



ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΑΝΟΙΚΤΑ ΑΚΑΔΗΜΑΪΚΑ ΜΑΘΗΜΑΤΑ

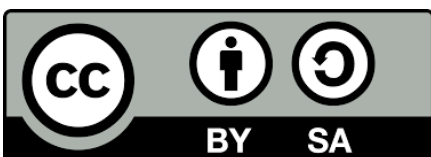


Τίτλος Μαθήματος: Εισαγωγή στον Προγραμματισμό

Ενότητα: Εισαγωγικά μαθήματος και Δυαδική αναπαράσταση

Διδάσκων: Λέκτορας Χάρης Παπαδόπουλος

Τμήμα: Μαθηματικών



Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

343 – Εισαγωγή στον Προγραμματισμό

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Ακαδημαϊκό Έτος 2013-2014

Χάρης Παπαδόπουλος
207δ, Β' όροφος
e-mail: charis@cs.uoi.gr

Ωρες Γραφείου:
Δευτέρα 11-13 & Παρασκευή 11-13

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2013				
Δ	Τ	Τ	Π	Π
14	15	16	17 Θ	18
21	22	23	24 Θ	25
28	29	30	31 Θ	

Νοέμβριος 2013				
Δ	Τ	Τ	Π	Π
4 Ε	5 Ε	6	7 Θ	8
11 Ε	12 Ε	13	14 Θ	15
18 Ε	19 Ε	20	21 Θ	22
25 Q	26	27	28 Θ	29

Δεκέμβριος 2013				
Δ	Τ	Τ	Π	Π
2 Ε	3 Ε	4	5 Θ	6
9 Ε	10 Ε	11	12 Θ	13
16 Q	17	18	19 Θ	20

Ιανουάριος 2014				
Δ	Τ	Τ	Π	Π
6	7	8	9 Θ	10
13	14	15	16 Θ	17

Εβδομάδα	Θέματα	Υλη βιβλιογραφίας
Πέ, 17 Οκτωβρίου	Εισαγωγικά μαθήματος & Δυναμική αναπαράσταση	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πέ, 24 Οκτωβρίου	Είσοδος/Εξοδος δεδομένων, τύποι δεδομένων & μεταβλητών	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Πέ, 31 Οκτωβρίου	Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές, Ροή ελέγχου: if/else	[1]: 2.1, 2.2 Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Δε Τρ, 4-5 Νοε	1 ^ο Εργαστήριο	
Πέ, 7 Νοεμβρίου	Ροή ελέγχου for, while, do-while	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε Τρ, 11-12 Νοε	2 ^ο Εργαστήριο	
Πέ, 14 Νοεμβρίου	Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε Τρ, 18-19 Νοε	3 ^ο Εργαστήριο	
Πέ, 21 Νοεμβρίου	Επανάληψη με Παραδείγματα	
Δε Τρ, 25-26 Νοε	1 ^ο Quiz	
Πέ, 28 Νοεμβρίου	Πίνακες (μονοδιάστατοι και πολυδιάστατοι)	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε Τρ, 2-3 Δεκ	4 ^ο Εργαστήριο	
Πέ, 5 Δεκεμβρίου	Αλφαριθμητικά και Συμβολοσειρές	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε Τρ, 9-10 Δεκ	5 ^ο Εργαστήριο	
Πέ, 12 Δεκεμβρίου	Δομές και χρήση αρχείων	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε Τρ, 16-17 Δεκ	2 ^ο Quiz	
Πέ, 19 Δεκεμβρίου	Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πέ, 9 Ιανουαρίου	Επανάληψη σε ταξινομήσεις και αναζητήσεις	
Πέ, 16 Ιανουαρίου	Επανάληψη	

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2013				
Δ	Τ	Τ	Π	Π
14	15	16	17 Θ	18
21	22	23	24 Θ	25
28	29	30	31 Θ	

Νοέμβριος 2013				
Δ	Τ	Τ	Π	Π
4 Ε	5 Ε	6	7 Θ	8
11 Ε	12 Ε	13	14 Θ	15
18 Ε	19 Ε	20	21 Θ	22
25 Q	26	27	28 Θ	29

Δεκέμβριος 2013				
Δ	Τ	Τ	Π	Π
2 Ε	3 Ε	4	5 Θ	6
9 Ε	10 Ε	11	12 Θ	13
16 Q	17	18	19 Θ	20

Ιανουάριος 2014				
Δ	Τ	Τ	Π	Π
6	7	8	9 Θ	10
13	14	15	16 Θ	17

Εβδομάδα	Θέματα	Υλη βιβλιογραφίας
Πέ, 17 Οκτωβρίου	Εισαγωγικά μαθήματος & Δυναμική αναπαράσταση	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πέ, 24 Οκτωβρίου	Είσοδος/Εξοδος δεδομένων, τύποι δεδομένων & μεταβλητών	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Πέ, 31 Οκτωβρίου	Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές, Ροή ελέγχου: if/else	[1]: 2.1, 2.2 Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Δε Τρ, 4-5 Νοε	1 ^ο Εργαστήριο	
Πέ, 7 Νοεμβρίου	Ροή ελέγχου for, while, do-while	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε Τρ, 11-12 Νοε	2 ^ο Εργαστήριο	
Πέ, 14 Νοεμβρίου	Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε Τρ, 18-19 Νοε	3 ^ο Εργαστήριο	
Πέ, 21 Νοεμβρίου	Επανάληψη με Παραδείγματα	
Δε Τρ, 25-26 Νοε	1 ^ο Quiz	
Πέ, 28 Νοεμβρίου	Πίνακες (μονοδιάστατοι και πολυδιάστατοι)	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε Τρ, 2-3 Δεκ	4 ^ο Εργαστήριο	
Πέ, 5 Δεκεμβρίου	Αλφαριθμητικά και Συμβολοσειρές	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε Τρ, 9-10 Δεκ	5 ^ο Εργαστήριο	
Πέ, 12 Δεκεμβρίου	Δομές και χρήση αρχείων	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε Τρ, 16-17 Δεκ	2 ^ο Quiz	
Πέ, 19 Δεκεμβρίου	Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πέ, 9 Ιανουαρίου	Επανάληψη σε ταξινομήσεις και αναζητήσεις	
Πέ, 16 Ιανουαρίου	Επανάληψη	

