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

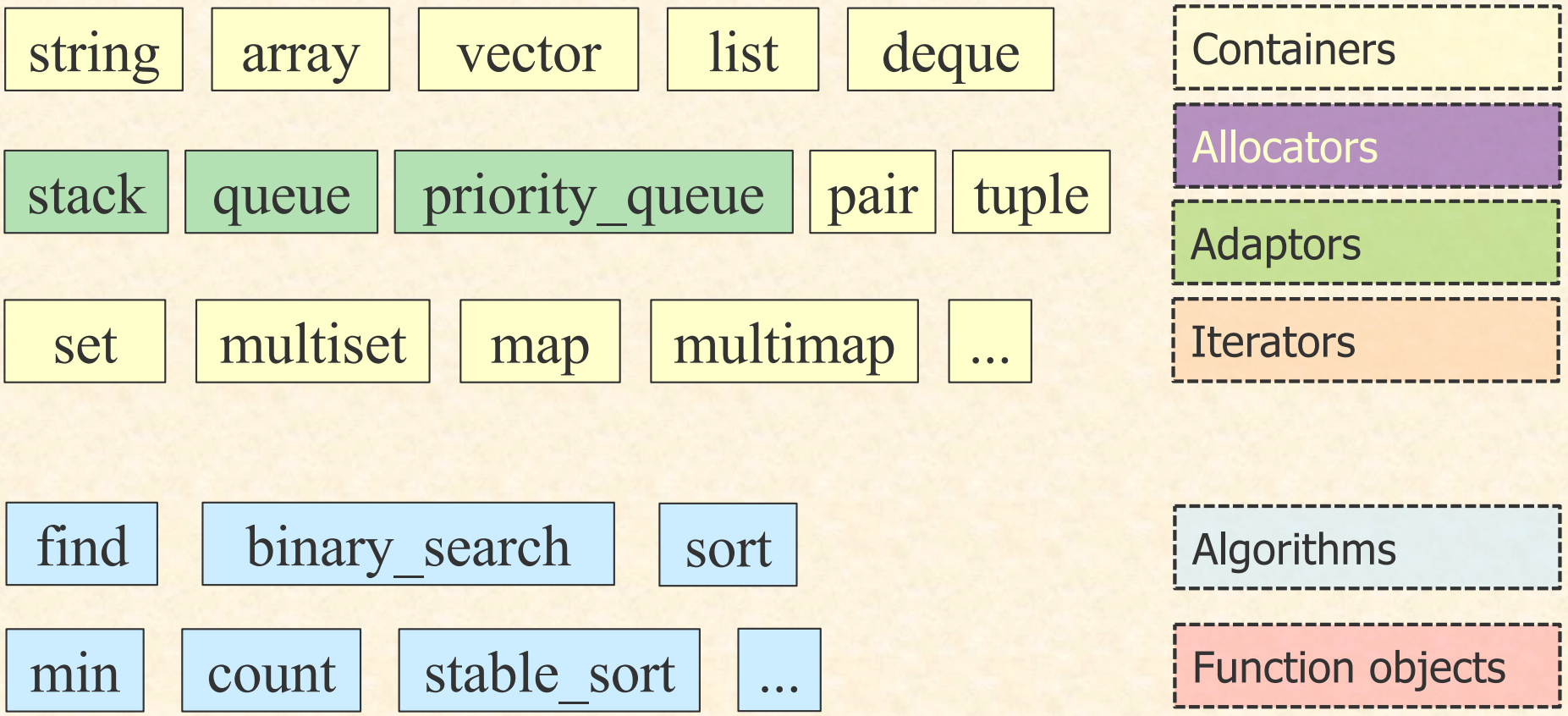
progtech@courses.softlab.ntua.gr

28/3/2025

Διαφάνειες παρουσιάσεων

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

Εισαγωγή στην STL



Εισαγωγή στην STL

www.cplusplus.com/reference/stl/

Container class templates

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)

Χρήσιμοι αλγόριθμοι στην STL

- Αναζήτηση
- Ταξινόμηση
- Λεξικογραφική διάταξη
- Γεννήτρια μεταθέσεων
- κ.λπ.

```
#include <algorithm>
```

Αναζήτηση (i)

- **`find(first, last, value)`**
  - σε χρόνο $O(n)$

- **`binary_search(first, last, value)`**
- **`binary_search(first, last, value, compare)`**
  - σε χρόνο $O(\log n)$

συνάρτηση που δέχεται ως παραμέτρους 2 αναφορές / αντικείμενα α, β του ίδιου τύπου και επιστρέφει true αν α "<" β

