ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Σχολή Θετικών Επιστημών
ΤΜΗΜΑ ΑΕΡΟΔΙΑΣΤΗΜΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

104
Αλγόριθμοι, Προγραμματισμός και Δομές Δεδομένων

Εργαστήριο – Κεφάλαιο 10β
Αλφαριθμητικά

Διδάσκουσα:

Σοφία Αλμπάνη

Άσκηση 10.1

10.1) Να γραφεί πρόγραμμα σε C το οποίο διαβάζει μια συμβολοσειρά από το πληκτρολόγιο και στη συνέχεια επεξεργάζεται τον κάθε χαρακτήρα της ως εξής:

- Κάθε κενό (' ') να αντικαθίσταται με χαρακτήρα αλλαγής γραμμής ('\n').
- Κάθε εμφάνιση του χαρακτήρα 'a' να αντικαθίσταται με τον χαρακτήρα 'p'.
- Να μετριούνται οι εμφανίσεις του χαρακτήρα 'b'.

Στο τέλος το πρόγραμμα να εμφανίζει:

- το συνολικό μήκος του string,
- πόσες φορές βρέθηκε ο χαρακτήρας 'b',
- και το τελικό επεξεργασμένο string.

Στόχος

Εξάσκηση στη διαχείριση και επεξεργασία strings χαρακτήρα–χαρακτήρα, στη χρήση βρόχου για σάρωση ολόκληρης συμβολοσειράς, στην αντικατάσταση χαρακτήρων μέσα σε string και στον μετρητή εμφάνισης συγκεκριμένων χαρακτήρων.

Ενδεικτική Λύση 10.1

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

int main()
{
        char str[100];
        int i, cnt;

        printf("Enter text: ");
        scanf("%[^\n]",str);
        cnt = 0;
        for(i = 0; str[i] != '\0'; i++)
        {
                if(str[i] == ' ')
                        str[i] = '\n';
                else if(str[i] == 'a')
                        str[i] = 'p';
                else if(str[i] == 'b')
                        cnt++;
        }
        printf("Len = %d Times = %d\nText = %s\n", i, cnt, str);
}
```

Άσκηση 10.2

10.2) Κάντε την άσκηση 10.1 χρησιμοποιώντας **δείκτη (pointer)** αντί για δείκτη πίνακα.

Στόχος:
Εξάσκηση στη σάρωση και επεξεργασία strings με χρήση pointers αντί για πίνακες, στη χρήση της έκφρασης ptr++ για μετακίνηση μέσα στο string, στον υπολογισμό μήκους συμβολοσειράς με έκφραση ptr - str, και στη σύγκριση της μεθόδου pointer με την αντίστοιχη με χρήση ευρετηρίου (10.1)

Ενδεικτική Λύση 10.2

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

int read_text(char str[], int size, int flag);

int main()
{
          char *ptr, str[100];
          int cnt;

          printf("Enter text: ");
          fgets(str, 100, stdin);
          str[strcspn(str, "\n")] = '\0’;

          cnt = 0;
          ptr = str;
          while(*ptr != '\0')
          {
                    if(*ptr == ' ')
                              *ptr = '\n';
                    else if(*ptr == 'a')
                              *ptr = 'p';
                    else if(*ptr == 'b')
                              cnt++;
                    ptr++;
          }
          printf("Len = %d Times = %d\nText = %s\n", ptr-str, cnt, str);

}
```

Άσκηση 10.3

10.3) Να γραφεί πρόγραμμα σε C το οποίο:

- Χρησιμοποιεί έναν δείκτη σε string και εμφανίζει, σε κάθε επανάληψη, ολόκληρη τη συμβολοσειρά ξεκινώντας από την τρέχουσα θέση του δείκτη.
- Σε κάθε βήμα ο δείκτης να αυξάνεται κατά μία θέση (str++) ώστε στην επόμενη εκτύπωση η συμβολοσειρά να εμφανίζεται από τον επόμενο χαρακτήρα.
  Ο βρόχος να συνεχίζεται μέχρι να φτάσει στο τερματικό '\0' του string.

