UiT
NORGES
ARKTISKE
UNIVERSITET

SENSORVEILEDNING

For eksamen i: INF-1100

Dato: Tirsdag 6. desember 2016

Sensorveiledningen er på 9 sider inklusiv forside

Fagperson/intern sensor: Robert Pettersen

Telefon: 47247552

Eksamen INF-1100
Innføring i programmering og datamaskiners virkemåte
Høst 2016

Løsningsforslag

Eksamenssettet består av 4 oppgaver.

Les oppgaveteksten grundig og disponer tiden slik at du får tid til å svare på alle oppgavene. I noen oppgaver kan det være nødvendig å tolke oppgaveteksten ved å gjøre noen antagelser - gjør i så fall rede for hvilke antagelser du har gjort, men pass på å ikke gjøre antagelser som trivialiserer oppgaven.

Der du skal utvikle eller beskrive en algoritme anbefales det at du først beskriver algoritmen på et høyt abstraksjonsnivå, f.eks. med figurer, før du går videre med detaljer og eventuell pseudokode. Dersom det spørres etter en implementasjon i C kreves det ikke 100% syntaktisk korrekt kode. Dersom det spørres etter pseudokode kan du kan også skrive ren C-kode om du ønsker.

Husk også at du kan referere tilbake til funksjoner du tidligere har definert.

Oppgave 1 - 25%

Gi en kort beskrivelse av von Neumann modellen og instruksjonssyklusen. Beskrivelsen bør omfatte de ulike komponentene i modellen og hvordan disse interagerer med hverandre.

Løsningsforslag

Alle komponentene skal være kort beskrevet, helst med en tegning/figur.

Alle stegene i instruksjons syklusen (Fetch, decode, evaluate adddress, fetch operand, execute, store result) skal være skissert.

Bonus med signaler, write enabled og lignende

Oppgave 2 - 25%

Gitt følgende program:

```
#include <stdio.h>

int main()
{
    int x = 128;
    char *y = "128";
    int z[] = { 1, 2, 8 };

    if (y[0] == z[0]) {
        printf("%d\n", x + z[1]);
    }
    else {
        printf("%d\n", x / z[1]);
    }
    return 0;
}
```

a) Forklar hva som er forskjellen på variablene x, y, og z, med spesielt fokus på hvilken type variablene har og hvordan verdiene deres er representert i datamaskinens minne.

b) Hvilket tall vil skrives ut på skjermen når programmet kjøres? Forklar hvordan du kommer frem til svaret ditt.

Løsningsforslag

a) x er av typen int (heltall), med verdien 128. x er en lokal variabel og ligger på stacken. y er en peker til et array av typen char (tekst streng) med 4 elementer; '1', '2', '8' og '\0' (0) som avslutter tekst strengen. y ligger på stacken og holder kun minneadressen til selve elementene som ligger i dynamisk minne. z er et array av typen int (heltall) med 3 elementer; 1, 2 og 8. y er en lokal variabl som ligger på stacken.

b) Programmet vil skrive ut integeret 64. If testen vil ikke slå til, siden char verdien '1' ikke er lik integer verdien 1. I else blir x delt på z[1] (128 / 2) som blir 64.

Oppgave 3 - 25%

Gitt et array *A* som inneholder *n* flyttall (float eller double i C).

a) Skriv en funksjon som beregner gjennomsnittet av tallene i *A* med følgende signatur:

```
double gjennomsnitt(double *A, int n);
```

Gjennomsnitt er summen av alle tallene delt på antall tall, og kan beregnes med følgende formel:

$$\frac{1}{n}\sum_{i=0}^{n-1} x_i$$

- x_i er tallet i posisjon i i array A.

For eksempel, gitt tallene 3 og 5 beregnes gjennomsnittet slik: (3+5)/2 = 4.

b) Skriv en funksjon som beregner standardavviket for tallene i *A*, med følgende signatur:

```
double standardavvik(double *A, int n)
```

Standardavvik er summen av kvadratene til hvert av tallene minus gjennomsnittet delt på antall tall, og kan beregnes med følgende formel:

$$\sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \overline{x})^2}$$

- x_i er tallet i posisjon i i array A.
- $\overline{x}$ er gjennomsnittet av alle tallene i A.
- For å beregne kvadratrot kan du anta at det eksisterer en funksjon *double sqrt(double n)*.

