ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

6η Εβδομάδα: Χρήση Αλφαριθμητικών & Συναρτήσεις Βιβλιοθήκης String.h (Β) - Εμβέλεια Μεταβλητών

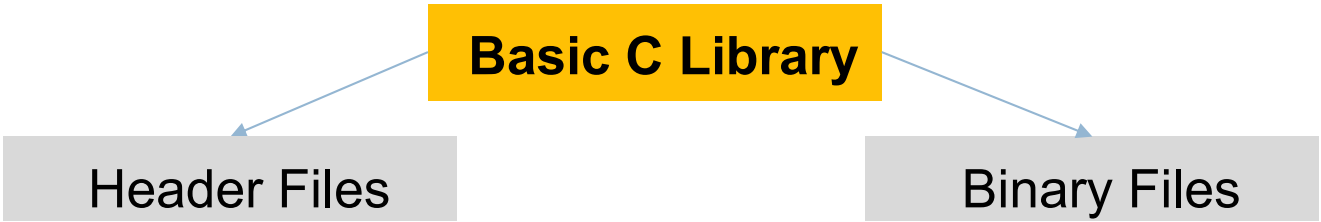
Αναφορές

Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου.**

Βιβλιοθήκες στη C

Βιβλιοθήκη είναι μια συλλογή μεταγλωττισμένων μονάδων που μπορούν να συνδεθούν (linked) στα προγράμματά μας μέσω των διεπαφών (header files - interfaces) που παρέχουν.

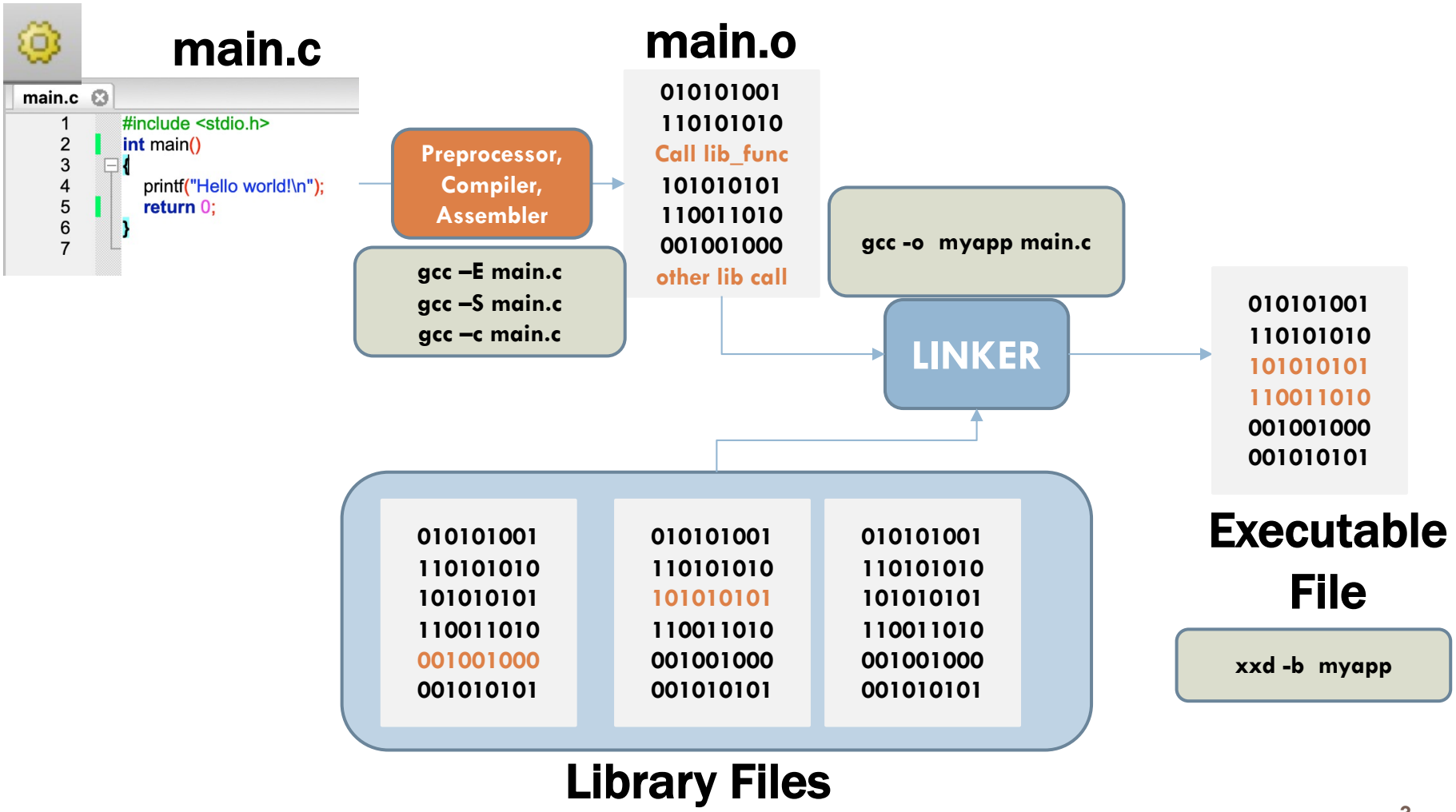
Με άλλα λόγια κάθε βιβλιοθήκη **αποτελείτε από** δυαδικά αρχεία σε object code (*.o) που περιέχουν υλοποιήσεις όλων των συναρτήσεων που έχουν δηλωθεί στα αρχεία επικεφαλίδων .h



