ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

8η Εβδομάδα: Δυναμική Διαχείριση Μνήμης & Δομές Δεδομένων

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.**

Στατική διαχείριση μνήμης στη C

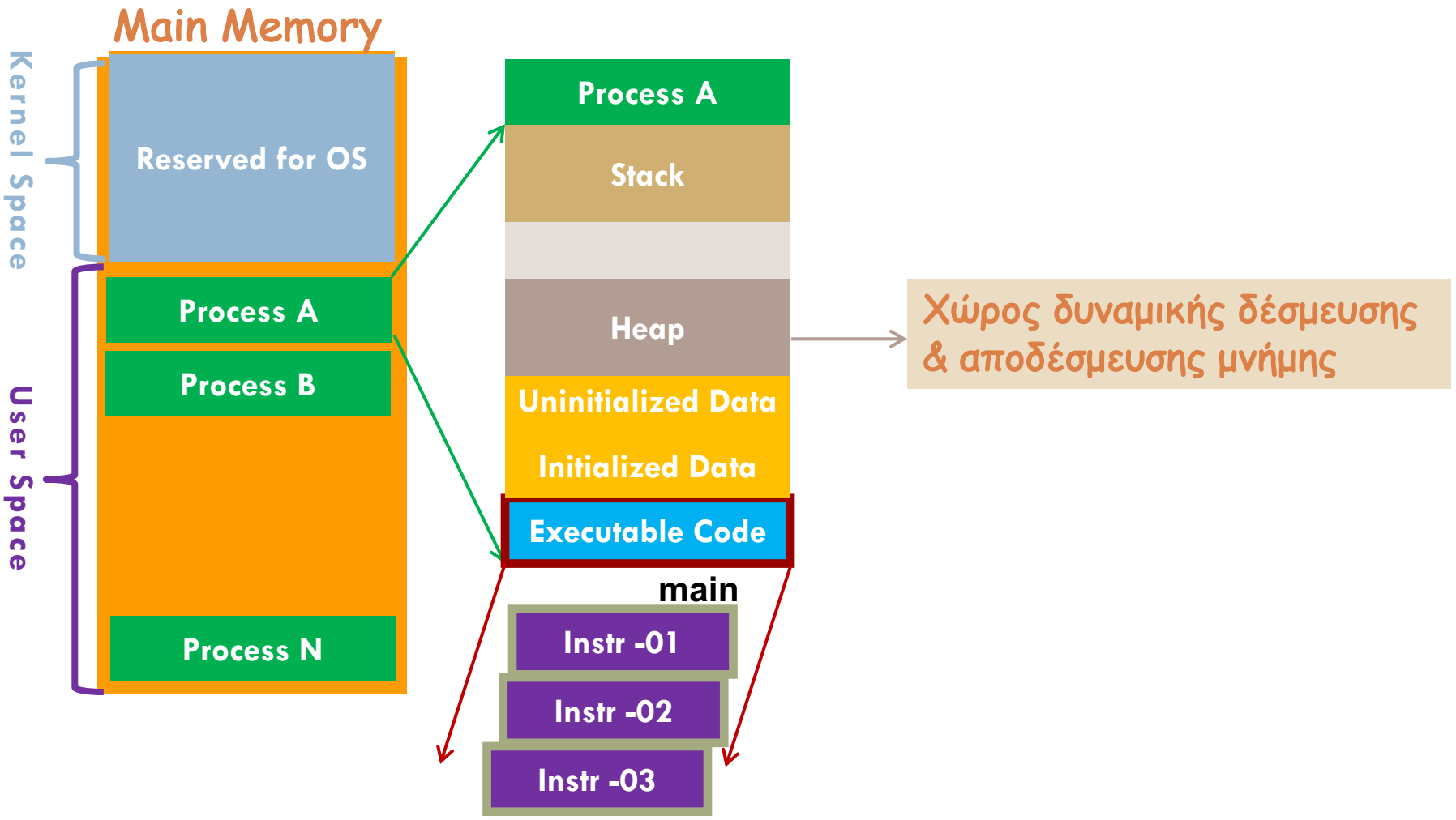
- Με τη δήλωση απλών μεταβλητών ο χώρος μνήμης που τους αποδίδεται παραμένει δεσμευμένος μέχρι το τέλος του προγράμματος ή μέχρι το τέλος του μπλοκ εντολών στο οποίο δηλώθηκαν.

  π.χ. int x;

  int y[10];

- Η παραπάνω τεχνική ονομάζεται στατική διαχείριση της μνήμης και απαιτεί από τον προγραμματιστή να γνωρίζει εκ των προτέρων το μέγεθος της μνήμης που θα χρειαστεί το πρόγραμμά του

Διαχείριση μνήμης – Σωρός (Heap)



Δυναμική διαχείριση μνήμης στη C

- Δέσμευση μνήμης:
  - **void** *malloc(size_t size);
    - Επιστρέφει δείκτη σε εξασφαλισμένη περιοχή μεγέθους size bytes ή NULL αν δεν υπάρχει τέτοια.
  - **void** *calloc(size_t nitems,size_t size);
- Απελευθέρωση μνήμης:
  - **void** free(**void** *pointer);

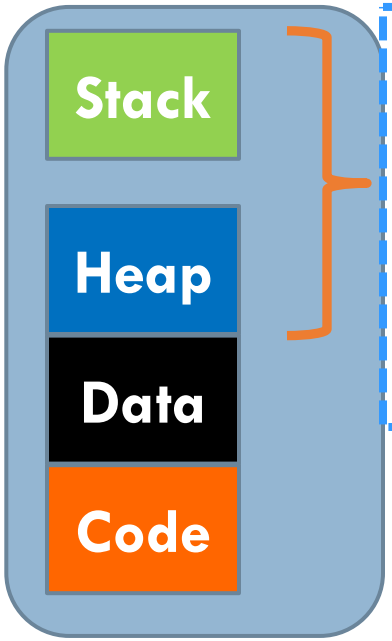
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int *dynamic_int;
    dynamic_int=(int *) malloc(sizeof(int));
    if (dynamic_int == NULL) {
        printf("error");
        return EXIT_FAILURE;
    }
    else{
        scanf("%d,",dynamic_int);
        printf("%d",*dynamic_int);
        free(dynamic_int);
    }
    return EXIT_SUCCESS;
}
```

Πώς δουλεύει ο μηχανισμός;

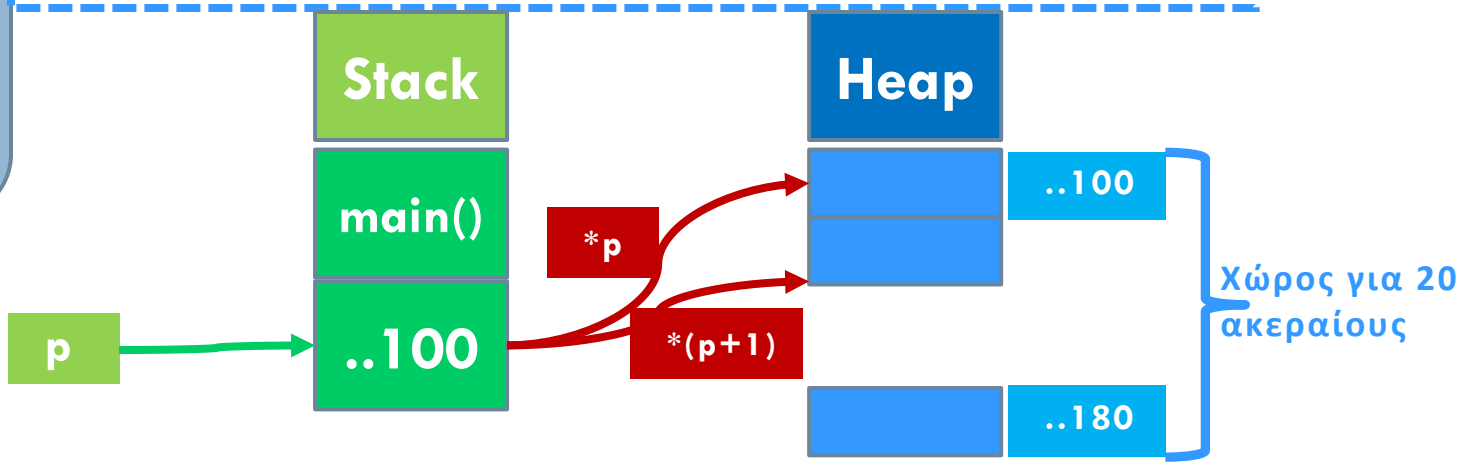
- Χρησιμοποιεί
  - Δεδομένα στο heap
  - Λεπτομερή διαχείριση ανά block
    - Διεύθυνση αρχής
    - Μέγεθος
- Μοιράζεται πληροφορία μεταξύ διαφορετικών συναρτήσεων
  - `malloc( ) , free( )`
  - Πώς γίνεται αυτό;

Δυναμική δέσμευση μνήμης

Application Memory

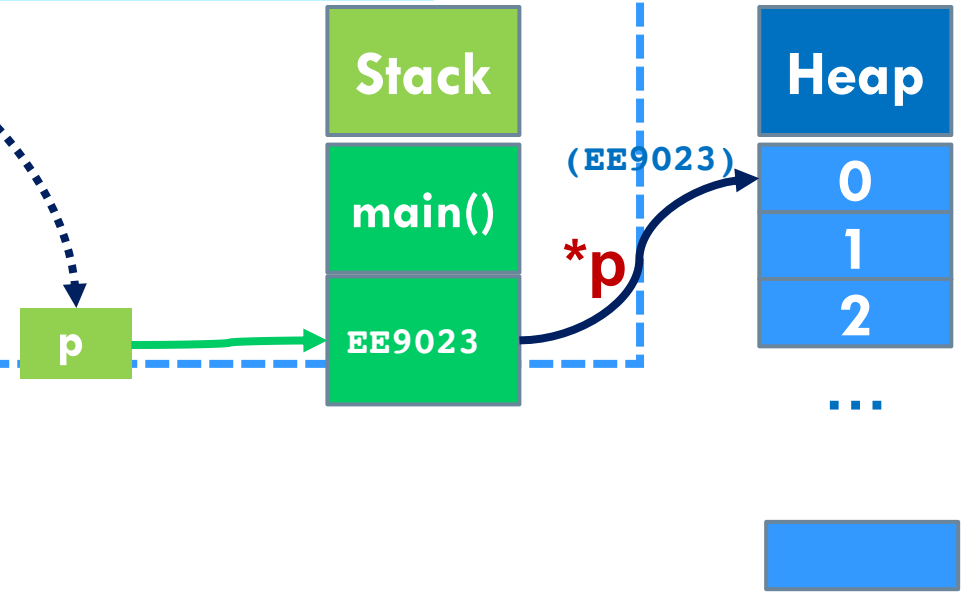



