



**AAB University**

**Faculty of Computer Sciences**

---

**Programimi i Orientuar në Objekte**

Java 7:

**Klasa *array* dhe *vector***

**Doc. Dr. Mentor Hamiti**

[mentor.hamiti@universitetiaab.com](mailto:mentor.hamiti@universitetiaab.com)

---



- Fushat (Arrays) dhe Vektorët (Vectors)
- Shembul të shfrytëzimit të Fushave (Arrays)
- Klasat në kuadër të librarisë standarde: **array** dhe **vector**



- **Fusha një-dimensionale (Vektori)** paraqet strukturë të të dhënave me karakteristikat vijuese:
  - **E fundme**
  - **Me madhësi fikse**
  - **E renditur**
  - **Homogjene**
  - **Me qasje direkte**

Cilat operacione janë të definuara mbi fusha?!



- Shembull:

**int numbers [10]**

numbers

[0]	Firs element
[1]	Second element
[2]	Third element
.	.
.	.
.	.
[9]	Last element



- Fushën e shoqërojnë karakteristikat vijuese:
  1. Numri i elementeve (***Number***)
  2. Adresa bazë ku fillon fusha (***Base***)
  3. Numri i lokacioneve memoruese i rezervuar për cdo element të fushës (***SizeOfElement***)



- Shembull:

**int** data [10]

**float** money [6]

**char** letters [26]

- *Funksioni I qasjes ë elemente të fushës në C++, përmes adresës bazë dhe indeksit:*

$$\text{Adres(Index)} = \text{Base} + \text{Index} * \text{SizeOfElement}$$

data [0]	100 + ( 0 * 4)	Address 100
data [8]	100 + ( 8 * 4)	Address 132
letters [1]	160 + ( 1 * 1)	Address 161
money [3]	140 + ( 3 * 4)	Address 152
letters [25]	160 + ( 25 * 1)	Address 185



- Fusha dy-dimensionale paraqet mënyrën natyrore të paraqitjes së të dhënave të renditura në rreshta dhe kolona:

```
const int row;
```

```
const int col;
```

```
int table [row][col]
```

```
int table [10][6]
```

		Columns			
		[0]	[1]	...	[5]
Rows	[0]				
	[1]				
	[2]				
	.	.	.	...	.
	.	.	.	...	.
	[9]				



- Fushat mund të jenë të dimensioneve të ndryshme:

- For one dimension:

[n]

{0, ..., n-1}

- For two dimensions:

[3][3]

{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)}

- For three dimensions:

[2][2][2]

{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)}

Running time?





- Fushat (Arrays) dhe Vektorët (Vectors)
- Shembul të shfrytëzimit të Fushave (Arrays)
- Klasat në kuadër të librarisë standarde: **array** dhe **vector**



## ■ Shembulli 1:

```
//Initializing an array's elements to zeros and printing the array
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    int n[ 10 ];    // n is an array of 10 integers

    // initialize elements of array n to 0
    for ( int i = 0; i < 10; ++i )
        n[ i ] = 0; // set element at location i to 0

    cout << "Element" << setw( 13 ) << "Value" << endl;

    // output each array element's value
    for ( int j = 0; j < 10; ++j )
        cout << setw( 7 ) << j << setw( 13 ) << n[ j ] << endl;

    cin.get();
    return 0;
}
```

Element	Value
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0



## ■ Shembulli 2:

```
// Initializing an array in a declaration.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    // use initializer list to initialize array n
    int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

    cout << "Element" << setw( 13 ) << "Value" << endl;

    // output each array element's value
    for ( int i = 0; i < 10; ++i )
        cout << setw( 7 ) << i << setw( 13 ) << n[ i ] << endl;

    cin.get();
    return 0;
}
```

Element	Value
0	32
1	27
2	64
3	18
4	95
5	14
6	90
7	70
8	60
9	37



## ■ Shembulli 3:

```
// Set array s to the even integers from 2 to 20
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    const int arraySize = 10;

    int s[ arraySize ]; // array s has 10 elements

    for ( int i = 0; i < arraySize; ++i ) // set the values
        s[ i ] = 2 + 2 * i;

    cout << "Element" << setw( 13 ) << "Value" << endl;

    // output contents of array s in tabular format
    for ( int j = 0; j < arraySize; ++j )
        cout << setw( 7 ) << j << setw( 13 ) << s[ j ] << endl;

    cin.get();
    return 0;
}
```