Ενότητα 14

ΠΙΝΑΚΕΣ

Πίνακες

- Ορισμός:
 - Συλλογή δεδομένων ίδιου τύπου
- Η πρώτη σύνθετη δομή τύπων:
 - σύνθετη: "ομαδοποίηση"
 - int, float, double, char, bool: απλές δομές τύπων
- Ομαδοποιεί κοινές μεταβλητές σε μια "λίστα"
 - Βαθμολογίες, θερμοκρασίες, ονόματα, κ.τ.λ.
 - Αποφεύγουμε να δηλώσουμε πολλές απλές μεταβλητές
 - Μπορεί να χειριστεί την "λίστα" ως μια οντότητα

Δήλωση Πινάκων

- Δήλωση πίνακα → ανοίγει κατάλληλες θέσεις στην μνήμη
 - 5 μεταβλητές τύπου int:
`int score[5];`
 - οι 5 μεταβλητές θα είναι:
`score[0], score[1], score[2], score[3], score[4]`
- Συστατικά ενός πίνακα:
 - όνομα
 - στοιχεία
 - δείκτης
 - δηλωμένο μέγεθος
 - βασικός τύπος

Επισημάνσεις

- Χρήση `for` για διαχείριση ενός πίνακα

```
for(i = 0; i < 5; i++)  
{  
    cout << score[i] << " " << (max - score[i]);  
}
```

- Οι δείκτες των πινάκων αρχίζουν πάντοτε από το 0
 - αρχίζουν πάντοτε από το 0
 - ...
 - τελειώνουν με τον ακέραιο που είναι μικρότερος κατά 1 από το μέγεθος του πίνακα

Σύνταξη δήλωσης πίνακα

Σύνταξη

```
όνομα_τύπου  όνομα_πίνακα[ Δηλωμένο_μέγεθος ] ;
```

Παράδειγμα

```
int  bigArray[100];  
double  a[3];  
double  b[5];  
char  grade[10], oneGrade;
```

- Ορίζει **Δηλωμένο_μέγεθος** στοιχεία:
όνομα_πίνακα[0], ..., όνομα_πίνακα[**Δηλωμένο_μέγεθος** – 1]
- Κάθε στοιχείο είναι μια μεταβλητή τύπου **όνομα_τύπου**
a[0] a[1] a[2] όλα τύπου double
b[0] b[1] b[2] b[3] b[4] όλα τύπου double

Προσοχή στο μέγεθος του πίνακα

- **Δεν επιτρέπεται** δήλωση πίνακα μεταβλητού μεγέθους

```
cout << "Δώσε αριθμό \n";  
cin >> number;  
int score[number];
```



- Χρήση καθορισμένης σταθεράς για το μέγεθος του πίνακα:
 - αντί να χρησιμοποιούμε έναν αριθμό (π.χ., 5) για το μέγεθος του πίνακα μπορούμε να κάνουμε χρήση προκαθορισμένης σταθεράς

```
const int NUM_OF_STUDENTS = 5;
```

– Τότε η δήλωση του πίνακα: `int score[NUM_OF_STUDENTS];`

– Και αντίστοιχα:

```
for(i = 0; i < NUM_OF_STUDENTS; i++)  
{  
    cout << score[i] << " " << (max - score[i]);  
}
```

Παράδειγμα με πίνακα

```
#include <iostream>
int main( )
{
    int i, score[5], max;

    cout << "Enter 5 scores:\n";
    cin >> score[0];
    max = score[0];
    for (i = 1; i < 5; i++)
    {
        cin >> score[i];
        if (score[i] > max)
            max = score[i];
    }

    cout << "Max is " << max << endl
         << "The scores and differences from max are:\n";
    for (i = 0; i < 5; i++)
        cout << score[i] << " " << (max - score[i]) << endl;
}
```

Παράδειγμα

Enter 5 scores::

5 9 2 10 6

Max is 10

The scores and differences from max are:

5 5

9 1

2 8

10 0

6 4

Παράδειγμα με προκαθορισμένο μέγεθος

```
#include <iostream>
int main( )
{
    const int NUM_OF_STUDENTS = 5;
    int i, score[NUM_OF_STUDENTS], max;
    cout << "Enter " << NUM_OF_STUDENTS << " scores:\n";
    cin >> score[0];
    max = score[0];
    for (i = 1; i < NUM_OF_STUDENTS; i++)
    {
        cin >> score[i];
        if (score[i] > max)
            max = score[i];
    }

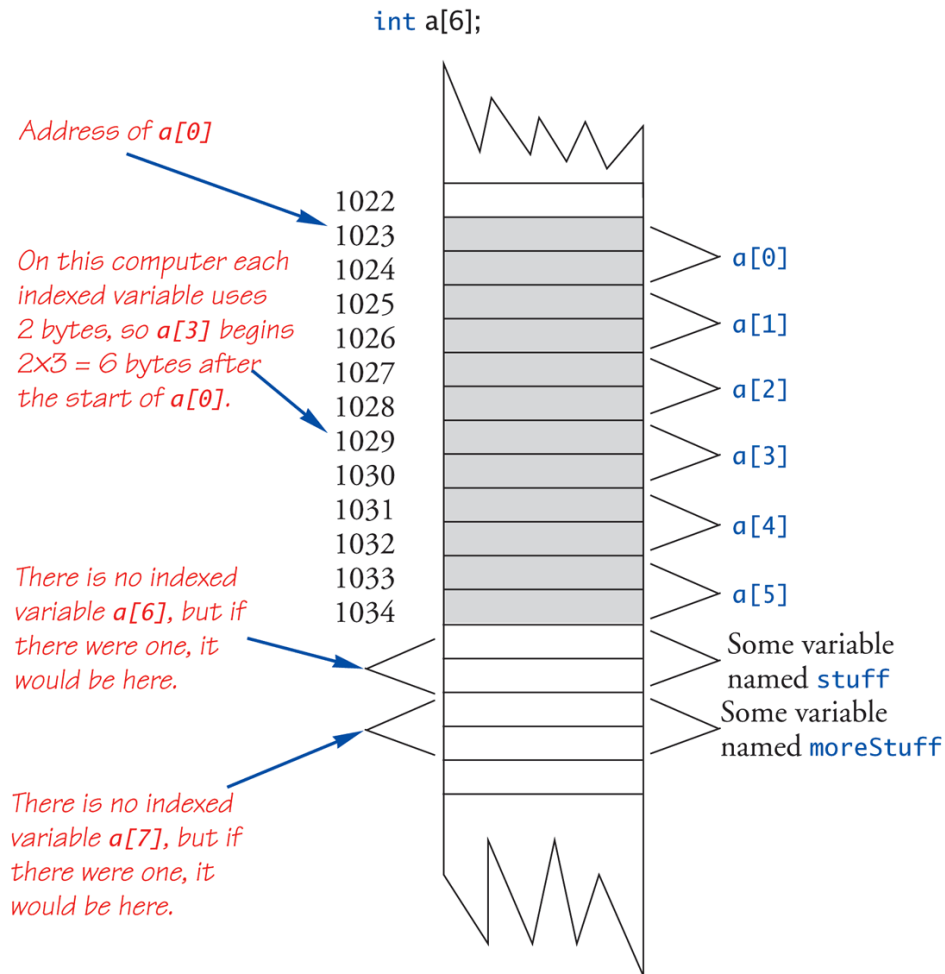
    cout << "Max is " << max << endl
         << "The scores and differences from max are:\n";
    for (i = 0; i < NUM_OF_STUDENTS; i++)
        cout << score[i] << " " << (max - score[i]) << endl;
}
```

Οι πίνακες στην μνήμη

- Μια απλή μεταβλητή περιγράφεται από
 - διεύθυνση μνήμης (αρχή του πρώτου byte)
 - τύπος μεταβλητής (πόσα byte καταλαμβάνει)
- Πίνακες:
 - όλα τα στοιχεία τοποθετούνται το ένα δίπλα στο άλλο
- `int a[6];`
 - Δεσμεύει 6 θέσεις για μεταβλητές τύπου `int`
 - Θυμάται την διεύθυνση του στοιχείου `a[0]` και κανενός άλλου

Οι πίνακες στην μνήμη

Display 5.2 An Array in Memory



Δείκτης πίνακα εκτός ορίων

- Η C++ σας επιτρέπει να αναφερθείτε σε στοιχεία εκτός ορίου
 - τα αποτελέσματα είναι ασαφή
 - ο μεταφραστής δεν αναγνωρίζει τέτοια λάθη
- Στο χέρι σας να αποφύγετε τέτοια λάθη
- Οι δείκτες κυμαίνονται από 0 έως δηλωμένο_μέγεθος – 1

```
double temperature[24];           // 24 : array size
temperature[24] = 5;
```

- Ο δείκτης 24 είναι "εκτός ορίων"!
- Δεν βγαίνει κάποιο μήνυμα: καταστροφικά αποτελέσματα
 - Π.χ., στη διεύθυνση `temperature[24]` υπάρχει άλλη μεταβλητή

Απόδοση αρχικών τιμών

- Αρχικές τιμές κατά την δήλωση ενός πίνακα με { }:

```
int children[3] = {2, 12, 1};
```

- Ισοδύναμο με:

```
int children[3];  
int children[0] = 2;  
int children[1] = 12;  
int children[2] = 1;
```

- Αν παραληφθεί κάποια τιμή μέσα σε { } :

```
int children[3] = {2, 12};
```

Θεωρεί ότι `children[2] = 0`

Γεμίζει τις τελευταίες τιμές με μηδενικές τιμές του βασικού τύπου

- Αν παραληφθεί το μέγεθος του πίνακα :

```
int b[ ] = {5, 12, 11};
```

Βρίσκει το ελάχιστο μέγεθος για τον πίνακα (`int b[3]`)

Παραδείγματα - Ερωτήσεις

```
char symbol[3] = {'a', 'b', 'c'};
for(int index = 0; index < 3; index++)
    cout << symbol[index];
```

```
double a[3] = {1.1, 2.2, 3.3};
cout << a[0] << " " << a[1] << " " << a[2] << endl;
a[1] = a[2];
cout << a[0] << " " << a[1] << " " << a[2] << endl;
```

```
int array[10];
for(int index = 1; index <= 10; index++)
    cout << array[index];
```

```
int b[10];

// ?? πώς θα διαβάσαμε;

for(int i = 0; i < 10; i++)
    cout << b[i] << " " << endl;
```

```
int i, a[10];
for(i = 0; i < 10; i++)
    a[i] = 2*i;
for(i = 0; i < 10; i++)
    cout << a[i] << " ";
cout << endl;
for(i=0;i<10;i = i + 2)
    cout << a[i] << " ";
cout << endl;
```