- `<stdio.h>` **είναι η διεπαφή (interface) που περιέχει συναρτήσεις I/O.**

Πηγαίος κώδικας της βασικής βιβλιοθήκης: http://ftp.gnu.org/gnu/glibc/

Βιβλιοθήκες στη C



Η Βιβλιοθήκη string.h

- Το αρχείο επικεφαλίδα (header file), string.h παρέχει συναρτήσεις για χειρισμό strings
  - `char * strncpy(char *dest, const char *source, size_t n);`
  - `char * strtok (char * str, const char * delim);`
  - `char* strncat(char* destination, const char* source, size_t num);`
  - `char * strchr(const char *, int);`
  - `char * strpbrk (const char *, const char * );`
  - `size_t strcspn ( const char * str1, const char * str2 );`
  - `int strncmp(const char *str1, const char *str2, size_t n)`

Συναρτήσεις `strcpy` και `strncpy`


```
int main(){
    char source[] = "KALHMERA SAS" ;
    char dest[30] = {};
    strcpy(dest, source);
    printf("source:%s   dest:%s\n", source, dest);
    return 0;
}
```

```
source:KALHMERA SAS    dest:KALHMERA SAS
```

```
int main(){
    char source[] = "KALHMERA SAS" ;
    char dest[30] = {};
    strncpy(dest, source, strlen(source));
    printf("source:%s   dest:%s\n", source, dest);
    return 0;
}
```

```
source:KALHMERA SAS    dest:KALHMERA SAS
```

Συνάρτηση strncpy

- Τι και αν θέλω να αντιγράψω μόνο τη λέξη "**KALHMERA**"
  - **char * strncpy(char *dest, const char *source, size_t n);** αντιγράφει n χαρακτήρες

```
int main(){
    char source[] = "KALHMERA SAS" ;
    char dest[30] = {};
    strncpy(dest, source, 8);
    dest[8]='\0';
    printf("source:%s   dest:%s\n", source, dest);
    return 0;
}
```

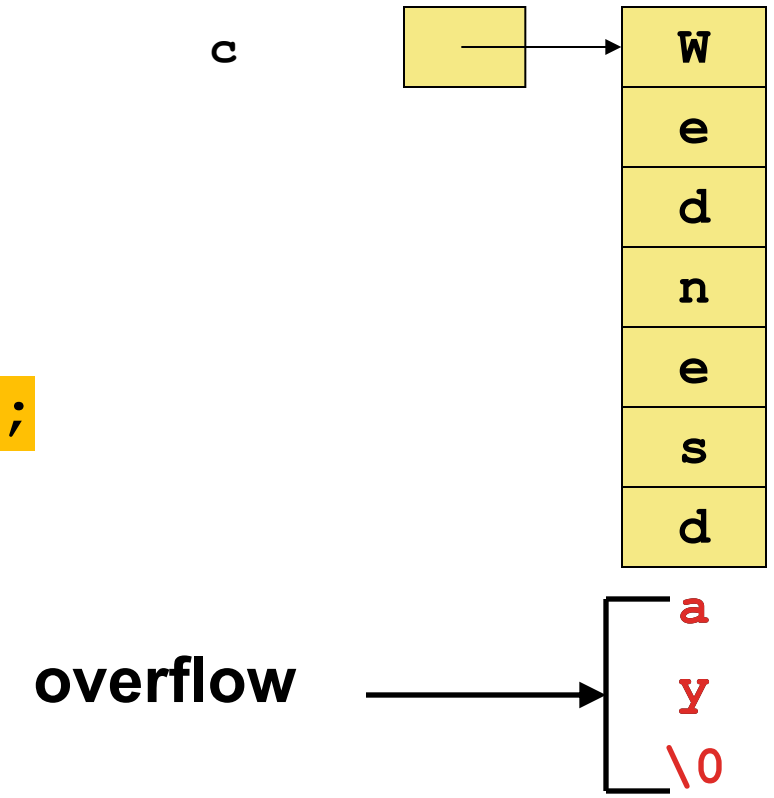
Συνάρτηση strncpy

- Τι και αν θέλω να αντιγράψω μόνο τη λέξη ”**SAS**”

```
int main(){
    char source[] = "KALHMERA SAS" ;
    char dest[30] = {};
    strncpy(dest, source+9, 4);
    printf("source:%s   dest:%s\n", source, dest);
    return 0;
}
```

