Arrayer

Arrayer

- Vanligt att våra program innehåller många referenser till precis samma saker, t ex till ett antal tal, till ett antal tecken, fotbollsresultat etc.
- T ex:

```
double tal1, tal2, tal3 tal4;
```

  – som vi använder för att referera till 4 stycken värden till temperatur.
- Men om vi skulle vilja lagra data temperaturen varje timma under ett dygn skulle vi behöva 24 st variabler.
- Men m.h.a en array kan vi skapa en variabel som kan lagra t ex 24 st värden med samma datatyp.

- Ex:

```
double temperaturer[5];
```

- Nu har vi en variabel temperatur som kan innehålla 5 stycken värden.
- Vi skulle kunna se temperatur som en namngiven låda som nu innehåller 5 stycken fack, där varje fack kan adresseras. Detta görs via att använda variabeln man följt av vilket fack vi vill refera till.
- T ex:

```
temperatur[3]
```

  - refererar till det 4 facket i "lådan". 4:e???? Ja, C börjar att räkna på noll (0). Så det första facket heter temperatur[0], den 2:dra temperatur[1] och det 5:e temperatur[4].

Deklaration

- Formellt

  ```
  datatyp namn[storlek];
  ```

  – Datatyp: char, int, float etc. (någon egen struct, typedef etc. mer om det på senare föreläsning)
  – Namn, vad vi vill namnge variabeln som
  – Storlek, hur många element vi vill ha
- Åtkomst via indexering, dvs. namn[vad], där vad är vilket element vi vill referera till

Inititering

- Startvärden, samma som en vanlig oinitierad variabel.
- Vi kan ge värden via tilldelningar

```
temperatur[0] = 12.34;
temperatur[1] = 42.65;
```

- eller vid deklarationen:

```
double temperatur[5] = { 12.74, 2.49, 8.35, 98.65, 3.49 };
double temperatur[] = { 12.74, 2.49, 8.35, 98.65, 3.49 };
```

- Båda ger en array med 5 doubles.

Exempel, skriva ut innehåll


```
int my_array[5] = { 2, 5, 1, 908, 34 };
for (int i = 0; i < 5; ++i) {
     printf("Element %d = %d\n", i, my_array[i]);
}
```

- Och baklänges:

```
printf("Reversed\n");
for (int i = 4; i >= 0; --i) {
     printf("Element %d = %d\n", i, my_array[i]);
}
```

Detta exempel finns i file skriva_ut.c

Flera dimensioner

- En array kan ha fler än en dimension, 2, 3 osv.
- Ex:

```
int many_numbers[10] [5];
```

- Detta skapar en array med 10 rader och 5 kolumner.
- Vi referar till elementen via indexering:

`many_numbers[1] [4]`, dvs den andra radens femte element

- Initiering kan göras via tilldelning

```
many_numbers[0][4] = 45609;
```

- Nu innehåller rad 0 kolumn 4 värdet 45609.
- eller vid deklarationen:

```
many_numbers[2][5] = { { 2, 876, 54, 20, 7 },
                       { 4, 554, 328, 888, 0}};
```

- En utskrift av denna array blir:

Exempel

```
int many_numbers[2][5] = { { 2, 876, 54, 20, 7 },
                           { 4, 554, 328, 888, 0}};
for (int i = 0; i < 2; ++i) {
       for (int j = 0; j < 5; ++j) {
          printf("Index %d %d has the value %d \n", i, j,
                     many_numbers[i] [j]);
       }
}
```

Detta exempel finns i filen skriva_ut.c

För framtiden

- Denna föreläsning kommer att efter föreläsningen om minneshantering och pekare att läggas ut i en ny variant som utnyttjar dynamisk minnehantering och pekare.