343 – Εισαγωγή στον Προγραμματισμό

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Ακαδημαϊκό Έτος 2015-2016

Χάρης Παπαδόπουλος
207δ, Β' όροφος
e-mail: charis@cs.uoi.gr

Ώρες Γραφείου:
Πέμπτη 11-13

2ο Quiz

- Το 2ο quiz θα διεξαχθεί την **Δευτέρα 14 Δεκεμβρίου**
- **Περιλαμβάνει όλες τις διαφάνειες μέχρι και Lecture-8.pdf**
  συναρτήσεις & πίνακες
- 14:00-18:30 στα ακόλουθα τμήματα:

Ώρες	Α. Μ.
14:00 - 14:45	ΤΜΗΜΑ Α1
14:45 - 15:30	ΤΜΗΜΑ Α2
15:30 - 16:15	ΤΜΗΜΑ Α3
16:15 - 17:00	ΤΜΗΜΑ Α4
17:00 - 17:45	ΤΜΗΜΑ Α5
17:45 - 18:30	ΤΜΗΜΑ Α+

- Αν λείψετε:
  - Δεν μετράει ως απουσία
  - Μηδενίζεται το 5%

Μέχρι τότε θα πρέπει μόνοι σας να γραφτείτε στο ecourse του μαθήματος:

- Δοκιμάστε 5 παραπλήσιες ερωτήσεις

ecourse.uoi.gr:
Εισαγωγή στον Προγραμματισμό

Σημειώστε:
- **username**
- **password**

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2015

Δ	Τ	Τ	Π	Π
			1	2
5	6	7	8	**9 Θ**
12	13	14	15	**16 Θ**
19 Ε	20	21	22	**23 Θ**
26	27	28	29	**30 Θ**

Νοέμβριος 2015

Δ	Τ	Τ	Π	Π
2 Ε	3	4	5	**6 Θ**
9 Ε	10	11	12	**13 Θ**
16 Q	17	18	19	**20 Θ**
23 Ε	24	25	26	**27 Θ**
30 Ε				

Δεκέμβριος 2015

Δ	Τ	Τ	Π	Π
	1	2	3	**4 Θ**
7 Ε	8	9	10	**11 Θ**
14 Q	15	16	17	**18 Θ**

Ιανουάριος 2016

Δ	Τ	Τ	Π	Π
4	5	6	7	8
11	12	13	14	**15 Θ**

Εβδομάδα	Θέματα	Ύλη βιβλιογραφίας
Πα, 9 Οκτωβρίου	**Εισαγωγικά μαθήματος & Δυαδική αναπαράσταση**	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πα, 16 Οκτωβρίου	**Είσοδος/Έξοδος δεδομένων, τύποι δεδομένων & μεταβλητών**	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Δε, 19 Οκτ	1ο Εργαστήριο	
Πα, 23 Οκτωβρίου	**Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές**	[1]: 2.1, Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Πα, 30 Οκτωβρίου	**Ροή ελέγχου: if/else, switch, for, while, do-while και ροή ελέγχου if/else**	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε, 2 Νοε	2ο Εργαστήριο	
Πα, 6 Νοεμβρίου	**Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή**	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε, 9 Νοε	3ο Εργαστήριο	
Πα, 13 Νοεμβρίου	**Επανάληψη Εργαστηρίων**	
Δε, 16 Νοε	**1ο Quiz**	
Πα, 20 Νοεμβρίου	**Επανάληψη με Παραδείγματα**	
Δε, 23 Νοε	4ο Εργαστήριο	
Πα, 27 Νοεμβρίου	**Πίνακες (μονοδιάστατοι και πολυδιάστατοι)**	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε, 30 Νοε	5ο Εργαστήριο	
Πα, 4 Δεκεμβρίου	**Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων**	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε, 7 Δεκ	6ο Εργαστήριο	
Πα, 11 Δεκεμβρίου	**Αλφαριθμητικά και Συμβολοσειρές**	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε, 14 Δεκ	**2ο Quiz**	
Πα, 18 Δεκεμβρίου	**Εγγραφές, δομές και χρήση αρχείων**	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πα, 15 Ιανουαρίου	**Επανάληψη**	

