Εισαγωγή στον Προγραμματισμό

Μέρος 4ο: Συμβολοσειρές

Εξάμηνο Σπουδών: 3ο

Κωδικός Μαθήματος: 343

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Μιχάλης Α. Μπέκος
bekos@uoi.gr

Εισαγωγή

- Συμβολοσειρές:
  - Πεπερασμένες ακολουθείες χαρακτήρων
  - Από τις πιο απλές δομές μετά τους βασικούς τύπους δεδομένων

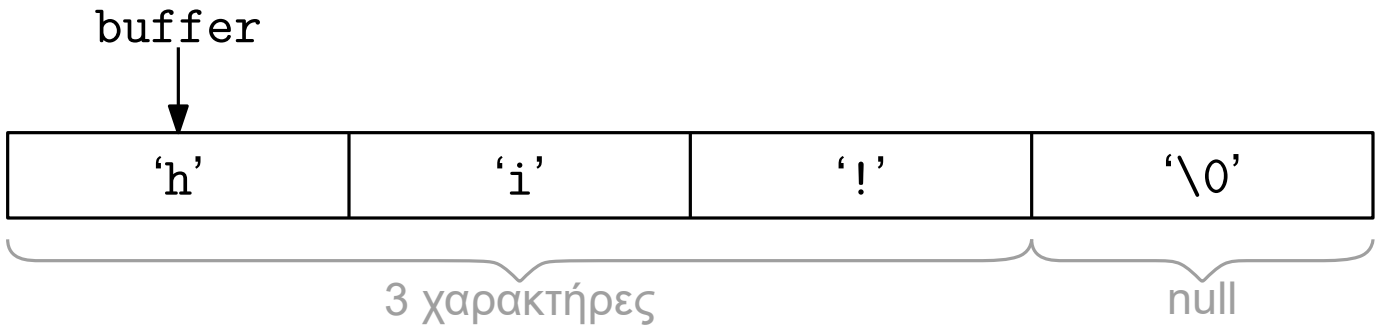
- Τύποι συμβολοσειρών:
  - Στην C: συμβολοσειρές ως πίνακες χαρακτήρων
  - Στην C++: συμβολοσειρές αντικείμενα

Μέρος 1°:
C-strings

C-strings

- Τα C-strings υλοποιούνται ως πίνακες χαρακτήρων που τερματίζονται με τον χαρακτήρα ‘\0’ (NULL).

```
char buffer[4];
strcpy(buffer, "hi!");
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 [π.χ., ”Hello World!”].

- Αν ένας πίνακας χαρακτήρων δεν τερματίζει με NULL, τότε δεν είναι C-string.

Αρχικοποίηση C-strings

- Η αρχικοποίηση μιας c-string γίνεται όμοια με αυτή ενός πίνακα.
- Ο τελευταίος χαρακτήρας θα πρέπει να είναι ο χαρακτήρας '\0' (NULL).
- Μπορούν επίσης να αρχικοποιηθούν χρησιμοποιόντας μια συμβολοσειρά σε "...".
  - Το μέγεθος του πίνακα μπορεί να παραληφθεί

    ```
    char saying[] = "A piece of cake";
    ```

  - Μόνο κατά τη δήλωση !

    ```
    char saying[20];
    saying = "A piece of cake"; // ILLEGAL
    ```

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

int main() {
  char vowels[6] = {'A','E','I',
                    'O','U','\0'};
  cout << "Vowels: " << vowels << endl;

  char scales[4] = {'K','F','C','\0'};
  cout << "Temperature scales: "
       << scales << endl;

  char digits[11] = {'0','1','2','3',
                     '4','5','6','7',
                     '8','9', '\0'};
  cout << "Digits: " << digits << endl;

  char saying[] = "A piece of cake";
  cout << "Saying: " << saying << endl;

  return 0;
}
```

