Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2018-08-09

Enkel listhantering

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

Mål

Du ska efter denna laboration kunna:

- använda pekarstrukturer
- förstå hur rekursion går till
- förstå hur en privat datatyp fungerar och hur en sådan implementeras

Tonvikt läggs på:

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

Uppgift

Du ska skriva ett paket "sorted_list" som hanterar lagringen av en enkel form av dynamiska listor. Datat som skall lagras i varje element i listan skall vara ett heltal i del A, och i del B skall det vara en post (mer specifikt din datatyp från lab 3, Hero_Type). De data som ska lagras i listan ska lagras sorterade i stigande ordning. För hjältarna görs detta med avseende på ålder, d.v.s. den yngsta hjälten först. Alla procedurer och funktioner som kräver genomgång av listor ska göras rekursiva.

I uppgifterna står inget om några huvudprogram. Givetvis måste sådana göras för att kunna testa paketen. Dessa huvudprogram behöver i sig inte göra något speciellt utan är till för dig som programmerare för att kunna på något sätt ”verifiera” eller åtminstone göra det troligt att paketen fungerar som de skall.

Nedan använder vi ordet "söknyckel" och "data". I del A kommer dessa att vara samma datatyp (själva talen som lagras i listan) men i del B refererar "söknyckel" till hjältens ålder, medans "data" refererar till själva hjälten. Detta medför i del B att alla hjältar i listan har unika åldrar.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2018-08-09

Del A:

Skapa först den privata datatypen 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”. Tänk efter innan du börjar koda så slipper du en massa besvär. För att skapa sig förståelse för hur pekarstrukturen fungerar är det viktigt att **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”. Du bör läsa igenom alla uppgifter innan du börjar programmera och lösa problemen. Det kan hända att du i tidigare uppgifter har nytta av rutiner som finns specificerade senare.

Funktionen Empty
Skriv en funktion som kontrollerar om listan är tom eller ej. Returnera ett sanningsvärde.

Proceduren Insert
Du ska skriva en procedur som stoppar in ett data sorterat i stigande ordning i en redan sorterad lista. De parametrar som behövs är en lista och ett heltal. 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. Du får själv välja 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).

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.

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 ett undantag (”exception”) resas. Dock behöver inte undantag tas om hand (fångas) i ditt program utan det får helt enkelt vara så att programkörningen avbryts. 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.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2018-08-09

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 Find
Du ska skriva en funktion som letar reda på ett element i en lista (ej ta bort elementet ur listan). Som indata fås en lista och en söknyckel. Funktionen ska returnera det data som matchar söknyckeln. Om det inte finns något element med denna söknyckel ska ett undantag (”exception”) resas.

Proceduren Find
Nu ska du skriva en procedur som motsvarar funktionen ovan. Samma funktionella krav som för funktionen ovan. Tips: Fundera på vad som behöver ändras och hur du på ett enkelt sätt slipper rätta i både funktionen och proceduren om man upptäcker att det är något fel.

Att man har samma namn på två underprogram kallas att man överlagrar underprogrammen. Detta innebär dock inte att det ena eller andra underprogrammet försvinner. Det innebär att båda finns tillgängliga samtidigt.

Funktionen Length
Skriv en funktion som beräknar längden av en lista. Längden definieras som antalet element i listan.

Del B:

Du skall nu göra ett nytt paket som hanterar sorterade listor av hjältar. Filerna för detta paket skall heta sorted_hero_list.ads och sorted_hero_list.ads.

Börja med att göra ett paket (hero_handling.ads/.adb) där du lägger allt som har med Hero_Type att göra. Lägg datatypen String_Type i ett separat paket som du kallar för string_handling. Givetvis skall alla datatyper vara privata i sina respektive paket.

Därefter kan du börja modifiera koden som du har från del A. Datat som lagras i varje element i listan är nu din Hero_Type. Du kommer behöva uppdatera de olika underprogrammen i paketet (insert, find, m.fl.) så att de jobbar med hjältar istället för heltal. Tänk på att "söknyckeln" nu är hjältens ålder, inte hela hjälten. T.ex. tar underprogrammet remove en söknyckel som parameter.

Var inte rädd bara för att det blir många paket. Paketen innehåller sina egna delar och det blir på detta sätt lättare att identifiera var olika fel uppkommit om detta händer.

OBS: Eftersom Hero_Type är en privat datatyp kommer du att behöva ett eller fler extra underprogram i ditt hjältepaket för att vissa av ovanstående underprogram skall fungera, vilket/vilka då?