Προγραμματισμός Ι (ECE115)

#12 συναρτήσεις strings

Χρήστος Αντωνόπουλος
cda@uth.gr

Βασικές συναρτήσεις της βιβλιοθήκης string

```
#include <string.h>

size_t strlen(const char *s);

int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t n);

char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t n);

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

char *strchr(const char *str, int c);
char *strrchr(const char *str, int c);

char *strstr(const char *str, const char *find);
char *strpbrk(const char *str, const char *any);
```

strlen

```
size_t strlen(const char *s);
```

- επιστρέφει τον αριθμό των χαρακτήρων μέχρι (αλλά χωρίς) το τερματικό στοιχείο (null byte)
- ο πίνακας που περνιέται σαν παράμετρος **πρέπει** να έχει τουλάχιστον ένα τερματικό στοιχείο

strcmp / strncmp

```
int strcmp(const char *s1, const char *s2);
```

- συγκρίνει τα περιεχόμενα του `s1` με τα περιεχόμενα του `s2`
- αν το `s1` είναι **μικρότερο** του `s2`, επιστρέφεται `< 0`
- αν το `s1` είναι **μεγαλύτερο** του `s2`, επιστρέφεται `> 0`
- αν το `s1` είναι **ίδιο** με το `s2`, επιστρέφεται `0`
- η σύγκριση γίνεται με βάση την κωδικοποίηση ASCII
- μέχρι να βρεθεί ένα τερματικό στοιχείο (null byte)

```
int strncmp(const char *s1, const char *s2,
            size_t n);
```

- η σύγκριση τερματίζεται **το αργότερο** μετά από `n` χαρακτήρες
- εκτός και αν νωρίτερα βρεθεί ένα τερματικό στοιχείο

```c
/* σύγκριση string */

#include <stdio.h>
#include <string.h>
#define N 32

int main(int argc, char *argv[]) {
  char s1[N], s2[N];
  int res;

  scanf("%31s %31s", s1, s2);

  printf("%s is %ld long\n", s1, strlen(s1));
  printf("%s is %ld long\n", s2, strlen(s2));

  res = strcmp(s1,s2);
  if (res < 0)
    printf("%s is smaller than %s\n", s1, s2);
  else if (res > 0)
     printf("%s is larger than %s\n", s1, s2);
  else
    printf("%s is equal to %s\n", s1, s2);

  return(0);
}
```

strcpy / strncpy

```
char *strcpy(char *dest, const char *src);
```

- αντιγράφει το `src` **πάνω** στο `dest`
- το `src` **πρέπει** να έχει τερματικό, που αντιγράφεται στο `dest`
- τα `src` και `dest` **δεν** πρέπει να έχουν επικαλυπτόμενη μνήμη

```
char *strncpy(char *dest, const char *src,
              size_t n);
```

- αντιγράφει το πολύ `n` χαρακτήρες από το `src` **πάνω** το `dest`
- αν το μήκος του `src` είναι **μικρότερο** από `n` τότε γράφονται **επιπλέον** null bytes έτσι ώστε να γραφτούν συνολικά `n` bytes
- αν **δεν** υπάρχει τερματικό null byte μέσα στους πρώτους `n` χαρακτήρες του `src` τότε **δεν** θα γραφτεί τερματικό στο `dest`
- τα `src` και `dest` **δεν** πρέπει να έχουν επικαλυπτόμενη μνήμη

strcat / strncat

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

- αντιγράφει το `src` **επιμηκύνοντας** αντίστοιχα το `dest`
- το `dest` **πρέπει** να έχει τερματικό, που είναι και το σημείο πάνω στο οποίο αρχίσει η αντιγραφή των περιεχομένων του `src`
- το `src` **πρέπει** να έχει τερματικό, που αντιγράφεται στο `dest`
- τα `src` και `dest` **δεν** πρέπει να έχουν επικαλυπτόμενη μνήμη

```
char *strncat(char *dest, const char *src,
              size_t n);
```

- αντιγράφει έως `n` χαρακτήρες από το `src` **επιμηκύνοντας** αντίστοιχα το `dest`, προσθέτοντας και το τερματικό null byte
- αν το `src` έχει τουλάχιστον `n` χαρακτήρες τότε **δεν** χρειάζεται να έχει τερματικό null byte
- τα `src` και `dest` **δεν** πρέπει να έχουν επικαλυπτόμενη μνήμη