Ενότητα 15

ΠΙΝΑΚΕΣ ΜΕΣΑ ΣΕ ΣΥΝΑΡΤΗΣΕΙΣ

Πίνακες σε συναρτήσεις

- Ως παράμετροι σε συναρτήσεις:
 - Μεμονωμένα στοιχεία του πίνακα
 - Ολόκληρο τον πίνακα
- Ως επιστρεφόμενες τιμές από συναρτήσεις:
 - Δεν θα το δούμε (Κεφάλαιο 10)

Στοιχεία πινάκων ως παράμετροι

- Χρησιμοποιούνται όπως οι απλές μεταβλητές
- Έστω η ακόλουθη συνάρτηση:

```
void myFunction( double par1 );
```

- Και οι ακόλουθες δηλώσεις:

```
int i;  
double n, a[10];
```

- Τότε μπορούμε να καλέσουμε την συν/ση με διάφορους τρόπους:

```
myFunction(i);    // i μετατρέπεται σε double  
myFunction(a[3]); // a[3] είναι double  
myFunction(n);   // n είναι double
```

Αποτίμηση των παραμέτρων

- Έστω:

```
myFunction( a[i] );
```

- Η τιμή της i καθορίζεται πρώτα
 - Καθορίζει ποιο στοιχείο του πίνακα να στείλει ως παράμετρο

```
myFunction( a[i*5] );
```

- Απολύτως συμβατό ως προς τον μεταφραστή
- Ο προγραμματιστής είναι υπεύθυνος για να κρατήσει τον δείκτη μέσα στα όρια του πίνακα

Παράδειγμα

- Έστω η ακόλουθη συνάρτηση:

```
void tripler( int& n )  
{  
    n = 3 * n;  
}
```

- Τι εκτυπώνουν τα ακόλουθα;

```
int a[3] = {4, 5, 6}, number = 2;  
  
tripler( a[2] );  
tripler( a[3] );  
tripler( a[number] );  
tripler( a );  
tripler( number );
```

Ολόκληροι πίνακες ως παράμετροι

- Αν θέλουμε να περάσουμε ως παράμετρο ολόκληρο τον πίνακα:
 - Χρησιμοποιούμε το όνομα του πίνακα
 - Παράμετρος πίνακα
- Στέλνουμε και το μέγεθος του πίνακα ως παράμετρο:
 - ως μια τυπική παράμετρο τύπου int

```
void fillUp(int a[], int size)
{
    cout << "Enter " << size << " numbers:\n";
    for (int i = 0; i < size; i++)
        cin >> a[i];
    cout << "The last array index:" << (size - 1)
        << endl;
}
```

```
int a[5], b[10];

fillUp(a, 5);
fillUp(b, 10);
```

- Παράμετρος πίνακα: σαν **παράμετρος με αναφορά**

Ολόκληροι πίνακες ως παράμετροι

- Στο κάλεσμα στέλνουμε ολόκληρο τον πίνακα μόνο με το όνομά του (χωρίς [])
- Πίνακας 3 συστατικά:
 - Διεύθυνση της πρώτης μεταβλητής (a[0])
 - βασικός τύπος
 - μέγεθος του πίνακα

```
void fillUp(int a[], int size)
{
    cout << "Enter " << size << " numbers:\n";
    for (int i = 0; i < size; i++)
        cin >> a[i];
    cout << "The last array index:" << (size - 1)
        << endl;
}
```

```
int a[5], b[10];

fillUp(a, 5);
fillUp(b, 10);
```

Ολόκληροι πίνακες ως παράμετροι

- Στο κάλεσμα στέλνουμε ολόκληρο τον πίνακα μόνο με το όνομά του (χωρίς [])
- Πίνακας 3 συστατικά:
 - Διεύθυνση της πρώτης μεταβλητής (a[0])
 - βασικός τύπος
 - μέγεθος του πίνακα

Όταν καλούμε
στέλνουμε μόνο
τη διεύθυνση a[0]

```
void fillUp(int a[], int size)
{
    cout << "Enter " << size << " numbers:\n";
    for (int i = 0; i < size; i++)
        cin >> a[i];
    cout << "The last array index:" << (size - 1)
        << endl;
}
```

```
int a[5], b[10];

fillUp(a, 5);
fillUp(b, 10);
```

- Παράμετρος πίνακα: σαν **παράμετρος με αναφορά**

Παράμετροι Πινάκων

- Φαίνονται λίγο περίεργοι:
 - Δεν έχουν [] στη παράμετρο
 - Πρέπει να στέλνουμε και το μέγεθος
- Μια καλή ιδιότητα:
 - Μπορούμε να χρησιμοποιούμε την ΙΔΙΑ συνάρτηση για να γεμίσουμε πίνακες διαφορετικού μεγέθους!
 - Επαναχρησιμοποίηση συναρτήσεων

```
int a[5], b[10];  
  
fillUp(a, 5);  
fillUp(b, 10);
```

- Πώς φαντάζεστε την ακόλουθη συνάρτηση;

```
void sumArray(double& sum, double a[], int size)
```

Ο προσδιορισμός παραμέτρων const

- Αν και η παράμετρος πίνακα φαίνεται καλή ιδιότητα διότι επιτρέπει την αλλαγή τιμών στα στοιχεία του πίνακα:
 - Μερικές φορές δεν είναι επιθυμητό
- Προστατεύουμε τα περιεχόμενα του πίνακα από πιθανή τροποποίηση
 - χρήση const πριν από την παράμετρο πίνακα
 - λέει στον μεταφραστή ότι δεν πρέπει να αλλάξουν τα στοιχεία

```
void showtheWorld(const int a[], int size)
{
    cout << "Ο πίνακας έχει τις ακόλουθες τιμές:\n";
    for (int i = 0; i < size; i++)
        cout << a[i] << " ";
    cout << endl;
}
```