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2015

Δ	Τ	Τ	Π	Π
			1	2
5	6	7	8	**9 Θ**
12	13	14	15	**16 Θ**
19 Ε	20	21	22	**23 Θ**
26	27	28	29	**30 Θ**

Νοέμβριος 2015

Δ	Τ	Τ	Π	Π
2 Ε	3	4	5	**6 Θ**
9 Ε	10	11	12	**13 Θ**
16 Q	17	18	19	**20 Θ**
23 Ε	24	25	26	**27 Θ**
30 Ε				

Δεκέμβριος 2015

Δ	Τ	Τ	Π	Π
	1	2	3	**4 Θ**
7 Ε	8	9	10	**11 Θ**
14 Q	15	16	17	**18 Θ**

Ιανουάριος 2016

Δ	Τ	Τ	Π	Π
4	5	6	7	8
11	12	13	14	**15 Θ**

Εβδομάδα	Θέματα	Ύλη βιβλιογραφίας
Πα, 9 Οκτωβρίου	**Εισαγωγικά μαθήματος & Δυαδική αναπαράσταση**	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πα, 16 Οκτωβρίου	**Είσοδος/Έξοδος δεδομένων, τύποι δεδομένων & μεταβλητών**	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Δε, 19 Οκτ	1ο Εργαστήριο	
Πα, 23 Οκτωβρίου	**Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές**	[1]: 2.1, Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Πα, 30 Οκτωβρίου	**Ροή ελέγχου: if/else, switch, for, while, do-while και ροή ελέγχου if/else**	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε, 2 Νοε	2ο Εργαστήριο	
Πα, 6 Νοεμβρίου	**Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή**	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε, 9 Νοε	3ο Εργαστήριο	
Πα, 13 Νοεμβρίου	**Επανάληψη Εργαστηρίων**	
Δε, 16 Νοε	**1ο Quiz**	
Πα, 20 Νοεμβρίου	**Επανάληψη με Παραδείγματα**	
Δε, 23 Νοε	4ο Εργαστήριο	
Πα, 27 Νοεμβρίου	**Πίνακες (μονοδιάστατοι και πολυδιάστατοι)**	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε, 30 Νοε	5ο Εργαστήριο	
Πα, 4 Δεκεμβρίου	**Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων**	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε, 7 Δεκ	6ο Εργαστήριο	
Πα, 11 Δεκεμβρίου	**Αλφαριθμητικά και Συμβολοσειρές**	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε, 14 Δεκ	**2ο Quiz**	
Πα, 18 Δεκεμβρίου	**Εγγραφές, δομές και χρήση αρχείων**	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πα, 15 Ιανουαρίου	**Επανάληψη**	

Ενότητα 20

ΑΛΦΑΡΙΘΜΗΤΙΚΑ (C-STRING)

Αλφαριθμητικά και Συμβολοσειρές

- Δύο ειδών συμβολοσειρές:
- C-string:
  - πίνακας από χαρακτήρες (char)
  - το τέλος της συμβολοσειράς μαρκάρεται με το σύμβολο '\0'
  - κλασικός τρόπος και στην απλή C
  - χρησιμοποιεί έτοιμες βιβλιοθήκες
- Η τυποποιημένη κλάση string:
  - χρησιμοποιεί έτοιμες βιβλιοθήκες με περισσότερες δυνατότητες

C-string

- Πίνακες από χαρακτήρες (τύπου char)

- Ένας χαρακτήρας για κάθε μεταβλητή τύπου char
- Ένας επιπλέον χαρακτήρας '\0' (*κάθετος μηδέν*)
  - καλείται null χαρακτήρας
  - σηματοδοτεί το τέλος της συμβολοσειράς

- Έχουμε ήδη χρησιμοποιήσει C-string
  - Η φράση "Γεια" αποθηκεύεται σε ένα C-string s:

    char s[10] = "Γεια";

C-string μεταβλητή

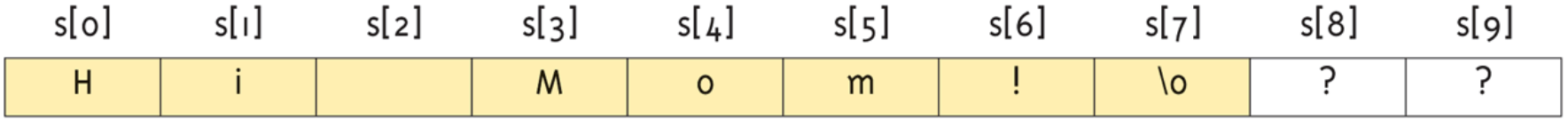
- Πίνακας από χαρακτήρες:
  char s[10];
  - Δηλώνει μια C-string μεταβλητή για να αποθηκεύσει μέχρι 9 χαρακτήρες
  - + έναν null χαρακτήρα
- Τυπικά είναι "μερικώς συμπληρωμένος" πίνακας
  - Δηλώνουμε αρκετά μεγάλο μέγεθος για να αποθηκεύσουμε μέχρι το μέγιστο μήκος συμβολοσειράς
  - Σηματοδοτούμε το τέλος με null
- Η μοναδική διαφορά με τους κλασικούς πίνακες:
  - Πρέπει να περιέχουν null χαρακτήρες

