Διαδικαστικός Προγραμματισμός

Βασίλης Παλιουράς
paliuras@ece.upatras.gr

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

int main( void ) {
        char names[] = "Giannis,Kostas Maria:Thanassis";
        char delims[] = ", :";
        char * ch_ptr;

        printf ("names before: %s\n", names);
        ch_ptr = strtok(names, delims) ;
        while (ch_ptr != NULL){
                printf("%s\n", ch_ptr);
                ch_ptr = strtok(NULL, delims);
        }
        printf("names after: %s\n", names);
        { int i;
                for(i=0; i<40; i++) {
                        printf("%c (%d) : ", names[i] , names[i]);
                }
        }

        return 0;
}
```

Άλλη διατύπωση χωρίς strtok στη συνθήκη εισόδου και συνέχειας του βρόχου

Άλλη διατύπωση με μια μόνο κλήση strtok

```c
#include <stdio.h>
#include <string.h>
#define SIZE 40

int main(void ) {
        char names[] = "Giannis,Kostas Maria:Thanassis";
        char delims[] = ", :";
        char * ch_ptr;
        char * arg_ptr;

        printf ("names before: %s\n", names);
        arg_ptr = names;
        while ((ch_ptr = strtok(arg_ptr, delims))!= NULL){
                printf("%s\n", ch_ptr);
                arg_ptr = NULL;
        }
        printf("names after: %s\n", names);
        { int i;
                for (i=0; i<SIZE; i++) {
                        printf("%c (%d) : ", names[i] , names[i]);
                }
        }

        return 0;
}
```

- Η μεταβλητή arg_ptr αρχικοποιείται ώστε να δείχνει στο names
- Κατά την εκτέλεση του βρόχου, η agr_ptr τίθεται πάντα σε NULL ώστε να συνεχίσει η strtok να επεξεργάζεται το ίδιο αλφαριθμητικό.
- Ο βρόχος ολοκληρώνεται όταν η strtok επιστρέψει NULL

```c
#include <stdio.h>
#include <string.h>
#define ROWS 2
#define COLS 3

int counter (int reset);

int main(void)
{

    printf("called %d times\n", counter(0));

    printf("called %d times\n", counter(0));

    printf("reset %d times\n", counter(1));

    printf("called %d times after reset\n", counter(0));

    printf("called %d times after reset\n", counter(0));

    printf("called %d times after reset\n", counter(0));

    return 0;
}
```

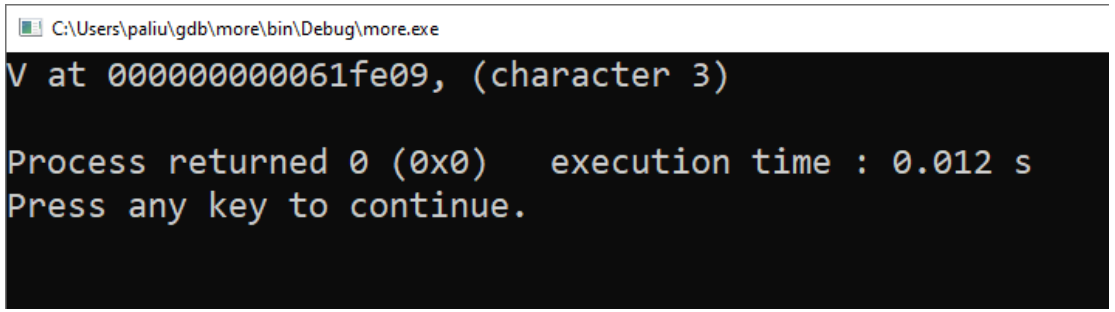
```c
int counter (int reset) {

    static int count = 0;

    if (reset)
        count = 0 ;
    else
        count ++ ;

    return count;
}
```

- Η `strtok` θυμάται την προηγούμενη κατάστασή της
- **`static`** qualifier

char * strchr(const char *, int)

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

int main(void) {
        char word[10] = "abcVcba";
        char ch = 'V';
        char * ch_ptr ;

        ch_ptr = strchr(word, ch);
        if (ch_ptr != NULL) {
                printf("%c at %p, (character %d)\n",
                        *ch_ptr , ch_ptr, (int) (ch_ptr-word));
                }
        else {
                printf("not existing");
             }
        return 0;
}
```