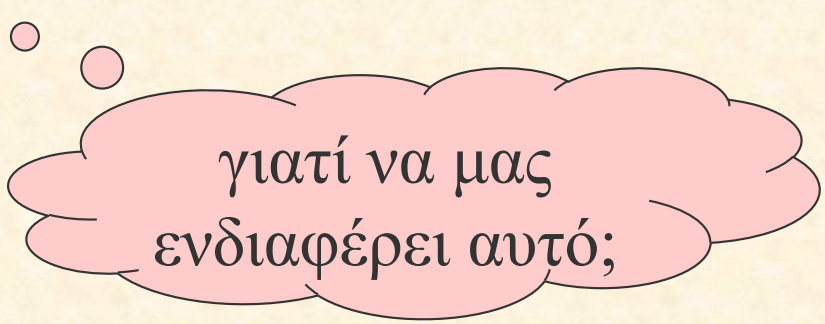
Αναζήτηση (ii)

```
vector<int> v;
...
vector<int>::iterator i =
  find(v.begin(), v.end(), 42);
if (i == v.end())
  cout << "not found" << endl;

// Αν οι αριθμοί είναι ταξινομημένοι:
vector<int>::iterator i =
  binary_search(v.begin(), v.end(), 42);
if (i != v.end())
  *i = 17;              // Άλλαξε το 42 σε 17
```

Ταξινόμηση (i)

- `sort(first, last)`
- `sort(first, last, compare)`
  - σε χρόνο $O(n \log n)$ στη μέση περίπτωση
- `stable_sort(first, last)`
- `stable_sort(first, last, compare)`
  - σε χρόνο $O(n \log n)$ στη χειρότερη περίπτωση
  - διατηρεί τη σειρά ίσων στοιχείων



Ταξινόμηση (ii)

```
vector<int> v;
...
// Σε αύξουσα σειρά:
sort(v.begin(), v.end());

// Σε φθίνουσα σειρά:
sort(v.begin(), v.end(), greater<int>());
```

- Το `greater<int>()` κατασκευάζει ένα **function object**. Χρειάζεται:

```
#include <functional>
```

Ταξινόμηση (iii)

```
bool my_less_than(int x, int y) {
  return abs(x) < abs(y);
}

struct my_comparator {
  bool operator()(int x, int y) {
    return abs(x) < abs(y);
  }
};

vector<int> v;
...
// Σε αύξουσα σειρά κατ' απόλυτο τιμή, εναλλακτικά:
sort(v.begin(), v.end(), my_less_than);
sort(v.begin(), v.end(), my_comparator());
```

Ταξινόμηση (iv)

```
// plain function: δέχεται ως παραμέτρους (α, β ) 2 αναφορές / αντικείμενα του
    ίδιου τύπου και επιστρέφει true αν α "<" β
bool my_less_than(int x, int y) {
  return abs(x) < abs(y);
}

// function object (functor): έχει κατάσταση (πεδία)
// πρέπει να κάνει overload το ()
struct my_comparator2 {
  bool byabs;
  my_comparator2(): byabs(false){}
  my_comparator2(bool aa): byabs(aa) {}
  bool operator()(int x, int y) {
    if (byabs)return abs(x) < abs(y);
    return x < y;
  }
};
sort(v.begin(),v.end(), my_comparator2(true));
```

Ταξινόμηση (v)

- **`partial_sort(first, middle, last)`**
- **`partial_sort(first, middle, last, compare)`**
  - σε χρόνο $O(n \log m)$ στη χειρότερη περίπτωση (n = last–first, m = middle–first)
  - ταξινομεί μόνο τα m πρώτα στοιχεία

```
vector<int> v;
...
partial_sort(v.begin(), v.begin() + 10,
             v.end());
```

Ταξινόμηση (vi)

- **`nth_element(first, nth, last)`**
- **`nth_element(first, nth, last, compare)`**
  - σε χρόνο $O(n)$ στη μέση περίπτωση
  - το ζητούμενο στοιχείο θα είναι στη θέση του
  - τα προηγούμενά του θα είναι μικρότερα, τα επόμενά του θα είναι μεγαλύτερα

```
vector<int> v;
...
nth_element(v.begin(), v.begin() + 10,
            v.end());
```

Εύρεση μικρότερου/μεγαλύτερου

- `min_element(first, last)`
- `min_element(first, last, compare)`
- `max_element(first, last)`
- `max_element(first, last, compare)`
  - σε χρόνο $O(n)$