Αποθήκευση C-string

- Ένας κλασικός πίνακας:

  char s[10];

  - Αν το s περιέχει την συμβολοσειρά " Hi Mom!", αποθηκεύεται ως:



Αρχικοποίηση

- Αρχικοποίηση C-string:
  char myMessage[20] = "Hi there.";
  - Δεν χρειάζεται να γεμίσουμε ολόκληρο τον πίνακα
  - Η αρχικοποίηση τοποθετεί το '\0' στο τέλος (αυτόματα)
- Μπορούμε να αποφύγουμε το μέγεθος του πίνακα:
  char shortString[] = "abc";
  - Αυτόματα μετατρέπει το μέγεθος **+1** από το μέγεθος της συμβολοσειράς μέσα σε " "
  - ΔΕΝ είναι το ίδιο με:
    char shortString[] = {'a', 'b', 'c'};

Δείκτες σε C-string

- Ένα C-string ΕΙΝΑΙ ένας πίνακας

- Πρόσβαση στις δεικτοδοτούμενες μεταβλητές:

char ourString[5] = "Hi";

  - ourString[0] είναι 'H'
  - ourString[1] είναι 'i'
  - ourString[2] είναι '\0'
  - ourString[3] είναι άγνωστο
  - ourString[4] είναι άγνωστο

Χειρισμός δεικτών C-string

- Μπορούμε να χειριστούμε δεικτοδοτούμενες μεταβλητές

```
char happyString[7] = "DoBeDo";
happyString[6] = 'Z';
```

  - Θέλει προσοχή!!
  - Εδώ, το '\0' (null) αντικαταστάθηκε από ένα 'Z'!

- Αν το null αντικαθίσταται, τότε το C-string δεν "συμπεριφέρεται" ως C-string!
  - Άγνωστα αποτελέσματα!

Παραδείγματα

```
char outstring[5] = "Γεια";

int index = 0;

while( outstring[index] != '\0' )
{
    outstring[index] = 'A';
    index++;
}
```

```
char outstring[5] = "Γεια";

int index = 0;

while( (outstring[index] != '\0') && (index < SIZE) )
{
    outstring[index] = 'A';
    index++;
}
```

προκαθορισμένη σταθερά ίση με το μέγιστο μέγεθος του πίνακα

Βιβλιοθήκες

- Δήλωση C-strings
  - Δεν απαιτεί κάποια C++ βιβλιοθήκη
  - Υπάρχει στην standard C++
- Χειρισμοί:
  - Απαιτεί την βιβλιοθήκη <cstring>

```
#include <cstring>
```

  - Συνήθως την καλούμε όταν χρησιμοποιούμε C-strings
    - Όταν θέλουμε να τις χειριστούμε με μεγάλη ευκολία

"=" και "==" με C-strings

- Τα C-strings δεν λειτουργούν όπως άλλες μεταβλητές
  - Δεν μπορούμε να αναθέσουμε ή να συγκρίνουμε:

```
char aString[10];
aString = "Hello";      // ΛΑΘΟΣ!!
```

    - Χρήση του "=" ΜΟΝΟ στη δήλωση του C-string!

- Πρέπει να χρησιμοποιήσουμε συναρτήσεις από βιβλιοθήκες για ανάθεση:

```
char aString[10];
strcpy(aString, "Hello");
```

  - `strcpy:` Συνάρτηση στη βιβλιοθήκη <cstring>
  - Θέτει την τιμή του aString ίση με "Hello"
  - ΔΕΝ ελέγχει για μέγεθος!
    - Ο έλεγχος γίνεται από τον προγραμματιστή, όπως στους υπόλοιπους πίνακες!

