

C-Strings, std::string, std::string_view και οι τελεστές [] και &

#13

Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)

Πανεπιστήμιο Ιωαννίνων

Γκόγκος Χρήστος

C-strings

- Τα C-strings υλοποιούνται ως πίνακες χαρακτήρων που τερματίζονται με το χαρακτήρα `'\0'` (NULL).

```
char buffer[5];  
strcpy(buffer, "hi!\n");  
cout << buffer;
```



- Όταν χρησιμοποιούμε τα διπλά εισαγωγικά `" "`, ο μεταγλωττιστής κατασκευάζει μια τερματιζόμενη με NULL, `const` ακολουθία χαρακτήρων την οποία γεμίζει με τους χαρακτήρες που ο προγραμματιστής έχει επιλέξει.
- Αν ένας πίνακας χαρακτήρων δεν τερματίζει με NULL τότε δεν είναι C-string.

C-string και C++

- Υπάρχουν ενσωματωμένες δυνατότητες χειρισμού C-strings στις standard βιβλιοθήκες της C++.
 - Η βιβλιοθήκη `<cstring>`
 - Περιέχει συναρτήσεις για συνηθισμένες λειτουργίες πάνω σε strings όπως αντιγραφή, συνένωση, εύρεση μήκους, αναζήτηση, διάσπαση c-string σε τμήματα και άλλα.
 - `strcpy()`, `strcat()`, `strlen()`, `strncat()`, `strcmp()`, `strncmp()`, `strstr()`, `strtok()`, ...
 - Η βιβλιοθήκη `<iostream>`
 - Περιέχει συναρτήσεις για το χειρισμό I/O των C-strings, όπως οι τελεστές εισαγωγής (`<<`) και εξαγωγής (`>>`), οι συναρτήσεις `get()`, `getline()` κ.α.
 - ```
char str1[140];
cout << str1; // τελεστής εισαγωγής για c-strings
cin >> str1; // τελεστής εξαγωγής για c-strings, διαβάζει μέχρι τον πρώτο κενό
 χαρακτήρα
cin.get(str1, 40, ','); // διαβάζει μέχρι να συναντήσει το διαχωριστικό κόμμα (,
 ή μέχρι να αναγνωστούν 39 = 40-1 χαρακτήρες
cin.getline(str1, 40); // διαβάζει μέχρι το διαχωριστικό (το προκαθορισμένο είναι
 η αλλαγή γραμμής), απορρίπτει το διαχωριστικό ή μέχρι να αναγνωστούν 39 χαρακτήρες
```

# Μειονεκτήματα των C-strings

- Σταθερό μέγεθος (ορίζεται όταν δηλώνεται το C-string ως στατικός πίνακας).
- Το όνομα του C-string λειτουργεί ως δείκτης.
- Τα όρια του πίνακα δεν επιβάλλονται με κάποιο τρόπο.
- Πρέπει να χρησιμοποιούν «άβολες» συναρτήσεις αντί για διαισθητικά εύκολα κατανοητούς τελεστές.
  - `strcpy(str1, str2)` αντί για `str1=str2`
  - `strcmp(str1, str2)` αντί για `str1==str2`
  - `strcat(str1, str2)` αντί για `str1+=str2`
- Η χρήση του NUL χαρακτήρα μπορεί να δημιουργήσει προβλήματα.

# Παραδείγματα κώδικα με C-strings

- <https://github.com/chgogos/oop/blob/master/variuous/COP3330/lect13/sample2.cpp>
- <https://github.com/chgogos/oop/blob/master/variuous/COP3330/lect13/sample3.cpp>
- <https://github.com/chgogos/oop/blob/master/variuous/COP3330/lect13/sample4.cpp>
- [https://github.com/chgogos/oop/blob/master/cpp\\_playground/ex085/c\\_string1.cpp](https://github.com/chgogos/oop/blob/master/cpp_playground/ex085/c_string1.cpp)

# std::string

- Πλεονεκτήματα std::string έναντι C-strings
  - Μεταβλητό μέγεθος
  - Εύρεση μήκους σε σταθερό χρόνο (και όχι σε γραμμικό)
  - Δεν απαιτούν εντολές διαχείρισης μνήμης
  - Αυτόματος χειρισμός των ορίων των λεκτικών
  - Διαισθητικά εύκολη ανάθεση τιμής με το = αντί για το strcpy
  - Διαισθητικά εύκολη σύγκριση με το == αντί για το strcmp
  - Διαισθητικά εύκολη συνένωση λεκτικών με το + αντί για το strcat
  - Μετατροπή σε C-string με τη συνάρτηση μέλος c\_str()
- Δείτε το string1.cpp

[https://github.com/chgogos/oop/blob/master/cpp\\_playground/ex085/string1.cpp](https://github.com/chgogos/oop/blob/master/cpp_playground/ex085/string1.cpp)

[https://github.com/chgogos/oop/blob/master/cpp\\_playground/ex085/string2.cpp](https://github.com/chgogos/oop/blob/master/cpp_playground/ex085/string2.cpp)