```
#include <stdio.h>
#include <stdlib.h>

int main(){
    int *p;
    p=(int *) malloc(20 * sizeof(int));
}
```



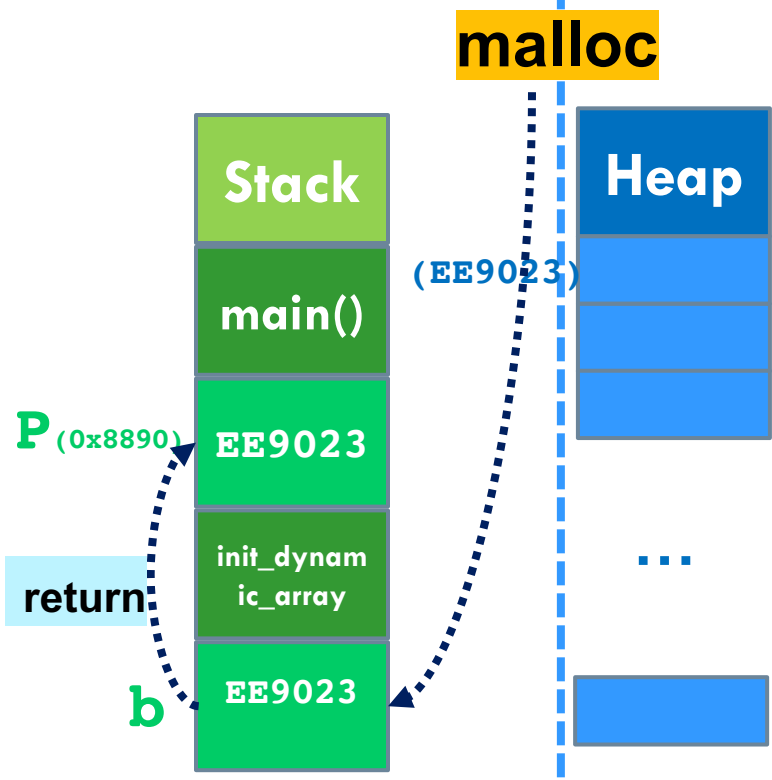
Δυναμική δέσμευση μνήμης


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

int main(void){
    int *p;
    p=(int *) malloc(20 * sizeof(int));
    *p=0;
    *(p+1)=1;
    p[2]=2;
    free(p);
    return 0;
}
```



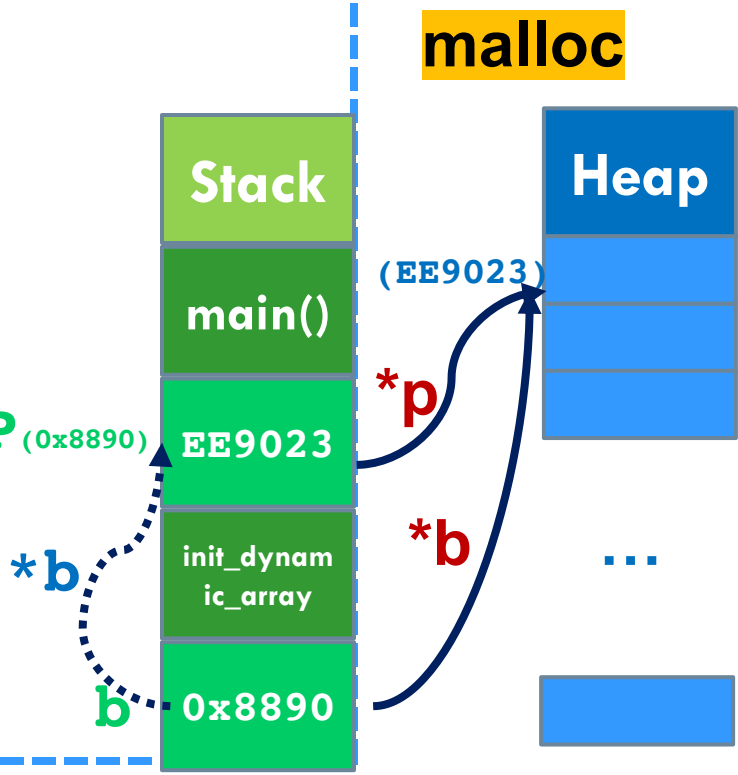
Δυναμική δέσμευση μνήμης – Με χρήση συνάρτησης (α)


```
#include <stdio.h>
#include <stdlib.h>
int * init_dynamic_array(int arrsize);
int main(void){
     int *p;
     p=init_dynamic_array(20);
     printf("%d",p[0]);
     free(p);
     return 0;
}

int * init_dynamic_array(int arrsize){
    int *b;
    b = malloc(sizeof(int) * arrsize);
    b[0] = 0;
    b[1] = 1;
    b[2] = 2;
    return b;
}
```



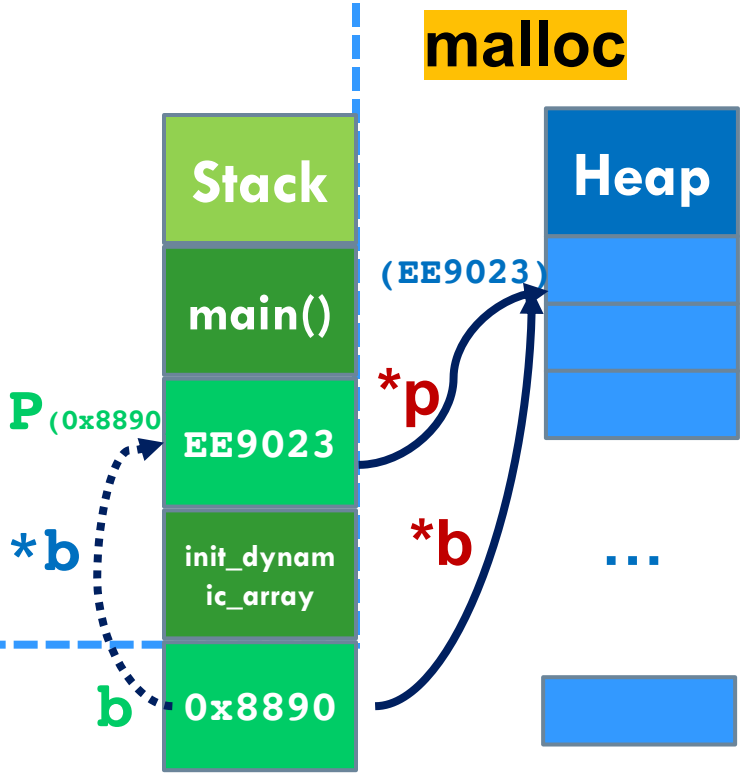
Δυναμική δέσμευση μνήμης με χρήση συνάρτησης (β)


```
#include <stdio.h>
#include <stdlib.h>
void init_dynamic_array(int ** b, int arrsize);
int main(){
     int * p;
     init_dynamic_array(&p,20);
     printf("%d",p[0]);
     free(p);
}

void init_dynamic_array(int ** b, int arrsize){
   *b = malloc(sizeof(int) * arrsize);
   (*b)[0] = 0;
   (*b)[1] = 1;
   (*b)[2] = 2;
}
```



Δυναμική δέσμευση μνήμης με χρήση συνάρτησης (γ)


```
#include <stdio.h>
#include <stdlib.h>
void init_dynamic_array(char ** b, int arrsize);
int main(){
     char * p;
     init_dynamic_array(&p,20);
     printf("%s",p);
     free(p);
}

void init_dynamic_array(char ** b, int arrsize){
   *b = malloc(sizeof(char) * arrsize);
   scanf("%s", *b);
}
```



```c
#include <stdio.h>
#include <stdlib.h>
#define N 10
int main (void ) {
     char matrix[N];
     char *dynamicdata;
     scanf("%s", matrix);
     printf("Hello %s!\n", matrix);
     dynamicdata = (char *) malloc( N * sizeof (char));
     scanf("%s", dynamicdata);
     printf("Hello dynamic %s!", dynamicdata);
     free(dynamicdata);
     return EXIT_SUCCESS;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define N 10
int main (void ) {
     char matrix[N];
     char *dynamicdata;
     scanf("%s", matrix);
     printf("Hello %s!\n", matrix);
     dynamicdata = (char *) malloc( N * sizeof (char));
     scanf("%s", dynamicdata);
     for (i=0; dynamicdata[i]!=0; i++)
          printf("%c\n", dynamicdata[i]);
     free(dynamicdata);
     return EXIT_SUCCESS;
}
```

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

int main ( void) {
      char *dynamicdata;
      int i, nchars;
      printf("How many chars?");
      scanf("%d", &nchars);
      dynamicdata = (char *) malloc( nchars * sizeof (char));
      scanf("%s", dynamicdata);
      printf("Hello dynamic %s!\n", dynamicdata);
      for (i=0;dynamicdata[i]!=0;i++) {
                        printf("%c\n", dynamicdata[i]);
      }
      free(dynamicdata);

      return EXIT_SUCCESS;
}
```

realloc

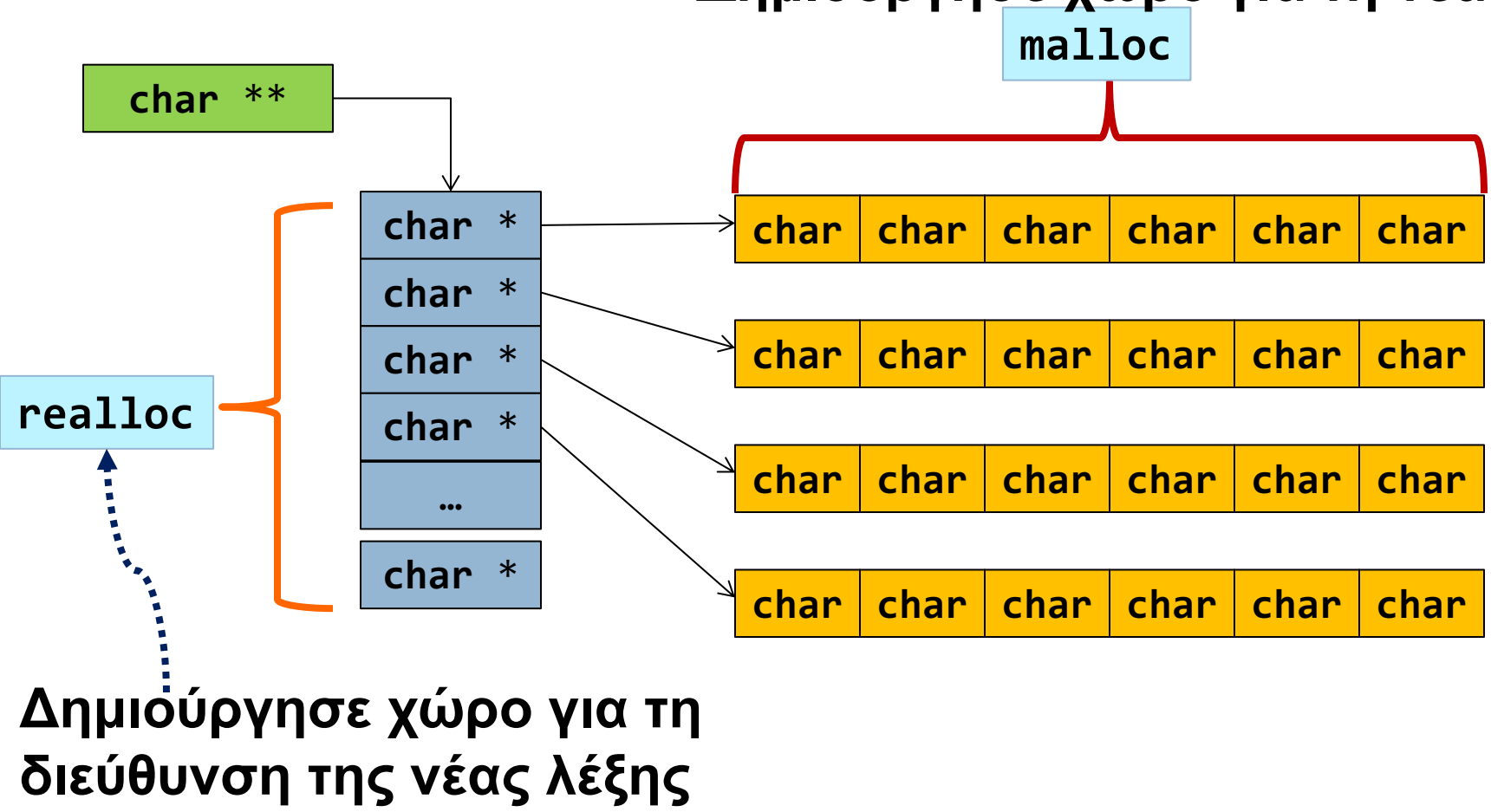
- **void** *realloc(**void** **ptr*, **size_t** *size*);
- Αλλάζει το μέγεθος περιοχής μνήμης με αρχή τη διεύθυνση *ptr* ώστε να έχει τελικό μέγεθος *size* bytes
- Μπορεί να επεκτείνει τη διαθέσιμη περιοχή αν είναι εφικτό ή να βρει νέα περιοχή μεταφέροντας δεδομένα.
- Η περιοχή μνήμης θα πρέπει να έχει ήδη ανατεθεί πριν την κλήση της realloc ή ο ptr να έχει την τιμή NULL
- Επιστρέφει NULL σε περίπτωση αποτυχίας

- Πρόγραμμα για ανάγνωση θετικών αριθμών και τοποθέτησή τους σε δυναμικό πίνακα.
- Όταν δοθεί ως είσοδος 0 ή αρνητικός αριθμός, εκτυπώνονται όσοι αριθμοί έχουν εισαχθεί νωρίτερα.

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

int main(void) {

        int *datatable = NULL;
        int d;
        int numbers = 0;
        int i;

        while (scanf("%d", &d), d>0) {
                numbers ++;
                datatable = realloc(datatable, numbers*sizeof(int));
                datatable[numbers - 1] = d;
        }

        for (i=0; i< numbers; i++)
                printf("%d\n", datatable[i]);

        free(datatable);

        return EXIT_SUCCESS;
}
```

