Προγραμματισμός Υπολογιστών

Αλφαριθμητικά

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Αλφαριθμητικά (Strings)

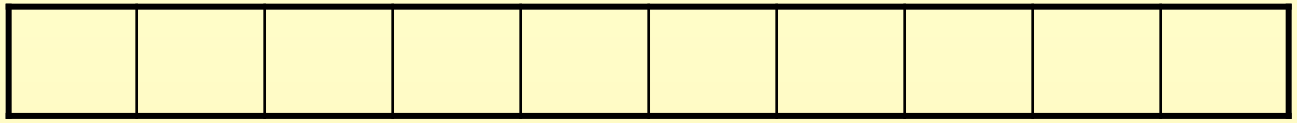
Σε όλα σχεδόν τα προγράμματα χρησιμοποιούμε μεταβλητές που είναι τύπου αλφαριθμητικά με σκοπό να αποθηκεύσουμε ονόματα, διευθύνσεις κ.λπ. Αυτού του τύπου τα δεδομένα έχουν σαν βασικό στοιχείο τους χαρακτήρες. Στην γλώσσα C δεν υπάρχει συγκεκριμένος τύπος δεδομένων για τις συμβολοσειρές αλλά αναπαριστώνται σαν πίνακες από χαρακτήρες. Έτσι λοιπόν η δήλωση μιας συμβολοσειράς με το όνομα address και δυνατότητα αποθήκευσης 10 χαρακτήρων θα γίνει όπως παρακάτω :

```
char address[10];
```

Η μεταβλητή address αναπαριστάται όπως κάθε πίνακας με συνεχόμενες θέσεις χαρακτήρων και η αρίθμηση των θέσεων ξεκινά από το μηδέν.

Κάθε συμβολοσειρά μπορεί κατά την διάρκεια του προγράμματος να αποθηκεύσει διαφορετικές τιμές, οι οποίες να έχουν μεταβλητό μήκος. Για παράδειγμα στην μεταβλητή address μπορούμε αρχικά να αποθηκεύσουμε την τιμή Good day και αργότερα να αποθηκεύσουμε την τιμή hello. Το συνολικό μήκος της επιθυμητής συμβολοσειράς δηλώνεται με την πρώτη εμφάνιση του χαρακτήρα '\0' ο οποίος ονομάζεται και null χαρακτήρας.

Οι συνολικές θέσεις της μεταβλητής address



Η μεταβλητή address μετά την απόδοση της τιμής Good day

G	o	o	d		d	a	y	\0	

Η μεταβλητή address μετά την απόδοση της τιμής hello

h	e	l	l	o	\0	a	y	\0	

Την επεξεργασία μιας συμβολοσειράς, π.χ. εκτύπωση, αντιγραφή κ.λπ., την διακόπτουμε μόλις συναντήσουμε το χαρακτήρα '\0'. Πρακτικά μπορούμε να καθαρίσουμε τα περιεχόμενα μιας συμβολοσειράς ως εξής :

```
address[0] = '\0';
```

Η αρχικοποίηση μιας συμβολοσειράς μπορεί να γίνει μόνο κατά την στιγμή της δήλωσης της. Για παράδειγμα

```
char address[] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

Εναλλακτικά μπορούμε να γράψουμε

```
char address[] = "Hello";
```

Οι δύο ανωτέρω εκφράσεις είναι ισοδύναμες και έχουν συνολικό μήκος 6 θέσεις (μετράμε και το χαρακτήρα null). Στην πρώτη περίπτωση η δήλωση του χαρακτήρα '\0' γίνεται συγκεκριμένα ενώ στην δεύτερη περίπτωση η δήλωση των διπλών εισαγωγικών " προσθέτει το χαρακτήρα '\0' σαν έναν επιπλέον χαρακτήρα.

Στις συμβολοσειρές όπως και σε άλλους τύπους πινάκων δεν μπορούμε να αποδώσουμε σαν τιμή έναν άλλο πίνακα υπονοώντας ότι κάθε στοιχείο του πρώτου πίνακα να πάρει σαν τιμή το αντίστοιχο στοιχείο του δευτέρου.

Η C διαθέτει την βιβλιοθήκη string.h η οποία περιέχει αρκετές συναρτήσεις για την διαχείριση των συμβολοσειρών.

Έτσι λοιπόν η απόδοση τιμής σε μια συμβολοσειρά μπορεί να γίνει είτε χαρακτήρα προς χαρακτήρα είτε με την χρήση της συνάρτησης strcpy

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
  char address[10];

  address[0] = 'G';    address[1] = 'o';    address[2] = 'o';    address[3] = 'd';
  address[4] = ' ';    address[5] = 'd';    address[6] = 'a';    address[7] = 'y';
  address[8] = '\0';
  printf("%s", address);
  strcpy(address, "The boy");
  printf("%s", address);
  return 0;
}
```

Η μορφοποίηση %s δηλώνει ότι θα γίνει εκτύπωση αλφαριθμητικού.

