Προγραμματισμός ΙΙ

Ακολουθίες Χαρακτήρων

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Χαρακτήρες και Πίνακες Χαρακτήρων

Μία συμβολοσειρά (string) είναι ένας πίνακας χαρακτήρων που τελειώνει με τον κενό χαρακτήρα (null character) `'\0'`.

```
char s[12] = "hello world";
```

Για να προσπελάσουμε μια συμβολοσειρά χρειαζόμαστε ενα δείκτη στο πρώτο στοιχείο της (χαρακτήρα).

Στο παραπάνω παράδειγμα ο identifier `s` είναι συνώνυμο με την διεύθυνση μνήμης του πρώτου χαρακτήρα.

Χαρακτήρες και Πίνακες Χαρακτήρων

Η συμβολοσειρά ”a” δεν είναι το ίδιο με τον χαρακτήρα ’a’.

Ο χαρακτήρας είναι

```
char c = 'a';
```

ενώ η συμβολοσειρά

```
char c[2] = { 'a', '\0' } ;
```

Βιβλιοθήκη Χειρισμού Χαρακτήρων

Η C μας παρέχει συναρτήσεις για τον χειρισμό χαρακτήρων.

- Η βιβλιοθήκη διαχείρισης χαρακτήρων περιέχει πολλές συναρτήσεις που εκτελούν ελέγχους και διαχειρίζονται δεδομένα χαρακτήρων
- Οι χαρακτήρες θεωρούνται ως ακέραιοι μήκους ενός byte
- Όταν χρησιμοποιούμε συναρτήσεις διαχείρισης χαρακτήρων πρέπει να κάνουμε

```
#include <ctype.h>
```

Βιβλιοθήκη Χειρισμού Χαρακτήρων

Κυριότερες Συναρτήσεις

- `int isdigit( char c )`
  true αν ο χαρακτήρας c είναι ψηφίο, αλλιώς false
- `int isalpha( char c )`
  true αν ο χαρακτήρας c είναι γράμμα, αλλιώς false
- `int isalnum( char c )`
  true αν ο χαρακτήρας c είναι γράμμα ή ψηφίο, αλλιώς false
- `int islower( char c )`
  true αν ο χαρακτήρας c είναι πεζό γράμμα, αλλιώς false

Βιβλιοθήκη Χειρισμού Χαρακτήρων

Κυριότερες Συναρτήσεις

- `int isupper( char c )`
  true αν ο χαρακτήρας c είναι κεφαλαίο γράμμα, αλλιώς false
- `int ispunct( char c )`
  true αν ο χαρακτήρας c είναι σημείο στίξης, αλλιώς false
- `int tolower(int c)`
  αν ο χαρακτήρας c είναι κεφαλαίο γράμμα επιστρέφει το αντίστοιχο πεζό, αλλιώς επιστρέφει τον c ανέπαφο
- `int toupper(int c)`
  αν ο χαρακτήρας c είναι μικρό γράμμα επιστρέφει το αντίστοιχο κεφαλαίο, αλλιώς επιστρέφει τον c ανέπαφο

Βιβλιοθήκη Χειρισμού Χαρακτήρων

Κυριότερες Συναρτήσεις

- `int isspace(int c)`
  επιστρέφει true αν ο χαρακτήρας c είναι λευκό-κενό (whitespace) δηλαδή ένας από:
  - `' '` - space
  - `'\n'` - new line
  - `'\f'` - form feed
  - `'\r'` - carriage return
  - `'\t'` - tab
  - `'\v'` - vertical tab

- `int iscntrl(int c)`
  επιστρέφει true αν ο χαρακτήρας c είναι control χαρακτήρας, δηλαδή έχει τιμή από 0 έως και 31 ή 127 στον πίνακα ASCII

Βιβλιοθήκη Χειρισμού Χαρακτήρων

Παράδειγμα

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

int main ()
{
    int ch;
    for (ch = 0; ch < 127 ; ch++)
        if (ispunct(ch))
            printf("%c ",ch);