Part04/cstrinit.cpp

Δείκτες C-strings

- Μια C-string **είναι** ένας πίνακα.
- Ο κάθε χαρακτήρας γίνεται προσβάσιμος μέσω του τελεστή `[ ]`
  - ```
    char word[] = "cute";
    word[0] = 'm';
    ```
- Αν ο χαρακτήρας `null` τροποποιηθεί, τότε δεν πρόκειται πλέον για C-string
  - Απρόβλεπτα αποτελέσματα

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

int main() {
  char greeting[5] = "hi!";

  // Hi!
  cout << greeting[0] << endl;
  cout << greeting[1] << endl;
  cout << greeting[2] << endl;

  // null
  cout << greeting[3] << endl;

  // unknown
  cout << greeting[4] << endl;

  // Better don't do
  greeting[3] = '!';

  return 0;
}
```

Part04/cstrindices.cpp
```

C-strings ως ορίσματα και παράμετροι συναρτήσεων

- Υπενθύμιση: Οι C-strings είναι πίνακες
- Άρα, ως παράμετροι συναρτήσεων τυπικό είναι να στέλνεται και το μέγεθος τους
  - Εναλλακτικά, χρησιμοποιείται το '\0' για τον εντοπισμό του τέλους
- Χρησιμοποιήστε τον προσδιοριστή `const` για την αποφυγή τροποποιήσεων των c-string ορισμάτων μιας συνάρτησης

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

void display(const char[]);

int main()
{
  char message[100] = "Hi there!";
  display(message);

  return 0;
}

void display(const char s[])
{
  cout << "Entered char array is: ";
  for (int i=0; ; i++) {
    if (s[i] != '\0')
      cout << s[i];
    else
      break;
  }
  cout << endl;
}
```

Part04/cstrparam.cpp

Η βιβλιοθήκη <cstring>

- Περιέχει συναρτήσεις για συνήθεις λειτουργίες συμβολοσειρών όπως αντιγραφή, συνένωση, μήκος, αναζήτηση, διάσπαση c-string σε τμήματα κ.α.
  - `strcpy()`, `strcat()`, `strlen()`, `strcmp()`, `strncat()`, `strncmp()`, `strstr()`, `strtok()`,...
- Περισσότερα:
  - https://cplusplus.com/reference/cstring/

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

int main () {
 char s1[12] = {'G','o','o','d','\0'};
 char s2[12] = "Morning";
 char s3[12];
 // copy s1 into s3 (cannot use =)
 strcpy(s3, s1);
 cout << "strcpy(s3,s1): " << s3 << endl;

 // concatenate s1 and s2 (cannot use +=)
 strcat(s1, s2);
 cout << "strcat(s1,s2): " << s1 << endl;

 // lenghth of s1 after concatenation
 int len = strlen(s1);
 cout << "strlen(s1): " << len << endl;

 // comparing s1 and s2 (cannot use ==)
 int result = strcmp(s1, s2);
 cout << "strcmp(s1, s2):" << result;
}
```

Part04/cstrfunctions.cpp

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.
- Ωστόσο, υπάρχουν διάφορα θέματα:
  - Στον τελεστή εξαγωγής (>>) το κενό θεωρείται ως “τέλος διαβάσματος”, ενώ η εισαγωγή νέα γραμμής δεν υποστηρίζεται
  - Προσοχή στο μέγεθος του C-string

```
#include <iostream>
#include <iomanip> // for setw
#include <cstring>
using namespace std;

int main() {
  const int MAX = 80;
  char s1[MAX], s2[MAX], s3[MAX];

  //here blanks are problematic
  cout <<"Enter a string: ";
  cin >> setw(MAX) >> s1;

  //here blanks are embedded
  cout << "Enter another string: ";
  cin.get(s2, MAX);

