Δ.Π.Θ Δομημένος Προγραμματισμός

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

Αλφαριθμητικά

Dr. Αθανάσιος Μπαλαφούτης
Εργαστηριακό Διδακτικό Προσωπικό
Τομέας Συστημάτων Παραγωγής
Εργαστήριο Ρομποτικής και Αυτοματισμών
abalafou@pme.duth.gr
Γραφείο 304, τηλ.: 25410 – 79892

Αλφαριθμητικά

Αλφαριθμητικό (String): Μια ακολουθία από χαρακτήρες (char) μέσα σε διπλά εισαγωγικά.

π.χ: “Γιάννης”

“Καλησπέρα”

“Δομημένος Προγραμματισμός”

Αλφαριθμητικά

Για την εκτύπωση αλφαριθμητικού χρησιμοποιούμε τον προσδιοριστή **%s**

```
int main(){

   printf("%s", "Δομημένος Προγραμματισμός");

   return 0;
}
```

Πίνακες Χαρακτήρων

Τα Αλφαριθμητικά αποθηκεύονται με τη βοήθεια μονοδιάστατου (1D) πίνακα

Μέγεθος πίνακα ≥ πλήθος χαρακτήρων + 1

π.χ: “Άννα” → μέγεθος πίνακα ≥ 5

Πίνακες Χαρακτήρων

Τα Αλφαριθμητικά αποθηκεύονται με τη βοήθεια μονοδιάστατου (1D) πίνακα

Μέγεθος πίνακα ≥ πλήθος χαρακτήρων + 1

π.χ: “Άννα” → μέγεθος πίνακα ≥ 5

```
char C[10];
C[0] = 'A';
C[1] = 'N';
C[2] = 'N';
C[3] = 'A';
```

A	N	N	A						

Πίνακες Χαρακτήρων

Τα Αλφαριθμητικά αποθηκεύονται με τη βοήθεια μονοδιάστατου (1D) πίνακα

Μέγεθος πίνακα ≥ πλήθος χαρακτήρων + 1

π.χ: “Άννα” → μέγεθος πίνακα ≥ 5

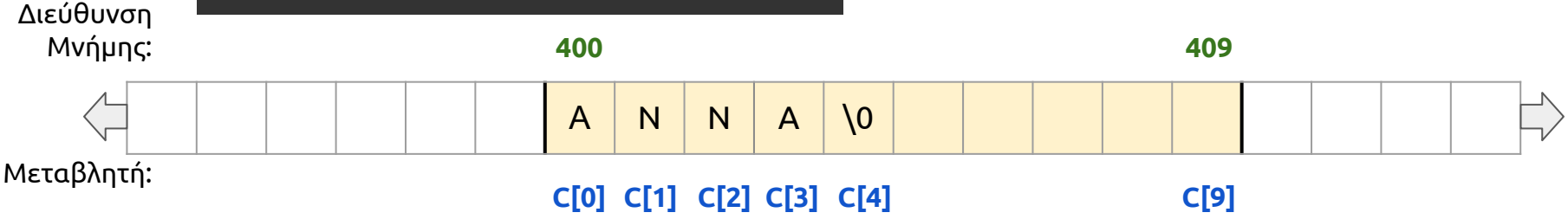
```
char C[10];
C[0] = 'A';
C[1] = 'N';
C[2] = 'N';
C[3] = 'A';
C[4] = '\0';
```

A	N	N	A	\0					

Χρειαζόμαστε έναν **ειδικό χαρακτήρα** που να δηλώνει το τέλος του αλφαριθμητικού μέσα στον πίνακα.

Αλφαριθμητικά (Strings)

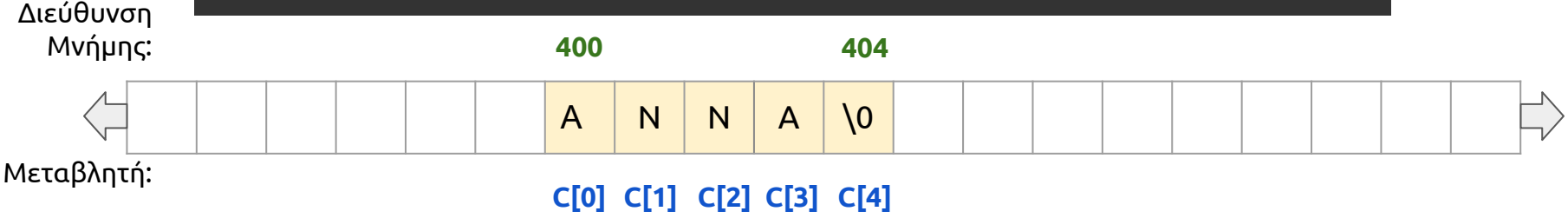

```c
#include <stdio.h>
int main(){
    char C[10];
    C[0] = 'A';
    C[1] = 'N';
    C[2] = 'N';
    C[3] = 'A';
    C[4] = '\0';
    printf("%s", C);
}
```



Αλφαριθμητικά (Strings)


