Imperativ programering

Lösningen till Inlämningsuppgift 1A sommaren 2007

Jesper Wilhelmsson

21 juni 2007

1 Program 1

1.1 C - `alfabet.c`

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

int main()
{
   char ch;

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

   return EXIT_SUCCESS;
}
```

1.2 C++ - `alfabet.cpp`

```
#include<iostream>
using namespace std;
main()
{
   for (char ch = 'A'; ch <= 'Z'; ch++)
      cout << ch;
   cout << endl;
}
```

1.3 Pascal - `alfabet.pas`

```
program Alfabet;
var ch : char;
begin
   for ch := 'A' to 'Z' do
      Write(ch);
   Writeln("");
end.
```

1.4 Ada - alfabet.adb

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Alfabet is
begin
   for Ch in Character range 'A' .. 'Z' loop
      Put(Ch);
   end loop;
   New_Line();
end Alfabet;
```

1.5 Java - Alfabet.java

```
public class Alfabet
{
    public static void main(String[] args)
    {
        for (char ch = 'A'; ch <= 'Z'; ch++)
            System.out.print(ch);
        System.out.println("");
    }
}
```

2 Program 2

2.1 C - faglar.c

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

char *int2str(int number)
{
   char *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;
}

int main()
{
   int n;

   for (n = 5; n != 0; n--)
      printf("%s %s satt på en gren, en trillade ner. "
             "Då var det bara %s%s kvar.\n",
             int2str(n),
             n == 1 ? "liten fågel" : "små fåglar",
             int2str(n - 1),
             n == 1 ? "grenen" : n == 2 ? " fågel" : " fåglar");

   return EXIT_SUCCESS;
}
```

2.2 C++ - faglar.cpp

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

2.3 Pascal - faglar.pas

```
program Faglar;
var n : integer;

function IntToStr(number : integer):string;
var text : string[4];
begin
   case number of
     1 : text := 'en';
     2 : text := 'två';
     3 : text := 'tre';
     4 : text := 'fyra';
     5 : text := 'fem';
   else
      text := '';
   end;

   IntToStr := text;
end;

function Storlek(n : integer):string;
begin
   if n = 1 then
      Storlek := 'liten'
   else
      Storlek := 'små';
end;

function Djur(n : integer):string;
begin
   if n = 0 then
      Djur := 'grenen'
   else if n = 1 then
      Djur := ' fågel'
   else
      Djur := ' fåglar';
end;

begin
   for n := 5 downto 1 do
      Writeln(IntToStr(n), ' ', Storlek(n), Djur(n),
              'satt på en gren, en trillade ner. Då var det bara ',
              IntToStr(n - 1), Djur(n - 1), ' kvar.');
end.
```

```
faglar.pas:4: varning: missing string capacity -- assuming 255
faglar.pas:20: varning: missing string capacity -- assuming 255
faglar.pas:28: varning: missing string capacity -- assuming 255
```

2.4 Ada - faglar.adb

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

procedure Faglar is
   function IntToStr(Number : Integer) return String is
   begin
      case Number is
         when 1 => return "en";
         when 2 => return "två";
         when 3 => return "tre";
         when 4 => return "fyra";
         when 5 => return "fem";
         when others => return "";
      end case;
   end IntToStr;

   function Storlek(n : Integer) return String is
   begin
      if n = 1 then
         return "liten";
      else
         return "små";
      end if;
   end Storlek;

   function Djur(n : Integer) return String is
   begin
      if n = 0 then
         return "grenen";
      elsif n = 1 then
         return " fågel";
      else
         return " fåglar";
      end if;
   end Djur;

begin

   for N in reverse Integer range 1 .. 5 loop
      Put_Line(IntToStr(N) & " " & Storlek(N) & Djur(N) &
               " satt på en gren, en trillade ner. Då var det bara " &
               IntToStr(N - 1) & Djur(N - 1) & " kvar.");
   end loop;
end Faglar;
```

2.5 Java - Faglar.java

```
public class Faglar
{
    private static 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;
    }

    public static void main(String[] args)
    {
        for (int n = 5; n != 0; n--)
            System.out.println(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.");
    }
}
```

3 Program 3

3.1 C - eratosthenes.c

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

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

#define MAX 20000

int main()
{
   int count, i, k, prime, size, maxsize;
   bool mark;
   bool flags[MAX];

   printf("Ange övre gräns (max %d): ", MAX);
   scanf("%d", &maxsize);
   size = (maxsize - 1) / 2;
   printf("\nI intervallet 3 .. %d finns följande primtal:\n", maxsize);
   printf("-----------------------------------------------\n");

   count = 0;
   mark = true;
   for (i = 0; i <= size; i++)
      flags[i] = true;

   for (i = 1; i <= size; i++)
      if (flags[i])
      {
         prime = i + i + 1;
         printf("%8d", prime);
         count++;
         if (mark)
         {
            k = (i + i) * (i + 1);
            if ((k > size) || (k < 0))
               mark = false;
            else
               while ((k <= size) && (k > 0))
               {
                  flags[k] = false;
                  k = k + prime;
               }
         }
      }

