Imperativ programering

Inlämningsuppgift 1 sommaren 2007

Jesper Wilhelmsson

12 juni 2007

1 Deluppgift A

Nedan finns fem program skrivna i fem olika språk. Er uppgift är att skriva alla fem programmen i alla fem språken. När ni är klara ska ni alltså ha 25 källkodsfiler (20 som ni skrivit själva plus de fem ni fått ut).

Programmeringsspråken är C, C++, Java, Ada och Pascal. Källkoderna finns även att ladda hem från kurshemsidan.

1.1 Program 1 - C

1.1.1 Källkodsfilen `alfabet.c`

```
#include<stdio.h>
#include<stdlib.h>

/* Skriver ut det engelska alfabetet. */

int main()
{
   char ch;

   for (ch = 'A'; ch <= 'Z'; ch++)
      printf("%c", ch);
   printf("\n");

   return EXIT_SUCCESS;
}
```

1.1.2 Att kompilera och köra

Kompilera med: `gcc alfabet.c`

Att köra programmet: `gcc` skapar en körbar fil som heter `a.out`. Denna kan startas direkt från ett terminalfönster. För att starta, skriv: `./a.out`

1.2 Program 2 - C++

1.2.1 Källkodsfilen `faglar.cpp`

```
#include<iostream>
using namespace std;

// Skriver ut texten till sången "Fem små fåglar..."

string int2str(int number)
{
   string text;

   switch (number)
   {
      case 1: text = "en"; break;
      case 2: text = "två"; break;
      case 3: text = "tre"; break;
      case 4: text = "fyra"; break;
      case 5: text = "fem"; break;
   }

   return text;
}

main()
{
   for (int n = 5; n != 0; n--)
     cout << int2str(n)
          << (n == 1 ? " liten fågel" : " små fåglar")
          << " satt på en gren, en trillade ner. Då var det bara "
          << int2str(n - 1)
          << (n == 1 ? "grenen" : n == 2 ? " fågel" : " fåglar")
          << " kvar."
          << endl;
}
```

1.2.2 Att kompilera och köra

Kompilera med: `g++ faglar.cpp`

Att köra programmet: Även `g++` skapar en körbar fil med namnet `a.out`. För att starta, skriv: `./a.out`

1.3 Program 3 - Pascal

1.3.1 Källkodsfilen `eratosthenes.pas`

```
program Eratosthenes;

(* Eratosthenes såll är en gammal grekisk algoritm som hittar alla primtal
 * upp till en given gräns.
 *)

const max = 20000;
var count, i, k, prime, size, maxsize : integer;
    mark                              : boolean;
    flags                             : array[0..max] of boolean;

begin
   Write('Ange övre gräns (max ', max, '): ');
   Readln(maxsize);
   size := (maxsize - 1) div 2;
   Writeln;
   Writeln('I intervallet 3 .. ', maxsize, ' finns följande primtal:');
   Writeln('-----------------------------------------------');

   count := 0;
   mark := true;
   for i := 0 to size do
      flags[i] := true;

   for i := 1 to size do
      if flags[i] then begin
         prime := i + i + 1;
         Write(prime:8);
         count := count + 1;
         if mark then begin
            k := (i + i) * (i + 1);
            if (k > size) or (k < 0) then
               mark := false
            else
               while (k <= size) and (k > 0) do begin
                  flags[k] := false;
                  k := k + prime;
               end;
         end;
      end;
   Writeln; Writeln;
   Writeln('Totalt ', count, ' primtal.');
end.
```

1.3.2 Att kompilera och köra

Kompilera med: `gpc eratosthenes.pas`

Att köra programmet: `gpc` använder `gcc` för att kompilera koden så även här skapas en körbar fil som heter `a.out`. För att starta, skriv: `./a.out`

1.4 Program 4 - Ada

1.4.1 Källkodsfilen `gissa.adb`

```
with Ada.Text_IO;         use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Ada.Numerics.Discrete_Random;

-- Ett litet gissa talet-spel.

procedure Gissa is
   subtype Nummer is Integer range 0 .. 100;
   package Random_Number is new Ada.Numerics.Discrete_Random (Nummer);
   Tal, G : Nummer;
   N      : Integer := 0;
   Namn   : String(1 .. 100);
   Seed   : Random_Number.Generator;
   Antal  : Natural;
begin
   Random_Number.Reset(Seed);
   Tal := Random_Number.Random(Seed);
   Put("Ange ditt namn: ");
   Get_Line(Namn, Antal);
   loop
      Put("Ange din gissning: ");
      Get(G);
      N := N + 1;
      if G < Tal then
         Put_Line("Nej " & Namn(1 .. Antal) &
                  ", den gissningen var för liten!");
      elsif G > Tal then
         Put_Line("Nej " & Namn(1 .. Antal) &
                  ", den gissningen var för stor!");
      else
         Put_Line("Det var rätt!!");
      end if;
      exit when G = Tal;
   end loop;
   Put_Line("Det tog dig " & Integer'Image(N) & " gissningar.");

exception
   when Constraint_Error =>
      put_line("Tyvärr vill jag endast ha gissningar mellan 0 och 100");

end Gissa;
```

1.4.2 Att kompilera och köra

Kompilera med: `gnat make gissa.adb`

Att köra programmet: `gnat` använder förvisso `gcc` precis som `gpc`, men skickar med flaggor för att ge resultatfilen ett trevligare namn — nämligen samma namn som källkodsfilen fast utan `.adb`. För att starta, skriv: `./gissa`