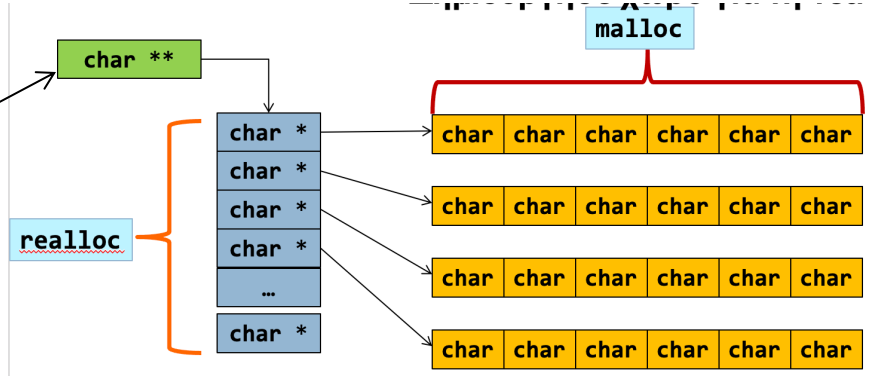
- Πρόγραμμα για ανάγνωση αλφαριθμητικών και τοποθέτησή τους σε δυναμικό πίνακα.
- Όταν δοθεί ως είσοδος «TELOS», να εκτυπώνονται όλες οι λέξεις που έχουν εισαχθεί νωρίτερα.

Δημιούργησε χώρο για τη νέα λέξη



Δημιούργησε χώρο για τη διεύθυνση της νέας λέξης

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char word[CHARS] = "";
        int i;
        while ( scanf("%s", word), strcmp(word,"TELOS") ) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = malloc (CHARS*sizeof(char));
                strcpy(mytext[words-1], word);
        }
        for (i=0; i<words; i++)
                printf("%s\n", mytext[i]);
        return EXIT_SUCCESS;
}
```



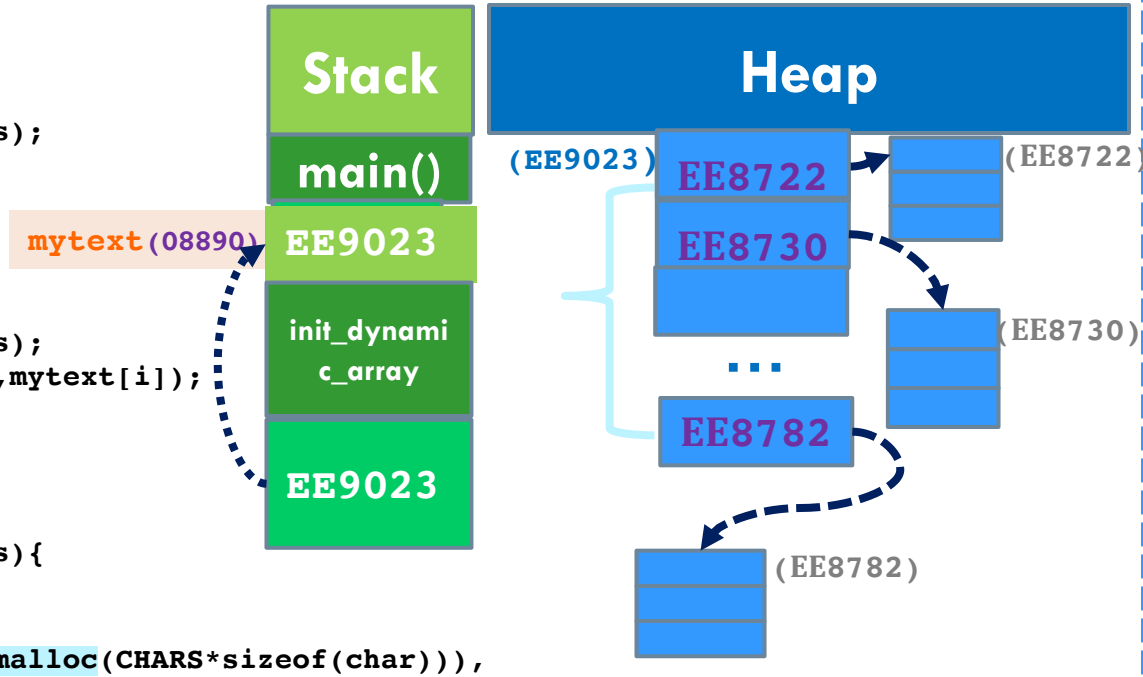
- **Πρόγραμμα για ανάγνωση αλφαριθμητικών και τοποθέτησή τους σε δυναμικό πίνακα.**
- **Όταν δοθεί ως είσοδος «TELOS», να εκτυπώνονται όλες οι λέξεις που έχουν εισαχθεί νωρίτερα.**

- Πρόγραμμα για ανάγνωση αλφαριθμητικών και τοποθέτησή τους σε δυναμικό πίνακα.
- Όταν δοθεί ως είσοδος «TELOS», να εκτυπώνονται όλες οι λέξεις που έχουν εισαχθεί νωρίτερα.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *word;
        int i;

        while (scanf("%s", word=malloc(CHARS*sizeof(char))),
                                                strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = word;
        }
        free(word);

        for (i=0; i<words; i++)
                printf("%s\n", mytext[i]);

        return EXIT_SUCCESS;
}
```

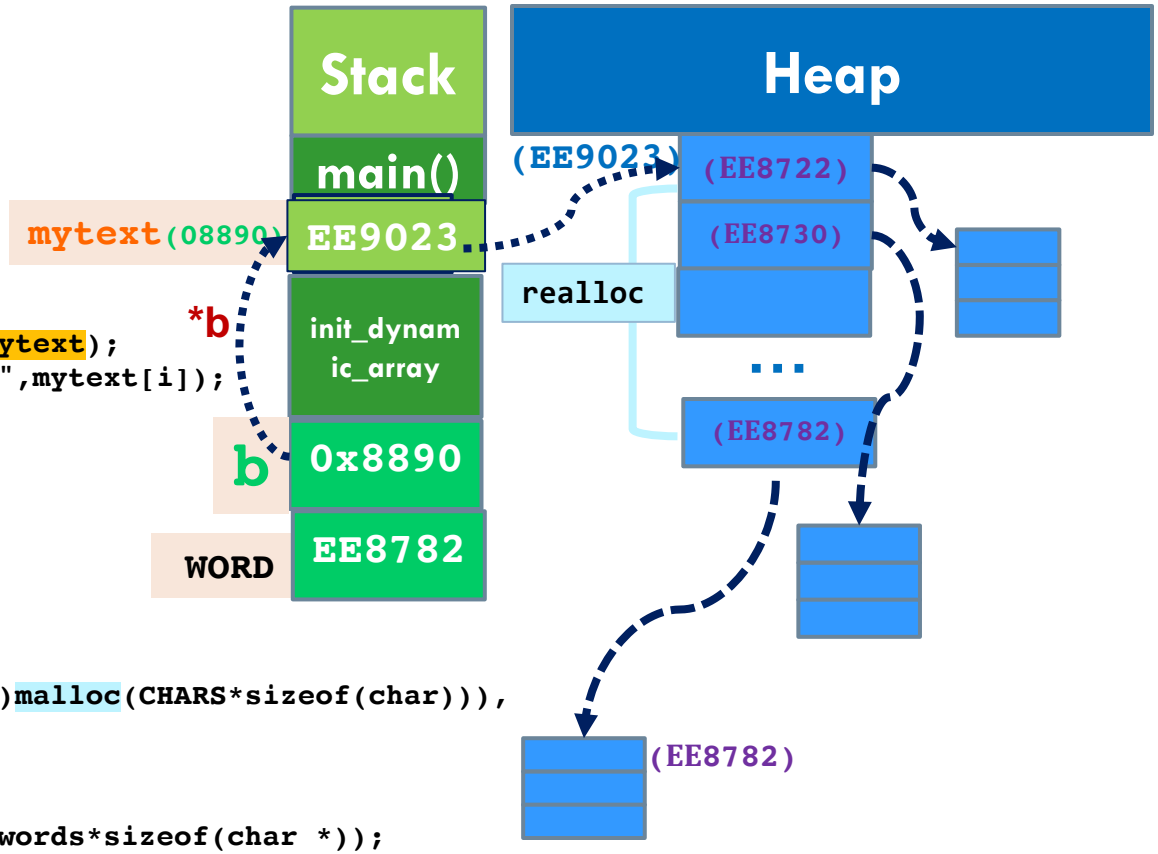
Δυναμική δέσμευση μνήμης με χρήση συνάρτησης


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 100
char** init_dynamic_arrayB(int *words);

int main()
{
    char **mytext=NULL;
    int i,words=0;
    mytext=init_dynamic_arrayB(&words);
    for(i=0;i<words;i++) printf("%s",mytext[i]);
    free(mytext);
    return 0;
}

char** init_dynamic_arrayB(int *words){
    char **mytext=NULL;
    char *WORD;
    while (scanf("%s", WORD= (char*)malloc(CHARS*sizeof(char))),
    strcmp(WORD,"TELOS")) {

        (*words)++;
        mytext = (char **) realloc(mytext, (*words)*sizeof(char *));
        mytext[ (*words)-1]=WORD;
    }
    free(WORD);
    return mytext;
}
```



Δυναμική δέσμευση μνήμης με χρήση συνάρτησης


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 100
int init_dynamic_array(char *** b);

int main()
{
    char **mytext=NULL;
    int i;
    int words=init_dynamic_array(&mytext);
    for(i=0;i<words;i++) printf("%s",mytext[i]);
    free(mytext);
    return 0;
}

int init_dynamic_array(char *** b){
    char *WORD;
    int words=0;

    while (scanf("%s", WORD= (char*)malloc(CHARS*sizeof(char))),
    strcmp(WORD,"TELOS")) {

        words++;
        *b = (char **) realloc(*b, words*sizeof(char *));
        (*b)[words-1]=WORD;
    }
    free(WORD);
    return words;
}
```



Διαχείριση πίνακα χαρακτήρων μεταβλητού μεγέθους

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

char * getname(void) ;

int main( ) {
  char other[] = "DO NOT ERASE ME";
  char *name;

  name = getname();

  printf("name : %s at %X\n", name, name);
  printf("size of name %d chars\n", strlen(name));
  printf("other: %s at %X\n", other, other);

  return EXIT_SUCCESS;
}
```

```
char *getname(void ) {
      int i = 0;
      int c ;
      char *more = (char *) malloc (1 * sizeof (char));

      while ((c = getchar())!='\n') {
         more[i] = c;
         if ((more = (char *) realloc (more, (1+(++i))*(sizeof (char))))==NULL){
                     printf("reallocation failed!");
                     exit(1);
          }
         printf("more: %X\n", more);
       }

       more[i] = '\0';
       printf("\ncharacters read i: %d\n", i);

   return more;
}
```

Η Δομή

```
struct <όνομα δομής> {
    <τύπος 1ου μέλους> <όνομα 1ου μέλους>;
    <τύπος 2ου μέλους> <όνομα 2ου μέλους>;
    <τύπος 3ου μέλους> <όνομα 3ου μέλους>;
    …
    <τύπος nου μέλους> <όνομα nου μέλους>;
} <λίστα ονομάτων μεταβλητών> ;
```

- **Παράδειγμα:**

