

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΕΣ ΤΕΧΝΙΚΕΣ 2024-2025

<https://helios.ntua.gr/course/view.php?id=869>

Διδάσκοντες:

Βασίλης Βεσκούκης, Καθ.
Νίκος Λεονάρδος, Επ.Καθ.
Άρης Παγουρτζής, Καθ.
Νίκος Παπασπύρου, Καθ.
Σωτήρης Κοκόσης, ΕΔΙΠ
Πέτρος Ποτίκας, ΕΔΙΠ
Γιώργος Σιόλας, ΕΔΙΠ

14/3/2025

progtech@courses.softlab.ntua.gr

Διαφάνειες παρουσιάσεων

- ✓ Η γλώσσα προγραμματισμού C++
- ✓ Αρχές αντικειμενοστρεφούς προγραμματισμού
- ✓ Δομές δεδομένων



Πίνακες ως ΑΤΔ (αφηρημένοι τύποι δεδομένων)

Πίνακες ως ΑΤΔ

- Σύνολο «δεικτών» (indices), I
- Σύνολο «τιμών» (values), V
- Βασική πράξη: **προσπέλαση** στοιχείου
$$a[i] \in V \quad \text{όπου } i \in I$$
- Μονοδιάστατοι πίνακες: $I = \{1, 2, 3, \dots, n\}$
- Πολυδιάστατοι πίνακες (d διαστάσεων):
$$I = \{1, 2, 3, \dots, n_1\} \times \{1, 2, 3, \dots, n_2\} \times \dots \times \{1, 2, 3, \dots, n_d\}$$

Πίνακες ως ΑΤΔ

- Συνήθως υλοποιούνται με κάποιο ΣΤΔ πινάκων (arrays)
- Ο ΣΤΔ του **μονοδιάστατου πίνακα** επαρκεί για την υλοποίηση κάθε ΑΤΔ πίνακα

```
int data[SIZE]; // στατικά
```

```
int *data = new int[SIZE]; // δυναμικά
```

- Κόστος προσπέλασης: **$O(1)$**
- Συνάρτηση *loc* : $\mathbf{I} \rightarrow \{0, 1, 2, \dots \text{SIZE} - 1\}$
υπολογίζει τη θέση ενός στοιχείου του ΑΤΔ πίνακα στο ΣΤΔ πίνακα της υλοποίησης

Πίνακες ως ΑΤΔ

- ΑΤΔ πίνακα δύο διαστάσεων $n \times m$

$i = 0$	0	1	2	3	4	5
$i = 1$	6	7	8	9	10	11
$i = 2$	12	13	14	15	16	17
	$j = 0$	1	2	3	4	5

$rows = 3$
 $cols = 6$

- Αρίθμηση κατά γραμμές

$$loc(i, j) = cols * i + j$$

$$i = loc / cols, j = loc \% cols$$

Πίνακες ως ΑΤΔ

(0,0)->0	(0,1)->1	(0,2)->2	(0,3)->3	(0,4)->4	(0,5)->5
(1,0)->6	(1,1)->7	(1,2)->8	(1,3)->9	(1,4)->10	(1,5)->11
(2,0)->12	(2,1)->13	(2,2)->14	(2,3)->15	(2,4)->16	(2,5)->17



```
for (i=0; i < rows; i++)  
  for (j=0; j < cols; j++)  
    loc = cols * i + j ;
```



```
for (loc=0; loc < rows*cols; loc++){  
  i = loc / cols;  
  j = loc % cols;  
}
```

0 ← (0,0)	1 ← (0,1)	2 ← (0,2)	3 ← (0,3)	4 ← (0,4)	5 ← (0,5)	6 ← (1,0)	7 ← (1,1)	8 ← (1,2)	9 ← (1,3)	10 ← (1,4)	11 ← (1,5)	12 ← (2,0)	13 ← (2,1)	14 ← (2,2)	15 ← (2,3)	16 ← (2,4)	17 ← (2,5)
--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------

