Προγραμματισμός Ι

Πίνακες

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Πίνακες

Πολλές φορές θέλουμε να κρατήσουμε στην μνήμη πολλά αντικείμενα ίδιου τύπου.

π.χ

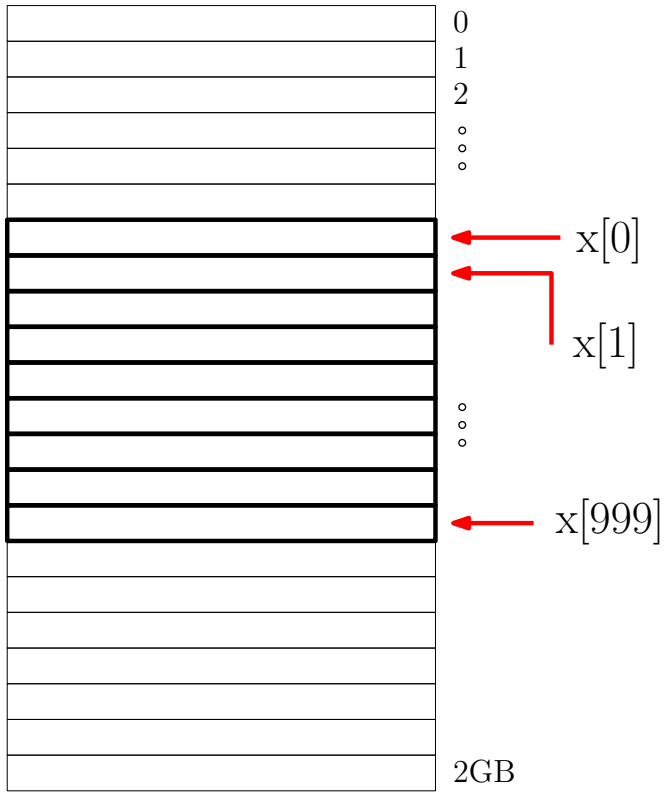
- μπορεί κάποιος να μας ζητήσει να διαβάσουμε 1000 ακεραίους και να τους βάλουμε σε μη-αύξουσα σειρά.
- μπορεί να θέλουμε να αποθηκεύσουμε 100 ονόματα φοιτητών με τους βαθμούς τους

Πίνακες

Ένας πίνακας είναι μια ομάδα συνεχόμενων θέσεων μνήμης
που έχουν το ίδιο όνομα και τον ίδιο τύπο.

Δήλωση Πίνακα σε C

```
1  int main() {
2      int x[1000];
3  }
```



Βασικά Χαρακτηριστικά Πινάκων στην C

```
1  int main() {
2      int x[1000];
3  }
```

1. Το πρώτο στοιχείο είναι πάντα το μηδενικό
2. Τα στοιχεία είναι πάντα συνεχόμενα στην μνήμη
3. Το μέγεθος ενός πίνακα πρέπει να είναι ακέραιο

Πρόσβαση σε Στοιχείο του Πίνακα

Για να πάρουμε ένα στοιχείο του πίνακα δίνουμε:

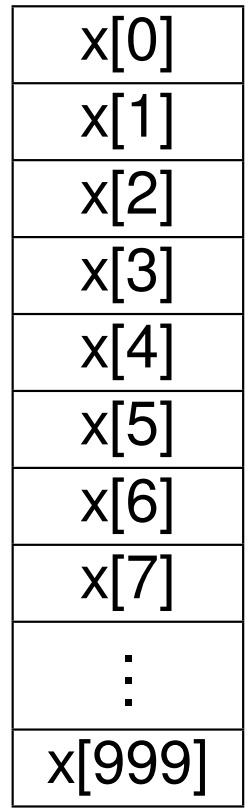
- όνομα πίνακα
- θέση

π.χ

```
x[3] = 100;
```

ή

```
printf("%d", x[3]);
```



Τα στοιχεία ενός πίνακα με μέγεθος n είναι προσβάσιμα με ακέραιους δείκτες από το μηδέν έως το $n - 1$.

