ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 9: C++ ΕΙΣΟΔΟΣ - ΕΞΟΔΟΣ / ΑΛΦΑΡΙΘΜΗΤΙΚΑ / ΑΡΧΕΙΑ

Αλφαριθμητικά

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Αλφαριθμητικά

Αρχικοποίηση

- Template class **`basic_string`**
  - Μεταχείριση αλφαριθμητικών (αντιγραφή, αναζήτηση κτλ.)
    - **`typedef basic_string< char > string;`**
    - Also **`typedef`** for **`wchar_t`**
  - Βιβλιοθήκη: **`<string>`**
- Αρχικοποίηση **`string`**
  - **`string s1("word");`** `word`
  - **`string s2( 5, 'x' );`** `xxxxx`
  - **`string s3 = "word"`** `word`
    - Έμμεσα καλεί τον δημιουργό

Αρχικοποίηση

- Δεν επιτρέπεται μετατροπή από **`int`** ή **`char`**
  - Τα παρακάτω προκαλούν σφάλματα:
    - **`string err1 = 'd';`**
    - **`string err2('x' );`**
    - **`string err3 = 45;`**
    - **`string err4( 5 );`**
  - Μπορούμε να αναθέσουμε έναν χαρακτήρα με :
    - **`s = 'n';`**

Χαρακτηριστικά

- **`string`**
  - Δεν είναι απαραίτητο να τερματίζονται με **`null`**
  - Η μέθοδος **`length`** επιστρέφει τον αριθμό χαρακτήρων: `s1.length()`
  - Ο τελεστής `[]` επιτρέπει προσπέλαση χαρακτήρων: **`s1[0]`**
  - Ένα **`string`** δεν είναι δείκτης
  - Ανάκτηση από κανάλι
    - `cin >> stringObject;`
    - `getline( cin, s)`
      - Διαβάζει μέχρι τον χαρακτήρα επόμενης γραμμής.

Ανάθεση

- Ανάθεση
  - **`s2 = s1;`** ή **`s2.assign(s1);`**
    - Δημιουργεί αντίγραφο
  - **`myString.assign(s, start, N);`**
    - αντιγράφει **`N`** χαρακτήρες από το string **`s`**, ξεκινώντας από τη θέση **`start`**
  - Ανάθεση συγκειμένων θέσεων (χαρακτήρων)
    - **`s1[0] = s2[4];`**

Μέθοδοι at και append

- Έλεγχος εμβέλειας
  - `s.at( index );`
    - Επιστρέφει τον χαρακτήρα στην θέση **`index`**
    - Μπορεί να πετάξει εξαίρεση **`out_of_range`** αν δοθεί αριθμός εκτός ορίων
  - Ο τελεστής **`[]`** δεν κάνει τέτοιο έλεγχο
- Προσθήκη στο τέλος (Concatenation)
  - `s.append("winter");` ή `s += "winter";`
    - Προσθέτουν το **`"pet"`** στο τέλος του **`s`**
  - **`s.append( ssrc, start, N );`**
    - Τοποθετεί στο τέλος του **s**, **N** χαρακτήρες από τον **ssrc**, ξεκινώντας από την θέση **start**

Σύγκριση

- Υπερφορτωμένοι τελεστές σύγκρισης:
  - **==**, **!=**, **<**, **>**, **<=** και **>=**
  - Επιστρέφουν τιμή **bool**
- **s1.compare(s2)**
  - Επιστρέφει θετικό αν το **s1** είναι μεγαλύτερο λεξικογραφικά
    - Σύγκριση γράμμα προς γράμμα
    - Αρνητικό αν είναι μικρότερο, 0 αν είναι ίσα
  - **s1.compare(start, length, s2, start, length)**
    - Συγκρίνει επιμέρους μέρη του **s1** και **s2**
  - **s1.compare(start, length, s2)**
    - Συγκρίνει επιμέρους μέρος του **s1** με ολόκληρο το **s2**

Μέθοδοι substr , switch

- Η **`substr`** επιστρέφει ένα μέρος του αλφαριθμητικού
  - **`s.substr( start, N );`**
  - Επιστρέφει ένα string παίρνοντας **`N`** χαρακτήρες του **`s`**, ξεκινώντας από την θέση **`start`**
- Μέθοδος **`switch`**
  - **`s1.swap(s2);`**
  - Ανταλλάσσει τα περιεχόμενα των δύο strings

Χρήσιμες Μεθόδοι

- **`s1.size()`** και **`s1.length()`**
  - Επιστρέφει το μέγεθος του string
- **`s1.capacity()`**
  - Επιστρέφει πόσα στοιχεία μπορούν να αποθηκευτούν χωρίς να γίνει reallocation
- **`s1.max_size()`**
  - Επιστρέφει το μέγιστο μέγεθος του string
- **`s1.empty()`**
  - Επιστρέφει true αν είναι άδειο
- **`s1.resize(newlength)`**
  - Αλλάζει το μέγεθος σε newlength