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

progtech@courses.softlab.ntua.gr

28/3/2025

Διαφάνειες παρουσιάσεων

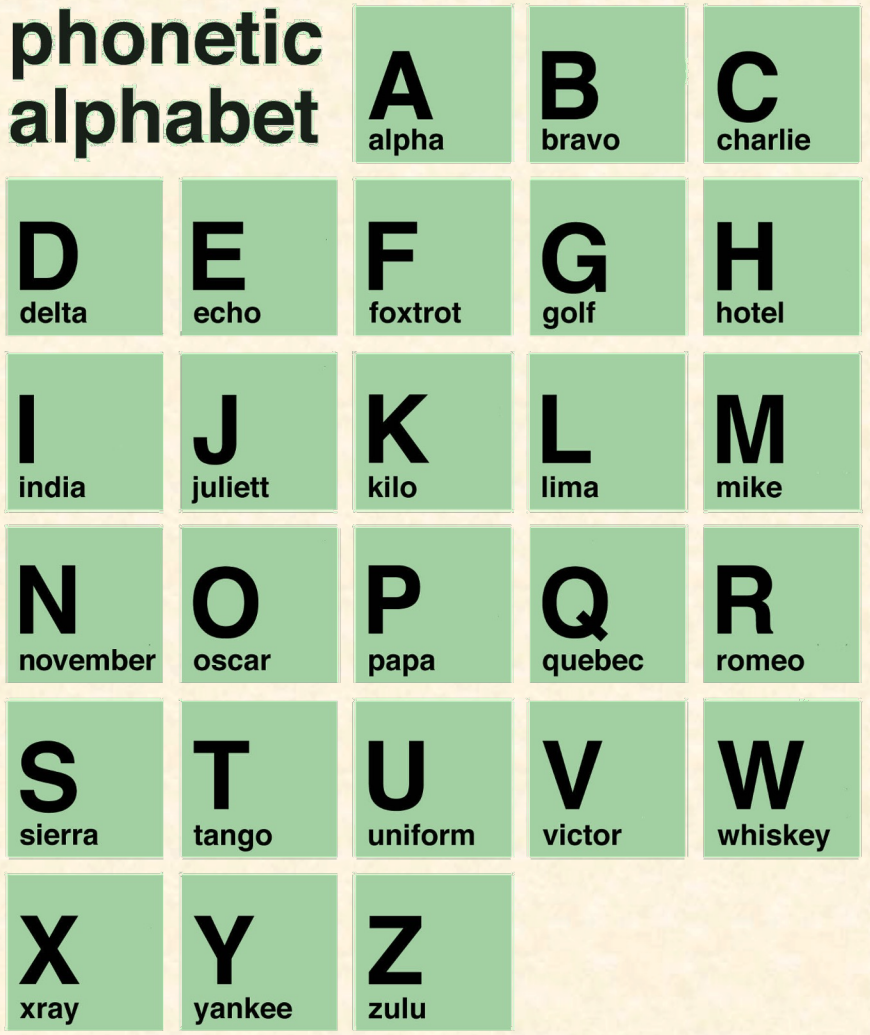
- ✓ Η γλώσσα προγραμματισμού C++
- ✓ Αρχές αντικειμενοστρεφούς προγραμματισμού
- ✓ Δομές δεδομένων

sierra tango lima echo x-ray alpha mike papa lima echo

Παραδείγματα STL

Παράδειγμα STL #1

- Μετατροπή από/προς το φωνητικό αλφάβητο



Παράδειγμα STL #1 - maps

- Υλοποίηση με map: functions
  - string char2phonetic
  - string string2phonetic
  - char phonetic2char
  - string phonetic2word (memoization, too)

```
string char2phonetic(char)
```
'a' → "alpha",
'b' → "bravo"

```
string string2phonetic(string)
```
"ece" →
"echo charlie echo"

```
char phonetic2char(string)
```
"alpha" → 'a',
"beta" → 'b'

```
string phonetic2word(string)
```
"echo charlie echo" →
"ece"

Παράδειγμα STL #1 - maps

```
char char2phonetic (const char &c) {
  static const string phoneticWords[] = {
    "alpha", "bravo", "charlie", "delta", "echo", "foxtrot",
    "golf", "hotel", "india", "juliett", "kilo", "lima",
    "mike", "november", "oscar", "papa", "quebec", "romeo",
    "sierra", "tango", "uniform", "victor", "whiskey", "x-ray",
    "yankee", "zulu", " ", "@", "."
  };
  static map<char, string> phonetic;
  // Setup the mapping only once!
  if (phonetic.empty()) {
    for (const string &w : phoneticWords)
      phonetic[w.front()] = w;
  }
  // Do the job!
  if (phonetic.find(c) == phonetic.end()) return "?*?*?";
  return phonetic[c];
}
```

Παράδειγμα STL #1 - maps

```
string string2phonetic(const string &ss) {
  string result = "";
  for (char cc : ss) result += char2phonetic(cc) + " ";
  return result.substr(0, result.length()-1);
}

// Use memoization, if you expect to see the same word many times!
string string2phonetic_better(const string &ss) {
  static map<string, string> cachedWords;

  // search for known word
  map<string, string>::iterator cw = cachedWords.find(ss);
  if (cw != cachedWords.end()) return cw->second;

  // word not known
  string result = "";
  for (char cc : ss) result += char2phonetic(cc) + " ";
  cachedWords[ss] = result.substr(0, result.length()-1);
  return result;
}
```

