Διάλεξη 9η: Αλφαριθμητικά (strings)

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Βασίζεται σε διαφάνειες του Κ. Παναγιωτάκη

Χαρακτήρες

- Αναπαριστώνται από έναν δυαδικό αριθμό 8 δυαδικών ψηφίων (8 bits) (256 διαφορετικές τιμές)
- Η κωδικοποίησή τους έχει τυποποιηθεί με τον κώδικα ASCII
- Εκτός από γράμματα και ψηφία υπάρχουν και πολλοί ειδικοί χαρακτήρες
- Σταθερές χαρακτήρων
  - Η έκφραση `'z'` είναι μια σταθερά και αντιστοιχεί σε έναν `int` με τα 8 τελευταία του bits ίσα με τον ASCII κωδικό του χαρακτήρα
  - `'z'`, `'\t'`, `'\n'`, κτλ

Αλφαριθμητικά

- Μια σειρά από χαρακτήρες που αντιμετωπίζονται σαν ένα αντικείμενο
  - Γράμματα, αριθμοί, ειδικοί χαρακτήρες (*, /, $) και όλοι οι εκτυπώσιμοι χαρακτήρες
  - Τιμές εισάγονται μέσα σε διπλά εισαγωγικά: ”Hello”
- Τα Strings είναι πάντα πίνακες από χαρακτήρες
  - Ένα String είναι δείκτης στον 1ο χαρακτήρα του πίνακα
  - Τιμή του string είναι η διεύθυνση του 1ου χαρακτήρα του πίνακα

Δηλώσεις αλφαριθμητικών

- Δηλώσεις αλφαριθμητικών
  - Σαν πίνακας από χαρακτήρες ή σαν δείκτης σε χαρακτήρα char *
    - `char color[] = ”blue”;`
    - `char *colorPtr = ”blue”;`
  - Κάθε string τελειώνει με `’\0’` και πρέπει να το λαμβάνουμε υπόψη στη δήλωση του πίνακα
    - Ο πίνακας color έχει 5 στοιχεία

Ανάγνωση

- Διάβασμα strings

Χρήση

```
scanf("%s", word);
scanf("%s", &(word[0]));
```

- Αντιγράφει στο `word[]`
- Δεν χρειάζεται & (επειδή είναι και δείκτης)
- Αφήνουμε χώρο στον πίνακα και για το `'\0'`

Παράδειγμα 1

strings.c

```
#include <stdio.h>

int main(void)
{
  char string1[20], string2[] = ”string literal”;
  int i;

  printf(”Please enter a string: ”);
  scanf(”%s”, string1);
  printf(”string1 is: %s\n”, string1);
  printf(”string2 is: %s\n”, string2);
  printf(”string1 with spaces between every character is:\n”);

  for(i = 0; string1[i] != ’\0’; i++) {
    printf(”%c ”, string1[i]);
  }
  printf(”\n”);
  return 0;
}
```

Παράδειγμα 2

reverse.c

```
#include <stdio.h>
#include <string.h>
#define N 100

void inversion(char *s) {
  int i;

  for(i = strlen(s) - 1; i >= 0; --i) {
    printf("%c", s[i]);
  }
  printf("\n");
}

int main(void) {
  char s[N];

  printf("Please enter a word up to %d characters long: ", N);
  scanf("%s", s);
  inversion(s);
  return 0;
}
```

Μετατροπές αλφαριθμητικών

- Συναρτήσεις βιβλιοθήκης για μετατροπή από αλφαριθμητικά

#include

```
int atoi(const char *nptr);
long atol(const char *nptr);
long long atoll(const char *nptr);
```

- Βρίσκονται στο `<stdlib.h>`
- Μετατρέπουν αλφαριθμητικά (αν είναι κατάλληλα) σε αριθμητικές τιμές

String formatting

- Συναρτήσεις βιβλιοθήκης που μορφοποιούν ή διαβάζουν strings

#include

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

- Χρήση:

string_io.c

```
int main(void) {
  char s[100], f[] = "1.45 2.2 0.12";
  float t1, t2, t3;

  sprintf(s,"%s", f);
  sscanf(s, "%f %f %f", &t1, &t2, &t3);
  printf("s = %s\nt1 = %f t2 = %f t3 = %f\n", s, t1, t2, t3);
}
```

String formatting

- Συναρτήσεις βιβλιοθήκης που μορφοποιούν ή διαβάζουν strings

Ισοδύναμη με την scanf μόνο που η είσοδος είναι από το string s και όχι από το πληκτρολόγιο

#include

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

- Χρήση:

string_io.c

```
int main(void) {
  char s[100], f[] = "1.45 2.2 0.12";
  float t1, t2, t3;

  sprintf(s,"%s", f);
  sscanf(s, "%f %f %f", &t1, &t2, &t3);
  printf("s = %s\nt1 = %f t2 = %f t3 = %f\n", s, t1, t2, t3);
}
```

