# **Programkod**
# Skivminne

**Henrik Bäck**
**850611-6253**
**Mathias Andersson**
**850424-6292**

**Operating Systems DAVB01**
**VT 2005**

Handledare: **Nils Dåverhög**
**Hans Hedbom**
**Thijs Jan Holleboom**

Inlämnad **2007-02-12**

## lab6.c

```
/* Program som simulerar olika algoritmer för diskarmsrörelser
   Se Silberschatz/Galvin/Gagne sid 493 och framåt */

#include <limits.h>
#define antalSpar 200
#define true 1
#define false 0


//Talar om om ett tal finns i vektorn
//Pre:  langd är vektorns längd, el är det sökta heltalet
//Post: Returnmerat true om elementet finns annars false
int isElement(int vektor[], int el, int langd)
{

  int i = 0;
  int ret = false;

  for(i=0;i<langd;i++)
    {

      if(vektor[i] == el)
      ret = true;
    }

  return ret;

}


/*   För alla funktioner nedan gäller:
     pre: size är storleken av vector och större än 0. size < 100.
     post: returnerar summan av rörelserna från parked genom hela vektorn */

int FCFS(int parked, int vector[],int size){
    int i, now, sum;
    sum= abs(vector[0]-parked);
    now= vector[0];
    for(i=1;i<size;i++){
        sum= sum+abs(vector[i]-now);
        now= vector[i];
    }
    return sum;
}

int SSTF(int parked, int vector[],int size){
    int i, now, sum, beenthere[100]={0};    /* beenthere =0, vi har inte varit där
än */
    int ready= 0, distance, nearest;
    sum=0;
    now= parked;
    while(!ready){
        /* look up the nearest among the ones we have not visited */
        distance= INT_MAX;
        for(i=0;i<size;i++){
                if(!beenthere[i]){
                        if(distance>abs(now-vector[i])){
                                distance=abs(now-vector[i]);
                                nearest= i;
                        }
                }
        }
        /* move */
        now= vector[nearest];
        //printf("visit %d\n",now);
        /* sum travel */
        sum= sum+ distance;
        /* set beenthere */
        beenthere[nearest]=1;
        /* ready yet? */
        ready= 1;
        for(i=0;i<size;i++){
                if(beenthere[i]==0){
                        ready= 0;
                }
        }
    }
    return sum;
}

int SCAN(int pre, int parked, int vektor[], int size)
{

  int sum = 0;
  int upp = false;
  int arSpar = false;
  int now, antKvar, i, next;
  int forst = 1;

  antKvar = size;

  if(pre < parked)
    upp = true;
  else
    upp = false;

  now = parked;

  while(antKvar > 0)
    {
      if(upp)
      {
        next = antalSpar-1;
        for(i=0;i<size;i++)
          {
            if((vektor[i] > now && vektor[i] <= next && vektor[i] > parked &&
            isElement(vektor,vektor[i],size) == true) || (vektor[i] == now &&
            vektor[i] <= next && vektor[i] == parked && forst == 1
              && isElement(vektor,vektor[i],size) == true))
            {
              next = vektor[i];
              arSpar = true;
            }
          }

        if(arSpar)
          {
            sum = sum + next - now;
            antKvar--;
            now = next;
            arSpar = false;
            forst = 0;
          }
        else
          {
            sum = sum + (antalSpar-1) - now;
            upp = false;
            now = antalSpar-1;
          }
      }
      else
      {

        next = 0;
        for(i=0;i<size;i++)
          {
            if((vektor[i] < now && vektor[i] >= next && vektor[i] < parked &&
            isElement(vektor,vektor[i],size) == true) ||(vektor[i] <= now &&
            vektor[i] >= next && vektor[i] == parked && forst == 1 &&
            isElement(vektor,vektor[i],size) == true))
            {
              next = vektor[i];
              arSpar = true;
            }
          }

        if(arSpar)
          {
            sum = sum + now - next;
            antKvar--;
            now = next;
            arSpar = false;
            forst = 0;
          }
        else
          {
            sum = sum + now;
            upp = true;
            now = 0;

          }
      }
    }

  return sum;
}


int CSCAN(int parked, int vektor[], int size)
{
  int sum = 0;
  int arSpar = false;
  int now, antKvar, i, next;
  int forst = 1;

  antKvar = size;
  now = parked;

  while(antKvar > 0)
    {
      next = antalSpar-1;
      for(i=0;i<size;i++)
      {
       if((vektor[i] > now && vektor[i] <= next && isElement(vektor,vektor[i],size)
      == true) || (vektor[i] == now && vektor[i] <= next && vektor[i] == parked &&
      forst == 1 && isElement(vektor,vektor[i],size) == true))
          {
            next = vektor[i];
            arSpar = true;
          }
      }

      if(arSpar)
      {
        sum = sum + next - now;
        antKvar--;
        now = next;
        arSpar = false;
        forst = 0;
      }
      else
      {
        sum = sum + (antalSpar-1) - now + antalSpar-1;