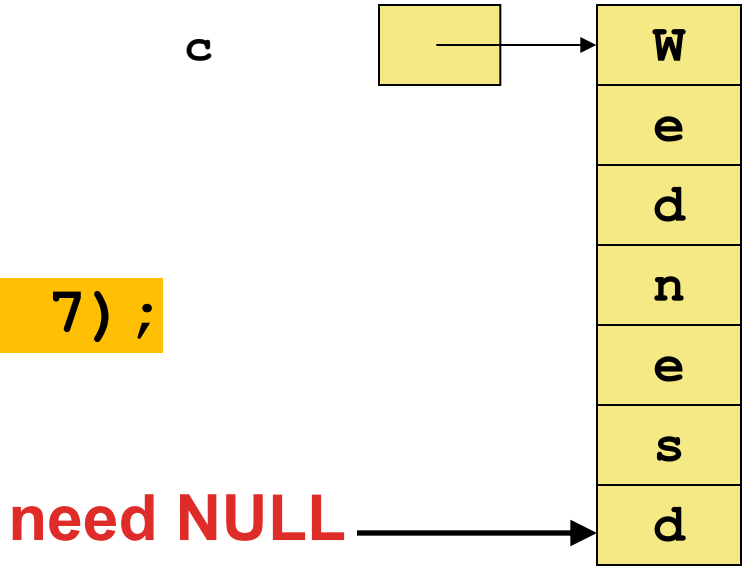
Προσοχή!

```
#include <string.h>
…
char c[7];
strcpy(c, “Wednesday”);
```



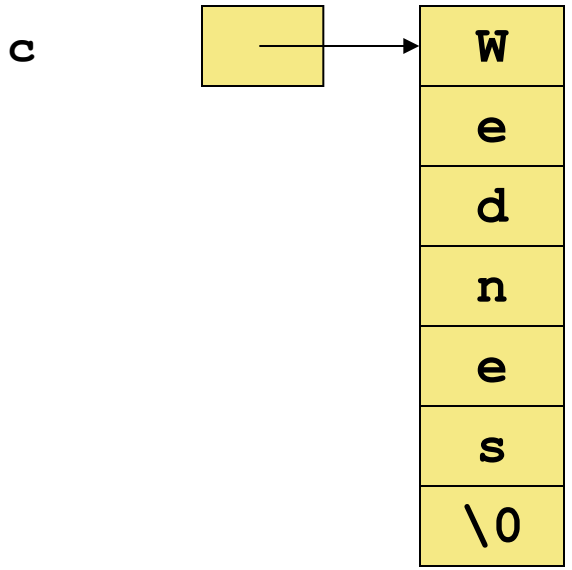
Καλύτερα με τη strncpy

```
#include <string.h>
…
char c[7];
strncpy(c, "Wednesday", 7);
```



Καλύτερα με τη strncpy

```
#include <string.h>
…
char c[7];
strncpy(c, “Wednesday”, 6);
c[6] = ‘\0’;
```



```
abcdefghijklmaaaaaaaaa copy! 0x7ffeea7e5a70 0x7ffeea7e5a66

Process returned -1 (0xFFFFFFFF)   execution time : 0.183 s
Press ENTER to continue.
```

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

int main(void) {

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


        printf("%s %s %p %p\n", str2, str1, str2, str1);

        strcpy(str1, str2);

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

        return 0;
}
```

```
abcdefghijklmaaaaaaaaa copy! 0x7ffeef2b6a70 0x7ffeef2b6a66
abcdefghijklmaaaaaaaaa abcdefghi
```

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

int main(void) {

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


        printf("%s %s %p %p\n", str2, str1, str2, str1);

        strncpy(str1, str2, sizeof str1);
        str1[9] = '\0';
        printf("%s %s\n", str2, str1);

        return 0;
}
```

