# Προγραμματισμός σε C - Αλφαριθμητικά

Ιωάννης Γ. Τσούλος

Τμήμα Πληροφορικής και τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων

2023

# Περίληψη

# Ορισμός

- Οι μεταβλητές χαρακτήρα χρησιμοποιούνται για να δηλώσουν ένα γράμμα όπως είναι πχ το A, το # κτλ.
- Ο τύπος που χρησιμοποιείται για τέτοιες μεταβλητές είναι ο τύπος **char**.
- Μια μεταβλητή **char** καταλαμβάνει ακριβώς 1 byte στην μνήμη του υπολογιστή.

# Ανάγνωση χαρακτήρα

```
# include <stdio.h>
int main()
{
 char a = 'a';
 char b = '1';
 printf("Enter␣char␣variable\n");
 scanf("%c",&b);
 printf("You␣have␣entered␣%c\n",b);
 return 0;
}
```

# Ανάγνωση χαρακτήρα με scanf

- Για την ανάγνωση χαρακτήρα με scanf απαιτείται ο προσδιοριστής c.
- Μετά από κάθε είσοδο θα πρέπει ο χρήστης να πατήσει το ENTER
- Δεν είναι βολικός αυτός ο τρόπος εισόδου, ειδικά όταν θα πρέπει να δωθούν συνεχόμενοι χαρακτήρες.
- Μια επιπλέον συνάρτηση που μπορεί να χρησιμοποιηθεί προκειμένου να γίνει ανάγνωση χαρακτήρων είναι η συνάρτηση getchar().
- Η συνάρτηση αυτή διαβάζει ένα γράμμα από την καθιερωμένη είσοδο και το επιστρέφει.

# Ανάγνωση χαρακτήρα με getchar()

```
# include <stdio.h>
int main()
{
        char ch;
        int charcount =0;
        while((ch=getchar())!='\n')
        {
                charcount++;
        }
        printf("Total Chars in line: %d \n",
      charcount);
        return 0;
}
```

# Ανάγνωση χαρακτήρα με getchar()

- Στο προηγούμενο παράδειγμα αυτό γίνεται ανάγνωση από την καθιερωμένη είσοδο μέχρι ο χρήστης να εισάγει την αλλαγή γραμμής.
- Σε κάθε επανάληψη ο μετρητής charcount αυξάνεται κατά 1.
- Στο τέλος γίνεται εμφάνιση του πλήθους των χαρακτήρων της γραμμής.

# Παράδειγμα εμφάνισης γραμμάτων και ASCII κωδικών.

```
# include <stdio.h>
int main()
{
        char letter;
        int code;
        char nextLetter;
        int nextCode;
        letter='A';
        code=letter;
        nextLetter=letter+1;
        nextCode=nextLetter;
        printf("To␣proto␣gramma␣%c␣\n",letter);
        printf("O␣protos␣kodikos␣%d␣\n",code);
        printf("To␣deutero␣gramma␣%c␣\n",nextLetter);
        printf("O␣epomenos␣kodikos␣%d\n",nextCode);
        return 0;
}
```

# Εμφάνιση γραμμάτων και ASCII κωδικών

- Παρόλο που το όνομα του τύπου είναι **char**, τέτοιες μεταβλητές μπορούν να χρησιμοποιηθούν και για αναπαράσταση ακέραιων αριθμών.
- Στο προηγούμενο παράδειγμα βλέπουμε πως οι μεταβλητές τύπου char μπορούν να χρησιμοποιηθούν και σαν ακέραιοι αριθμοί με προσθέσεις, αναθέσεις κτλ.
- Επιπλέον οι κωδικοί 65 και 66 που εμφανίζοναι για τα γράμματα A και B είναι οι αντίστοιχοι κωδικοί αυτών των γραμμάτων στον πίνακα ASCII.