Ο προσδιορισμός παραμέτρων const

- Αν και η παράμετρος πίνακα φαίνεται καλή ιδιότητα διότι επιτρέπει την αλλαγή τιμών στα στοιχεία του πίνακα:
 - Μερικές φορές δεν είναι επιθυμητό
- Προστατεύουμε τα περιεχόμενα του πίνακα από πιθανή τροποποίηση
 - χρήση const πριν από την παράμετρο πίνακα
 - λέει στον μεταφραστή ότι δεν πρέπει να αλλάξουν τα στοιχεία

```
void showtheWorld(const int a[], int size)
{
    cout << "Ο πίνακας έχει τις ακόλουθες τιμές:\n";
    for (int i = 0; i < size; a[i]++)
        cout << a[i] << " ";
    cout << endl;
}
```

Δεν το
επιτρέπει η
χρήση const

Μερικώς συμπληρωμένοι πίνακες

- Είναι σχετικά δύσκολο να γνωρίζουμε το ακριβές μέγεθος του πίνακα που χρειαζόμαστε
- Στη δήλωση πρέπει να το δηλώσουμε όσο γίνεται μεγαλύτερο (χωρίς υπερβολές!!!)
 - Τότε πρέπει να σημειώνουμε τα κανονικά δεδομένα του πίνακα
 - Χρήση επιπλέον μεταβλητής που παρακολουθεί το πλήθος των αναγκαίων στοιχείων
 - `int numberUsed;`
 - Παρακολουθεί το πλήθος στοιχείων ενός πίνακα
 - Την μεταβλητή αυτή πρέπει να την δηλώνουμε σαν ξεχωριστή μεταβλητή στα ορίσματα των συναρτήσεων

```
void fillArray(int a[], int size, int& numberUsed)
{
    cout << "Δώστε μέχρι " << size << " μη αρνητικούς.\n"
         << "σημειώστε το τέλος με έναν αρνητικό αριθμό.\n";

    int next, index = 0;
    cin >> next;
    while ((next >= 0) && (index < size))
    {
        a[index] = next;
        index++;
        cin >> next;
    }

    numberUsed = index;
}
```

```
const int MAX_NUMBER_SCORES = 10;

int score[MAX_NUMBER_SCORES], numberUsed;

cout << "Δώστε βαθμολογίες:\n";
fillArray(score, MAX_NUMBER_SCORES, numberUsed);
```

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

const int MAX_NUMBER_SCORES = 10;

void fillArray(int a[], int size, int& numberUsed);

double computeAverage(const int a[], int numberUsed);

void showDifference(const int a[], int numberUsed);

int main( )
{
    int score[MAX_NUMBER_SCORES], numberUsed;

    cout << " Δώστε βαθμολογίες:\n";
    fillArray(score, MAX_NUMBER_SCORES, numberUsed);
    showDifference(score, numberUsed);

    return 0;
}
```



```

double computeAverage(const int a[], int numberUsed)
{
    double total = 0.0;
    for (int index = 0; index < numberUsed; index++)
        total = total + a[index];
    if (numberUsed > 0)
    {
        return (total/numberUsed);
    }
    else
    {
        cout << "ERROR.\n" << "computeAverage returns 0.\n";
        return 0;
    }
}

void showDifference(const int a[], int numberUsed)
{
    double average = computeAverage(a, numberUsed);
    cout << "Μέσος όρος είναι" << average << endl
        << "Με διαφορές:\n";
    for (int index = 0; index < numberUsed; index++)
        cout << a[index] << " " << (a[index] - average) << endl;
}

```

Ενότητα 16

ΠΟΛΥΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ

Πολυδιάστατοι πίνακες

- Πίνακες με παραπάνω από ένα δείκτη

- `char page[30][100];`

- Δύο δείκτες: "πίνακας από πίνακες"
 - Οπτικοποιούμε ως:

```
page[0][0], page[0][1], ..., page[0][99]
page[1][0], page[1][1], ..., page[1][99]
...
page[29][0], page[29][1], ..., page[29][99]
```

- Η C++ επιτρέπει οποιοδήποτε αριθμό από δείκτες
 - Δεν θα ασχοληθούμε με παραπάνω από δύο

Παράδειγμα

- Ποια είναι η έξοδος του ακόλουθου παραδείγματος;

```
int myArray[4][4], index1, index2;

for(index1 = 0; index1 < 4; index1++)
    for(index2 = 0; index2 < 4; index2++)
        myArray[index1][index2] = index2;

for(index1 = 0; index1 < 4; index1++)
{
    for(index2 = 0; index2 < 4; index2++)
        cout << myArray[index1][index2] << " ";
    cout << endl;
}
```

Παράδειγμα

- Ποια είναι η έξοδος του ακόλουθου παραδείγματος;

```
int myArray[4][4], index1, index2;

for(index1 = 0; index1 < 4; index1++)
    for(index2 = 0; index2 < 4; index2++)
        myArray[index1][index2] = index2;

for(index1 = 0; index1 < 4; index1++)
{
    for(index2 = 0; index2 < 4; index2++)
        cout << myArray[index1][index2] << " ";
    cout << endl;
}
```

Παράδειγμα

```
0 1 2 3
0 1 2 3
0 1 2 3
0 1 2 3
```

Πολυδιάστατοι πίνακες ως παράμετροι

- Παρόμοια με μιας διάστασης πίνακα
- Με σημαντικές διαφορές:
 - 1^η διάσταση δεν δίνεται []
 - Πρέπει να δώσουμε μια επιπλέον παράμετρο για το μέγεθος
 - 2^η διάσταση ΔΙΝΕΤΑΙ
- Παράδειγμα:

```
void DisplayPage(const char p[][100], int sizeDimension1)
{
    for (int index1=0; index1<sizeDimension1; index1++)
    {
        for (int index2=0; index2 < 100; index2++)
            cout << p[index1][index2];
        cout << endl;
    }
}
```