Παράδειγμα STL #1 - maps

```
char phonetic2char(const string &searchString) {
  static const string phoneticWords[] = {
    "alpha", "bravo", "charlie", "delta", "echo", "foxtrot",
    "golf", "hotel", "india", "juliett", "kilo", "lima",
    "mike", "november", "oscar", "papa", "quebec", "romeo",
    "sierra", "tango", "uniform", "victor", "whiskey", "x-ray",
    "yankee", "zulu", " ", "@", "."
  };
  static map<string, char> invPhonetic;
  // Setup the mapping only once!
  if (invPhonetic.empty()) {
    for (const string &w : phoneticWords)
      invPhonetic[w] = w.front();
  }
  // Do the job!
  map<string, char>::iterator i = invPhonetic.find(searchString);
  if (i == invPhonetic.end()) return '?';
  return i->second;
}
```

Παράδειγμα STL #1 - maps

```
string phonetic2word(const string &src) {
  string result = "";
  string oneWord;
  istringstream stringStream(src);
  while (stringStream >> oneWord)
    result += phonetic2char(oneWord);
  return result;
}
```

DONE!

`string char2phonetic(char)`	'a' → "alpha", 'b' → "bravo"
`string string2phonetic(string)`	"ece" → "echo charlie echo"
`char phonetic2char(string)`	"alpha" → 'a', "beta" → 'b'
`string phonetic2word(string)`	"echo charlie echo" → "ece"

Υλοποίηση χωρίς κλάσεις

```
string originalText = "";
string oneWord, originalWord;

while(inFile >> inWord) {
  string p = string2phonetic (inWord);
  cout << p << " " << endl;

  originalWord = phonetic2word(p);
  originalText += originalWord + " / ";
}
```

Παράδειγμα STL #1 – maps με κλάσεις

- Σχεδίαση κλάσης **readable**

```
class readable {
private:
   string normalText, string phoneticText;
   map<char, string> phoneticMap;
   map<string, char> invPhoneticMap;
   string phoneticWords[29];
   string char2phonetic(const char &c) ;           // direct
   string text2phonetic(const string &txt) ;
   char phonetic2char(const string &searchString); // reverse
   string phonetic2text(const string &t) ;
public:
   readable();
   readable(const string & sometext);
   string phonetic();     // getter
   string text();    // getter
   void setPhonetic(const string & pText); // setter
   void setText(const string & nText);     // setter
};
```

Παράδειγμα STL #1 – maps με κλάσεις

- Σχεδίαση κλάσης **`readable`** (better!)

```
class readable {
private:
  string normalText; string phoneticText;
  static map<char, string> phoneticMap;
  static map<string, char> invPhoneticMap;
  static const string phoneticWords[29];
  static void initializeMappings();
  static string char2phonetic(const char &c);              // direct
  static string text2phonetic(const string &txt);
  static char phonetic2char(const string &searchString); // reverse
  static string phonetic2text(const string &t);
public:
  readable();
  readable(const string &sometext);
  string phonetic() const;                    // getter
  string text() const;                        // getter
  void setPhonetic(const string &pText);      // setter
  void setText(const string &nText);          // setter
};
```

Αφαίρεση και δομές δεδομένων

- Οι λεπτομέρειες υλοποίησης επηρεάζουν την πολυπλοκότητα των αλγορίθμων και τη χρήση πόρων γενικά
- Η χρήση ΑΤΔ ελαχιστοποιεί τις αλλαγές που απαιτούνται σε ένα πρόγραμμα που χρησιμοποιεί έναν ΑΤΔ, όταν αλλάξει ο τρόπος υλοποίησης

Παράδειγμα STL #1 – maps με κλάσεις

```
const string readable::phoneticWords[29] = {
        "alpha", "bravo", "charlie", "delta", "echo", "foxtrot",
        "golf", "hotel", "india", "juliett", "kilo", "lima",
        "mike", "november", "oscar", "papa", "quebec", "romeo",
        "sierra", "tango", "uniform", "victor", "whiskey", "x-ray",
        "yankee", "zulu", " ", "@", "."};
readable::readable() : normalText(""), phoneticText("") {
  initializeMappings();
}
readable::readable(const string &sometext) : normalText(sometext) {
  initializeMappings();
  phoneticText = text2phonetic(sometext);
}
void initializeMappings() {
  // Initialize the mappings only once!
  if (!phoneticMap.empty()) return;
  for (const string &word : phoneticWords) {
    phoneticMap[word.front()] = word;
    invPhoneticMap[word] = word.front();
  }
}
```

Παράδειγμα STL #1 – maps με κλάσεις

```
string readable::char2phonetic(const char &c) {
  if (phoneticMap.find(c) == phoneticMap.end()) return "?";
  return phoneticMap[c];
}

string readable::text2phonetic(const string &txt) {
  string result = "";
  for (char c : txt) result += char2phonetic(c) + " ";
  return result.substr(0, result.length()-1);
}

char readable::phonetic2char(const string &searchString) {
  auto ipi = invPhoneticMap.find(searchString);
  return ipi == invPhoneticMap.end()? '?' : ipi->second;
}

string readable::phonetic2text(const string &t) {
  string w = "", result = "";
  istringstream stringStream(t);
  while (stringStream >> w) result += phonetic2char(w);
  return result;
}
```

