Imperativ programmering

1DL126 3p

Föreläsning 10

Vad är programmering?

Programmering

- Hacka kod - Kroppsarbete
- Jaga buggar - Detektivarbete
- Programmeringsspråk - Verktyg
- Problemlösning - Tankearbete

Erfarenhet är A och O

Problemlösning

- Standardproblem - Standardlösningar
- Sortering, sökning, listor, träd
  - Skaffa en grundpalett med favoriter

- Språkkonstruktioner
  - for - while
  - exceptions
  - call-by-value - call-by-reference

Problemlösning

- Divide and conquer
  - Förutsätt att funktionerna du behöver redan finns och använd dem.
- Flow
  - Fortsätt skriva så länge tanken finns kvar! Bekymra dig inte om detaljer - det löser sig senare :)

Problemlösning

Premature optimization is
the root of all evil!

Lös problemet först
Optimera senare

Optimering

- Optimering - värt besväret?
- Bara om koden används mycket!
  - Om koden som optimeras inte tar mer än 1% av den totala körtiden kan man inte tjäna mer än 1% på att optimera den!
- gprof
  - gcc -pg

Optimering

- Generella tips:
  - Deklarera variabler så lokalt som möjligt - återanvänd aldrig index-variabler!
  - Bryt ut uttryck från loopar
  - Utför endast beräkningar en gång
  - Identifiera beräkningar!!

Optimering

- Lär känna din omgivning.
  - Den som vet mest vinner mest!
- I vilken ordning exekveras delarna i en for-loop?
- Har ordningen i en if betydelse?

Problemlösning

- Skriv ut elementen i en lista tills dess att listan är slut eller värdet 10 påträffas:

```
while ( list.data != 10 && list != null)
{
  print(list.data);
}
```

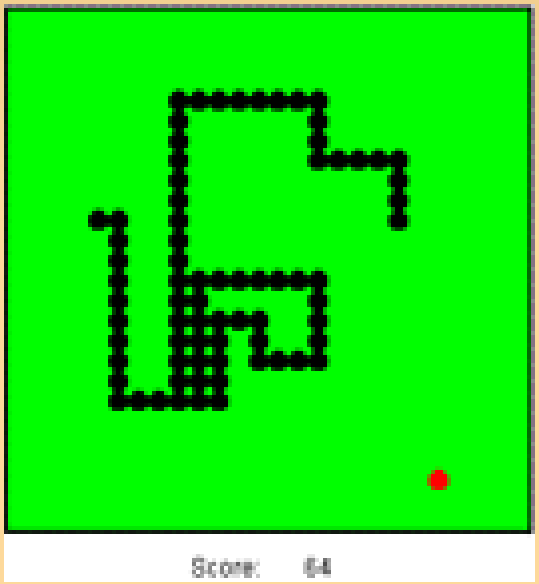
```
while ( list != null && list.data != 10)
{
  print(list.data);
}
```

Optimering

Kan man utnyttja att variabler har en begränsad storlek?

Problemlösning

Masken-spel



Vi behöver lagra varje punkts position för att kunna sudda bort den senare.

Problemlösning

Array eller lista?

Behöver vi en dynamisk datastruktur?
Nej - en naturlig övre gräns finns!

Bryt ut koden från loopen!
Array allokeras en gång innan spelet börjar
Listan allokerar element kontinuerligt

Array!

Problemlösning

Cirkulär array

- Vi vill inte flytta värden i arrayen, så vi flyttar start- och slut-index!

```
while (game is running) {
   start++; end++;
   if (start == size) start = 0;
   if (end == size) end = 0;
   plot(array[start]);
   erase(array[end]);
}
```

Optimering

Kan det göras snabbare?

Naturligtvis, men hur?

```
while (game is running) {
   start++; end++;
   if (start == size) start = 0;
   if (end == size) end = 0;
   plot(array[start]);
   erase(array[end]);
}
```

Optimering

Använd char som index och sätt storleken på arrayen till 256

```
while (game is running) {
   start++; end++;
   plot(array[start]);
   erase(array[end]);
}
```

Not: a++ är snabbare än a += 1 med gcc

Problemlösning

- Tänk baklänges!
  - Vad händer om vi vänder på det?
- Boyer-Moore
  - Effektivare textsökning genom att läsa nyckeln baklänges.
- Mark-Split
  - Effektivare minneshantering om vi ser skräpsamlingen från andra hållet.

Problemlösning

- Växla mellan två värden i en variabel:

```
if (n == 1)
   n = 2;
else
   n = 1;
```

Det imperativa sättet:

```
n = 3 - n;
```

Generell formel: $n = (v_1 + v_2) - n$

Problemlösning

- Byta plats på två värden:

```
tmp = a;
a = b;
b = tmp;
```

Går det att göra utan temporär?

```
a = a - b;
b = a + b;
a = b - a;
```

Problemlösning

- Avrunda ett tal till två decimaler:

```
((int)(n * 100 + 0.5))/100.0
```

- Givet ett slumptal mellan 0 och 1, tillverka ett slumptal mellan 50 och 100:

```
(int)(rnd * (100 - 50) + 50)
```

- Givet ett slumptal mellan 0 och 32768, tillverka ett slumptal mellan 50 och 100:

```
((int)((rnd / ((double)32768 + 1)) *
        (100 - 50)) + 50);
```

Problemlösning

- Rita en cirkel:

```
for (i = 0; i < 2 * PI; i += 0.01)
  plot(sin(i) * size, cos(i) * size);
```

- Rita en blomma:

```
for (int i = 0; i < leavs; i++) {
    circle(x + (int)(sin(i*2*PI / leavs) *
                     size),
           y + (int)(cos(i*2*PI / leavs) *
                     size),
           size);
```

Felsökning

- Assertions
  - Ta inget för givet - kolla allt!
- Spårutskrifter
  - Sätt en utskrift i början av varje funktion och en i varje loop. Skriv ut värdet på relevanta variabler.
- Debugger: gdb
  - gcc -ggdb

gdb

- Kompilera med flaggan -ggdb (gcc)
- Kör: 'gdb program'
- I gdb startar man programmet med 'run'

Obligatorisk uppgift

Skriv ner det viktigaste / det du minns bäst från dagens föreläsning