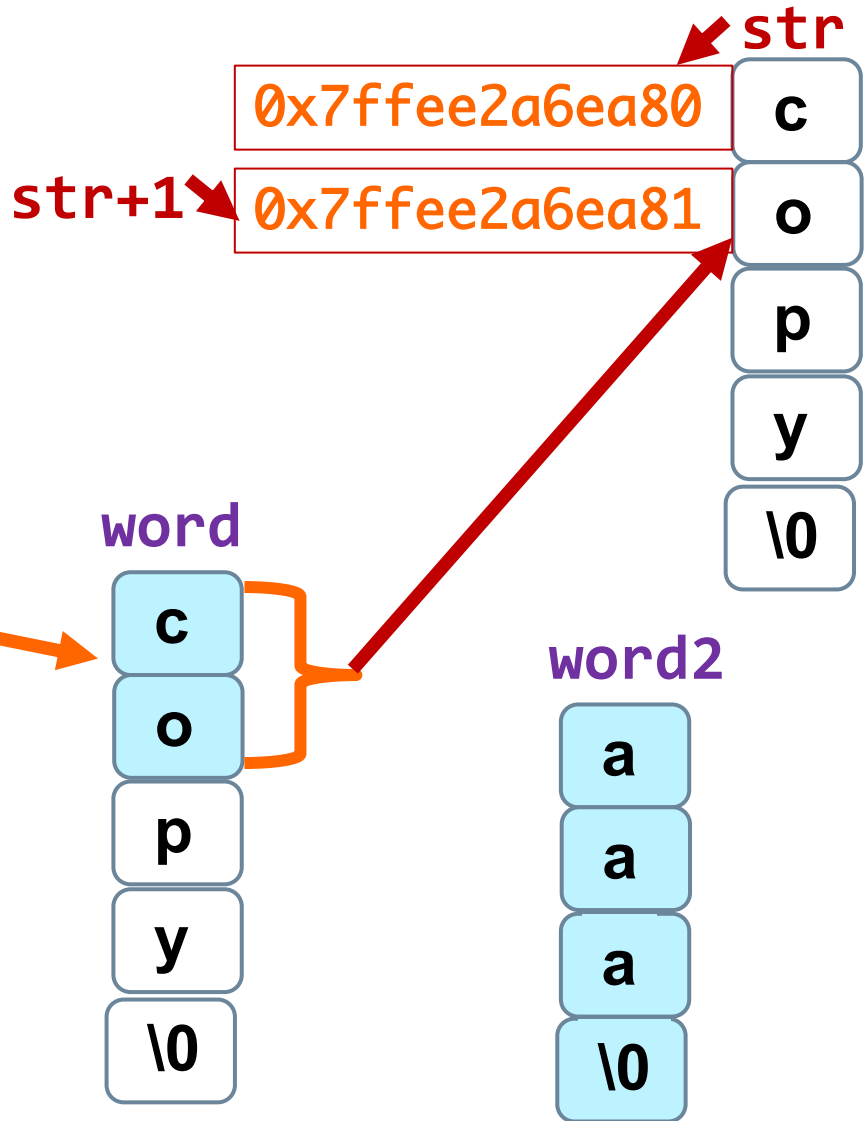
```
abcdefghijklmaaaaaaaaa copy! 0x7ffeee8d6a70 0x7ffeee8d6a66
abcdefghijklmaaaaaaaaa abcde
```

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

int main(void) {

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


        printf("%s %s %p %p\n", str2, str1, str2, str1);

        strncpy(str1, str2, strlen(str1));
        str1[9] = '\0';
        printf("%s %s\n", str2, str1);

        return 0;
}
```

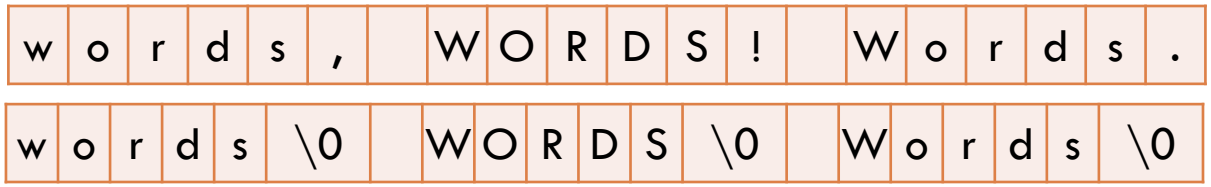
Ακόμη ένα παράδειγμα της strncpy

```
copy!
ccoy!
aaa
```

```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);

        return 0;
}
```



char * strtok (char * str, const char * delim)



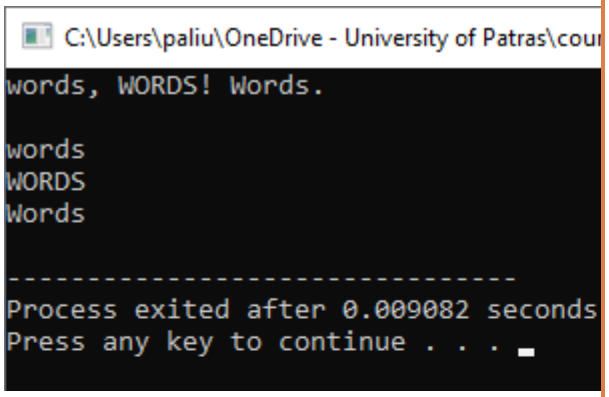
Αυτή η συνάρτηση σπάει τη συμβολοσειρά **str** σε μια σειρά από συμβολοσειρές με βάση του οριοθέτη **delim**.

