# Δομές Δεδομένων

**Διονύσης "dionyziz" Ζήνδρος**

Camp Προετοιμασίας ΠΔΠ

Άνοιξη 2011

# Απλοί τύποι δεδομένων

- Ακέραιος: 3, 5, -7, 65535
- Ρητός: 3.5, 9.7777, 4.000001
- Χαρακτήρας: 'a', 'c', '$'
- Λογική τιμή: Αλήθεια ή Ψεύδος
- Αλφαριθμητικά: "mitsos", "mpaniera"

Δεν μας αρκούν! Πώς θα αποθηκεύαμε...

- Ένα σύνολο μαθητών;
- Μία λίστα από πόλεις;
- Μαθητές μαζί με τους βαθμούς τους;

# Δομή δεδομένων

- Τρόπος αναπαράστασης σύνθετων δεδομένων
- Τρόπος σκέψης

Με την κατάλληλη δομή δεδομένων...

- Προγράμματα που τρέχουν γρηγορότερα
- Χρησιμοποιούν λιγότερη μνήμη
- Επίλυση προβλημάτων που δεν μπορούσαμε να λύσουμε

# Δομές που θα εξετάσουμε

- Πίνακες
- Λίστες
- Ουρές (προτεραιότητας)
- Στοίβες
- Χάρτες
- Δέντρα

# Πίνακες

# Πίνακες

- Ένα διατεταγμένο σύνολο από αντικείμενα
- Αντικείμενα ίδιου τύπου
- Σε συνεχείς θέσεις μνήμης
- Σταθερό πλήθος στοιχείων

- Οι πρώτοι 10 πρώτοι αριθμοί:
  2, 3, 5, 7, 11, 13, 17, 19, 23, 29

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| **2** | **3** | **5** | **7** | **11** | **13** | **17** | **19** | **23** | **29** |

# Τα καλά με τους πίνακες

- Γράφεις και διαβάζεις σε όποια θέση θέλεις άμεσα:

```
a[ 5 ] = 14;
printf( "%i\n", a[ 5 ] );
```

Χρόνος O( 1 )

Χρόνος O( 1 )

- Εύκολη σκέψη και χρήση
- Πιάνουν τον ελάχιστο δυνατό χώρο

# Τα κακά με τους πίνακες

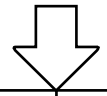
- Δεν αλλάζει το πλήθος στοιχείων
- Διαγραφή στοιχείου είναι αργή
- Προσθήκη στοιχείου είναι αργή

# Αρχικοποίηση

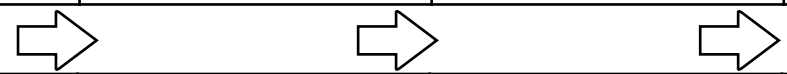
```
for ( i = 0; i <= N; ++i ) {
    names[ i ] = 0;
}
```

# Προσθήκη στοιχείου

- Εύκολα προσθέτουμε ένα στοιχείο στο τέλος.
- Για να προσθέσουμε στην αρχή ή στη μέση μετακινούμε όλα τα επόμενα.



| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Γιώργος | Νίκος | Πέτρος | Βασίλης | |



| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Γιώργος | | Νίκος | Πέτρος | Βασίλης |

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Γιώργος | Μίλτος | Νίκος | Πέτρος | Βασίλης |

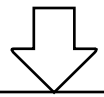
# Προσθήκη στοιχείου

```
string names[ 5 ] =
{"Giorgos", "Nikos", "Petros", "Basilis", "" };

for ( i = 3; i >= 1; --i ) { // 1, 2, 3
    names[ i + 1 ] = names[ i ];
}

names[ 1 ] = "Miltos";
```

Χρόνος O( n ).