   printf("\n\nTotalt %d primtal.\n", count);
   return EXIT_SUCCESS;
}
```

3.2 C++ - eratosthenes.cpp

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

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

#define MAX 20000

main()
{
   int count, k, prime, size, maxsize;
   bool mark;
   bool flags[MAX];

   cout << "Ange övre gräns (max " << MAX << "): ";
   cin  >> maxsize;
   size = (maxsize - 1) / 2;
   cout << endl << "I intervallet 3 .. " << maxsize
        << " finns följande primtal:" << endl
        << "-----------------------------------------------" << endl;

   count = 0;

   mark = true;
   for (int i = 0; i <= size; i++)
      flags[i] = true;

   for (int i = 1; i <= size; i++)
   {
      if (flags[i])
      {
         prime = i + i + 1;
         printf("%8d", prime);
         count++;
         if (mark)
         {
            k = (i + i) * (i + 1);
            if ((k > size) || (k < 0))
               mark = false;
            else
               while ((k <= size) && (k > 0))
               {
                  flags[k] = false;
                  k = k + prime;
               }
         }
      }
   }

   cout << endl << endl << "Totalt " << count << " primtal." << endl;
}
```

3.3 Pascal - 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.
```

3.4 Ada - eratosthenes.adb

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

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

procedure Eratosthenes is
Max                           : constant Integer := 20000;
Count, K, Prime, Size, Maxsize : Integer;
Mark                          : Boolean;
Flags                         : array(0 .. Max) of Boolean;

begin
   Put("Ange övre gräns (max " & Integer'Image(Max) & "): ");
   Get(Maxsize);
   Size := (Maxsize - 1) / 2;
   New_Line(1);
   Put_Line("I intervallet 3 .. " & Integer'Image(Maxsize) &
            " finns följande primtal:");
   Put_Line("-----------------------------------------------");

   Count := 0;

   Mark := true;
   for I in Integer range 0 .. Size loop
      Flags(I) := true;
   end loop;

   for I in Integer range 1 .. Size loop
      if Flags(I) then
         prime := I + I + 1;
         Put(Prime, Width=>8);
         Count := Count + 1;
         if Mark then
            K := (I + I) * (I + 1);
            if (K > Size) or (K < 0) then
               Mark := False;
            else
               while (K <= Size) and (K > 0) loop
                  Flags(K) := false;
                  K := K + Prime;
               end loop;
            end if;
         end if;
      end if;
   end loop;
   New_Line(2);
   Put_Line("Totalt " & Integer'Image(Count) & " primtal.");
end Eratosthenes;
```

3.5 Java - Eratosthenes.java

```
import java.io.*;

public class Eratosthenes
{
    /**
     * Eratosthenes såll är en gammal grekisk algoritm som hittar alla
     * primtal upp till en given gräns.
     */

    static private final int MAX = 20000;

    public static void main(String args[])
    {
        int count, k, prime, size, maxsize, len;
        boolean mark;
        boolean flags[] = new boolean[MAX];
        byte[] tmp = new byte[100];

        System.out.print("Ange övre gräns (max " + MAX + "): ");
        try {
            len = System.in.read(tmp);
            maxsize = Integer.parseInt(new String(tmp).substring(0, len - 1));
        }
        catch (IOException ioe) {
            maxsize = 1000;
        }
        size = (maxsize - 1) / 2;
        System.out.println("\nI intervallet 3 .. " + maxsize +
                           " finns följande primtal:");
        System.out.println("-----------------------------------------------");

        count = 0;

        mark = true;
        for (int i = 0; i <= size; i++)
            flags[i] = true;
```

```
        for (int i = 1; i <= size; i++)
        {
            if (flags[i])
            {
                prime = i + i + 1;
                System.out.print("        ".substring(("" + prime).length()) +
                                 prime);
                count++;
                if (mark)
                {
                    k = (i + i) * (i + 1);
                    if ((k > size) || (k < 0))
                        mark = false;
                    else
                        while ((k <= size) && (k > 0))
                        {
                            flags[k] = false;
                            k = k + prime;
                        }
                }
            }
        }

        System.out.println("\n\nTotalt " + count + " primtal.");
    }
}
```

4 Program 4

4.1 C - gissa.c

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

int main()
{
   int tal, g, n;
   char namn[100];

   srand(time(NULL));
   tal = (int)((rand() / (double)RAND_MAX) * 100);
   printf("Ange ditt namn: ");
   scanf("%s", namn);
   do {
      printf("Ange din gissning: ");
      scanf("%d", &g);
      n++;
      if (g < tal)
         printf("Nej %s, den gissningen var för liten!\n", namn);
      else if (g > tal)
         printf("Nej %s, den gissningen var för stor!\n", namn);
      else
         printf("Det var rätt!!\n");
   } while (g != tal);
   printf("Det tog dig %d gissningar.\n", n);

   return EXIT_SUCCESS;
}
```

4.2 C++ - gissa.cpp

```
#include<iostream>
using namespace std;
main()
{
   int tal, g;
   int n = 0;
   string namn;

   srand(time(NULL));
   tal = (int)((rand() / (double)RAND_MAX) * 100);
   cout << "Ange ditt namn: ";
   cin >> namn;
   do {
      cout << "Ange din gissning: ";
      cin >> g;
      n++;
      if (g < tal)
         cout << "Nej " << namn << ", den gissningen var för liten!" << endl;
      else if (g > tal)
         cout << "Nej " << namn << ", den gissningen var för stor!" << endl;
      else
         cout << "Det var rätt!!" << endl;
   } while (g != tal);
   cout << "Det tog dig " << n << " gissningar." << endl;
}
```

4.3 Pascal - gissa.pas