```
#include <stdio.h>
#include <string.h>
#define SIZE 20
int main(void) {
        int i;
        char str[] = "words, WORDS! Words.";

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

        word = str;
        while ( (word = strtok(word,",! .",))!=NULL) {
                printf("%s\n", word);
                word = NULL;
        }

        return 0;
}
```

```
C:\Users\paliu\OneDrive - University of Patras\courses\PP
words, WORDS! Words.

words
WORDS
Words

--------------------------------
Process exited after 0.009082 seconds with
Press any key to continue . . .
```

```c
#include <stdio.h>
#include <string.h>
#define SIZE 20
int main(void) {
    int i;
    char str[] = "words, WORDS! Words.";

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

    word = str;
    while ( (word = strtok(word,",! .")))!=NULL) {
          printf("%s\n", word);
          word = NULL;
          }

return 0;
}
```

Η πρώτη κλήση στο **strtok** πρέπει να περάσει τη συμβολοσειρά που θέλουμε να διακριτικοποιήσουμε, και **οι επόμενες κλήσεις πρέπει να καθορίσουν** το **NULL** ως πρώτο όρισμα, το οποίο λέει στη συνάρτηση να συνεχίσει να κάνει διακριτικοποίηση στη συμβολοσειρά που μεταβιβάσαμε πρώτα.




```c
#include <stdio.h>
#include <string.h>
#define SIZE 20
int main(void) {
   int i;
   char str[] = "words, WORDS! Words.";

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

   for (word = str; word = strtok(word,",! ."); word = NULL)
      printf("%s\n", word);

   return 0;
}
```

Ακόμη ένα παράδειγμα της strtok

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

        return 0;
}
```

Ακόμη ένα παράδειγμα της strtok

```
#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;
}
```

```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;

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

        return 0;
}
```

char * strcat(char *, const char *)

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

int main(void) {
 char word[10] = "more ";
 char base[100] = "";

 int i;
 for (i=0; i<5; i++) {
    strcat(base, word);
    printf("%s %d\n", base, strlen(base));
 }
 return 0;
}
```

```
more  5
more more  10
more more more  15
more more more more  20
more more more more more  25
```

char* strncat(char* destination, const char* source, size_t num);


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

int main(void) {
 char word[10] = "more ";
 char base[100] = "";

 int i;
 for (i=0; i<5; i++) {
    strncat(base, word,3);
    printf("%s %d\n", base, strlen(base));
 }
 return 0;
}
```

```
mor 3
mormor 6
mormormor 9
mormormormor 12
mormormormormor 15
```

char * strchr(const char *str, int c)

```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, ch_ptr-word);
                }
        else {
                printf("not existing");
              }
        return 0;
}
```

Αυτή η συνάρτηση αναζητά την πρώτη εμφάνιση του χαρακτήρα **c** στη συμβολοσειρά **str**.

```
V at 0x7ffee10b1ae1, (character 3)
```

char * strpbrk (**const char** * s1, **const char** * s2);

```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", str_i-str, *str_i, strlen(str_i));
  str_i = strpbrk(str_i+1, chars);
 }

 return 0;
}
```

Αυτή η συνάρτηση βρίσκει τον πρώτο χαρακτήρα στη συμβολοσειρά **s1** που ταιριάζει με οποιονδήποτε χαρακτήρα που καθορίζεται στο **s2** (*Αποκλείει τους μηδενικούς χαρακτήρες τερματισμού*).

```
4 a 6
6 c 4
7 a 3
9 c 1
```

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

```
#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;
}
```

Αυτή η συνάρτηση βρίσκει την πρώτη εμφάνιση της συμβολοσειράς **s** στην συμβολοσειρά **str**. Οι χαρακτήρες τερματισμού '\0' δεν συγκρίνονται.

```
testabcabc
testFGHabc
```

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

Αυτή η συνάρτηση **υπολογίζει το μήκος του αριθμού των χαρακτήρων πριν από την 1η εμφάνιση χαρακτήρα που υπάρχει και στις δύο συμβολοσειρές.**

```
#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;
}
```

```
4 a
```

```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], strlen(s));

	return 0;
}
```

```
 4 a 10
10  10
```

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

- *strcmp* or *strncmp*
  - Compares two strings (good for sorting)

```
strcmp("Saturday", "Sunday");  //answer is
  -1

strncmp("way", "wash", 2);     //answer is
  0
```

- *strlen*
  - Returns the number of characters in "*Saturday*"

```
strlen("Saturday")                  //answer is
```

Διαχείριση Χαρακτήρων <ctype.h>

- *isalpha* είναι ο χαρακτήρας γράμμα του αλφαβήτου;
- *isdigit* είναι ο χαρακτήρας αριθμός;

Και άλλες όπως:

- *islower, isupper*
- *ispunct*
- *isspace*

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/

ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

6η Εβδομάδα: Χρήση Αλφαριθμητικών & Συναρτήσεις Βιβλιοθήκης String.h (Β) - Εμβέλεια Μεταβλητών

