

Πίνακες αντικειμένων και διανύσματα αντικειμένων

#8

Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)

Πανεπιστήμιο Ιωαννίνων

Γκόγκος Χρήστος

Πίνακες: επισκόπηση

- Ένας πίνακας είναι μια δεικτοδοτημένη συλλογή από στοιχεία δεδομένων του ίδιου τύπου (ομοιογενής δομή)
- Για παράδειγμα η ακόλουθη δήλωση:
`int a[10];`
Δηλώνει έναν πίνακα με 10 στοιχεία
Κάθε στοιχείο του πίνακα είναι ένας ακέραιος
- Ποιο είναι το εύρος αποδεκτών τιμών για το δείκτη στον παραπάνω πίνακα;
 - Από 0 μέχρι και 9
- Τι αναπαριστούν τα 10 στοιχεία του πίνακα;
 - Αναπαριστούν συνεχόμενες θέσεις μνήμης, κάθε μια με μέγεθος `sizeof(int)`

Συνηθισμένες λειτουργίες πινάκων

- Διάσχιση πίνακα
- Ενημέρωση τιμών πίνακα
- Εύρεση αθροίσματος
- Εύρεση μεγίστου/ελαχίστου
- Αναζήτηση στοιχείου στον πίνακα
- Ταξινόμηση πίνακα

<https://github.com/chgogos/oop/blob/master/variou.../COP3330/lect8/arrays2.cpp>

<https://github.com/chgogos/oop/blob/master/variou.../COP3330/lect8/arrays1.cpp>

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

int main() {
    int arr[] = {5, 2, 9, 1, 5, 6};
    cout << "Array elements: ";
    for (int i = 0; i < sizeof(arr) / sizeof(arr[0]); i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
    return 0;
}
```

Array elements: 5 2 9 1 5 6

Περιορισμοί πινάκων

- Έχουν στατικό μέγεθος
- Για να αυξηθεί ή να μειωθεί το μέγεθος του πίνακα θα πρέπει να γραφεί επιπλέον κώδικας
- Δεν υπάρχει έλεγχος ορίων (ο κώδικας μπορεί να προσπελάσει και στοιχεία εκτός των ορίων του πίνακα, με απρόβλεπτες συνέπειες)

[https://github.com/chgogos/oop/blob/master/variou...
0/lect8/no_bounds_check.cpp](https://github.com/chgogos/oop/blob/master/variou...)

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

int main() {
    int arr[3] = {10, 20, 30};
    cout << "Valid index access: arr[2] = " << arr[2] << endl;
    // Out of bounds access – undefined behavior!
    cout << "Invalid index access: arr[5] = " << arr[5] << endl;
    return 0;
}
```

Valid index access: arr[2] = 30
Invalid index access: arr[5] = 931703248

Δήλωση πινάκων αντικειμένων

- Η δήλωση πινάκων με αντικείμενα είναι παρόμοια με τη δήλωση πινάκων για πρωτογενείς τύπους δεδομένων (π.χ. int, float).

 Fraction rationals[20];

 Complex nums[50];

 Hydrant fireplugs[10];

- Σε κάθε θέση του πίνακα υπάρχει ένα αντικείμενο.

Η δήλωση:

 Fraction rationals[20];

δηλώνει 20 αντικείμενα Fraction:

 rationals[0], rationals[1], ..., rationals[19]

Αρχικοποίηση πινάκων αντικειμένων

- Παρόμοια με τη δήλωση ενός πίνακα ακεραίων.
 - Δεν χρειάζεται να κάνουμε τίποτα για να χρησιμοποιηθεί ο προκαθορισμένος κατασκευαστής.

```
int x;
Fraction num;
Fraction nums[4];
```
 - Για να αρχικοποιήσουμε με συγκεκριμένο τρόπο, καλούμε ρητά τον κατάλληλο κατασκευαστή.

```
int x(10);
Fraction f(10,20);
```
 - Πώς γίνεται να αρχικοποιήσουμε έναν πίνακα αντικειμένων; Χρειαζόμαστε έναν τρόπο έτσι ώστε να καθοριστούν διαφορετικοί κατασκευαστές για διαφορετικά αντικείμενα.

```
Fraction numlist[3]={Fraction(2,4), Fraction(5), Fraction()};
```