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

int main(){
    char C[] = "ANNA";
    printf("Μέγεθος πίνακα σε bytes: %d \n", sizeof(C)); // 5
    int len = strlen(C);
    printf("Μήκος πίνακα: %d \n", len);  // 4
    printf("%s", C);
}
```

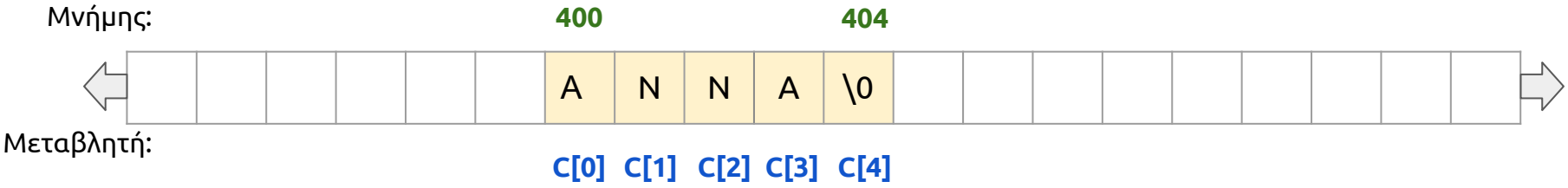


Αλφαριθμητικά (Strings)


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

int main(){
    char C[] = "ANNA";
    printf("Μέγεθος πίνακα σε bytes: %d \n", sizeof(C)); // 5
    int len = strlen(C);
    printf("Μήκος πίνακα: %d \n", len);  // 4
    printf("%s", C);
}
```

```c
char C[5] = {'A', 'N', 'N', 'A', '\0'};
```



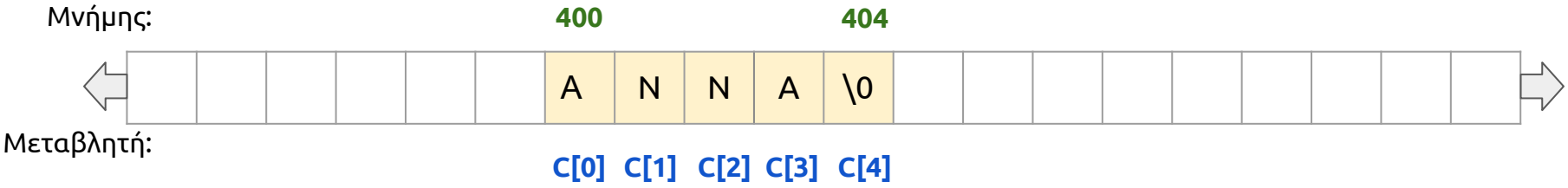
Αλφαριθμητικά (Strings)


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

int main(){
    char C[] = "ANNA";
    printf("Μέγεθος πίνακα σε bytes: %d \n", sizeof(C)); // 5
    int len = strlen(C);
    printf("Μήκος πίνακα: %d \n", len);  // 4
    printf("%s", C);
}
```

```c
char *C = "ANNA";
```

Σταθερά
Το αλφαριθμητικό **δεν** μπορεί να τροποποιηθεί



Αλφαριθμητικά (Strings)


```c
#include <stdio.h>

int main(){
   char C[] = "ANNA";
   C[1] = 'B';

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

```c
#include <stdio.h>

int main(){
   char *C = "ANNA";
   *(C+1) = 'B';

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

Segmentation fault

Αλφαριθμητικά και Συναρτήσεις


```c
#include <stdio.h>

void print(char *c){

    int i = 0;
    while(c[i] != '\0'){
        printf("%c", c[i]);
        i++;
    }
    printf("\n");
}

int main(){
    char C[20] = "ANNA";
    print(C);
}
```

Αλφαριθμητικά και Συναρτήσεις


```c
#include <stdio.h>

void print(char *c){

    int i = 0;
    while(c[i] != '\0'){
        printf("%c", c[i]);
        i++;
    }
    printf("\n");
}

int main(){
    char C[20] = "ANNA";
    print(C);
}
```

Ισοδύναμα προγράμματα

```c
#include <stdio.h>

void print(char *c){

    while(*c != '\0'){
        printf("%c", *c);
        c++;
    }
    printf("\n");
}

int main(){
    char C[20] = "ANNA";
    print(C);
}
```

Εκτύπωση Αλφαριθμητικών

Ο προσδιοριστής **“%.ns”**, χρησιμοποιείται για να τυπώσουμε ένα τμήμα του αλφαριθμητικού, όπου **n**, το πλήθος των χαρακτήρων που θα εμφανιστούν

```
char *ptr="Hello World";

printf("%.5s", ptr);
```

Έξοδος

```
Hello
```

Εκτύπωση Αλφαριθμητικών

Συνάρτηση **puts()**: Εκτυπώνει αλφαριθμητικά και προσθέτει αυτόματα την αλλαγή γραμμής (“\n”)

```
char *ptr="Hello World";

puts(ptr);
puts(ptr);
```

Έξοδος

```
Hello World
Hello World
```

Ανάγνωση Αλφαριθμητικών

Συνάρτηση **scanf()**: Διαβάζει αλφαριθμητικά ως πίνακες χαρακτήρων

```
char a[10];

scanf("%s", a);
puts(a);
```

Στους πίνακες δεν βάζουμε **&**

Ανάγνωση Αλφαριθμητικών

Συνάρτηση **scanf()**: Διαβάζει αλφαριθμητικά ως πίνακες χαρακτήρων

```
char a[10];

scanf("%s", a);
puts(a);
```

Στους πίνακες δεν βάζουμε **&**

Η συνάρτηση **scanf** θα σταματήσει την ανάγνωση αλφαριθμητικών στον 1ο κενό χαρακτήρα που θα συναντήσει.

Ανάγνωση Αλφαριθμητικών

Συνάρτηση **gets()**: Διαβάζει αλφαριθμητικά, ακόμη και αν αυτά περιέχουν κενούς χαρακτήρες

