Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4105 IT Grunnkurs
Høst 2014

Øving 9

1 Teori

a) Hva er en protokoll?

b) Hva er HTTP og når brukes den?

c) Hva er de fem lagene i internets protokollmodell?

2 Strukturer

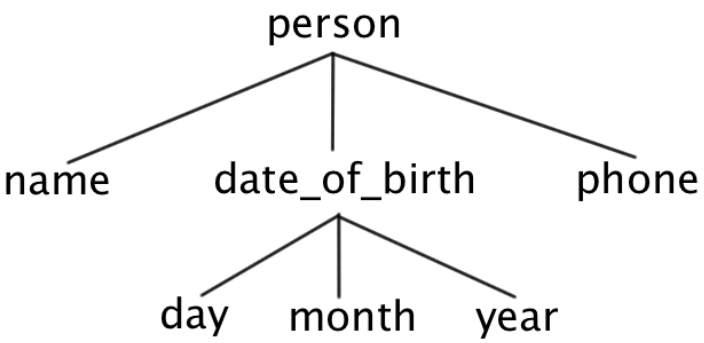
a) Lag variabelen `date` som en struktur med feltene `day`, `month` og `year` og verdiene 20, 5 og 1990

b) Lag funksjonen `print_date(date)`. Den skal ta en datostruktur som parameter og skrive ut datoen slik:

```
20.05.1990
```

c) Lag variabelen `person` som en struktur med feltene `name`, `date_of_birth` og `phone`, og verdiene 'Per Persson' for name og 48151623 for phone. `date_of_birth` skal selv være en struktur med feltene `day`, `month` og `year` og verdiene 20, 5 og 1990.

Den sammensatte strukturen ser slik ut:



d) Lag funksjonen `print_person(person)`. Den skal ta en personstruktur som parameter og skrive ut navnet, fødselsdatoen og telefonnummeret til personen slik:

```
Per Persson, 20.05.1990, 48151623
```

Tips: Bruk `print_date` for å skrive ut datoen.

e) Lag funksjonen `prompt_person()` som ber brukeren skrive inn et navn, en dag, en måned, et år og et telefonnummer. Funksjonen skal returnere en personstruktur med gitt navn, fødselsdag og telefonnummer.

Eksempel på kall med denne funksjonen:

```
Hva heter du? Per Persson
Hvilken dato er du født? 20
Hvilken måned er du født? 5
Hvilket år er du født? 1990
Hva er telefonnummeret ditt? 48151623
```

f) Lag funksjonen `get_age(person)` som tar en personstruktur som parameter og returnerer alderen til personen.

Dagens dato kan finnes slik: `[Y M D] = datevec(now);`

```
get_age(struct('day_of_birth',
  struct('day', 20, 'month', 5, 'year', 1990'))) % skal skrive ut 23
get_age(struct('day_of_birth',
  struct('day', 15, 'month', 12, 'year', 1990'))) % skal skrive ut 22
```

g) `(frivillig)` Lag funksjonen `batch_register_persons()`. Funksjonen skal be brukeren registre en person og så spørre om brukeren vil registrere flere personer. Funksjonen skal fortsette å registrere personer så lenge brukeren vil. Når brukeren er ferdig med å registrere, skal funksjonen returnere en `vektor` med alle de registrerte personene. Eksempel på kall av funksjonen er slik:

```
Hva heter du? Per
Hvilken dato er du født? 15
Hvilken måned er du født? 7
Hvilket år er du født? 1990
Hva er telefonnummeret ditt? 48151623
Skal du registrere flere personer (ja/nei)? ja
Hva heter du? Pål
Hvilken dato er du født? 13
Hvilken måned er du født? 4
Hvilket år er du født? 1991
Hva er telefonnummeret ditt? 49432434
Skal du registrere flere personer (ja/nei)? nei
```

h) Lag funksjonen `list_persons( list_of_persons )` som tar inn en vektor med personer som parameter og skriver ut hver person som i oppgave d.