Π.χ. για το string "this" το πρόγραμμα πρέπει να εμφανίσει: this his is s

Στόχος

Εξάσκηση στη χρήση δεικτών για σάρωση και εκτύπωση μέρους μιας συμβολοσειράς, στη χρήση της συνθήκης *str ως έλεγχο μέχρι τον τερματικό χαρακτήρα, και στη χρήση post-increment (str++) για μετακίνηση δείκτη μέσα σε string,

Ενδεικτική Λύση 10.3

```
#include <stdio.h>
int main()
{
        char str[]= "this";
        char *ptr;

        ptr=str;

        for(; *ptr; printf("%s ", ptr++));

}
```

Άσκηση 10.4

10.4) Να γραφεί πρόγραμμα σε C το οποίο:

- Διαβάζει μία συμβολοσειρά από το πληκτρολόγιο και επαναλαμβάνει την είσοδο μέχρι η συμβολοσειρά να έχει **τουλάχιστον 3 χαρακτήρες**.
- Ζητά από τον χρήστη να εισαγάγει έναν χαρακτήρα.
- Ελέγχει αν ο χαρακτήρας αυτός εμφανίζεται **τρεις συνεχόμενες φορές** μέσα στο string (π.χ. xxx, aaa, bbb).
- Αν βρεθεί τέτοια τριάδα, εμφανίζει τη θέση στην οποία ξεκινούν οι τρεις διαδοχικοί ίδιοι χαρακτήρες.
- Αν δεν βρεθεί, εμφανίζει κατάλληλο μήνυμα ότι δεν υπάρχει τέτοια ακολουθία.

Στόχος της άσκησης

Εξάσκηση στον έλεγχο της εγκυρότητας μήκους ενός string, στη χρήση της getchar() για ανάγνωση μεμονωμένου χαρακτήρα, και στη διαχείριση της θέσης μέσα σε string.

Ενδεικτική Λύση 10.4

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

int main()
{
          char ch, str[100];
          int i, len;

          do
          {
                    printf("Enter text (more than 2 chars): ");
                    fgets(str, sizeof(str), stdin);
                    str[strcspn(str, "\n")] = '\0'; /* Αφαίρεση του \n αν υπάρχει */
                    len = strlen(str);
          } while(len < 3);

          printf("Enter character: ");
          ch = getchar();

          for(i = 0; i <= len-3; i++)
                    if(str[i] == ch && str[i+1] == ch && str[i+2] == ch)
                    {
                              printf("There are three successive '%c's in position %d\n", ch, i+1);
                              return 0;
                    }
          printf("There aren't three successive '%c's\n", ch);
          }
```

Ενδεικτική Λύση 10.4

```
# Με χρήση της read_text
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int read_text(char str[], int size, int flag);

int main()
{
          char ch, str[100];
          int i, len;

          do
          {
                    printf("Enter text (more than 2 chars): ");
                    len = read_text(str, sizeof(str), 1);
          } while(len < 3);

          printf("Enter character: ");
          ch = getchar();