 - το numlist[0] αρχικοποιείται με τον κατασκευαστή Fraction(2,4)
 - το numlist[1] αρχικοποιείται με τον κατασκευαστή Fraction(5)
 - το numlist[2] αρχικοποιείται με τον κατασκευαστή Fraction()

Χρήση πινάκων αντικειμένων

- Η δεικτοδότηση λειτουργεί με τον ίδιο τρόπο όπως και στους πίνακες πρωτογενών τύπων δεδομένων.
 - Η αναφορά σε κάθε αντικείμενο του πίνακα γίνεται ως `arrayName[index]`;
- Ο τελεστής τελεία λειτουργεί με τον ίδιο τρόπο όπως και με τα απλά ονόματα αντικειμένων.
ονομαΑντικειμένου.όνομαΜέλους
- Το ονομαΑντικειμένου λαμβάνει τη μορφή ενός στοιχείου του πίνακα:
ονομαΠίνακα[δείκτης].όνομαΜέλους
- Παράδειγμα:

```
Fraction rationals[20];  
...  
rationals[2].Show();  
rationals[6].Input();  
for(int i=0;i<20;i++)  
    rationals[i].SetVal(1,2);
```

Άσκηση #1: Εκφώνηση

- Να γραφεί κλάση Student με πεδία name, age και grade, η οποία να διαθέτει προεπιλεγμένο κατασκευαστή, κατασκευαστή με δύο ορίσματα (name, age) και κατασκευαστή με τρία ορίσματα (name, age, grade), καθώς και υπερφόρτωση του τελεστή <<.
- Να δημιουργηθεί πίνακας από τρία αντικείμενα τύπου Student, τα οποία θα αρχικοποιούνται με διαφορετικό κατασκευαστή το καθένα, και να εμφανιστούν με τον τελεστή << διασχίζοντας τον πίνακα.

Άσκηση #1: Λύση

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

class Student {
private:
    string name;
    int age;
    double grade;

public:
    Student() : name("Unknown"), age(18), grade(0.0) {}
    Student(const string& n, int a) : name(n), age(a), grade(0.0) {}
    Student(const string& n, int a, double g) : name(n), age(a), grade(g) {}

    friend ostream& operator<<(ostream& os, const Student& s) {
        os << "Name: " << s.name << ", Age: " << s.age << ", Grade: " << s.grade;
        return os;
    }
};

int main() {
    Student students[3] = {Student(), Student("Nikos", 20),
                           Student("Maria", 22, 8.7)};
    cout << "Students information:\n";
    for (int i = 0; i < 3; i++) cout << students[i] << endl;
    return 0;
}
```

<https://github.com/chgogos/oop/blob/master/Various/COP3330/lect8/exercise1.cpp>

Students information:
Name: Unknown, Age: 18, Grade: 0
Name: Nikos, Age: 20, Grade: 0
Name: Maria, Age: 22, Grade: 8.7

Διανύσματα (std::vector)

- Το std::vector είναι ένας δυναμικός πίνακας ομοειδών στοιχείων που ορίζεται στην STL (Standard Template Library)
- Αλλάζει μέγεθος αυτόματα καθώς προστίθενται ή αφαιρούνται στοιχεία
- Η δήλωση και αρχικοποίηση ενός std::vector γίνεται ως εξής:
`std::vector<int> v {1, 2, 3}; // uniform αρχικοποίηση`
`std::vector<int> u={1, 2, 3}; // εναλλακτικός τρόπος αρχικοποίησης`
- Η πρόσβαση και η τροποποίηση στοιχείων γίνεται με: [], .at(), .front(), .back()
- Η προσθήκη και η διαγραφή στοιχείων γίνεται με: push_back(), pop_back(), insert(), erase(), clear()
- Το πλήθος των στοιχείων λαμβάνεται με το .size()

Διάσχιση ενός vector (1/2)

Με τον κλασσικό τρόπο

```
vector<int> v{1, 2, 3};  
for (int i = 0; i < v.size(); i++) {  
    cout << v[i] << " "  
}
```

1 2 3

<https://github.com/chgogos/oop/blob/master/various/COP3330/lect8/vector1.cpp>

Με range based for

```
vector<int> v{1, 2, 3};  
for (int x : v) {  
    cout << x << " "  
}
```

1 2 3

Διάσχιση ενός vector (2/2)

Με iterator