3 Persondatabase

I denne oppgaven skal vi lage en persondatabase. Det innebærer å lage en fil som inneholder strengrepresentasjoner av personstrukturen fra oppgaven over.

a) Lag funksjonen `serialize_person(person)`. Funksjonen tar inn en personstruktur og skal returnere en tekstrepresentasjon av strukturen. Tekstrengen skal være på følgende format:

`<navn>#<dato>#<tlf>`

Her separerer `#` - tegnet de forskjellige feltene. F. eks. blir strukturen fra oppgave 2 c) seende slik ut: `Per Persson#20.05.1990#48151623`

Tips: Skriv funksjonen `serialize_date(date)` som returnerer tekstrengen `<dag>.<måned>.<år>`

b) Lag funksjonen `deserialize_person(person)`. Funksjonen tar inn en tekststreng lik den du lagde i forrige oppgave. Den skal returnere personstrukturen som er lagret i tekststrengen.

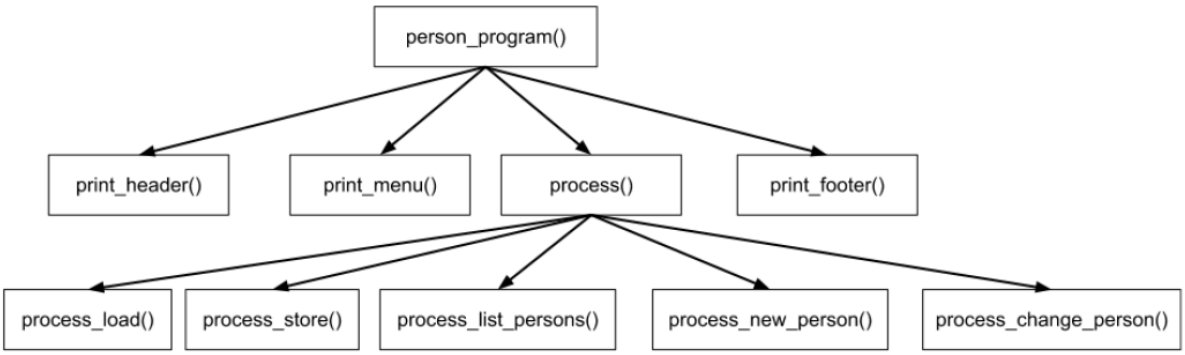
Tips: Skriv funksjonen `deserialize_date(date)` som returnerer en datostruktur. For eksempel vil `deserialize_date("20.05.1990")` gi struct('day', 20, 'month', 5, 'year', 1990)

c) Lag funksjonen `store(filename, persons)`. `persons` er en vektor med personstrukturer og `filename` er en tekststreng med navnet på filen som kalleren av funksjonen ønsker å lagre personene i. Funksjonen skal bruke `serialize_person` for å gjøre om personstrukturene til tekststrenger som skal skrives på hver sin linje i filen.

d) Lag funksjonen `loadfile(filename)`. Funksjonen skal lese filen `filename` linje for linje. For hver linje skal den bruke `deserialize_person` for å lage en personstruktur av den leste linjen. Funksjonen skal returnere en vektor med alle personene fra filen.

4 Personprogram

I denne oppgaven skal vi sette sammen funksjonene vi har laget til et kjørende program. Programmet skal vise en meny til brukeren og la brukeren velge hva den vil gjøre.



Hittil i øvingsopplegget har oppgaveteksten lagt opp til å skrive i en bunn-til-topp (bottom-up) metode. Det vil si å skrive de laveste funksjonen i kalltreet (se figur) først og de øverste til slutt. Den motsatte metoden, topp-til-bunn (top-down) er også mye i bruk, det vil si å skrive de øverste funksjonene først. Dette krever at man holder litt flere tanker i hodet samtidlig, men fører etter trening ofte til en bedre løsning.