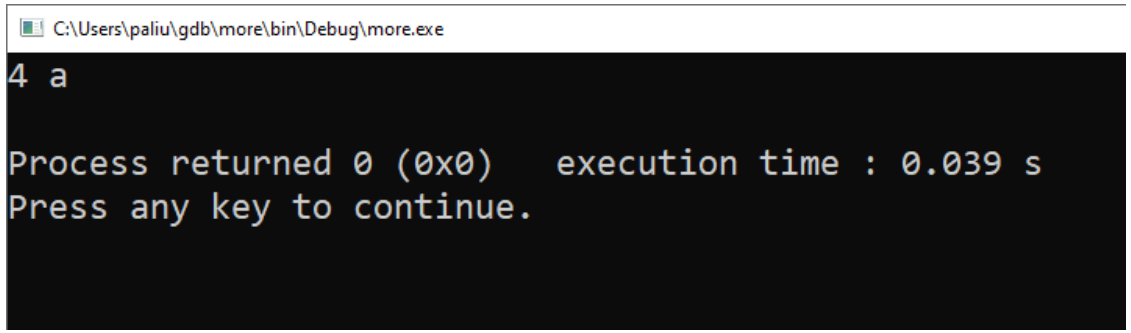
- Αναζητά χαρακτήρα σε string
- Επιστρέφει τη διεύθυνση της πρώτης εμφάνισης, αν βρεθεί ο χαρακτήρας
- NULL, αν δεν βρεθεί



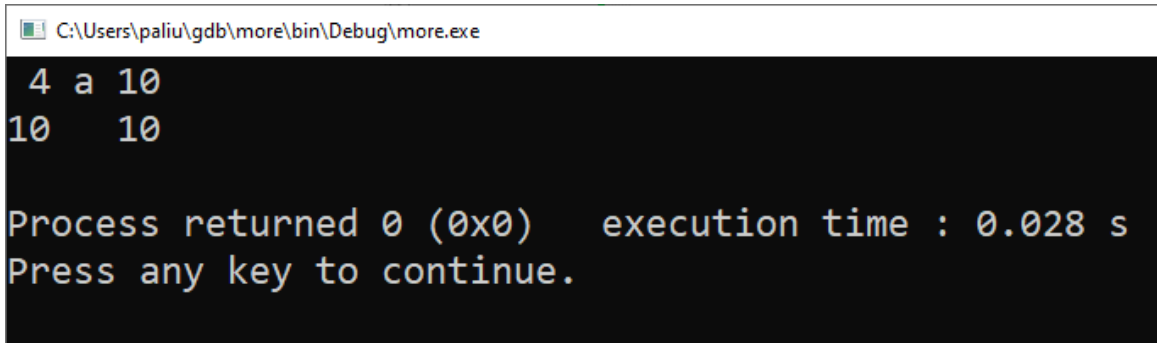
`size_t strcspn ( const char * str1, const char * str2 );`

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

int main(void) {
        char str1[ ] = "testabcabc";
        int i;

        i = strcspn(str1, "abc");

        printf("%d %c\n",i, str1[i]);

        return 0;
}
```

- Αναζητά οποιονδήποτε από τους χαρακτήρες του `str2`, στο `str1`
- Επιστρέφει τη θέση της πρώτης εμφάνισης, αν βρεθεί κάποιος χαρακτήρας
- `strlen(str1)`, αν δεν βρεθεί
- **Προσοχή** θέση 0,1,2,…, όχι διεύθυνση!



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

int teststrcspn(char *, char *);

int main(void) {
	char str1[ ] = "testabcabc";

	teststrcspn(str1,"abc");
	teststrcspn(str1,"fgh");

	return 0;
}

int teststrcspn(char *s, char *chars) {
	int i;

	i = strcspn(s, chars);

	printf("%2d %c %d\n",i, s[i], (int) strlen(s));

	return 0;
}
```