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

#define N 32

int main(int argc, char *argv[]) {
  char s1[N], s2[N], s3[3*N];
  int res;

  scanf("%31s %31s", s1, s2);
  printf("s1 is %s and s2 is %s\n", s1, s2);

  strcpy(s3, s1);
  printf("s3 is %s\n", s3);

  strcat(s3, s2);
  printf("s3 is %s\n", s3);

  strcat(s3, s1);
  printf("s3 is %s\n", s3);

  return(0);
}
```

strchr / strrchr

```
char *strchr(const char *str, int c);
```

- επιστρέφει δείκτη στην θέση του `src` όπου συναντάται για **πρώτη φορά** ο χαρακτήρας με τιμή `c`
- αν δεν βρεθεί χαρακτήρας με τιμή `c`, επιστρέφεται `NULL`

```
char *strrchr(const char *str, int c);
```

- επιστρέφει δείκτη στην θέση του `src` όπου συναντάται για **τελευταία φορά** ο χαρακτήρας με τιμή `c`
- αν δεν βρεθεί χαρακτήρας με τιμή `c`, επιστρέφεται `NULL`

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

#define N 64

int main(int argc, char *argv[]) {
  char str[N], c, *p;

  scanf("%63s %c", str, &c);

  p = strchr(str, c);
  if (p == NULL) {
    printf("%c not found\n", c);
  }
  else {
    printf("%c first found at pos %ld\n", c, p - str);
  }

  p = strrchr(str, c);
  if (p == NULL) {
    printf("%c not found\n", c);
  }
  else {
    printf("%c last found at pos %ld\n", c, p - str);
  }

  return(0);
}
```

strstr / strpbrk

```
char *strstr(const char *str, const char *find);
```

- επιστρέφει δείκτη στην θέση του `src` όπου συναντάται για (πρώτη φορά) **ολόκληρη** η συμβολοσειρά `find`
- αν δεν βρεθεί η συμβολοσειρά `find`, επιστρέφεται `NULL`

```
char *strpbrk(const char *str, const char *any);
```

- επιστρέφει δείκτη στην θέση του `src` όπου συναντάται για (πρώτη φορά) **κάποιος** από τους χαρακτήρες που βρίσκονται στην συμβολοσειρά `any`
- αν δεν βρεθεί κανένας χαρακτήρας της `any`, επιστρέφεται `NULL`

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

#define N 64

int main(int argc, char *argv[]) {
  char str[N], match[N], *p;

  scanf("%63s %63s", str, match);

  p = strstr(str, match);
  if (p == NULL) {
    printf("%s not found\n", match);
  }
  else {
    printf("%s found at pos %ld\n", match, p - str);
  }

  p = strpbrk(str, match);
  if (p == NULL) {
    printf("no char in %s found\n", match);
  }
  else {
    printf("%c found at pos %ld\n", *p, p - str);
  }

  return(0);
}
```

Συναρτήσεις βιβλιοθήκης stdio για εκτύπωση σε / ανάγνωση από strings

```
#include <stdio.h>

int sprintf(char *str, const char *format, ...);

int snprintf(char *str, size_t size, const char *format, ...);

int sscanf(const char *str, const char *format, ...);
```

sprintf / snprintf / sscanf

```
int sprintf(char *str, const char *format, ...);
```

- λειτουργεί όπως η `printf`
- τα δεδομένα **δεν** γράφονται την έξοδο αλλά στο `str`
- στο τέλος γράφεται **και** το τερματικό

```
int snprintf(char *str, size_t size,
             const char *format, ...);
```

- λειτουργεί όπως η `sprint`
- στο `str` γράφονται **το πολύ** `size` χαρακτήρες
- συμπεριλαμβανομένου **και** του τερματικού

```
int sscanf(const char *str, const char *format, ...);
```

- λειτουργεί όπως η `scanf`
- τα δεδομένα **δεν** διαβάζονται από την είσοδο αλλά από το `str`

```c
#include <stdio.h>

#define N 64

int main(int argc, char *argv[]) {
  char s1[N], s2[N];
  int v, res;
  float f;

  sprintf(s1, "%s %d %f", "hello_world", 123, 4.567);
  printf("%s\n", s1);

  res = sscanf(s1, "%s %d %f", s2, &v, &f);
  printf("scanned %d symbols: %s %d %f\n", res, s2, v, f);

  snprintf(s1, 15, "%s %d %lf", "hello_world", 123, 4.567);
  printf("%s\n", s1);

  res = sscanf(s1, "%s %d %f", s2, &v, &f);
  printf("scanned %d symbols: %s %d %f\n", res, s2, v, f);

  return(0);
}
```