    return 0;
}
```

το παραπάνω πρόγραμμα τυπώνει

```
! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~
```

Μετατροπές Συμβολοσειρών

Η γλώσσα C μας επιτρέπει μέσω της βιβλιοθήκης της να κάνουμε μετατροπές μεταξύ συμβολοσειρών και αριθμών:

- ακεραίων
- κινητής υποδιαστολής

Αυτές οι συναρτήσεις βρίσκονται στο αρχείο επικεφαλίδας `stdlib.h`.

Οι συναρτήσεις που ακολουθούν πέρνουν ως όρισμα μια συμβολοσειρά (προσοχή στην χρήση του `const`) και επιστρέφουν έναν αριθμό.

Μετατροπές Συμβολοσειρών

Συναρτήσεις

- `int atoi( const char* str )`
  μετατρέπει μια συμβολοσειρά σε ακέραιο
- `long atol( const char* str )`
  μετατρέπει μια συμβολοσειρά σε ακέραιο
- `double atof( const char* str )`
  μετατρέπει μια συμβολοσειρά σε αριθμό κινητής υποδιαστολής

Μετατροπές Συμβολοσειρών

Παράδειγμα

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

int main()
{
    char *num = "315.2";
    double n = atof(num);

    printf("half␣of␣%s␣is␣%lf\n", num, n/2.0);
}
```

Το παραπάνω πρόγραμμα τυπώνει

```
half of 315.2 is 157.600000
```

Μετατροπές Συμβολοσειρών

Περισσότερες Συναρτήσεις

- `double strtod( const char* nptr, char** endptr )`

  μετατρέπει μια συμβολοσειρά σε αριθμό κινητής υποδιαστολής
  - ο δείκτης `nptr` είναι η είσοδος
  - ο δείκτης `endptr` θα δείχνει στον δείκτη που θα δείχνει στον πρώτο χαρακτήρα μετά το τμήμα του string που μετατράπηκε σε double (έχουμε διπλό δείκτη γιατί είναι call-by-reference)

- `long strtol(const char* nptr, char** endptr, int base)`

  όπως παραπάνω, μόνο που η τιμή μετατρέπεται σε ακέραιο
  - `base` είναι η βάση μετατροπής. 0 ή 10 είναι το δεκαδικό.

Οι παραπάνω συναρτήσεις βολεύουν για να διαβάζουμε πολλούς αριθμούς συνεχόμενους.

Μετατροπές Συμβολοσειρών

Παράδειγμα

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

int main()
{
    char *n = "20 237 100";
    char *p = n;

    long l, k, m;

    l = strtol(p, &p, 0);
    k = strtol(p, &p, 8);
    m = strtol(p, &p, 16);

    printf("l = %ld\n", l);
    printf("k = %ld\n", k);
    printf("m = %ld\n", m);

    return 0;
}
```

το πρόγραμμα αυτό τυπώνει

```
l = 20
k = 159
m = 256
```

Συναρτήσεις Εισόδου/Εξόδου

- `char *gets( char *s )`

  Διαβάζει μια γραμμή από την τυπική είσοδο στον πίνακα `s` μέχρι να βρει ένα `newline` ή `EOF`, το οποίο αντικαθίστατε με `'\0'`.
  Επιστρέφει `s` εαν διάβασε με επιτυχία ή `NULL` άμα έγινε κάποιο λάθος ή τελείωσε η είσοδος ενώ δεν έχει διαβαστεί κανένας χαρακτήρας.

- `int getchar(void)`

  Διαβάζει ένα χαρακτήρα από την τυπική είσοδο και τον επιστρέφει ως `unsigned int` ο οποίος έχει μετατραπεί (cast) σε `int`. Επιστρέφει `EOF` σε περίπτωση λάθους ή άμα τερματιστεί η είσοδος.

Συναρτήσεις Εισόδου/Εξόδου

Παράδειγμα

```c
#include <stdio.h>

int main()
{
    int count = 0, c;

    while((c = getchar()) != EOF)
    {
        count++;
    }

    printf("You typed %d characters\n", count);
    return 0;
}
```

Συναρτήσεις Εισόδου/Εξόδου

- `int *puts(const char *s)`
  - Τυπώνει την συμβολοσειρά `s` και μια νέα γραμμή (newline) στην τυπική έξοδο.
  - Επιστρέφει ένα μη-αρνητικό αριθμό εαν ολοκληρώθηκε με επιτυχία, ή `EOF` εάν συνέβη λάθος.

- `int putchar(int c)`
  - Γράφει τον χαρακτήρα *c* μετατρέποντας τον σε `unsigned char` στην τυπική έξοδο.
  - Επιστρέφει τον χαρακτήρα που τυπώθηκε ως `unsigned char` αφού τον μετατρέψει σε `int`, ή `EOF` σε περίπτωση λάθους.

Συναρτήσεις Εισόδου/Εξόδου

Παράδειγμα

