Αρχές & Γλώσσες Προγραμματισμού 2006-2007

12η Διάλεξη

Π. Σταθοπούλου
pstath@ece.upatras.gr ή
pstath@upatras.gr

Ομάδα Α΄ (Φοιτητές με μονό αριθμό Μητρώου)

- Διδασκαλία : Παρασκευή 11πμ-13μμ ΗΛ7
- Φροντιστήριο : Δευτέρα 11πμ-12πμ ΗΛ4

Προηγούμενη Διάλεξη

Ανάγνωση & εγγραφή Πληροφοριών

Διαχείριση των Ροών για Είσοδο/Έξοδο

Εμβέλεια Μεταβλητών

Σήμερα

Χειρισμός Ακολουθιών Χαρακτήρων

Αρθρωτός Προγραμματισμός

Η πιο κοινή χρήση των μονοδιάστατων πινάκων στην C είναι:

οι αλφαριθμητικοί χαρακτήρες (Strings)

Χειρισμός Ακολουθιών Χαρακτήρων

Σε αντίθεση με άλλες γλώσσες προγραμματισμού η C **δεν διαθέτει** συγκεκριμένο τύπο δεδομένων για τα αλφαριθμητικά.

Στην C οι **αλφαριθμητικοί** εκλαμβάνονται σαν **μονοδιάστατοι πίνακες** χαρακτήρων που τερματίζουν με το **κενό (null)**.

Συνεπάγεται λοιπόν ότι το **μέγεθος των πινάκων** για αλφαριθμητικά πρέπει να είναι κατά **ένα byte μεγαλύτερο** από το μέγεθος των **αλφαριθμητικών.**

Χειρισμός Ακολουθιών Χαρακτήρων

- Ο **χειρισμός** των συμβολοσειρών γίνεται με συναρτήσεις της τυπικής βιβλιοθήκης της C.
- Στις συναρτήσεις των συμβολοσειρών κατά κανόνα μεταβιβάζουμε την **διεύθυνση του 1ου χαρακτήρα τους**
- Η C προσφέρει μια **ποικιλία συναρτήσεων** για διαφόρου είδους χειρισμούς.

Χειρισμός Ακολουθιών Χαρακτήρων

ΚΑΤΗΓΟΡΙΕΣ ΣΥΝΑΡΤΗΣΕΩΝ ΓΙΑ ΤΟΝ ΧΕΙΡΙΣΜΟ ΤΩΝ STRINGS	
Καθορισμός μήκους ακολουθίας χαρακτήρων	string.h (Αρχείο επικεφαλίδας)
Αντιγραφή ακολουθιών χαρακτήρων	
Συνένωση ακολουθιών χαρακτήρων	
Σύγκριση ακολουθιών χαρακτήρων	
Αναζήτηση ακολουθιών χαρακτήρων	
Μετατροπές ακολουθιών χαρακτήρων	stdio.h
Συναρτήσεις που δεν εμπίπτουν σε κατηγορία	string.h
Συναρτήσεις ελέγχου χαρακτήρων	ctype.h

Χειρισμός Ακολουθιών Χαρακτήρων

Συνάρτηση υπολογισμού του **μήκους** μιας ακολουθίας χαρακτήρων

size_t strlen(char *str)

όπου **size_t** επιστρεφόμενη τιμή

unsigned δηλ. μη προσημασμένο ακέραιο

int

***str** όρισμα

είναι ο δείκτης ενός μονοδιάστατου πίνακα

που εμπεριέχει μια ακολουθία χαρακτήρων

Χειρισμός Ακολουθιών Χαρακτήρων

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
  size_t length;
  char buf[80];
   while (1)    {
     puts("\nEnter a line of text, a blank line to exit.");
     gets(buf);
     length = strlen(buf);
     if (length != 0)
        printf("\nThat line is %u characters long.", length);
     else
        break;
   }
  system("PAUSE");
  return 0;
}
```

printf()

scanf()

Πίνακας χαρακτήρων διαστάσεων 80

Strlen()

unsigned

Χειρισμός Ακολουθιών Χαρακτήρων

Οι Συναρτήσεις αντιγραφής ακολουθιών χαρακτήρων είναι δύο:

strcpy() & strncpy()

Η Συνάρτηση **strcpy()** αντιγράφει μια ολόκληρη ακολουθία χαρακτήρων σε μια άλλη θέση μνήμης.

char *strcpy(char *destination, const char *strcpy)

όπου η επιστρεφόμενη τιμή της είναι **δείκτης**

είναι **διεύθυνση** της νέας ακολουθίας χαρακτήρων

Χειρισμός Ακολουθιών Χαρακτήρων

Πίνακας χαρακτήρων διαστάσεων όσο του υπαγορεύει η ανάθεση