Σύγκριση C-strings

- Επίσης δεν μπορούμε να κάνουμε χρήση του τελεστή ==

```
char aString[10] = "Hello";
char anotherString[10] = "Goodbye";
if( aString == anotherString )   // NOT allowed!
{ ... }
```

- Πρέπει να κάνουμε χρήση συνάρτησης:

```
char aString[10] = "Hello";
char anotherString[10] = "Goodbye";
if ( strcmp(aString, anotherString) )
        cout << "Strings NOT same.";
else
        cout << "Strings are same.";
```

Η βιβλιοθήκη <cstring> (1/2)

Display 9.1 Some Predefined C-String Functions in `<cstring>`

FUNCTION	DESCRIPTION	CAUTIONS
`strcpy(`*Target_String_Var*`,` *Src_String*`)`	Copies the C-string value *Src_String* into the C-string variable *Target_String_Var*.	Does not check to make sure *Target_String_Var* is large enough to hold the value *Src_String*.
`strcpy(`*Target_String_Var*`,` *Src_String, Limit*`)`	The same as the two-argument `strcpy` except that at most *Limit* characters are copied.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcpy`. Not implemented in all versions of C++.
`strcat(`*Target_String_Var*`,` *Src_String*`)`	Concatenates the C-string value *Src_String* onto the end of the C-string in the C-string variable *Target_String_Var*.	Does not check to see that *Target_String_Var* is large enough to hold the result of the concatenation.

(continued)

Η βιβλιοθήκη <cstring> (2/2)

Display 9.1 Some Predefined C-String Functions in `<cstring>`

FUNCTION	DESCRIPTION	CAUTIONS
`strcat(`*Target_String_Var*`,` *Src_String, Limit*`)`	The same as the two argument `strcat` except that at most *Limit* characters are appended.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcat`. Not implemented in all versions of C++.
`strlen(`*Src_String*`)`	Returns an integer equal to the length of *Src_String*. (The null character, `'\0'`, is not counted in the length.)	
`strcmp(`*String_1*`,` *String_2*`)`	Returns 0 if *String_1* and *String_2* are the same. Returns a value < 0 if *String_1* is less than *String_2*. Returns a value > 0 if *String_1* is greater than *String_2* (that is, returns a nonzero value if *String_1* and *String_2* are different). The order is lexicographic.	If *String_1* equals *String_2*, this function returns 0, which converts to `false`. Note that this is the reverse of what you might expect it to return when the strings are equal.
`strcmp(`*String_1*`,` *String_2, Limit*`)`	The same as the two-argument `strcat` except that at most *Limit* characters are compared.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcmp`. Not implemented in all versions of C++.

Η συνάρτηση strlen()

- Επιστρέφει το μήκος της συμβολοσειράς
- Είναι συχνά χρήσιμο να γνωρίζουμε το μέγεθος:
  - και για να μην ξεφεύγουμε από τα όρια του πίνακα

```
char myString[10] = "dobedo";
cout << strlen(myString);
```

  - Επιστρέφει το πλήθος των χαρακτήρων
    - Χωρίς να περιέχει τον null ('\0') χαρακτήρα
  - Το πάνω αποτέλεσμα θα είναι:

6

Η συνάρτηση strcat()

- strcat(s1,s2)
  - συνενώνει την συμβολοσειρά s2 στο τέλος της s1

```
char stringVar[20] = "The rain";
strcat(stringVar, "in Spain");
```

- Στο αποτέλεσμα:
  η stringVar τώρα θα είναι "The rainin Spain"
- Χρειάζεται προσοχή:
  - Χρήση των κενών!
  - Δεν επιβεβαιώνει αν η s1 είναι αρκετά μεγάλη για να δεχθεί την s2

Παράμετροι και Ορίσματα των C-strings

- Θυμηθείτε: το C-string είναι πίνακας
- Επομένως ως παράμετρος C-string είναι παράμετρος πίνακα
  - Τα C-strings που περνάμε στις συναρτήσεις μπορούν να αλλάξουν τιμή (παράμετροι με αναφορά)!
- Όπως στους πίνακες, συνήθως στέλνουμε και το μέγεθος του πίνακα
  - Η συν/ση "μπορεί" να κάνει χρήση του '\0' για να βρει το τέλος
  - Δηλαδή το μέγεθος δεν είναι απαραίτητο αν η συνάρτηση δεν αλλάζει την παράμετρο C-string
  - Χρήση του "const" για προστασία των ορισμάτων C-string

Παραδείγματα

- Ποια είναι ισοδύναμα μεταξύ τους;

```
char stringVar[10] = "Γεια";
char stringVar[10] = {'Γ', 'ε', 'ι', 'α', '\0'};
char stringVar[10] = {'Γ', 'ε', 'ι', 'α'};
char stringVar[5] = "Γεια";
char stringVar[] = "Γεια";
```

- Υπάρχει κάποιο λάθος;

```
char stringVar[] = "Γεια";
strcat(stringVar," και αντίο.");
cout << stringVar ;
```

