

# Η ΓΛΩΣΣΑ C++

## Μάθημα 9:

### Η κλάση string

Δημήτρης Ψούνης



www.psounis.gr

## Περιεχόμενα Μαθήματος

### A. Θεωρία

#### 1. Βασική λειτουργικότητα της κλάσης string

1. Εισαγωγή
2. Αρχικοποίηση
3. Διάβασμα (1. Υπερφόρτωση του >>)
4. Διάβασμα (2. getline())
5. Εκτύπωση
6. Παράδειγμα

#### 2. Υπερφορτωμένοι τελεστές

1. Τελεστές +, =, +=
2. Τελεστές [] και <, >, ==, !=
3. Παράδειγμα

#### 3. Μέγεθος Συμβολοσειράς

1. Μέθοδοι length() και size()
2. Χωρητικότητα και προσαρμογή πραγματικού μεγέθους
3. Παράδειγμα

#### 4. Τροποποίηση Συμβολοσειράς

1. Προσθήκη
2. Διαγραφή
3. Ενημέρωση
4. Παράδειγμα

#### 5. Αναζήτηση

4. Η μέθοδος find()
5. Οι μέθοδοι rfind(), find\_first\_of() και find\_last\_of
6. Παράδειγμα

### Ασκήσεις

## A. Θεωρία

### 1. Βασική λειτουργικότητα της κλάσης string

#### 1. Εισαγωγή

- Είδαμε στα προηγούμενα μαθήματα την (δική μας) κλάση STRING η οποία ήταν ένας περιτυλιχτής για τη συμβολοσειρά της C.
- Για καλή μας τύχη, έχει οριστεί μια τεράστια κλάση στην C++ και συγκεκριμένα **η κλάση string**:
  - Απαιτεί την ενσωμάτωση κεφαλίδας string:

```
#include <string>
```
  - και έχει οριστεί στον χώρο ονομάτων std
- Έχουν οριστεί δεκάδες μέθοδοι για πολλές πιθανές χρήσεις μιας συμβολοσειράς και θα δούμε αρκετές από αυτές στο μάθημα αυτό.
- Είναι σημαντικό ότι αυτή είναι η πρώτη ενσωματωμένη κλάση της C++ που μελετάμε.
  - Δεν μας ενδιαφέρουν οι λεπτομέρειες της υλοποίησής της (αν και μπορούμε να υποθέσουμε αρκετές από αυτές)
  - Αλλά μας ενδιαφέρει η λειτουργικότητα (μέθοδοι της κλάσης) που μας προσφέρονται

## A. Θεωρία

### 1. Βασική λειτουργικότητα της κλάσης string

#### 2. Αρχικοποίηση

- Η αρχικοποίηση ενός string, γίνεται με κάποιον από τους constructors που έχουν οριστεί:

- Πρώτος Constructor (default)

```
string()
```

- Δεν δέχεται ορίσματα, και αρχικοποιεί το αντικείμενο με μία συμβολοσειρά μήκους 0:

```
string s;
```

- Δεύτερος Constructor

```
string(const char *s)
```

- Δέχεται ως όρισμα μια σταθερή συμβολοσειρά, δηλαδή ένα string της C με διπλά εισαγωγικά, π.χ.:

```
string s("This is a string");
```

## A. Θεωρία

### 1. Βασική λειτουργικότητα της κλάσης string

#### 2. Αρχικοποίηση

- 3<sup>ος</sup> Constructor

```
string(size_type n, char c)
```

- Κατασκευάζει μία συμβολοσειρά μήκους n, όπου κάθε θέση έχει αρχικοποιηθεί με το χαρακτήρα c

```
string s(20, 'x');
```

- Copy Constructor

```
string(const string &s)
```

- Με την προφανή λειτουργικότητα ενός copy constructor.
- Π.χ.:

```
string s2(s1);
```

- ή

```
string s2=s1;
```

#### Παρατήρηση:

- Η κλάση επιλέγει τον κατάλληλο τύπο δεδομένων για το μήκος της συμβολοσειράς.
- Έτσι έχει οριστεί (μέσω typedef) ο τύπος ακεραίου που χρησιμοποιείται για το μήκος, και οτιδήποτε θέλουμε να μετράμε πρέπει να είναι τύπου δεδομένων: size\_type)

