INF-1100

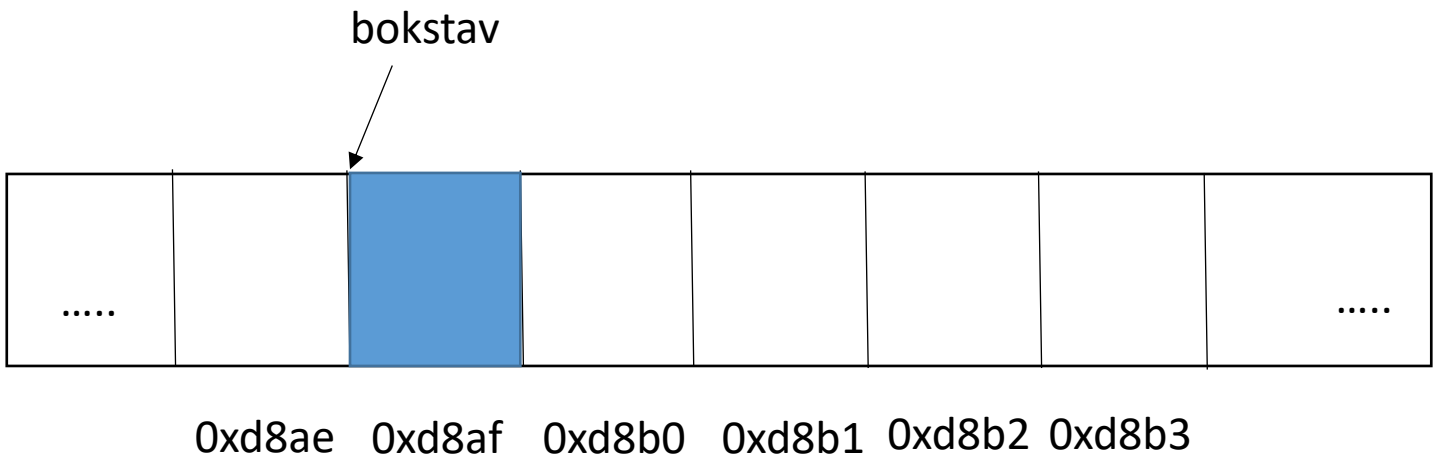
Pekere (pointers)

Tabeller (arrays)

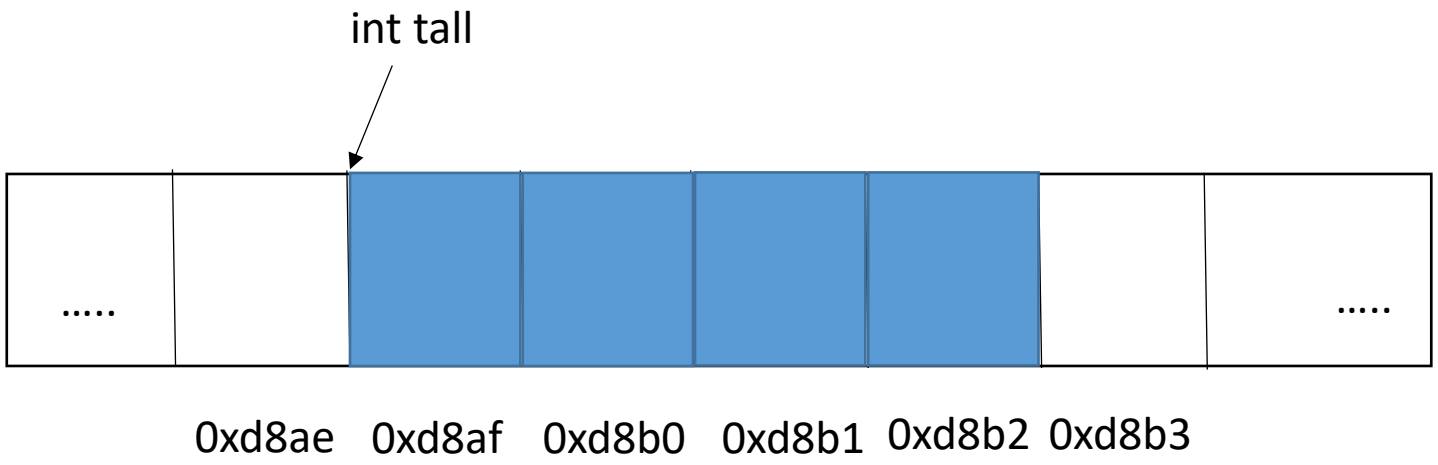
Tekststrenger (strings)

Tilbakeblikk på variabler

- Hva skjer når vi deklarerer en variabel?
  - char bokstav;
- C reserverer en byte i minnet (her siden det er datatype char)
  - Hvor i minnet? En adresse (minneadresse)



