INF-1100: Dynamisk minnehåndtering

Einar Holsbø, UiT – Norges arktiske universitet

H24

Oversikt

1. Litt om `struct`: dvs sammensatte datastrukturer
2. Dynamisk minneallokering
3. Lenket liste: en dynamisk datastruktur

Structs samler flere variabler under samme navn

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

typedef struct kontakt {
  char navn[8];
  int nummer;
} kontakt_t;

int main() {
  struct koordinat punkt_1;
  koordinat_t punkt_2;
  kontakt_t array[3];
}
```

Hvordan ser disse ut i minnet?

```c
typedef struct kontakt {
  char navn[8];
  int nummer;
} kontakt_t;

...

kontakt_t person;
kontakt_t flere_personer[3];
```

Aksess av elementer i strukt

```
koordinat_t punkt_a;

// bruker . for å aksessere et av feltene i struct
punkt_a.x = 1;
punkt_a.y = 1;
```

Peker til strukt: litt annerledes

```
// peker
koordinat_t *punkt_b = &punkt_a;
punkt_b.x = 5;
```

Peker til strukt: litt annerledes

```
// peker
koordinat_t *punkt_b = &punkt_a;
punkt_b.x = 5;

// test.c:23:8: error: member reference type 'koordinat_t *'
// (aka 'struct koordinat *') // is a pointer; did you mean to use '->'?
// punkt_b.x = 5;
// ~~~~~~~^
//        ->
// 1 error generated.
```

Peker til strukt: litt annerledes

```c
// peker
koordinat_t *punkt_b = &punkt_a;
punkt_b.x = 5;

// test.c:23:8: error: member reference type 'koordinat_t *'
// (aka 'struct koordinat *') // is a pointer; did you mean to use '->'?
// punkt_b.x = 5;
// ~~~~~~~^
//        ->
// 1 error generated.

punkt_b->x = 5;     // fungerer
(*punkt_b).x = 5;   // fungerer men for mye styr
```

Tegning: . vs ->

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

...

koordinat_t punkt_a; koordinat_t *punkt_b = &punkt_a;
punkt_a.y = 1;

punkt_b.y = 1;   // funker ikke fordi??
punkt_b->y = 1;  // synonym for (*punkt_b).y, deref + .
```

Sammensatt `struct`

```
typedef struct {
  koordinat_t punkt_1;
  koordinat_t punkt_2;
  koordinat_t punkt_3;
} trekant_t;
```

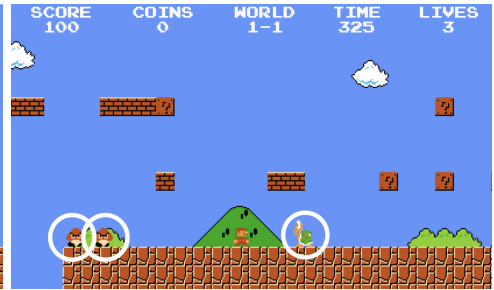
Sammensatt `struct`

```
typedef struct {
  koordinat_t punkt_1;
  koordinat_t punkt_2;
  koordinat_t punkt_3;
} trekant_t;

trekant_t dobbel(trekant_t tr) {
  trekant_t ret;
  ret.punkt_1.x = 2*tr.punkt_1.x;
  ...
  ret.punkt_3.y = 2*tr.punkt_3.y;
  return ret;
}
```

Dynamisk minneallokering: når du ikke helt vet hva du vil

I mange tilfeller er det vanskelig å vite nøyaktig hvor mye minne vi trenger



- YouTube: noen videoer har 2 kommentarer, noen har 2000 kommentarer
- En tekstfil inneholder vilkårlig mange linjer med tekst

Den vanlige tegninga

Heldigvis kan man *dynamisk* angi hvor mye minne man trenger

```c
#include <stdlib.h>       // contains malloc/calloc, free

// void *malloc(size_t size);
// void *calloc(size_t count, size_t size);
int main() {
  char *random_string; // pointer to new memory allcation

  random_string = malloc(sizeof(char));
  random_string = malloc(100*sizeof(char));

  random_string = calloc(100, sizeof(char));

  free(random_string);
}
```

Spørsmål: hva skjer med det minnet vi allokerte først???

```c
char *random_string;

random_string = malloc(sizeof(char));
*random_string = '2';

// what happens to my value of '2'?
random_string = malloc(100*sizeof(char));
```

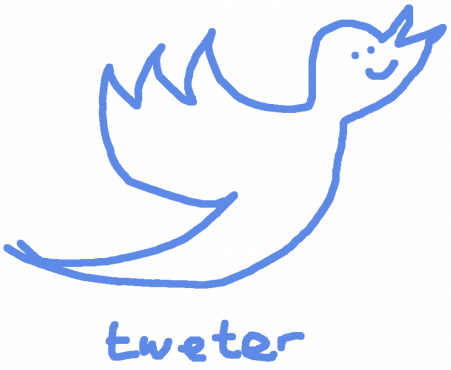


Figure 1: tweter.png

Oppgave: https://bit.ly/3fr4sZ6

Løsningsforslag: https://bit.ly/3frLx04

Lenket liste: en dynamisk datastruktur

```
En liste av som lagrer 3 objekter i 3 noder:
    start-> [one]-> [two]-> [three]-> NULL

Struktur:
    [data | peker]

    [ one | &two ]
```

Noen mulige strukturer fra minst til mest generell

```c
struct node {
  twete_t twete;       // stores the data itself
  struct node *next;
};

struct node {
  twete_t *twete;      // stores pointer to the data
  struct node *next;
};

struct node {
  void *data;          // stores a generic pointer
  struct node *next;
};
```