# Μετατροπή μικρών γραμμάτων σε κεφαλαία

```
# include <stdio.h>
int main()
{
        char letter,uppercase;
        do{
          printf("Doste gramma # gia termatismo \n");
          scanf("%c",&letter);
          if(letter>='a' && letter<='z'){
          uppercase=letter-('a'-'A');//32
          printf("Kefalaio gramma %c\n",uppercase);
          }
        else
          printf("Eidiko gramma %c\n",letter);
        }while(letter!='#');
        return 0;
}
```

# Μετατροπή γραμμάτων σε κεφαλαία

- Στον πίνακα ASCII τα κεφαλαία γράμματα βρίσκονται πριν από τα μικρά στις θέσεις 65..89.
- Τα μικρά γράμματα βρίσκονται μετά την θέση 97.
- Επιπλέον τα γράμματα είναι συνεχόμενα, που σημαίνει πως για παράδειγμα το B βρίσκεται μετά το γράμμα A. Στον προηγούμενο αλγόριθμο εκμεταλλευόμαστε αυτήν την κατάσταση και γίνεται μετατροπή κάθε μικρού γράμματος στο αντίστοιχο κεφαλαίο.
- Η ανάγνωση των γραμμάτων γίνεται επαναληπτικά και σταματά με την είσοδο του ειδικού γράμματος #.
- Μπορούμε να κάνουμε έλεγχο για το διάστημα ενός γράμματος με χρήση απλά των λογικών τελεστών, μιας και οι μεταβλητές χαρακτήρα είναι και ακέραιες τιμές.

# Συναρτήσεις του ctype.h

| Συνάρτηση | Σημασία |
|---|---|
| int isalnum(int x); | Αληθές αν το x είναι γράμμα ή ψηφίο. |
| int isalpha(int x); | Αληθές, αν το x είναι γράμμα. |
| int iscntrl(int x); | Αληθές, αν το x είναι χαρακτήρας ελέγχου. |
| int isdigit(int x); | Αληθές, αν το x ειναι αριθμητικό σύμβολο. |
| int islower(int x); | Αληθές, αν το x είναι μικρό λατινικό γράμμα. |
| int isprint(int x); | Αληθές, αν το x είναι εκτυπώσιμο |
| int ispunct(int x); | Αληθές, αν το x είναι σημείο στίξης. |
| int isspace(int x); | Αληθές, αν το x είναι κενό. |
| int isupper(int x); | Αληθές, αν το x είναι κεφαλαίο |
| int isxdigit(int x); | Αληθές, αν το x είναι δεκαεξαδικός αριθμός |
| int tolower(int x); | Επιστρέφει το αντίστοιχο πεζό |
| int toupper(int x); | Επιστρέφει το αντίστοιχο κεφαλαίο |

# Παράδειγμα χρήσης συναρτήσεων από το ctype.h

```
# include <stdio.h>
# include <ctype.h>
int main()
{
        char ch;
        int charcount =0,digitcount =0,
         lowercount =0;
        while((ch=getchar())!='\n')  {
                charcount++;
                if(isdigit(ch)) digitcount++;
                if(islower(ch)) lowercount++;
        }
        printf("Total Chars in line: %d \n",charcount);
        printf("Digits: %d\n",digitcount);
        printf("Lower letters: %d\n",lowercount);
        return 0;
}
```

# Εισαγωγικά στοιχεία για αλφαριθμητικά

- Βάζοντας πολλά γράμματα μαζί σε έναν πίνακα δημιουργείται ένα αλφαριθμητικό για την αναπαράσταση λέξεων και προτάσεων.
- Ο προγραμματιστής πρέπει να ξέρει που βρίσκεται το τέλος ενός αλφαριθμητικού μέσα σε έναν πίνακα.
- Αυτό γίνεται με την χρήση του ειδικού χαρακτήρα '\0' για τον καθορισμό του τέλους.
- Προσοχή πρέπει να δοθεί στο γεγονός πως αυτός ο χαρακτήρας (NULL) δεν είναι το '0' αλλά το γράμμα που βρίσκεται στην θέση 0 του πίνακα ASCII, το οποίο είναι μη εκτυπώσιμο.

# Εισαγωγή γραμμάτων σε αλφαριθμητικό με διαδοχικές αναθέσεις

```
# include <stdio.h>
int main()
{
        char X[10];
        X[0]='A';
        X[1]='R';
        X[2]='T';
        X[3]='A';
        X[4]='\0';
        printf("X is %s\n",X);
        return 0;
}
```