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

int main(void) {
 char str[ ] = "testabcabc";
 char chars[] = "ac";
 char * str_i;
 size_t i;
 int loc = 0;

 str_i = str;                    /* addresses only ! */

 for (i=strcspn(str_i,chars); i < strlen(str_i);
                 i = strcspn(str_i, chars)) {
        loc = loc + i ;
        printf("%2d %c %d\n", loc, str_i[i], (int) strlen(str_i));
        str_i = str_i + i + 1;
        loc ++;
        }

 return 0;
}
```

```
C:\Users\paliu\gdb\more\bin\Debug\more.exe
 4 a 10
 6 c 5
 7 a 3
 9 c 2

Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.
```

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

int main(void) {
        char str[ ] = "testabcabc";
        char chars[] = "ac";
        char * str_i;
        size_t i;
        int loc = 0;

        str_i = str;   /* holds addresses only! */

        for (; (i = strcspn(str_i,chars)) < strlen(str_i); ){
                loc += i ;
                printf("%2d %c %d\n", loc, str_i[i], (int) strlen(str_i));
                str_i += i + 1;
                loc ++;
                }

        return 0;
}
```

```
C:\Users\paliu\gdb\more\bin\Debug\more.exe
 4 a 10
 6 c 5
 7 a 3
 9 c 2

Process returned 0 (0x0)   execution time : 0.055 s
Press any key to continue.
```

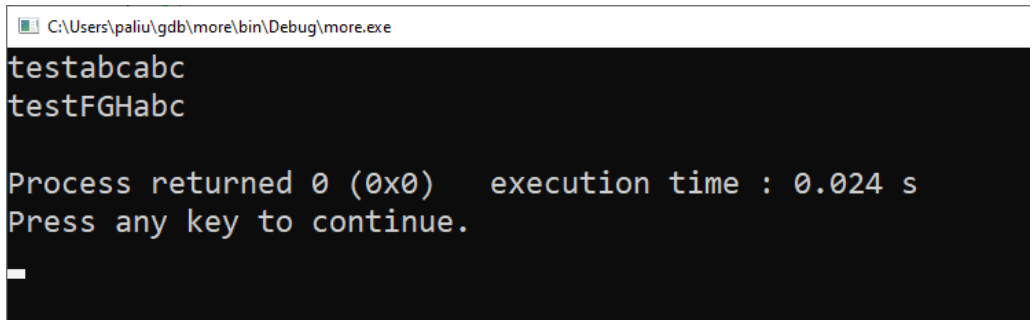
char * strpbrk (const char *, const char *);

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

int main(void) {
 char str[ ] = "testabcabc";
 char chars[] = "ac";
 char * str_i;

 str_i = strpbrk(str, chars);
 while (str_i!=NULL) {
  printf("%2d %c %d\n", (int) (str_i-str), *str_i, (int) strlen(str_i));
  str_i = strpbrk(str_i+1, chars);
 }

 return 0;
}
```

char * strstr(const char *str1, const char *str2);

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

int main(void) {
        char str[ ] = "testabcabc";
        char * ch_ptr ;

        printf("%s\n", str);

        ch_ptr = strstr(str, "abc");
        strncpy(ch_ptr, "FGH", 3);

        printf("%s\n", str);

        return 0;
}
```

- Αναζητά αλφαριθμητικό που αρχίζει στη διεύθυνση `str2` μέσα σε αλφαριθμητικό που αρχίζει στη διεύθυνση `str1`
- Επιστρέφει τη διεύθυνση της πρώτης εμφάνισης, αν υπάρχει
- NULL, αν δεν υπάρχει



Εδώ η `strncpy` αντιγράφει ***ακριβώς*** τρεις χαρακτήρες στη θέση της πρώτης εμφάνισης του `"abc"` στο `str`