          for(i = 0; i <= len-3; i++)
                    if(str[i] == ch && str[i+1] == ch && str[i+2] == ch)
                    {
                              printf("There are three successive '%c's in position %d\n", ch, i+1);
                              return 0;
                    }
          printf("There aren't three successive '%c's\n", ch);
}
```

Άσκηση 10.5

10.5) Να γραφεί πρόγραμμα σε C το οποίο διαβάζει μία συμβολοσειρά και στη συνέχεια:

- Μετρά πόσες φορές εμφανίζεται κάθε πεζό λατινικό γράμμα ('a'–'z'), αποθηκεύοντας τις εμφανίσεις σε πίνακα 26 θέσεων.
- Μετρά πόσες φορές εμφανίζεται κάθε ψηφίο ('0'–'9'), αποθηκεύοντας τις εμφανίσεις σε πίνακα 10 θέσεων.
- Εντοπίζει ποιο πεζό γράμμα εμφανίζεται **τις περισσότερες φορές** και πόσες.
- Εμφανίζει:
  - για κάθε γράμμα που εμφανίζεται τουλάχιστον μία φορά, πόσες φορές βρέθηκε,
  - για κάθε ψηφίο που εμφανίζεται τουλάχιστον μία φορά, πόσες φορές βρέθηκε,
  - και, τέλος, το γράμμα με τις περισσότερες εμφανίσεις.

Το πρόγραμμα πρέπει να αγνοεί χαρακτήρες που δεν είναι πεζά γράμματα ή ψηφία.

Στόχος

Εξάσκηση στη σάρωση συμβολοσειράς χαρακτήρα–χαρακτήρα, στη χρήση πινάκων για καταμέτρηση συχνοτήτων (frequency counting),και στην εύρεση του στοιχείου με τη μέγιστη συχνότητα εμφάνισης.

Ενδεικτική Λύση 10.5

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

int main(void)
{
        char ch, max_ch, str[100];
        int i, len, max_times, low_let[26] = {0};
        int dig[10] = {0};
        printf("Enter text: ");
        fgets(str, sizeof(str), stdin);
        str[strcspn(str, "\n")] = '\0';
        len=strlen(str);

        max_ch = max_times = 0;
        for(i = 0; i < len; i++)
        {
                if((str[i] >= 'a') && (str[i] <= 'z'))
                {
                        ch = str[i]-'a';
                        low_let[ch]++;
                        if(low_let[ch] > max_times)
                        {
                                max_times = low_let[ch];
                                max_ch = str[i];
                        }
                }
                else if((str[i] >= '0') && (str[i] <= '9'))
                        dig[str[i]-'0']++;
        }
```

Ενδεικτική Λύση 10.5

```
#         printf("***** Lower case letters appearances\n");
          for(i = 0; i < 26; i++)
                    if(low_let[i] != 0) /* Έλεγχος για το αν υπάρχει μία τουλάχιστον εμφάνιση του γράμματος. */
                              printf("Letter %c appeared %d times\n", 'a'+i, low_let[i]);

          printf("***** Digits appearances\n");
          for(i = 0; i < 10; i++)
                    if(dig[i] != 0)
                              printf("Digit %d appeared %d times\n", i, dig[i]);
          if(max_times != 0)
                    printf("'%c' appears %d times\n", max_ch, max_times);
          return 0;
}
```

Άσκηση 10.6

10.6) Να γραφεί πρόγραμμα σε C το οποίο:

- Διαβάζει **20 ονόματα με χρήση της συνάρτησης read_text** και τα αποθηκεύει σε έναν δισδιάστατο πίνακα χαρακτήρων.
- Χρησιμοποιεί έναν πίνακα δεικτών (ptr[]), όπου κάθε δείκτης δείχνει στο αντίστοιχο όνομα του δισδιάστατου πίνακα.
- Ταξινομεί τα ονόματα **αλφαβητικά**, όχι μετακινώντας τα ίδια τα strings, αλλά **αντιμεταθέτοντας τους δείκτες** του πίνακα ptr.
- Μετά την ταξινόμηση, εμφανίζει τα ονόματα με τη σωστή αλφαβητική σειρά.

Η ταξινόμηση να γίνεται συγκρίνοντας τα strings με τη συνάρτηση strcmp.

Στόχος της άσκησης

Εξάσκηση στο πώς αποθηκεύονται πολλά strings σε δισδιάστατο πίνακα χαρακτήρων, πώς λειτουργεί η αλφαβητική ταξινόμηση με χρήση της strcmp, πώς γίνεται η αντιμετάθεση δεικτών (pointer swapping), και τη διαφορά ανάμεσα σε “μετακίνηση δεδομένων” και “μετακίνηση δεικτών”.