```c
#include <stdio.h>

int main()
{
    int c;

    while((c = getchar()) != EOF)
    {
        if (c >= 'a' && c <= 'z')
            putchar(c+'A'-'a');
    }

    return 0;
}
```

Συναρτήσεις Εισόδου/Εξόδου

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

  Δουλεύει όπως η συνάρτηση `printf()` αλλά στέλνει την έξοδο στον πίνακα χαρακτήρων `str` αντί για την τυπική έξοδο.

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

  Δουλεύει όπως η συνάρτηση `scanf()` αλλά διαβάζει από την συμβολοσειρά `str` αντί για την τυπική είσοδο.

Συναρτήσεις Εισόδου/Εξόδου

Παράδειγμα

Η συνάρτηση **sscanf**() βολεύει όταν θέλουμε να χωρίσουμε είσοδο σε πιο απλά κομμάτια.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char *input = "3.4 1.2 5.6";
6      float num[3];
7
8      sscanf(input, "%f %f %f", &num[0], &num[1], &num[2]);
9      printf("%f, %f, and %f\n", num[0], num[1], num[2]);
10
11     return 0;
12 }
```

Συναρτήσεις Χειρισμού

- `char* strcpy( char *s1, const char *s2)`

  αντιγράφει την συμβολοσειρά `s2` στον πίνακα `s1`. επιστρέφεται η τιμή του `s1`.

- `char* strncpy( char *s1, const char *s2, size_t n )`

  αντιγράφει τους `n` χαρακτήρες της συμβολοσειράς `s2` στον πίνακα `s1`. επιστρέφεται η τιμή του `s1`.

Συναρτήσεις Χειρισμού

Παράδειγμα

Απλό παράδειγμα αντιγραφής συμβολοσειράς.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main()
5  {
6      char *source = "Hello world!";
7      char target[13];
8
9      strcpy(target, source);
10     printf("%s\n", target);
11
12     return 0;
13 }
```

Συναρτήσεις Χειρισμού

- `char* strcat( char *s1, const char *s2 )`
  - Προσαρτά την συμβολοσειρά `s2` στον πίνακα `s1`. Ο πρώτος χαρακτήρας της `s2` υπερεγγράφει τον τερματικό κενό χαρακτήρα του `s1`. Επιστρέφεται η τιμή του `s1`.
  - Οι συμβολοσειρές δεν πρέπει να έχουν κοινό κομμάτι και η συμβολοσειρά `s1` πρέπει να έχει αρκετό χώρο.

- `char* strncat( char *s1, const char *s2, size_t n )`

  Προσαρτά τους *n* χαρακτήρες της συμβολοσειράς `s2` στον πίνακα `s1`. Ο πρώτος χαρακτήρας της `s2` υπερεγγράφει τον τερματικό κενό χαρακτήρα του `s1`. Επιστρέφεται η τιμή του `s1`.

Συναρτήσεις Χειρισμού

Παράδειγμα

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

int main()
{
    char s[100] = { '\0' };

    strcat(s, "Hello ");
    strcat(s, "world!");

    printf("%s\n", s);

    return 0;
}
```

Σύγκριση Συμβολοσειρών

Συναρτήσεις

Το αρχείο `string.h` περιέχει τις εξής συναρτήσεις για σύγκριση συμβολοσειρών.

- `int strcmp( const char *s1, const char *s2)`
  - συγκρίνει δύο strings χαρακτήρα χαρακτήρα
  - επιστρέφει εναν ακέραιο μικρότερο, ίσο ή μεγαλύτερο του μηδέν εαν το string s1 είναι αντίστοιχα μικρότερο, ίσο ή μεγαλύτερο λεξικογραφικά από το s2

- `int strncmp( const char *s1, const char *s2, size_t n )`
  - όπως παραπάνω αλλά συγκρίνονται το πολύ *n* χαρακτήρες
  - δεν συγκρίνονται χαρακτήρες μετά το `'\0'`

