4 Structured data types

1. Arrays. array<>. Multidimensional arrays. Arrays as parameters
2. String of chars. string. string as parameters.
3. Structs. The data type struct. Structs as parameters.
4. Examples

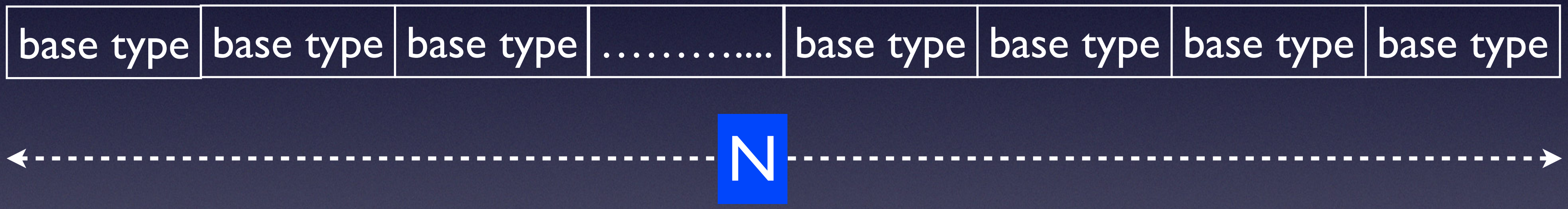
Prof. Juan Falgueras
1st. year ETSII

4.2 Arrays

1. **Arrays. array<>.** Multidimensional arrays. **array as parameter**
2. Structs. The data type struct. Structs as parameters.
3. String of chars. string. string as parameter.
4. Examples

Arrays

N repetitions of a base type



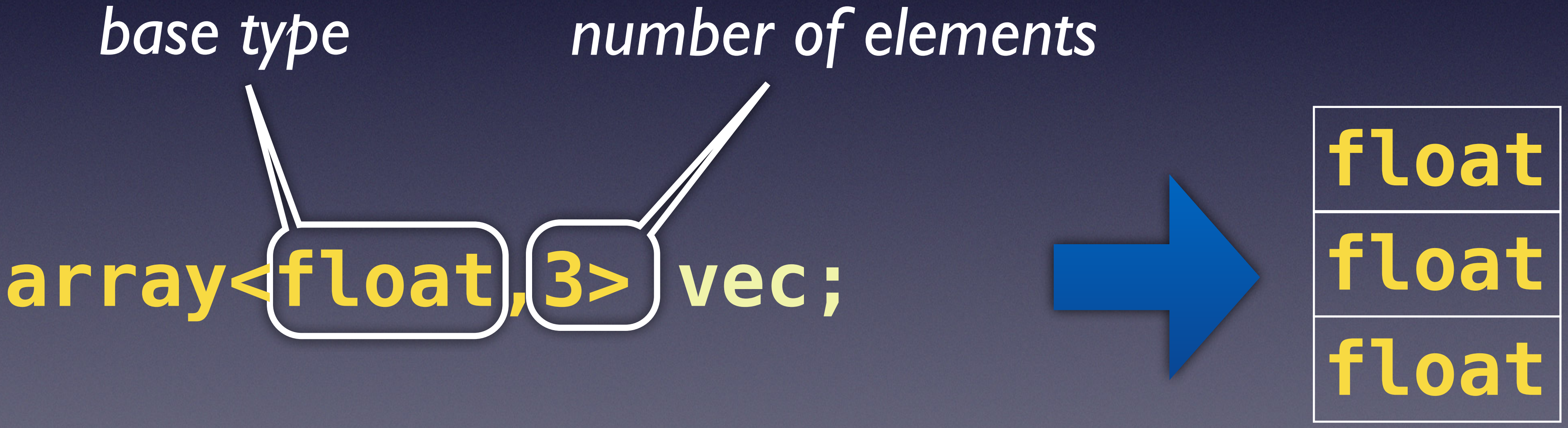
base type | base type | base type | ………..... | base type | base type | base type | base type