Πίνακες ως ΑΤΔ

- ΑΤΔ πίνακα δύο διαστάσεων $n \times m$

$i = 0$	0	3	6	9	12	15
$i = 1$	1	4	7	10	13	16
$i = 2$	2	5	8	11	14	17
	$j = 0$	1	2	3	4	5

$rows = 3$
 $cols = 6$

- Εναλλακτικά, αρίθμηση κατά στήλες

$$loc(i, j) = rows * j + i$$

$$i = loc \% rows, j = loc / rows$$

Πίνακες ως ΑΤΔ

(0,0)->0	(0,1)->3	(0,2)->6	(0,3)->9	(0,4)->12	(0,5)->15
(1,0)->1	(1,1)->4	(1,2)->7	(1,3)->10	(1,4)->13	(1,5)->16
(2,0)->2	(2,1)->5	(2,2)->8	(2,3)->11	(2,4)->14	(2,5)->17



```
for (j=0; j < cols; j++) {  
    for (i = 0; i < rows; i++)  
        loc = rows * j + i;
```



```
for (loc=0; loc < rows*cols; loc++){  
    j = loc / rows;  
    i = loc % rows;  
}
```

0 ← (0,0)	1 ← (1,0)	2 ← (2,0)	3 ← (0,1)	4 ← (1,1)	5 ← (2,1)	6 ← (0,2)	7 ← (1,2)	8 ← (2,2)	9 ← (0,3)	10 ← (1,3)	11 ← (2,3)	12 ← (0,4)	13 ← (1,4)	14 ← (2,4)	15 ← (0,5)	16 ← (1,5)	17 ← (2,5)
--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------

Πίνακες ως ΑΤΔ

- ΑΤΔ κάτω τριγωνικού πίνακα $n \times n$

$i = 1$	0				
$i = 2$	1	2			
$i = 3$	3	4	5		
$i = 4$	6	7	8	9	
$i = 5$	10	11	12	13	14
	$j = 1$	2	3	4	5

$n = 5$

$$loc(i, j) = i(i - 1) / 2 + j - 1$$

- Ομοίως για συμμετρικούς πίνακες

Πίνακες ως ΑΤΔ

- ΑΤΔ τριδιαγώνιου πίνακα $n \times n$

$i = 1$	0	1			
$i = 2$	2	3	4		
$i = 3$		5	6	7	
$i = 4$			8	9	10
$i = 5$				11	12
	$j = 1$	2	3	4	5

$n = 5$

$$\begin{aligned} loc(i, j) &= 3(i - 1) - 1 + j - i + 1 \\ &= 2i + j - 3 \end{aligned}$$

Πίνακες ως ΑΤΔ

- ΑΤΔ αραιού πίνακα $n \times m$

$i = 1$	a_1				
$i = 2$			a_2		
$i = 3$		a_3	a_4		
$i = 4$					a_5
	$j = 1$	2	3	4	5

$rows = 4$
 $cols = 5$

- Υλοποίηση με δυαδικό πίνακα
- Υλοποίηση με τρεις πίνακες

$row = [1, 2, 3, 3, 4]$ $col = [1, 3, 2, 3, 5]$
 $val = [a_1, a_2, a_3, a_4, a_5]$

Κόστος
προσπέλασης;



Do it yourself!

Υλοποίηση ΑΤΔ πινάκων

ΑΤΔ για μονοδιάστατους πίνακες

(i)

```
template <typename T>
class Array {
public:
    Array(unsigned n = 0, int b = 0);
    Array(const Array &a);
    ~Array();
    Array & operator=(const Array &a);
    T & operator[](int i);
    const T & operator[](int i) const;
};
```

ΑΤΔ για μονοδιάστατους πίνακες

(ii)

protected:

T *data;

int base;