Σύγκριση Συμβολοσειρών

Παράδειγμα

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

int main()
{
    char *x = "Hello world";
    char *y = "Hello John";
    char *z = "Hello world";

    printf("strcmp(x,y) = %d\n", strcmp(x,y));
    printf("strcmp(y,x) = %d\n", strcmp(y,x));
    printf("strcmp(x,z) = %d\n", strcmp(x,z));
    printf("strncmp(y,z,6) = %d\n", strncmp(y,z,6));

    return 0;
}
```

το παραπάνω πρόγραμμα τυπώνει

```
strcmp(x,y) = 1
strcmp(y,x) = -1
strcmp(x,z) = 0
strncmp(y,z,6) = 0
```

Συναρτήσεις Αναζήτησης

- `char *strchr( const char *s, int c)`
  - Επιστρέφει ένα δείκτη στην πρώτη εμφάνιση του χαρακτήρα `c` στην συμβολοσειρά `s`. Επιστρέφει NULL άμα δεν βρεθεί ο χαρακτήρας.
- `char *strrchr( const char *s, int c)`
  - Επιστρέφει ένα δείκτη στην τελευταία εμφάνιση του χαρακτήρα `c` στην συμβολοσειρά `s`. Επιστρέφει NULL άμα δεν βρεθεί ο χαρακτήρας.
- `char *strstr( const char *haystack, const char *n )`
  - Αναζητά την πρώτη εμφάνιση της υπο-συμβολοσειράς `n` στην συμβολοσειρά `haystack`. Οι τερματικοί `'\0'` χαρακτήρες δεν συγκρίνονται.
  - Επιστρέφει ένα δείκτη στην αρχή της εμφάνισης της υπο-συμβολοσειράς ή NULL άμα δεν βρεθεί η υπο-συμβολοσειρά.

Συναρτήσεις Αναζήτησης

```
char *strtok( char *s1, const char *s2 )
```

- Μια ακολουθία κλήσεων της `strtok()` χωρίζει το s1 σε λεκτικά σύμβολα (tokens) που το κάθε ένα οριοθετείται από χαρακτήρες που έχουν οριστεί στο αλφαριθμητικό s2.
- Η πρώτη κλήση περιέχει το s1 ως πρώτο όρισμα, και οι επόμενες κλήσεις, που υποδηλώνονται με NULL ως πρώτο όρισμα, επιστρέφουν τα επόμενα λεκτικά σύμβολα αρχίζοντας την αναζήτηση μετά το τέλος του προηγούμενου.
- Επιστρέφεται ένας δείκτης στο τρέχον λεκτικό σύμβολο, και NULL αν δεν υπάρχουν άλλα.
- Το αλφαριθμητικό s2 μπορεί να είναι διαφορετικό σε κάθε κλήση.

Συναρτήσεις Αναζήτησης

Παράδειγμα `strtok()`

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

int main() {
    char email[80], token[] = "@.", *c;

    printf("type email address: ");
    scanf("%s", email);

    c = strtok(email, token);
    if (c != NULL) {
        printf("%s", c);
        while ((c = strtok(NULL, token)) != NULL)
            printf(", %s", c);
    }

    return 0;
}
```

παράδειγμα εκτέλεσης

```
type email address: michail@hua.gr
michail, hua, gr
```

Συναρτήσεις Μνήμης

Η βιβλιοθήκη της C μας παρέχει και συναρτήσεις που διαχειρίζονται μνήμη χωρίς να ενδιαφέρονται για τον τύπο των δεδομένων που είναι αποθηκευμένα εκεί.

- `void *memcpy( void *dest, const void *src, size_t n )`
  - Αντιγράφει `n` bytes από το σημείο της μνήμης που δείχνει ο δείκτης `src` στο σημείο της μνήμης που δείχνει ο `dest`.
  - Επιστρέφει έναν δείκτη στο `dest`.
  - Τα κομμάτια της μνήμης δεν πρέπει να επικαλύπτονται.

- `void *memmove( void *dest, const void *src, size_t n )`
  Όπως η `memcpy` αλλά τα δύο κομμάτια μνήμης μπορούν να επικαλύπτονται. Τα περιεχόμενα αντιγράφονται πρώτα σε ένα βοηθητικό πίνακα.

Συναρτήσεις Μνήμης

- `void *memset( void *s, int c, size_t n )`
  - Γεμίζει τα πρώτα `n` bytes του σημείου της μνήμης που δείχνει ο `s` με την πληροφορία της σταθεράς byte `c`.
  - Επιστρέφει έναν δείκτη στο `s`.

- `int memcmp( const void *s1, const void *s2, size_t n )`
  - Συγκρίνει τα πρώτα `n` bytes των κομματιών της μνήμης που δείχνουν οι δείκτες `s1` και `s2`.
  - Επιστρέφει έναν ακέραιο μικρότερο, ίσο ή μεγαλύτερο από το μηδέν εάν τα πρώτα `n` bytes του `s1` είναι, αντίστοιχα, μικρότερα, ίσα ή μεγαλύτερα από τα πρώτα `n` bytes του `s2`.