- Ποιο είναι το αποτέλεσμα:

```
char song[10] = "I did it ";
char fsong[20];
strcpy(fsong,song);
strcat(fsong,"my way!");
cout << fsong << endl ;
```

Ενότητα 21

ΕΙΣΟΔΟΣ/ΕΞΟΔΟΣ ΜΕ ΑΛΦΑΡΙΘΜΗΤΙΚΑ (C-STRING)

Έξοδος με C-string

- Μπορούμε να τα εκτυπώσουμε με τον τελεστή εισαγωγής <<
- Ήδη το χρησιμοποιούμε:
  cout << news << " Γεια.\n";
  - Όπου *news* είναι μια μεταβλητή C-string
- Προσοχή στον τελεστή << :
  σαν να συνενώνουμε C-strings!

Είσοδος με C-string

- Μπορούμε να εισάγουμε με τον τελεστή εξαγωγής >>
  - Ωστόσο υπάρχουν ορισμένα θέματα
- Το κενό " " θεωρείται ως "τέλος διαβάσματος"
  - Στηλοθέτης, κενό, εισαγωγή γραμμής: "παραλείπονται"
  - Το διάβασμα στην είσοδο σταματά στο "τέλος διαβάσματος"
- Προσοχή στο μέγεθος του C-string
  - Πρέπει να είναι μεγάλο ώστε να μπορεί να αποθηκεύσει την συμβολοσειρά που εισάγεται!
  - Η C++ δεν δίνει κάποια "προειδοποίηση" σε τέτοια θέματα!

Παράδειγμα εισαγωγής

```
char a[80], b[80];
cout << "Δώστε κάποια είσοδο: \n";
cin >> a >> b;
cout << a << b << "ΤΕΛΟΣ ΕΙΣΟΔΟΥ\n";
```

Παράδειγμα

```
Δώστε κάποια είσοδο:
Γεια και χαρά σου !
ΓειακαιΤΕΛΟΣ ΕΙΣΟΔΟΥ!
```

- Η C-string *a* αποθηκεύει: "Γεια"
- Η C-string *b* αποθηκεύει: "και"

Διάβασμα ολόκληρης γραμμής

- Μπορούμε να αποθηκεύσουμε ολόκληρη γραμμή (με κενά) σε ένα C-string
- Χρήση του getline():
  - μια προκαθορισμένη συνάρτηση στην κλάση cin
  - το δεύτερο όρισμα : μέγιστο αριθμό χαρακτήρων που θα διαβάσει

```
char a[80];
cout << "Δώστε κάποια είσοδο: \n";
cin.getline(a, 80);
cout << a << "ΤΕΛΟΣ ΕΙΣΟΔΟΥ\n";
```

Παράδειγμα1

```
Δώστε κάποια είσοδο:
Γεια και χαρά σου !
Γεια και χαρά σου !ΤΕΛΟΣ ΕΙΣΟΔΟΥ!
```

Περισσότερα για την getline()

- Μπορεί να αναφέρει το πλήθος των χαρακτήρων που θέλει να διαβάσει:

```
char a[5];
cout << "Δώστε κάποια είσοδο: \n";
cin.getline(a, 5);
cout << a << "ΤΕΛΟΣ ΕΙΣΟΔΟΥ\n";
```

Παράδειγμα2

```
Δώστε κάποια είσοδο:
Γειαχαρά
ΓειαΤΕΛΟΣ ΕΙΣΟΔΟΥ!
```

- Αναγκάζει ΤΕΣΣΕΡΙΣ χαρακτήρες μόνο να διαβάσει
  - Θυμηθείτε την ανάγκη για τον null ('\0') χαρακτήρα!

Παραδείγματα

```
char a[80], b[80];
cout << "Δώστε κάποια είσοδο: \n";
cin >> a >> b;
cout << a << b << "ΤΕΛΟΣ ΕΙΣΟΔΟΥ\n";
```

Παράδειγμα1

```
Δώστε κάποια είσοδο:
Έφτασε
        η ώρα!
????????????????????????
```

```
char mystring[80];
cout << "Δώστε κάποια είσοδο: \n";
cin.getline(mystring,6);
cout << mystring << "ΤΕΛΟΣ ΕΙΣΟΔΟΥ\n";
```

Παράδειγμα2

```
Δώστε κάποια είσοδο:
Όσα δε φέρνει ο χρόνος,
????????????????????????
```