```
char array[50][100];
```

```
DisplayPage(array, 50);
```

Ενότητες 14-16

ΟΛΟΚΛΗΡΩΜΕΝΑ ΠΑΡΑΔΕΙΓΜΑΤΑ

Μη-αύξουσα τάξη

- Γράψτε μια συνάρτηση η οποία θα παίρνει ως παραμέτρους έναν πίνακα τύπου `double` και μια παράμετρο `int` και θα επιστρέφει μια τιμή `int`.
- Θα ελέγχει αν ο πίνακας είναι ταξινομημένος κατά αύξουσα τάξη:

$$a[0] \leq a[1] \leq a[2] \leq \dots$$

- Επιστρέφει `-1` αν ο πίνακας είναι ταξινομημένος διαφορετικά
- επιστρέφει τον δείκτη του πρώτου στοιχείου (από αριστερά προς τα δεξιά) που είναι εκτός σειράς.
- Π.χ., για `double a[6] = { 1.2, 2.1, 3.3, 2.5, 1.4, 0.0 };`
επιστρέφει `3`


```
int outoforder(double a[ ], int size)
{
    for (int i = 0; i < size; i++)
    {
        if( a[i] > a[i+1] )
            return (i+1);
    }
    return -1;
}
```

```
int outoforder(double a[ ], int size)
{
    for (int i = 0; i < size; i++)
    {
        if( a[i] > a[i+1] )
            return (i+1);
    }
    return -1;
}
```

```
int outoforder(double a[ ], int size)
{
    for (int i = 0; i < size - 1; i++)
    {
        if( a[i] > a[i+1] )
            return (i+1);
    }
    return -1;
}
```

Εισαγωγή χαρακτήρων

- Γράψτε ένα πρόγραμμα που θα εισάγει μέχρι 10 χαρακτήρες μέσα σε έναν πίνακα και θα τα εμφανίζει με αντίστροφη σειρά.
- Για παράδειγμα αν η είσοδος είναι:
abcd.
τότε η έξοδος θα είναι:
dcba
- Χρησιμοποιείστε μια τελεία . για να σηματοδοτήσετε το τέλος εισόδου.

```
#include <iostream>
const int FULL_SIZE = 10;

int main( )
{
    cout << "Enter 10 letters:";

    char letters[FULL_SIZE], next;
    int i = 0, num;

    cin >> next;
    while ( (next != '.') && (i < FULL_SIZE) )
    {
        letters[i] = next;
        i++;
        cin >> next;
    }

    num = i;

    cout << "Reversed:\n";
    for( i = num - 1; i >= 0; i--)
        cout << letters[i];

    return 0;
}
```

Αναζήτηση στοιχείου

- Γράψτε μια συνάρτηση που θα παίρνει ως παράμετρο έναν πίνακα `int`, έναν ακέραιο αριθμό `target` και ψάχνει αν ο `target` υπάρχει στον πίνακα.
- Θα επιστρέφει το δείκτη του πρώτου στοιχείου που υπάρχει αλλιώς
- Θα επιστρέφει `-1` (αν δεν υπάρχει)

```
int search(int a[ ], int numberUsed, int target)
{
    int index = 0;
    bool found = false;
    while ((!found) && (index < numberUsed))
    {
        if (target == a[index])
            found = true;
        else
            index++;
    }

    if (found)
        return index;
    else
        return -1;
    return -1;
}
```

```
int search(int a[ ], int numberUsed, int target)
{
    int index = 0;
    bool found = false;
    while ((!found) && (index < numberUsed))
    {
        if (target == a[index])
            found = true;
        else
            index++;
    }

    if (found)
        return index;
    else
        return -1;
    return -1;
}
```

Αν μας έδιναν τον ορισμό της συνάρτησης:

```
bool search(int a[ ], .....
```

```
bool search(int a[ ], int numberUsed, int target, int& where)
{
    int index = 0;
    bool found = false;
    while ((!found) && (index < numberUsed))
    {
        if (target == a[index])
            found = true;
        else
            index++;
    }

    if (found)
        where = index;
    return found;
}
```

Αν μας έδιναν τον ορισμό της συνάρτησης:

```
bool search(int a[ ], .....)
```


Ιστόγραμμα

- Υπάρχουν 4 γραμμές παραγωγής σε ένα εργοστάσιο.
 - Θέλουμε να διαβάζουμε τιμές που αντιστοιχούν στις ποσότητες προϊόντων από κάθε γραμμή παραγωγής (άγνωστο σε πλήθος) και να εκτυπώνουμε μια "ράβδος από *" για κάθε γραμμή παραγωγής.
- Η ράβδος να έχει κλιμάκωση στα 1000 προϊόντα
 - π.χ., κάθε * αντιστοιχεί σε 1000, με στρογγυλοποίηση
 - για 1600 προϊόντα: * *

Παράδειγμα

Δώσε δεδομένα για το εργοστάσιο 1 (αρνητικό για τέλος):

2000 3000 1000 -1

Σύνολο = 6000

Δώσε δεδομένα για το εργοστάσιο 2 (αρνητικό για τέλος):

2050 3002 1300 -1

Σύνολο = 6352

Δώσε δεδομένα για το εργοστάσιο 3 (αρνητικό για τέλος):

5000 4020 500 4338 -1

Σύνολο = 13868

Δώσε δεδομένα για το εργοστάσιο 4 (αρνητικό για τέλος):

2507 6050 1809 -1

Σύνολο = 10306

...συνέχεια

Εργοστάσιο 1: *****

Εργοστάσιο 2: *****

Εργοστάσιο 3: *****

Εργοστάσιο 4: *****