N

The number N of elements is static, unchangeable

Syntax

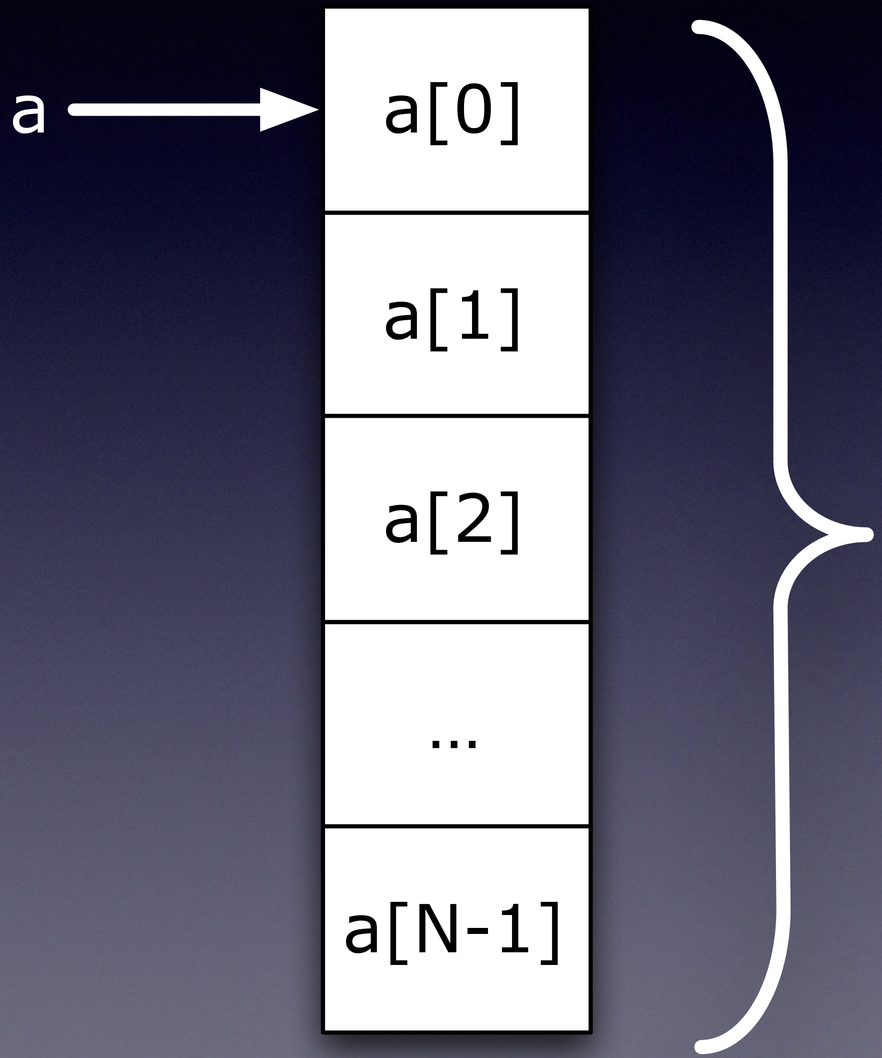
```
array<int,100> ar;
```



```
array<float,3> vec;
```

how are arrays laid out in RAM?

```
array<int,N> a;
```



N items in a row, consecutively

Arrays

- The size of an array, once declared, can't be modified

Static size

once declared, its size cannot be enlarge or shrunk

2.1 Arrays typedef

typedef 'ing is quite **useful** with **array<>**

```
const int N = 3;
const int NPER = 55;

typedef  array<float,N> TVector;
typedef  array<int,NPER> TAges;
```

…and types
typedef 'ing

```
const int N = 3;
const int NPER = 55;

typedef  array<float,N> TVector;
typedef  array<int,NPER> TAges;
```