  //support for multiple lines
  cout << "Enter one more ($ to end): ";
  cin.get(s3, MAX, '$');

  cout << "You entered: " << s1 << ", "
       << s2 << " and "  << s3 << endl;
}
```

Part04/cstringin.cpp

Παραδείγματα

- Ποια είναι ισοδύναμα μεταξύ τους;

```
char stringVar[10] = "Bye!";
char stringVar[10] = {'B', 'y', 'e', '!', '\0'};
char stringVar[10] = {'B', 'y', 'e', '!'};
char stringVar[5] = "Bye!";
char stringVar[]  = "Bye!";
```

- Ποιο είναι το αποτέλεσμα;

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

Μειονεκτήματα των C-strings

- Οι συμβολοσειρές της C δεν είναι πάντοτε αποδοτικότερες.
  - Π.χ., η συνάρτηση `strlen()` δεν έχει σταθερό κόστος, αλλά λογαριθμικό ως προς το μήκος της συμβολοσειράς.
- Σταθερό μέγεθος (ορίζεται όταν δηλώνεται το C-string ως στατικός πίνακας).
- Τα όρια του πίνακα δεν επιβάλλονται με κάποιο τρόπο.
- Το όνομα του C-string λειτουργεί ως δείκτης.

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

int main() {
  char input[]="Oh, Captain, my Captain";

  const int MAX = 80;
  char copy[MAX];

  // copy using a for loop
  int i;
  for (i=0; i<strlen(input); i++)
    copy[i] = input[i];
  copy[i] = '\0';//insert NULL at the end

  //Alternatively:
  //strcpy(copy, input);

  cout << "copy = " << copy << endl;

  return 0;
}
```

Part04/cstrcopy.cpp

Μειονεκτήματα των C-strings

- Χρησιμοποιούν “άβολες” συναρτήσεις αντί για εύκολα κατανοητούς τελεστές.
  - `strcpy(str1, srt2)` αντί για `str1 = str2`
  - `strcmp(str1, srt2)` αντί για `str1 == str2`
  - `strcat(str1, srt2)` αντί για `str1 += str2`
- Η χρήση του NULL χαρακτήρα μπορεί να δημιουργήσει προβλήματα.
- Stroustrup: “... η συμβολοσειρά της C είναι ένα σύνολο συμβάσεων που υποστηρίζονται από λίγες χρήσιμες συναρτήσεις.”

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

int main() {
  char input[]="Oh, Captain, my Captain";

  const int MAX = 80;
  char copy[MAX];

  // copy using a for loop
  int i;
  for (i=0; i<strlen(input); i++)
    copy[i] = input[i];
  copy[i] = '\0';//insert NULL at the end

  //Alternatively:
  //strcpy(copy, input);

  cout << "copy = " << copy << endl;

  return 0;
}
```

Part04/cstrcopy.cpp

Μέρος 2°:
std::string

std::string

- Ορίζονται στη βιβλιοθήκη <string>, η οποία πρέπει να περιληφθεί στο πρόγραμμα σας, αν σκοπεύετε να τις χρησιμοποιήσετε:
  - #include <string>
- Στην βιβλιοθήκη αυτή, η συμβολοσειρά ορίζεται ως αντικείμενο (σ.σ., υποστηρίζει μεθόδους)
- Είναι πολύ βολική και κάνει την επεξεργασία των συμβολοσειρών πιο εύκολη από ότι στη C
- https://cplusplus.com/reference/string/string/

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

int main() {
  //initialization methods
  string s1("Man");
  string s2 = "Beast";

  //assignment
  string s3 = s1;
  cout << "s3 = " << s3 << endl;

  //concatenations
  s3 = "Neither " + s1 + " nor " + s2;
  cout << "s3 = " << s3 << endl;

  //swap s1 and s2
  s1.swap(s2);
  s3 = "Neither " + s1 + " nor " + s2;
  cout << "s3 = " << s3 << endl;

  return 0;
}
```