```
#include <iostream>

const int NUMBER_OF_PLANTS = 4;

void inputData(int a[], int lastPlantNumber);

void scale(int a[], int size);

void graph(const int asteriskCount[], int lastPlantNumber);

void getTotal(int& sum);

int round(double number);

void printAsterisks(int n);

int main( )
{
    int production[NUMBER_OF_PLANTS];

    inputData(production, NUMBER_OF_PLANTS);
    scale(production, NUMBER_OF_PLANTS);
    graph(production, NUMBER_OF_PLANTS);
    return 0;
}
```

```

void inputData(int a[], int lastPlantNumber)
{
    for (int p = 1; p <= lastPlantNumber; p++)
    {
        cout << "Δώσε δεδομένα για το εργοστάσιο " << p ;
        getTotal(a[p - 1]);
    }
}

```

```

void getTotal(int& sum)
{
    cout << "(αρνητικό για τέλος) : \n";
    sum = 0;
    int next;
    cin >> next;
    while (next >= 0)
    {
        sum = sum + next;
        cin >> next;
    }

    cout << "Σύνολο = " << sum << endl;
}

```

```

void scale(int a[], int size)
{
    for (int index = 0; index < size; index++)
        a[index] = round(a[index]/1000.0);
}

int round(double number)
{
    return static_cast<int>(floor(number + 0.5));
}

void graph(const int asteriskCount[], int lastPlantNumber)
{
    for (int p = 1; p <= lastPlantNumber; p++)
    {
        cout << "Εργαστήριο " << p << ": ";
        printAsterisks(asteriskCount[p - 1]);
        cout << endl;
    }
}

void printAsterisks(int n)
{
    for (int count = 1; count <= n; count++)
        cout << "*";
}

```

Εκλογές

- 10 φοιτητές είναι υποψήφιοι και 40 ψήφοι έχουν καταχωρηθεί σε έναν πίνακα “responses”.
- Γράψτε μια συνάρτηση που δέχεται τον πίνακα “responses” και εκτυπώνει τα αποτελέσματα των εκλογών.
 - Δηλαδή πόσους ψήφους πήρε ο κάθε ένας από τους 10 υποψηφίους.

- Παράδειγμα:

```
int responses[] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,  
                  1, 6, 3, 8, 6, 10, 3, 8, 2, 7,  
                  6, 5, 7, 6, 8, 6, 7, 5, 6, 6,  
                  5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
```

Παράδειγμα

Υπ.	Ψήφοι
1	2
2	2
3	2
4	2
5	5
6	11
7	5
8	7
9	1
10	3

```

#include <iostream>

void elections( int responses[], int size );

int main( )
{
    int responses[] = {1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6, 10, 3, 8, 2, 7,
                       6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
    elections( responses, 40 );
    return 0;
}

void elections( int responses[], int size )
{
    int psifoi[11];

    for ( int i = 0; i < 11; i++ )
        psifoi[ i ] = 0 ;

    for ( int k = 0; k < size; k++ )
        psifoi[ responses[ k ] ] = psifoi[ responses[ k ] ] + 1;

    cout << "Υποψ. \t Ψήφοι\n");

    for ( int i = 0; i < 11; i++ )
        cout << i << "\t" << psifoi[ i ] << endl;
}

```

Διάβασμα και εκτύπωση 4x5 πίνακα

- Γράψτε ένα πρόγραμμα το οποίο θα συμπληρώνει έναν πίνακα `a` με αριθμούς που εισάγονται από το πληκτρολόγιο. Οι αριθμοί θα εισάγονται 5 ανά γραμμή σε 4 γραμμές.

- Ο πίνακας θα είναι δηλωμένος:

```
int a[4][5];
```

- Στη συνέχεια γράψτε συνάρτηση τύπου `void` με το όνομα `echo` που εκτυπώνει τον πίνακα που δημιουργήσατε.

- Θα την καλείτε ως εξής:

```
echo(a, 4);
```

```
void echo(int a[][5], int lines);

int main( )
{
    int a[4][5];
    int i,j;

    for(i = 0; i < 4; i++)
        for(j = 0; j < 5; j++)
            cin >> a[i][j];

    echo(a, 4);
}

void echo(int a[][5], int lines)
{
    int i,j;
    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 5; j++)
            cout << a[i][j] << " ";
        cout << endl;
    }
}
```


Παράδειγμα μέσου όρου

- Δυδιάστατος πίνακας (grades) με βαθμολογίες φοιτητών
 - Μία γραμμή ανά φοιτητή
 - Στήλες – βαθμολογία σε test
- Εκτύπωση πίνακα, μέσου όρου / φοιτητή, μέσου όρου / test

	test1	test2	test3
φοιτητής1	10,	10,	10
φοιτητής2	2,	0,	1
φοιτητής3	8,	6,	9
φοιτητής3	8,	4,	10

Παράδειγμα μέσου όρου

- Δυδιάστατος πίνακας (grades) με βαθμολογίες φοιτητών
 - Μία γραμμή ανά φοιτητή
 - Στήλες – βαθμολογία σε test
- Εκτύπωση πίνακα, μέσου όρου / φοιτητή, μέσου όρου / test

	test1	test2	test3
φοιτητής1	grades[0][0]	grades[0][1]	grades[0][2]
φοιτητής2	grades[1][0]	grades[1][1]	grades[1][2]
φοιτητής3	grades[2][0]	grades[2][1]	grades[2][2]
φοιτητής3	grades[3][0]	grades[3][1]	grades[3][2]

Παράδειγμα μέσου όρου

- Δυδιάστατος πίνακας (grades) με βαθμολογίες φοιτητών
 - Μία γραμμή ανά φοιτητή
 - Στήλες – βαθμολογία σε test
- Εκτύπωση πίνακα, μέσου όρου / φοιτητή, μέσου όρου / test

	test1	test2	test3
φοιτητής1	grades[0][0]	grades[0][1]	grades[0][2]
φοιτητής2	grades[1][0]	grades[1][1]	grades[1][2]
φοιτητής3	grades[2][0]	grades[2][1]	grades[2][2]
φοιτητής3	grades[3][0]	grades[3][1]	grades[3][2]
	↓	↓	↓
quizAve[]:	quizAve[0]	quizAve[1]	quizAve[2]

Παράδειγμα μέσου όρου

- Δυδιάστατος πίνακας (grades) με βαθμολογίες φοιτητών
 - Μία γραμμή ανά φοιτητή
 - Στήλες – βαθμολογία σε test
- Εκτύπωση πίνακα, μέσου όρου / φοιτητή, μέσου όρου / test

	test1	test2	test3	stAv[]:
φοιτητής1	grades[0][0]	grades[0][1]	grades[0][2] →	stAv[0]
φοιτητής2	grades[1][0]	grades[1][1]	grades[1][2] →	stAv[1]
φοιτητής3	grades[2][0]	grades[2][1]	grades[2][2] →	stAv[2]
φοιτητής3	grades[3][0]	grades[3][1]	grades[3][2] →	stAv[3]
	↓	↓	↓	
quizAve[]:	quizAve[0]	quizAve[1]	quizAve[2]	