```
struct Person {
   char   firstName[15];
   char   lastName[15];
   char   gender;
    int    age;
};
```

```
int main(void){
  struct Person x, y, z;
}
```

Δομές

- Η λέξη struct εισάγει τη δήλωση μιας δομής. Οι μεταβλητές που κατονομάζονται μέσα στη δομή ονομάζονται *μέλη* ή *πεδία*.

- Μια **δήλωση struct** ορίζει ένα τύπο. Για να δηλώσουμε μεταβλητές ή πίνακες τύπου δομής γράφουμε:

```
struct Person x;
struct Person people[40];
```

```
struct Person {
    char   firstName[15];
    char   lastName[15];
    char   gender;
    int    age;
};
```

- Εναλλακτικά μπορούμε επίσης να παραλείψουμε την ετικέτα της δομής και να περιγράψουμε απλά τα μέλη της:

```
struct {
        char    firstName[15];
        char    lastName[15];
        char    gender;
        int     age;
} x, people[40];
```

Δομές

- **Αρχικοποίηση** μιας μεταβλητής ή πίνακα τύπου δομής γίνεται αποδίδοντας τιμές στα μέλη ως εξής:

```
struct Person x = {“fname”, “lname”, ’M’, 43};
struct Person people[] =
                {{“fname1”, “lname1”,’M’, 43},
                 {“fname2”, “lname2”,’F’, 38},
                 {“fname3”, “lname3”, ’M’, 14}
                 };
```

- **Αναφορά σε μέλος μιας δομής** γίνεται μέσω της κατασκευής:

```
όνομα-δομής.μέλος
if (x.age > 40)
  printf(“%s age: %d \n”, x.lastName, x.age);
```

typedef

- Για να αποφεύγετε η συνεχής χρήση του «**struct structname», μπορούμε να χρησιμοποιήσουμε την typedef**
- π.χ.

typedef int myinteger;

Φτιάχνει τον τύπο uint16. Έτσι αποφεύγουμε να γράφουμε κάθε φορά «unsigned short int» και δηλώνουμε απλά uint16 X, Y;

Παρόμοια, typedef char* String;

- Τέλος μπορεί να χρησιμοποιηθεί για να δώσει ονόματα σε δομές:

```
struct Person{
   char   firstName[15];
   char   lastName[15];
   char   gender;
   int    age;
};
struct Person p;
p.age = 12;
```

```
typedef struct{
   char   firstName[15];
   char   lastName[15];
   char   gender;
   int    age;
} Person;
Person p;
p.age = 12;
```

typedef:Παράδειγμα

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
struct address {
 char street[20];
 int number;
 int code;
 char city[20];
} ;
typedef struct address Address;

void report (Address) ;
Address readaddress (void) ;

int main (void ) {

     Address myaddress ;

     myaddress = readaddress( );
     report (myaddress);

return EXIT_SUCCESS;
}
```

```c
Address readaddress (void)  {
     Address localaddress;
     printf("Odos:\t");
     scanf("%s", localaddress.street);
     printf("Ar. :\t");
     scanf("%d", &localaddress.number);
     printf("Code:\t");
     scanf("%d", &localaddress.code);
     printf("Poli:\t");
     scanf("%s", localaddress.city);

     return localaddress;
}

void report (Address local) {
     printf("Odos: %20s\n", local.street);
     printf("Ar. : %20d\n", local.number);
     printf("T.C.: %20d\n", local.code);
     printf("Poli: %20s\n", local.city);
   return ;
}
```

Φωλιασμένες Δομές

- Δομές μπορεί να είναι αλληλένθετες, δηλαδή να φωλιάζονται η μια μέσα στην άλλη, δημιουργώντας πιο πολύπλοκες δομές:

```
struct Person {
        char firstName[15];
        int age;
};
struct family {
        struct Person father;
        struct Person mother;
        int numofchild;
        struct Person children[5];
};

int main(void){
  struct Person x, y, z;
  struct family fml;

  fml.numofchild = 2;
  strcpy(fml.father.firstName, “Joe”);
  strcpy(fml.children[0].firstName, “Mary”);
  …
}
```

ή

(καλύτερα)

```
typedef struct {
        char firstName[15];
        int age;
} Person;
typedef struct {
        Person father;
        Person mother;
        int numofchild;
        Person children[15];
} Family;

int main(void){
  Person x, y, z;
  Family fml;

  fml.numofchild = 2;
  strcpy(fml.father.firstName, “Joe”);
  strcpy(fml.children[0].firstName, “Mary”);
  …
}
```

Δομές και Συναρτήσεις

- Επιτρεπτές πράξεις σε μια δομή είναι:
  - η **αντιγραφή** της, η **ανάθεση** τιμής σ' αυτήν (σαν σύνολο), η **εξαγωγή** της διεύθυνσής της, και η **προσπέλαση των στοιχείων** της.
  - Μεταβλητές τύπου δομής μπορούν να περαστούν ως ορίσματα σε συναρτήσεις όπως επίσης και να επιστραφούν ως αποτελέσματα συναρτήσεων.
- Παράδειγμα:

```
Person inc_age (Person x){
        x.age += 1;
        return x;
}
…
```

Πέρασμα δια τιμής

(το x αντιγράφετε μέσα στην συνάρτηση)

Δείκτες σε Δομές

- Μπορούμε επίσης να χρησιμοποιήσουμε **δείκτες σε δομές**.

- Για παράδειγμα η δήλωση

```
Person *pp, p;
```

δηλώνει ότι η μεταβλητή pp είναι δείκτης προς μια δομή τύπου Person . Έτσι μπορούμε να γράψουμε

```
pp = &p;
printf("%d", (*pp).age);
```

όπου ***pp** είναι η δομή που δείχνεται από τον δείκτη, ενώ **(*pp).firstName** είναι το πρώτο πεδίο της δομής.

- Προσοχή: η προτεραιότητα του τελεστή μέλους δομής, '.', είναι *μεγαλύτερη* από αυτή του τελεστή έμμεσης αναφοράς, '*'. **Επομένως οι παρενθέσεις στο (*pp).firstName είναι απαραίτητες.**

Δηλαδή το "*a.age=5" θα δώσει "compile error"

Δείκτες σε Δομές

- Δείκτες για δομές χρησιμοποιούνται τόσο συχνά που παρέχεται ο πιο κάτω εναλλακτικός συμβολισμός ως συντομογραφία:

```
(*p).μέλος_δομής    =    p->μέλος_δομής
```

- Αν μια μεγάλη δομή πρόκειται να μεταβιβαστεί για επεξεργασία σε μια συνάρτηση γενικά είναι αποτελεσματικότερη **η μεταβίβαση ενός δείκτη προς τη δομή και όχι η αντιγραφή της**. Αυτό μπορεί να γίνει όπως και σε απλούστερες δομές δεδομένων (δηλαδή με το &).

```
Person p;
. . .
initPerson(&p);
. . .
void initPerson(Person *p){
  strcpy( p->firstName, "Ανδρέας");
  strcpy( p->lastName, "Ανδρέου");
  p->gender = 'M';
  p->age =  43;
}
```

ή

```
Person *p;
(δέσμευση χώρου για το *p [malloc])
....
initPerson(p);
. . .
void initPerson(Person *p){
  strcpy( p->firstName, "Ανδρέας");
  strcpy( p->lastName, "Ανδρέου");
  p->gender = 'M';
  p->age =  43;
}
```

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

8η Εβδομάδα: Δυναμική Διαχείριση Μνήμης & Δομές Δεδομένων

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.**

Δυναμική Δέσμευση Μνήμης

- **`void *malloc(num_of_bytes):`** η malloc() δεσμευει δυναμικά ένα συνεχόμενο block μνήμης (μεγέθους **`num_of_bytes`**) και επιστρέφει ένα δείκτη στο χώρο μνήμης που δέσμευσε.

- Η malloc() επιστρέφει NULL όταν η αίτηση δεν μπορεί να ικανοποιηθεί (δηλαδή δεν υπάρχει άλλη διαθέσιμη μνήμη για να δοθεί).

- Για παράδειγμα η malloc(sizeof(int)) δεσμεύει μνήμη για την αποθήκευση ενός ακεραίου και επιστρέφει ένα δείκτη (τη διεύθυνση δηλαδή) του χώρου μνήμης που δέσμευσε.

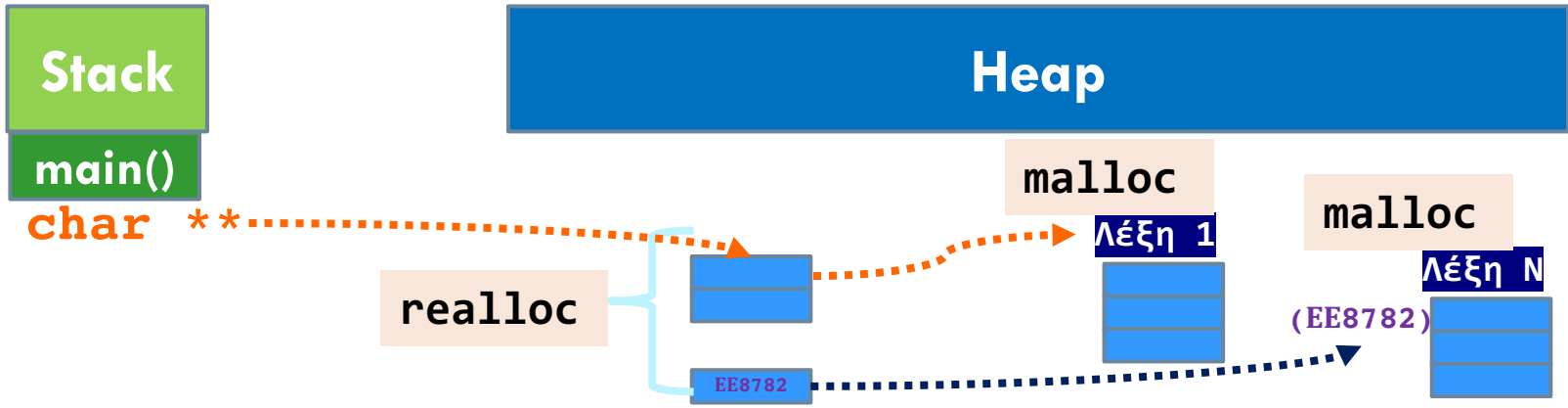
- Γιατί sizeof(int) και όχι απλά 4 => portability (αν κάποια πλατφόρμα χρησιμοποιεί 2 Bytes για αναπαράσταση ακεραίων τότε το πρόγραμμα μας εξακολουθεί να είναι σωστό!)