Ενότητα 22

ΧΕΙΡΙΣΜΟΙ ΧΑΡΑΚΤΗΡΩΝ

Χαρακτήρες Ε/Ε

- Είσοδος και Έξοδος δεδομένων
  - ΟΛΑ τα χειριζόμαστε ως δεδομένα χαρακτήρων
  - π.χ., ο αριθμός 10 εκτυπώνεται ως '1' και '0'
  - Η μετατροπή γίνεται αυτόματα
    - Χρησιμοποιεί χαμηλού-επιπέδου χαρακτηριστικά
- Μπορούμε να χρησιμοποιήσουμε χαμηλού-επιπέδου στοιχεία

Η συνάρτηση get()

- Διαβάζει έναν χαρακτήρα (char) τη φορά
- Είναι συνάρτηση του αντικειμένου cin:

```
char nextSymbol;
cin.get(nextSymbol);
```

  - Διαβάζει τον επόμενο χαρακτήρα και τον αναθέτει στη μεταβλητή nextSymbol
  - Το όρισμα πρέπει να είναι τύπου char
    - ΌΧΙ "συμβολοσειρά"!

Η συνάρτηση put()

- Εκτυπώνει έναν χαρακτήρα τη φορά
- Ανήκει στο αντικείμενο cout :

```
cout.put('a');
```

  - Εκτυπώνει το γράμμα "a"

```
char myString[10] = "Hello";
cout.put(myString[1]);
```

  - Εκτυπώνει το γράμμα "e"

Παράδειγμα (1/2)

```
cout << "Δώστε μια γραμμή εισόδου: \n";
char symbol;
do
{
   cin.get(symbol);
   cout << symbol;
}
while( symbol != '\n');
```

Παράδειγμα

```
Δώστε μια γραμμή εισόδου:
Τρα λα λα 1 2   22
Τρα λα λα 1 2   22
```

Παράδειγμα (2/2)

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

void newLine( );

void getInt(int& number);

int main( )
{
    int n;

    getInt(n);
    cout << n ";

    return 0;
}
```

```
void newLine( )
{
    char symbol;
    do
    {
        cin.get(symbol);
    } while (symbol != '\n');
}

void getInt(int& number)
{
    char ans;
    do
    {
        cout << "Enter input: ";
        cin >> number;
       cout <<number<<" correct? (yes/no):";
        cin >> ans;
        newLine( );
    } while ((ans == 'N') || (ans == 'n'));
}
```

Παράδειγμα

```
Enter input: 57
57 correct? (yes/no): no no!
Enter input: 75
75 correct? (yes/no): yes
75
```

Περισσότερες συν/σεις χαρακτήρων

- putback()
  - Όταν διαβάζει τοποθετεί πάλι τον χαρακτήρα στην είσοδο
  - cin.putback(lastChar);
- peek()
  - Επιστρέφει τον επόμενο χαρακτήρα που θα διαβαστεί, αλλά τον αφήνει στην είσοδο
  - peekChar = cin.peek();
- ignore()
  - Παράλειψη της εισόδου, μέχρι έναν συγκεκριμένο χαρακτήρα
  - cin.ignore(1000, "\n");
    - Παραλείπει το πολύ 1000 χαρακτήρες μέχρι "\n"

Συναρτήσεις χειρισμού χαρακτήρων (1/3)

- Βρίσκονται στην βιβλιοθήκη cctype:

```
#include <cctype>
```

Display 9.3 Some Functions in <cctype>

FUNCTION	DESCRIPTION	EXAMPLE
`toupper(`*Char_Exp*`)`	Returns the uppercase version of *Char_Exp* (as a value of type `int`).	`char c = toupper('a');` `cout << c;` **Outputs:** A
`tolower(`*Char_Exp*`)`	Returns the lowercase version of *Char_Exp* (as a value of type `int`).	`char c = tolower('A');` `cout << c;` **Outputs:** *a*
`isupper(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is an uppercase letter; otherwise, returns `false`.	`if (isupper(c))` `  cout << "Is uppercase.";` `else` `  cout << "Is not uppercase.";`

Συναρτήσεις χειρισμού χαρακτήρων (2/3)

Display 9.3 **Some Functions in <cctype>**

FUNCTION	DESCRIPTION	EXAMPLE
`islower(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a lowercase letter; otherwise, returns `false`.	`char c = 'a';` `if (islower(c))` `  cout << c << " is lowercase.";` **Outputs:** `a is lowercase.`
`isalpha(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a letter of the alphabet; otherwise, returns `false`.	`char c = '$';` `if (isalpha(c))` `  cout << "Is a letter.";` `else` `  cout << "Is not a letter.";` **Outputs:** `Is not a letter.`
`isdigit(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is one of the digits `'0'` through `'9'`; otherwise, returns `false`.	`if (isdigit('3'))` `  cout << "It's a digit.";` `else` `  cout << "It's not a digit.";` **Outputs:** `It's a digit.`
`isalnum(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is either a letter or a digit; otherwise, returns `false`.	`if (isalnum('3') && isalnum('a'))` `  cout << "Both alphanumeric.";` `else` `  cout << "One or more are not.";` **Outputs:** `Both alphanumeric.`

Συναρτήσεις χειρισμού χαρακτήρων (3/3)

`isspace(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a whitespace character, such as the blank or newline character; otherwise, returns `false`.	`//Skips over one "word" and sets c` `//equal to the first whitespace` `//character after the "word":` `do` `{` `  cin.get(c);` `} while (! isspace(c));`
`ispunct(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a printing character other than whitespace, a digit, or a letter; otherwise, returns `false`.	`if (ispunct('?'))` `  cout << "Is punctuation.";` `else` `  cout << "Not punctuation.";`
`isprint(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a printing character; otherwise, returns `false`.	
`isgraph(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a printing character other than whitespace; otherwise, returns `false`.	
`isctrl(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a control character; otherwise, returns `false`.	

Παραδείγματα