Element	Value
0	2
1	4
2	6
3	8
4	10
5	12
6	14
7	16
8	18
9	20



## ■ Shembulli 4:

```
// Compute the sum of the elements of the array
#include <iostream>
using namespace std;

int main()
{
    const int arraySize = 10; // constant variable indicating size of array
    int a[ arraySize ] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };

    int total = 0;

    for ( int i = 0; i < arraySize; ++i )
        total += a[ i ];

    cout << "Total of array elements: " << total << endl;

    cin.get();
    return 0;
}
```

```
Total of array elements: 849
```



## Shembulli 5:

```
// Bar chart printing program
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    const int arraySize = 11;
    int n[ arraySize ] = { 0, 0, 0, 0, 0, 0, 1, 2, 4, 2, 1 };

    cout << "Grade distribution:" << endl;

    // for each element of array n, output a bar of the chart
    for ( int i = 0; i < arraySize; ++i )
    {
        // output bar labels ("0-9:", ..., "90-99:", "100:" )
        if ( i == 0 )
            cout << " 0-9: ";
        else if ( i == 10 )
            cout << " 100: ";
        else
            cout << i * 10 << "-" << ( i * 10 ) + 9 << ": ";

        // print bar of asterisks
        for ( int stars = 0; stars < n[ i ]; ++stars )
            cout << '*';

        cout << endl; // start a new line of output
    } // end for

    cin.get(); return 0;
}
```

```
Grade distribution:
 0-9:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: *
70-79: **
80-89: ****
90-99: **
100: *
```



## ■ Sembulli 6:

```
// Roll a six-sided die 6,000,000 times
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    const int arraySize = 7;           // ignore element zero
    int frequency[ arraySize ] = {};  // initialize elements to 0

    srand( time( 0 ) );                // seed random number generator

    // roll die 6,000,000 times; use die value as frequency index
    for ( int roll = 1; roll <= 6000000; ++roll )
        ++frequency[ 1 + rand() % 6 ];

    cout << "Face" << setw( 13 ) << "Frequency" << endl;

    // output each array element's value
    for ( int face = 1; face < arraySize; ++face )
        cout << setw( 4 ) << face << setw( 13 ) << frequency[ face ] << endl;

    cin.get(); return 0;
}
```

Face	Frequency
1	1000319
2	1001198
3	999219
4	998883
5	1000363
6	1000018



## Shembulli 7:

```
// Poll analysis program
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    // define array sizes
    const int responseSize = 20; // size of array responses
    const int frequencySize = 6; // size of array frequency

    // place survey responses in array responses
    const int responses[ responseSize ] = { 1, 2, 5, 4, 3, 5, 2, 1, 3,
                                             1, 4, 3, 3, 3, 2, 3, 3, 2, 2, 5 };

    // initialize frequency counters to 0
    int frequency[ frequencySize ] = {};

    // for each answer, select responses element and use that value
    // as frequency subscript to determine element to increment
    for ( int answer = 0; answer < responseSize; ++answer )
        ++frequency[ responses[ answer ] ];

    cout << "Rating" << setw( 17 ) << "Frequency" << endl;

    // output each array element's value
    for ( int rating = 1; rating < frequencySize; ++rating )
        cout << setw( 6 ) << rating << setw( 17 ) << frequency[ rating ] << endl;

    cin.get(); return 0;
}
```

```
Rating          Frequency
1              3
2              2
3              5
4              3
5              2
```





- Fushat (Arrays) dhe Vektorët (Vectors)
- Shembul të shfrytëzimit të Fushave (Arrays)
- Klasat në kuadër të librarisë standarde: **array** dhe **vector**



- Fusha (**array**) parqet grup të lokacioneve kontinuale memoruese të tipit të njejtë
- Specifikimi i numrit të elementeve dhe tipit të tyre bëhet në formën vijuese:

**array**< type, arraySize > *arrayName*;

- Nocioni <**type**, **arraySize**> ka të bëjë me inkuadrimin e klasës “**class template**“
- Kompajleri rezervon kapacitetin përkatës të fushës, bazuar në tipin e elementeve **type** dhe në madhësinë e fushës **arraySize** së deklaruar