Αναφορές

Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου.**

Μηχανισμοί κλήσης συναρτήσεων

- Μηχανισμός κλήσης κατ' αξία (call by value)
  - δημιουργούνται τοπικά αντίγραφα των τιμών των ορισμάτων
  - Αν τα ορίσματα έχουν μήκος πολλών bytes, επιβαρύνεται η εκτέλεση.
  - Η συνάρτηση δεν μπορεί να αλλάξει τις τιμές ορισμάτων στο σημείο κλήσης.
- Μηχανισμός κλήσης με αναφορά (call by reference)
  - δεν δημιουργούνται τοπικά αντίγραφα τιμών,
    - μόνο των διευθύνσεων !!!
    - η συνάρτηση ενημερώνεται για τη θέση μνήμης στην οποία είναι αποθηκευμένο ένα όρισμα.
    - μπορεί να είναι ταχύτερο
  - είναι δυνατόν να αλλάξουν οι τιμές ορισμάτων στο σημείο κλήσης.
    - χρειάζεται προσοχή, μπορεί να γίνει εκ παραδρομής.
    - Είναι ένας τρόπος να *επιστρέψω περισσότερες από μία τιμές*
    - *Για τη C, καταχρηστικά λέγεται έτσι : επί της ουσίας είναι κατ' αξία πέρασμα δεικτών.*

Δείκτες σε ακέραιο

```
#include <stdio.h>
#include <stdlib.h>
void sumdiff(int, int, int *, int *);

int main(int argc, char *argv[])
{
int a = 3, b =4 ;
int sum;
int diff;

  sumdiff(a,b, &sum, &diff);

  printf("%d %d\n", sum, diff);

  system("PAUSE");
  return 0;
}

void sumdiff(int a, int b, int *s, int *d) {
      *s = a + b;
      *d = a - b;
      a++;
      b++;
      printf("%d %d\n",a,b);
      }
```

Διευθύνσεις των μεταβλητών που θα αλλάξουν τιμές

Η s αρχικοποιείται στην τιμή &sum

Αλλάζουν τιμές τα *περιεχόμενα* των θέσεων με διευθύνσεις s, d.

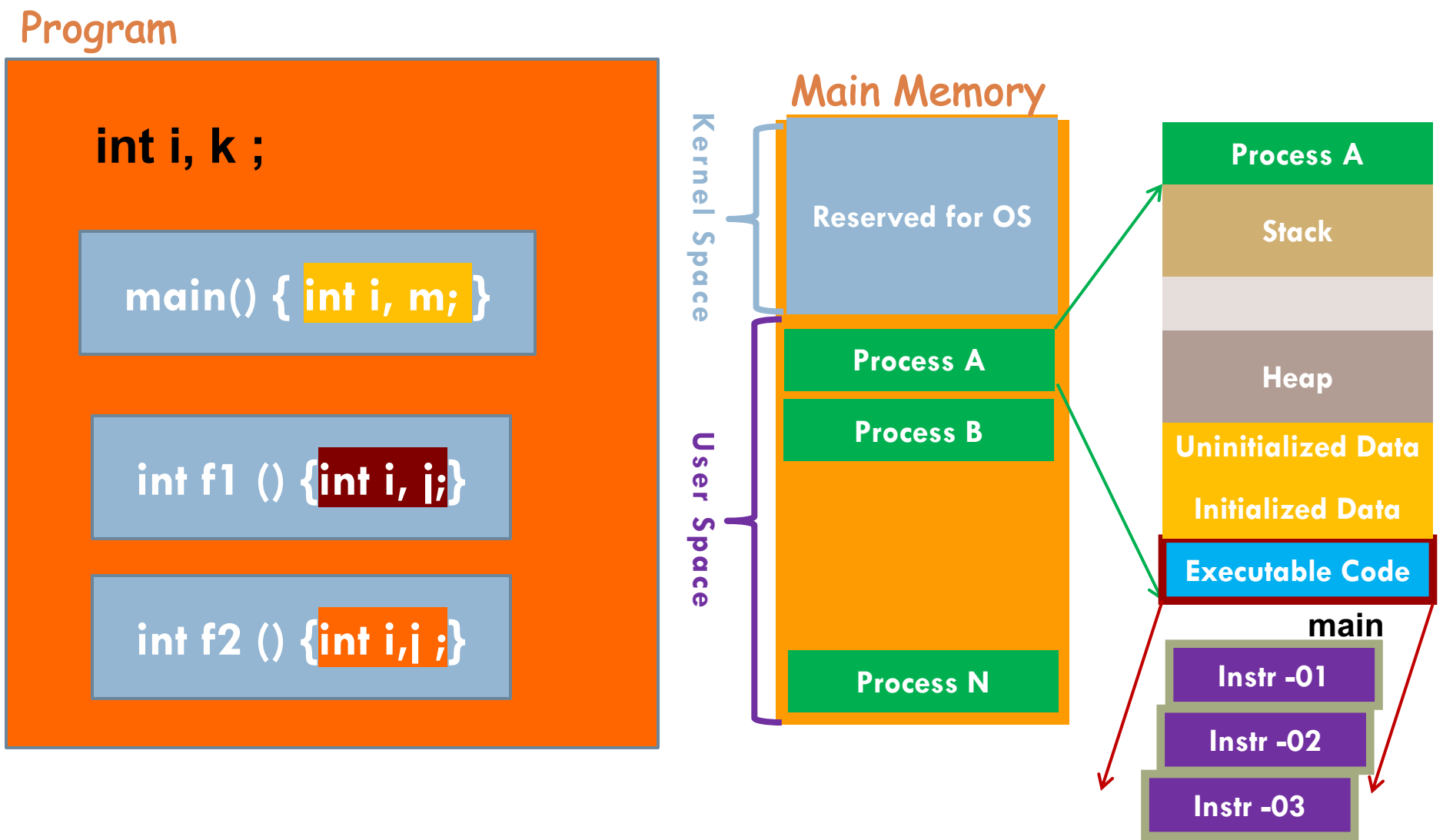
Στη sumdiff( ) ορίζονται *νέες μεταβλητές* a,b