- Στη C μιλάμε πάντα για *δείκτες ενός συγκεκριμένου τύπου* πρέπει πάντα να κάνουμε cast τον δείκτη που επιστρέφει η malloc() στον αντίστοιχο τύπο. Δηλαδή

  (int *) malloc(sizeof(int))

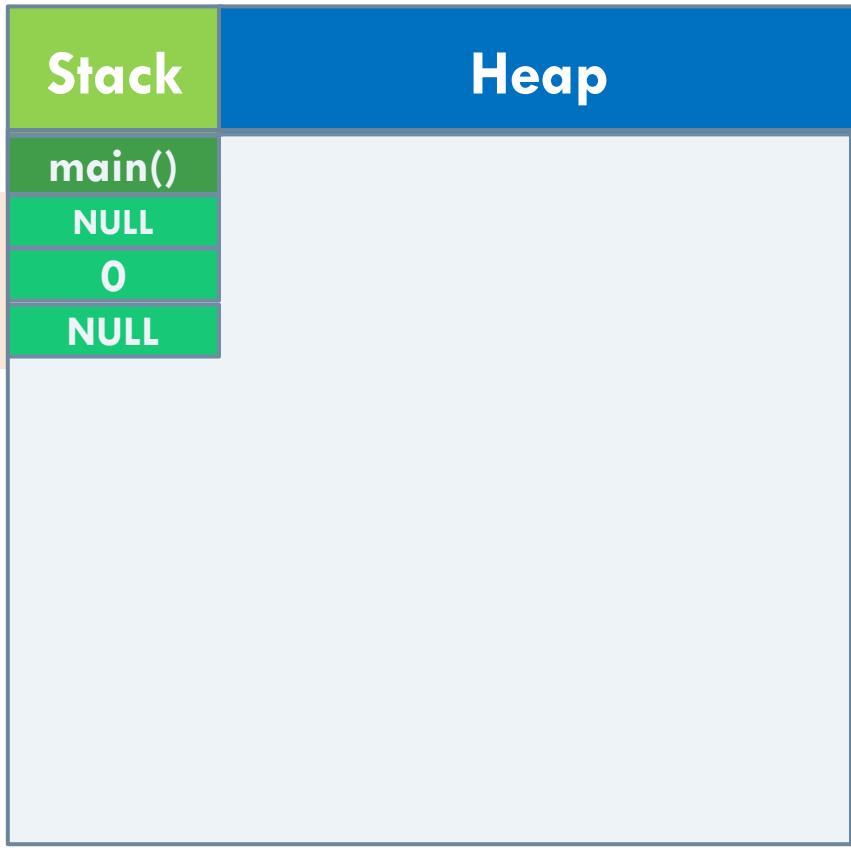
Άσκηση

- Πρόγραμμα για ανάγνωση αλφαριθμητικών και τοποθέτησή τους σε δυναμικό πίνακα.
- Όταν δοθεί ως είσοδος «TELOS», να εκτυπώνονται όλες οι λέξεις που έχουν εισαχθεί νωρίτερα.
- *Σχεδιασμός της λύσης (αφαιρετικό επίπεδο)*

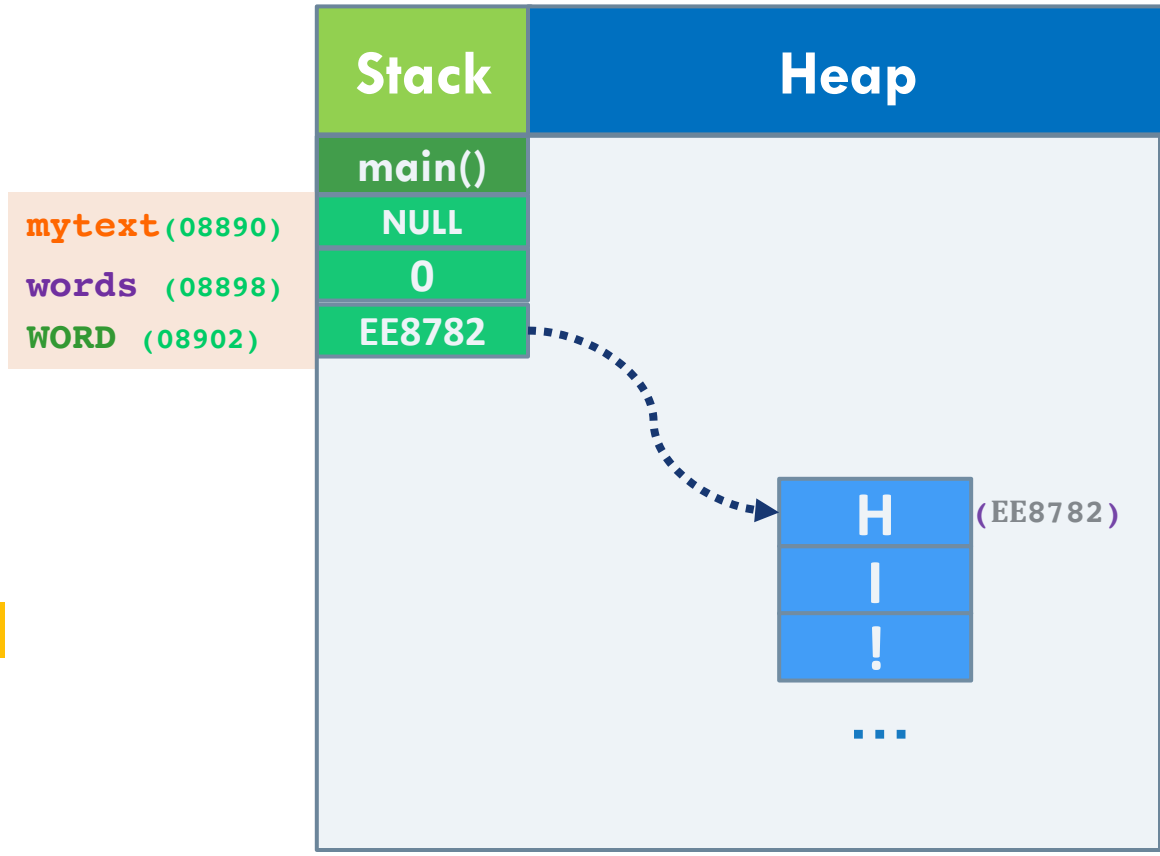


Ενδεικτική λύση

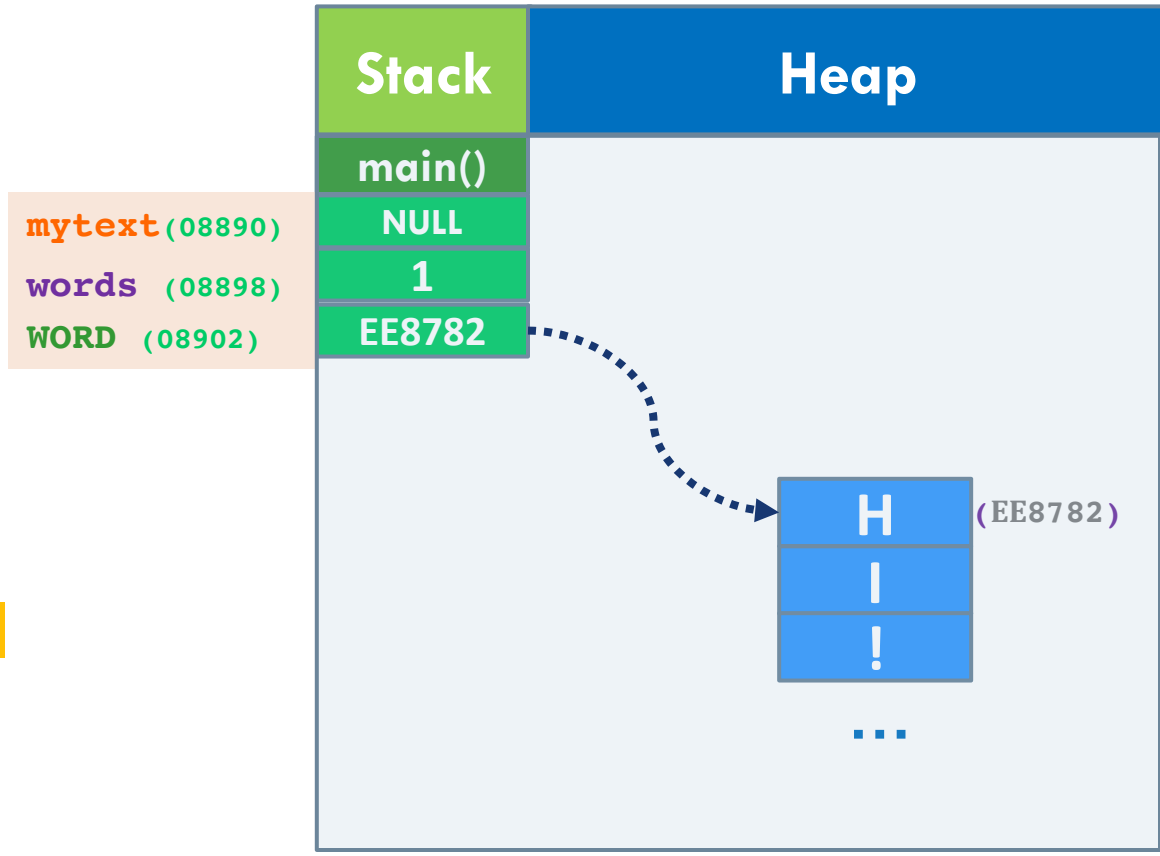

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *WORD=NULL;


        while (scanf("%s", WORD=malloc(CHARS*sizeof(char))),
                                        strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = WORD;
        }
        free(WORD);
        return EXIT_SUCCESS;
}
```



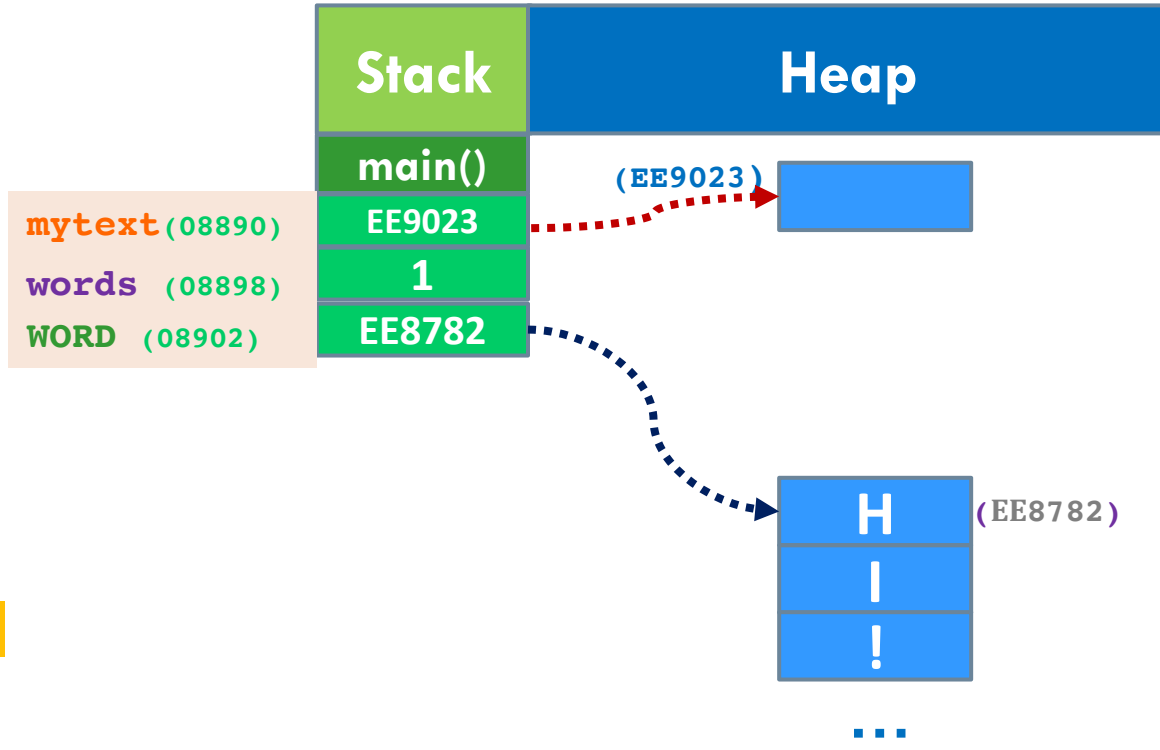

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *WORD=NULL;

        while (scanf("%s", WORD=malloc(CHARS*sizeof(char))),
                                        strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = WORD;
        }
        free(WORD);
        return EXIT_SUCCESS;
}
```