Part04/sstrass.cpp

std::string - Είσοδος και έξοδος

- Ο χειρισμός I/O των std::string γίνεται όπως αυτός των C-strings:
  - Με τους τελεστές εισαγωγής (<<) και εξαγωγής (>>)
  - Μέσω της συνάρτησης `getline()`

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

int main () {
  string fullname, nickname, address;
  string gretting("Hello ");

  cout << "Enter your full name: ";
  getline(cin, fullname);
  cout << "Your name is " << fullname
       << endl;

  cout << "Enter your nickname: ";
  cin >> nickname;
  gretting += nickname;
  cout << gretting << endl;

  cout << "Enter your address ";
  cout << "(terminate with '$'): ";
  getline(cin, address, '$');
  cout << "Your address is: " << address
       << endl;
}
```

Part04/sstrio.cpp

std::string - Χρήσιμοι τελεστές

- `=` Αναθέτει την τιμή της δεξιάς συμβολοσειράς στην αριστερή
- `+=` Παραθέτει στο τέλος της αριστερής συμβολοσειράς τη δεξιά
- `+` Επιστρέφει την παράθεση των δύο συμβολοσειρών
- `[ ]` Πρόσβαση στους χαρακτήρες της συμβολοσειράς
- `<<` Γράφει στην έξοδο τη δοθείσα συμβολοσειρά
- `>>` Διαβάζει από την είσοδο μια συμβολοσειρά (whitespace-delimited)

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

int main() {
  string word1, phrase;
  string word2 = " Dog";

  cout << "Enter a word: ";
  cin >> word1;      //user enters "Hot"

  phrase = word1 + word2;   // "Hot Dog"
  phrase += " on a bun";

  for (int i=0; i<phrase.length(); i++)
  {
    cout << phrase[i];
  }
  cout << endl;

  return 0;
}
```

Part04/sstrchng.cpp

std::string - Χρήσιμες μέθοδοι

- `int find(string str):`
  επιστρέφει τη θέση της str στη συμβολοσειρά
- `int find(char ch):`
  επιστρέφει τη θέση του χαρακτήρα ch στη συμβολοσειρά
- `int find(string str, int x):`
  επιστρέφει τη θέση της str στη συμβολοσειρά μετά τη θέση x
- `int find(char ch, int x):`
  επιστρέφει τη θέση του χαρακτήρα ch μετά τη θέση x
- `void insert(int x, string str):`
  εισάγει τη συμβολοσειρά str στη θέση x
- `void insert(int x, char ch):`
  εισάγει τον χαρακτήρα ch στη θέση x
- `void append(string str):`
  παραθέτει τη συμβολοσειρά str στο τέλος

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

int main() {
  string x = "FROM:someone@uoi.gr";

  //find the first occuence of ':'
  int colonPos = x.find(':');

  //substrings of the input
  string prefix = x.substr(0, colonPos);
  string suffix = x.substr(colonPos + 1);

  //output a part
  cout << "This message is from "
       << suffix << endl;

  //alternatively, one could try to do
  //the same using C-strings
  char y[x.length()+1];
  strcpy(y, x.c_str());
  //...
}
```

Part04/sstrfind.cpp

std::string - Λεξικογραφική σύγκριση

- Η σύγκριση συμβολοσειρών γίνεται μέσω των τελεστών σύγκρισης

  `<`, `<=`,

  `>`, `>=`,

  `==`, `!=`

- Η σύγκριση είναι λεξικογραφική
- Το αποτέλεσμα της είναι `true` ή `false`
- Η σύγκριση υποθέτει ότι τουλαχιστον ένας από τους τελεστέους είναι std::string. Ό αλλος μπορεί να είναι std::string, ή C-string.

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

int main()
{
  string aName = "George";
  string userName;

  cout << "Enter your first name : ";
  cin >> userName;

  if (userName == aName) {
    cout << "Greetings, George" << endl;
  }
  else if (userName < aName) {
    cout << "You come before George"
         << endl;
  }
  else {
    cout << "You come after George"
         << endl;
  }
}
```

