Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

Εισαγωγή στον Προγραμματισμό

Η γλώσσα προγραμματισμού C

Συναρτήσεις Αλφαριθμητικών

- <stdio.h>

```
char *gets(char *s)    // διαβάζει string από το stdin στο s
                       // και επιστρέφει το s

int puts(const char *s)  // γράφει το string s στο stdout

Char * fgets(char*s, int n, FILE *fstream); // διαβάζει το πολύ n-1
   //χαρακτήρες σταματώντας αν συναντήσει χαρακτήρα νέας γραμμής
```

- <string.h>

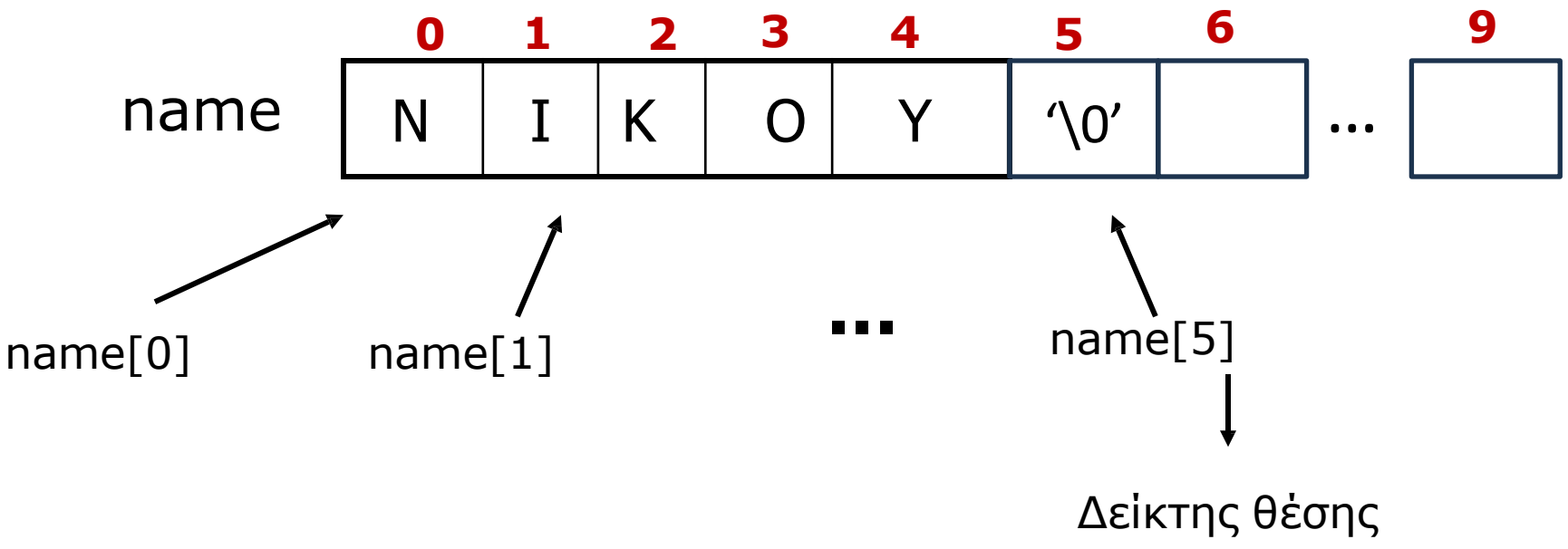
```
int  strlen(cs)
int strcmp(cs,ct)                    int strncmp(cs, ct, count);
char *strcpy(dest,source)          char *strncpy(dest, source, count)
char *strcat(destination, source)
```

Αλφαριθμητικά (εμφάνιση με printf()

Μπορούμε να εμφανίσουμε όποιο τμήμα του αλφαριθμητικού θέλουμε ξεκινώντας από την αντίστοιχη θέση. Για παράδειγμα για να εκτυπώσουμε στο αλφαριθμητικό name (char name[]) από την τέταρτη θέση και μετά, αρκεί να γράψουμε printf("%s", name+3) ή printf(“%s”, &name[3]);

```
char name[N]="NIKOY";
/* char name[N]="NIKOY";
   char name[]="NIKOY"; */
int len, i;
len=strlen(name);
for (i=0; i<len; i++)
    // printf("%s\n", name+i);
     printf("%s\n", &name[i]);
return 0;
```

```
#define N 10
char name[N]="NIKOY";
```



Μεταβίβαση πινάκων σε συναρτήσεις

Πίνακες μίας διάστασης

```
#include <stdio.h>
#include <stdlib.h>
double getAverage(int array[], int size);
/* Ισοδύναμα double getAverage(int *array, int size); */
main (){
   /* Ορισμός πίνακα */
   int balance[5] = {30, 15, 5, 0, 6};
   double avg;
   system("chcp 1253>nul");
   /* Μεταβιβάζουμε τον πίνακα στη συνάρτηση ως δείκτη */
   avg = getAverage(balance, 5);
   printf("Η μέση τιμή είναι: %f ", avg );
}
double getAverage(int array[], int size){
  int i;
  double sum=0.0;
  for (i = 0; i < size; i++)
    sum += array[i];
  return sum/size;
}
```

Άσκηση με αλφαριθμητικό

Α) Γράψτε ένα πρόγραμμα σε C σύμφωνα με το οποίο το σύστημα θα διαβάζει ένα αλφαριθμητικό από την βασική είσοδο και ένα χαρακτήρα και θα τυπώνει το πλήθος εμφανίσεων του χαρακτήρα στο αλφαριθμητικό.