## A. Θεωρία

### 1. Βασική λειτουργικότητα της κλάσης string

#### 3. Διάβασμα (2. getline())

- Η φιλική συνάρτηση getline:

```
getline(stream, str);
```

- Διαβάζει μία συμβολοσειρά, μέχρι και το χαρακτήρα αλλαγής γραμμής (\n)
  - τον οποίο ΔΕΝ αποθηκεύει στη συμβολοσειρά

- Π.χ.:

```
getline(cin, str);
```

- Υπερφορτωμένη εκδοχή:

```
getline(stream, str, delim)
```

- Διαβάζει τη συμβολοσειρά μέχρι και το χαρακτήρα delim (αντικαθιστά το \n)
- Π.χ.:

```
getline(cin, str, '#');
```

## A. Θεωρία

### 1. Βασική λειτουργικότητα της κλάσης string

#### 3. Διάβασμα (1. Υπερφόρτωση του >>)

- Για το διάβασμα της συμβολοσειράς έχουμε δύο βασικούς τρόπους (>> και getline)

- Υπερφόρτωση του >>. Π.χ. :

```
stream>>str;
```

- Όπου str είναι ένα αντικείμενο τύπου string

- Π.χ.:

```
cin>>str;
```

- Προσοχή ότι το διάβασμα σταματάει είτε στο κενό είτε στο \n

- Συνεπώς π.χ. αν πληκτρολογήσουμε ως είσοδο το

```
This is a string
```

- Θα αποθηκευτεί στο str, μόνο το "This"

## A. Θεωρία

### 1. Βασική λειτουργικότητα της κλάσης string

#### 4. Εκτύπωση

- Για την εκτύπωση της συμβολοσειράς στην οθόνη χρησιμοποιούμε τον τελεστή <<

- Υπερφόρτωση του <<

```
stream<<str;
```

- Όπου str είναι ένα αντικείμενο τύπου string

- Π.χ.:

```
cout<<str;
```

## A. Θεωρία

### 1. Βασική λειτουργικότητα της κλάσης string

#### 5. Παράδειγμα

- Βλέπουμε και ένα παράδειγμα με τη βασική λειτουργικότητα που έχουμε δει:

```
/* cpp9.string.cpp */

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

int main()
{
    string s("Initial string");

    cout<<s<<endl;

    cout<<"Give new string: ";
    getline(cin, s);

    cout<<s;

    return 0;
}
```

## A. Θεωρία

### 2. Υπερφορτωμένοι τελεστές

#### 1. Τελεστές +, =, +=

- Υπερφόρτωση του +
  - Συνενώνει τις δύο συμβολοσειρές
  - Π.χ. αν str1 = "xx" και str2="yy", τότε η εντολή:
 

```
cout<<str1 + str2;
```

    - Θα εκτυπώσει: "xxyy"
- Υπερφόρτωση του =
  - Προφανής υπερφόρτωση του τελεστή ανάθεσης.
  - Π.χ.:
 

```
str1 = str2;
```
- Υπερφόρτωση του +=
  - Επαυξάνει μία συμβολοσειρά, π.χ. αν str1="xx" και str2="yy"
 

```
str1 += str2;
```

    - Νέα τιμή στο str1 = "xxyy"

#### Σημείωση:

- Δουλεύουν και με c strings:
- Παραδείγματα: str1+="yy"; str1+="xx"; str1 = "xx";

## A. Θεωρία

### 2. Υπερφορτωμένοι τελεστές

#### 2. Τελεστές [] και <, >, ==, !=

- Υπερφόρτωση του []
  - Δίνει πρόσβαση σε έναν χαρακτήρα της συμβολοσειράς
  - Π.χ. για να θέσουμε τον 3<sup>ο</sup> χαρακτήρα ίσο με 'c'
 

```
str[2]='c';
```
- Υπερφόρτωση του <
  - Επιστρέφει true/false ανάλογα με το αν το 1<sup>ο</sup> string είναι μικρότερο (αλφαβητικά) του 2<sup>ου</sup>
  - Π.χ.:
 

```
if (str1 < str2) ...;
```
- Αντίστοιχα έχουν υπερφορτωθεί και οι υπόλοιποι σχεσιακοί τελεστές:
  - >, <=, >=,
  - ==(ισα), !=(διαφορετικά)