Deklarerer int tall; 4 bytes blir satt av i minnet



Peker (pointer)

- Er en variabel som inneholder en minneadresse til en variabel (f.eks char, int, double…)
- En kraftig og fleksibel måte å manipulere data i programmer, men kan være vanskelig (lett å gjøre feil -> segmentation fault).
- Nært beslektet med tabeller (arrays) og tekststrenger (strings)

Hvorfor pekere

- Kan referere til store datastruktuerer
- Kan dele data mellom forskjellige deler i et program
  - Pass-by-reference (som egentlig er pass-by-value, men vi sender inn minneadresser og kan manipulere verdiene på disse minneadressene slik at de kan brukes i andre deler i programmet).
- Dynamisk minneallokering
  - Kan allokere nytt minne under kjøring av et program
  - Mer om det neste uke

Pekeroperatorer

- Adresseoperatoren & gir oss minneadressen til en variabel

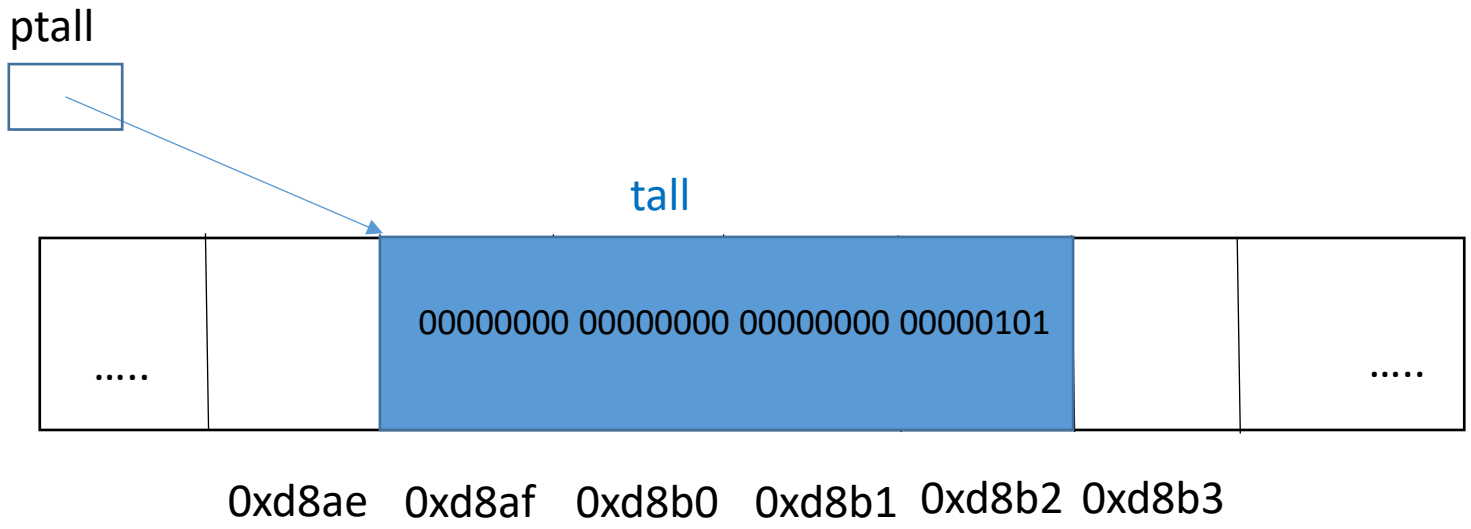
- Unær-operatoren * har forskjellig betydning:
  - I deklarasjoner betyr den at det er en peker
  - Ellers betyr det at vi vil ha tilgang til dataen den peker til

- Pekere deklareres som f.eks. int *ptall;
  - leses ptall er en peker til et hetall
  - ptall er en heltallspeker

- De-referans-operatoren * gir oss verdien som ligger lagret på en minneadresse (via en peker)

Minneadresse

```
int tall = 5;//setter tall til 5
int *ptall = &tall; //setter peker til å peke
                    // til minneadressen hvor 5 befinner seg
```



Eksempler med minneadresser og de-referering

Pass-by-reference vs pass-by-value

- Pass-by-value: En funksjon får sendt inn kopier av verdiene til argumentene
  - Endringer av verdiene i funksjonen vil ikke bli sett utenfor funksjonen

- Pass-by-reference: En funksjon får referanser til minneaddressene til argumentene
  - Endringer av disse verdiene i funksjonen blir også endret utenfor funksjonene
  - Egentlig pass-by-value i C, men med pekere så oppfører det seg på samme måte som pass-by-reference i andre språk

Eksempel et tall og funksjoner

void pekere

- void *pnavn;
- Void pekere er pekere som peker til verdi uten spesiell datatype
  - Genererisk peker
  - Tillatter å peke til hvilken som helst datatype
- Data som er pekt til av en void peker kan ikke bli direkte de-referert
  - Må bruke eksplitt type casting, før vi de-referere det (Hvorfor?)
    - F.eks en int -> *(int*)pnavn
    - F.eks en char -> *(char*)pnavn

Eksempel med void pekere

Tabeller (arrays)

- En tabell (array) er et antall elementer av samme datatype som ligger etter hverandre i minnet.