Definer følgende funksjoner:

```
function P = process(P, choice )
  fprintf('Du har valgt %i\n', choice);
end

function print_header()
  fprintf('Skriver ut en velkomstmelding\n');
end
```

```
function print_menu()
  fprintf('Skriver ut en meny\n');
end

function print_footer()
  fprintf('Skriver ut en avskjedsmelding\n');
end
```

Dette kalles funksjonsstubber og er en teknikk som brukes når man skriver i en topp-ned metode. Stubben muliggjør at man kan skrive funkjsonen `person_program` som f.eks. kaller på `process` før `process`-funksjonen er skrevet ferdig. I senere oppgaver skal vi bytte ut `fprintf`-funksjon med den faktiske implementasjonen av funksjonen.

a) Lag funksjonen `person_program()`. Funksjonen skal gjøre følgende

1. Skrive ut en velkomstmelding
2. Skrive ut menyen
3. Be brukeren om et tall som representerer et valg i menyen
   (a) Hvis tallet er 0, avslutt funksjonen
   (b) Ellers kall funksjonen `process`
4. Repeter steg 2-4 så lenge brukeren ikke skriver inn tallet 0

Eksempel på utførelse av denne funksjonen:

```
Skriver ut en velkomstmelding
Skriver ut en meny
Velg et tall: 1
Du har valgt 1
Skriver ut en meny
Velg et tall: 0
Skriver ut en avskjedsmelding
```

b) Fyll ut funksjonen `print_header()` slik at den skriver ut en passende *velkomstmelding*.

Hovedmenyen skal se slik ut:

```
1. Hent database fra fil
2. Lagre database til fil
3. List alle personer
4. Legg inn ny person
5. Endre en person
0. Avslutt programmet
```

c) Lag funksjonen `print_menu()` som skriver ut denne menyen.

d) Fyll ut funksjonen `print_footer()` slik at den skriver ut en passende *avskjedsmelding*.

e) Definer funksjonsstubbene `P = process_load()`, `process_store(P)`, `process_list_persons(P)`, `P = process_new_person(P)` og `P = process_change_person(P)`. Legg inn en `fprintf` som forteller hva hver av funksjonen skal gjøre. (`process_load` har `fprintf('Laster fil');`, osv.).
PS. `P` er en vektor hvor hvert element er en personstruktur lik den i oppgave 2.

f) Skriv om `P = process(P, choice)`-funksjonen slik at den kaller riktig `process_...` funksjon. Hvis `choice` er lik 1 så skal `process_load` kalles osv. Hvis `choice` < 1 eller > 5, skal funksjonen skrive ut 'Ugyldig valg'.

g) Fyll ut `P = process_load()`-funksjonen. Den skal be brukeren om å fylle inn et navn på en fil. Så skal den kalle på `loadfile`-funksjonen fra oppgave 3 med navnet som argument. Funksjonen skal returnere vektoren som `loadfile` returnerer.

h) Fullfør `process_store(P)`-funksjonen. Den skal be brukeren om å fylle inn et navn på en fil. Så skal den kalle `store`-funksjonen (fra oppgave 3) med navnet og parameteren `P`.

i) Skriv ferdig `process_list_persons(P)`. Funksjonen skal kalle funksjonen `list_persons` fra oppgave 2 med parameteren `P` som argument.

j) Fyll ut `P = process_new_person(P)`. Den skal kalle på `prompt_person` fra oppgave 2. Personstrukturen som `prompt_person` returnerer skal legges til på slutten av parameteren `P`. Merk at parameteren `P` skal også returneres.

k) `(frivillig)` Fullfør `P = process_change_person(P)` funksjonen. Den skal først be brukeren om å skrive inn et tall, hvor tallet representerer nummeret på personen som brukere vil endre på. Deretter skal funksjonen skrive ut denne personen. Så kaller den på `prompt_person` for å la brukeren skrive inn oppdatert informasjon. Overskriv til slutt personen i parameteren `P`. Denne funksjonen skal også returnere `P`.

l) `(frivillig)` Det er vanlig at et program spør om du vil lagre ulagrede endringer før programmet avsluttes. Implementer dette i programmet.

m) `(frivillig)` Legg til et menyvalg for å søke etter en person på navn og implementer funksjonaliteten for dette. Søket skal printe ut personen med `print_person` funksjonen.