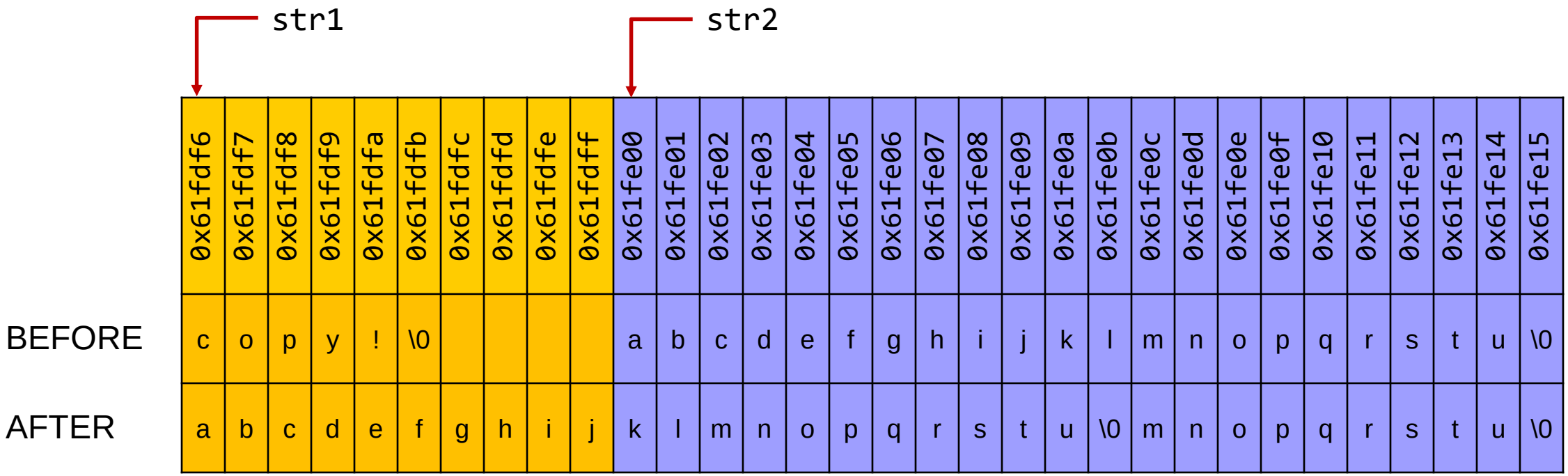
Buffer overflow!

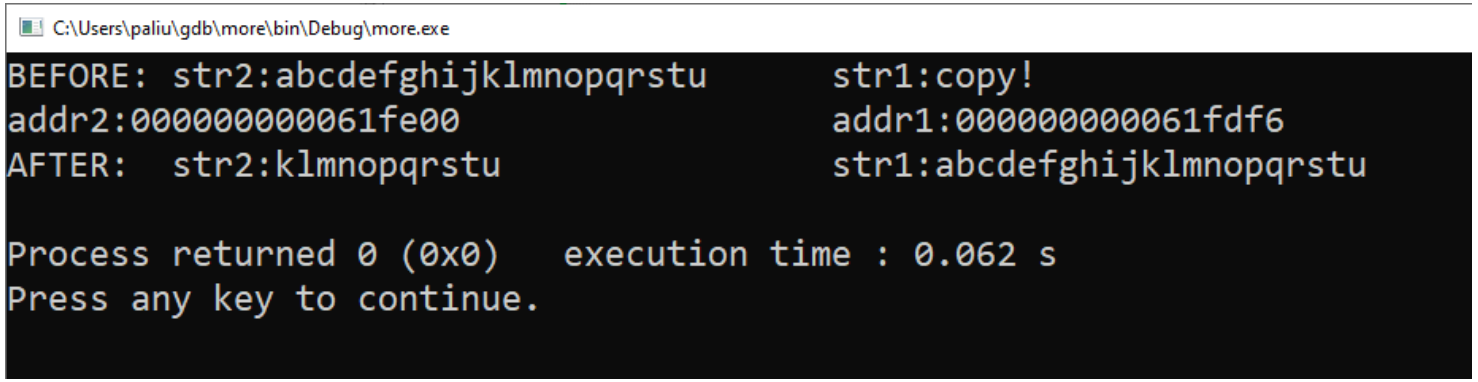
```
C:\Users\paliu\gdb\more\bin\Debug\more.exe
BEFORE: str2:abcdefghijklmnopqrstu      str1:copy!
addr2:000000000061fe00                  addr1:000000000061fdf6
AFTER:  str2:klmnopqrstu                str1:abcdefghijklmnopqrstu

Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```