# Εισαγωγή στοιχείων αλφαριθμητικού με διαδοχικές αναθέσεις

- Για την εμφάνιση του αλφαριθμητικού με την χρήση της συνάρτησης printf() χρησιμοποιείται ο προσδιοριστής %s.
- Αν και το αλφαριθμητικό έχει δηλωθεί να έχει 10 στοιχεία, στην πραγματικότητα σε αυτό γίνεται εισαγωγή 4 γραμμάτων (A,R,T,A) και του ειδικού χαρακτήρα τερματισμού στο τέλος των γραμμάτων. Στις υπόλοιπες 5 θέσεις δεν γίνεται κάποια εισαγωγή.
- Επομένος ο πίνακας X έχει τα ακόλουθα περιεχόμενα

| ΠΕΡΙΕΧΟΜΕΝΟ | A | R | T | A | \0 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ΘΕΣΗ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

# Αρχικοποιήσεις αλφαριθμητικών με αναθέσεις

```
# include <stdio.h>
int main()
{
 char word1[10]="Simple";
 char word2[]="Another";
 printf("Entered␣Strings␣%s␣%s␣\n",word1,word2);
 return 0;
}
```

# Αρχικοποιήσεις αλφαριθμητικών με αναθέσεις

- Στο προηγούμενο παράδειγμα παρουσιάζονται δύο εναλλακτικοί τρόποι με τους οποίους μπορεί κανείς να γεμίσει ένα αλφαριθμητικό.
- Στην πρώτη περίπτωση ο χρήστης δηλώνει έναν πίνακα γραμμάτων με 10 θέσεις και σε αυτόν γίνεται εισαγωγή 6 γραμμάτων και του χαρακτήρα τερματισμού.
- Στην δεύτερη περίπτωση, αφήνοντας κενή την διάσταση του πίνακα, το μέγεθος προσδιορίζεται δυναμικά από τον μεταγλωττιστή.
- Στην δεύτερη περίπτωση ο πίνακας θα έχει χωρητικότητα $7+1 = 8$ θέσεων.

# Εισαγωγή αλφαριθμητικών

- Να κάνει εισαγωγή τα γράμματα του αλφαριθμητικού ένα προς ένα. Φυσικά αυτός ο τρόπος είναι αρκετά απλός αλλά δεν μπορεί να χρησιμοποιηθεί παρά μόνον σε μικρά αλφαριθμητικά και θα πρέπει ο προγραμματιστής να μεριμνήσει ώστε να εισάγει τον χαρακτήρα τερματισμού στο τέλος του αλφαριθμητικού.
- Να κάνει χρήση της συνάρτησης scanf() με τον προσδιοριστή %s.
- Να κάνει χρήση της συνάρτησης gets().

# Εισαγωγή αλφαριθμητικού με χρήση του scanf

```
# include <stdio.h>

int main()
{
        char X[100];
        printf("Enter␣X␣\n");
        scanf("%s",X);
        printf("X␣=␣%s\n",X);
        return 0;
}
```

# Εισαγωγή αλφαριθμητικών με χρήση της scanf

- Στο προηγούμενο παράδειγμα, βλέπουμε πως η εισαγωγή αλφαριθμητικών γίνεται με τον προσδιοριστή %s αλλά χωρίς τον χαρακτήρα διεύθυνσης.
- Δηλαδή η σωστή έκφραση είναι scanf("%s",X) και όχι scanf("%s",&X).
- Ωστόσο η scanf() μπορεί να διαβάσει μόνον αλφαριθμητικά μιας λέξής και όχι προτάσεις.
- Αν για παράδειγμα σε αυτό το πρόγραμμα δώσουμε την πρόταση TODAY IS MONDAY, ακόμα και αν αυτή η πρόταση χωράει στις 99 διαθέσιμες θέσεις του πίνακα X, τελικά μόνο η λέξη TODAY θα αποθηκευτεί και θα εμφανιστεί στην οθόνη.
- Το πρόβλημα μπορεί να λυθεί με την χρήση της gets().