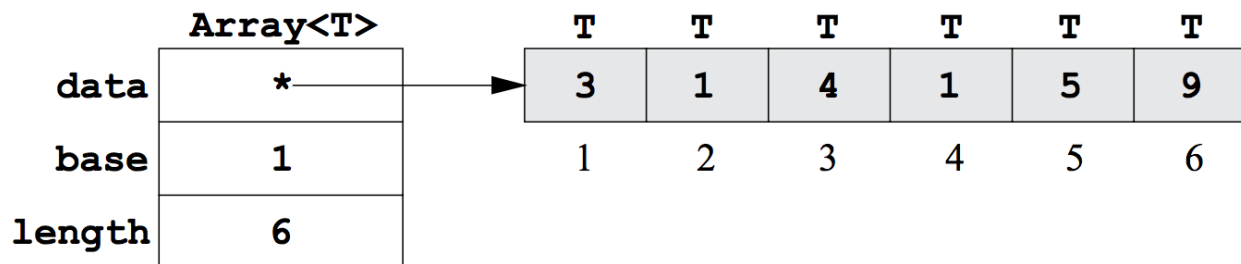
unsigned length;

unsigned loc(int i) const;

// may throw out_of_range

ΣΧΗΜΑ 4.1

Σχηματική αναπαράσταση της δομής αντικειμένου τύπου **Array<T>**



ΑΤΔ για μονοδιάστατους πίνακες (iii)

```
Array(unsigned n = 0, int b = 0)
    : data(new T[n]), base(b), length(n) {}
```

```
Array(const Array &a)
    : data(new T[a.length]), base(a.base),
      length(a.length) {
    for (unsigned i = 0; i < length; ++i)
        data[i] = a.data[i];
}
```

```
~Array() {
    delete [] data;
}
```

ΑΤΔ για μονοδιάστατους πίνακες

(iv)

```
Array & operator=(const Array &a) {  
    delete [] data;  
    base = a.base;  
    length = a.length;  
    data = new T[length];  
    for (unsigned i = 0; i < length; ++i)  
        data[i] = a.data[i];  
    return *this;  
}
```


ΑΤΔ για μονοδιάστατους πίνακες (v)

```
unsigned loc(int i) const {
    int di = i - base;
    if (di < 0 || di >= length)
        throw out_of_range("invalid index");
    return di;
}

T & operator[](int i) {
    return data[loc(i)];
}

const T & operator[](int i) const {
    return data[loc(i)];
}
```

ΑΤΔ για διδιάστατους πίνακες

(i)

```
template <typename T>
class Array2D {
public:
    Array2D(unsigned n = 0, unsigned m = 0,
            int bi = 0, int bj = 0);
    Array2D(const Array2D &a);
    ~Array2D();
    Array2D & operator=(const Array2D &a);
    T & select(int i, int j);
    const T & select(int i, int j) const;
    Row operator[](int i);
    ConstRow operator[](int i) const;
};
```

ΑΤΔ για διδιάστατους πίνακες

(ii)

protected:

T *data;

int baseRow, baseCol;

unsigned numRows, numCols;

unsigned loc(int i, int j) const;

// may throw out_of_range

ΑΤΔ για διδιάστατους πίνακες

(iii)

```
Array2D(unsigned n = 0, unsigned m = 0,
        int bi = 0, int bj = 0)
: data(new T[n*m]), baseRow(bi),
  baseCol(bj), numRows(n), numCols(m) {}

Array2D(const Array2D &a)
: data(new T[a.numRows * a.numCols]),
  baseRow(a.baseRow), baseCol(a.baseCol),
  numRows(a.numRows), numCols(a.numCols) {
  for (unsigned i = 0;
       i < numRows * numCols; ++i)
    data[i] = a.data[i];
}

~Array2D() { delete [] data; }
```

ΑΤΔ για διδιάστατους πίνακες

(iv)

```
Array2D & operator=(const Array2D &a) {  
    delete [] data;  
    baseRow = a.baseRow;  
    baseCol = a.baseCol;  
    numOfRows = a.numRows;  
    numOfCols = a.numCols;  
    data = new T[numRows * numCols];  
    for (unsigned i = 0;  
         i < numRows * numCols; ++i)  
        data[i] = a.data[i];  
    return *this;  
}
```

ΑΤΔ για διδιάστατους πίνακες

(v)

```
unsigned loc(int i, int j) const {
    int di = i - baseRow, dj = j - baseCol;
    if (di < 0 || di >= numRows ||
        dj < 0 || dj >= numCols)
        throw out_of_range("invalid index");
    return di * numCols + dj;
}

T & select(int i, int j) {
    return data[loc(i, j)];
}

const T & select(int i, int j) const {
    return data[loc(i, j)];
}
```