Πρόσβαση σε Στοιχείο του Πίνακα

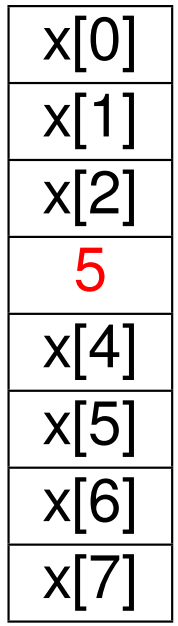
```
1  int main() {
2      int i;
3      int x[1000];
4
5      for(i = 0; i < 1000; i++) {
6          x[i] = 0;
7      }
8
9      return 0;
10 }
```

Τα στοιχεία ενός πίνακα με μέγεθος n είναι προσβάσιμα με ακέραιους δείκτες από το μηδέν έως το $n - 1$.

Στοιχεία Πίνακα

Τα στοιχεία ενός πίνακα χρησιμοποιούνται σαν συνηθισμένες μεταβλητές.

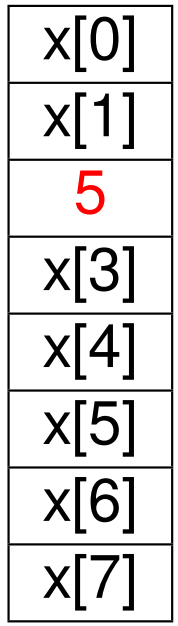
```
1  int main() {
2      int x[8];
3
4      x[3]= 5;
5
6      printf("%d\n", x[3]);
7
8      return 0;
9  }
```



Στοιχεία Πίνακα

Μπορούμε να κάνουμε πράξεις μέσα στον δείκτη.

```
1  int main() {
2      int x[8], j;
3
4      j = 1;
5      x[j+1] = 5;
6
7      if(x[4-2] == 5) {
8          printf("x[2] = 5\n");
9      }
10
11     return 0;
12 }
```



Στοιχεία Πίνακα σε Εκφράσεις

Πρίν χρησιμοποιηθεί ο πίνακας πρέπει να υπολογιστεί η έκφραση μέσα στα [ ].

```
1  int cyclenext8(int i) {
2      if (i >= 8||i < 0) {
3          return 0;
4      } else {
5          return i+1;
6      }
7  }
8
9  int main() {
10     int i, y[8], x[8];
11
12     for(i=0; i < 8; i++)
13         x[i] = i;
14
15     for(i = 0; i < 8; i++)
16         y[i] = x[cyclenext8(i)];
17
18     return 0;
19 }
```

Επίσης ο τύπος ενός πίνακα πρέπει να ταιριάζει με τον τύπο που μια έκφραση περιμένει.

Προτεραιότητα Τελεστών

Η προτεραιότητα υπολογισμού της έκφρασης [ ] είναι η υψηλότερη δυνατή.

1. **παρενθέσεις:** $()$, $[]$, **expr**++ ή **expr**−−
   Υπολογίζονται πρώτα, από τα αριστερά προς τα δεξιά. Εάν υπάρχουν ένθετες υπολογίζονται πρώτα οι εσωτερικές.
2. **μοναδιαίοι τελεστές:** $+$, $-$, **++expr**, −−**expr** και $!$
   Υπολογίζονται από δεξιά προς τα αριστερά.
3. **πολλαπλασιασμός, διαίρεση και υπόλοιπο:** $*$, $/$, ή $\%$
   Υπολογίζονται δεύτερα από αριστερά προς τα δεξιά.
4. **πρόσθεση, αφαίρεση:** $+$ ή $-$
   Εάν υπάρχουν πολλοί, υπολογίζονται από τα αριστερά προς τα δεξιά.
5. **Σχεσιακοί:** $<$, $>$, $<=$, $>=$
   Υπολογίζονται από τα αριστερά προς τα δεξιά.
6. **Ισότητας:** $==$, $!=$
   Υπολογίζονται από τα αριστερά προς τα δεξιά.
7. **λογικό AND:** && Από αριστερά προς τα δεξιά.
8. **λογικό OR:** || Από αριστερά προς τα δεξιά.
9. **εκχώρησης:** $=$, $+=$, $-=$, $*=$, $/=$, $\%=$
   Από δεξιά προς τα αριστερά.