```
/* Demonstrates strcpy(). */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

char source[] = "The source string.";
```

Χειρισμός Ακολουθιών Χαρακτήρων

```
int main(int argc, char *argv[])
{
   char dest1[80];
   char *dest2, *dest3;

    printf("\nsource: %s", source );

/* Copy to dest1 is okay because dest1 points to */
/* 80 bytes of allocated space. */

    strcpy(dest1, source);
    printf("\ndest1:  %s", dest1);
```

Πίνακας χαρακτήρων διαστάσεων 80

Δείκτες πινάκων χαρακτήρων

Χειρισμός Ακολουθιών Χαρακτήρων

```
/* To copy to dest2 you must allocate space. */

    dest2 = (char *)malloc(strlen(source) +1);
    strcpy(dest2, source);
    printf("\ndest2:  %s\n", dest2);

/* Copying without allocating destination space is a no-no. */
/* The following could cause serious problems. */

/* strcpy(dest3, source); */

  system("PAUSE");
  return 0;
}
```

Χειρισμός Ακολουθιών Χαρακτήρων

Η Συνάρτηση **strcnpy()** αντιγράφει μια ακολουθία χαρακτήρων σε μια άλλη θέση μνήμης όπως και **strcpy()** αλλά μας επιτρέπει να καθορίσουμε πόσους χαρακτήρες θα αντιγράψουμε.

char *strcnpy(char *destination, const char *source, size_t n)

Όπου

source	δείκτης ακολουθίας προέλευσης
destination	δείκτης ακολουθίας προορισμού
n	το σύνολο των χαρακτήρων αντιγραφής

η επιστρεφόμενη τιμή της είναι **δείκτης = διεύθυνση** της νέας ακολουθίας χαρακτήρων

ıf n> length of source τότε μηδενικά else χάνει '\0'

Χειρισμός Ακολουθιών Χαρακτήρων

```
/* Using the strncpy() function. */
#include <stdio.h>
#include <string.h>
char dest[] = ".........................";
char source[] = "abcdefghijklmnopqrstuvwxyz";
int main(int argc, char *argv[])
{
   size_t n;
   while (1)
   {
      puts("Enter the number of characters to copy (1-26)");
      scanf("%d", &n);
      if (n > 0 && n< 27)
      break;
   }
```

Χειρισμός Ακολουθιών Χαρακτήρων

```
printf("\nBefore strncpy destination = %s", dest);

   strncpy(dest, source, n);

printf("\nAfter strncpy destination = %s\n", dest);

 system("PAUSE");
 return 0;
}
```


Χειρισμός Ακολουθιών Χαρακτήρων

Συνάρτηση αντιγραφής ακολουθιών χαρακτήρων είναι η **strdup()** η οποία είναι παρόμοια με την **strcpy()**

char *strdup(char *source)

διαφέρει από την **strcpy()** διότι κάνει την δική της κατανομή μνήμης καλώντας την συνάρτηση **malloc()**

```
/* The non-ANSI strdup() function. */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

char source[] = "The source string.";

int main(int argc, char *argv[])
{
   char *dest;
```

```
if ( (dest = strdup(source)) == NULL)
  {
   fprintf(stderr, "Error allocating
memory.");
    exit(1);
  }

  printf("The destination = %s\n", dest);