```

#include <iostream>
const int NUMBER_ST = 4, NUMBER_QZ = 3;

void computeStAve(const int grade[][NUMBER_QZ], double stAve[]);

void computeQuizAve(const int grade[][NUMBER_QZ], double quizAve[]);

void display(const int grade[][NUMBER_QZ],
             const double stAve[], const double quizAve[]);

int main( )
{
    int grade[NUMBER_ST][NUMBER_QZ];
    double stAve[NUMBER_ST], quizAve[NUMBER_QZ];

    grade[0][0] = 10; grade[0][1] = 10; grade[0][2] = 10;
    grade[1][0] = 2;  grade[1][1] = 0;  grade[1][2] = 1;
    grade[2][0] = 8;  grade[2][1] = 6;  grade[2][2] = 9;
    grade[3][0] = 8;  grade[3][1] = 4;  grade[3][2] = 10;

    computeStAve(grade, stAve);
    computeQuizAve(grade, quizAve);
    display(grade, stAve, quizAve);
}

```

```

void computeStAve(const int grade[][NUMBER_QZ], double stAve[])
{
    for (int stNum = 1; stNum <= NUMBER_ST; stNum++)
    {
        double sum = 0;
        for (int quizNum = 1; quizNum <= NUMBER_QZ; quizNum++)
            sum = sum + grade[stNum-1][quizNum-1];

        stAve[stNum-1] = sum/NUMBER_QZ;
    }
}

void computeQuizAve(const int grade[][NUMBER_QZ], double quizAve[])
{
    for (int quizNum = 1; quizNum <= NUMBER_QZ; quizNum++)
    {
        double sum = 0;
        for (int stNum = 1; stNum <= NUMBER_ST; stNum++)
            sum = sum + grade[stNum-1][quizNum-1];

        quizAve[quizNum-1] = sum/NUMBER_ST;
        //Average for quiz quizNum is the value of quizAve[quizNum-1]
    }
}

```

```
void display(const int grade[][NUMBER_QZ],
             const double stAve[], const double quizAve[])
{
    for (int stNum = 1; stNum <= NUMBER_ST; stNum++)
    {
        cout << stNum << stAve[stNum-1] << " ";
        for (int quizNum = 1; quizNum <= NUMBER_QZ; quizNum++)
            cout << grade[stNum-1][quizNum-1];
        cout << endl;
    }

    cout << "Quiz averages = ";
    for (int quizNum = 1; quizNum <= NUMBER_QZ; quizNum++)
        cout << quizAve[quizNum-1];
    cout << endl;
}
```

Πίνακες (σύννοψη)

- Είναι μια συλλογή από μεταβλητές
- Βρόχοι for ταιριάζουν απόλυτα για τους πίνακες
- Είστε υπεύθυνοι για να μην βγείτε έξω από τα όρια του πίνακα
- Η παράμετρος Πίνακα είναι ένα "νέος" τύπος
 - Παρόμοια με το παράμετρο με αναφορά
- Τα στοιχεία του πίνακα αποθηκεύονται σειριακά
 - "Συνεχόμενο" κομμάτι στην μνήμη
 - Μόνο η διεύθυνση του 1^{ου} στοιχείου περνάει σε συν/σεις
- Μερικώς συμπληρωμένοι πίνακες → περισσότερες μεταβλητές
- Πολυδιάστατοι πίνακες
 - "πίνακας από πίνακες"

Καλή Μελέτη

- **Βιβλιογραφία**

[1] W. Savitch, Πλήρης C++, Εκδόσεις Τζιόλα, 2011

[2] H. Deitel and P. Deitel, C++ Προγραμματισμός 6η Εκδοση, Εκδόσεις Μ. Γκιούρδας, 2013

Ύλη βιβλιογραφίας

[1]: 5.1, 5.2, 5.4

[2]: Κεφ. 7

Ανοικτά Ακαδημαϊκά Μαθήματα

Πανεπιστήμιο Ιωαννίνων

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Ιωαννίνων**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



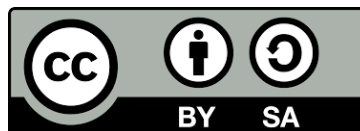
Σημειώματα

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Ιωαννίνων, Διδάσκων: Λέκτορας Χάρης Παπαδόπουλος «Εισαγωγή στον Προγραμματισμό». Έκδοση: 1.0. Ιωάννινα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: <http://ecourse.uoi.gr/course/view.php?id=1105>.

Σημείωμα Αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Παρόμοια Διανομή, Διεθνής Έκδοση 4.0 [1] ή μεταγενέστερη.



[1] <https://creativecommons.org/licenses/by-sa/4.0/>.