        now = 0;
      }
    }



  return sum;
}


int LOOK(int pre, int parked, int vektor[], int size)
{

  int sum = 0;
  int upp = false;
  int arSpar = false;
  int now, antKvar, i, next;
  int forst = 1;

  antKvar = size;

  if(pre < parked)
    upp = true;
  else
```

```
    upp = false;

  now = parked;

  while(antKvar > 0)
    {
      if(upp)
      {
        next = antalSpar-1;
        for(i=0;i<size;i++)
          {
            if((vektor[i] > now && vektor[i] <= next && vektor[i] > parked &&
            isElement(vektor,vektor[i],size) == true) || (vektor[i] == now &&
            vektor[i] <= next && vektor[i] == parked && forst == 1 &&
            isElement(vektor,vektor[i],size) == true))
            {
              next = vektor[i];
              arSpar = true;
            }
          }

        if(arSpar)
          {
            sum = sum + next - now;
            antKvar--;
            now = next;
            arSpar = false;
            forst = 0;
          }
        else
          {
            upp = false;
          }
      }
      else
      {

        next = 0;
        for(i=0;i<size;i++)
          {
            if((vektor[i] < now && vektor[i] >= next && vektor[i] < parked &&
            isElement(vektor,vektor[i],size) == true) ||
             (vektor[i] == now && vektor[i] >= next && vektor[i] == parked && forst
== 1 && isElement(vektor,vektor[i],size) == true))
            {
              next = vektor[i];
              arSpar = true;
            }
          }

        if(arSpar)
          {
            sum = sum + now - next;
            antKvar--;
            now = next;
            arSpar = false;
            forst = 0;
          }
        else
          {
            upp = true;
          }
      }
    }

  return sum;
}


int CLOOK(int parked, int vektor[], int size)
{
  int sum = 0;
  int arSpar = false;
  int now, antKvar, i, next;
  int forst = 1;

  antKvar = size;
  now = parked;

  while(antKvar > 0)
    {
      next = antalSpar-1;
      for(i=0;i<size;i++)
      {
      if((vektor[i] > now && vektor[i] <= next && isElement(vektor,vektor[i],size)
      == true) || (vektor[i] == now && vektor[i] <= next && vektor[i] == parked &&
      forst == 1 && isElement(vektor,vektor[i],size) == true))
          {
            next = vektor[i];
            arSpar = true;
          }
      }

      if(arSpar)
      {
        sum = sum + next - now;
        antKvar--;
        now = next;
        arSpar = false;
        forst = 0;
      }
      else
      {
        for(i=0;i<size;i++)
          {

            if(vektor[i] < next)
            {
              next = vektor[i];
            }
          }
        sum = sum + (now-next);
        now = next;
        antKvar--;
      }
    }
  return sum;
}

main()
{
  printf("\tFCFS\tSSTF\tSCAN\tC-SCAN\tLOOK\tC-LOOK");

  printf("\nKö 1\t");
  int v[]={98,183,37,122,14,124,65,67};
  printf("%d\t",FCFS(53,v,sizeof(v)/sizeof(int)));
  printf("%d\t",SSTF(53,v,sizeof(v)/sizeof(int)));
  printf("%d\t", SCAN(22,53,v,sizeof(v)/sizeof(int)));
  printf("%d\t", CSCAN(53,v,sizeof(v)/sizeof(int)));
  printf("%d\t", LOOK(22,53,v,sizeof(v)/sizeof(int)));
  printf("%d\t\n",CLOOK(53,v,sizeof(v)/sizeof(int)));

  printf("Kö 2\t");
  int v2[]={183,37,122,14,124,65,67,98};
  printf("%d\t",FCFS(98,v2,sizeof(v2)/sizeof(int)));
  printf("%d\t",SSTF(98,v2,sizeof(v2)/sizeof(int)));
  printf("%d\t", SCAN(22,98,v2,sizeof(v2)/sizeof(int)));
  printf("%d\t", CSCAN(98,v2,sizeof(v2)/sizeof(int)));
  printf("%d\t", LOOK(22,98,v2,sizeof(v2)/sizeof(int)));
  printf("%d\t\n",CLOOK(98,v2,sizeof(v2)/sizeof(int)));

  printf("Kö 3\t");
  int v3[]={37,122,14,124,65,67,98,182};
  printf("%d\t",FCFS(183,v3,sizeof(v3)/sizeof(int)));
  printf("%d\t",SSTF(183,v3,sizeof(v3)/sizeof(int)));
  printf("%d\t", SCAN(22,183,v3,sizeof(v3)/sizeof(int)));
  printf("%d\t", CSCAN(183,v3,sizeof(v3)/sizeof(int)));
  printf("%d\t", LOOK(22,183,v3,sizeof(v3)/sizeof(int)));
  printf("%d\t\n",CLOOK(183,v3,sizeof(v3)/sizeof(int)));

  printf("Kö 4\t");
  int v4[]={122,14,124,65,67,98,183,38};
  printf("%d\t",FCFS(37,v4,sizeof(v4)/sizeof(int)));
  printf("%d\t",SSTF(37,v4,sizeof(v4)/sizeof(int)));
  printf("%d\t", SCAN(22,37,v4,sizeof(v4)/sizeof(int)));
  printf("%d\t", CSCAN(37,v4,sizeof(v4)/sizeof(int)));
  printf("%d\t", LOOK(22,37,v4,sizeof(v4)/sizeof(int)));
  printf("%d\t\n",CLOOK(37,v4,sizeof(v4)/sizeof(int)));

  printf("Kö 5\t");
  int v5[]={98,183,37,122,14,124,65,199};
  printf("%d\t",FCFS(122,v5,sizeof(v5)/sizeof(int)));
  printf("%d\t",SSTF(122,v5,sizeof(v5)/sizeof(int)));
  printf("%d\t", SCAN(22,122,v5,sizeof(v5)/sizeof(int)));
  printf("%d\t", CSCAN(122,v5,sizeof(v5)/sizeof(int)));
  printf("%d\t", LOOK(22,122,v5,sizeof(v5)/sizeof(int)));
  printf("%d\t\n\n",CLOOK(122,v5,sizeof(v5)/sizeof(int)));

}
```