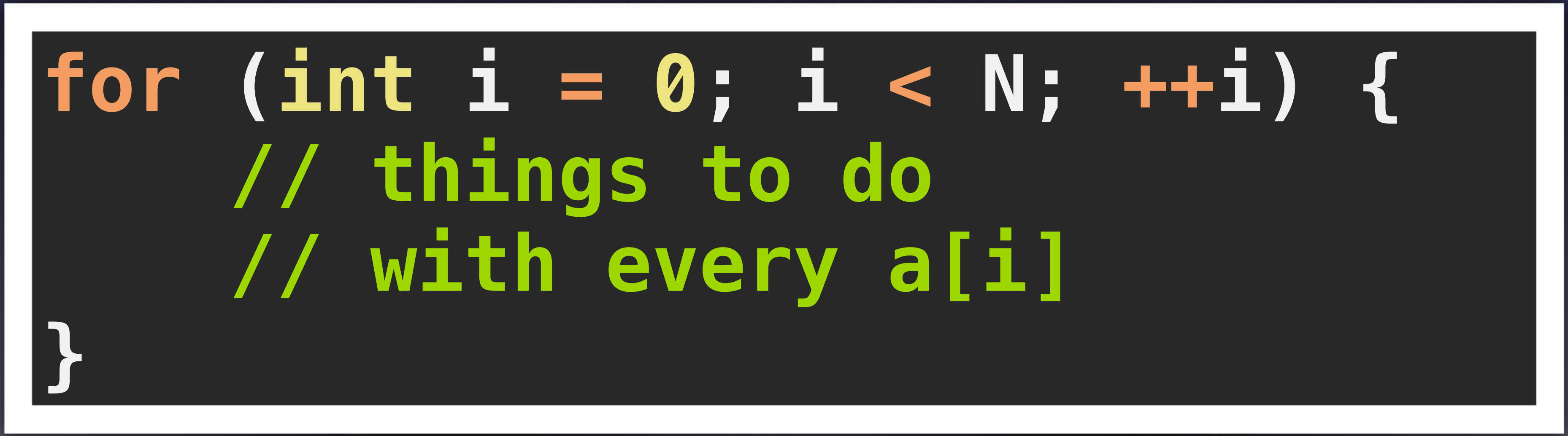
new names are easier to read;
***necessary** when parameters*

traversing the whole array

if

```
array<int,N> a;
```

N a const previously defined



```
for (int i = 0; i < N; ++i) {
     // things to do
     // with every a[i]
}
```

declaring arrays

Quick {{initialisation}}

```
typedef array<float,3> TVector;

TVector v = {{3, 5, -2}}; // quick init
TVector v = {{1,2}}; // first 2 initialised
```

access to the elements

- `a[0]` is the first element
- Typically

```
for (int i=0; i < N; i++)
  cout << a[i] << endl;
```

.size() == N

- `a[0]` is the first element
- Typically

```
for (int i=0; i < a.size(); i++)
  cout << a[i] << endl;
```

Exercices

- Define an array able to contain:

1. The rainfalls in a year, day to day
2. Marks in a class of N_STUDENTS students
3. TPoly, a series of coefficients for a polynomial, for example 3, -2, 5 would be: $5x^2 - 2x + 3$

