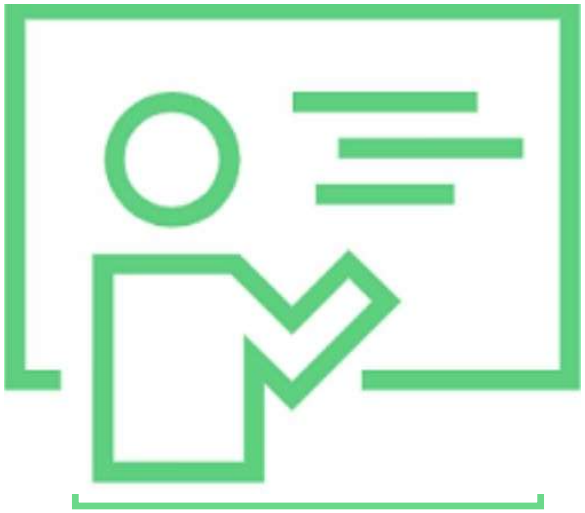


Arrayer

Stefan Holmberg, Systementor AB

Arrayer

- En **Array** i C är en samling av värde av samma typ som lagras i ***minnesplatser efter varandra***

- int a[100] - en lista med 100 heltal

- char msg[12] - en lista med 12 tecken (en sträng)

Ang data locality: https://www.youtube.com/watch?v=YQs6IC-vgmo

Array: deklaration

listor1.c

```
int main() {
    int a[10];
    int b[3] = { 10, 20, 30 };
    int c[] = { 5, 4, 3, 2, 1 };

    return 0;
}
```

Deklarera en lista med:

- typ och storlek
- typ, storlek och initialvärde
- typ och initialvärde
  - storlek räknas ut implicit

Arrray: läsa

listor2.c

```
int main() {
    int a[5] = { 5, 10, 15, 20, 25 };
    printf( "%d", a[0] );
    printf( "%d", a[4] );

    return 0;
}
```

Läsa av från en lista:

- Använd hakparanteser
- Börjar från 0
  - *a[0]* - är första element, dvs 5
  - *a[4]* - är sista, dvs 25

Arrray: indexera

listor3.c

```c
int main() {
    int a[5];
    a[-1] = 10;
    a[10] = 10;

    return 0;
}
```

Funkar koden?

Arrray: indexera

listor3.c

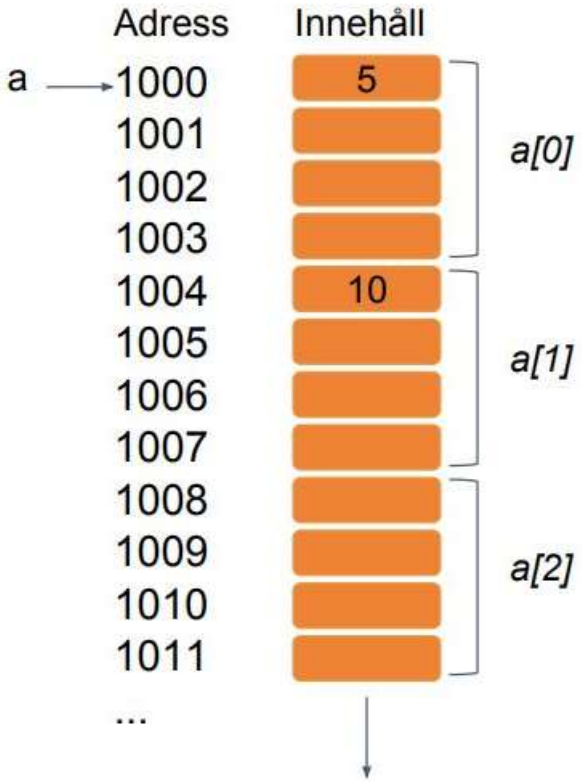
```
int main() {
    int a[5];
    a[-1] = 10;
    a[10] = 10;

    return 0;
}
```

Funkar koden?

Kanske, kanske inte:

- C kollar aldrig att du indexer din lista korrekt
- *a[-1]* betyder adressen till platsen i minne före listan
- Men den platsen kanske används till något viktigt!

Arrray: minne




```
int main() {
    int a[5];
    a[0] = 5;
    a[1] = 10;

    return 0;
}
```

En array är lite som en pekare (m
samma!):

- Den allokerar och pekar till ett o
minne
- int a[10] - nu har vi minne till 10
- Du kommer åt element i listan n
- = (&a + 5 * sizeof(int))
- Eftersom en int tar 4 bytes: (a +
- Det är lätt att det blir fel, t.ex a[-
- = (&a - 1*4