Παράδειγμα STL #1 – maps με κλάσεις

```
string readable::phonetic() { return phoneticText; }
string readable::text()     { return normalText; }

void readable::setPhonetic(const string & pText) {
  phoneticText = pText;
  normalText   = phonetic2text(pText);
}

void readable::setText(const string &nText)  {
  normalText = nText;
  phoneticText = text2phonetic(nText);
}

// ...
readable myRR("treis laloun kai dyo chorevoun (paroimia)");
cout << myRR.text() << endl << myRR.phonetic();
```

```
treis laloun kai dyo chorevoun (paroimia)
tango romeo echo india sierra   lima alpha lima oscar uniform november   kilo alpha india
delta yankee oscar   charlie hotel oscar romeo echo victor oscar uniform november   ?*?*?
papa alpha romeo oscar india mike india alpha ?*?*?
```

Αφαίρεση και δομές δεδομένων

- Οι λεπτομέρειες υλοποίησης επηρεάζουν την πολυπλοκότητα των αλγορίθμων και τη χρήση πόρων γενικά
- Η χρήση ΑΤΔ ελαχιστοποιεί τις αλλαγές που απαιτούνται σε ένα πρόγραμμα που χρησιμοποιεί έναν ΑΤΔ, όταν αλλάξει ο τρόπος υλοποίησης

Ιδέα:

Δοκιμάστε να αλλάξετε το **private** τμήμα της κλάσης **readable** χωρίς να αλλάξει τίποτε στο **public**

Παράδειγμα #2: maps & sets

Διαγωνισμοί προμηθειών

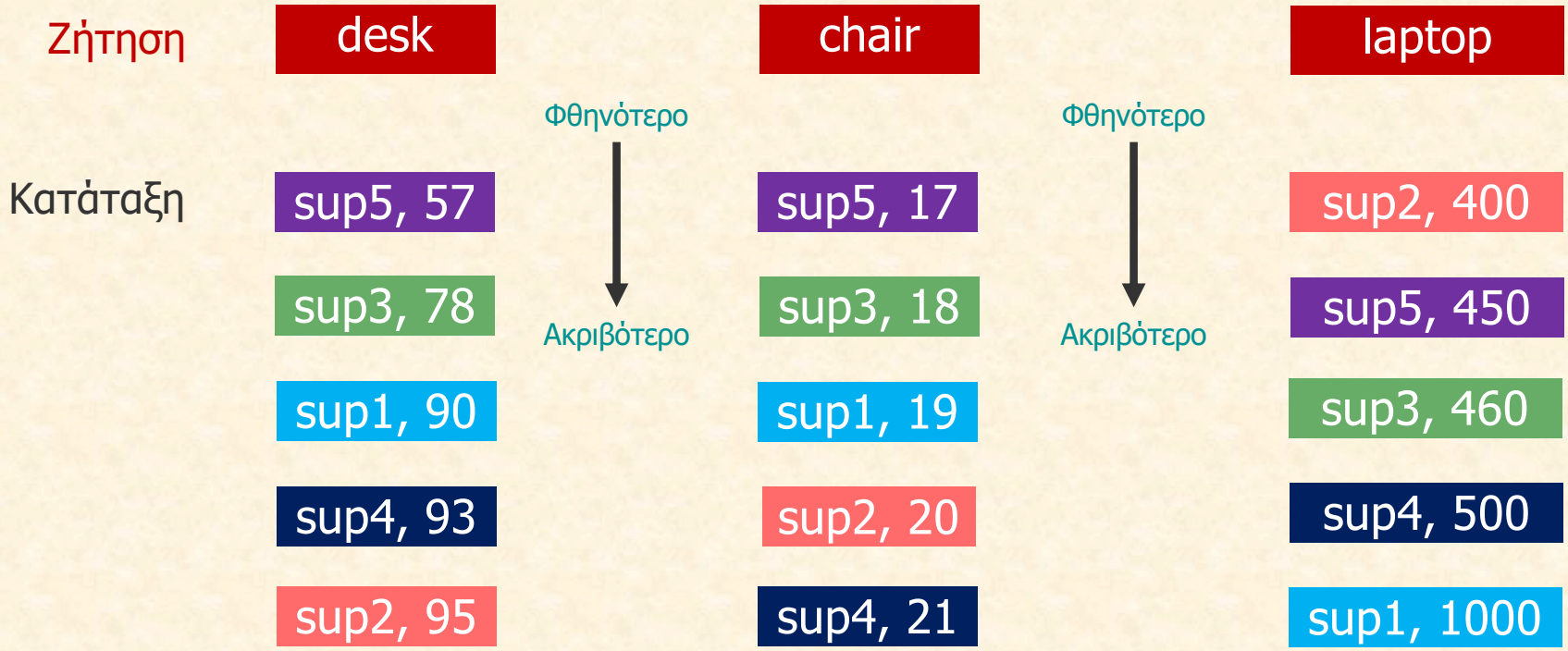
- Αρχές λειτουργίας
  - Ζήτηση προϊόντων
  - Προσφορές από προμηθευτές (αγορά)
  - Επιλογή προμηθευτή
    - Για κάθε προϊόν επιλέγουμε αυτόν που προσφέρει τη μικρότερη τιμή αγοράς

