har nytta av rutiner som finns specificerade senare.
**OBS! Om du har paket som redan är skapade ska dessa inkluderas. Du ska inte kopiera koden ifrån dessa.**

**Funktionen Empty**

Skriv en funktion som kontrollerar om listan är tom eller ej. Returnera ett sanningsvärde. Notera att listan ska vara opåverkad av denna funktion. Använd denna funktion i senare underprogram.

**Proceduren Insert**

Du ska skriva en procedur som stoppar in ett heltal sorterat i stigande ordning i en redan sorterad lista. De parametrar som behövs är en lista och ett heltal (som både är själva datat och det man söker efter, söknyckeln). Proceduren ska inte stoppa in datat om det redan existerar ett data med samma söknyckel i listan.

**Proceduren Put**

Skriv en procedur som skriver ut hela listan på skärmen. Se körexemplen nedan för hur utskriften ska se ut. Som indata fås en lista. Observera att listan ska vara opåverkad efter denna rutin. Ett bra test är att skriva ut listan två gånger (från testprogrammet) och se så att listan är densamma vid båda utsktrifterna.

**Funktionen Member**

Du ska skriva en funktion som går igenom den sorterade listan och letar efter ett data med en viss söknyckel och returnera ett sanningsvärde som talar om ifall datat existerade i listan eller ej. Indata till funktionen är listan och en söknyckel. Notera att listan ska vara opåverkad av denna funktion.

**Proceduren Remove**

Nu ska du skriva en procedur som plockar bort ett element ur en sorterad lista (d.v.s. listan ska bli ett element "kortare"). De parametrar som ska finnas är en lista och den söknyckel som anger vilket data som ska tas bort. Du får förutsätta att alla söknycklar är unika i listan. Om det inte finns något element med denna söknyckel ska undantaget *No_Such_Element_Error* resas / kastas.

Observera att man måste återlämna minnesutrymmet för de poster man länkar ur listan. I annat fall får man något som kallas minnesläckor och detta gör i värsta fall att datorns minne tar slut.

**Proceduren Delete**

Du ska nu skriva en procedur som tar bort en hel lista (återlämnar minnesutrymmet för alla element i listan). Som parameter till proceduren fås en lista. Givetvis ska pekaren till listan vara NULL efter avklarat verk.

**Funktionen Length**

Skriv en funktion som beräknar längden av en lista. Längden definieras som antalet element i listan. Som parameter till funktionen fås en lista. Notera att listan ska vara opåverkad av denna funktion.

**Körexempel 1:**

```
Mata in heltal. Avsluta med -1: 10 20 33 -1
Listan innehåller nu elementen 10 20 33
Listan är inte tom.
Mata in ett värde: 10
Listan innehåller värdet 10.
Mata in ett till värde: 1
Listan innehåller inte värdet 1.
Mata in ett värde att ta bort: 25
Felaktigt element angivet, programmet avslutas.
```

**Körexempel 2:**

```
Mata in heltal. Avsluta med -1: 10 9 8 7 6 1 2 3 4 5 -1
Listan innehåller nu elementen 1 2 3 4 5 6 7 8 9 10
Listan är inte tom.
Mata in ett värde: 10
Listan innehåller värdet 10.
Mata in ett till värde: 15
Listan innehåller inte värdet 15.
Mata in ett värde att ta bort: 4
Listan innehåller elementen 1 2 3 5 6 7 8 9 10
Listan innehåller nu 9 element.
Rensar listan.
Listan innehåller elementen
Listan innehåller nu 0 element.
```

**Körexempel 3:**

```
Mata in heltal. Avsluta med -1: 1 1 1 2 2 2 1 1 1 2 2 2 3 3 3 -1
Listan innehåller nu elementen 1 2 3
Listan är inte tom.
Mata in ett värde: 2
Listan innehåller värdet 2.
Mata in ett till värde: 8
Listan innehåller inte värdet 8.
Mata in ett värde att ta bort: 3
Listan innehåller elementen 1 2
Listan innehåller nu 2 element.
Rensar listan.
Listan innehåller elementen
Listan innehåller nu 0 element.
```

**Körexempel 4:**

```
Mata in heltal. Avsluta med -1: -1
Listan innehåller nu elementen
Listan är tom.
Mata in ett värde: 1
Listan innehåller inte värdet 1.
Mata in ett till värde: 2
Listan innehåller inte värdet 2.
Mata in ett värde att ta bort: 5
Felaktigt element angivet, programmet avslutas.
```

**Körexempel 5:**

```
Mata in heltal. Avsluta med -1: 1 -1
Listan innehåller nu elementen 1
Listan är inte tom.
Mata in ett värde: 1
Listan innehåller värdet 1.
Mata in ett till värde: 2
Listan innehåller inte värdet 2.
Mata in ett värde att ta bort: 1
Listan innehåller elementen
Listan innehåller nu 0 element.
Rensar listan.
Listan innehåller elementen
Listan innehåller nu 0 element.
```