Β) Δώστε μια άλλη εκδοχή του παραπάνω προγράμματος ορίζοντας και αξιοποιώντας μια συνάρτηση η οποία δέχεται ως όρισμα ένα αλφαριθμητικό και ένα χαρακτήρα και θα επιστρέφει το πόσες φορές εμφανίζεται στο αλφαριθμητικό ο συγκεκριμένος χαρακτήρας.

Η συνάρτηση `strlen()`

Η συνάρτηση `strlen()` δηλώνεται στο αρχείο `string.h` και επιστρέφει τον αριθμό των χαρακτήρων που περιέχει ένα αλφαριθμητικό, χωρίς να μετράει τον τερματικό χαρακτήρα (`'\0'`)

Η συνάρτηση δηλώνεται ως:

```
int strlen(char str[]);
```

```
#include <stdio.h>
#include <string.h>
#define N 100
int main()
{
        int len;
        char name[N];
        printf("Give name: ");
        scanf("%s", name);

        len=strlen(name);
        printf("The length is %d\n", len);
        return 0;
}
```

Παράδειγμα (1)

```
int strlen(char  s[ ])
{
        int  i=0;
        while ((s[i]!='\0') i++;
        return i;
}
```

```
int strlen(char *s)
{
        int i=0;
        while (*s != '\0')  {s++; i++;}
        return i;
}
```

Η συνάρτηση `strcpy()`

```
char source[10];
char dest[10];
dest=source; /* Λάθος */
```

Ένας σωστός τρόπος να αντιγράψουμε ένα αλφαριθμητικό είναι με τη συνάρτηση `strcpy()` που δηλώνεται στο αρχείο `string.h` και χρησιμοποιείται για την αντιγραφή ενός αλφαριθμητικού σε μία άλλη θέση μνήμης (string copy)

Η συνάρτηση strcpy() δέχεται σαν παραμέτρους δύο αλφαριθμητικά και αντιγράφει το δεύτερο αλφαριθμητικό στο πρώτο.

Τ

Το πρωτότυπό της δηλώνεται ως εξής:

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

Η συνάρτηση `strcpy()` τερματίζει και επιστρέφει τον δείκτη `dest`.

Παράδειγμα (2)

```
char * strcpy(char s[ ], char t[ ])
{
        int  i=0;
        while ((s[i]=t[i])!='\0')
        i++;
        return s;
}
```

```
char * strcpy(char *s, char *t)
{
        while ((*s=*t) != '\0') { s++; t++;}
        return s;
}
```

```
char * strcpy(char *s, char *t)
{
        while ((* s++=*t++) != '\0') ;
        return s;
}
```

```c
#include <stdio.h>

char * strcpy(char s[], char t[])
{
        int i=0;
        while ((s[i]=t[i])!='\0') i++;
        return s;
}


int main()
{
        char s[10];
        char t[10];
        scanf("%s", s);
        scanf("%s", t);
        printf("%s", strcpy(s,t));
}
```

Η συνάρτηση `strcmp()`

Δηλώνεται στο αρχείο `string.h` και χρησιμοποιείται για τη σύγκριση αλφαριθμητικών. Η συνάρτηση `strcmp()` δέχεται σαν παραμέτρους δύο δείκτες και συγκρίνει το αλφαριθμητικό στο οποίο δείχνει ο δείκτης str1 με το αλφαριθμητικό στο οποίο δείχνει ο δείκτης str2

Το πρωτότυπό της δηλώνεται ως εξής:

```
int strcmp(char str1[], char str2[]);
```

Αν τα δύο αλφαριθμητικά είναι ακριβώς ίδια, τότε η συνάρτηση strcmp() επιστρέφει την τιμή μηδέν (0)

Αν το πρώτο αλφαριθμητικό είναι μικρότερο από το δεύτερο, τότε επιστρέφει μία αρνητική τιμή, ενώ αν είναι μεγαλύτερο επιστρέφει μία θετική τιμή

Η συνάρτηση `strcmp()`

Ένα αλφαριθμητικό θεωρείται μικρότερο από κάποιο άλλο αν ισχύει ένα από τα παρακάτω:

α) οι πρώτοι n χαρακτήρες των αλφαριθμητικών είναι ίδιοι, αλλά η τιμή του πρώτου μη κοινού χαρακτήρα στο πρώτο αλφαριθμητικό είναι μικρότερη από την τιμή του αντίστοιχου χαρακτήρα στο δεύτερο αλφαριθμητικό

β) οι χαρακτήρες τους είναι οι ίδιοι, αλλά το μήκος του πρώτου αλφαριθμητικού είναι μικρότερο