array<> *can be treated* as ‘single-unit’ vars

```
array <int,3> a, b;
```

`a = b` ✓

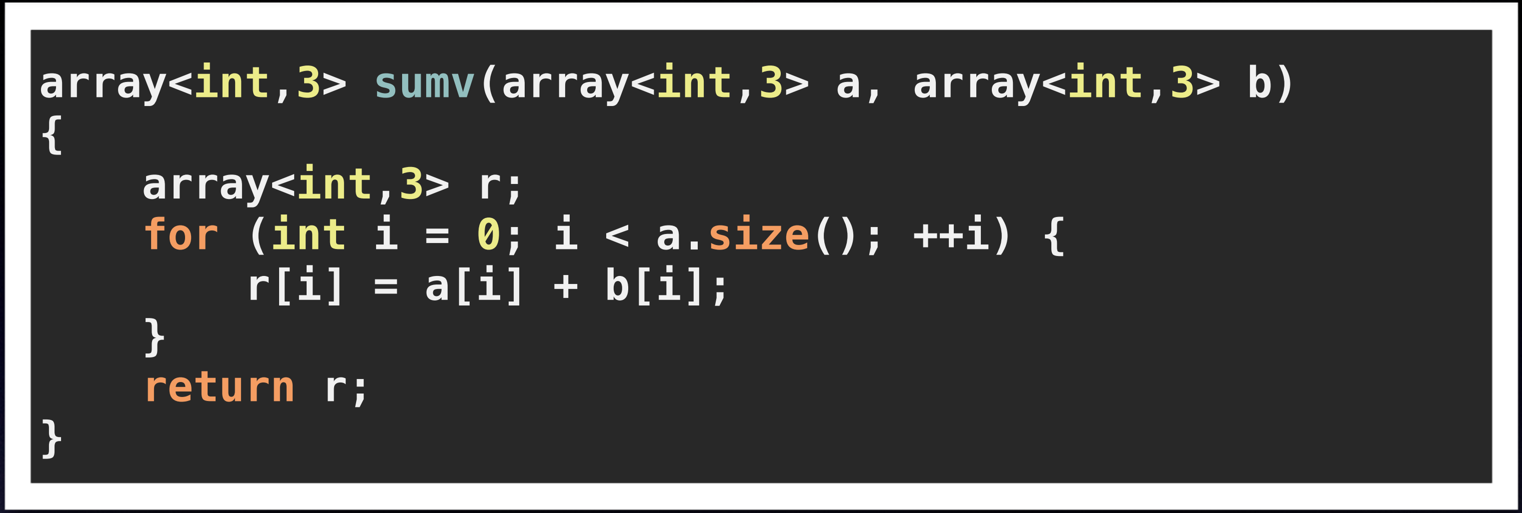
`a == b` ✓

`a > b` ✓

…

typedef

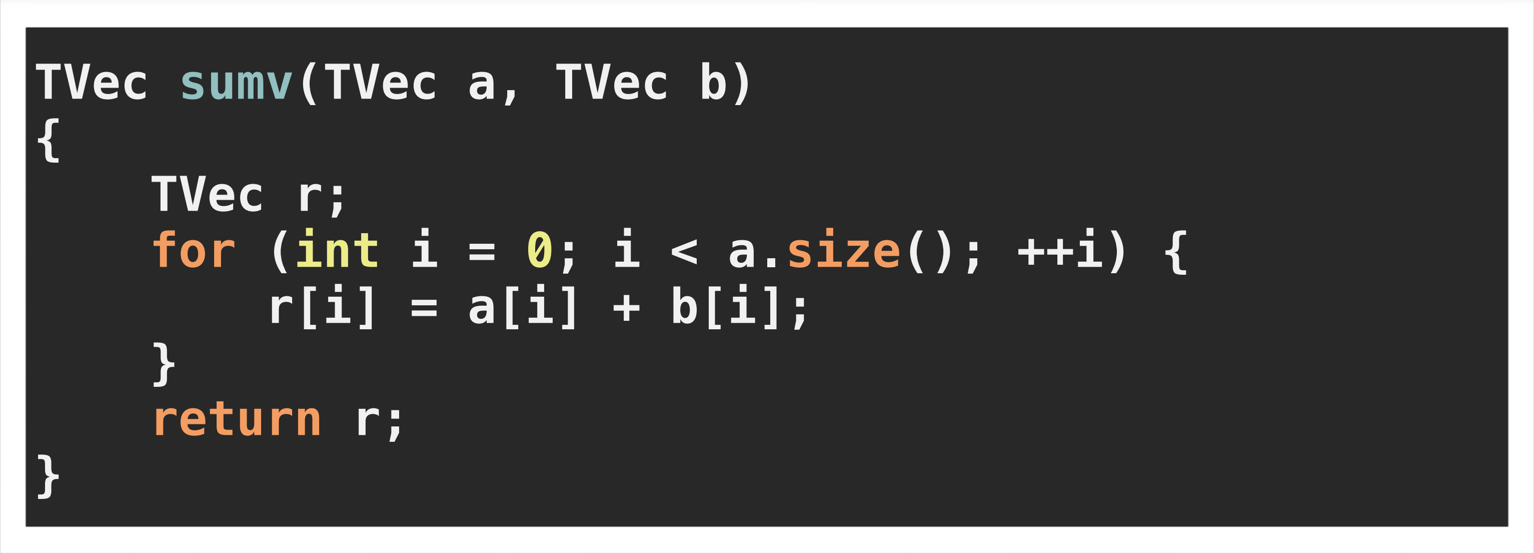
sending and receiving array<>



```
array<int,3> sumv(array<int,3> a, array<int,3> b)
{
    array<int,3> r;
    for (int i = 0; i < a.size(); ++i) {
        r[i] = a[i] + b[i];
    }
    return r;
}
```