Οι a,b της main() είναι διαφορετικές μεταβλητές σε σχέση με τις a,b της sumdiff

Συναρτήσεις: Εμβέλεια Μεταβλητών (Scope)

- Κάθε συνάρτηση είναι **σαν ξεχωριστό υπο-πρόγραμμα**
- Κάθε υπο-πρόγραμμα έχει το **δικό του χώρο στη μνήμη**
- **Κάθε μεταβλητή** που **δηλώνεται** μέσα σε μια συνάρτηση, **είναι ορατή μόνο σε αυτή τη συνάρτηση** και οι **τιμές** που **αποθηκεύουν δεν μπορούν** να **διαβαστούν από καμία άλλη συνάρτηση (local variables)**.
  - **local variables** (τοπικές):
    - Δηλώνονται στην αρχή μέσα στο σώμα της συνάρτησης
    - Χρήση μόνο μέσα στην εμβέλεια (σώμα) της συνάρτησης
  - **global variables** (σφαιρικές/καθολικές):
    - Δηλώνονται έξω από συναρτήσεις
    - Ορατές από όλες τις συναρτήσεις. Χρήση οπουδήποτε μέσα στο πρόγραμμα.

Εμβέλεια ονομάτων



Διάρκεια ζωής μεταβλητής

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

void function (void);
void function1 (void);
int main () {

        function1();
        function ( ) ;
        function1 ( ) ;
        function ( ) ;

        return 0;
}

void function1() {
    int j = 0;
    j++;
    printf("function1:%p ", &function1);
    printf(", j: %d at %p\n", j, &j);

}
void function ( ) {
    int i ;
    i++ ;
    printf("function:%p ", &function);
    printf(", i: %d at %p\n", i, &i);

}
```

Η τοπική μεταβλητή

/δεν αρχικοποιείται!

- **i, j, είναι τοπικές μεταβλητές**
- **Διάρκεια ζωής: όσο εκτελείται η συνάρτηση**
- **Η θέση μνήμης επαναχρησιμοποιείται**

```
function1:0x10c457c10, j: 1 at 0x7ffee37abacc
function:0x10c457c50, i: 2 at 0x7ffee37abacc
function1:0x10c457c10, j: 1 at 0x7ffee37abacc
function:0x10c457c50, i: 2 at 0x7ffee37abacc
```

Διάρκεια ζωής μεταβλητής

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

void function (void);
void function1 (void);
int main () {

        function1();
        function ( ) ;
        function1 ( ) ;
        function ( ) ;

        return 0;
}

void function1() {
    int j = 1;
    j++;
    printf("function1:%p ", &function1);
    printf(", j: %d at %p\n", j, &j);

}
void function ( ) {
    int i ;
    i++ ;
    printf("function:%p ", &function);
    printf(", i: %d at %p\n", i, &i);

}
```

Η τοπική μεταβλητή

/δεν αρχικοποιείται!

- **i, j, είναι τοπικές μεταβλητές**
- **Διάρκεια ζωής: όσο εκτελείται η συνάρτηση**
- **Η θέση μνήμης επαναχρησιμοποιείται**

```
function1:0x106143c10, j: 2 at 0x7ffee9abfacc
function:0x106143c50, i: 3 at 0x7ffee9abfacc
function1:0x106143c10, j: 2 at 0x7ffee9abfacc
function:0x106143c50, i: 3 at 0x7ffee9abfacc
```

Διάρκεια μεταβλητής (2)

```
void function1 (void);
void function2 (void);
void function(void);
main ( ) {
        function();
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
}
void function() {
        int k = 0;
}
void function1( ) {
        int i;
        i++ ;
        printf("f1:%d\n", i);
}
void function2( ) {
        int j;
        j++ ;
        printf("f2:%d\n", j);
}
```

Αν η κλήση της function1( ) γίνεται εναλλάξ με την function2( ), η συμπεριφορά αλλάζει...