Part04/sstrcom.cpp

Πλεονεκτήματα `std::string` έναντι C-strings

- Μεταβλητό μέγεθος
- Εύρεση μήκους σε σταθερό χρόνο (και όχι σε λογαριθμικό)
- Δεν απαιτούν εντολές διαχείρισης μνήμης
- Αυτόματος χειρισμός των ορίων των λεκτικών (δεν απαιτείται τερματικός χαρακτήρας)
- Διαισθητικά εύκολη ανάθεση τιμής με το = αντί για το `strcpy`
- Διαισθητικά εύκολη σύγκριση με το == αντί για το `strcmp`
- Διαισθητικά εύκολη παράθεση λεκτικών με το + αντί για το `strcat`
- Μετατροπή σε C-string με τη συνάρτηση μέλος `c_str`

Μέρος 3°:
Παραδείγματα

Συχνότητα χαρακτήρων σε μια συμβολοσειρά

- **Πρόβλημα:** Να γραφεί μια συνάρτηση, η οποία δοθέντος μιας συμβολοσειράς `str` και ενός χαρακτήρα `ch` επιστρέφει το πλήθος των εμφανίσεων του `ch` στην `str`.

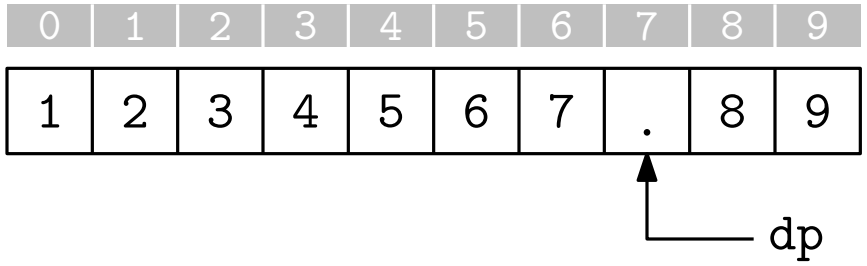
- **Λύση:** Με διαπέραση των χαρακτήρων της συμβολοσειράς.

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

// Determines character's frequency
int chfrequency(string& str, char& c);

int main () {
  string moto("C++ is awesome");
  char checkChar = 'a';
  int cnt = chfrequency(moto, checkChar);

  cout << "Frequency = " <<
       << to_string(cnt) << endl;
  return 0;
}

int chfrequency(string& str, char& c) {
  int frequency = 0;
  for (int i=0; i<str.length(); i++)
    if (str[i] == c)
      frequency++;
  return frequency;
}
```

Part04/sstrchfreq.cpp

Μορφοποίηση συμβολοσειρών

- **Πρόβλημα:** Να γραφεί μια συνάρτηση, η οποία να μορφοποιεί μια συμβολοσειρά όπως η ”1234567.89” σε ”$1,234,567.89”.
- **Λύση:** Με τη χρήση ενός δείκτη προς τον αριστερότερο μη-αριθμητικό χαρακτήρα (σ.σ., ΄.΄ ή ’,’).




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

// Formats a string like "1234567.89"
// to "$1,234,567.89".
void format(string& str);

int main () {
  string x("1234567.89");
  format(x);

  cout << "Formatted string is: " << x;
  return 0;
}

void format(string& str) {
  int dp = str.find('.');
  while (dp > 3) {
    str.insert(dp-3, ",");
    dp = str.find(',');
  }
  str.insert(0,"$");
}
```

Part04/sstrformat.cpp

Η δική σας σειρά

- **Πρόβλημα:** Να γραφεί μια συνάρτηση, η οποία δοθέντος μιας συμβολοσειράς επιστρέφει το πλήθος των λέξεων της.
- **Παραδοχή:** Όλες οι λέξεις οριοθετούνται από ένα μόνο διάστημα.
- **Λύση:** ???