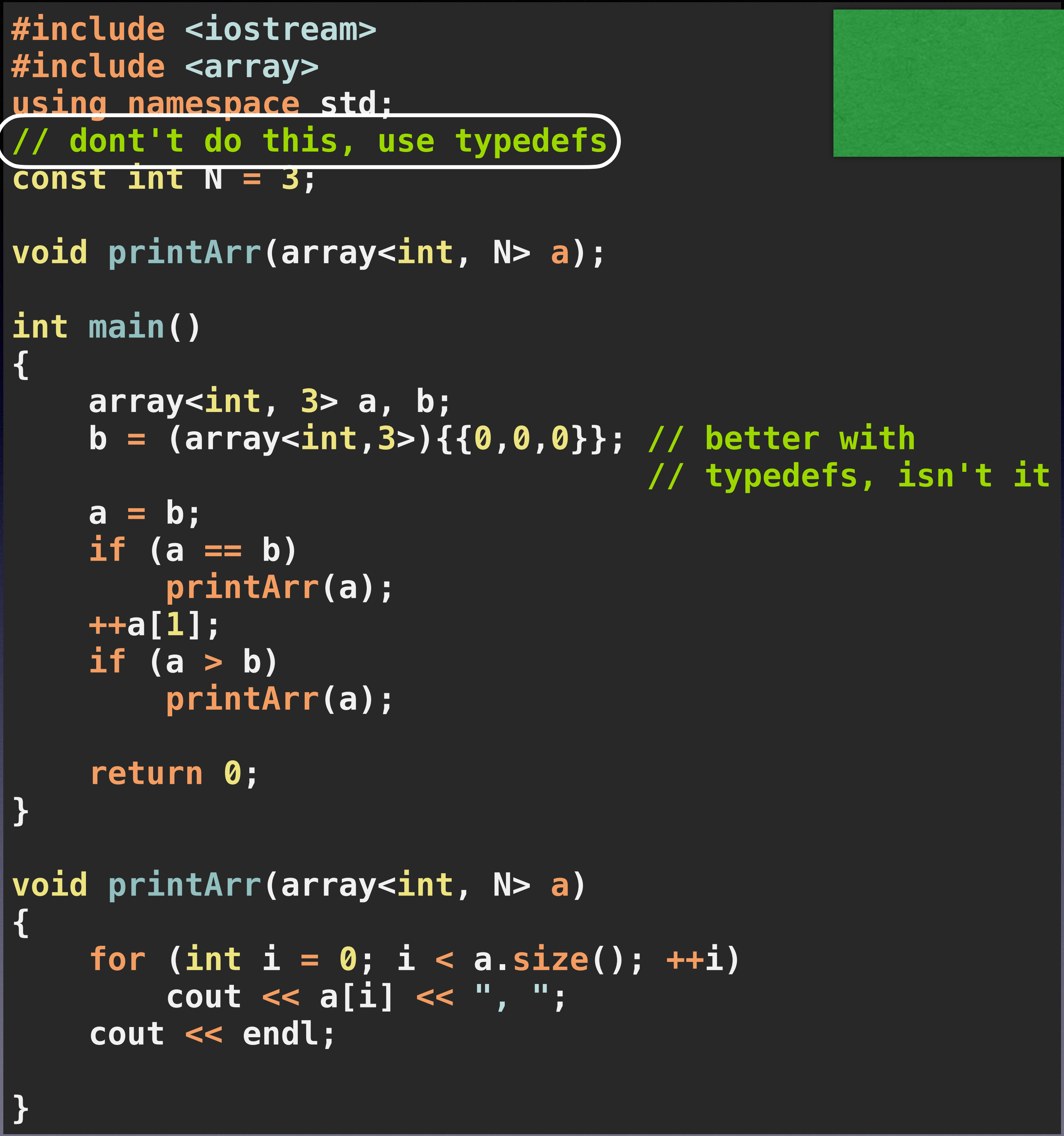
better with…

```
typedef array<int,3> TVec;
```

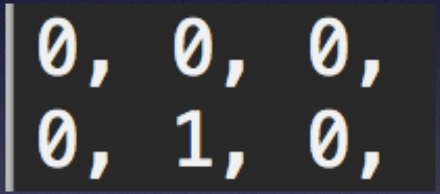


```
TVec sumv(TVec a, TVec b)
{
    TVec r;
    for (int i = 0; i < a.size(); ++i) {
        r[i] = a[i] + b[i];
    }
    return r;
}
```

