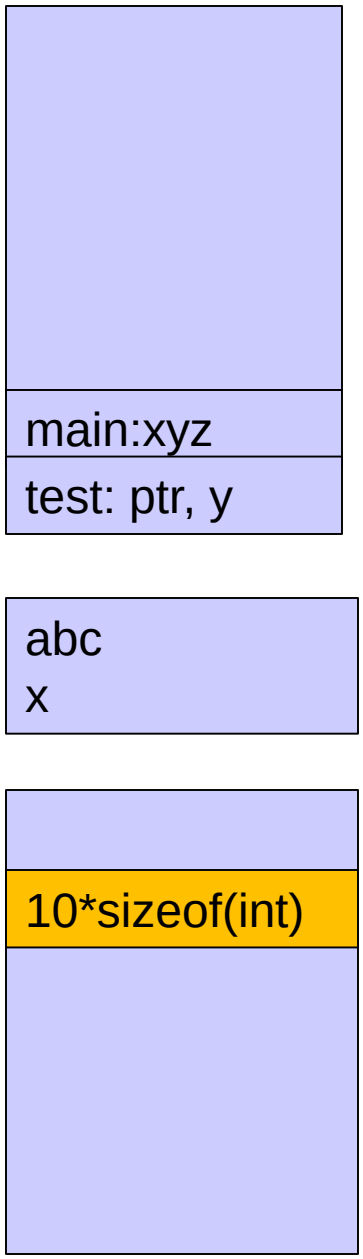
Διαδικαστικός Προγραμματισμός

Βασίλης Παλιουράς

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

int abc = 7;

int test (void);

int main(void) {
        int xyz = 3;
        printf("%d\n",test());
        printf("%d\n",test());
        printf("%d\n",test());
        printf("%d\n", xyz);
        return EXIT_SUCCESS;
}

int test (void) {
        static int x = 0;
        int * ptr ;
        int y = 0;
        x ++ ;
        y ++;
        ptr = malloc (10 * sizeof (int));
        ptr[0] = abc;
        printf("function: x: %d  y:%d ptr[0]:%d\n",x,y, ptr[0]);
        free(ptr);
        return x ;
}
```

Περιοχές μνήμης



Stack

Static/global

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.
- Απελευθέρωση μνήμης:
  - `void free(void *pointer);`

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;

```c
#include <stdio.h>
#include <stdlib.h>
#define N 10

int main ( void) {

   char matrix[N];

   scanf("%s", matrix);

   printf("Hello %s!\n", matrix);

   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);

   printf("Hello dynamic %s!", dynamicdata);

   return EXIT_SUCCESS;