Παράδειγμα (3)

```
int strcmp(char s[ ], char  t[ ])
{
int  i=0;
for (ı=0; (s[i]==t[i])&&s[i]!='\0'; i++);
return s[i]-t[i];
}
```

```
int strcmp(char *s, char *t)
{
for (; (*s==*t) && *s!='\0'; s++, t++);
return  *s-*t;
}
```

```c
#include <stdio.h>

int strcmp(char s[ ], char  t[ ])
{
        int i=0;
        for (i=0; (s[i]==t[i])&&s[i]!='\0'; i++);
        return s[i]-t[i];
}

int main()
{
        char s[20];
        char t[10];
        scanf("%s", s);
        scanf("%s", t);
        printf("%d", strcmp(s,t));
        return 0;
}
```

Η συνάρτηση `strcat()`

Η συνάρτηση strcat() δηλώνεται στο αρχείο string.h και συνενώνει ένα αλφαριθμητικό με ένα άλλο Η συνάρτηση strcat() δέχεται σαν παραμέτρους δύο αλφαριθμητικά και προσθέτει το αλφαριθμητικό source στο τέλος του dest Ο τερματικός χαρακτήρας '\0' προστίθεται αυτόματα στο τέλος του νέου αλφαριθμητικού

Το πρωτότυπό της δηλώνεται ως εξής:

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

Παράδειγμα (3)

```
char* strcat(char s[], char t[])
{
        int  i=0, j=0;
        for (ι=0;  s[i]!='\0'; i++);
        while ((s[i]=t[j])!='\0')  { i++;  j++;}
        return s;
}
```

```c
#include <stdio.h>
char * strcat(char s[], char t[])
{
        int i=0, j=0;
        for (i=0; s[i]!='\0'; i++);
        while ((s[i]=t[j])!='\0')  { i++;  j++;}
        return s;
}
int main()
{
        char s[20];
        char t[10];
        scanf("%s", s);
        scanf("%s", t);
        printf("%s“, strcat(s,t));
        printf("%s", s);
        return 0;
}
```

Δ3 – Τέσσερις Πράξεις (Menu)

Αναπτύξτε ένα πρόγραμμα με όνομα Menu σύμφωνα με το οποίο το σύστημα θα εκτελεί τις υπολογιστικές διεργασίες που περιγράφει η Άσκηση 3 (Κεφάλαιο 11).

Η άσκηση ως λειτουργικότητα είναι πολύ απλή αλλά εισάγει τη δόμηση του προγράμματος με βάση ένα μενού επιλογής που δίνει τη δυνατότητα στον χρήστη να επιλέξει τη διεργασία που αυτός επιθυμεί να εκτελέσει η μηχανή. Αποτελεί καλή εξάσκηση για τις προτάσεις ελέγχου ροής και τη διαδικασία ανάπτυξης με την incremental development τεχνική. Αγνοήστε την Άσκηση 3β (παράγραφος 11.3.3)

switch: Πολλαπλή επιλογή

- Όταν βάσει μιας έκφρασης θέλουμε να επιλέξουμε ανάμεσα σε πολλές επιλογές, είναι πιο «βολική» η χρήση της **switch**.

```
switch (<έκφραση>)
{
  case <σταθ-εκφρ-1>: <προτ-1>; break;
  case <σταθ-εκφρ-2>: <προτ-2>; break;
      …
  case <σταθ-εκφρ-N>: <προτ-N>; break;
  default: <πρόταση>; break;
}
```

switch: Πολλαπλή επιλογή

- Κάθε <σταθ-έκφρ-i> πρέπει να είναι **μία τιμή int ή char** ή μία έκφραση μόνο με τέτοιες τιμές
- Δύο <σταθ-εκφρ-i> δεν μπορούν να έχουν την ίδια τιμή
- Αν <έκφραση>=<σταθ-εκφρ-x> τότε εκτελούνται όλες οι παρακάτω της x προτάσεις
  - ✓ Για να το αποτρέψουμε αυτό, χρειάζεται η **break**!
- Η <πρόταση> εκτελείται μόνο όταν καμμιά από τις <προτ-i> δεν ικανοποιείται
- Δεν υποστηρίζονται περιοχές τιμών (ranges), μόνο ισότητα
- Η default δεν είναι απαραίτητο να είναι στο τέλος

Παράδειγμα

```
switch (choice)   {
      case 1:
        x=a+b;
        break;
      case 2:
        x=a-b;
        break;
      case 3:
        x=a*b;
        break;
      case 4:
        x=a/b;
        break;
      default:
        printf("Ανύπαρκτη επιλογή");
        break;
}
```

Παράδειγμα

```
switch (choice) {
      case 1: x=a+b; break;
      case 2: x=a-b; break;
      case 3: x=a*b;break;
      case  4:  x=a/b;break;
      default:
        printf("Ανύπαρκτη επιλογή");
        break;
}
```

```
/* Εκτέλεση λειτουργίας */
switch(choice){
        case 1 :
                /* Εισαγωγή των δύο αριθμών *
                        insert(a,b) …
                        break;
        case 'A' :
                /* πρόσθεση δύο αριθμών */
                        add(a,b) ...
                        break;
        …

        default :
                printf("Unknown command\n");
                break;
                }
```