# Χρήση της gets() για ανάγνωση αλφαριθμητικών

```
# include <stdio.h>

int main()
{
        char X[100];
        printf("Enter␣x␣\n");
        gets(X);
        printf("X␣=␣%s\n",X);
        return 0;
}
```

# Εύρεση μήκους αλφαριθμητικού

- Σε πολλές περιπτώσεις χρειάζεται ο προγραμματιστής να γνωρίζει πόσα σύμβολα υπάρχουν σε ένα αλφαριθμητικό το οποίο είναι διαφορετικό από την διάσταση του πίνακα.
- Για παράδειγμα ο πίνακας char X[10] και περιεχόμενο "TODAY", έχει διάσταση 10 σαν πίνακας αλλά σαν αλφαριθμητικό έχει μήκος όσο και η λέξη TODAY, δηλαδή 5.
- Στην πράξη η γλώσσα κάνει την ακόλουθη αποθήκευση

| ΣΥΜΒΟΛΟ: | T | O | D | A | Y | \0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ΘΕΣΗ: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

# Η συνάρτηση mystrlen

```
# include <stdio.h>
int mystrlen(char x[]){
        int count=0;
        while(x[count]!='\0') count++;
        return count;
}
int main(){
        char x[100];int len;
        printf("Doste␣mia␣frasi\n");
        gets(x);len=mystrlen(x);
        printf("To␣mikos␣einai␣%d␣\n",len);
        return 0;
}
```

# Η συνάρτηση mystrlen

- Στην συνάρτηση mystrlen(char x[]) ο μετρητής count χρησιμοποιείται για να μετρήσουμε θέσεις μέχρι να βρεθεί ο χαρακτήρας τερματισμού.
- Η συνάρτηση επιστρέφει αυτήν το πλήθος.
- Υπάρχει έτοιμη συνάρτηση στην βιβλιοθήκη string.h που κάνει την ίδια ακριβώς λειτουργία με το όνομα strlen().

# Παράδειγμα χρήσης της συναρτήσεως strlen()

```
# include <stdio.h>
# include <string.h>
int main()
{
        char X[100];
        int len;
        printf("Enter␣X␣\n");
        gets(X);
        printf("X␣=␣%s\n",X);
        len = strlen(X);
        printf("length␣=␣%d␣\n",len);
        return 0;
}
```

# Αντιγραφή αλφαριθμητικού σε άλλο

- Μιας και τα αλφαριθμητικά είναι πίνακες δεν μπορεί να χρησιμοποιηθεί η απευθείς ανάθεση προκειμένου να αναθέσουμε ένα αλφαριθμητικό σε ένα άλλο.
- Ένας τρόπος επίλυσης του θέματος θα ήταν η αντιγραφή ενός αλφαριθμητικού σε άλλο στοιχείο προς στοιχείο.
- Στην βιβλιοθήκη string.h υπάρχει η συνάρτηση strcpy() για αυτόν τον σκοπό.

# Παράδειγμα χρήσης της συναρτήσεως strcpy()

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

int main()
{
        char x1[100],x2[100];
        printf("Doste␣mia␣lexi␣\n");
        scanf("%s",x1);
        strcpy(x2,x1);
        printf("x1:␣%s␣x2:␣%s\n",x1,x2);
        strcpy(x1,"This␣is␣a␣test␣");
        printf("x1:␣%s␣x2:␣%s␣\n",x1,x2);
        return 0;
}
```

# Συνένωση αλφαριθμητικών

- Μια πολύ συχνή περίπτωση χρήσης αλφαριθμητικών είναι η συνένωση αλφαριθμητικών.
- Για παράδειγμα αν X="ΚΑΛΗ" και Υ="ΜΕΡΑ", τότε η συνένωσή τους παράγει το αλφαριθμητικό "ΚΑΛΗΜΕΡΑ".
- Στην βιβλιοθήκη string.h η συνάρτηση που χρησιμοποιείται για αυτό τον λόγο είναι η strcat(x,y).
- Η συνάρτηση αυτή συνενώνει το x με το y και το αποτέλεσμα μπαίνει στο αλφαριθμητικό x.
- Φυσικά ο προγραμματιστής πρέπει να έχει δώσει αρκετό χώρο στον πίνακα x, ώστε να χωρέσουν και τα δύο αλφαριθμητικά σε αυτόν.

# Μια προτεινόμενη εκδοχή συνάρτησης για την συνένωση αλφαριθμητικών