Τις λειτουργίες της βιβλιοθήκης string.h μπορούμε να τις επιτύχουμε με την συγγραφή και δικών μας συναρτήσεων. Π.χ η λειτουργία της συνάρτησης strcpy μπορεί να γίνει όπως παρακάτω :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void MyStrcpy (char strDest [], char strSource []) {
  int n=0;

  do {
    strDest[n] = strSource[n];
  } while (strSource[n++] != '\0');
}

int main(int argc, char *argv[]) {
    char address[10];

    MyStrcpy(address, "hello");

    printf("%s", address);
   return 0;
}
```

Χρήσιμες Συναρτήσεις στην βιβλιοθήκη string.h

char * strcat (char * *dest*, const char * *src*);

Η strcat προσαρτά τη συμβολοσειρά src στην συμβολοσειρά dest και επιστρέφει την συμβολοσειρά dest.

```
int main(int argc, char *argv[]) {
  char test[30] = "Good ";
  strcat(test, "morning ");
  strcat(test, "nick");
  printf("%s\n", test);
  return 0;
}
```

```
int main(int argc, char *argv[]) {
  char test[30] = "Good ";

  printf("%s\n", strcat(test, "day") );

  return 0;
}
```

Η συνάρτηση δεν δεσμεύει δυναμικά μνήμη και θα πρέπει η συμβολοσειρά dest να έχει αρκετές θέσεις για την προσάρτηση.

size_t strlen (const char * *string*);

Επιστρέφει το πλήθος των χαρακτήρων του string χωρίς να μετρά τον χαρακτήρα τερματισμού.

Το size_t ορίζεται σαν ένας unsigned integer τουλάχιστον 16 bit

```
int main(int argc, char *argv[]) {
  char test[30] = "Good";
  printf("%d\n", strlen(test) );
  return 0;
}
```

Εκτυπώνει 4

char * strchr (const char * *string*, int *c*);

Επιστρέφει ένα δείκτη στην πρώτη εμφάνιση του χαρακτήρα c μέσα στο string. Αν δεν υπάρχει ο χαρακτήρας τότε επιστρέφει το NULL.

```
int main(int argc, char *argv[]) {
 char test[30] = "Good Morning";
  char *s = strchr(test, 'r');
  printf("%s\n", s);
 return 0;
}
```

Εκτυπώνει rning

char * strstr (const char * *string1*, const char * *string2*);

Επιστρέφει ένα δείκτη στην πρώτη εμφάνιση της συμβολοσειράς string2 μέσα στη συμβολοσειρά string1. Αν δεν υπάρχει το string2 τότε επιστρέφει το NULL.

```
int main(int argc, char *argv[]) {
 char test[30] = "Good Morning";
  char *s = strstr(test, "or");
  printf("%s\n", s);
 return 0;
}
```

Εκτυπώνει orning

int strcmp (const char * *string1*, const char * *string2*);

Η συνάρτηση strcmp συγκρίνει το string1 με το string2 και επιστρέφει :

-1 string1 προηγείται του string2
0 string1 ίδιο με το string2
1 string1 έπεται του string2

```
int main(int argc, char *argv[]) {
 char strA[10] = "A";
 char strB[20] = "B";
 char strC[5] = "A";

 printf("%d\n", strcmp(strA, strB));
 printf("%d\n", strcmp(strA, strC));
 printf("%d\n", strcmp(strB, strA));
 return 0;
}
```

void * memset (void * *buffer*, int *c*, size_t *num*);

Γεμίζει ένα καθορισμένο πλήθος θέσεων ενός string με ένα συγκεκριμένο χαρακτήρα.

```
int main(int argc, char *argv[]) {
 char strA[10] = "AAAAA";

 memset(strA, '9', 2);
 printf("%s\n", strA);
 return 0;
}
```

Εκτυπώνει 99AAA