Εμφανίζονται εξαρτήσεις (είναι δυνατόν να ...)

```
f1:1
f2:2
f1:3
f2:4
f1:5
f2:6
```

Διάρκεια μεταβλητής (2)

```
void function1 (void);
void function2 (void);
void function(void);
main ( ) {
        function();
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
}
void function() {
        int k = 0;
        printf("f0:%d %p \n",k,&k);
        function1();

}
void function1( ) {
        int i;
        i++ ;
        printf("f1:%d  %p\n", i,&i);
}
void function2( ) {
        int j;
        j++ ;
        printf("f2:%d\n", j);
}
```

Αν η κλήση της function1( ) γίνεται μέσα στην function( ), η συμπεριφορά αλλάζει...

```
f0:0 0x7ffeecf76adc
f1:801001936  0x7ffeecf76abc
f1:1  0x7ffeecf76adc
f2:2
f1:3  0x7ffeecf76adc
f2:4
f1:5  0x7ffeecf76adc
f2:6
```

```
void function1 (void);
void function2 (void);

main ( ) {
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
}
void function1 () {
        int i = 0;
        i++ ;
        printf("f1:%d\n", i);
}
void function2 () {
        int j = 0;
        j++ ;
        printf("f2:%d\n", j);
}
```

Λύση;

Αρχικοποιώντας τις μεταβλητές

κάθε φορά που καλείται η συνάρτηση,

οι τοπικές μεταβλητές μηδενίζονται κάθε

φορά που καλείται η συνάρτηση.

Η λέξη κλειδί **static** σε δηλώσεις τοπικών μεταβλητών

```
void function1 (void);
void function2 (void);

main ( ) {
        function1( )  ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
}
void function1 ( ) {
        int z=0;
        static int i = 0;
        i++ ;
        printf("f1:%d\n", i, &i , &z);

}
void function2 ( ) {
        int z=0;
        static int j = 0;
        j++ ;
        printf("f2:%d %p %p \n", j, &j, &z);
}
```

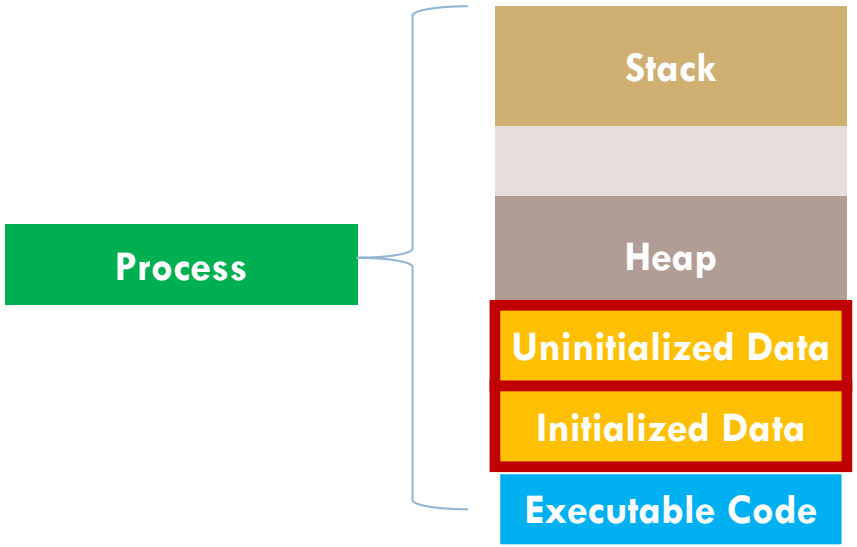
static: ο χώρος μνήμης της μεταβλητής δεν αποδεσμεύεται όταν ολοκληρωθεί μια εκτέλεση της συνάρτησης.

```
f1:1 0x108d5a020  0x7ffee6eadadc
f2:1 0x108d5a024 0x7ffee6eadadc
f1:2 0x108d5a020  0x7ffee6eadadc
f2:2 0x108d5a024 0x7ffee6eadadc
f1:3 0x108d5a020  0x7ffee6eadadc
f2:3 0x108d5a024 0x7ffee6eadadc
```

Η λέξη κλειδί **static** σε δηλώσεις τοπικών μεταβλητών

```
void function1 (void);
void function2 (void);

main ( ) {
        function1( )  ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
}
void function1 ( ) {
        static int i = 0;
        i++ ;
        printf("f1:%d\n", i);
}
void function2 ( ) {
        static int j = 0;
        j++ ;
        printf("f2:%d\n", j);
}
```

static: ο χώρος μνήμης της μεταβλητής δεν αποδεσμεύεται όταν ολοκληρωθεί μια εκτέλεση της συνάρτησης.



Άλλη χρήση της **static**: Ορισμός εμβέλειας αρχείου

- αρχείο πηγαίου κώδικα: code1.c

```
int func (int);
void report(void);

main () {
int i ;

for ( i=0; i< 5; i++)
    printf("%d\n", func(i));

report();
}
```

- αρχείο πηγαίου κώδικα: code2.c

```
static int sum = 0;

int func(int k) {
  int i;
  for (i=0;i<5; i++)
    sum += k ;  /* sum = sum + k */
  return 2 * k;
}
void report() {
    printf("%d\n", sum);
}
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/