```c
#include <stdio.h>
#include <string.h>
#define N 64

int main(void) {

  char str2[] = "abcdefghijklmnopqrstu" ;
  char str1[10] = "copy!";

  printf("BEFORE: str2:%s\tstr1:%s\naddr2:%p\t\t\taddr1:%p\n",
                str2, str1, str2, str1);

  strcpy(str1, str2);

  printf("AFTER:  str2:%s\t\tstr1:%s\n", str2, str1);

  return 0;
}
```

- Η `strcpy` αντιγράφει διαδοχικούς χαρακτήρες από το `str2` στο `str1`, μέχρις ότου βρει το `\0`
- Θα πρέπει ο χώρος στο `str1` να είναι επαρκής για να αποφύγουμε buffer overflow

Τοποθέτηση στη μνήμη, buffer overflow





char * strncpy(char * d, const char * s, size_t num);

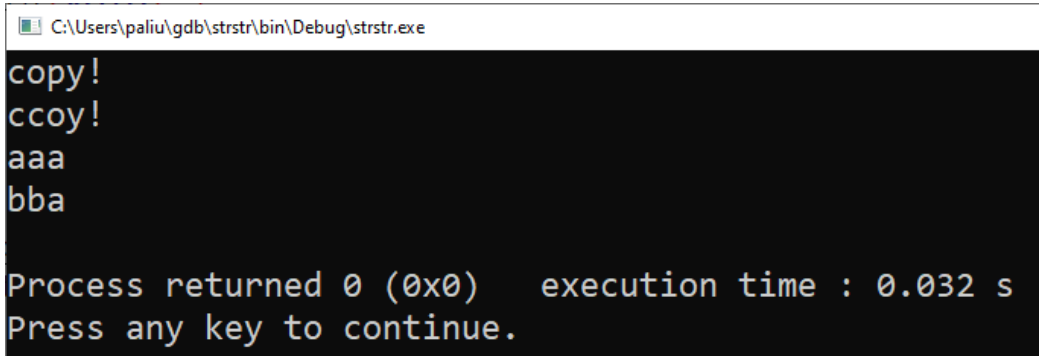
```c
#include <stdio.h>
#include <string.h>
#define N 64

int main(void) {
  char str[N] ;
  char word[] = "copy!";
  char word2[] = "aaa";

  strcpy(str, word);
  printf("%s\n", str);

  strncpy(str+1, word, 2);
  printf("%s\n", str);

  strncpy(str, word2, 4);
  printf("%s\n", str);

  strncpy(str, "bb", 2);
  printf("%s\n", str);

  return 0;
}
```

Αντιγράφει το word στο str (πέντε χαρακτήρες και \0)

Αντιγράφει δύο χαρακτήρες από το word ξεκινώντας από τη δεύτερη θέση του str, δηλ. την str+1 Δεν τοποθετείται \0

4=3+1: Αντιγράφει τρεις χαρακτήρες του word2 στο str και τοποθετεί ένα \0.

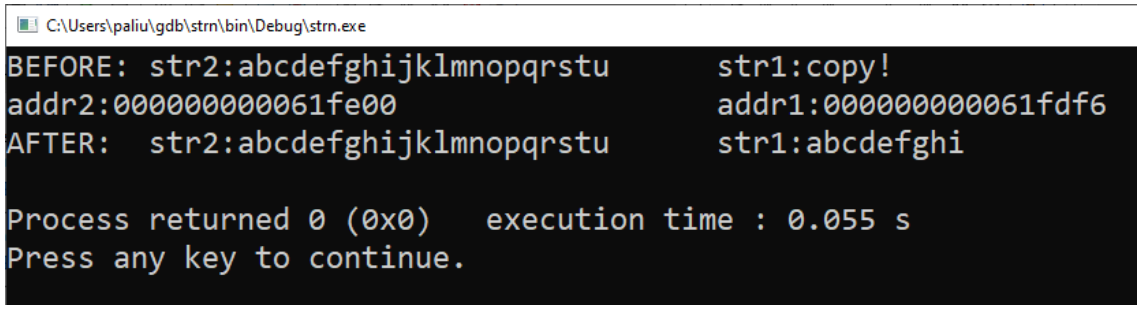
Αντιγράφει δύο χαρακτήρες του "bb" στο str Δεν τοποθετεί \0.



- Αντιγράφει num χαρακτήρες ξεκινώντας από τη διεύθυνση s σε αλφαριθμητικό που αρχίζει στη διεύθυνση d
- Αν στο s υπάρχουν λιγότεροι από num χαρακτήρες, τους αντιγράφει και συμπληρώνει με \0 μέχρι το num
- Συνεπώς αν στο s υπάρχουν περισσότεροι από num, δεν τοποθετείται \0.

Αποφυγή buffer overflow με `strncpy`

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

int main(void) {

 char str2[] = "abcdefghijklmnopqrstu" ;
 char str1[10] = "copy!";

 printf("BEFORE: str2:%s\tstr1:%s\naddr2:%p\t\t\taddr1:%p\n",
                      str2, str1, str2, str1);

 strncpy(str1, str2, sizeof str1);
 str1[9] = '\0';

 printf("AFTER:  str2:%s\tstr1:%s\n",
        str2, str1);

 return 0;
}
```