```
char a[10];

gets(a);
puts(a);
```

Ανάγνωση Αλφαριθμητικών

Συνάρτηση **gets()**: Διαβάζει αλφαριθμητικά, ακόμη και αν αυτά περιέχουν κενούς χαρακτήρες

```
char a[10];

gets(a);
puts(a);
```

Οι συναρτήσεις **scanf** και **gets** δεν ελέγχουν αν έχουμε ξεπεράσει το μέγεθος του πίνακα

Συνάρτηση Ανάγνωσης Αλφαριθμητικών

Θα φτιάξουμε μια δική μας συνάρτηση για τη σωστή ανάγνωση αλφαριθμητικών.
Θέλουμε να έχει τα παρακάτω χαρακτηριστικά:

1. Να μπορεί να διαβάσει αλφαριθμητικά που περιέχουν κενούς χαρακτήρες
2. Να σταματά την ανάγνωση μόλις συναντήσει αλλαγή γραμμής (\n)
3. Να προσθέτει στο τέλος του αλφαριθμητικού τον ειδικό χαρακτήρα (\0)
4. Να ελέγχει αν έχουμε ξεπεράσει το μέγεθος του πίνακα
5. Να επιστρέφει το πλήθος των χαρακτήρων του αλφαριθμητικού

Συνάρτηση Ανάγνωσης Αλφαριθμητικών

Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char
...	...	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

Η συνάρτηση **getchar()**, διαβάζει από το πληκτρολόγιο έναν χαρακτήρα, ως ακέραια μεταβλητή τύπου int.

Αυτή η ακέραια τιμή αντιστοιχεί στον **πίνακα ASCII**

Συνάρτηση Ανάγνωσης Αλφαριθμητικών


```c
#include <stdio.h>

int input(char str[], int n){
    int ch;
    int i = 0;
    while((ch = getchar()) != '\n'){
        if (i < n){
            str[i] = ch;
            i++;
        }
    }
    str[i] = '\0';
    return i;
}
```

```c
int main() {
    char str[10];

    int n = input(str, 10);
    printf("%d %s", n, str);

    return 0;
}
```

Ειδικές Συναρτήσεις - strcpy

Συνάρτηση **strcpy()**: Αντιγράφει ένα αλφαριθμητικό σε ένα άλλο (string copy)

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

int main() {
   char str1[20] = "Hello World";
   char str2[20];

   strcpy(str2, str1);
   printf("%s", str2);

   return 0;
}
```

Ειδικές Συναρτήσεις - strlen

Συνάρτηση **strlen()**: Επιστρέφει το μήκος του αλφαριθμητικού

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

int main() {
   char str1[20] = "Hello World";

   printf("%zu", strlen(str1));

   return 0;
}
```

Σε μερικούς compiles το **%d** δημιουργεί σφάλμα

Ειδικές Συναρτήσεις - strlen

Συνάρτηση **strlen()**: Επιστρέφει το μήκος του αλφαριθμητικού

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

int main() {
   char str1[20] = "Hello World";

   printf("%zu", strlen(str1));

   return 0;
}
```

Η **strlen()** υπολογίζει το μήκος της συμβολοσειράς σε bytes, όχι σε χαρακτήρες Unicode.

Οι ελληνικοί χαρακτήρες σε UTF-8 κωδικοποιούνται με 2 ή περισσότερα bytes, ενώ οι αγγλικοί με 1 byte.

Αν π.χ. δώσεις "**ΑΒΓ**" (3 χαρακτήρες), η **strlen()** μπορεί να επιστρέψει **6** αντί για **3**.

Ειδικές Συναρτήσεις - strcat

Συνάρτηση **strcat()**: Συνένωση αλφαριθμητικών

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

int main() {
   char str1[50], str2[20];

   strcpy(str1, "Welcome to ");
   strcpy(str2, "C programming");
   strcat(str1, str2);
   printf("%s", str1);

   return 0;
}
```

Σε περίπτωση που το μέγεθος του πίνακα δε χωρά το νέο αλφαριθμητικό, η συμπεριφορά του προγράμματος είναι **απρόβλεπτη**

Ειδικές Συναρτήσεις - strcmp

Συνάρτηση **strcmp()**: Σύγκριση αλφαριθμητικών

```c
#include <stdio.h>
#include <string.h>
int main() {
   char *s1 = "abcd";
   char *s2 = "abce";
   if (strcmp(s1, s2) < 0)
      printf("s1 < s2");
   else if(strcmp(s1, s2) > 0)
      printf("s1 > s2");
   else
      printf("s1 == s2");
   return 0;
}
```

Επιστρέφει τιμές:

- <0, αν s1 < s2
- >0, αν s1 > s2
- 0, αν s1 == s2

Ειδικές Συναρτήσεις - isspace

Συνάρτηση **isspace()**: ελέγχει αν ένας χαρακτήρας είναι κενό (whitespace)