 system("PAUSE");
 return 0;
}
```

Χειρισμός Ακολουθιών Χαρακτήρων

Οι Συναρτήσεις συνένωσης ακολουθιών χαρακτήρων είναι δύο:

strcat() & strncat()

Η Συνάρτηση **strcat()** προσθέτει το αντίγραφο μιας ακολουθίας χαρακτήρων **str2** στο τέλος της **str1** μεταφέροντας τον τερματικό κενό χαρακτήρα στο τέλος.

char *strcat(char *str1, const char *str2)

Όπου

str1 προορισμού **str2** προέλευσης

η επιστρεφόμενη τιμή της είναι **δείκτης**

είναι **διεύθυνση** της νέας ακολουθίας χαρακτήρων

Χειρισμός Ακολουθιών Χαρακτήρων

Η Συνάρτηση **strncat()** εκτελεί την συνένωση δύο ακολουθιών χαρακτήρων όπως και η **strcat()** αλλά καθορίζοντας και τον αριθμό των μεταφερόμενων χαρακτήρων.

char *strncat(char *str1, const char *str2, size_t n)

Όπου

str1 προορισμού **str2** προέλευσης

n το σύνολο των χαρακτήρων συνένωσης

η επιστρεφόμενη τιμή της είναι **δείκτης**

είναι **διεύθυνση** της νέας ακολουθίας χαρακτήρων

Χειρισμός Ακολουθιών Χαρακτήρων

```
/* The strcat() function. */
#include <stdio.h>
#include <string.h>
char str1[27] = "a";
char str2[2];
int main( void )
{
   int n;
   /* Put a null character at the end of str2[]. */

   str2[1] = '\0';
   for (n = 98; n< 123; n++)
   {
      str2[0] = n;
      strcat(str1, str2);
      puts(str1);
   }
   return 0;
}
```


Χειρισμός Ακολουθιών Χαρακτήρων

```
/* The strncat() function. */
#include <stdio.h>
#include <string.h>
char str2[] = "abcdefghijklmnopqrstuvwxyz";

int main( void )
{
   char str1[27];
   int n;

   for (n=1; n< 27; n++)
   {
      strcpy(str1, "");
      strncat(str1, str2, n);
      puts(str1);
   }
   return 0;
}
```

Χειρισμός Ακολουθιών Χαρακτήρων

Οι Συναρτήσεις σύγκρισης ακολουθιών χαρακτήρων είναι δύο:

ANSI C

strcmp() & strncmp()

strcmpl(), stricmp(), strcmpi()

Χειρισμός Ακολουθιών Χαρακτήρων

Η Συνάρτηση **strcmp()** συγκρίνει δύο ακολουθίες **str2, str1** χαρακτήρα προς χαρακτήρα

int strcmp(const char *str1, const char *str2);

Όπου

Σύγκριση πεζών, κεφαλαίων

τα ορίματα **str1** και **str2** είναι **δείκτες**

ενώ η επιστρεφόμενη τιμή τύπου **int**

Τιμή επιστροφής	Σημασία
<0	Η str1 είναι μικρότερη str2
=0	Η str1 είναι μικρότερη str2
>0	Η str1 είναι μικρότερη str2

Χειρισμός Ακολουθιών Χαρακτήρων

Η Συνάρτηση **strncmp()** συγκρίνει ένα καθορισμένο πλήθος χαρακτήρων σε δύο ακολουθίες **str2, str1**

int strncmp(const char *str1, const char *str2, size_t n);

Όπου

τα ορίματα **str1** και **str2** είναι **δείκτες**
n το πλήθος των χαρακτήρων προς σύγκριση

ενώ η επιστρεφόμενη τιμή τύπου **int**

Τιμή επιστροφής	Σημασία
<0	Η str1 είναι μικρότερη str2
=0	Η str1 είναι μικρότερη str2
>0	Η str1 είναι μικρότερη str2

Χειρισμός Ακολουθιών Χαρακτήρων

```
                    /* The strcmp() function. */
#include <stdio.h>
#include <string.h>
int main( void )
{
   char str1[80], str2[80];
   int x;
   while (1)   {                    /* Input two strings. */
      printf("\n\nInput the first string, a blank to exit: ");
      gets(str1);
      if ( strlen(str1) == 0 )
         break;
      printf("\nInput the second string: ");
      gets(str2);
                    /* Compare them and display the result. */
      x = strcmp(str1, str2);
      printf("\nstrcmp(%s,%s) returns %d", str1, str2, x);
   }
    return 0;
}
```

Χειρισμός Ακολουθιών Χαρακτήρων