- int arr[i];
  - arr er en peker til det første elementet i tabellen (arrayet)

Pekere og tabeller

- Tabeller har mange likhetstrekk med pekere, er ikke det samme men vi kan bruke dem likt:
  - Begge peker til første elementet (normalt, såfremt du ikke ber pekeren til å peke lengre inn i tabellen)

```c
int main() {

    int arr[5] = {1,2,3,4,5};

    int *pt = arr;//Peker på første elementet
    int *pa = &arr[3];//Peker til fjerde elementet

    printf("%d\n",*pt);//gir tallet 1
    printf("%d\n",*pa);//gir tallet 4

    return 0;
}
```

Peker-aritmetikk vs tabellnotasjon

- Kan velge selv hvilken vi ønsker å bruke

- Aritmetikk vs tabellnot:

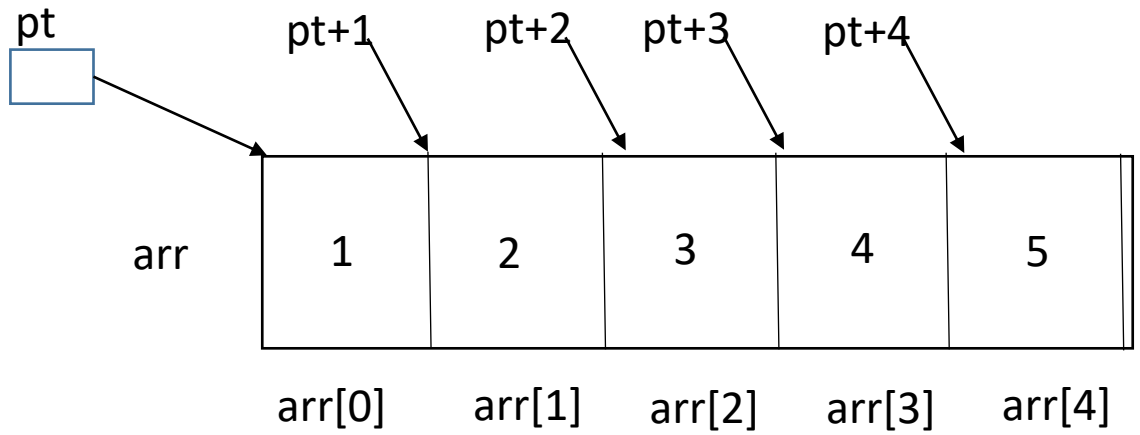
arr++	arr[i++]	arr[++i]
arr--	arr[i--]	arr[--i]
arr+1	arr[i+1]	
arr+2	arr[i+2]	
arr-1	arr[i-1]	
arr-2	arr[i-2]	

  - Etc.

```c
int main() {

    int arr[5] = {1,2,3,4,5};

    int *pt = arr;//Peker på første elementet
    printf("%d\n",arr[2]);//gir tallet 3
    printf("%d\n",*(pt+2));//gir tallet 3

    return 0;
}
```

Pekere og tabeller

```
int arr[5] = {1,2,3,4,5};
int *pt = arr;//Peker på første elementet
```




```
printf("%p\n",&arr[2]);//gir addressen til 3 elementet
printf("%p\n",pt+2);//gir addressen til 3 elementet
```

```
0x7fff80d1b788
0x7fff80d1b788
```

Tekstrenger

- Er tabeller (arrays) av typen char

```
char ord[10]
word[0] = ‘g’;
word[1] = ‘i’;
word[2] = ‘r’;
word[3] = ‘a’;
word[4] = ‘f’;
word[5] = ‘f’;
word[6] = ‘e’;
word[7] = ‘\0’; ←For å vite når tekststrengen er ferdig
printf("%s", word);
```

Tekstrenger

char ord[10] = "Hello"; \\ \\<- "\0" legges til automatisk Kan endre verdiene i tabellen, legges på stakken

printf("%s",word);

char *pord ="Hello"; \\<- "\0" legges til automatisk , legges i read-only minne

printf("%s",word);

Dobbelpeker

- Vi kan ha en peker som peker til en peker

```
char a = 97;
char *pt1 = &a;
char **pt2 = &pt1;

printf("%d\n", a); // printer 97
printf("%d\n", *pt1); // printer 97
printf("%d\n", **pt2); // printer 97
```

Hva skjer her?

dobbelpeker