1.5 Program 5 - Java

1.5.1 Källkodsfilen `Calc.java`

```
import java.io.*;

/**
 * This class evaluates simple mathematical expressions. It supports +
 * - * / and ( ).
 */
public class Calc
{
    /**
     * <code>calc</code> evaluates mathematical expressions.
     * @param something The input string containing an expression to evaluate.
     * @returns The result or Syntax error if there is an error in the
     * input string.
     */
    private static String calc(String something)
    {
        try {
            StringReader inputStream = new StringReader(filter(something));
            int a = readExpression(inputStream);

            if (inputStream.read() != -1)
                return "Syntax error";
            return "" + a;
        }
        catch (SyntaxErrorException se) {
            return "Syntax Error";
        }
        catch (IOException ioe) {
            return ioe.getMessage();
        }
    }

    /**
     * <code>filter</code> removes blanks from the given string.
     * @param something The input string.
     * @returns The string without any whitespaces.
     */
    private static String filter(String something)
    {
        String result = "";

        for (int i = 0; i < something.length(); i++) {
            switch (something.charAt(i)) {
                case ' ':
                case '\t': break;
                default: result += Character.toLowerCase(something.charAt(i));
            }
        }
        return result;
    }
```

```
/**
 * <code>readExpression</code> reads an expression from the given
 * StringReader. An expression can be an addition, a subtraction,
 * or a term.
 * @param sr The StringReader to read from.
 * @returns The result of the term.
 * @throws IOException if something is wrong with the reader.
 * @throws SyntaxErrorException if there is an error in the input.
 */
private static int readExpression(StringReader sr)
    throws IOException, SyntaxErrorException
{
    int a = readTerm(sr);
    int next;

    next = sr.read();
    while (next == '+' || next == '-') {
        sr.mark(1);
        if (next == '+')
            a += readTerm(sr);
        else
            a -= readTerm(sr);
        next = sr.read();
    }
    sr.reset();
    return a;
}

/**
 * <code>readTerm</code> reads a term from the given StringReader.
 * A term can be a multiplication, a division, or a factor.
 * @param sr The StringReader to read from.
 * @returns The result of the term.
 * @throws IOException if something is wrong with the reader.
 * @throws SyntaxErrorException if there is an error in the input.
 */
private static int readTerm(StringReader sr)
    throws IOException, SyntaxErrorException
{
    int a = readFactor(sr);
    int next;

    next = sr.read();
    while (next == '*' || next == '/') {
        sr.mark(1);
        if (next == '*')
            a *= readFactor(sr);
        else
            a /= readFactor(sr);
        next = sr.read();
    }
    sr.reset();
    return a;
}
```

```
/**
 * <code>readFactor</code> reads a factor from the given
 * StringReader. A factor can be a number, a negative number, or
 * an expression within parenthesis.
 * @param sr The StringReader to read from.
 * @returns The result of the factor.
 * @throws IOException if something is wrong with the reader.
 * @throws SyntaxErrorException if there is a syntax error in the
 * input.
 */
private static int readFactor(StringReader sr)
    throws IOException, SyntaxErrorException
{
    int next = sr.read();

    /* ( expr ) */
    if (next == '(') {
        sr.mark(1);
        int a = readExpression(sr);
        next = sr.read();
        if (next != ')') {
            throw new SyntaxErrorException();
        }
        sr.mark(1);
        return a;
    }

    /* - factor */
    if (next == '-') {
        sr.mark(1);
        return -readFactor(sr);
    }

    /* number */
    if (Character.isDigit(next)) {
        sr.reset();
        return readNumber(sr);
    }

    throw new SyntaxErrorException();
}
```

```
    /**
     * <code>readNumber</code> reads a number from the given
     * StringReader. It is assumed that at least one of the next
     * characters in the reader are digits.
     * @param sr The StringReader to read from.
     * @returns The next available number.
     * @throws IOException if something is wrong with the reader.
     */
    private static int readNumber(StringReader sr) throws IOException
    {
        int a = 0;
        int next = sr.read();

        while (Character.isDigit(next)) {
            sr.mark(1);
            a = a * 10 + (next - '0');
            next = sr.read();
        }

        sr.reset();
        return a;
    }

    public static void main(String[] args)
    {
        System.out.println(calc(args[0]));
    }
}

class SyntaxErrorException extends RuntimeException { }
```

1.5.2 Att kompilera och köra

Kompilera med: `javac Calc.java`

Att köra programmet: `javac` skapar en klassfil som kan köras i en virtuell maskin. Klassfilen får samma namn som klassen i filen. Detta program tar in argument från kommandoraden. För att starta, skriv: `java Calc "1+2*3"`

2 Deluppgift B

Välj ett imperativt språk som ni **inte kan sedan tidigare**, försäkra er om att ingen annan väljer samma språk. Undersök språket och testa vad som är och inte är tillåtet. Vad är ett värde (R-value) i ditt språk? Vad finns det för typer? Vilka paradigmer hittar du stöd för i språket? Hitta egenskaper hos ditt språk som skiljer det från andra språk du känner till.

3 Redovisning

Båda deluppgifterna redovisas på seminarie 1. Till seminariet ska du ha med dig samtliga 25 källkodsfiler från uppgift A utskrivna på papper och minnesanteckningar så att du kan diskutera och redovisa uppgift B.

Källkoden ska vara snyggt skriven så att du med gott samvete kan lämna in den för granskning. Minnesanteckningarna är det bara du själv som ska läsa...