```
char next;
do
{
   cin.get(next);
   if( isspace(next) )
      cout << '-';
   else
      cout << next;
}
while( next != '.');
```

Παράδειγμα

```
Εεεεε  γεια και χαρά σας.
Εεεεε--γεια-και-χαρά-σας.
```

```
char next;
do
{
   cin.get(next);
   cout << next;
}
while(!isdigit(next) && (next !='\n') );
```

Παράδειγμα

```
Θα σε δω στις 10:30μμ.
??????????????????????????
```

```
char next;
do
{
   cin.get(next);
   if( !isupper(next) )
      cout << next;
}
while( next != '\n');
```

Ενότητα 23

Η ΚΛΑΣΗ STRING

Η κλασική string στη C++

- Ορίζεται στη βιβλιοθήκη:
  #include <string>
  using namespace std;
- Μεταβλητές τύπου String και εκφράσεις
  - Όπως οι απλοί τύποι μεταβλητών
- Μπορεί να αναθέτει, να συγκρίνει, να προσθέτει:
  string s1, s2, s3;
  s3 = s1 + s2; //συνένωση
  s3 = "Hello Mom!" //ανάθεση
  - Σημειώστε ότι το C-string "Hello Mom!" μετατρέπεται αυτόματα σε τύπου string!

Παράδειγμα

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

int main( )
{
    string phrase;
    string adjective("τηγανιτά"), noun("μυρμήγκια");
    string wish = "Bon appetite!";

    phrase = "Μ΄ αρέσοουν " + adjective + " " + noun + "!";
    cout << phrase << endl
         << wish << endl;

    return 0;
}
```

Αρχικοποιείται στο κενό αλφαριθμητικό

Δύο ισδύναμοι τρόποι για απόδοση αρχικών τιμών

Παράδειγμα

```
M' αρέσουν τα τηγανιτά μυρμήγκια!
Bon appetite!
```

Ε/Ε με την κλάση String

- Ακριβώς όπως και σε άλλους τύπους!
- string s1, s2;
  cin >> s1;
  cin >> s2;
- Αποτέλεσμα:
  Ο χρήστης πληκτρολογεί:
  May the hair on your toes grow long and curly!
- Η εξαγωγή ακόμα παραβλέπει κενά:
  s1 έχει τιμή "May"
  s2 έχει τιμή "the"

getline() με την κλάση String

- Για ολόκληρες γραμμές:

```
string line;
cout << "Enter a line of input: ";
getline(cin, line);
cout << line << "END OF OUTPUT";
```

Παράδειγμα

```
Enter a line of input:
Γεια και χαρά σου!!
Γεια και χαρά σου!!END OF OUTPUT
```

- Ακριβώς όπως η getline() του C-string

Άλλες εκδόσεις της getline()

- Μπορούμε να ορίσουμε χαρακτήρα τέλους:

```
string line;
cout << "Enter input: ";
getline(cin, line, "?");
```

- Διαβάζει την είσοδο μέχρι να εμφανιστεί το "?"

Προσοχή στον συνδυασμό

- Προσοχή όταν αναμειγνύουμε cin >> var και getline()

```
int n;
string line;
cin >> n;
getline(cin, line);
```

  - Αν η είσοδος είναι: 42
    Hello hitchhiker.
    - Η μεταβλητή n έχει τιμή 42
    - Η line είναι η κενή συμβολοσειρά!!!!!
  - cin >> n
    παραλείπει κενούς χαρακτήρες, αφήνοντας τον χαρακτήρα "\n"
    στην είσοδο για την getline()!