example



```
#include <iostream>
#include <array>
using namespace std;
// dont't do this, use typedefs
const int N = 3;

void printArr(array<int, N> a);

int main()
{
    array<int, 3> a, b;
    b = (array<int,3>){{0,0,0}}; // better with
                                 // typedefs, isn't it
    a = b;
    if (a == b)
        printArr(a);
    ++a[1];
    if (a > b)
        printArr(a);

    return 0;
}

void printArr(array<int, N> a)
{
    for (int i = 0; i < a.size(); ++i)
        cout << a[i] << ", ";
    cout << endl;

}
```



```
0, 0, 0,
0, 1, 0,
```

example

Sum all the elements of an array of integers and return the result

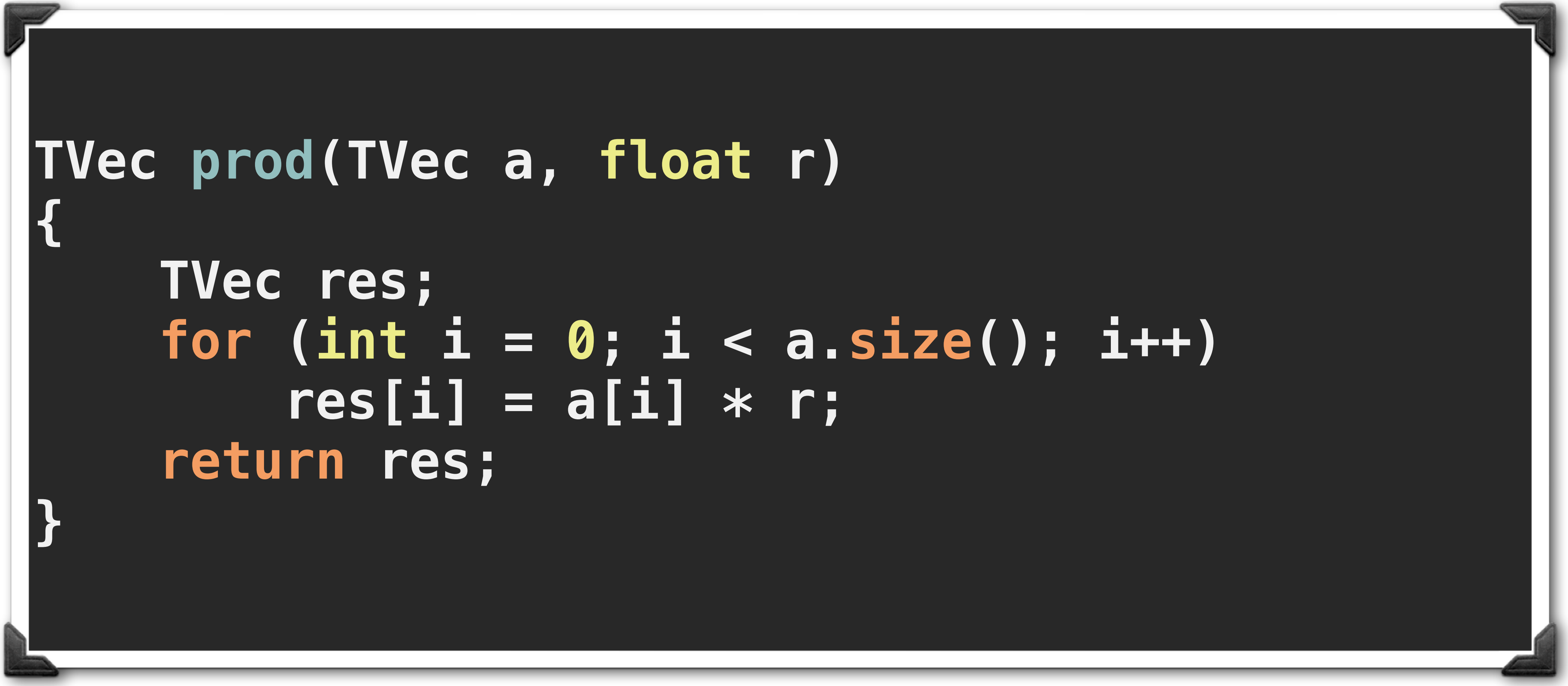
```
int sumarr(array <int,N> a)
{
    int result = 0;
    for ( int i = 0; i < N; ++i )
        result += a[i];
    return result;
}
```