Αρχικοποίηση Πίνακα

Όπως όλες οι μεταβλητές στην C έτσι και ο πίνακας χρειάζεται αρχικοποίηση:

```
1  int main() {
2      int i;
3      int x[1000];
4
5      for(i = 0; i < 1000; i++) {
6          x[i] = 0;
7      }
8  }
```

Αρχικοποίηση Πίνακα

Σε περίπτωση που ο πίνακας είναι μικρός μπορούμε να τον αρχικοποιήσουμε απευθείας την ώρα της δήλωσης του:

```
1  int main() {
2      int x[10] = { 32, 27, 64, 18, 95,
3                    14, 90, 70, 60, 37 };
4  }
```

32
27
64
18
95
14
90
70
60
37

Αρχικοποίηση Πίνακα

Ένας πίνακας δεν αρχικοποιείτε αυτόματα.

```
1  int main() {
2      int x[1000] = { 0 };
3  }
```

Εάν όμως αρχικοποιήσουμε λιγότερα στοιχεία από το μέγεθος του ο μεταγλωττιστής φροντίζει τα υπόλοιπα στοιχεία να πάρουν την τιμή μηδέν.

Αρχικοποίηση Πίνακα

Ο μεταγλωττιστής μπορεί να καταλάβει το μέγεθος του πίνακα από την λίστα αρχικοποίησης.

```
1  int main() {
2      int x[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
3  }
```

Ο πίνακας **x**[] είναι στην ουσία **x**[9].

Μέγεθος Πίνακα και Σταθερές

Θυμηθείτε τις σταθερές που υποστηρίζει ο προ-επεξεργαστής:

```
1  #include <stdio.h>
2
3  #define SIZE 100
4
5  int main() {
6      int a[SIZE];
7      for(int i = 0; i < SIZE; i++) {
8          a[i] = i;
9      }
10
11     for(int i = 0; i < SIZE; i++) {
12         printf("%d", a[i]);
13
14         if (i < SIZE-1)
15             printf(" ");
16         else
17             printf("\n");
18     }
19     return 0;
20 }
```

Μέγεθος Πίνακα και Σταθερές

Θα δούμε αργότερα πως μερικές φορές το μέγεθος ενός πίνακα δεν μπορεί να είναι γνωστό κατά την διάρκεια της μεταγλώττισης.

Για αυτό η C παρέχει δύο τύπους πίνακα:

1. στατικούς πίνακες
2. δυναμικούς πίνακες

Οι δυναμικοί πίνακες **δεν** θα μας απασχολήσουν ακόμη. Όλοι οι πίνακες που έχουμε δει ως τώρα ήταν στατικοί.