- `void *memchr( const void *s, int c, size_t n )`
  - Ψάχνει τα πρώτα `n` bytes της μνήμης που δείχνει ο δείκτης `s` για τον χαρακτήρα `c`. Η αναζήτηση σταματάει στο πρώτο byte που ταιριάζει.
  - Επιστρέφει έναν δείκτη στο byte που ταιριάζει ή NULL άμα ο χαρακτήρας δεν υπάρχει στο κομμάτι της μνήμης αυτό.

Μήκος Ακολουθίας Χαρακτήρων

- `size_t strlen( const char *s )`

  Υπολογίζει το μήκος της συμβολοσειράς `s` χωρίς να συμπεριλάβει τον χαρακτήρα τερματισμού `'\0'`.

Το παρακάτω πρόγραμμα

```
1  int main()
2  {
3      char s[] = "Hello world!";
4
5      printf("length = %d\n", strlen(s));
6
7      return 0;
8  }
```

τυπώνει

```
length = 12
```

Γραμμή Εντολών

Κάθε πρόγραμμα σε C έχει την συνάρτηση `main()`.

Ο γενικός ορισμός είναι:

```
1  int main(int argc, char* argv[])
2  {
3      /* code */
4  }
```

Κατά την κλήση του προγράμματος μας οι παράμετροι στην γραμμή εντολών δίνονται στο πρόγραμμα μέσω αυτών των δυο παραμέτρων.

Κλήση με Παραμέτρους

Παράδειγμα σε Σύστημα GNU/Linux

Έστω το αρχείο `example.c` με το παρακάτω πρόγραμμα:

```
1  #include <stdio.h>
2
3  int main(int argc, char* argv[])
4  {
5      printf("Hello world\n");
6
7      return 0;
8  }
```

Κάνουμε compile το πρόγραμμα με τον compiler GCC με

```
gcc example.c -o example
```

και εκτελούμε το πρόγραμμα με

```
./example
```

Κλήση με Παραμέτρους

Παράδειγμα σε Σύστημα GNU/Linux

Έστω το αρχείο `example.c` με το παρακάτω πρόγραμμα:

```
1  #include <stdio.h>
2
3  int main(int argc, char* argv[])
4  {
5      printf("Hello world\n");
6
7      return 0;
8  }
```

Εκτελώντας με

```
./example arg1 arg2
```

οι συμβολοσειρές `arg1` και `arg2` δίνονται στην συνάρτηση `main` μέσω των παραμέτρων `argc` και `argv`.

Κλήση με Παραμέτρους

```
1  int main(int argc, char* argv[])
2  {
3      /* code */
4  }
```

Η παράμετρος `argc` υποδηλώνει τον αριθμό των παραμέτρων που δώθηκαν την στιγμή εκτέλεσης του προγράμματος. Το όνομα του προγράμματος θεωρείται ως παράμετρος και άρα αυτή η μεταβλητή έχει πάντα 1 παραπάνω από τον αριθμό των παραμέτρων που έδωσε ο χρήστης.

Η παράμετρος `argv` είναι ένας πίνακας με συμβολοσειρές. Κάθε συμβολοσειρά είναι και μία παράμετρος.

- Οι παράμετροι περιέχονται στον πίνακα με την σειρά που δώθηκαν στην γραμμή εντολών.
- Στην θέση 0 του πίνακα υπάρχει πάντα το όνομα του προγράμματος

Εκτύπωση Παραμέτρων

Έστω το παρακάτω πρόγραμμα **printargs.c**.

```c
#include <stdio.h>

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

    for( i = 0; i < argc; i++ )
        printf("parameter %d is %s\n", i, argv[i]);

    return 0;
}
```

Εκτελώντας

```
./printargs arg1 arg2
```

πέρνουμε την έξοδο

```
parameter 0 is ./printargs
parameter 1 is arg1
parameter 2 is arg2
```

Εκτύπωση Παραμέτρων

Έστω το παρακάτω πρόγραμμα **printargs.c**.

```c
#include <stdio.h>

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

    for( i = 0; i < argc; i++ )
        printf("parameter %d is %s\n", i, argv[i]);

    return 0;
}
```

Εκτελώντας

```
./printargs -i input.txt -o output.txt
```

πέρνουμε την έξοδο

```
parameter 0 is ./test
parameter 1 is -i
parameter 2 is input.txt
parameter 3 is -o
parameter 4 is output.txt
```