```c
#include <stdio.h>
#include <ctype.h>
int main() {
   char str[] = "Hello, how are you?\nI am fine.\tThanks!";

   int count = 0;
   for (int i = 0; str[i] != '\0'; i++) {
       if (isspace(str[i])) {
           count++;
       }
   }
   printf("Number of whitespace characters: %d\n", count);
   return 0;
}
```

Θεωρούνται κενά οι εξής χαρακτήρες:

- ' ' (κενό διάστημα)
- '\t' (tab)
- '\n' (νέα γραμμή)

Ειδικές Συναρτήσεις - isdigit

Συνάρτηση **isdigit()**: ελέγχει αν ένας χαρακτήρας είναι ψηφίο (0-9)

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

int main() {
    char c;

    printf("Δώσε έναν χαρακτήρα: ");
    scanf("%c", &c);
    if (isdigit(c))
        printf("Ο χαρακτήρας '%c' είναι ψηφίο.\n", c);
    else
        printf("Ο χαρακτήρας '%c' δεν είναι ψηφίο.\n", c);
    return 0;
}
```

Επιστρέφει μη μηδενική τιμή (συνήθως 1) αν ο χαρακτήρας είναι ψηφίο, διαφορετικά επιστρέφει 0

Μετατροπή Χαρακτήρα σε Ακέραιο

Αν έχω έναν αριθμητικό χαρακτήρα, π.χ. '5', πως μπορώ να τον μετατρέψω σε ακέραιο;

```c
#include <stdio.h>

int main() {
    char c;
    int num;

    printf("Δώσε έναν χαρακτήρα: ");
    scanf("%c", &c);
    num = c - '0';
    printf("Ψηφίο: %d\n", num);
    return 0;
}
```

Ψηφίο	ASCII Τιμή
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Άσκηση 6.1

Να γραφεί μια συνάρτηση με όνομα `word_cnt` που να δέχεται ως είσοδο ένα αλφαριθμητικό μέγιστου μήκους `80` χαρακτήρων και να επιστρέφει το πλήθος των λέξεων που περιέχει. Οι λέξεις στο αλφαριθμητικό διαχωρίζονται μεταξύ τους με ένα κενό διάστημα.

Στη συνέχεια να γραφεί ένα πρόγραμμα που να χρησιμοποιεί τη συνάρτηση `word_cnt`. Η εισαγωγή των δεδομένων στη `main()` να γίνει με τη χρήση της `gets()`.
ΥΠΟΔΕΙΞΗ : μπορείτε να χρησιμοποιήσετε τη συνάρτηση `isspace` για να ελέγχετε αν ένας χαρακτήρας είναι το κενό διάστημα.

Άσκηση 6.1

i=0
count=0
in_word=0

ΑΝ (συναντήσεις κενό) ΤΟΤΕ
in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
count++
in_word=1

Άσκηση 6.1

i=0
count=1
in_word=1

ΑΝ (συναντήσεις κενό) ΤΟΤΕ
  in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1

ψευδής

αληθης

Άσκηση 6.1

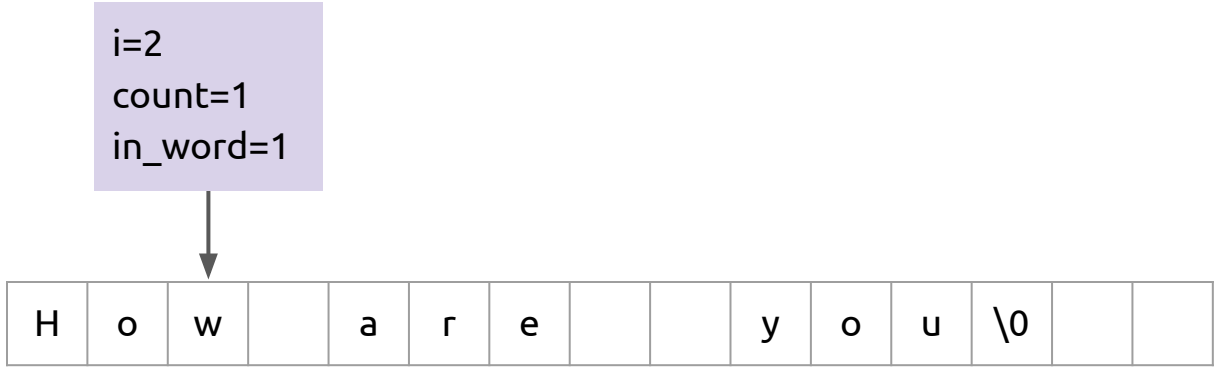
i=1
count=1
in_word=1

ΑΝ (συναντήσεις κενό) ΤΟΤΕ
in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
count++
in_word=1

ψευδής

ψευδής

Άσκηση 6.1

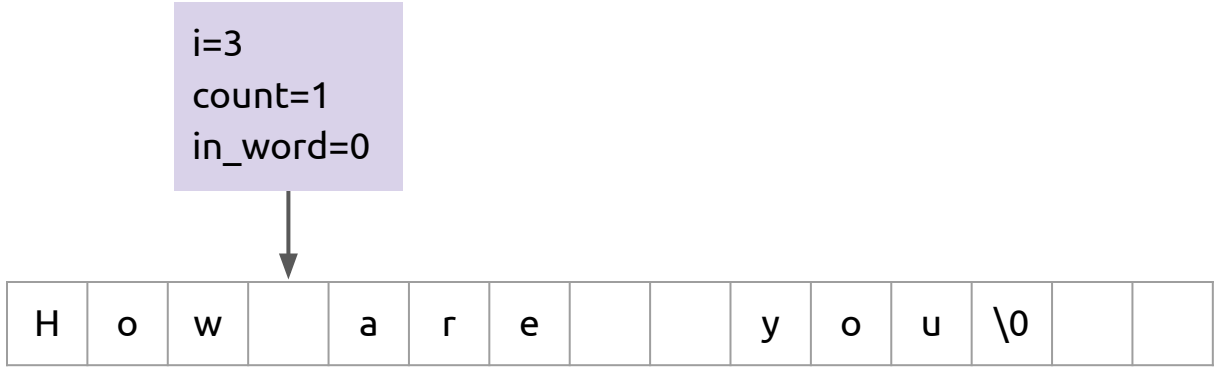



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                                in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