Ενδεικτική Λύση 10.6

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

#define SIZE 20

int read_text(char str[], int size, int flag);

int main(void)
{
        char *tmp, *ptr[SIZE], str[SIZE][100];
        int i, j;

        for(i = 0; i < SIZE; i++)
        {
                printf("Enter name: ");
                read_text(str[i], sizeof(str[i]), 1);
                ptr[i] = str[i];                    /* Τα στοιχεία του πίνακα δεικτών δείχνουν στα αλφαριθμητικά. */
        }
        for(i = 0; i < SIZE-1; i++)
        {
                for(j = i+1; j < SIZE; j++)
                {
                        if(strcmp(ptr[j], ptr[i]) < 0)
                        {
                                tmp = ptr[j];
                                ptr[j] = ptr[i];
                                ptr[i] = tmp;
                        }
                }
        }
        for(i = 0; i < SIZE; i++)
                printf("%s\n", ptr[i]);
        }
```

```
/* Αν το αλφαριθμητικό στο οποίο δείχνει ο ptr[j]
είναι μικρότερο από το αλφαριθμητικό στο οποίο
δείχνει ο ptr[i], αντιμεταθέτουμε τους
αντίστοιχους δείκτες. */
```

Ενδεικτική Λύση 10.6

Η συνάρτηση read_text:

```
int read_text(char str[], int size, int flag)
{
    int len;

    if(fgets(str, size, stdin) == NULL)
    {
        printf("Error: fgets() failed\n");
        exit(EXIT_FAILURE);
    }

    len = strlen(str);
    if(len > 0)
    {
        if(flag && (str[len - 1] == '\n'))
        {
            str[len - 1] = '\0';
            len--;
        }
    }
    else
    {
        printf("Error: No input\n");
        exit(EXIT_FAILURE);
    }

    return len;
}
```

Συνοπτικά

Μέθοδος	Συντακτικό / Παράδειγμα	Τι διαβάζει	Πλεονεκτήματα	Μειονεκτήματα
`scanf("%s", str);`	`scanf("%49s", str);`	Μόνο **μία λέξη** (σταματά σε space, tab, newline)	✓ Απλό στη χρήση✓ Γρήγορο	✗ Δεν διαβάζει φράσεις ✗ Εύκολο buffer overflow αν δεν βάλεις όριο
`fgets()`	`fgets(str, size, stdin);`	**Ολόκληρη γραμμή**, μαζί με το `\n`	✓ Πιο ασφαλές ✓ Επιτρέπει φράσεις με κενά ✓ Έλεγχος μεγέθους	✗ Κρατάει το `\n` και πρέπει να αφαιρεθεί
Αφαίρεση newline από `fgets`	`str[strcspn(str,"\n")] = '\0';`	—	✓ Κάνεις σωστό καθάρισμα string	—
`getchar()` (πολλές φορές)	`while((ch=getchar())!='\n') str[i++]=ch;`	Ό,τι γράψει ο χρήστης μέχρι `Enter`	✓ Απόλυτος έλεγχος εισόδου ✓ Κατάλληλο για ασκήσεις χαρακτήρα-χαρακτήρα	✗ Θέλει χειροκίνητη διαχείριση τερματισμού ✗ Περισσότερος κώδικας
`scanf("%[^\n]s", str);`	`scanf("%49[^\n]", str);`	Όλη τη γραμμή **εκτός του `\n`**	✓ Διαβάζει φράση ✓ Δεν κρατά newline	✗ Δεν διαβάζει κενή γραμμή ✗ Παγιδεύεται αν υπάρχει προηγούμενο newline στο buffer
`scanf(" %c", &ch);` (μόνο χαρακτήρας)	`scanf(" %c", &ch);`	Έναν χαρακτήρα, αγνοώντας whitespace	✓ Ιδανικό για menu επιλογών	✗ Όχι για strings

Ανάγκη	Καλύτερη μέθοδος
Μία λέξη	`scanf("%s", str)`
Φράση με κενά	`fgets(str, size, stdin)`
Πλήρης έλεγχος χαρακτήρα-χαρακτήρα	`getchar()`
Λήψη γραμμής χωρίς newline	`scanf("%[^\n]", str)` *ή* `fgets` + αφαίρεση `\n`
Μόνο ένας χαρακτήρας	`scanf(" %c", &ch)`