```
                    /* The strncmp() function. */
#include <stdio.h>
#include <string.h>
char str1[] = "The first string.";
char str2[] = "The second string.";
int main( void )
{
   size_t n, x;
   puts(str1);
   puts(str2);
   while (1)   {
      puts("\n\nEnter number of characters to compare, 0 to exit.");
      scanf("%d", &n);
      if (n <= 0)
         break;
      x = strncmp(str1, str2, n);
      printf("\nComparing %d characters, strncmp() returns %d.", n, x);
   }
   return 0;
}
```


Χειρισμός Ακολουθιών Χαρακτήρων

Οι Συναρτήσεις αναζήτησης ακολουθιών χαρακτήρων είναι:

strchr(), strrchr(), strcspn(), strpbrk(), strstr()

Η Συνάρτηση **strchr()** εντοπίζει την **1η** εμφάνιση συγκεκριμένου χαρακτήρα και το πρότυπό της είναι:

char *strchr(const char *str, int ch);

Όπου **str** η ακολουθία αναζήτησης

ch ο αναζητούμενος χαρακτήρας

η επιστρεφόμενη τιμή **δείκτης = διεύθυνση** του εντοπιζόμενου χαρακτήρα

Χειρισμός Ακολουθιών Χαρακτήρων

```
        /* Searching for a single character with strchr(). */
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
   char *loc, buf[80];
   int ch;          /* Input the string and the character. */
   printf("Enter the string to be searched: ");
   gets(buf);
   printf("Enter the character to search for: ");
   ch = getchar();                    /* Perform the search. */
   loc = strchr(buf, ch);
   if ( loc == NULL )
      printf("The character %c was not found.", ch);
   else
      printf("The character %c was found at position %d.\n",ch, loc-buf+1);
 system("PAUSE");
 return 0;
}
```

Δείκτης

Χειρισμός Ακολουθιών Χαρακτήρων

Η Συνάρτηση **strrchr()** εντοπίζει την **τελευταία** εμφάνιση συγκεκριμένου χαρακτήρα και το πρότυπό της είναι το ίδιο με της **strchr()**

Η Συνάρτηση **strcspn(),** ψάχνει την εμφάνιση οποιουδήποτε χαρακτήρα μιας ακολουθίας χαρακτήρων σε μια δεύτερη ακολουθία. Το πρότυπό της είναι:

size_t strcspn(const char *str1, const char * str2);

Όπου

str1 η ακολουθία αναζήτησης

str2 η αναζητούμενη ακολουθία

Επιστρέφει την μετατόπιση από την αρχή της **str1**

Χειρισμός Ακολουθιών Χαρακτήρων

```
                    /* Searching with strcspn(). */
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
   char  buf1[80], buf2[80];
   size_t loc;          /* Input the strings. */
   printf("Enter the string to be searched: ");
   gets(buf1);
   printf("Enter the string containing target characters: ");
   gets(buf2);          /* Perform the search. */
   loc = strcspn(buf1, buf2);
   if ( loc ==  strlen(buf1) )
      printf("No match was found.");
   else
      printf("The first match was found at position %u.\n", loc);
 system("PAUSE");
 return 0;
}
```


Χειρισμός Ακολουθιών Χαρακτήρων

Οι Συναρτήσεις μετατροπής ακολουθιών χαρακτήρων σε χαρακτήρες είναι δύο:

strupr() & strlwr() — δεν είναι στο πρότυπο ANSI

Οι Συναρτήσεις μετατροπής ακολουθιών χαρακτήρων σε χαρακτήρες είναι δύο:

toupper() & tolower() — είναι στο πρότυπο ANSI

Χειρισμός Ακολουθιών Χαρακτήρων

```
/* The character conversion functions strlwr() and strupr(). */

#include <stdio.h>
#include <string.h>
int main( void )
{
   char buf[80];
   while (1)
   {
      puts("Enter a line of text, a blank to exit.");
       gets(buf);
      if ( strlen(buf) == 0 )
         break;
      puts(strlwr(buf));
      puts(strupr(buf));
   }
   return 0;
}
```


Χειρισμός Ακολουθιών Χαρακτήρων