Επεξεργασία μεταβλητών τύπου string

- Ίδιες λειτουργίες με τις C-strings
- και παραπάνω!
  - Περισσότερες από 100 συναρτήσεις της κλάσης string
- Ορισμένες συναρτήσεις:
  - .length()
    - επιστρέφει το μήκος της συμβολοσειράς
  - .at(i)
    - επιστρέφει αναφορά στον χαρακτήρα της θέσης i

Συναρτήσεις της κλάσης string (1/2)

Display 9.7 **Member Functions of the Standard Class `string`**

EXAMPLE	REMARKS
Constructors	
`string str;`	Default constructor; creates empty `string` object `str`.
`string str("string");`	Creates a `string` object with data `"string"`.
`string str(aString);`	Creates a `string` object `str` that is a copy of `aString`. `aString` is an object of the class `string`.
Element access	
`str[i]`	Returns read/write reference to character in `str` at index `i`.
`str.at(i)`	Returns read/write reference to character in `str` at index `i`.
`str.substr(position, length)`	Returns the substring of the calling object starting at `position` and having `length` characters.
Assignment/Modifiers	
`str1 = str2;`	Allocates space and initializes it to `str2`'s data, releases memory allocated for `str1`, and sets `str1`'s size to that of `str2`.
`str1 += str2;`	Character data of `str2` is concatenated to the end of `str1`; the size is set appropriately.
`str.empty( )`	Returns `true` if `str` is an empty `string`; returns `false` otherwise.

(continued)

Συναρτήσεις της κλάσης string (2/2)

Display 9.7 **Member Functions of the Standard Class `string`**

EXAMPLE	REMARKS
`str1 + str2`	Returns a `string` that has `str2`'s data concatenated to the end of `str1`'s data. The size is set appropriately.
`str.insert(pos, str2)`	Inserts `str2` into `str` beginning at position `pos`.
`str.remove(pos, length)`	Removes substring of size `length`, starting at position `pos`.
Comparisons	
`str1 == str2  str1 != str2`	Compare for equality or inequality; returns a Boolean value.
`str1 < str2  str1 > str2`	Four comparisons. All are lexicographical comparisons.
`str1 <= str2  str1 >= str2`	
`str.find(str1)`	Returns index of the first occurrence of `str1` in `str`.
`str.find(str1, pos)`	Returns index of the first occurrence of string `str1` in `str`; the search starts at position `pos`.
`str.find_first_of(str1, pos)`	Returns the index of the first instance in `str` of any character in `str1`, starting the search at position `pos`.
`str.find_first_not_of (str1, pos)`	Returns the index of the first instance in `str` of any character *not* in `str1`, starting search at position `pos`.

Μετατροπή από C-string σε string

- Αυτόματες μετατροπές τύπων
  - Από C-string σε string :
    char aCString[] = "My C-string";
    string stringVar;
    stringVar = aCstring;
    - Έγκυρο και κατάλληλο!!
  - aCString = stringVar;
    - ΜΗ ΝΟΜΙΜΟ!
    - Δεν μπορεί να μετατραπεί αυτόματα σε C-string
  - Πρέπει να κάνουμε χρήση μετατροπών:
    strcpy(aCString, stringVar.c_str());

Σύνοψη

- Η μεταβλητή C-string είναι "πίνακας από χαρακτήρες"
  - Με επιπλέον τον κενό χαρακτήρα '\0'
- Τα C-strings συμπεριφέρονται όπως οι πίνακες
  - Δεν μπορούν να αναθέσουν, να συγκρίνουν όπως οι απλές μεταβλητές
- Οι βιβλιοθήκες <cctype> & <string> έχουν χρήσιμες συν/σεις χειρισμού
- cin.get() διαβάζει τον επόμενο χαρακτήρα
- getline() επιτρέπουν διάβασμα ολόκληρης γραμμής
- Τα αντικείμενα string συμπεριφέρονται καλύτερα από τα απλά C-strings

Καλή Μελέτη

- **Βιβλιογραφία**

[1] W. Savitch, Πλήρης C++, Εκδόσεις Τζιόλα, 2011

[2] H. Deitel and P. Deitel, C++ Προγραμματισμός 6η Εκδοση, Εκδόσεις Μ. Γκιούρδας, 2013

Ύλη βιβλιογραφίας
[1]: 9.1, 9.2, 9.3
[2]: 6.7, 6.8, Κεφ. 18