String formatting

- Συναρτήσεις βιβλιοθήκης που μορφοποιούν ή διαβάζουν strings

Ισοδύναμη με την printf μόνο
που η έξοδος είναι στο string
s και όχι στην οθόνη

#include

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

- Χρήση:

string_io.c

```
int main(void) {
  char s[100], f[] = "1.45 2.2 0.12";
  float t1, t2, t3;

  sprintf(s,"%s", f);
  sscanf(s, "%f %f %f", &t1, &t2, &t3);
  printf("s = %s\nt1 = %f t2 = %f t3 = %f\n", s, t1, t2, t3);
}
```

String formatting

- Συναρτήσεις βιβλιοθήκης που μορφοποιούν ή διαβάζουν strings

#include

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

Ισοδύναμη με την sprintf μόνο που γράφει το πολύ `size` bytes

- Χρήση:

string_io.c

```
int main(void) {
  char s[100], f[] = "1.45 2.2 0.12";
  float t1, t2, t3;

  sprintf(s,"%s", f);
  sscanf(s, "%f %f %f", &t1, &t2, &t3);
  printf("s = %s\nt1 = %f t2 = %f t3 = %f\n", s, t1, t2, t3);
}
```

Παράδειγμα

- **Πρόβλημα:** Η γραμματεία αποθηκεύει τη βαθμολογία του κάθε φοιτητή σε ένα αρχείο, χρησιμοποιώντας το login του φοιτητή και τη χρονιά εγγραφής ως όνομα αρχείου, π.χ. `mpap_2008.txt`. Να γράψετε πρόγραμμα που θα διαβάζει το login και τη χρονιά εγγραφής, και θα υπολογίζει το όνομα του αρχείου.

```
#include <stdio.h>
#define N 100

int main(void) {
  char name[N], file_name[N];
  int year;

  scanf("%s %d", name, &year);
  sprintf(file_name, "%s_%d.txt", name, year);
  printf("%s\n", file_name);
  return 0;
}
```

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

#include

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

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

Επιστρέφει το μήκος ενός string, χωρίς το τελικό `'\0'`

#include

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

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

Δεσμεύει όση μνήμη χρειάζεται και αντιγράφει ένα string. Η μνήμη πρέπει να απελευθερωθεί στο τέλος με `free`.

#include

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

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

```
#include

size_t   strlen(const char *);
char    *strdup(const char *);

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

Ενώνει δύο strings. Σβήνει το byte `'\0'` από το τέλος του πρώτου και αντιγράφει εκεί το δεύτερο. Πρέπει να υπάρχει αρκετή μνήμη για να το χωρέσει.

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

#include

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

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

Όπως και η `strcat`, αλλά θα αντιγράψει το πολύ `N` bytes από το δεύτερο string.

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

#include

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

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

Συγκρίνει δύο strings και επιστρέφει αρνητικό, μηδέν ή θετικό αποτέλεσμα αν το πρώτο string είναι μικρότερο, ίσο ή μεγαλύτερο από το δεύτερο, αντίστοιχα.

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

#include

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

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

Όπως και η `strcmp`, αλλά συγκρίνει το πολύ `N` bytes

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

#include

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

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

Αντιγράφει το δεύτερο string στο πρώτο, μαζί με το `'\0'`

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Συναρτήσεις βιβλιοθήκης χειρισμού strings

- Η βιβλιοθήκη `string.h` περιέχει διάφορες βοηθητικές συναρτήσεις

#include

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

char    *strcat(char *restrict, const char *restrict);
char    *strncat(char *restrict, const char *restrict, size_t);

int      strcmp(const char *, const char *);
int      strncmp(const char *, const char *, size_t);

char    *strcpy(char *restrict, const char *restrict);
char    *strncpy(char *restrict, const char *restrict, size_t);
```

Όπως και η `strcpy`, αλλά αντιγράφει το πολύ `N` bytes

- Προτιμάτε τις `strncat`, `strncmp`, `strncpy`.
  - Συχνά λάθη: όχι αρκετός χώρος για το string, λείπει το `'\0'` στο τέλος, κλπ.

Παράδειγμα

freq.c

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

/* Συνάρτηση get_max
 * Δέχεται ένα πίνακα με ακέραιους
 * και το μήκος N του πίνακα
 * Επιστρέφει τη θέση του πίνακα με
 * τη μεγαλύτερη τιμή
 */
int get_max(int h[], int N) {
  int i, max_position = 0;

  for(i = 1; i < N; i++) {
    if(h[i] > h[max_position]) {
      max_position = i;
    }
  }
  return max_position;
}
```

```c
/* Συνάρτηση char_freq
 * Δέχεται ένα string
 * Επιστρέφει τον πιο συχνό χαρακτήρα
 */
char char_freq(const char *str) {
  int i, w = 0;
  int histogram[256];
  char result;

  for(i = 0; i < 256; i++) {
    histogram[i] = 0;
  }

  for(i = 0; i < strlen(str); i++) {
    histogram[str[i]]++;
  }

  result = (char) get_max(histogram, 256);
  return result;
}
```