Σφάλμα Εκτός Ορίων

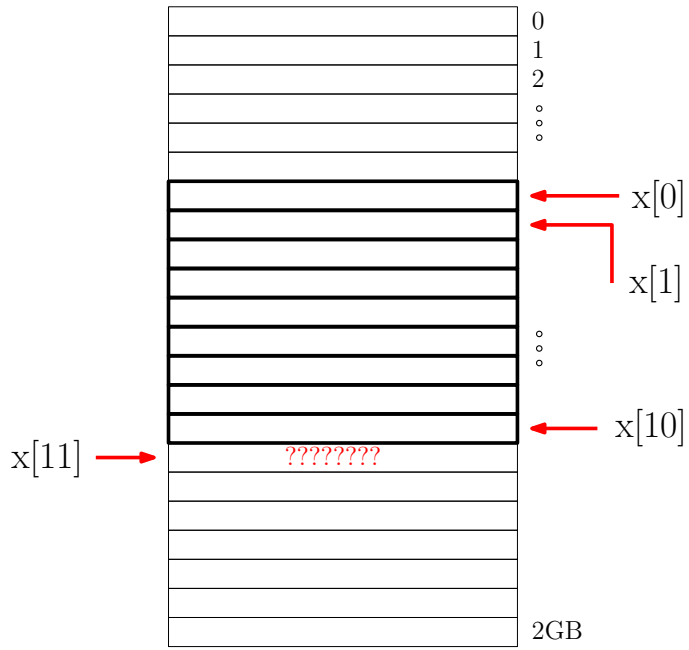
Τι συμβαίνει άμα φτιάξουμε ένα πίνακα 10 στοιχείων και κατά λάθος χρησιμοποιήσουμε την 11η τιμή;

```
1  #include <stdio.h>
2
3  int main() {
4      int a[10] = { 10, 1, 3, 4, 5, 9, 2, 9, 10, 10 };
5
6      for(int i = 0; i < 11; i++) {   /* RUN-TIME ERROR */
7          printf("%d\n", a[i]);
8      }
9
10     return 0;
11 }
```

Επειδή δεν ξέρουμε τι υπάρχει στην θέση μνήμης που ακολουθεί τον πίνακα, μπορεί να συμβεί οτιδήποτε.

Σφάλμα Εκτός Ορίων

Τι συμβαίνει άμα φτιάξουμε ένα πίνακα 10 στοιχείων και κατά λάθος χρησιμοποιήσουμε την 11ή τιμή;



Συνήθως είτε παίρνουμε κάτι που είναι ”σκουπίδια” ή τερματίζει απότομα το πρόγραμμα μας (crash).

Παράδειγμα Χρήσης Πίνακα

Άθροισμα

```
1  #include <stdio.h>
2  #define SIZE 12
3
4  int main() {
5      int a[SIZE] = { 1, 3, 5, 4, 7, 2, 99,
6                      16, 45, 67, 89, 45 };
7      int i, total = 0;
8
9      for(i = 0; i < SIZE; i++) {
10         total += a[i];
11     }
12
13     printf("Total of array elements is %d\n", total);
14
15     return 0;
16 }
```

εκτυπώνει

Total of array elements is 383

Παράδειγμα

Μέσος Όρος

```
1  #include <stdio.h>
2  #define SIZE 6
3
4  int main() {
5      double a[SIZE] = { 1.0, 2.3, 1.3333, 7.102, 1.523, 1.0001 };
6      int i;
7      double avg = 0.0;
8
9      for(i = 0; i < SIZE; i++) {
10         avg += a[i];
11     }
12
13     avg /= SIZE;
14
15     printf("average is %lf\n", avg);
16
17     return 0;
18 }
```

εκτυπώνει

average is 2.376400

Πίνακες Χαρακτήρων

Τα αλφαριθμητικά που βλέπουμε διαρκώς στην χρήση της **`printf`** είναι στην ουσία πίνακες χαρακτήρων με το ιδιαίτερο χαρακτηριστικό πως ο τελευταίος χαρακτήρας είναι ο ειδικός κενός χαρακτήρας `'\0'`.

```
1  int main() {
2      char string[6];
3
4      string[0] = 'f';
5      string[1] = 'i';
6      string[2] = 'r';
7      string[3] = 's';
8      string[4] = 't';
9      string[5] = '\0';
10 }
```

Ο κενός χαρακτήρας είναι απαραίτητος ώστε να μπορούν οι διάφορες συναρτήσεις να ξέρουν το τέλος του αλφαριθμητικού χωρίς να τους δίνουμε το μέγεθος του πίνακα.

Πίνακες Χαρακτήρων

Το παρακάτω πρόγραμμα

```
1  int main() {
2      char string[6];
3
4      string[0] = 'f';
5      string[1] = 'i';
6      string[2] = 'r';
7      string[3] = 's';
8      string[4] = 't';
9      string[5] = '\0';
10
11     printf( "%s\n", string );
12     printf( "first\n" );
13 }
```

τυπώνει 2 φορές την λέξη *first*.