Παράδειγμα #2: maps & sets

Διαγωνισμοί προμηθειών

Ζήτηση	desk	chair	laptop
Προσφορά	sup1, 90	sup1, 19	sup1, 1000
	sup2, 95	sup2, 20	sup2, 400
	sup3, 78	sup3, 18	sup3, 460
	sup4, 93	sup4, 21	sup4, 500
	sup5, 57	sup5, 17	sup5, 450

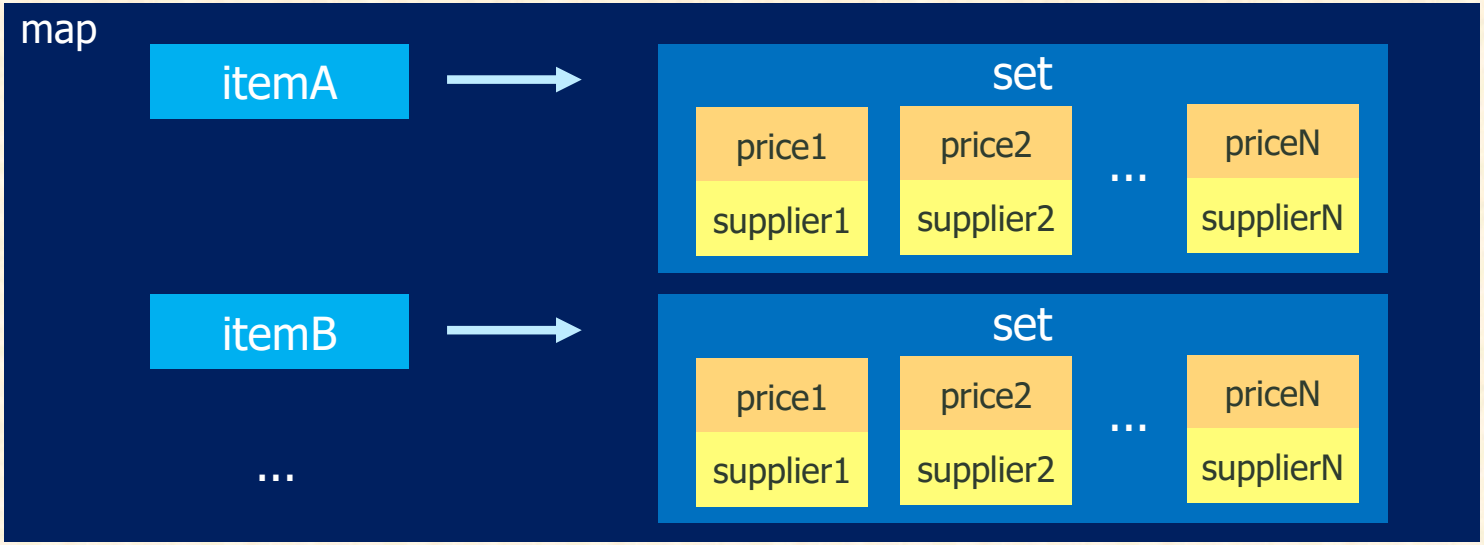
Παράδειγμα #2: maps & sets



Συζήτηση: υλοποίηση με STL maps & sets

- Αντιστοίχιση: Είδος → Προσφορές
  - map
  - string
  - set (offers)
  - Περιεχόμενο προσφορών
    - Τιμή μονάδας
    - Προμηθευτής

    struct { double price; string supplier }
  - Ταξινόμηση προσφορών: set, με κατάλληλο operator<
    - Τιμή μονάδας