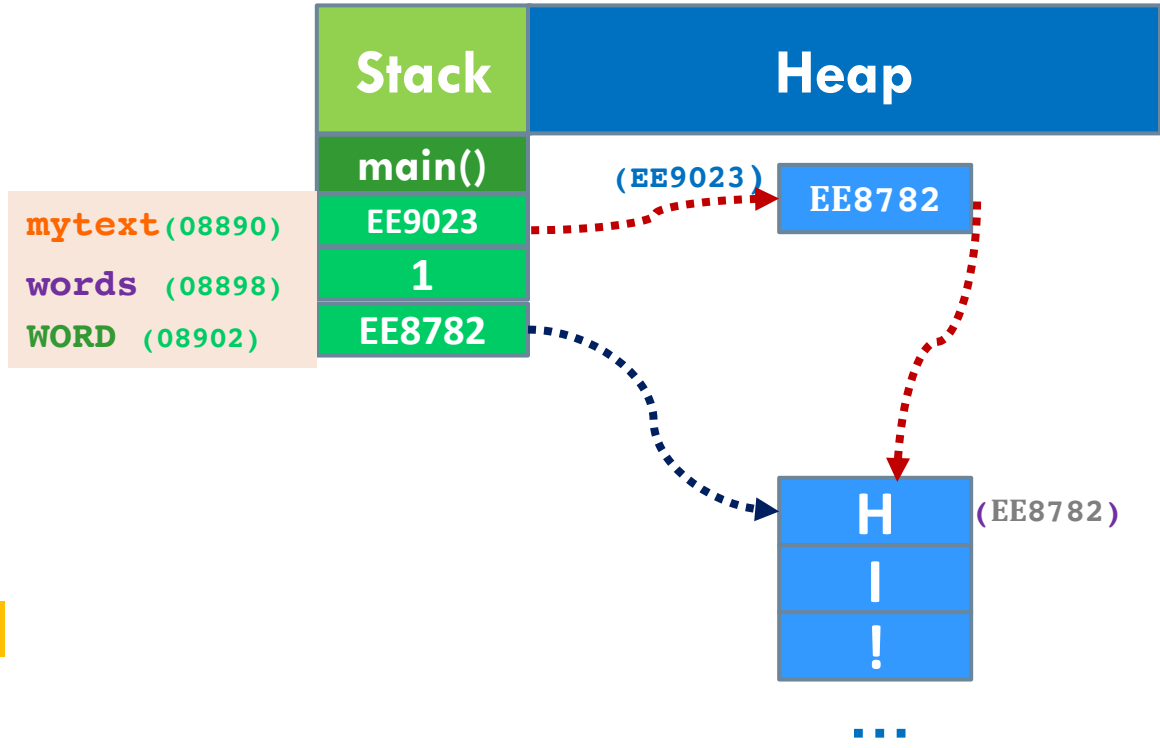

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *WORD=NULL;

        while (scanf("%s", WORD=malloc(CHARS*sizeof(char))),
                                        strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = WORD;
        }
        free(WORD);
        return EXIT_SUCCESS;
}
```




```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *WORD=NULL;

        while (scanf("%s", WORD=malloc(CHARS*sizeof(char))),
                                        strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = WORD;
        }
        free(WORD);
        return EXIT_SUCCESS;
}
```

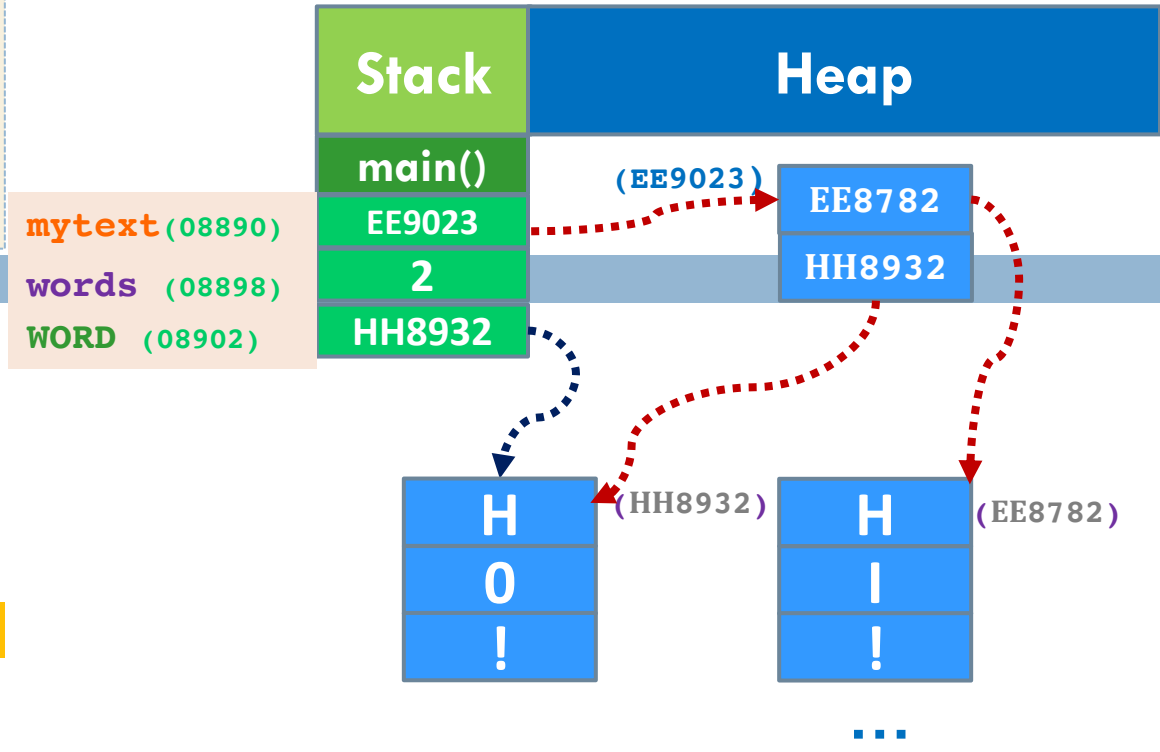



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *WORD=NULL;


        while (scanf("%s", WORD=malloc(CHARS*sizeof(char))),
                                                strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = WORD;
        }
        free(WORD);
        return EXIT_SUCCESS;
}
```



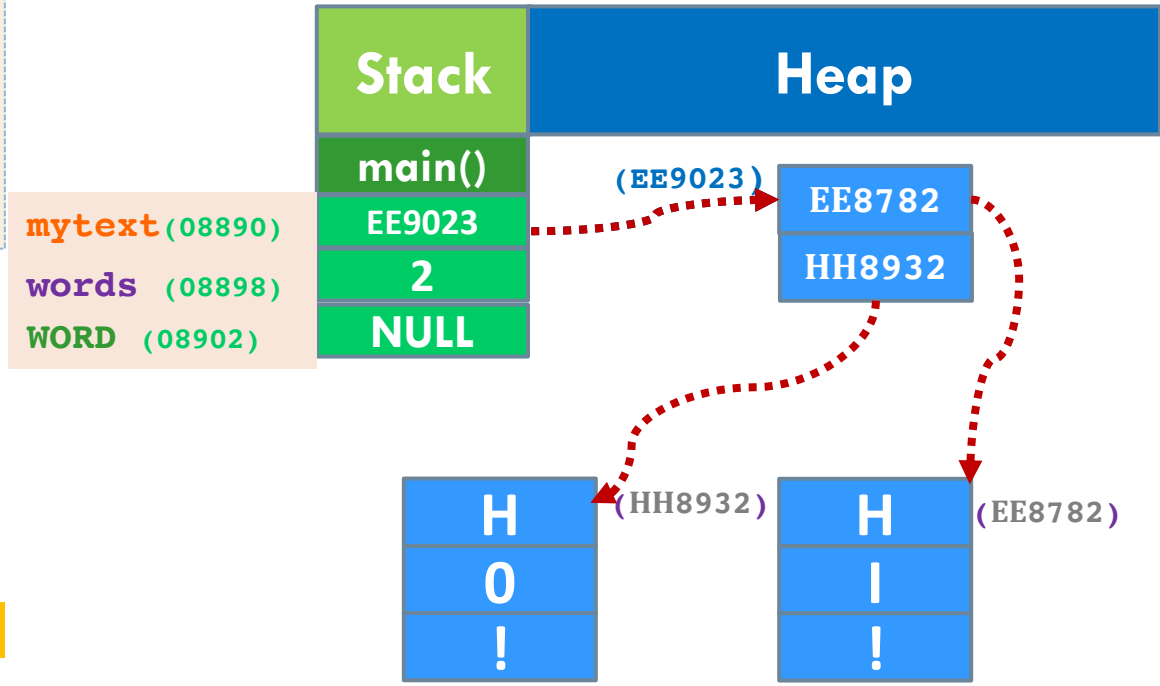

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *WORD=NULL;


        while (scanf("%s", WORD=malloc(CHARS*sizeof(char))),
                                                strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = WORD;
        }
        free(WORD);
        return EXIT_SUCCESS;
}
```

Μέχρι ο χρήστης να πληκτρολογήσει TELOS επαναλαμβάνεται η διαδικασία




```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *WORD=NULL;


        while (scanf("%s", WORD=malloc(CHARS*sizeof(char))),
                                                strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = WORD;
        }
        free(WORD);
        return EXIT_SUCCESS;
}
```

Μέχρι ο χρήστης να πληκτρολογήσει TELOS επαναλαμβάνεται η διαδικασία

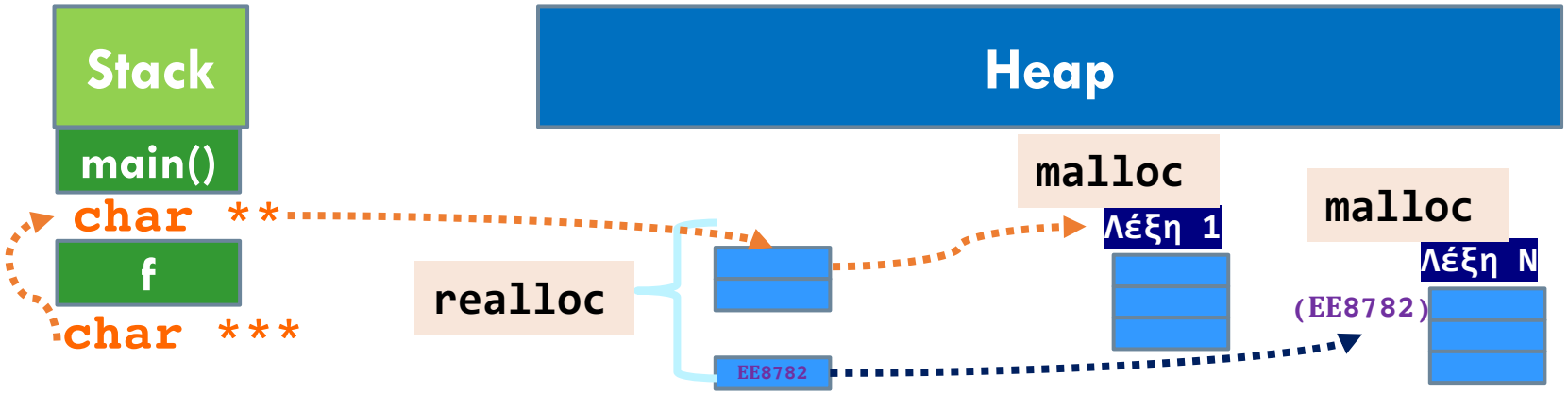



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
int main( void ) {
        char **mytext = NULL;
        int words = 0;
        char *WORD=NULL;
        int i;

        while (scanf("%s", WORD=malloc(CHARS*sizeof(char))),
                                        strcmp(word,"TELOS")) {
                words++;
                mytext = realloc(mytext, words*sizeof(char *));
                mytext[words-1] = WORD;
        }
        free(WORD);
        for (i=0; i<words; i++)
                printf("%s\n", mytext[i]);
```