ψευδής

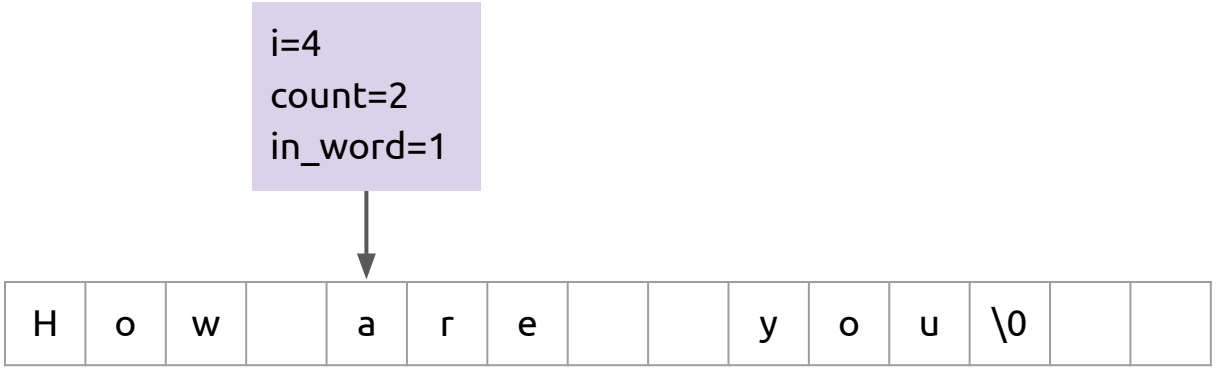
ψευδής

Άσκηση 6.1



ΑΝ (συναντήσεις κενό) ΤΟΤΕ — αληθής
  in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ — ψευδής
  count++
  in_word=1

Άσκηση 6.1

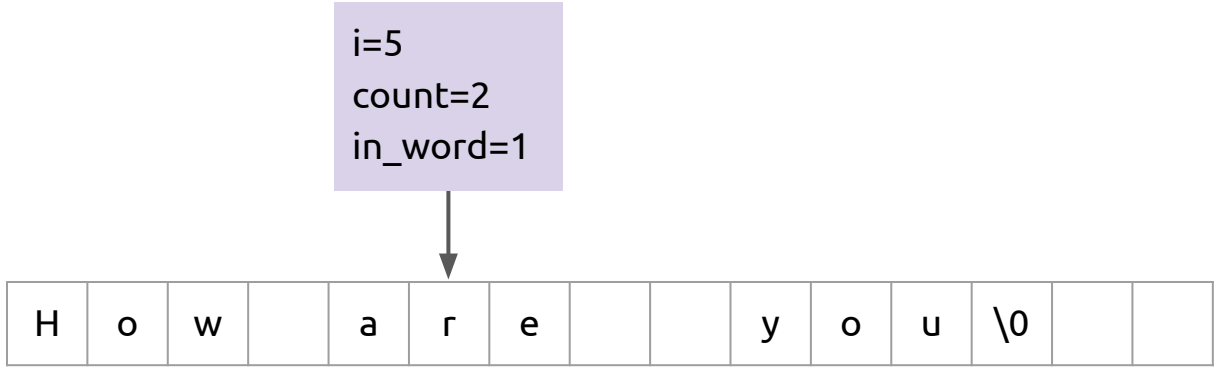



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                          in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

ψευδής

αληθής

Άσκηση 6.1

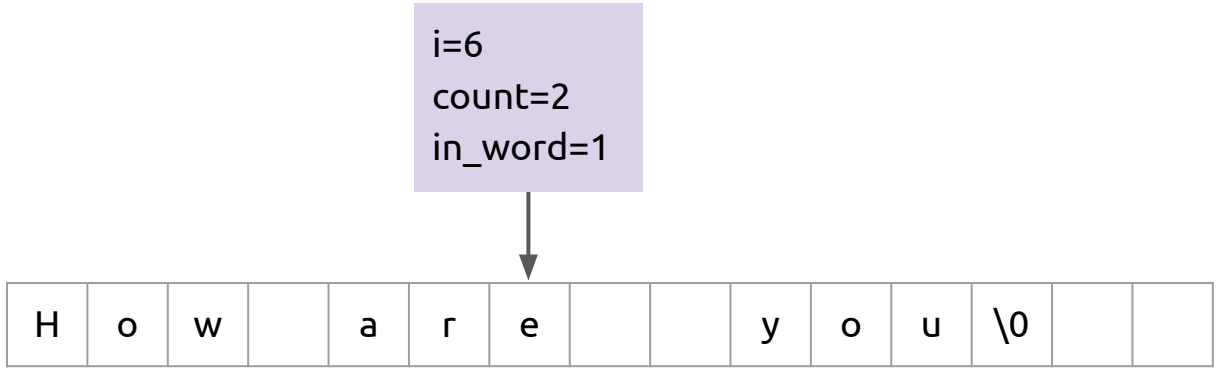



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                                in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