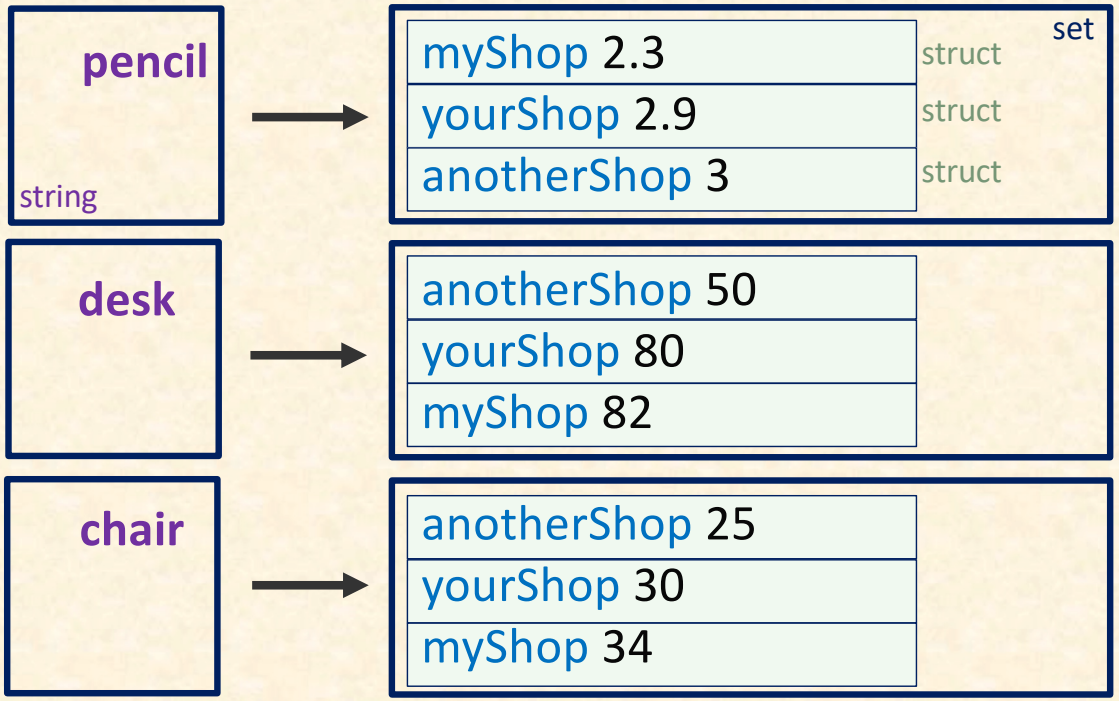


Συζήτηση: υλοποίηση με STL maps & sets

INPUT

myShop **pencil** 2.3
myShop **desk** 82
myShop **chair** 34
myShop **paper** 2.7
myShop **laptop** 400
yourShop **pencil** 2.9
yourShop **desk** 80
yourShop **chair** 30
yourShop **paper** 3.1
yourShop **laptop** 460
anotherShop **pencil** 3
anotherShop **desk** 50
anotherShop **chair** 25
anotherShop **paper** 3.8
anotherShop **laptop** 550

STL STRUCTURES

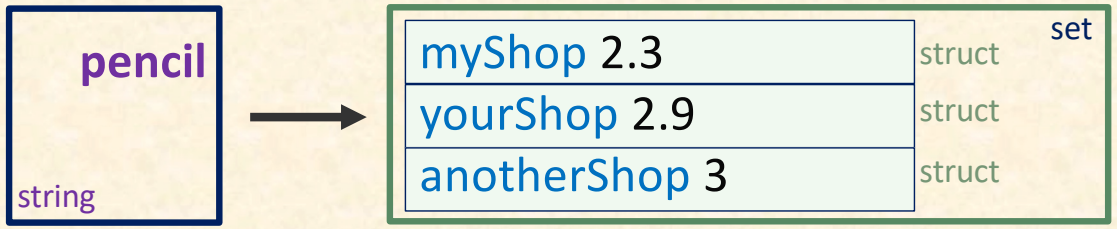


Συζήτηση: υλοποίηση με STL maps & sets

INPUT

myShop **pencil** 2.3
myShop **desk** 82
myShop **chair** 34
myShop **paper** 2.7
myShop **laptop** 400
yourShop **pencil** 2.9
yourShop **desk** 80
yourShop **chair** 30
yourShop **paper** 3.1
yourShop **laptop** 460
anotherShop **pencil** 3
anotherShop **desk** 50
anotherShop **chair** 25
anotherShop **paper** 3.8
anotherShop **laptop** 550

STL STRUCTURES




```
struct offer {
  string supplier;
  double price;
};
```

```
map<string, set<offer>>
```

```
typedef map<string, set<offer>> bids;
```

Συζήτηση: υλοποίηση με STL maps & sets

```
struct offer {
  string supplier;
  double price;
};
typedef map<string, set<offer>> bids;
bool operator< (const offer &o1, const offer &o2) {
  if (o1.price != o2.price) return o1.price < o2.price;
  return o1.supplier < o2.supplier;
}
int main() {
  ifstream input("sample_offers.txt");
  string sup, itm;
  double prc;
  bids receivedOffers = {};
  while (input >> sup >> itm >> prc) {
    offer currentOffer = {sup, prc};
    receivedOffers[itm].insert(currentOffer);
  }
  input.close();
  for (bids::iterator iitem=receivedOffers.begin(); iitem!=receivedOffers.end(); ++iitem) {
    cout << iitem->first << ":" << endl;
    for (set<offer>::iterator oo = iitem->second.begin(); oo!=iitem->second.end(); ++oo)
      cout << "    " << oo->supplier << " " << oo->price << endl;
  }
}
```