Sol

$$ra = \{ra_i\}$$

- Build a function that multiplies a vector by a number

```
TVec prod(TVec a, float v)
```



```
TVec prod(TVec a, float r)
{
    TVec res;
    for (int i = 0; i < a.size(); i++)
        res[i] = a[i] * r;
    return res;
}
```

Exercices

Define the **interface** (prototype) of:

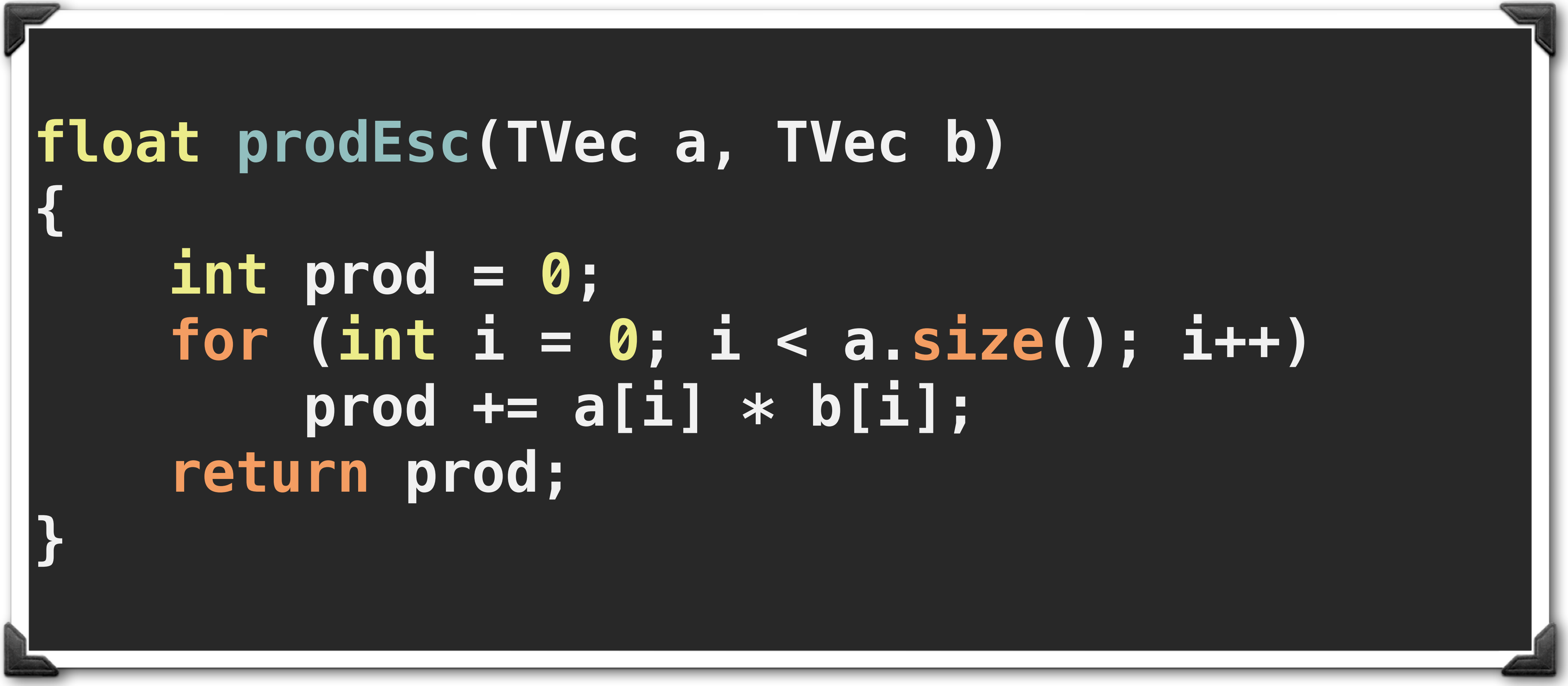
1. a function that returns the value of TPoly at a given **x**
2. a function that returns the derivative of a TPoly
3. a function to return the scalar product of 2 vectors

Solution

$$\mathrm{prodEsc}(\mathbf{a},\mathbf{b}) = \mathbf{a}\cdot\mathbf{b} = \sum_{i=0}^{N-1} a_i b_i$$