ψευδής

ψευδής

Άσκηση 6.1

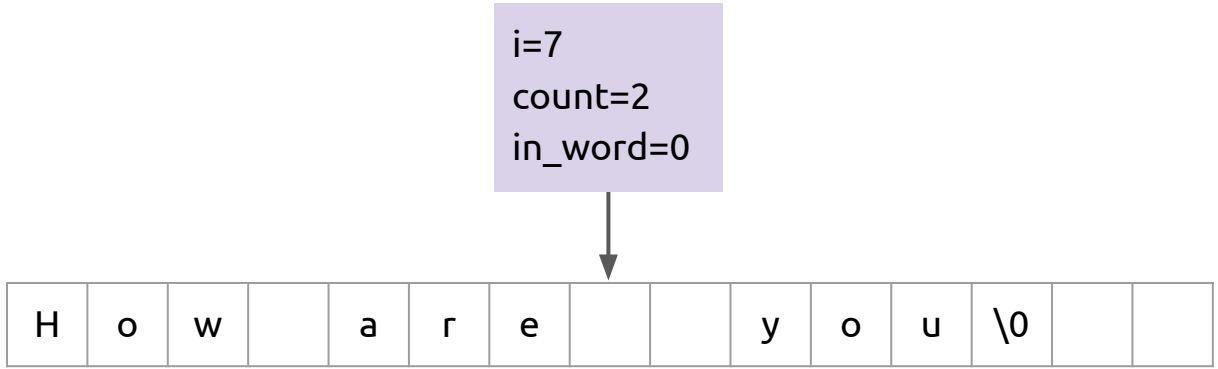



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                                in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

ψευδής

ψευδής

Άσκηση 6.1

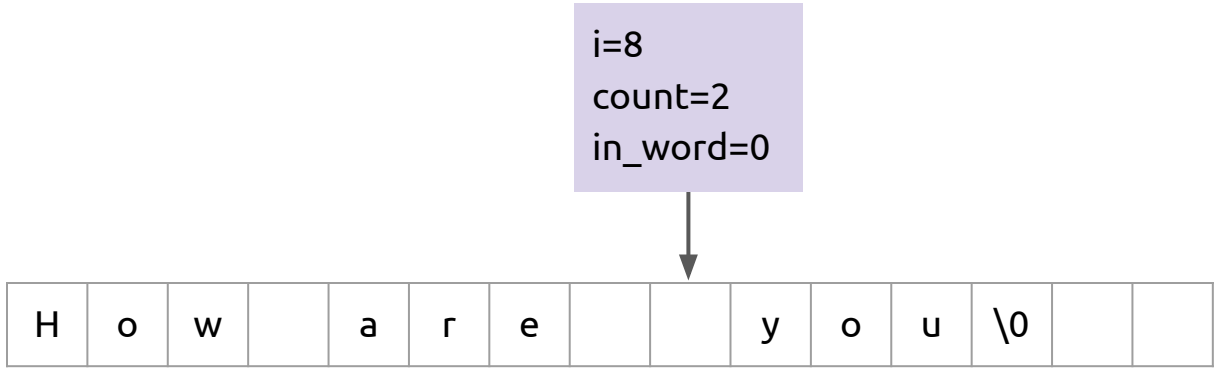



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                          in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

αληθής

ψευδής

Άσκηση 6.1

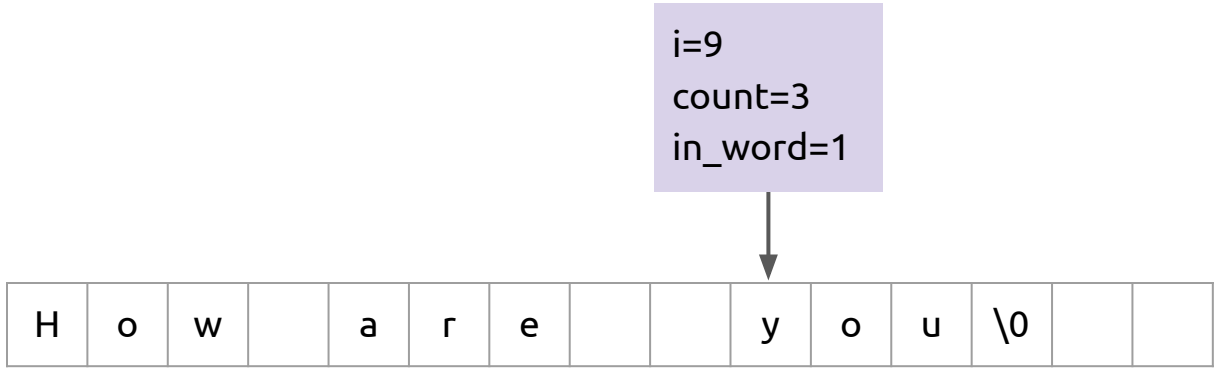



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                          in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