```
vector<int> v{1, 2, 3};  
for (auto it = v.begin(); it != v.end(); ++it) {  
    cout << *it << " ";  
}
```

1 2 3

<https://github.com/chgogos/oop/blob/master/various/COP3330/lect8/vector1.cpp>

Με reverse iterator

```
vector<int> v{1, 2, 3};  
for (auto rit = v.rbegin(); rit != v.rend(); ++rit) {  
    cout << *rit << " ";  
}
```

3 2 1

Ταξινόμηση με την std::sort

- Η std::sort βρίσκεται στο `<algorithm>`
- Λειτουργεί σε συνδυασμό με τους iterators `begin()`, `end()`
- Ο προκαθορισμένος τρόπος ταξινόμησης είναι σε αύξουσα σειρά

<https://github.com/chgogos/oop/blob/master/variables/COP3330/lect8/sort1.cpp>

```
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

int main() {
    std::vector<int> v{3, 1, 4, 2};
    std::sort(v.begin(), v.end());
    for (int x : v) {
        cout << x << " ";
    }
    cout << endl;
}
```

1 2 3 4

Ταξινόμηση με την std::sort σε φθίνουσα σειρά

- Για να κάνει ταξινόμηση σε φθίνουσα σειρά μπορεί να χρησιμοποιηθεί το function object:

std::greater<>

<https://github.com/chgogos/oop/blob/master/various/COP3330/lect8/sort2.cpp>

```
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

int main() {
    std::vector<int> v{3, 1, 4, 2};
    std::sort(v.begin(), v.end(), std::greater<>());
    for (int x : v) {
        cout << x << " ";
    }
    cout << endl;
}
```

4 3 2 1

Άσκηση #2: Εκφώνηση

- Δημιουργήστε ένα πρόγραμμα που διαχειρίζεται μια λίστα βιβλίων, όπου κάθε βιβλίο έχει τίτλο (title), και αριθμό σελίδων (pages). Αποθηκεύστε τα βιβλία σε ένα `std::vector<Book>` με πέντε στοιχεία.
- Ορίστε μέσα στην κλάση `Book` τους τελεστές σύγκρισης `operator<` και `operator>` με βάση τον αριθμό σελίδων, ώστε να μπορεί να γίνει ταξινόμηση των βιβλίων πρώτα σε αύξουσα και μετά σε φθίνουσα σειρά χρησιμοποιώντας τη συνάρτηση `std::sort`. Εκτυπώστε τη λίστα βιβλίων μετά από κάθε ταξινόμηση για να φαίνεται η σειρά τους.

Άσκηση #2: Λύση

```
class Book {           https://github.com/chgogos/oop/blob/master/variou...  
private:               https://github.com/chgogos/oop/blob/master/variou... (εναλλακτική λύση με std::rel_ops)  
...  
    bool operator<(const Book& other) const { return pages < other.pages; }  
    bool operator>(const Book& other) const { return pages > other.pages; }  
    friend ostream& operator<<(ostream& os, const Book& other) {  
        os << other.title << " " << other.pages;  
        return os;  
    }  
};  
  
int main() {  
    vector<Book> library = {Book("C++ Basics", 250),  
                             Book("Algorithms", 500),  
                             Book("Data Structures", 300),  
                             Book("Machine Learning", 450),  
                             Book("Databases", 350)};  
  
    cout << "Original list:\n";  
    for (const auto& book : library) cout << book << endl;  
    sort(library.begin(), library.end());  
    cout << "\nSorted by pages (ascending):\n";  
    for (const auto& book : library) cout << book << endl;  
    sort(library.begin(), library.end(), greater<Book>());  
    cout << "\nSorted by pages (descending):\n";  
    for (const auto& book : library) cout << book << endl;  
    return 0;  
}
```

<https://github.com/chgogos/oop/blob/master/variou...>

<https://github.com/chgogos/oop/blob/master/variou...> (εναλλακτική λύση με std::rel_ops)

Sorted by pages (ascending):

C++ Basics 250

Data Structures 300

Databases 350

Machine Learning 450

Algorithms 500

Sorted by pages (descending):

Algorithms 500

Machine Learning 450

Databases 350

Data Structures 300

C++ Basics 250

Αναφορές

- <http://www.cs.fsu.edu/~xyuan/cop3330/>