Arrays

1d and 2d

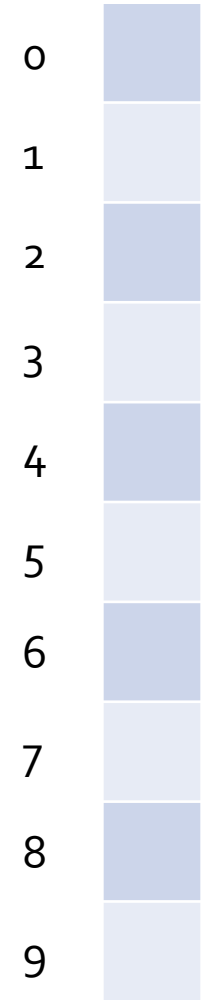
Christos Katsoulas

An array has:

- A name
- One or more dimensions
- A data type

- Refer to a single element of the array:
- temp[1] is the second element.

float temp[10]



Be aware

In C++ we start indexing from 0.
So, temp[0] is the first element of the array.

Input to array

- Declare an array
  - `int records[5]={13,7,9,5,16};`
- Assign a value to an element
  - `records[1]=7;`
- Use the keyboard

```
for(i=0;i<5;i++)
   cin >> records[i];
```

records[5]

0	13
1	7
2	9
3	5
4	16

Output to the console

```
for(i=0;i<5;i++)
    cout << records[i];
```

records[5]

0	13
1	7
2	9
3	5
4	16

Advantages & Disadvantages

- Read and write an element in O(1) time.
  - array[10]=n; //insert n to the 11th position.
  - cout<<array[4]; //output the 5th position
- Minimum memory
- Easy to use

VS

- Define the size of the array at the beginning (when declaring it)
- Insertion and deletion of an element is a hard task

Find maximum value and position

```
max=records[0];
pos_max=0
for(i=0;i<5;i++) {
      if(records[i]>max)
      {
      max=records[i];
      pos_max=i;
      }
}
```

records[5]

0	13
1	7
2	9
3	5
4	16

Find minimum value and position

```
min=records[0];
pos_min=0;
for (i=0;i<5;i++)
{
 if (records[i]<min)
 {
     min=records[i];
     pos_min=i;
 }
}
```

records[5]

0	13
1	7
2	9
3	5
4	16

How many and where > 15

```
count=0;
for(i=0;i<5;i++)
 if(records[i]>15)
      {
      count++;
      cout << i;
      }
```

records[5]

0	13
1	7
2	9
3	5
4	16

Sum and average

How many elements are smaller than the average

```
sum=0;
for(i=0;i<5;i++){
  sum=sum+records[i];
}
average=sum/5;
count=0;   //μετρητής
for(i=0;i<5;i++){
  if(records[i]<average)
      count++;
}
```

records[5]

0	13
1	7
2	9
3	5
4	16

Product of the elements

```
prod=1;
for(i=0;i<5;i++)
 prod=prod*records[i];
cout << prod;
```

records[5]

0	13
1	7
2	9
3	5
4	16

Delete an element

- Suppose we want to delete the 3rd element:

```
for (i=2; i<4; ++i) {
        records[i]=records[i+1];
}
```

- The elements 3 and 4 of the array are shifted one position up.
- In the 5th position number 16 is staying, as there are no more elements to move.

records[5]