αληθής

ψευδής

Άσκηση 6.1

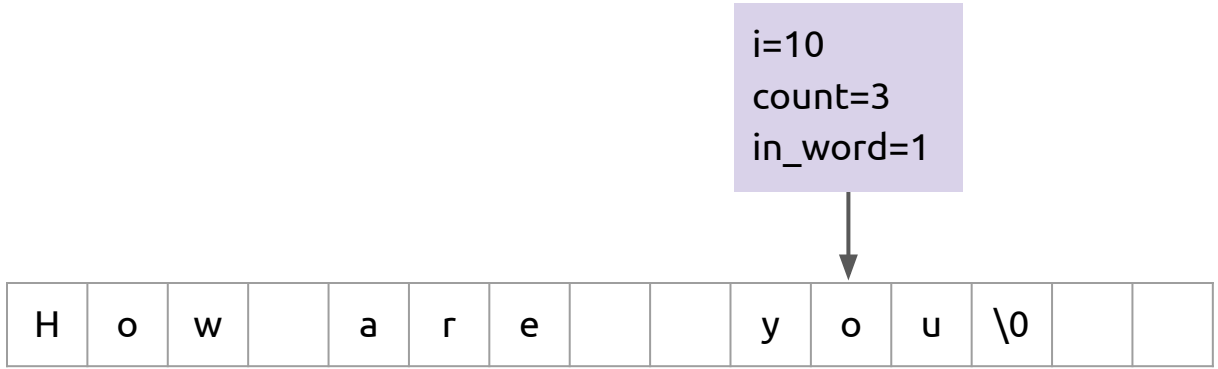



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                          in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

ψευδής

αληθής

Άσκηση 6.1

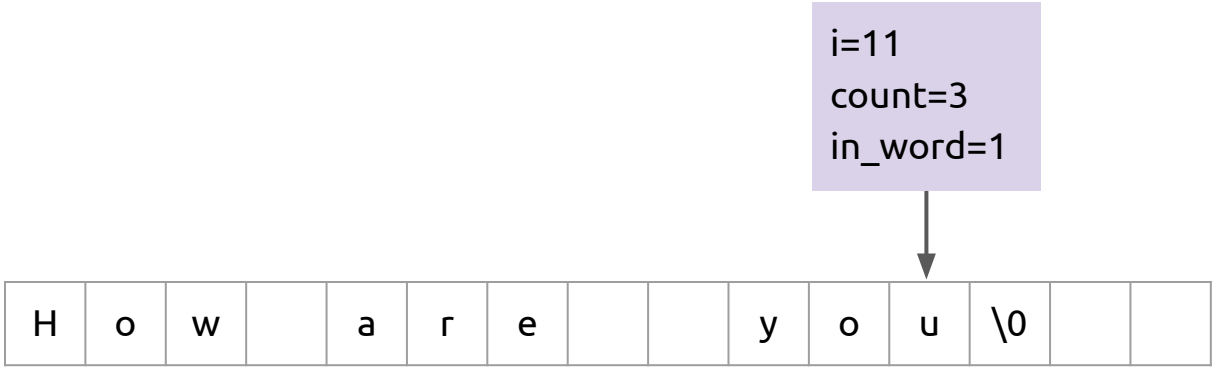



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                                in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

ψευδής

ψευδής

Άσκηση 6.1

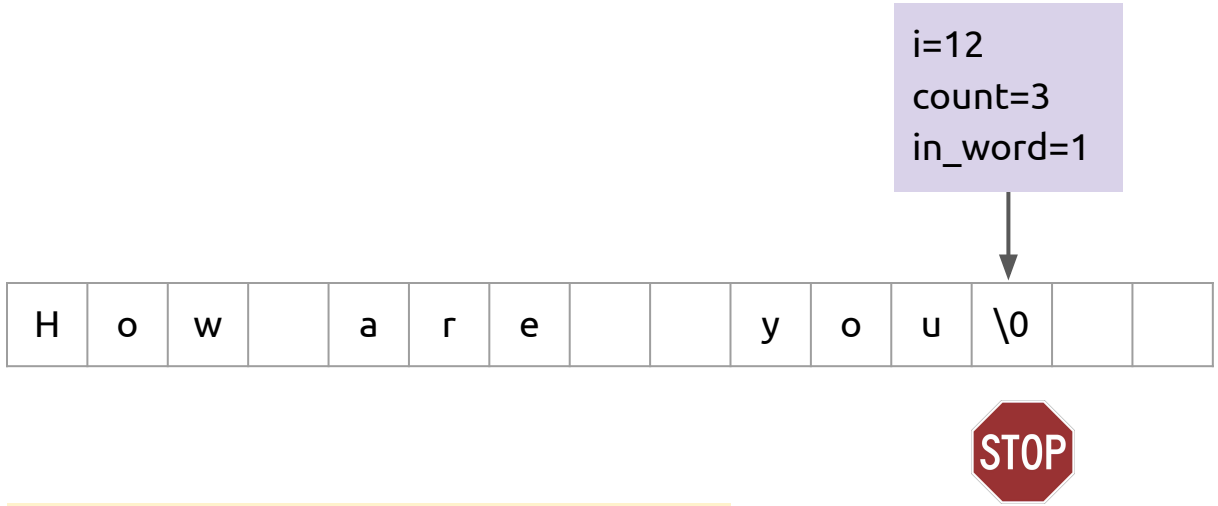



```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                          in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

ψευδής

ψευδής

Άσκηση 6.1




```
ΑΝ (συναντήσεις κενό) ΤΟΤΕ
                                in_word=0
ΑΛΛΙΩΣ ΑΝ (in_word==0) ΤΟΤΕ
  count++
  in_word=1
```

Άσκηση 6.1 - Λύση