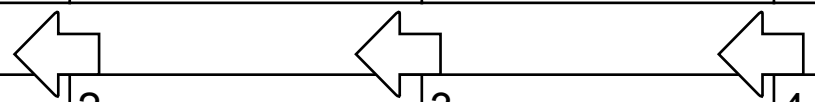
# Διαγραφή στοιχείου

- Εύκολα διαγράφουμε ένα στοιχείο στο τέλος.
- Για να διαγράψουμε στην αρχή ή στη μέση μετακινούμε όλα τα επόμενα.



| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Γιώργος | Μίλτος | Νίκος | Πέτρος | Βασίλης |

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Γιώργος | | Νίκος | Πέτρος | Βασίλης |



| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Γιώργος | Νίκος | Πέτρος | Βασίλης | |

# Διαγραφή στοιχείου

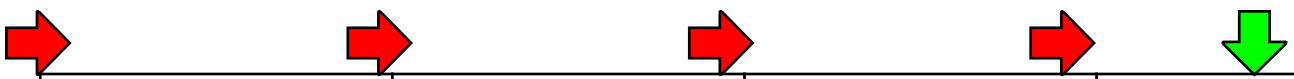
```
for ( i = 2; i >= 4; ++i ) { // 2, 3, 4
    names[ i - 1 ] = names[ i ];
}
```

Χρόνος O( n ).

# Γραμμική αναζήτηση

- Ψάχνουμε αν μία τιμή υπάρχει στον πίνακα
- Και τη θέση της αν υπάρχει
- Ελέγχουμε ένα-ένα τα στοιχεία του πίνακα

Πού εμφανίζεται ο "Πέτρος";



| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Γιώργος | Μίλτος | Νίκος | Πέτρος | Βασίλης |

# Γραμμική αναζήτηση

```
flag = 0;
for ( i = 0; i < 5; ++i ) {
    if ( names[ i ] == "Petros" ) {
        printf( "%i\n", i );
        flag = 1;
        break;
    }
}
if ( flag == 0 ) {
    printf( "Den brethike\n" );
}
```

Χρόνος O( n ).

# Σύνθετοι τύποι δεδομένων

- Κάθε στοιχείο είναι ένας συνδυασμός απλών τύπων
- Ο ίδιος συνδυασμός για κάθε στοιχείο
- Μαθητής:
  - Όνομα (string)
  - Επώνυμο (string)
  - Βαθμός (int)
- Κάθε μαθητής είναι string - string - int

# Σύνθετοι τύποι δεδομένων

```
struct student {
    string first;
    string last;
    int grade;
};

student Totos, Mpompos;

Totos.first = "Totos"; Totos.last = "Totakis";
Totos.grade = 0;

printf( "%i\n", Totos.grade );
```

# Σύνθετοι τύποι και πίνακες

- Πίνακας από σύνθετους τύπους:

```
student Camp_Juniors[ 20 ];
```

- Σύνθετος τύπος με πίνακα:

```
struct student {
    string first;
    string last;
    int grades[ 5 ];
};
```

# Ταξινόμηση πίνακα

- Βάζει τα στοιχεία σε μία σειρά
  - αύξουσα, φθίνουσα
  - αύξουσα με βάση το επώνυμο
  - φθίνουσα με βάση το βαθμό
  - κλπ.
- Εμείς καθορίζουμε τη σειρά

Στην C++ για πίνακες χρησιμοποιούμε την qsort.

# Ταξινόμηση πίνακα

#include <cstdlib>

qsort( πίνακας, πλήθος, μέγεθος, σειρά );

- πλήθος: Πόσα στοιχεία έχει ο πίνακας;

μέγεθος: sizeof( τύπος ) π.χ. sizeof( student ) ή sizeof( int )

- σειρά: Δείκτης σε συνάρτηση που γράφουμε εμείς

Χρόνος O( n * log( n ) )

# Ταξινόμηση πίνακα

```
int cmp( const void* a, const void* b ) {
    int* aa = ( int* )a;
    int* bb = ( int* )b;
    int one = *aa;
    int two = *bb;

    if ( one > two ) {
        return 1;
    }
    else if ( two > one ) {
        return -1;
    }
    return 0;
}
```