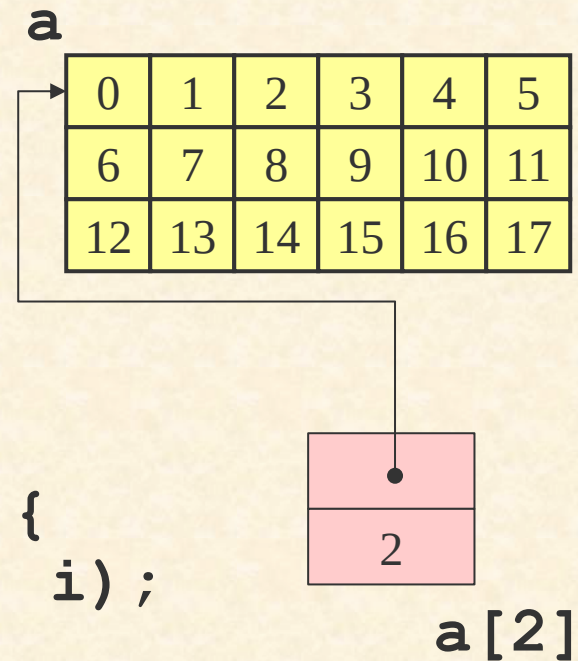
ΑΤΔ για διδιάστατους πίνακες

(vi)

```
Row operator [] (int i) {  
    return Row(*this, i);  
}
```

```
class Row {  
public:  
    Row(Array2D &a, int i)  
        : array2D(a), row(i) {}  
    T & operator[] (int i) const {  
        return array2D.select(row, i);  
    }  
}
```

```
private:  
    Array2D &array2D;  
    int row;  
};
```



ΑΤΔ για διδιάστατους πίνακες

(vii)

```
ConstRow operator[] (int i) const {  
    return ConstRow(*this, i);  
}
```

```
class ConstRow {  
public:
```

```
    ConstRow(const Array2D &a, int i)  
        : array2D(a), row(i) {}
```

```
    const T & operator[] (int i) const {  
        return array2D.select(row, i);  
    }
```

```
private:
```

```
    const Array2D &array2D;  
    int row;
```

```
};
```

Ίδιο με το **Row**
αλλά για σταθερούς πίνακες!