```
program Gissa;
var tal  : integer;
    g    : integer;
    n    : integer;
    namn : string[100];
begin
   tal := Random(100);
   Write('Ange ditt namn: ');
   Readln(namn);
   repeat
      Write('Ange din gissning: ');
      Readln(g);
      n := n + 1;
      if g < tal then
         Writeln('Nej ', namn, ', den gissningen var för liten!')
      else if g > tal then
         Writeln('Nej ', namn, ', den gissningen var för stor!')
      else
         Writeln('Det var rätt!!');
   until g = tal;
   Writeln('Det tog dig ', n, ' gissningar.');
end.
```

4.4 Ada - 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;
```

4.5 Java - Gissa.java

```
import java.io.*;

public class Gissa
{
    public static void main(String[] args)
    {
        int tal, g, len;
        int n = 0;
        String namn;
        byte[] tmp = new byte[100];

        try {
            tal = (int)(Math.random() * 100);
            System.out.print("Ange ditt namn: ");
            len = System.in.read(tmp);
            namn = new String(tmp).substring(0, len - 1);
            do {
                System.out.print("Ange din gissning: ");
                len = System.in.read(tmp);
                g = Integer.parseInt(new String(tmp).substring(0, len - 1));
                n++;
                if (g < tal)
                    System.out.println("Nej " + namn +
                                       ", den gissningen var för liten!");
                else if (g > tal)
                    System.out.println("Nej " + namn +
                                       ", den gissningen var för stor!");
                else
                    System.out.println("Det var rätt!!");
            } while (g != tal);
            System.out.println("Det tog dig " + n + " gissningar.");
        }
        catch (IOException e)
        {
            System.err.println("And then everything goes black...");
        }
    }
}
```

5 Program 5

5.1 C - calc.c

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

/**
 * This class evaluates simple mathematical expressions. It supports +
 * - * / and ( ).
 */

int readNumber(char** sr)
{
   int a = 0;
   char next = *sr[0];

   while (isdigit(next))
   {
      (*sr)++;
      a = a * 10 + (next - '0');
      next = *sr[0];
   }

   return a;
}

int readFactor(char** sr)
{
   char next = *sr[0];

   /* ( expr ) */
   if (next == '(')
   {
      (*sr)++;
      int a = readExpression(sr);
      next = *sr[0];
      if (next != ')')
      {
         fprintf(stderr,"Syntax Error\n");
         exit(EXIT_FAILURE);
      }
      (*sr)++;

      return a;
   }
```

```
   /* - factor */
   if (next == '-')
   {
      (*sr)++;
      return -readFactor(sr);
   }

   /* number */
   if (isdigit(next))
   {
      return readNumber(sr);
   }

   fprintf(stderr,"Syntax Error\n");
   exit(EXIT_FAILURE);
}

int readTerm(char** sr)
{
   int a = readFactor(sr);
   char next;

   next = *sr[0];
   while (next == '*' || next == '/')
   {
      (*sr)++;
      if (next == '*')
         a *= readFactor(sr);
      else
         a /= readFactor(sr);

      next = *sr[0];
   }
   return a;
}

int readExpression(char** sr)
{
   int a = readTerm(sr);
   char next;

   next = *sr[0];
   while (next == '+' || next == '-')
   {
      (*sr)++;
      if (next == '+')
         a += readTerm(sr);
      else
         a -= readTerm(sr);

      next = *sr[0];
   }
   return a;
}
```

```
void filter(char* something, char* result)
{
   int i, j;

   for (i = 0, j = 0; something[i] != '\0'; i++)
   {
      switch (something[i])
      {
         case ' ':
         case '\t': break;
         default: result[j++] = tolower(something[i]);
      }
   }
}

int main(int argc, char* argv[])
{
   char* filtered = malloc(strlen(argv[1]));

   filter(argv[1], filtered);

   int a = readExpression(&filtered);

   if (filtered[0] != '\0')
   {
      fprintf(stderr,"Syntax error\n");
      return EXIT_FAILURE;
   }
   else
      printf("%d\n",a);

   return EXIT_SUCCESS;
}
```

5.2 C++ - `calc.cpp`

5.3 Pascal - `calc.pas`

5.4 Ada - `calc.adb`

5.5 Java - 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 a syntax error in the input.
 */
private static int readExpression(StringReader sr)
    throws IOException, SyntaxErrorException
{
    int a = readTerm(sr);
    int 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 a syntax error in the input.
 */
private static int readTerm(StringReader sr)
    throws IOException, SyntaxErrorException
{
    int a = readFactor(sr);
    int 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 { }
```