Πίνακες Χαρακτήρων

Μπορούμε όμως να αρχικοποιήσουμε πιο γρήγορα τον πίνακα χαρακτήρων:

```
1  int main() {
2      char string[6] = "first";
3
4      printf("%s\n", string);
5      printf("first\n");
6  }
```

ο μεταγλωττιστής βάζει αυτόματα το `'\0'`.

Πίνακες Χαρακτήρων

Μπορούμε και να παραλείψουμε το μέγεθος:

```
1  int main() {
2      char string[] = "first";
3
4      printf("%s\n", string);
5      printf("first\n");
6  }
```

ο μεταγλωττιστής το βρίσκει αυτόματα.

Προσοχή γιατί το μέγεθος του πίνακα είναι ένα παραπάνω από το μήκος του αλφαριθμητικού.

Πίνακες Χαρακτήρων

Η θέση του `'\0'`

Η θέση του κενού χαρακτήρα καθορίζει το τέλος ενός αλφαριθμητικού:

```
1  #include <stdio.h>
2
3  int main() {
4      char string[11] = { 't', 'w', 'o', '\0', ' ', 'w',
5                          'o', 'r', 'd', 's', '\0' };
6
7      printf("%s\n", string);
8  }
```

Το παραπάνω πρόγραμμα τυπώνει την λέξη *two* και όχι *two words*, αφού η `printf` σταματάει μόλις δει τον κενό χαρακτήρα.

Πίνακες Χαρακτήρων

Αρχικοποίηση σε κενό αλφαριθμητικό

Η θέση του κενού χαρακτήρα καθορίζει το τέλος ενός αλφαριθμητικού:

```
1  #include <stdio.h>
2
3  int main() {
4      char string[11] = { 't', 'w', 'o', '\0', ' ', 'w',
5                          'o', 'r', 'd', 's', '\0' };
6
7      printf("%s\n", string);
8
9      string[0] = '\0';
10 }
```

Μπορούμε να δώσουμε σε έναν πίνακα χαρακτήρων την τιμή του κενού αλφαριθμητικού βάζοντας τον πρώτο χαρακτήρα να πάρει την τιμή `'\0'`.

Πίνακες και Συναρτήσεις

Για να ορίσουμε μια συνάρτηση που παίρνει ένα πίνακα ως όρισμα:

```
1  void swap(int array[], int i, int j) {
2      /* code to swap element array[i] and array[j]
3  }
```

έπειτα την καλούμε:

```
1  int main() {
2      int a[5] = { 1, 2, 2, 3, 1 };
3
4      swap(a, 2, 3);
5  }
```

δίνοντας μόνο το όνομα του πίνακα.

Πίνακες και Call-By-Value

Παρόλο που η C χρησιμοποιεί πάντα **κλήση μέσω τιμής**, οι τιμές των στοιχείων ενός πίνακα μπορούν να αλλάξουν μέσα σε μια συνάρτηση.

```
1  #include <stdio.h>
2
3  void increment(int[] a, int n) {
4      int i;
5      for(i = 0; i < n; i++) {
6          a[i]++;
7      }
8  }
9
10 int main() {
11     int a[] = { 1, 2, 3 };
12
13     increment(a, 3);
14
15     /* here a[] = { 2, 3, 4} */
16 }
```

Πίνακες και Call-By-Value

Η χρήση του **const** λέει στον μεταγλωττιστή να μην επιτρέπει καμία αλλαγή στοιχείου πίνακα μέσα σε μια συνάρτηση.

```
1  #include <stdio.h>
2
3  void increment(const int[] a, int n) {   /* const here */
4      int i;
5      for(i = 0; i < n; i++) {
6          a[i]++;    /* ERROR */
7      }
8  }
9
10 int main() {
11     int a[] = { 1, 2, 3 };
12
13     increment(a, 3);
14 }
```