For eksempel, gitt tallene 3 og 5 som har gjennomsnitt 4, beregnes standardavviket slik: $\frac{\sqrt{(3-4)^2+(5-4)^2}}{2}$

c) Skriv en funksjon som flytter rundt på tallene i *A* slik at de blir liggende i stigende rekkefølge. Funksjonen skal ha følgende signatur:

```
void sorter(float *A, int n);
```

Løsningsforslag

a)
```
double gjennomsnitt(double *A, int n)
{
    double avg = 0;
    int i;

    for(i = 0; i < n; i++)
        avg += A[i];

    return (avg / n);
}
```

b)
```
double standardavvik(double *A, int n)
{
    double avg;
    double stddev = 0;
    int i;

    avg = gjennomsnitt(A, n);

    for(i = 0; i < n; i++)
        stddev += ((A[i] - avg) * (A[i] - avg));

    return sqrt(stddev / n);
}
```

c)
```
void sorter(double *A, int n)
{
    double x, y, tmp;

    for(x = 0; x < n; x++) {
        for(y = x; y < n; y++) {
            if (A[y] < A[x]) {
                tmp = A[y];
                A[y] = A[x];
                A[x] = tmp;
            }
        }
    }
}
```

Oppgave 4 - 25%

Denne oppgaven involverer bruk av lister og et angitt sett med listefunksjoner. Bruk de angitte listefunksjonene i besvarelsen. **Ikke** gjør antagelser om hvordan listene er implementert.

Gitt en datastruktur som spesifiserer et koordinat:

```
typedef struct koordinat koordinat_t;
struct koordinat {
    int x;
    int y;
};
```

Koordinater er organisert i lister som hver representerer en geometrisk figur (trekant, firkant, osv).

a) Skriv en funksjon som tar en liste med koordinater og en enkelt koordinat som argument. Funksjonen skal addere den enkle koordinaten til alle koordinatene i listen.

b) Skriv en funksjon som tar en liste med koordinater som argument, og tegner linjer mellom koordinatene i listen. For eksempel, hvis listen inneholder 3 koordinater skal funksjonen tegne en trekant, og hvis listen inneholder 4 koordinater skal funksjonen tegne en firkant.

c) Skriv en funksjon som tar en liste med koordinater som argument og returnerer en ny liste med koordinater til det minste rektangelet (boundingbox) som kan omslutte figuren beskrevet i argument listen.

Du kan anta at følgende funksjoner er tilgjengelige:

```
// Tegn en linje mellom punktet (x1, y1) og (x2, y2)
void drawLine(int x1, int y1, int x2, int y2);

// Lag en ny liste
list_t *list_create(void);

// Sett inn ett element først i en liste
int list_addfirst(list_t *list, void *element);

// Lag en ny liste-iterator som peker på første element i listen
list_iterator_t *list_createiterator(list_t *list);
```

```
// Returnerer elementet som pekes på av iteratoren og
// lar iteratoren peke på neste element.
// NULL returneres når en kommer til slutten av listen
void *list_next(list_iterator_t *iterator);

// Frigir iteratoren
void list_destroyiterator(list_iterator_t *iterator);
```

Løsningsforslag

a)
```
void flyttFigur(list_t *koordinater, koordinat_t *koordinat)
{
    list_iterator_t *it;
    koordinat_t *koord;

    it = list_createiterator(koordinater);

    koord = list_next(it);
    while (koord != NULL) {
        koord->x += koordinat->x;
        koord->y += koordinat->y;
        koord = list_next(it);
    }

    list_destroyiterator(it);
}
```

b)
```
void tegnFigur(list_t *koordinater)
{
    list_iterator_t *it;
    koordinat_t *koord, *first, *next;

    it = list_createiterator(koordinater);

    first = koord = list_next(it);
    while (koord != NULL) {
        next = list_next(it);

        if (next != NULL)
            drawLine(koord->x, koord->y, next->x, next->y);
        else
            drawLine(first->x, first->y, koord->x, koord->y);

        koord = next;
    }

    list_destroyiterator(it);
}
```

c)
```
void addKoord(list_t *liste, int x, int y)
{
    koordinat_t *koord;

    koord = malloc(sizeof(koordinat_t));
    koord->x = x;
    koord->y = y;

    list_addfirst(liste, koord);
}

list_t *beregnBoundingbox(list_t *koordinater)
{
    list_iterator_t *it;
    koordinat_t *koord;
    list_t *boundingbox = NULL;
    int minX, minY, maxX, maxY;

    it = list_createiterator(koordinater);
    koord = list_next(it);
    if (koord == NULL)
        goto done;

    minX = maxX = koord->x;
    minY = maxY = koord->y;
    koord = list_next(it);
    while (koord != NULL) {
        minX = (koord->x > minX) ? minX : koord->x;
        minY = (koord->y > minY) ? minY : koord->y;
        maxX = (koord->x < maxX) ? maxX : koord->x;
        maxY = (koord->y < maxY) ? maxY : koord->y;
        koord = list_next(it);
    }

    boundingbox = list_create();
    addKoord(boundingbox, minX, minY);
    addKoord(boundingbox, minX, maxY);
    addKoord(boundingbox, maxX, minY);
    addKoord(boundingbox, maxX, maxY);
done:
    list_destroyiterator(it);
    return boundingbox;
}
```