```c
#include <stdio.h>
#include <ctype.h>
int word_cnt(char *str) {
    int count = 0; int i = 0;
    int in_word = 0;
    while(str[i] != '\0'){
        if (isspace(str[i])) {
            in_word = 0;
        } else {
            if (in_word == 0) {
                count++;
                in_word = 1;
            }
        }
        i++;
    }
return count;
}

int main() {
    char input[81];

    printf("Εισάγετε ένα αλφαριθμητικό: ");
    gets(input);

    int word_count = word_cnt(input);
    printf("Πλήθος λέξεων: %d\n", word_count);

    return 0;
}
```

Άσκηση 6.2

Να γραφεί μια συνάρτηση με όνομα `reverse_string` που θα δέχεται ως είσοδο ένα αλφαριθμητικό μέγιστου μήκους `80` χαρακτήρων και θα επιστρέφει το αλφαριθμητικό ανεστραμμένο (π.χ. από το αλφαριθμητικό `“production & management”` θα προκύπτει `“tnemeganam & noitcudorp”`).

Στη συνέχεια να γραφεί ένα πρόγραμμα που να χρησιμοποιεί τη συνάρτηση `reverse_string`. Η εισαγωγή των δεδομένων στη `main()` να γίνει με τη χρήση της `gets()`.

Να διερευνήσετε τη δυνατότητα χρήσης δυναμικής παραχώρησης μνήμης.

Άσκηση 6.2 - reverse_string


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

char* reverse_string(char* str) {
    int length = strlen(str);

    char* reversed = (char*)malloc((length + 1) * sizeof(char));

    for (int i = 0; i < length; i++) {
        reversed[i] = str[length - 1 - i];
    }
    reversed[length] = '\0';

    return reversed;
}
```

Άσκηση 6.2 - main


```c
int main() {
    char input[81];

    printf("Εισάγετε ένα αλφαριθμητικό: ");
    gets(input);

    char* reversed = reverse_string(input);

    printf("Ανεστραμμένο αλφαριθμητικό: %s\n", reversed);
    free(reversed);

    return 0;
}
```

Άσκηση 6.3

Χρησιμοποιώντας τις συναρτήσεις συμβολοσειρών που παρέχει η γλώσσα C για το χειρισμό αλφαριθμητικών (strings) να γράψετε ένα πρόγραμμα για το παρακάτω πρόβλημα:

1. Να εισάγετε από το πληκτρολόγιο ένα αλφαριθμητικό μέγιστου μήκους `80` χαρακτήρων με χρήση της `gets()`.
2. Να βρείτε και να εμφανίσετε τη συχνότητα εμφάνισης κάθε χαρακτήρα, με αλφαβητικό σειρά (δηλ. πρώτα τις συχνότητες των χαρακτήρων A,B,C,... και μετά τις συχνότητες των a, b, c,....). Για κάθε έναν από τους χαρακτήρες θα εμφανίζεται η τιμή του (δηλ. A, B, ...) και δίπλα το πλήθος των εμφανίσεων του στο αλφαριθμητικό ως εξής :

```
A    2
B    0
.....
g    1
.....
```

ΥΠΟΔΕΙΞΗ : δεν πρέπει να χρησιμοποιήσετε την εντολή `if`. Να χρησιμοποιήσετε τις τιμές των χαρακτήρων στον κώδικα ASCII δηλώνοντας κατάλληλα ένα μονοδιάστατο αριθμητικό πίνακα ακεραίων, με αρχικές μηδενικές τιμές, που θα περιέχει στο τέλος τις συχνότητες εμφάνισης.

Άσκηση 6.3 - main


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

int main() {
    char str[81];
    int freq[256] = {0};

    printf("Εισάγετε ένα αλφαριθμητικό: ");
    gets(str);

    for (int i = 0; i < strlen(str); i++)
        freq[str[i]]++;
```

```c
    for (int i = 'A'; i <= 'Z'; i++)
        printf("%c %d\n", i, freq[i]);
    for (int i = 'a'; i <= 'z'; i++)
        printf("%c %d\n", i, freq[i]);

    return 0;
}
```

Άσκηση 6.4

Να γραφεί μία συνάρτηση με όνομα `single_ch` που να έχει ως παράμετρο εισόδου ένα αλφαριθμητικό μέγιστου μήκους `80` χαρακτήρων. Η συνάρτηση να επιστρέφει ένα νέο αλφαριθμητικό που να περιέχει κατά σειρά όλους τους χαρακτήρες του αλφαριθμητικού εισόδου αλλά μόνον μία φορά τον καθένα δηλ. αν το αλφαριθμητικό εισόδου είναι `"engineering"` το νέο αλφαριθμητικό να είναι `"engir"`.

Στη συνέχεια να γραφεί ένα πρόγραμμα που να χρησιμοποιεί τη συνάρτηση `single_ch`. Η εισαγωγή των δεδομένων στη `main()` να γίνει με τη χρήση της `gets()`.

Άσκηση 6.4 - Λύση


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

void single_ch(char *input, char *output) {
    int freq[256] = {0};

    int j = 0;

    for (int i = 0; i < strlen(input); i++) {
        if (freq[input[i]] == 0) {
            output[j++] = input[i];
            freq[input[i]]++;
        }
    }
    output[j] = '\0';
}
```

```c
int main() {
    char input[81];
    char output[81];

    printf("Εισάγετε ένα αλφαριθμητικό: ");
    gets(input);

    single_ch(input, output);

    printf("Νέο αλφαριθμητικό: %s\n", output);

    return 0;
}
```