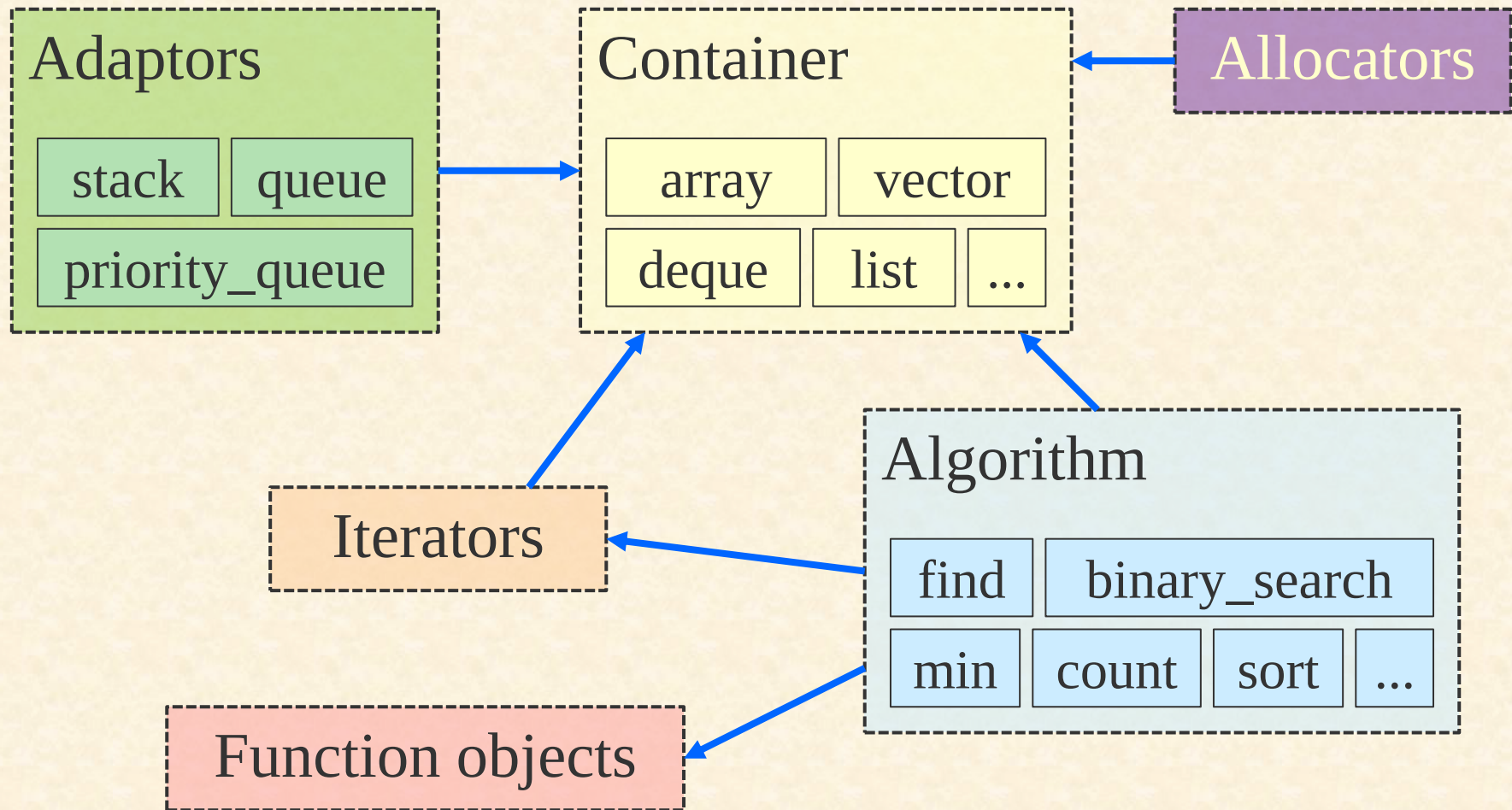
Standard Template Library

Εισαγωγή στην STL

Εισαγωγή στην STL

- Standard Template Library
- Μέρος της βιβλιοθήκης της C++
- Αρχική σχεδίαση: Alexander Stepanov
- Περιέχει χρήσιμες υλοποιήσεις:
 - δομών δεδομένων
 - αλγορίθμων
- Για να καταλάβουμε σε βάθος την STL:
 - templates
 - operator overloading
 - ιεραρχίες κλάσεων

Εισαγωγή στην STL



Εισαγωγή στην STL

Container class templates

www.cplusplus.com/reference/stl/

Sequence containers:

array C++11	Array class (class template)
vector	Vector (class template)
deque	Double ended queue (class template)
forward_list C++11	Forward list (class template)
list	List (class template)

Container adaptors:

stack	LIFO stack (class template)
queue	FIFO queue (class template)
priority_queue	Priority queue (class template)

Associative containers:

set	Set (class template)
multiset	Multiple-key set (class template)
map	Map (class template)
multimap	Multiple-key map (class template)

Unordered associative containers:

unordered_set C++11	Unordered Set (class template)
unordered_multiset C++11	Unordered Multiset (class template)
unordered_map C++11	Unordered Map (class template)
unordered_multimap C++11	Unordered Multimap (class template)

Συμβολοσειρά

- Τύπος `string`
 - περιτύλιγμα (wrapper) για array από `char`
- Χρήσιμες μέθοδοι
 - `size`, `length`
 - `c_str`
 - `empty`
 - `operator[]`, `substr`
 - `append`, `operator+=`
 - `compare`, `operator==`, `operator<`, κ.λπ.
 - `insert`, `replace`
 - `find`, `rfind`