string → set

empty map

Εύκολη διάτρεξη: auto

```
struct data {
   int number;
   string s;
};

void set_demo() {
     set<data> mySet;
     . . .

     for (auto const &item: mySet)
       cout << item.number << " " << item.s << endl;
}
```

set<data>:
Ο τύπος των δεδομένων του container που διατρέχουμε

mySet:
Ο container που διατρέχουμε

auto:
Ορίζει αυτόματα τον τύπο της μεταβλητής με την οποία διατρέχουμε τον container

item:
Μεταβλητή που παίρνει τιμές από τα περιεχόμενα του container που διατρέχουμε

Με τον τρόπο αυτό διατρέχουμε ολόκληρο τον container

Συζήτηση: υλοποίηση με STL maps & sets

```
struct offer {
  string supplier;
  double price;
};

typedef map<string, set<offer>> bids;

bool operator< (const offer &o1, const offer &o2) {
  if (o1.price != o2.price) return o1.price < o2.price;
  return o1.supplier < o2.supplier;
}

int main() {
  ifstream input("sample_offers.txt");
  string sup, itm;
  double prc;
  bids receivedOffers = {};
  while (input >> sup >> itm >> prc) {
    offer currentOffer = {sup, prc};
    receivedOffers[itm].insert(currentOffer);
  }
  input.close();
  for (const auto &iitem : receivedOffers) {
    cout << iitem.first << ":" << endl;
    for (const auto &oo : iitem.second)
      cout << "    " << oo.supplier << " " << oo.price << endl;
  }
}
```

auto:
Απλούστερη διάτρεξη με auto

Ιδέα για περεταίρω εξάσκηση

Αγορά ηλεκτρικής ενέργειας

Η αγορά ηλεκτρικής ενέργειας λειτουργεί με δημοπρασίες στις οποίες συμμετέχουν παραγωγοί ενέργειας που για κάθε χρονική περίοδο (πχ 1h) προσφέρουν μια ποσότητα ενέργειας σε κάποια τιμή. Υποθέτουμε ότι ο διαχειριστής του συστήματος λειτουργεί ως εξής: ταξινομεί τις προσφορές σε αύξουσα τάξη και για κάθε ώρα επιλέγει όσες απαιτείται προκειμένου να καλυφθεί η ζήτηση με το ελάχιστο κόστος.

Για παράδειγμα, έστω ότι η ζήτηση που πρέπει να καλυφθεί την ώρα που ξεκινά στις 2022-03-20-00:00 είναι **5752 MWh**, ενώ οι προσφορές που έχουν υποβληθεί έχουν ως ακολούθως:

- 2022-03-20-00:00 1000 MWh με κόστος 10 €
- 2022-03-20-00:00 4000 MWh με κόστος 12 €
- 2022-03-20-00:00 3000 MWh με κόστος 9 €

Ιδέα για περεταίρω εξάσκηση

Ταξινόμηση προσφορών σε αύξουσα τάξη τιμής:

- 2022-03-20-00:00 3000 MWh με κόστος 9 €
- 2022-03-20-00:00 1000 MWh με κόστος 10 €
- 2022-03-20-00:00 4000 MWh με κόστος 12 €

Κάλυψη ζήτησης **5752 MWh**

- 3000 MWh από την φθηνότερη με κόστος 3000*9 = 27000
- 1000 MWh από την επόμενη με κόστος 1000*10 = 10000
- 1752 MWh από την επόμενη με κόστος 1752*12 = 21024
- Σύνολο 58024 €

Να γράψετε πρόγραμμα C++ που διαβάζει από την είσοδο τη ζήτηση και τις προσφορές και υπολογίζει το κόστος κάλυψης της ζήτησης κάθε περιόδου. Αν η συνολική προσφερόμενη ενέργεια δεν επαρκεί για να καλύψει τη ζήτηση, το πρόγραμμα να εμφανίζει "IMPOSSIBLE".

Ιδέα για περεταίρω εξάσκηση

ΠΑΡΑΔΕΙΓΜΑ

ΕΙΣΟΔΟΣ

```
3
2022-03-20-04:00    4749
2022-03-20-00:00    5752
2022-03-20-23:00    6079
7
2022-03-20-00:00    1000    10
2022-03-20-04:00    2000    12
2022-03-20-04:00    2000    9
2022-03-20-00:00    4000    12
2022-03-20-00:00    3000    9
2022-03-20-04:00    4000    11
2022-03-20-04:00    2000    10
```

ΕΞΟΔΟΣ

```
2022-03-20-00:00 58024
2022-03-20-04:00 46239
2022-03-20-23:00 IMPOSSIBLE
```

Ιδέα για περεταίρω εξάσκηση