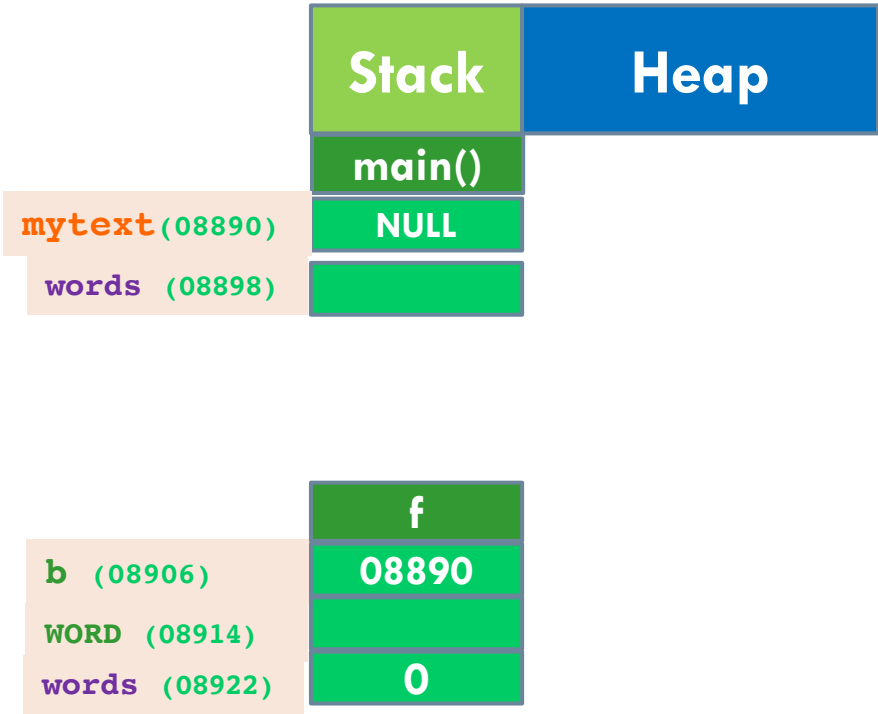
Για να αποδεσμευτεί στη σωρό ο χώρος που δεσμεύτηκε για τη λέξη TELOS

Άσκηση

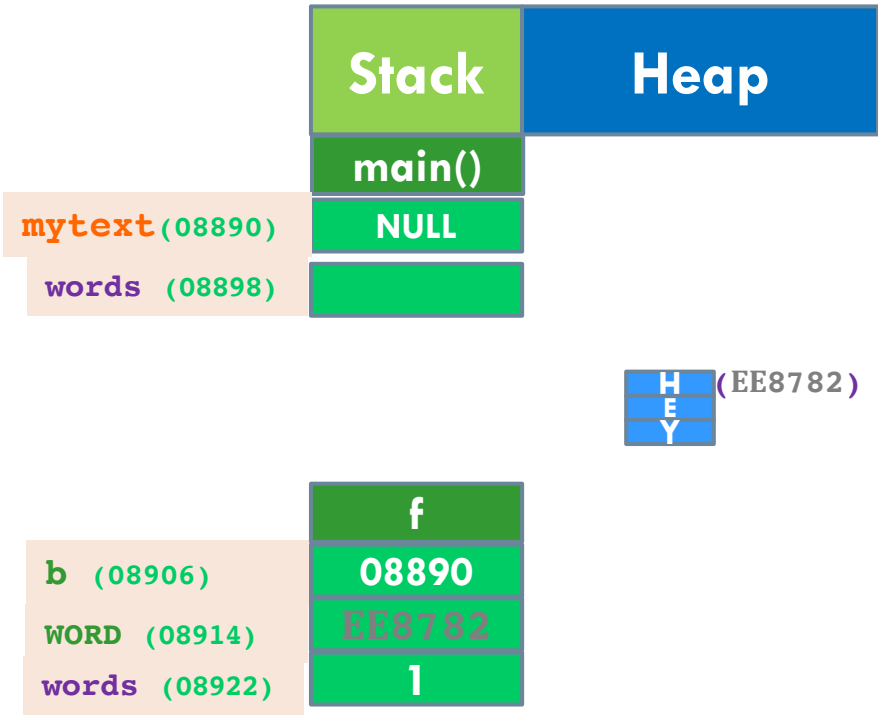
- Πρόγραμμα για ανάγνωση αλφαριθμητικών και τοποθέτησή τους σε δυναμικό πίνακα μέσω συνάρτησης με πρότυπο `int f(char ***b );`
- Όταν δοθεί ως είσοδος «TELOS», να εκτυπώνονται όλες οι λέξεις που έχουν εισαχθεί νωρίτερα.
- Σχεδιασμός της λύσης



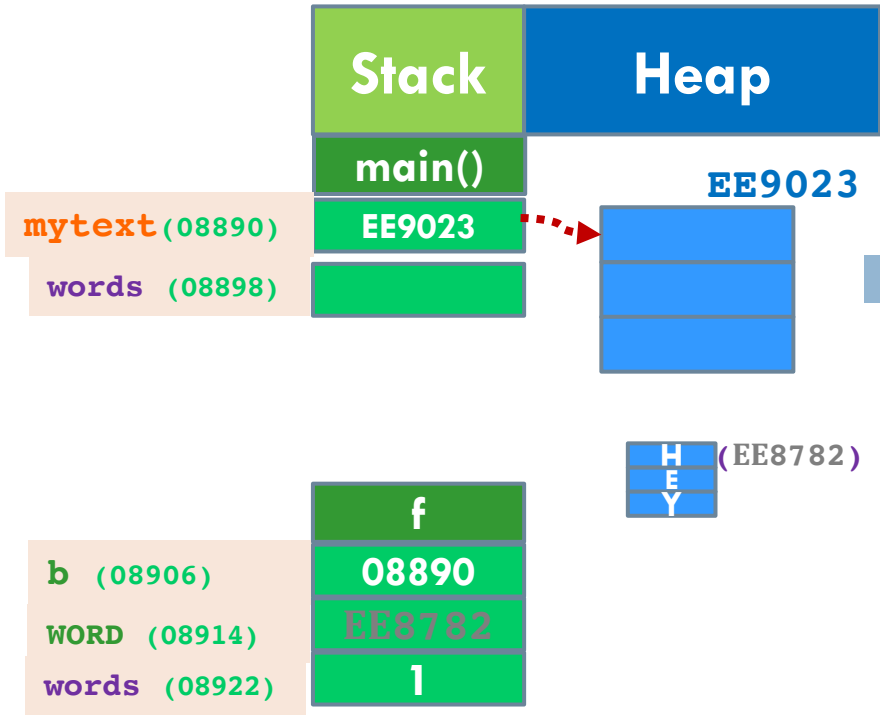
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 100
int f(char *** b);
int main()
{
  char **mytext=NULL;
  int words=f(&mytext);
  free(mytext);
  return 0;
}

int f(char *** b){
    char *WORD;
    int words=0;
    while (scanf("%s", WORD= (char*)malloc(CHARS*sizeof(char))),
    strcmp(WORD,"TELOS")) {

        words++;
        *b = (char **) realloc(*b, words*sizeof(char *));
        (*b)[words-1]=WORD;
    }
    free(WORD);
    return words;
}
```



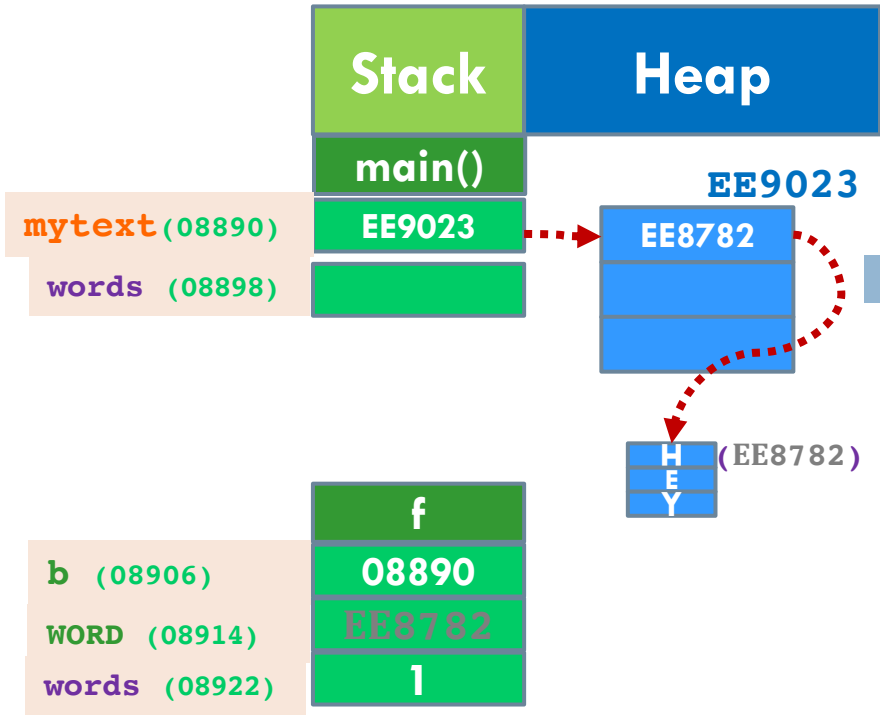
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 100
int f(char *** b);
int main()
{
  char **mytext=NULL;
  int words=f(&mytext);
  free(mytext);
  return 0;
}

int f(char *** b){
    char *WORD;
    int words=0;
    while (scanf("%s", WORD= (char*)malloc(CHARS*sizeof(char))),
    strcmp(WORD,"TELOS")) {

        words++;
        *b = (char **) realloc(*b, words*sizeof(char *));
        (*b)[words-1]=WORD;
    }
    free(WORD);
    return words;
}
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 100
int f(char *** b);
int main()
{
  char **mytext=NULL;
  int words=f(&mytext);
  free(mytext);
  return 0;
}

int f(char *** b){
    char *WORD;
    int words=0;
    while (scanf("%s", WORD= (char*)malloc(CHARS*sizeof(char))),
    strcmp(WORD,"TELOS")) {

        words++;
        *b = (char **) realloc(*b, words*sizeof(char *));
        (*b)[words-1]=WORD;
    }
    free(WORD);
    return words;
}
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 100
int f(char *** b);
int main()
{
  char **mytext=NULL;
  int words=f(&mytext);
  free(mytext);
  return 0;
}

int f(char *** b){
    char *WORD;
    int words=0;
    while (scanf("%s", WORD= (char*)malloc(CHARS*sizeof(char))),
    strcmp(WORD,"TELOS")) {

        words++;
        *b = (char **) realloc(*b, words*sizeof(char *));
        (*b)[words-1]=WORD;
    }
    free(WORD);
    return words;
}
```



Αυτοαναφορικές δομές

```
struct Employee {
        char                name[20];
        int                 age;
        struct Employee manager;
};
```

- Στην πιο πάνω δομή θα δοθεί "compile error", γιατί το struct Employee χρησιμοποιείται κατά την διάρκεια της δήλωσης του.