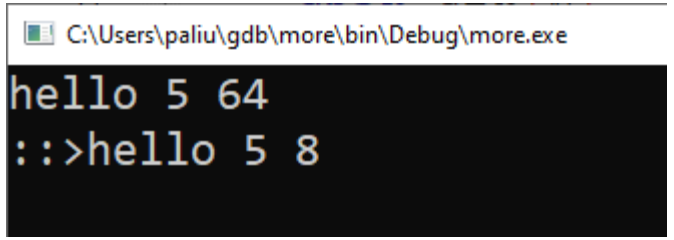


- Με το **`sizeof`** λαμβάνουμε το μέγεθος του πίνακα `str1`, *επειδή αυτός δημιουργείται στο ίδιο block κώδικα.*
- Εξασφαλίζεται ότι δεν θα γίνει overflow (γιατί;)
- Τοποθετείται χαρακτήρας τερματισμού στην τελευταία θέση του `str1`

`sizeof` και πίνακες, σύνηθες «λάθος»

```c
#include <stdio.h>
#include <string.h>
#define N 64

void sizeofinfunction(char str[N]);

int main(void) {

        char str[N] = "hello" ;

        printf("%s %d %d\n", str,
                (int) strlen(str), (int) sizeof str);

        sizeofinfunction(str);

    return 0;
}

void sizeofinfunction(char str[N]) {

        printf("::>%s %d %d\n",
                str, (int) strlen(str), (int) sizeof str);

    return ;
}
```

Στο ίδιο block, επιστρέφει το μέγεθος του πίνακα



Σε συνάρτηση, για πίνακα που δημιουργείται αλλού, επιστρέφει το μέγεθος του δείκτη

*warning: 'sizeof' on array function parameter 'str' will return size of 'char *' [-Wsizeof-array-argument]|*

Μερικές συναρτήσεις για διαχείριση συνεκτικών περιοχών μνήμης

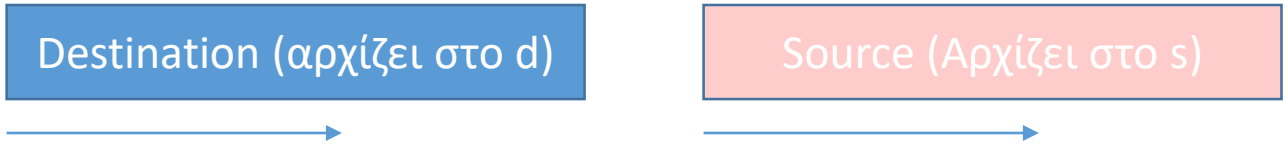
- **void** * memcpy (**void** * destination, **const void** * source, size_t num);
  Αντιγράφει από source σε destination num bytes

- **void** * memmove (**void** * destination, **const void** * source, size_t num);
  Αντιγράφει από source σε destination num bytes και διαχειρίζεται επικαλύψεις

