Inlämningsuppgift 3

Algoritmer och Datastrukturer

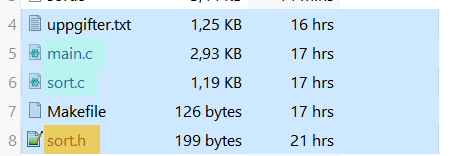
**Johan Kämpe**  
UIS16

2017-02-XX

# Svar på deluppgifter:

1. **Ladda ner och packa upp filen metriker.zip från Moodle.**

Svar:



1. **Kompilera och kör programmet med anropet main test.txt.**

**Finns det felmeddelanden vid körning? Vad skall göras för att få bort dem?**

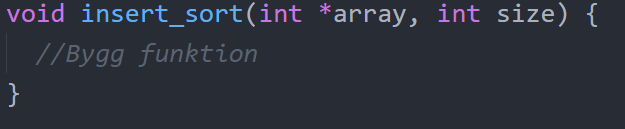
Svar:

Funktionen ***check\_if\_sorted*** i main.c ger returvärde 0 om en array inte blev korrekt sorterad efter ett anrop av en sorteringsfunktion.

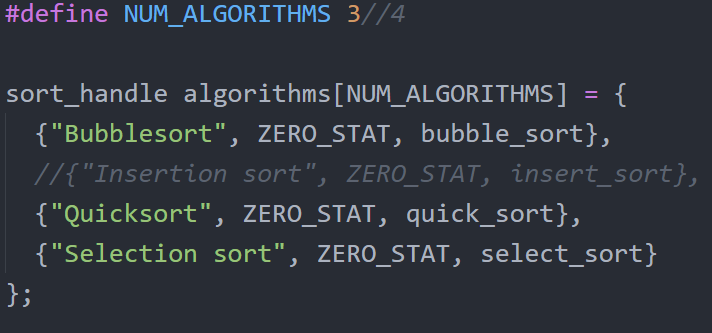
Om detta är fallet skrivs den angivna sorteringsfunktions namn ut. I vårat fall får vi meddelandet: ***ERROR: algorithm Insertion sort doesn't sort the array….***

Felet uppstår därför att funktionen **insert\_sort** anropas, men funktionen innehåller ingen kod, och sorterar således inte arrayen.

Det finns flera sätt att få bort felmeddelandena, men det bästa vore antagligen att implementera **insertion\_sort** i filen *sort.c*.



Andra alternativ är att ta bort anropet till ***check\_if\_sorted***, eller att ta bort **insert\_sort** från struktur-arrayen **algorithms** och ändra dess antal element till 3.

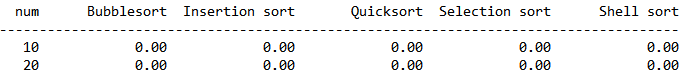


1. **Analysera de tre algoritmerna *bubble sort, quick sort* och *selection sort.* Undersök filen *test.txt* och koden.**
   1. **Har algoritmen enkel kod jämfört med de andra?**
   2. **Anropar algoritmen sig själv?**
   3. **Hur snabb är algoritmen jämfört med de två andra algoritmerna om arrayen har storleken 10? Ange procent!**
   4. **Ange snabbhet likt uppgift 3c, fast för arraystorlek 100.**
   5. **Ange snabbhet likt uppgift 3c, fast för arraystorlek 1000.**

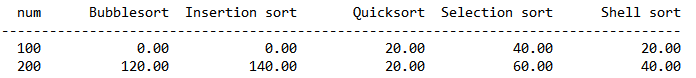
Svar:

Analys av bubble sort

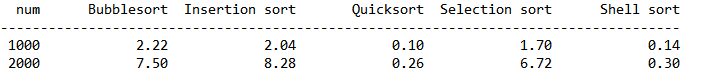
1. Bubble sort har den kod som ser mest enkel ut enligt mig, jämfört med de två andra.
2. Bubble sort-algoritmen anropar inte sig själv.
3. Jag får tiden 0.00 för samtliga algoritmer på arraystorlek 10. Och således kan de anses vara lika snabba för en sådan arraystorlek: **0 % Kommer fixas av tomki**



1. Resultatet i sorteringstid för arraystorlek 100 skiftar mellan körningar, det pendlar mellan 0,00 och 40,00 för samtliga algoritmer. Jag gör antagandet att de är ungefär lika snabba för denna storlek: **0 % Kommer fixas av tomki**



1. Sorteringshastigheter för arraystorlek 1000:



Bubble sort sorterar arrayen på **130,6 %** av tiden det tar för selection sort.  
Bubble sort sorterar arrayen på **2220 %** av tiden det tar för quick sort.

Analys av selection sort

1. Selection sort har, enligt mig, kod som är mer svårförstådd kod än bubble sort, och enklare kod än quick sort.
2. Selection sort -algoritmen anropar inte sig själv.
3. Se uppgift c för bubble sort-analysen: **0 %**
4. Se uppgift d för bubble sort-analysen: **0 %**
5. Selection sort sorterar arrayen på **76,6 %** av tiden det tar för bubble sort.  
   Selection sort sorterar arrayen på **1700 %** av tiden det tar för quick sort.

Analys av quick sort

1. Quick sort har, enligt mig, kod som är mycket mer svårförstådd än de två andra algoritmernas kod. Den använder sig dessutom av två extra stödfunktioner, utöver funktionen \_swap.
2. Quick sort-algoritmen anropar sig själv.
3. Se uppgift c för bubble sort-analysen: **0 %**
4. Se uppgift d för bubble sort-analysen: **0 %**
5. Quick sort sorterar arrayen på **4,5 %** av tiden det tar för bubble sort.  
   Quick sort sorterar arrayen på **5,9 %** av tiden det tar för selection sort.
6. **Rita upp de tre algoritmerna i ett diagram för arraystorlekarna 100-900.**

Svar: OBS EJ TID I MILLISEKUNDER!!!!!

1. **Implementera insertion sort i *sort.c***

Svar: Se kod.

1. **Utvärdera insertion sort enligt uppgift 3**
   1. *Insertion sort* har, enligt mig, kod som är likställt avancerad med *Selection sort*, dvs. svårare än bubble sort, fast enklare än *quick sort*.
   2. Insertion sort-algoritmen med min implementation anropar inte sig själv.
   3. Se uppgift c för bubble sort-analysen: **0 %**
   4. Se uppgift d för bubble sort-analysen: **0 %**
   5. Selection sort sorterar arrayen på **91,9 %** av tiden det tar för bubble sort.

Selection sort sorterar arrayen på **120 %** av tiden det tar för selection sort.  
Selection sort sorterar arrayen på **2040 %** av tiden det tar för quick sort.

1. **Implementera shell sort i *sort.c***

Svar: Se kod.

1. **Utvärdera shell sort enligt uppgift 3**
   1. *Shell sort* har, enligt mig, mycket mer avancerad kod än *Selection sort*, *bubble sort* och *Insertion sort*. Jag har svårt att avgöra om den känns enklare än koden för quick sort.
   2. Shell sort-algoritmen med min implementation anropar inte sig själv.
   3. Se uppgift c för bubble sort-analysen: **0 %**
   4. Se uppgift d för bubble sort-analysen: **0 %**
   5. Shell sort sorterar arrayen på **6,3 %** av tiden det tar för bubble sort.

Shell sort sorterar arrayen på **8,2 %** av tiden det tar för selection sort.

Shell sort sorterar arrayen på **6,7 %** av tiden det tar för insertion sort.  
Shell sort sorterar arrayen på **140 %** av tiden det tar för quick sort.