0	13
1	7
2	9
3	5
4	16

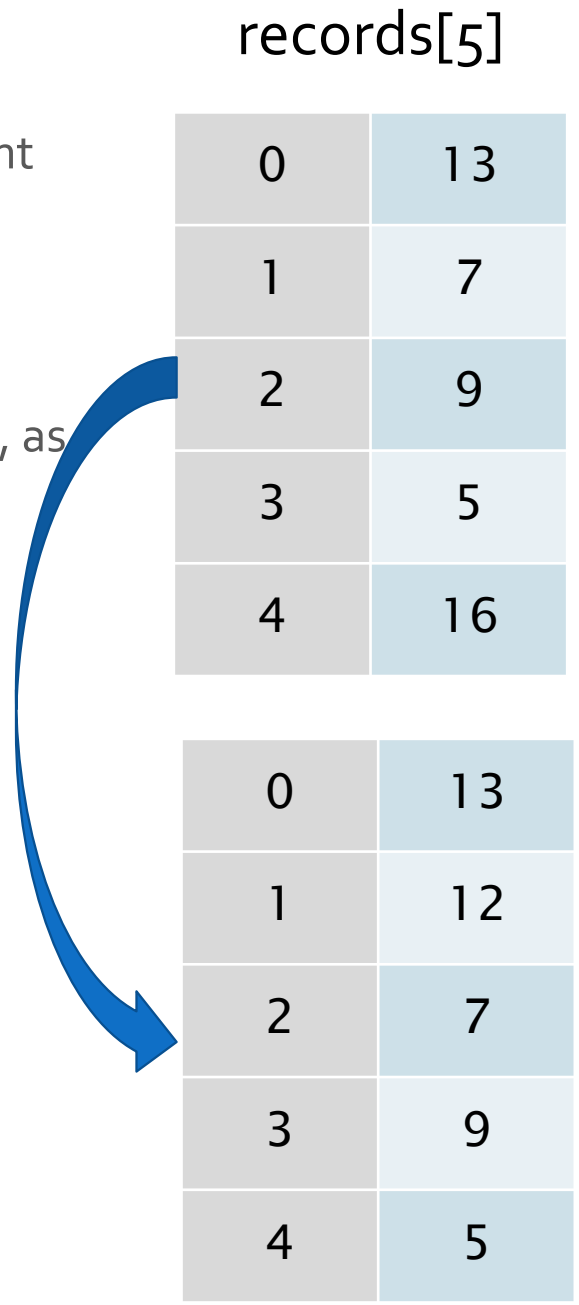
0	13
1	7
2	5
3	16
4	16

Insert an element

- Let's suppose we want to insert a new element (e.g. 12) in the 1st position of the array.
- All elements from position 2 and below must shift.
- The last element in position 4 (16) will be lost, as the array has a size of 5.

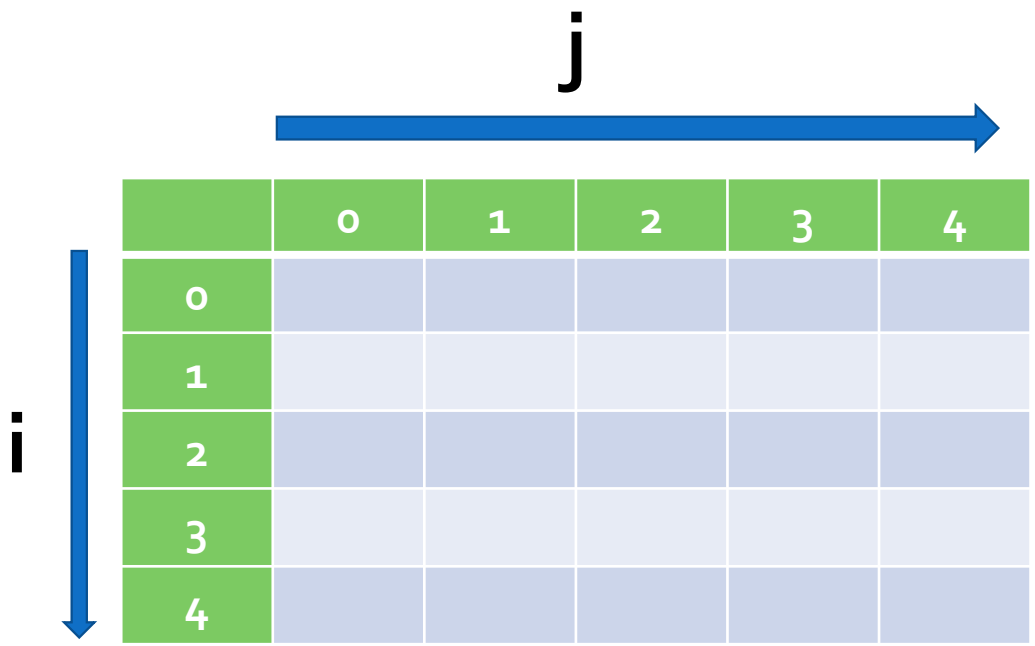
```
for (i=4; i>1; i--) {
        records[i]=records[i+1];
}
records[1]=12;
```

- We start from the last element and move upwards.
- In every position we assign the previous element.
- In position 1 we assign 12.



2D Arrays

Loop inside loop




```
for (int i =0; i<5;i++){
  for (int j=0;j<5;j++){
    cout<<array[i][j];
  }
}
```

Input – output from 2-d array:

- For i=0, traverse the 1st line, increasing j: (for j from 0 to 4).
- For i=1, traverse the 2nd line, increasing j: (for j from 0 to 4).
- Keep going until i=4 and repeat the same steps.
- At the end, 5x5 elements will be printed.

In reality:

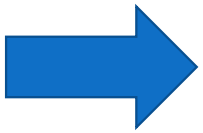
- 1d array where each element is an 1-d array.
- E.g. array 10x5:
  - 1-d array of 10 elements
  - Each element is an 1-d array of 5 elements

Vice versa:

0,0	0, 1	0, 2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9

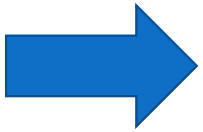
0	1	2	3	4	5	6	7	8	9

Initialization



```
for ( i = 0; i <= n; ++i ) {
      for ( j = 0; j <= n; ++j ) {
            array[ i ][ j ] = 0;
      }
}
```

Average per row



```
for ( i = 0; i <= n; ++i ) {
      int sum=0;
      for ( j = 0; j <= n; ++j ) {
            sum=sum+array[ i ][ j ];
      }
      cout<<sum/n;
}
```