## A. Θεωρία

### 2. Υπερφορτωμένοι τελεστές

#### 3. Παράδειγμα

- Βλέπουμε και ένα παράδειγμα χρήσης των (υπερφορτωμένων) τελεστών:

```
/* cpp9.string_operators.cpp */

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

int main()
{
    string s1("abcd");
    string s2;

    s2=s1+s1+s1;

    s2[4]='A';

    if (s1>s2)
        cout<<s1;
    else
        cout<<s2;

    return 0;
}
```



## A. Θεωρία

### 3. Μέγεθος Συμβολοσειρας

#### 1. Μέθοδοι length() και size()

- Οι μέθοδοι:
 

```
string::length()
```
- και
 

```
string::size()
```
- εκτελούν την ίδια ενέργεια: Μετράνε το μέγεθος (σε χαρακτήρες) της συμβολοσειράς
  - (Δεν μετριέται το \0)
- Π.χ. αν `str="xxxx"` τότε η κλήση:
 

```
str.length();
```

  - θα επιστρέψει 4



## A. Θεωρία

### 3. Μέγεθος Συμβολοσειρας

#### 2. Χωρητικότητα και προσαρμογή πραγματικού μεγέθους

- Προσοχή! Ο χώρος που έχει δεσμεύσει δυναμικά ένα αντικείμενο – συμβολοσειρά δεν είναι όσο το μήκος της συμβολοσειράς.
  - Αντίθετα η υλοποίηση δεσμεύει γενικά παραπάνω χώρο, ώστε να «προβλέπει» προσθέσεις χαρακτήρων και να μη χρειάζεται κάθε φορά να επαναδεσμεύει χώρο.
- Η μέθοδος:
 

```
string::capacity()
```

  - επιστρέφει το πλήθος των bytes που έχει πραγματικά δεσμεύσει η συμβολοσειρά (χωρητικότητα συμβολοσειράς).
- Η μέθοδος
 

```
string::shrink_to_fit()
```

  - Μειώνει τη χωρητικότητα της συμβολοσειράς ώστε να είναι ίση με το πραγματικό μέγεθος της συμβολοσειράς.
- Η μέθοδος
 

```
string::resize(size_t n)
```

  - Προσαρμόζει τη χωρητικότητα (είτε αυξανόμενη, είτε μειούμενη) σε μέγεθος `n` bytes



## A. Θεωρία

### 3. Μέγεθος Συμβολοσειρας

#### 3. Παράδειγμα

- Βλέπουμε και ένα παράδειγμα χρήσης περί της χωρητικότητας:

```
/* cpp9.string_capacity.cpp */
```

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

```
int main()
```

```
{
```

```
    string s1;
    string s2("Medium");
    string s3("A rather large one");
```

```
    cout<<"Capacity s1: "<<s1.capacity()<<endl;
    cout<<"Length s1:"<<s1.size()<<endl;
```

```
    cout<<"Capacity s2: "<<s2.capacity()<<endl;
    cout<<"Length s2:"<<s2.length()<<endl;
```

```
    cout<<"Capacity s3: "<<s3.capacity()<<endl;
    cout<<"Length s3:"<<s3.length()<<endl;
```

```
    s3+=s1+s2;
```

```
    cout<<"Capacity s3: "<<s3.capacity()<<endl;
    cout<<"Length s3:"<<s3.length()<<endl;
```

```
    s3.resize(10);
```

```
    cout<<"Capacity s3: "<<s3.capacity()<<endl;
    cout<<"Length s3:"<<s3.length()<<endl;
```

```
    s3.shrink_to_fit();
```

```
    cout<<"Capacity s3: "<<s3.capacity()<<endl;
    cout<<"Length s3:"<<s3.length()<<endl;
```

```
    return 0;
```



## A. Θεωρία

### 4. Τροποποίηση Συμβολοσειράς

#### 1. Προσθήκη

- Η μέθοδος:
 

```
string::append(X)
```

  - προσθέτει το `X` στο τέλος της συμβολοσειράς.
  - Το `X` μπορεί να είναι `string` ή πίνακας χαρακτήρων
- Η μέθοδος:
 

```
string::insert(size_t pos, X)
```

  - Εισάγει στη θέση `pos` το `X` (`string` ή πίνακας χαρακτήρων)
- Η μέθοδος:
 

```
string::push_back(char c)
```

  - προσθέτει το χαρακτήρα `c` στο τέλος της συμβολοσειράς



## A. Θεωρία

### 4. Τροποποίηση Συμβολοσειράς

#### 2. Διαγραφή

- Η μέθοδος:

```
string::clear()
```

- διαγράφει το περιεχόμενο της συμβολοσειράς (το μέγεθος γίνεται 0)

- Η μέθοδος:

```
string::erase(size_t pos = 0, size_t len = npos)
```

- διαγράφει len χαρακτήρες ξεκινώντας από τη θέση pos.
- Αν το 2<sup>ο</sup> όρισμα παραληφθεί είναι σαν να λέμε μέχρι το τέλος της συμβολοσειράς.