# Συντομότερα

```
int cmp( const void* a, const void* b ) {
    return ( *( int* )a - *( int* )b );
}
```

# Ταξινόμηση πίνακα

```
int a[ 5 ] = { 5, 2, 4, 0, -1 };

qsort( a, 5, sizeof( int ), cmp );
```

# Ένα παιχνίδι με αριθμούς

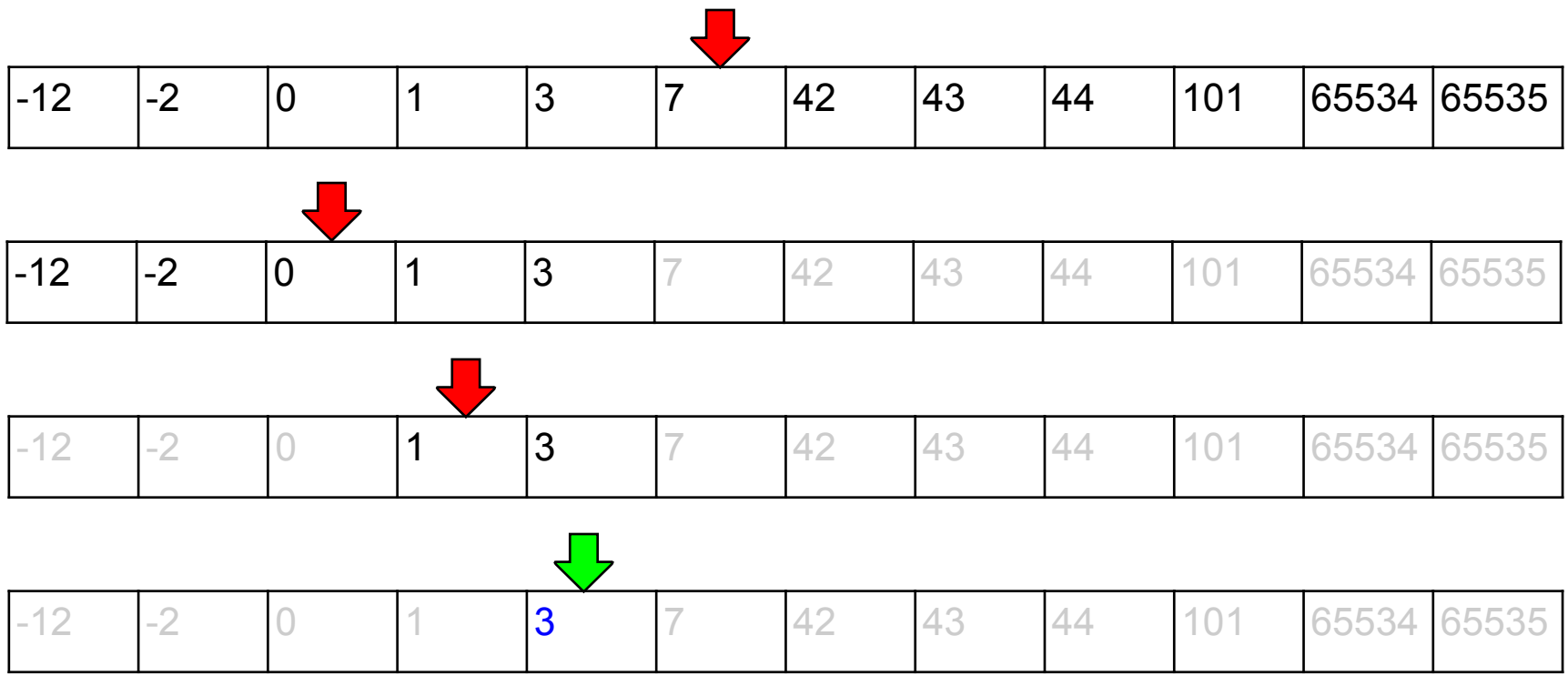
- Ο αντίπαλος σκέφτεται έναν αριθμό 1...100
- Προσπαθούμε να τον μαντέψουμε με ερωτήσεις τύπου "ναι" ή "όχι".
- Όσες πιο λίγες ερωτήσεις γίνεται.

# Γρήγορη αναζήτηση

- Binary search - δυαδική αναζήτηση
- Απαιτεί ο πίνακας να είναι ταξινομημένος
  - Εύκολο με qsort
- Χρήσιμο όταν θέλουμε να αναζητήσουμε πολλές φορές στον ίδιο πίνακα.
- Κοιτάμε το μεσαίο στοιχείο του πίνακα.
- Αν είναι ίδιο τελείωσαμε.
- Αν είναι μικρότερο, κοιτάμε το δεξί μέρος του πίνακα.
- Αν είναι μεγαλύτερο, κοιτάμε το αριστερό.