- Procedura vijuese në C++ **range-based for statement** ofron mundësinë e shfrytëzimit të të gjitha elementeve të fushës pa përdorim të numëruesve
- Sintaksa për procedurën në fjalë:

**for** ( rangeVariableDeclaration : expression )  
*statement*

- rangeVariableDeclaration *përmban tipin dhe identifikatorin* (p.sh., int item)
- *type në kuadër të rangeVariableDeclaration patjetër të jetë e njejtë me type dhe elementet e array*



## ■ Shembulli 8:

```
//Range-Based for Statement
#include <iostream>
#include <array>
using namespace std;

int main()
{
    array <int, 5> items = { 1, 2, 3, 4, 5};

    cout<<"\nItems before modificaton: ";
    for(int item: items)
        cout<<item<<" ";

    //multiply the elements of items by 2
    for(int &itemR: items)
        itemR *= 2;

    cout<<"\n\nItems after modificaton: ";
    for(int item: items)
        cout<<item<<" ";

    cin.get(); return 0;
}
```

```
Items before modificaton: 1 2 3 4 5
Items after modificaton: 2 4 6 8 10
```



```
11
21
31
```

```
#include <iostream>

using namespace std;

int main()
{
    int myarray[3] = {10,20,30};

    for (int i=0; i<3; ++i)
        ++myarray[i];

    for (int elem : myarray)
        cout << elem << '\n';

    cin.get(); return 0;
}
```

```
11
21
31
```

```
#include <iostream>
#include <array>
using namespace std;

int main()
{
    array<int,3> myarray = {10,20,30};

    for (int i=0; i<myarray.size(); ++i)
        ++myarray[i];

    for (int elem : myarray)
        cout << elem << '\n';

    cin.get(); return 0;
}
```



- Klasa **vector** e Librarisë Standarde në C++ është e ngjajshme me klasën **array**, por konsiderohet si më fleksibile
  - *Kjo klasë **vector**, për dallim nga **array** posedon mundësinë e zvoglimit dhe rritjes së madhësisë*
- Zakonisht (by default), të gjitha elementet e vektorit janë të inicializuar me **0**

***vector** <int> myVector (10);*



## ■ Shembulli 9:

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    vector <int> myVector;           //Vector to store integers
    myVector.push_back(3);          //Add 3 onto the vector
    myVector.push_back(10);         //Add 10 to the end
    myVector.push_back(33);         //Add 33 to the end

    for(int x=0; x<myVector.size(); x++)
    {
        cout<<myVector[x]<<" ";    //Output: 3 10 33
    }

    cin.get(); return 0;
}
```

3 10 33



- Differenca kryesore mes klasëve **vector** dhe **array** qëndron në faktin se vektori mund të ndryshon në mënyrë dinamike për të akomoduar vendosjen e numrit të kërkuar të elementeve





- Ushtrimi 1:

*Të shkruhet kodi në C++ që ofron shfaqjen e të dhënave vijuese:*

```
Read vector:
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 5

Print vector:
A[5] = < 1 , 2 , 3 , 4 , 5 >_
```



- Ushtrimi 2:

Të shkruhet kodi në C++ që ofron shfaqjen e të dhënave vijuese:

```
Read matrix:
```

```
A[0][0] = 1
```

```
A[0][1] = 2
```

```
A[0][2] = 3
```

```
A[0][3] = 4
```

```
A[1][0] = 5
```

```
A[1][1] = 5
```

```
A[1][2] = 5
```

```
A[1][3] = 5
```

```
A[2][0] = 1
```

```
A[2][1] = 2
```

```
A[2][2] = 3
```

```
A[2][3] = 4
```

```
Print matrix:
```

```
1      2      3      4
```

```
5      5      5      5
```

```
1      2      3      4
```



- Projekti 1 (*opcional, për bonus pikë 😊*):

**//Complete the following code for sorting matrice:**

```
# include .....
```

```
.....
```

```
int main()
```

```
{
```

```
    readMatrice(A,n,m);
```

```
    printMatrice(A,n,m);
```

```
    convertMatriceToVector(A,n,m,V);
```

```
    writeVector(V);
```

```
    sortVector(V);
```

```
    writeVector(V);
```

```
    convertVectorToMatrice(V,n,m,A);
```

```
    printMatrice(A,n,m);
```

```
cin.get(); return 0;
```

```
}
```



- Pyetje eventuale?!