```
void concat(char z[],char x[],char y[])
{
 int count=0;
  int i;
 for(i=0;i<strlen(x);i++) z[count++]=x[i];
 for(i=0;i<strlen(y);i++) z[count++]=y[i];
 z[count]='\0';
}
```

# Σύγκριση αλφαριθμητικών

- Δεν μπορεί να γίνει απευθείας σύγκριση μεταξύ αλφαριθμητικών τιμών, καθώς πρόκειται για πίνακες.
- Για την λεξικογραφική σύγκριση αλφαριθμητικών τιμών η βιβλιοθήκη string.h παρέχει την συνάρτηση strcmp(x,y) η οποία έχει την ακόλουθη επιστροφή τιμών:

$$\text{strcmp(x,y)} = \begin{cases} > 0 & x > y \\ 0 & x = y \\ < 0 & x < y \end{cases}$$

# Επαναληπτική είσοδος δύο ονομάτων μέχρι να γίνει ταύτιση

```
# include <stdio.h>
# include <string.h>
int main()
{
        char X[10],Y[10];
        do{
        printf("Enter x\n");
         gets(X);
         printf("Enter y\n");
         gets(Y);
        }while(strcmp(X,Y)!=0);
        return 0;
}
```

# Η συνάρτηση sscanf()

- Η συνάρτηση sscanf() μπορεί να χρησιμοποιηθεί για ανάγνωση δεδομένων όχι από το πληκτρολόγιο αλλά από ένα αλφαριθμητικό.
- Στην ανάγνωση χρησιμοποιούνται οι ίδιο προσδιοριστές όπως και στην περίπτωση της scanf().

# Ανάγνωση ημερομηνίας από αλφαριθμητικό με χρήση της sscanf()

```c
#include <stdio.h>
#include <string.h>
int main()
{
 char info[100];
 int day,month,year;
 strcpy(info,"Day:10 Month:8 Year:2021");
 sscanf(info,"Day:%d Month:%d Year:%d",
    &day,&month,&year);
 printf("Current date: %d/%d/%d\n",
   day,month,year);
 return 0;
}
```

# Η συνάρτηση sprintf()

- Όπως υπάρχει η συνάρτηση sscanf() με την οποία γίνεται ανάγνωση δεδομένων από αλφαριθμητικό, υπάρχει και η συνάρτηση sprintf() με την οποία η εκτύπωση δεδομένων γίνεται σε αλφαριθμητικό και όχι στην καθιερωμένη έξοδο.
- Χρησιμοποιούνται οι ίδιοι προσδιοριστές όπως και στην printf().
- Σε πολλές περιπτώσεις πληροφορία αποτυπώνεται πρώτα σε αλφαριθμητικό (άρα στην μνήμη) και στην συνέχεια αυτή η πληροφορία γράφεται στην οθόνη ή σε αρχεία για μεγαλύτερη ταχύτητα.

# Εκτύπωση πίνακα σε αλφαριθμητικό

```c
#include <stdio.h>
#include <string.h>
int main()
{
 char info[200];
 int x[5]={10,20,30,40,50};
 int i;
 strcpy(info,"");
 for(i=0;i<5;i++)
  sprintf(info,"%sx[%d]=%d\n",info,i,x[i]);
 printf(info);
  return 0;
}
```

# Διαχωρισμός στοιχείων με την strtok()

- Μια ακόμα συνάρτηση που χρησιμοποιείται ειδικά σε κατασκευή μεταφραστών είναι η συνάρτηση strtok().
- Η συνάρτηση μπορεί να χρησιμοποιηθεί για τον διαχωρισμό ενός αλφαριθμητικού σε τμήματα με βάση κάποιο αλφαριθμητικό διαχωρισμού.
- Την πρώτη φορά που θα κληθεί δέχεται σαν όρισμα το αλφαριθμητικό από το οποίο θα γίνει εξαγωγή της πληροφορίας καθώς και το αλφαριθμητικό διαχωρισμού.
- Σε κάθε επόμενη κλήση στην θέση του πρώτου ορίσματος μπαίνει ο δείκτης NULL.
- Η συνάρτηση θα επιστρέψει NULL όταν τελειώσει με την διαδικασία διαχωρισμού.

# Παράδειγμα χρήσεως της strtok()

```
# include <stdio.h>
# include <string.h>
int main()
{
 char str[80] = "info␣-␣separated␣-␣by␣-␣char";
 char s[2] = "-";
 char *token;
 token = strtok(str, s);
 while( token != NULL )
 {
  printf( "␣%s\n", token );
  token = strtok(NULL, s);
  }
  return 0;
}
```

# Σύνοψη

- Μεταβλητές χαρακτήρα
- Αλφαριθμητικά σε μορφή πίνακα
- Συναρτήσεις αλφαριθμητικών