Παράδειγμα

Εναλλαγή Στοιχείων Πίνακα

```
1  #include <stdio.h>
2
3  void swap(int a[], int i, int j);
4
5  int main() {
6      int n[] = { 1, 2, 3 };
7      swap(n, 0, 1);
8      swap(n, 1, 2);
9  }
10
11 void swap(int a[], int i, int j) {
12     int temp = a[i];
13     a[i] = a[j];
14     a[j] = temp;
15 }
```

Παράδειγμα

Αντιστροφή Πίνακα

```
1  #include <stdio.h>
2
3  void swap(int a[], int i, int j) {
4      int temp = a[i];
5      a[i] = a[j];
6      a[j] = temp;
7  }
8
9  void reverse(int a[], int n) {
10     int i;
11     for(i = 0; i < n/2; i++) {
12         swap(a, i, n-i-1);
13     }
14 }
15
16 int main() {
17     int a[] = { 1, 2, 3, 4, 5 };
18     reverse(a, 5);
19 }
```

Πίνακες Πολλών Διαστάσεων

Η C μας επιτρέπει να ορίσουμε και πολυδιάστατους πίνακες:

```
1 int marray[3][4];
```

π.χ ένας πίνακας ακεραίων με 3 γραμμές και 4 στήλες που ονομάζεται **marray**.

Πίνακες Πολλών Διαστάσεων

$a[0][0]$	$a[0][1]$	$a[0][2]$	$a[0][3]$
$a[1][0]$	$a[1][1]$	$a[1][2]$	$a[1][3]$
$a[2][0]$	$a[2][1]$	$a[2][2]$	$a[2][3]$

και ένα κλασσικό **loop** για να αρχικοποιήσουμε τον πίνακα

```
1  for(i = 0; i < 3; i++) {
2      for(j = 0; j < 4; j++) {
3          a[i][j] = 0;
4      }
5  }
```

Αρχικοποίησης Πινάκων Πολλών Διαστάσεων

```
1  int a[2][2] = { { 1, 2 }, { 3, 4 } };
```

Οι τιμές ομαδοποιούνται ανά σειρά σε άγκιστρα.

Πίνακες Πολλών Διαστάσεων και Συναρτήσεις

```
1  #include <stdio.h>
2  #define COLUMNS  3
3
4  int max(const int a[][COLUMNS], int rows, int columns) {
5      int i,j;
6      int max = a[0][0];
7
8      for(i = 0; i < rows; i++) {
9          for(j = 0; j < columns; j++) {
10             max = a[i][j] > max?a[i][j]:max;
11         }
12     }
13
14     return max;
15 }
16
17 int main() {
18     const int a[2][COLUMNS] = { { 1, 2, 3 }, { 1000, 0, 1 } };
19     printf("maximum = %d\n", max(a, 2, COLUMNS));
20 }
```

Είναι δυνατόν να παραλείψουμε την πρώτη διάσταση σε μια συνάρτηση, αλλά όχι και την δεύτερη αφού πρέπει να ξέρει ο μεταγλωττιστής την μορφή του πίνακα.

Πίνακες Πολλών Διαστάσεων και Συναρτήσεις

Οι διδιάστατοι πίνακες υλοποιούνται με μονοδιάστατους στην C, π.χ σε έναν πίνακα `int a[3][4]` έχουμε

- `a[i][j]` = `a[4*i + j]`

Για αυτό τον λόγο πρέπει ο μεταγλωττιστής να ξέρει τον αριθμό στηλών (το 4 στο προηγούμενο παράδειγμα) όταν καλείτε μια συνάρτηση.

Διδιάστατοι Πίνακες

Πολλαπλασιασμός Πινάκων

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{np} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{pmatrix}$$

```
1  for(i = 0; i < n; i++) {
2      for(j = 0; j < p; j++) {
3          for(k = 0, c[i][j] = 0.0; k < m; k++) {
4              c[i][j] += a[i][k] * b[k][j];
5          }
6      }
7  }
```