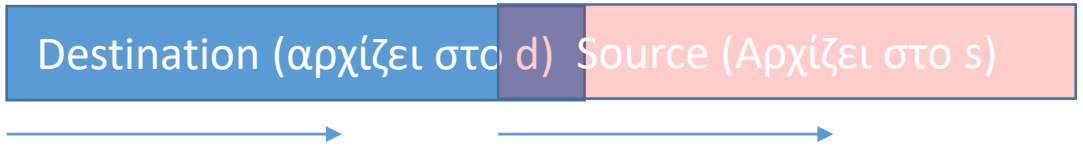
- **void** * memset (**void** * ptr, **int** value, size_t num);
  Θέτει στην τιμή value, την οποία ερμηνεύει ως **unsigned char**, καθένα από τα num το πλήθος διαδοχικά bytes που ξεκινούν από τη διεύθυνση ptr

memmove *vs.* memcpy

memcpy: δεν κάνει έλεγχο σχετικής θέσης των περιοχών source και destination



(χωρίς επικάλυψη) δεν έχει θέμα



(με επικάλυψη, destination πριν από source) δεν έχει θέμα

(με επικάλυψη, destination μετά από source) θα χάσει τα τελευταία του source

memmove: ελέγχει αν d < s, και αν όχι αντιγράφει από το τέλος προς την αρχή

Εδώ δεν έχει θέμα (ούτως ή άλλως)

Ούτε εδώ έχει θέμα (σε αντίθεση με την memcpy)

```c
#include <stdio.h>
#include <string.h>
#define N 4

int main(void)
{
    int data[]={1,2,3,4};
    int copy[N] ;
    int i;

    memmove(copy, data, sizeof copy);

    for (i=0;i<N;i++) printf("%d", copy[i]);

    return 0;
}
```

```
1234
Process ret
```

```c
#include <stdio.h>
#include <string.h>
#define ROWS 2
#define COLS 3

int main(void)
{
    int data[][COLS]={{1,2,3},{4,5,6}};
    int copy[ROWS][COLS] ;
    int i,j;

    memmove(copy, data, sizeof copy);

    for (i=0;i<ROWS;i++) {
        for (j=0;j<COLS;j++)
            printf("%d ", copy[i][j]);
        printf("\n");
    }

    return 0;
}
```

```
1 2 3
4 5 6
```

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

int main(void)
{
    int data[]={1,2,3,4};
    int i;

    memset(data, 0, 4*sizeof (int));

    for (i=0;i<4;i++)
       printf("%d",data[i]);

    return 0;
}
```

```
0000
```

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

int main(void)
{
    int data[]={1,2,3,4};
    int i;

    memset(data, 0, 4*sizeof data[0]);

    for (i=0;i<4;i++)
       printf("%d",data[i]);

    return 0;
}
```

```
0000
```

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

int main(void)
{
    int data[]={1,2,3,4};
    int i;

    memset(data, 0, sizeof data);

    for (i=0;i<4;i++)
        printf("%d", data[i]);

    return 0;
}
```

```
0000
```

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

int main(void)
{
    int data[]={1,2,3,4};
    int i;

    memset(data, 0, 4);

    for (i=0;i<4;i++)
        printf("%d", data[i]);

    return 0;
}
```

```
0234
```

Μηδενίζει μόνο το πρώτο στοιχείο (γιατί;)

warning: 'memset' used with length equal to number of elements without multiplication by element size [-Wmemset-elt-size]|

```c
#include <stdio.h>
#include <string.h>
#define N 64

int main(void) {
  char str[N] ;

  memset(str, 'A', sizeof str);
  str[N-1] = '\0';
  printf("%s\n", str);

  memset(str, 'B', -1 + sizeof str);
  printf("%s\n", str);

  memset(str, 'C', (sizeof str)/2);
  printf("%s\n", str);

  memset(str+(sizeof str)/2, '-', -1+(sizeof str)/2);
  printf("%s\n", str);

  return 0;
}
```

Εδώ, ως byte value χρησιμοποιείται ένας ASCII κώδικας σε κάθε κλήση της memset