- Build the Scalar Product function:

```
float prodEsc(TVec a, TVec b)
```



```
float prodEsc(TVec a, TVec b)
{
    int prod = 0;
    for (int i = 0; i < a.size(); i++)
        prod += a[i] * b[i];
    return prod;
}
```

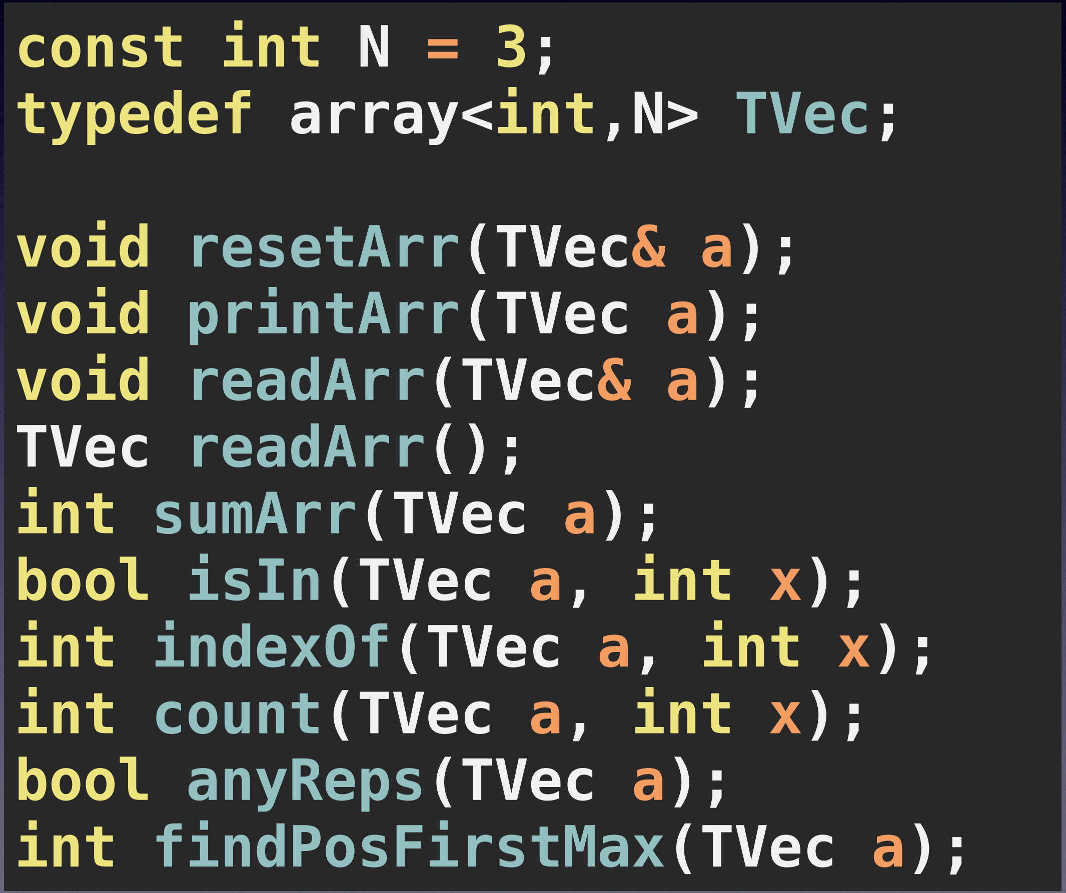
exercises

Given the array:

typedef **array<int,N> TVec;**

(N a const),

1. make all its values be 0
2. print all of its values sep by spaces
3. read all of its values from keyboard
4. add up all of its values
5. guess wether an value **x** is in the array
6. find the first position in it of a given **x**
7. count the times an element appears
8. guess if any element is repeated
9. find the index of the first time the max values is



```
const int N = 3;
typedef array<int,N> TVec;

void resetArr(TVec& a);
void printArr(TVec a);
void readArr(TVec& a);
TVec readArr();
int sumArr(TVec a);
bool isIn(TVec a, int x);
int indexOf(TVec a, int x);
int count(TVec a, int x);
bool anyReps(TVec a);
int findPosFirstMax(TVec a);
```

Open Arrays

In order to use fewer cells than the capacity of our arrays, we soon will see how to build arrays, yet with their static size, but with an extra integer number telling the actual amount of occupied cells.

We will need structures for this, so before, we need to see the struct part of this long chapter