Räckor, arrays

- Antag att du skall skriva ett program som läser in *n* värden och sedan använder sig av dessa
- Problemet kan lösas genom att deklarera *n* stycken variabler
  - Opraktiskt och oflexibelt
- En *räcka* (eng. array) kan ses som en *ordnad mängd* (ordered set) innehållande element av en viss typ
- I stället för att deklarera *n* stycken heltalsvariabler kan vi deklarera **en** räcka som **innehåller** *n* heltalselement

Deklaration av räckor

- En räcka kan endast innehålla värden av en **viss typ** (heltal, flyttal, bokstäver...)
- När en räcka deklareras måste man ange hur många element som finns i räckan
- För att deklarera en räcka som innehåller 20 heltal:
  ```
  int myIntArray[20];
  ```
- För att deklarera en räcka som innehåller 40 bokstäver:
  ```
  char myCharArray[40];
  ```

Elementen i räckan

- För att komma åt ett element *i* som finns i räckan *x* används notationen **`x[i]`**
- **`i`** kallas för ett **index**.
- Det första elementet i en räcka har alltid indexet **`0`**.
- För att komma åt det **första** elementet i en räcka **`x`** med **`n`** element anges därmed **`x[0]`**
- För att komma åt det **sista** elementet anges **`x[n-1]`**
- För att t.ex. komma åt det **tredje** elementet i vår heltalsräcka och tilldela detta element värdet 100: **`myIntArray[2] = 100;`**

Räckor i datorns minne

Hur lagras en räcka deklarerad som
`int values[10]` i datorns minne?

`values[0]`	`?`
`values[1]`	`?`
`values[2]`	`?`
`values[3]`	`?`
`values[4]`	`?`
`values[5]`	`?`
`values[6]`	`?`
`values[7]`	`?`
`values[8]`	`?`
`values[9]`	`?`

Efter deklarationen

`values[0]`	`197`
`values[1]`	`?`
`values[2]`	`-101`
`values[3]`	`547`
`values[4]`	`?`
`values[5]`	`350`
`values[6]`	`?`
`values[7]`	`?`
`values[8]`	`?`
`values[9]`	`35`

Efter några tilldelningar

Räckor, exempel

```
int myArray[10]; // Declare an array with 10 integers
int index;

/* Loop through the array,
   initialize each element to index*2 */

for (index = 0; index < 10; index++)
{
    // Note that the value of index ranges from 0 to 9!
    myArray[index] = index*2;
}

printf("The 5th element in the array is %d\n",
        myArray[4]);
```

Direkt initialisering av räckor

- Också möjligt att initialisera räckan samtidigt som den deklareras
- Praktiskt användbart endast för små räckor
- Inte nödvändigt att ange hur många element räckan innehåller – **om** du initialiserar varje element i räckan
- Exemplet nedan deklarerar och initaliserar en räcka med 5 element:

```
double aShortArray[] = { 1.0, 3.14, 2.2, 4.3, -3.23 };
```

Ingen storlek behöver anges

'Curly braces' för att avgränsa mängden

Direkt initialisering av räckor

- Kan användas för automatisk initialisering av räckans element:

```
int myArray[10] = { 1, 2, 3 };
```

myArray[0] = 1, myArray[1] = 2, myArray[2] = 3
myArray[3..9] får värdet 0

```
int myArray[10] = { 0 };
```

Alla element får värdet 0

```
int myArray[10] = {};
```

Alla element får värdet 0 ***om*** *kompilatorn stöder detta.* ***Inte*** *standard-c!*
(pröva att kompilera med **`gcc -pedantic`**)

Räckor, exempel

- Ett program för att beräkna statistik för enkätsvar
- Läs in x antal värden mellan 1 och 10
  (som svarar mot 10 olika “kategorier”)
- Lagra antal svar per kategori i en räcka
- Skriv ut resultatet genom att gå igenom räckans element
- Källkoden finns i katalogen *Exempelprogram* i Moodle
- Observera hur programmet beaktar att en räckas första index är noll: För att göra programmet mera lättförståeligt skapas en räcka med 11 element, trots att vi egentligen endast behöver 10 element.
  => Kan använda elementen 2-11 (med *index* 1-10)

“Variable-length arrays”

- Räckor vars längd inte har slagits fast i kompileringsskedet – sådana räckors längd specificeras genom *värdet av en variabel*
- Fortfarande **inte** möjligt att ändra storleken på räckan **efter** att den har skapats

```
int size = 0;

printf("How many elements do you need?");
scanf("%d", &size);

// Declare an array with 'size' number of elements
double mySizeOnDemandArray[size];
```

Lookup tables

```
/* Declare an array containing the
   number of days per month */
int daysInMonth[12] = {31,28,31,30,31,30,31,
                       31,30,31,30,31};
int month = 0;
do {
printf("Enter month (1-12):");
scanf("%d", &month);
}
while ((month < 1) || (month > 12));

/* Perform a lookup to get the amount
   of days for this month. Note usage of -1 to
   account for array index starting from 0 */
printf("This month has %d days\n", daysInMonth[month-1]);
```

Flerdimensionella räckor

- Ett element i en räcka kan innehålla en annan räcka => *flerdimensionell* räcka
- En tvådimensionell räcka med *i* rader och *j* kolumner kan även ses som en *matris*
- För att deklarera en tvådimensionell räcka med 5 rader och 10 kolumner, och sedan tilldela värdet 42 till elementet i rad 3, kolumn 7:

```
int myTwoDimensionalArray[5][10];
myTwoDimensionalArray[2][6] = 42;
```

Flerdimensionella räckor, exempel

```
int mArray[3][5] = { {1, 2, 3, 4, 5 },
                     {0, 0, 0, 0, 0 },
                     {0, 0, 0, 0, 0 } };

int i;
for (i=1; i<3; i++)
{
    for (int j=0; j<5; j++)
    {
        mArray[i][j] = mArray[0][j] * (i+1);
    }
}
// Contents of mArray after program execution is?
```

Visualisering av flerdimensionella räckor

1	2	3	4	5
0	0	0	0	0
0	0	0	0	0

Efter initialisering (i==0)

1	2	3	4	5
2	4	6	8	10
0	0	0	0	0

Efter första yttre iterationen (i==1)

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15

Efter andra yttre iterationen (i==2)