Μέθοδοι αναζήτησης

- Επιστρέφουν την θέση που βρέθηκε
- Αν δεν βρεθεί επιστρέφουν, **`string::npos`** (σταθερά)
  - **`s1.find( s2 )`**
  - **`s1.rfind( s2 )`**
    - Ξεκινάει την αναζήτηση από δεξιά
  - **`s1.find_first_of( s2 )`**
    - Επιστρέφει την πρώτη εμφάνιση στο **s1** οποιουδήποτε χαρακτήρα του **`s2`**
  - **`s1.find_last_of( s2 )`**
    - Επιστρέφει την τελευταία εμφάνιση στο **s1** οποιουδήποτε χαρακτήρα του **`s2`**
  - **`s1.find_first_not_of( s2 )`**
    - Επιστρέφει την πρώτη εμφάνιση στο **s1** οποιουδήποτε χαρακτήρα που δεν υπάρχει στο **`s2`**
  - **`s1.find_last_not_of( s2 )`**
    - Επιστρέφει την τελευταία εμφάνιση στο **s1** οποιουδήποτε χαρακτήρα που δεν υπάρχει στο **`s2`**

Αφαίρεση - Αντικατάσταση

- **`s1.erase( start )`**
  - Αφαίρεση από τη θέση **`start`** μέχρι το τέλος
- **`s1.replace( begin, N, s2)`**
  - Ξεκινώντας από την θέση **begin** του **s1** αντικατέστησε τους επόμενους **N** χαρακτήρες, με χαρακτήρες από το **s2**
- **`s1.replace( begin, N, s2, index, num )`**
  - Όμοια με παραπάνω αλλά καθορίζεται επιπλέον από ποιο σημείο index του s2 θα ξεκινήσουμε και πόσους χαρακτήρες θα χρησιμοποιήσουμε
- Η θέση **`string::npos`** αναπαριστά το μέγιστο μέγεθος του string

Ένθεση

- **`s1.insert( index, s2 )`**
  - Ενθέτει το **`s2`** πριν από τη θέση **`index`**
- **`s1.insert( index, s2, index2, N );`**
  - Ενθέτει ένα μέρος του **`s2`** **(N** χαρακτήρες ξεκινώντας από τη θέση **index2)** πριν από τη θέση **`index`**

Μετατροπή

- Μέθοδοι μετατροπής:
  - **`s1.copy( ptr, N, index )`**
    - αντιγράφει **`N`** χαρακτήρες στον πίνακα **`ptr (*char)`**
    - Ξεκινάει από την θέση **`index`**
    - Πρέπει να τερματιστεί με null
  - **`s1.c_str()`**
    - Επιστρέφει **`const char *`**
    - Δεν χρειάζεται να τερματιστεί με null
  - **`s1.data()`**
    - Επιστρέφει **`const char *`**
    - Πρέπει να τερματιστεί με null

Iterators

- Iterators
  - Διαπέραση χαρακτήρων (από αρχή ή από το το τέλος)
  - Πρόσβαση στους επιμέρους χαρακτήρες
  - Όμοια με δείκτες

- Βασική Χρήση
  - Δημιουργία
    - `string::const_iterator i = s.begin();`
    - **`const`**, δεν επιτρέπεται τροποποίηση
  - Αναφορά
    - `*i;  // αναφορά στον χαρακτήρα`
    - `++i;  // μεταφορά στον επόμενο χαρακτήρα`
  - Έλεγχος για τέλος
    - `i != s.end()`
    - **`end`** επιστρέφει iterator μετά το τελευταίο στοιχείο του **`s`**

String Streams

- I/O of strings to and from memory
  - Called in-memory I/O or string stream processing
  - Classes
    - **`istringstream`** (είσοδος από string)
    - **`ostringstream`** (έξοδος σε string)
    - **`<sstream>`** **`<iostream>`**
- Κανάλι εξόδου ostringstream
  - **`ostringstream outputString;`** Δημιουργία string-καναλιού
  - **`outputString << s1 << s2;`** Αποθήκευση strings στο string-κανάλι
  - Μέθοδος : **`str`**
    - Επιστρέφει ως **`string`** το περιεχόμενο του `string-καναλιού`
    - **`outputString.str()`**
- Κανάλι εισόδου istringstream
  - **`istringstream inputString ( myString );`** Δημιουργία string-καναλιού από string
  - **`inputString >> string1 >> string2`** Ανάκτηση από το string-κανάλι σε strings
  - Με παρόμοιο τρόπο που διαβάζουμε από την κύρια είσοδο, **`cin`**

Πρόσθετο Υλικό

- Μελετήστε και τα παραδείγματα από το **Κεφάλαιο 21** του βιβλίου:
  «C++ How to Program, 9/e Paul & Harvey Deitel»
  http://media.pearsoncmg.com/ph/esm/deitel/cpp_htp_9/code_examples/Code_Examples.zip

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

- Οι διαφάνειες βασίζονται στο βιβλίο «C++ How to Program, 8th Edition, Harvey M. Deitel, Paul J. Deitel, Prentice Hall.»