}
```

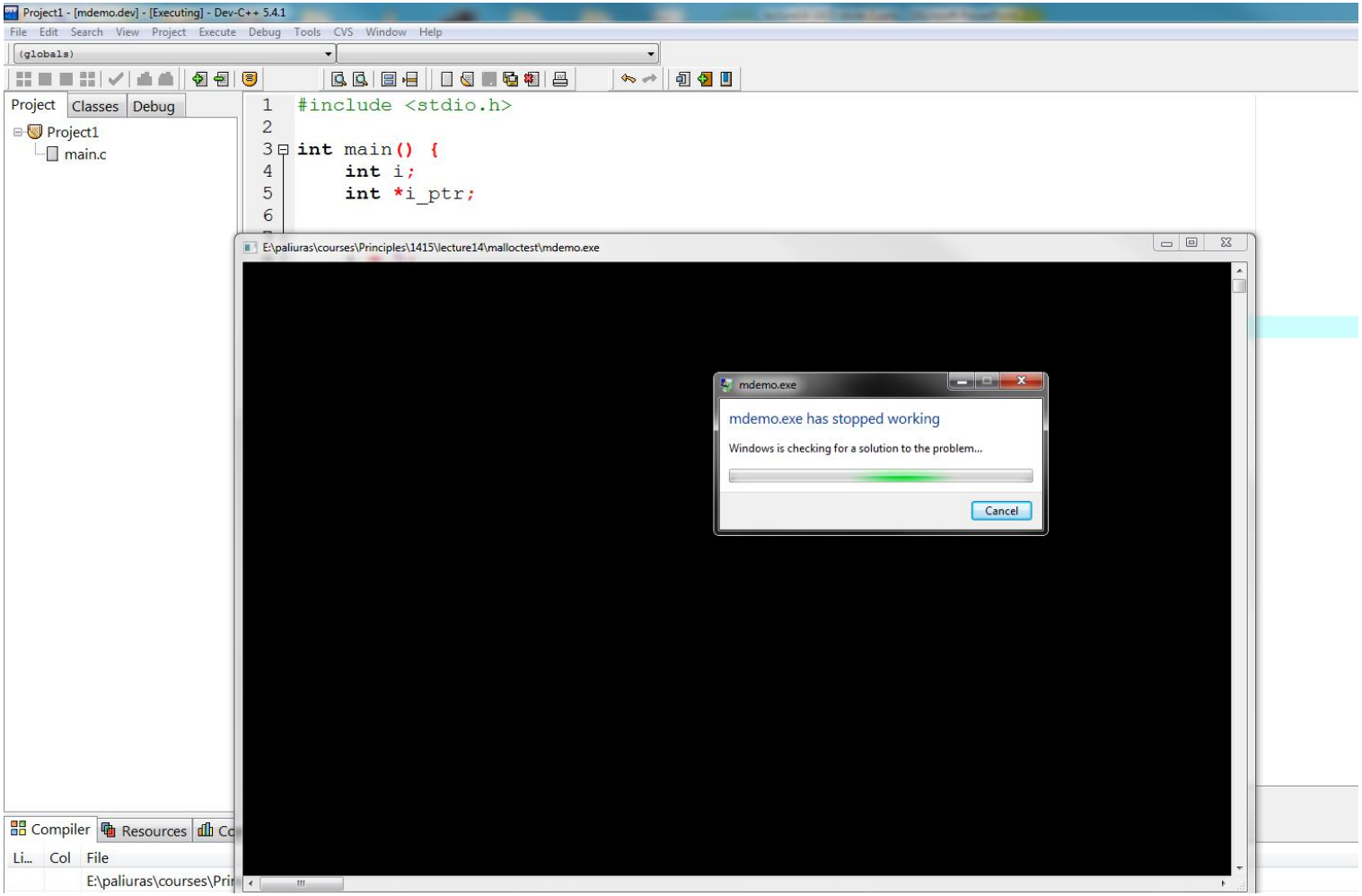
```c
#include <stdio.h>
#include <stdlib.h>
#define N 10

int main (void ) {

    char matrix[N];
    char *dynamicdata;
    int i;

    scanf("%s", matrix);

    printf("Hello %s!\n", matrix);

    dynamicdata = (char *) malloc( N * sizeof (char));
    scanf("%s", dynamicdata);

    printf("Hello dynamic %s!\n", dynamicdata);

    for (i=0; dynamicdata[i]!=0; i++)
         printf("%c\n", dynamicdata[i]);

    return EXIT_SUCCESS;

}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define N 10

int main ( void) {
    char matrix[N];
    char *dynamicdata;
    int i, nchars;

    scanf("%s", matrix);
    printf("Hello %s!\n", matrix);

    while (1) {
          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;
}
```

Βρείτε γιατί δεν τρέχει το πρόγραμμα, χρησιμοποιώντας τον debugger!

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

int main(void) {
        int i;
        int *i_ptr;


        i = 5;
        *i_ptr = -6;
        printf("%d\n", i);

        return EXIT_SUCCESS;
}
```

(τα warnings προειδοποιούν!!!)