# Άσκηση #1: C-strings, std::string

- Γράψτε μια συνάρτηση με όνομα `cstr_length` που να δέχεται ως όρισμα ένα αλφαριθμητικό `char*` και χωρίς τη χρήση της συνάρτησης `strlen` να επιστρέφει το μήκος του αλφαριθμητικού. Καλέστε τη συνάρτηση από τη `main` για ένα κείμενο που εισάγει ο χρήστης που μπορεί να περιέχει και κενά. Χρησιμοποιήστε τη `cin.getline()` και θεωρήστε ότι η είσοδος του χρήστη δεν θα υπερβαίνει τους 100 χαρακτήρες.
- Γράψτε μια συνάρτηση με όνομα `interpolate_spaces` που να δέχεται ως όρισμα ένα αλφαριθμητικό `std::string` και να επιστρέφει νέο αλφαριθμητικό `std::string` με ένα κενό ανάμεσα από κάθε δύο γράμματα του ορίσματος. Καλέστε τη συνάρτηση από τη `main` για το αλφαριθμητικό "INFORMATION"

# Άσκηση #1: Λύση



```
int cstr_length(const char* s) {
 int len = 0;
 while (s[len] != '\0') {
 ++len;
 }
 return len;
}
```



```
string interpolate_spaces(const string& s) {
 string result;
 for (int i = 0; i < s.size(); ++i) {
 result.push_back(s[i]);
 if (i < s.size() - 1) {
 result.push_back(' ');
 }
 }
 return result;
}
```

<https://github.com/chgogos/oop/blob/master/variuous/COP3330/lect13/exercise1.cpp>



# std::string\_view (C++17)

- Στη C++17, με τη χρήση της std::string\_view μπορούν να αποφευχθούν περιττές αντιγραφές λεκτικών που θα συνέβαιναν με τη std::string
- Ένα std::string\_view μπορεί να αναφέρεται τόσο σε ένα std::string όσο και σε ένα C-string
- Τα std::string\_view έχουν το ίδιο API με τα std::string

```
const char *s = "ABCDEF";
char s2[10];
strcpy(s2, s);
string_view sv{s2};
cout << s << " " << s2 << " " << sv << endl;
s2[0] = '*';
cout << s << " " << s2 << " " << sv << endl;
```

```
ABCDEF ABCDEF ABCDEF
ABCDEF *BCDEF *BCDEF
```

[https://github.com/chgogos/oop/blob/master/cpp\\_playground/ex085/string\\_view1.cpp](https://github.com/chgogos/oop/blob/master/cpp_playground/ex085/string_view1.cpp)

[https://github.com/chgogos/oop/blob/master/cpp\\_playground/ex085/string\\_view2.cpp](https://github.com/chgogos/oop/blob/master/cpp_playground/ex085/string_view2.cpp)

## Άσκηση #2: `std::string_view`

- Να γραφεί πρόγραμμα που να συγκρίνει τον χρόνο εκτέλεσης δύο συναρτήσεων: η πρώτη θα δέχεται ως όρισμα ένα `std::string` και θα υπολογίζει το πλήθος των γραμμάτων 'f' που διαθέτει, ενώ η δεύτερη θα δέχεται `std::string_view` και θα πραγματοποιεί την ίδια ακριβώς εργασία χωρίς δημιουργία αντιγράφου.
- Στη `main()` να δημιουργηθεί μία συμβολοσειρά με 1000 τυχαία πεζά γράμματα του αγγλικού αλφαβήτου.
- Να κληθούν και οι δύο συναρτήσεις 1.000.000 φορές μέσα σε βρόχο και να χρονομετρηθούν ώστε να γίνει εμφανής η διαφορά ταχύτητας.

# Υπερφόρτωση του τελεστή `[]`

- Γίνεται με δύο συναρτήσεις μέλη:
  - `τύπος_επιστρεφόμενης_τιμής operator[](τύπος_δείκτη index) const;`
  - `τύπος_επιστρεφόμενης_τιμής& operator[](τύπος_δείκτη index);`
- Η `const` συνάρτηση μέλος επιτρέπει την ανάγνωση στοιχείων από ένα `const` αντικείμενο.
- Η συνάρτηση μέλος που δεν είναι `const` επιστρέφει μια αναφορά στο στοιχείο που μπορεί να τροποποιηθεί.

<https://github.com/chgogos/oop/blob/master/variuous/COP3330/lect13/sample5.cpp>

# Υπερφόρτωση τελεστή &

- Ο τελεστής & μπορεί να υπερφορτωθεί όπως και οποιοσδήποτε άλλος τελεστής.

<https://github.com/chgogos/oop/blob/master/variuous/COP3330/lect13/sample6.cpp>

# Ερωτήσεις σύνοψης

- Πως υλοποιείται στη C ένα λεκτικό;
- Τι είναι η συνάρτηση `strcmp` και ποια επικεφαλίδα πρέπει να γίνει `include` έτσι ώστε να μπορεί να χρησιμοποιηθεί;
- Τι επιτυγχάνουμε με τη συνάρτηση `getline()` του αντικειμένου `cin`;
- Γιατί υπάρχουν δύο υπερφορτώσεις συναρτήσεων για το `operator[]`;

# Ερωτήσεις σύνοψης

- Πως υλοποιείται στη C ένα λεκτικό;
- Τι είναι η συνάρτηση `strcmp` και ποια επικεφαλίδα πρέπει να γίνει `include` έτσι ώστε να μπορεί να χρησιμοποιηθεί;
- Τι επιτυγχάνουμε με τη συνάρτηση `getline()` του αντικειμένου `cin`;
- Γιατί υπάρχουν δύο υπερφορτώσεις συναρτήσεων για το `operator[]`;

# Αναφορές

- <http://www.cs.fsu.edu/~xyuan/cop3330/>
- <http://www.cs.fsu.edu/~myers/c++/notes/strings.html>
- <https://embeddedartistry.com/blog/2017/07/26/stdstring-vs-c-strings/>