# Γρήγορη αναζήτηση

- Βρες το 3



# Γρήγορη αναζήτηση

```
#include <cstdlib>

bsearch( κλειδί,
        πίνακας, πλήθος, μέγεθος, σειρά );
```

κλειδί: Δείκτης στην τιμή που ψάχνουμε

- Επιστρέφει: Δείκτη στο στοιχείο που βρέθηκε
- Ή NULL αν δεν βρεθεί

Χρόνος O( log( n ) )

# Γρήγορη αναζήτηση

```
int a[ 5 ] = { 5, 2, 4, 0, -1 };
int key = -1;
int* location;
location = bsearch( &key, a, 5,
                    sizeof( int ), cmp );
if ( location == NULL ) {
    printf( "Den vrethike\n" );
}
else {
    printf( "Thesi: %i\n", location - a );
}
```

# Πολυδιάστατοι πίνακες

Μαθήματα

| a | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 20 | 19 | 20 | 20 | 20 |
| 1 | 7 | 3 | 0 | 9 | 11 |
| 2 | 15 | 12 | 13 | 15 | 14 |
| 3 | 9 | 11 | 12 | 10 | 13 |
| 4 | 17 | 17 | 17 | 19 | 18 |

j

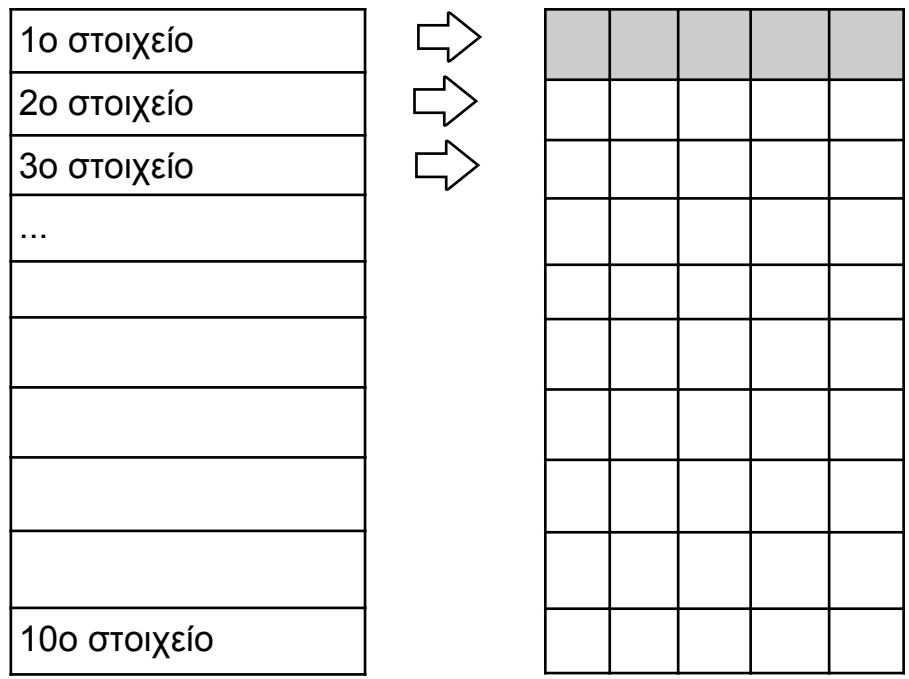
Μαθητές

i

a[ i ][ j ]

# Πολυδιάστατοι πίνακες

- Πίνακας που κάθε στοιχείο του είναι ένας πίνακας
- Διδιάστατος πίνακας 10x5:
  - Είναι ένας μονοδιάστατος πίνακας 10 στοιχείων
  - Κάθε στοιχείο είναι ένας μονοδιάστατος πίνακας 5 στοιχείων



# Αρχικοποίηση

```
for ( i = 0; i <= N; ++i ) {
    for ( j = 0; j <= N; ++j ) {
        names[ i ][ j ] = 0;
    }
}
```

# Παράδειγμα: Μέσος όρος

```
avg = 0;

for ( i = 0; i <= n; ++i ) {
    for ( j = 0; j <= m; ++j ) {
        avg += a[ i ][ j ];
    }
}

avg = avg / ( m * n );
```