Συμβολοσειρά

```
string s = "to be, or not to be? that is the question!";
string q = s;
cout << q << endl;
to be, or not to be? that is the question!

q += " _W.Shakespeare!";
cout << q << endl;
to be, or not to be? that is the question! _W.Shakespeare!

cout << "size=" << s.size()
      << " length=" << s.length() << endl;
size=42 length=42

cout << s[1] << endl;
o

cout << s.substr(3, 2) << endl;
be

s.replace(21, 4, "maybe THAT");
cout << s << endl;
to be, or not to be? maybe THAT is the question!
```

Συμβολοσειρά

```
s.insert(34, " indeed");
```

```
cout << s << endl;
```

to be, or not to be? maybe that is indeed the question!

```
int w = s.find("indeed", 0);
```

```
cout << w << endl;
```

35

```
w = s.find("really", 0);
```

```
cout << w << endl;
```

-1

```
const char *raw_pointer = s.data(); // or s.c_str()
```

```
cout << raw_pointer << endl;
```

to be, or not to be? maybe that is indeed the question!

```
w = s.find("o");
```

```
int x = s.rfind("o");
```

```
cout << w << " " << x << endl;
```

1 52

Containers: Πίνακας

- Τύπος `array<T, n>`
 - παρόμοια συμπεριφορά με τους απλούς πίνακες
 - σταθερού μεγέθους, γνωστού at compile time

```
#include <array>
```

```
...
```

```
array<int, 10> a;    // παρόμοιο με int a[10];
```

```
array<string, 20> s;
```

```
a[1] = 42;
```

```
s[2] = "Hello";
```


Containers: Πίνακας

- Χρήσιμες μέθοδοι της κλάσης `array`
 - `size()`: επιστρέφει το μέγεθος του πίνακα
 - `at()`: λειτουργεί ως `getter` και `setter`
 - `operator[]`: όπως το `at()` αλλά χωρίς έλεγχο ορίων

```
array<int, 10> a;  
for (int i=0; i < a.size(); i++)  
    a.at(i) = i * 10;           // set  
for (int i=0; i < a.size(); i++)  
    cout << a.at(i) << " ";   // get
```

0	10	20	30	40	50	60	70	80	90
---	----	----	----	----	----	----	----	----	----

Containers: Πίνακας

- `for()` με `iterator(*)`

```
array<int, 10> a;
```

```
for (int i=0; i < a.size(); i++)  
    a.at(i) = i * 10;
```

```
// for (int i=0; i < a.size(); i++)  
//     cout << a.at(i) << " ";
```

```
for (int d : a)  
    cout << d << " ";
```

```
for (type var : container)  
    ... do something
```

0 10 20 30 40 50 60 70 80 90

Containers: Πίνακας

- `for ()` με `iterator(*)`

```
array<int, 10> a;
```

```
// for (int i=0; i < a.size(); i++)  
//     a.at(i) = 42;
```

```
for (int &d : a)  
    d = 42;
```

```
for (int d : a)  
    cout << d << " ";
```

```
for (type var : container)  
    ... do something
```

0	10	20	30	40	50	60	70	80	90
---	----	----	----	----	----	----	----	----	----

Containers: Πίνακας

- 2+ διαστάσεις

```
array<int, 10> a;
```

```
array<array<int, 10>, 10> b;
```

```
for (int i=0; i<10; i++)  
    for (int j=0; j<10; j++)  
        b[i][j] = i + j;  
    // b.at(i).at(j) = i + j;
```

```
for (array<int,10> row : b) {  
    for (int d : row)  
        cout << d << "\t";  
    cout << endl;  
}
```

```
for (const array<int,10> &row : b) ...
```

0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10
2	3	4	5	6	7	8	9	10	11
3	4	5	6	7	8	9	10	11	12
4	5	6	7	8	9	10	11	12	13
5	6	7	8	9	10	11	12	13	14
6	7	8	9	10	11	12	13	14	15
7	8	9	10	11	12	13	14	15	16
8	9	10	11	12	13	14	15	16	17
9	10	11	12	13	14	15	16	17	18