Project1 - [mdemo.dev] - [Executing] - Dev-C++ 5.4.1
File Edit Search View Project Execute Debug Tools CVS Window Help
(globals)
Project Classes Debug
Project1
main.c
1 #include <stdio.h>
2
3 int main() {
4 int i;
5 int *i_ptr;
6
E:\paliuras\courses\Principles\1415\lecture14\malloctest\mdemo.exe
mdemo.exe
mdemo.exe has stopped working
Windows is checking for a solution to the problem...
Cancel
Compiler Resources
Li... Col File
E:\paliuras\courses\Pri

Μέγεθος stack

- Οι πίνακες χρησιμοποιούν το stack
  - Όπως οι αυτόματες μεταβλητές στη C
- Περιορισμένο μέγεθος stack
- Μπορεί να αυξηθεί
  - με οδηγία στο linker
    - `-Wl,--stack,<μέγεθος σε bytes>`
  - Σε linux, πχ με την εντολή ulimit

Global/static

- Οι πίνακες global, static δεν χρησιμοποιούν το stack

- Θέματα που σχετίζονται με το OS, τον compiler, την έκδοση κ.ά.

Δυναμική διαχείριση

- Ο χώρος μνήμης που διατίθεται με `malloc`, `realloc`, `calloc` δεν είναι στο stack

- Μπορώ να διαλέξω στο Project Options>Compiler>Code generation το pointer width (32 bit / 64 bit) σε περίπτωση που ενδιαφέρει διαχείριση μνήμης > 4GB

- 32 bit pointer $\rightarrow 2^{32}$ θέσεις = $4 \times 2^{30} = 4$ G

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

- Να γραφεί 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.

- Διάβασε έναν αριθμό
- Όσο ο αριθμός είναι θετικός:
  - Αύξησε το πλήθος των θετικών κατά ένα
  - Αύξησε το μέγεθος του πίνακα κατά μία θέση ακεραίου
  - Αποθήκευσε τον αριθμό στον πίνακα
  - Διάβασε έναν αριθμό

- Διάβασε τον ακέραιο d
- Όσο d>0
  - Αύξησε το numbers κατά ένα
  - Αύξησε το μέγεθος του πίνακα datatable κατά μία θέση ακεραίου
  - Αποθήκευσε το d στην τελευταία θέση του πίνακα datatable
  - Διάβασε τον ακέραιο d

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

int main(void) {
        int *datatable = NULL;
        int d;
        int numbers = 0;
        int i;

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

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

        free(datatable);

        return EXIT_SUCCESS;
}
```

```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;
}
```

```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;
}
```

```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;
}
```

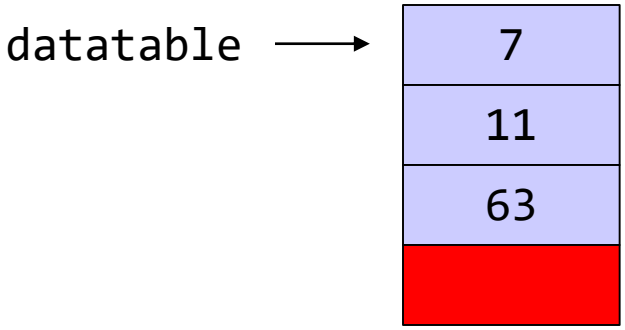
Αρχικοποίηση σε NULL

Αυξανόμενο μέγεθος πίνακα στο heap

Διεύθυνση πρώτης θέσης του νέου block

Το d καταχωρείται στην τελευταία θέση του block


datatable
NULL

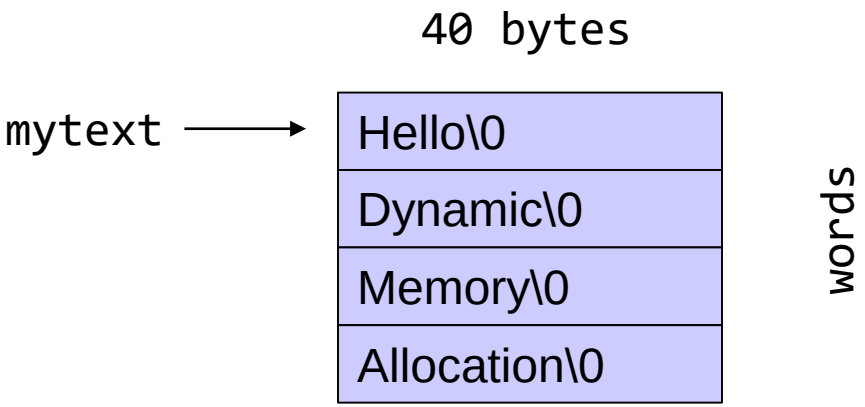

datatable
7
11
63


35
numbers = 4

typedef

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

typedef char Word[CHARS];

int main(void) {
 Word myword;

 scanf("%s", myword);
 printf("%s\n", myword);
 printf("%d %d", (unsigned int) strlen(myword), (unsigned int) sizeof myword);

 return EXIT_SUCCESS;
}
```

