OSM, G-opgave 1

Anders Iversen
Mikkel Mastek
Anders Bjerg Pedersen

14. februar 2008

G1.1 Stak

C-compileren vil forsøge at allokere plads på stakken, når programmet oversættes. Compileren løber koden igennem og allokerer plads til f.eks. erklærede variable i mainmetoden. Når eventuelle underprocedurer køres, allokeres der undervejs ekstra plads på stakken.
Vedlagte program (stack.c) er et simpelt program, der indeholder en main-metode og en undermetode (mult). Compileren vil i vores tilfælde først allokere de tre variable i, j og k. Når programmet køres, allokeres variablen x undervejs på stakken. Programmet udskriver undervejs adresserne på de 4 variable, og i nedenstående output ses det, at stakken vokser nedad (adresserne på variablerne formindskes), og da x for det første allokeres senere OG anvender kopier af i og j (call-by-value), ligger adressen på x længere væk fra de andre.

```
app-1 > ./stack
Adressen paa i er bfce5fb0
Adressen paa j er bfce5fac
Adressen paa x er bfce5f84
Adressen paa k er bfce5fa8
app-1 >
```

G1.2 Prioritetskø

Først lidt om vores antagelser. Vi har vedtaget, at der ikke kan indsættes elementer i køen, hvor data er NULL. Ligeledes skal prioriteten være ¿0. Vores datastruktur er konstrueret, så head peger på første element i køen. Men da vi kan komme ud for at kunne indsætte et nyt førsteelement, skal head kunne flyttes. Derfor erklæres den globalt i headerfilen. Selve elementerne i køen er implementeret vha. en dobbelthægtet liste, hvor elementer indsættes i stigende rækkefølge efter prioritet, desuden med ældste elementer længst til højre. Derfor indeholder hvert element en pointer til next, prev og data. Sidste element i køen peger på next = NULL, så vi undgår at cykle rundt. Desuden peger første element på prev = (sidste element i køen), da vi dermed altid hurtigt kan tilgå

det element, der skal udtages (det vil altid være det sidste i køen).

Testprogrammet opfylder kravene stillet i opgaven, ligesom det også testes, om der kan indsættes et nyt førsteelement, og om der kan udtages elementer af en tom kø. Forventet output af data vil være 2,0,0,1,2,1,3,3, som vi heldigvis også får:

```
app-2 > ./pqt
Første element indsat. (pqueue.c)
Der blev indsat 1
Der blev indsat 0
Der blev indsat 2
Der blev indsat 3
Der blev indsat 1
Der blev indsat 0
Der blev indsat 2
Data for fjernet element: 2
Data for fjernet element: 0
Data for fjernet element: 0
Data for fjernet element: 1
Data for fjernet element: 2
Data for fjernet element: 1
Data for fjernet element: 3
Data for fjernet element: 3
Køen er tom
app-2 >
```

G1.3 Prioriteret arbejdskø

Den prioriterede arbejdskø hviler på mange af de samme principper som prioritetskøen og er da også blot en udvidelse af denne. Den største forskel ligger i, at datastrukturen i wqueue.h tilføjes en pointer til funktionen, og at wqueu_run adskiller sig en smule fra pqueue_remove, idet funktionen skal kaldes på de givne data.

I testprogrammet opfyldes igen kravene stillet i opgaven. Vi har dog kort inden afleveringsfristen opdaget, at vi er kommet til at navngive sum funktionen "mult"(men dette antager vi, er okay...). Der er dog gjort et par antagelser: de medleverede data til funktionen kræves at ligge i datastrukturen int_mult, ligesom write-funktionen kun accepterer strenge (char[] af længde 10). Dette er gjort for at begrænse funktionaliteten til kun det nødvendige.

Det har voldt os en del problemer at få konverteret en int til en string, derfor har mult- og write-funktionerne begrænset funktionalitet men fungerer dog efter hensigten.

De første 4 indsættelser i arbejdskøen i testprogrammet er write-kald, som derfor bør komme ud i rækkefølge efter prioritet (data vil så komme i rækkefølgen 0,1,2,3). Indblandet heri vil så være mult-kaldene, der tildeles en random positiv proritet, når deres write-kald indsættes i køen:

```
app-2 > ./wqt
Mult har indsat 13 med pri 3
Mult har indsat 12 med pri 6
Data er: 12
Data er: 0
Data er: 13
Data er: 1
Data er: 2
Mult har indsat 11 med pri 7
Data er: 11
Data er: 3
Mult har indsat 5 med pri 5
Data er: 5
Køen er tom
app-2 >
```