Å putte inn data i en lenket liste

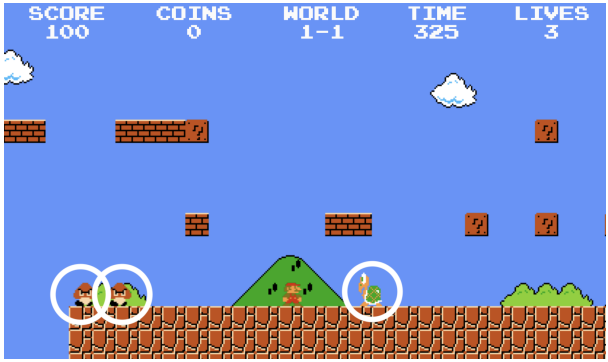


Figure 2: F.eks når det dukker opp nye figurer på skjermen

- Opprett en ny node,

Å putte inn data i en lenket liste

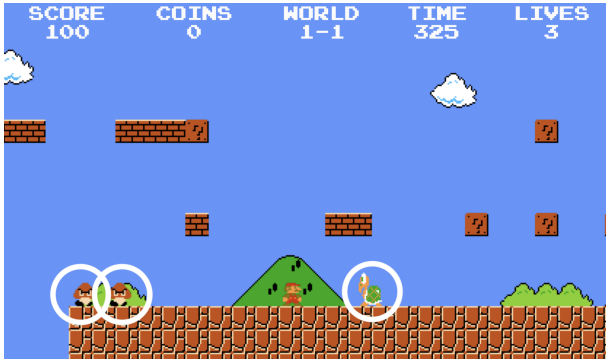


Figure 2: F.eks når det dukker opp nye figurer på skjermen

- Opprett en ny node,
- Oppdater `next`-pekerne for å "lenke den inn"

Å putte data inn først

```c
node_t *new = mallloc(sizeof(node_t));
new->data = whatever;

// you always have a pointer to the start of the list

new->next = start;      // new node's next is old start of list
start = new;            // start of list is now the new node

// Q: what would happen if I did those lines in reverse order?
```

Å putte data inn sist

```
node_t *new = mallloc(sizeof(node_t));
new->data = whatever;

node_t *end;

// ... code that sets end pointer to last item in list

end->next = new;      // next of end node is the new one
new->next = NULL;     // new node has no next, it's the end node
```

Å putte data inn i midten

```
node_t *new = mallloc(sizeof(node_t));
new->data = whatever;

node_t *current;

// ... code that sets current to wherever we want to insert after

new->next = current->next;
current->next = new;

// Q: what would happen if I did those lines in reverse order?
```

Å fjerne data fra en lenket liste

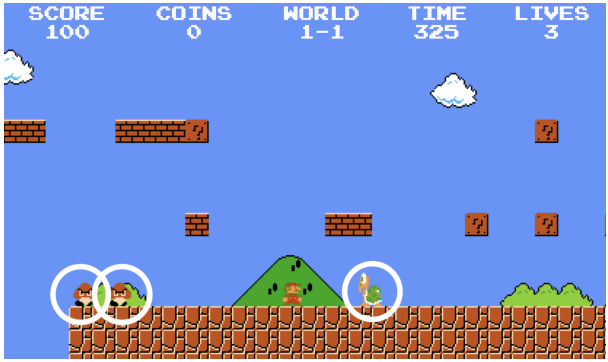


Figure 3: F.eks når det forsvinner figurer fra skjermen

- Oppdater `next`-pekerne for å "lenke ut" noden

Å fjerne data fra en lenket liste

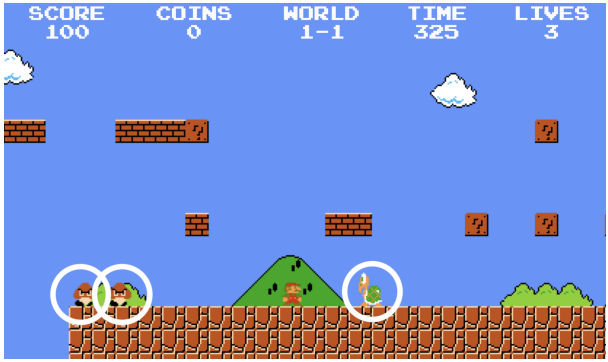


Figure 3: F.eks når det forsvinner figurer fra skjermen

- Oppdater `next`-pekerne for å "lenke ut" noden
- Slett den med `free()`, eller gjør hva du ellers vil

Å fjerne data fra starten av en lenket liste

```
node_t *tmp = start;
start = start->next;

// Q: what is the idea with the tmp pointer?
```

Å fjerne data fra slutten av en lenket liste

```
node_t *tmp;
node_t *current;

// ... code that sets current to SECOND TO LAST node

tmp = current->next;
curent->next = NULL;
```

Å fjerne data fra midten av en lenket liste

```
node_t *tmp;
node_t *current;

// ... code that sets current to the one BEFORE what we want to remove

tmp = current->next;
curent->next = current->next->next;   // note: 2x next

// Q: do we need special code to remove the last node
//    or is this enough?
```

“code that sets current to. . . ”

```
node_t *current = start;

while(node isn't the one we want) current = current->next;
```

Spørsmål: hva hvis vi vil gå bakover i lista?

Figure 4: tweter2.png

Oppgave: https://bit.ly/3CjB0NC

Løsningsforslag: https://bit.ly/3CjVPbS