- Η μέθοδος:

```
string::pop_back()
```

- Επιστρέφει (και διαγράφει από τη συμβολοσειρά) τον τελευταίο χαρακτήρα του string



## A. Θεωρία

### 4. Τροποποίηση Συμβολοσειράς

#### 3. Ενημέρωση – Αποκοπή μέρους

- Η μέθοδος:

```
string::replace(size_t pos, size_t len, const string &s2, size_t subpos, size_t sublen)
```

- αντικαθιστά το μέρος του string (από τη θέση pos για len χαρακτήρες) με το μέρος του string s2 (από τη θέση subpos για sublen χαρακτήρες)

- Η μέθοδος:

```
string::substr(size_t pos=0, size_t len=npes)
```

- επιστρέφει το μέρος του string (από τη θέση pos για len χαρακτήρες)

#### Σημείωση:

- Οι περισσότερες από τις μεθόδους έχουν και υπερφορτωμένες εκδόσεις:
- βλ. <http://www.cplusplus.com/reference/string/string/>



## A. Θεωρία

### 4. Τροποποίηση Συμβολοσειράς

#### 4. Παράδειγμα

- Βλέπουμε και ένα παράδειγμα τροποποιήσεων μιας συμβολοσειράς:

```
/* cpp9.string_add_rem_upd.cpp */
```

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

```
int main()
{
    string s("0123456789");

    cout<<"Initial:  "<<s<<endl;
```

```
s.append("abcd");
s.push_back('.');
s.insert(3,"ABC");
```

```
cout<<"Insertions: "<<s<<endl;
```

```
s.pop_back();
s.erase(3,3);

cout<<"Deletions:  "<<s<<endl;

s.replace(2,4,string("ABCD"),0,4);

cout<<"Replace:   "<<s<<endl;

s.clear();

cout<<"Cleared:   "<<s<<endl;

return 0;
}
```



## A. Θεωρία

### 5. Αναζήτηση

#### 1. Η μέθοδος find()

- Η μέθοδος

```
string::find(const string &str, size_type pos=0)
```

- Ψάχνει για τη συμβολοσειρά str ξεκινώντας από τη θέση pos
- Επιστρέφει τη θέση της πρώτης εμφάνισης

- Υπερφορτωμένες εκδόσεις:

```
string::find(const char *str, size_type pos=0)
```

- Αναζητά τον πίνακα χαρακτήρων (c string) στη συμβολοσειρά

```
string::find(char c, size_type pos=0)
```

- Αναζητά το χαρακτήρα c στη συμβολοσειρά

- Όλες οι παραλλαγές της find:

- Επιστρέφουν τη θέση της πρώτης εμφάνισης του στοιχείου.
- ή τη σταθερά string.npos αν το στοιχείο δεν υπάρχει (μέγιστο μήκος string)

#### Προσοχή:

- Το npes όταν χρησιμοποιείται ως όρισμα σε μέθοδο σημαίνει ως το τέλος του string
- Αν κάνουμε πρόσβαση (string.npos) επιστρέφει το μέγιστο μέγεθος string



## A. Θεωρία

### 5. Αναζήτηση

#### 2. Οι μέθοδοι rfind(), find\_first\_of() και find\_last\_of

- Η μέθοδος

```
string::rfind(const string &str, size_type pos=npos)
```

- Ψάχνει για τη συμβολοσειρά str ξεκινώντας από τη θέση pos (και πηγαίνοντας ανάποδα)

- Η μέθοδος

```
string::find_first_of(const string &str, size_type pos=0)
```

- Ψάχνει για οποιοδήποτε από τους χαρακτήρες της συμβολοσειράς str

- Η μέθοδος

```
string::find_last_of(const string &str, size_type pos=npos)
```

- Ψάχνει για οποιοδήποτε από τους χαρακτήρες της συμβολοσειράς str ξεκινώντας από το τέλος (και πηγαίνοντας ανάποδα)

#### Σημείωση:

- Υπάρχουν υπερφορτωμένες εκδόσεις, ώστε να δουλεύουν και με 1<sup>ο</sup> όρισμα c-string, ή απλά έναν χαρακτήρα.