- Εντούτις μπορούμε να ορίσουμε δομές που αναφέρονται στον εαυτό τους, όπως στην προκειμένη περίπτωση δομή Employee όπου για κάθε στοιχείο της έχουμε πεδίο που αναφέρεται στον διευθυντή του εργοδοτημένου, χρησιμοποιώντας δείκτες:

```
struct Employee {
        char                name[20];
        int                 age;
        struct Employee *manager;
};
```

printf("%d", sizeof(struct Employee)); ➔ επιστρέφει 32

Δομή structure – Αυτοαναφορικές δομές

- Όταν χρησιμοποιούμε δυναμική δέσμευση μνήμης συνήθως το κάνουμε για την αποθήκευση όχι τόσο απλών τύπων δεδομένων (int, float, char κλπ.) **αλλά αντικειμένων τύπου structure.**

- Αυτό γιατί μπορούμε μέσω αντικειμένων τύπου structure να φτιάξουμε κόμβους, να τους συνδέσουμε μεταξύ τους και να δημιουργήσουμε έτσι μία συνδεδεμένη λίστα ή άλλες **εξελιγμένες δομές όπως στοίβες, ουρές, λίστες αναμονής, δέντρα κλπ.**

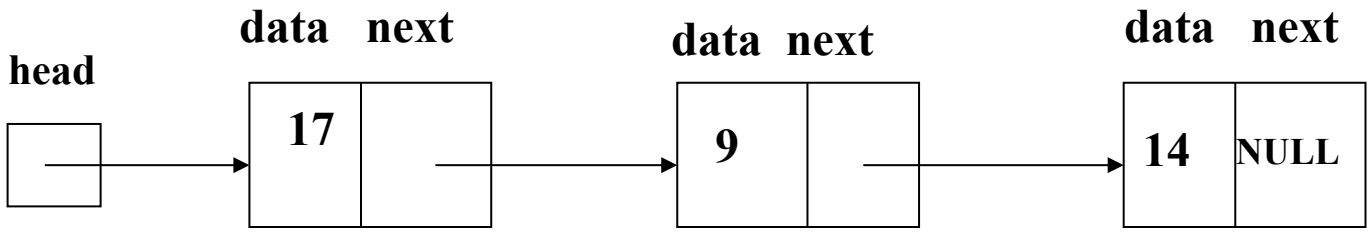
- Ένας απλός ορισμός κόμβου μιας **συνδεδεμένης λίστας** είναι ο εξής:

```
struct node {
        int data;
        struct node *next;
};
```

Προσέξτε ότι ένα πεδίο της δομής που υλοποιεί τον κόμβο είναι δείκτης στην ίδια δομή που ορίζεται. Αυτό το φαινόμενο όπως είπαμε ονομάζεται **αυτοαναφορική δομή (self-referential structures).**

Δομή τύπου structure (συν.)

- Έχοντας λοιπόν καθορίσει τη μορφή ενός κόμβου μπορούμε να φανταστούμε πως θα είναι μία συνδεδεμένη λίστα με κόμβους τύπου struct node που ορίσαμε νωρίτερα:



Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Πάμε να φτιάξουμε ένα κόμβο για την λίστα με τον αριθμό 17!

Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Πάμε να φτιάξουμε ένα κόμβο για την λίστα με τον αριθμό 17!

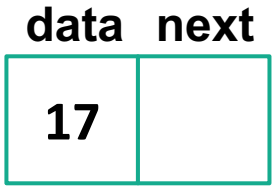
Πόσα πεδία έχει ο συγκεκριμένος κόμβος;

Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Πάμε να φτιάξουμε ένα κόμβο για την λίστα με τον αριθμό 17!



Πόσα πεδία έχει ο συγκεκριμένος κόμβος;
1 integer και 1 δείκτη σε struct node!

Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Πάμε να φτιάξουμε ακόμα ένα κόμβο για την λίστα με τον αριθμό 9!

Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

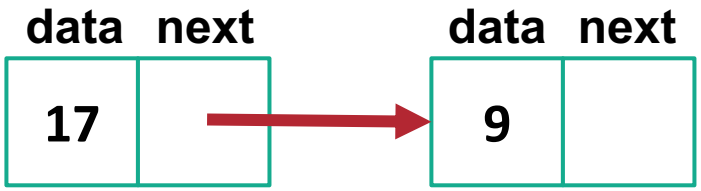
- Τι χρειάζεται να γίνει ώστε να θεωρηθεί λίστα;;;

Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Τι χρειάζεται να γίνει ώστε να θεωρηθεί λίστα;;;

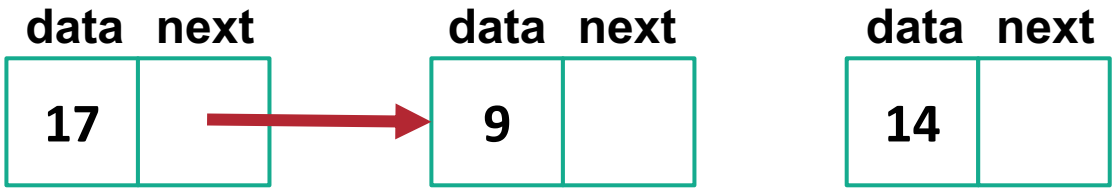


Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Ας προσθέσουμε ακόμα ένα κόμβο με τον αριθμό 14.

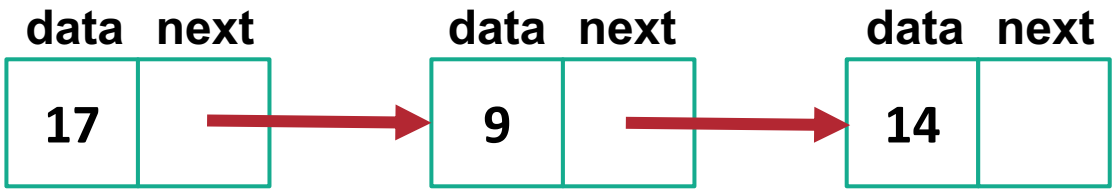


Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Ας τον ενώσουμε.

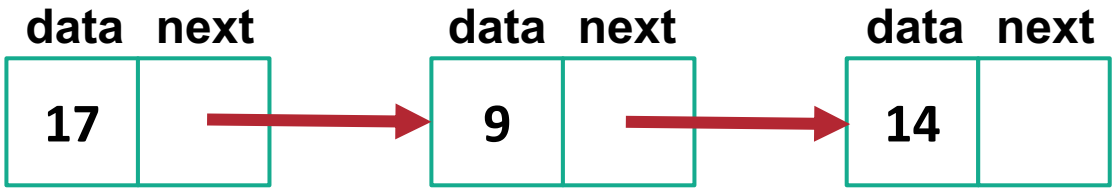


Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Μας λείπει κάτι;;;

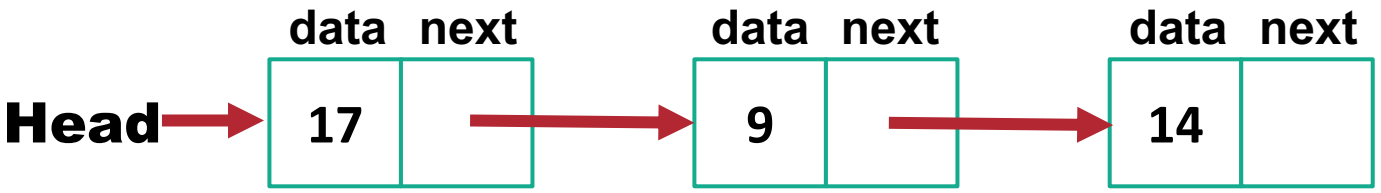


Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Βάλαμε και την αρχή της λίστας μας. Πού τελειώνει όμως;;;

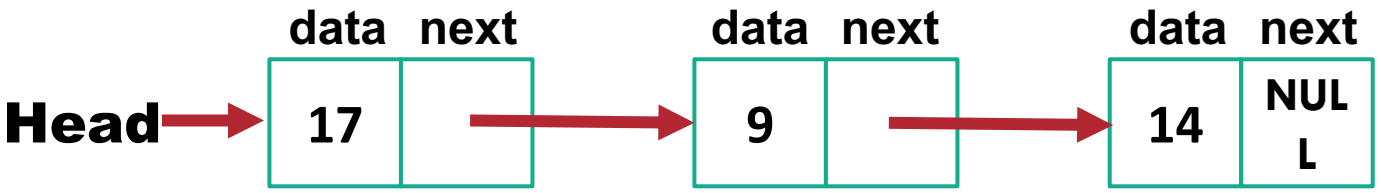


Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Τώρα η λίστα μας είναι ολοκληρωμένη!

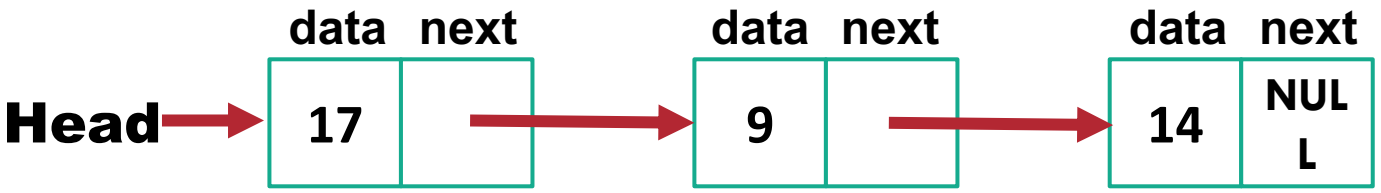


Λίστες

- Απλός ορισμός κόμβου συνδεδεμένης λίστας:

```
struct node {
        int data;
        struct node *next;
};
```

- Τώρα η λίστα μας είναι ολοκληρωμένη!



Δομή τύπου structure (συν.)

- Μια λίστα 10 στοιχείων θα μπορούσε να υλοποιηθεί ως εξής:

```
struct node {
   int data;
   struct node *next;
};
```

```
struct node *head, *temp;
int i;

head  = (struct node *) malloc (sizeof(struct node));
temp = head;
for(i=0; i<10; i++) {
        temp->data = i;
        if (i==9) break;
        temp->next = (struct node *) malloc(sizeof(struct node));
        temp = temp->next;
}
temp->next = NULL;
temp = head;

for(i=0; i<10; i++) {
        printf("%d %d\n", temp->data, temp->next);
        temp = temp->next;
}
```

- *Σημείωση: ο δείκτης `temp` είναι βοηθητικός για την υλοποίηση της λίστας.*

Η δήλωση typedef στη C

- Η C παρέχει μέσω της δήλωσης **`typedef`** τη δημιουργία νέων ονομάτων σε τύπους δεδομένων που ήδη υπάρχουν.

- Προσοχή: δεν δημιουργούμε νέους τύπους δεδομένων. Απλά **"βαφτίζουμε"** με νέα ονόματα τύπους που ήδη υπάρχουν. Μερικά παραδείγματα είναι τα εξής:

```
typedef int  Akeraios;
typedef struct node NODE;      /* το struct node
                                  έχει οριστεί νωρίτερα */
```

- Έτσι ορισμοί μεταβλητών μπορούν να υπάρξουν τώρα ως εξής:

```
Akeraios   len, l, arr[20];
NODE       head, tail, *temp;
```

Η δήλωση typedef στη C (συν.)

- Πολλές φορές **βαφτίζουμε** μία δομή ταυτόχρονα με τον **ορισμό** της, όπως στο παράδειγμα:

```
typedef struct node {
  int data;
  struct node *next;
} NODE;
```

- Είναι καλή συνήθεια να χρησιμοποιούμε ως όνομα στο `typedef` το ίδιο όνομα της δομής αλλά με κεφαλαία γράμματα.
- Γιατί να χρησιμοποιούμε τη δήλωση `typedef` ;
  1. Γιατί δημιουργεί πιο αναγνώσιμο και πιο κατανοητό κώδικα . Το να δηλώσεις `NODE *ptr;` είναι πιο κατανοητό από το να δηλώσεις ένα δείκτη σε μία πολύπλοκη δομή (`struct node *ptr;`).
  2. Ο κώδικας γίνεται πιο λιτός. Π.χ.

```
ptr = (NODE *)malloc(sizeof(NODE));
```

Δομή τύπου structure (συν.)

- Τώρα μια λίστα 10 στοιχείων θα μπορούσε να υλοποιηθεί και ως εξής:

```
NODE *head, *temp;
int i;

head  = (NODE *) malloc (sizeof(NODE));
temp = head;
for(i=0; i<10; i++) {
        temp->data = i;
        if (i==9) break;
        temp->next = (NODE *) malloc(sizeof(NODE));
        temp = temp->next;
}
temp->next = NULL;
temp = head;

for(i=0; i<10; i++) {
        printf("%d %d\n", temp->data, temp->next);
        temp = temp->next;
}
```

```
typedef struct node {
  int data;
  struct node *next;
} NODE ;
```

- *Σημείωση: ο δείκτης* `temp` *είναι βοηθητικός για την υλοποίηση της λίστας.*

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/