```
/* The character conversion functions strlwr() and strupr(). */
#include <ctype.h>
#include <stdio.h>
#include <string.h>
int main( void )
{
 char buf[80];
 int  ctr;
 while (1)
   {
    puts("Enter a line of text, a blank to exit.");
    gets(buf);
    if ( strlen(buf) == 0 )
    break;
    for ( ctr = 0; ctr< strlen(buf); ctr++)
       {
        printf("%c", tolower(buf[ctr]));
       }
printf("\n");
```

```
for ( ctr = 0; ctr< strlen(buf); ctr++)
{
        printf("%c", toupper(buf[ctr]));
    }
    printf("\n");
  }
  return 0;
}
```

Χειρισμός Ακολουθιών Χαρακτήρων

Οι Συναρτήσεις μετατροπής ακολουθιών χαρακτήρων είναι δύο:

strrev() & strset(), strnset()

δεν είναι στο πρότυπο ANSI

strrev()	αντιστρέφει τους χαρακτήρες μιας ακολουθίας
strset() **strnset()**	αλλάζει όλους ή ένα καθορισμένο πλήθος χαρακτήρων μιας ακολουθίας

Χειρισμός Ακολουθιών Χαρακτήρων

Οι Συναρτήσεις μετατροπής ακολουθιών χαρακτήρων σε αριθμούς είναι:

Μετατροπή σε ακεραίους	**atoi();**
Μετατροπή σε long	**atol();**
Μετατροπή σε long long	**atoll();**
Μετατροπή σε float	**atof();**

Χειρισμός Ακολουθιών Χαρακτήρων

```
/* Demonstration of atof(). */
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
int main( void )
{
   char buf[80];
   double d;
   while (1)
  {
      printf("\nEnter the string to convert (blank to exit):     ");
      gets(buf);
      if ( strlen(buf) == 0 )
         break;
      d = atof( buf );
      printf("The converted value is %f.", d);
   }
   return 0;
}
```


Χειρισμός Ακολουθιών Χαρακτήρων

Μακροεντολή	Ενέργεια
isalnum()	*TRUE if ch is γράμμα ή ψηφίο*
isalpha()	*TRUE if ch is γράμμα*
isblank()	*TRUE if ch is κενό*
iscntrl()	*TRUE if ch is χαρακτήρας ελέγχου*
isdigit()	*TRUE if ch is ψηφίο*
isgraph()	*TRUE if ch is εκτυπώσιμος χαρακτήρας εκτός κενού*
islower()	*TRUE if ch is πεζό γράμμα*
isprint()	*TRUE if ch is εκτυπώσιμος χαρακτήρας & κενού*
ispunct()	*TRUE if ch is σημείο στίξης.*
isspace()	*TRUE if ch is χαρακτήρας κενού (tab,\n,\f,\0….)*
isupper()	*TRUE if ch is κεφαλαίο γράμμα*
isxdigit()	*TRUE if ch is δεκαεξαδικό ψηφίο*

Αρθρωτός Προγραμματισμός

- Ο **αρθρωτός προγραμματισμός** είναι στενά συνδεδεμένος με τον **δομημένο προγραμματισμό.**
- Ο **δομημένος προγραμματισμός** (structured programming) βασίζεται στην οργάνωση του προγράμματος σε μικρές, αυτόνομα κατανοητές, οντότητες.

Η στρατηγική μιας τέτοιας οργάνωσης επιτρέπει την επίλυση πολύπλοκων προβλημάτων είναι η σωστότερη διαχείριση

Αρθρωτός Προγραμματισμός

- Οι τεχνικές του **αρθρωτού προγραμματισμού** υπαγορεύουν την ανάπτυξη προγραμμάτων με όσο το δυνατόν περισσότερες **συναρτήσεις γενικού σκοπού.**
- Η συγκέντρωση αυτών των συναρτήσεων σε αρχεία ξεχωριστά επιτρέπουν την **χρήση-κλήση** τους από διάφορα αρχεία.
- Έτσι έχουμε:

Προγραμματισμός με πολλαπλά Αρχεία

Πηγαίου κώδικα

- Κάθε πηγαίο αρχείο κώδικα ονομάζεται

λειτουργική μονάδα

Αρθρωτός Προγραμματισμός

- Ένα πρόγραμμα C μπορεί μία να έχει μόνο συνάρτηση **main()**.
- Η λειτουργική μονάδα που περιέχει την main() ονομάζεται ***κύρια λειτουργική μονάδα***.
- Οι άλλες λειτουργικές μονάδες ονομάζονται ***δευτερεύουσες λειτουργικές μονάδες***.

Αρθρωτός Προγραμματισμός