## A. Θεωρία

### 6. ...Και μία παρατήρηση

- Υπάρχουν αρκετές ακόμη μέθοδοι οι οποίες έχουν πιο σπάνιες χρήσεις.
- Και πως θα τα θυμόμαστε όλα αυτά;
  - Εδώ έχουμε μία σημαντική παρατήρηση.
    - Δεν πρέπει να τα θυμόμαστε όλες τις μεθόδους
    - Με μία πρώτη μελέτη έχουμε μια βασική εικόνα για τις μεθόδους που υπάρχουν
    - Και έπειτα έχουμε σαν κεντρική αναφορά κάποια ιστοσελίδα, ή δικές μας σημειώσεις.
    - Στη C++ η καλύτερη αναφορά είναι οι ιστοσελίδες:
      - <http://www.cplusplus.com/reference/>
      - <https://en.cppreference.com/w/>



## A. Θεωρία

### 5. Αναζήτηση

#### 3. Παράδειγμα

- Βλέπουμε και ένα παράδειγμα τροποποιήσεων μιας συμβολοσειράς:

```
/* cpp9.string_find.cpp */

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

int main()
{
    string s("0123456789");
    string f("345");

    s+=s;
    cout<<"str- find-345c: pos: "<<s.find("345")<<endl;
    cout<<"str- find-345s: pos: "<<s.find(f)<<endl;
    cout<<"str- find-1c : pos: "<<s.find('1')<<endl;
    cout<<"str- rfind-345 : pos: "<<s.rfind("345")<<endl;
    cout<<"str-firstof-345 : pos: "<<s.find_first_of("345")<<endl;
    cout<<"str-last of-345 : pos: "<<s.find_last_of("345")<<endl;
    return 0;
}
```



## B. Ασκήσεις

### Άσκηση 1: Ταξινόμηση Συμβολοσειρών

Στο μάθημα «Αλγόριθμοι σε C – Μάθημα 3» είδαμε τον αλγόριθμο ταξινόμησης InsertionSort:

```
/* Taksinomisi me Eisagogi */
for (i=1; i<N; i++)
{
    for (j=i; j>=1; j--)
    {
        if (pinakas[j]<pinakas[j-1])
            swap(&pinakas[j], &pinakas[j-1]);
        else
            break;
    }
}
```

Υλοποιήστε μία συνάρτηση η οποία να δέχεται ως όρισμα ένα πίνακα συμβολοσειρών και να ταξινομεί τις συμβολοσειρές σε αύξουσα σειρά.



## Β. Ασκήσεις

### Άσκηση 2: Κρεμάλα!

Κατασκευάστε μία απλή έκδοση της κρεμάλας

Hints:

- Να υπάρχει ένας πίνακας λέξεων από τις οποίες θα διαλέγεται τυχαία η κρυφή λέξη (hidden)
- Να υπάρχει μία συμβολοσειρά (game) η οποία να αρχικοποιείται με παύλες
- Να υπάρχει μία συμβολοσειρά (bad\_guesses) στην οποία θα αποθηκεύονται οι λάθος μαντεψιές
- Να υπάρχει μία συμβολοσειρά (right\_guesses) στην οποία θα αποθηκεύονται οι σωστές μαντεψιές
- Να υπάρχει ένας ακέραιος (MAX) με το μέγιστο πλήθος λάθων που μπορεί να κάνει ο παίκτης

Αλγόριθμος:

- Επαναληπτικά:
  - Ο χρήστης μαντεύει ένα γράμμα.
  - Ελέγχεται αν έχει ήδη επιλεχθεί το γράμμα σε προηγούμενο βήμα
  - Ελέγχεται αν υπάρχει στη hidden:
    - Αν υπάρχει, διορθώνεται η game
      - Αν έχει ολοκληρωθεί η λέξη, τελειώνει το παιχνίδι με νίκη
    - Αν δεν υπάρχει, διορθώνεται η badguesses
      - Αν έχουν γίνει MAX λάθη, τελειώνει το παιχνίδι με ήττα