```
#include <stdio.h>
#include <stdlib.h>
long sqr(int x);
int main(int argc, char *argv[])
{
  int x;
  printf("Enter an integer value: ");
  scanf("%d", &x);
  printf("\nThe square of %d is %ld.\n", x, sqr(x));

  system("PAUSE");
  return 0;
}
long sqr(int x)
{
  return ((long)x * x);
}
```

Αρθρωτός Προγραμματισμός

```
#include <stdio.h>
#include <stdlib.h>
long sqr(int x);
int main(int argc, char *argv[])
{
  int x;
  printf("Enter an integer value: ");
  scanf("%d", &x);
  printf("\nThe square of %d is %ld.\n", x, sqr(x));
  system("PAUSE");
  return 0;
}
long sqr(int x)
{
  return ((long)x * x);
}
```


Αρθρωτός Προγραμματισμός

```
/* calc.h: header file for calc.c. */

long sqr(int x);

/* end of calc.h */
```

Αρχείο calc.h

```
#include "calc.h"

long sqr(int x)
{
  return ((long)x * x);
}
```

Αρχείο calc.c

Αρθρωτός Προγραμματισμός

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

#include "calc.h"

int main(int argc, char *argv[])
{
   int x;
   printf("Enter an integer value: ");
   scanf("%d", &x);
   printf("\nThe square of %d is %ld.\n", x, sqr(x));
   system("PAUSE");
   return 0;
}
```


Αρθρωτός Προγραμματισμός

Συστατικά Λειτουργικών Μονάδων

Αρχείο επικεφαλίδας (xxxx.h)

- Πρότυπα για συναρτήσεις στις δευτερεύουσες λειτουργικές μονάδες.
- Οδηγίες **#define**
- Ορισμούς εξωτερικών μεταβλητών ή δομών
- Εντολές προεπεξεργαστή για μεταγλώττιση υπό συνθήκες .

  #if, #elif, #else & #endif, #if defined
- Προκαθορισμένες μακροεντολές

  DATE,_TIME_, _LINE_,_FILE_

Αρθρωτός Προγραμματισμός

Εξωτερικές Μεταβλητές &

Αρθρωτός Προγραμματισμός

Οι μεταβλητές που χρησιμοποιούνται θα πρέπει να είναι ορατές σε όλες τις λειτουργικές μονάδες που τις καλούν.

Κύρια λειτουργική μονάδα

```
int x,y;
main()
{
.....
}
```

Δευτερεύουσα mod1.c

```
extern x,y;
func1()
{
.....
}
```

Δευτερεύουσα mod1.c

```
extern int x;
func4()
{
.....
}
```


Αρθρωτός Προγραμματισμός

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

#define NUMBER 1
#define MAXOP 100
#define STACK_SIZE 100

double stack_buf[STACK_SIZE];
char buffer[MAXOP];

void push(double f);
double pop(void);
int getop(char input[]);
int getch(void);
void ungetch(int c);
int sp=0;
int bufp=0;
int main(int argc, char *argv[])
{
.....
.....
system("PAUSE");
  return 0;
}
```

Calc.h

```
#define NUMBER 1
#define MAXOP 100
#define STACK_SIZE 100

void push(double f);
double pop(void);
int getop(char input[]);
int getch(void);
void ungetch(int c);
```

GetOp.c

```
#include "Calc.h"
{
.....
}
```

GetUnget.c

```
#include "Calc.h"
char buffer[MAXOP];
static int bufp=0;
{
.....
}
```

PushPop.c

```
#include "Calc.h"
double stack_buf[STACK_SIZE];
static int sp=0;
{
.....
}
```