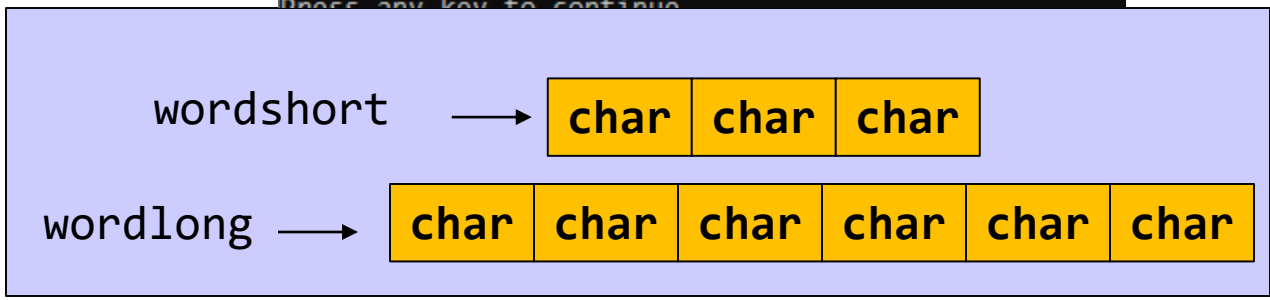
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 40

typedef char Word[CHARS];
typedef Word * Text;
int main(void) {
        Word myword;
        Text mytext = NULL;
        int words = 0;

        while (scanf("%s", myword), strcmp(myword,"bye")) {
                words ++;
                mytext = realloc(mytext, words * sizeof (Word) ) ;
                strcpy(mytext[words-1], myword);
        }
        {
                int i;
                for(i = 0; i<words; i++) printf("%s ", mytext[i]);
        }
        free(mytext);
        return EXIT_SUCCESS;
}
```


40 bytes
mytext
Hello\0
Dynamic\0
Memory\0
Allocation\0
words

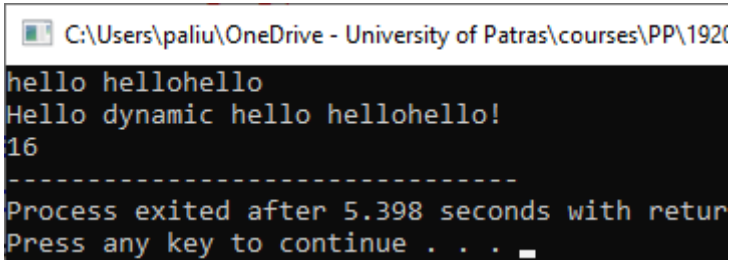

```c
#include <stdio.h>
#include <stdlib.h>
#define N 40

int main (void ) {

        char * dynamicdata;

        dynamicdata = malloc( N * sizeof (char));
        scanf("%s", dynamicdata);

        printf("Hello dynamic %s!\n", dynamicdata);

        free(dynamicdata);

        return EXIT_SUCCESS;

}
```

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

int main (void ) {

        char * wordshort;
        char * wordlong;

        wordshort = malloc( 20 * sizeof (char));
        wordlong  = malloc( 60 * sizeof (char));

        scanf("%s", wordshort);
        scanf("%s", wordlong);

        printf("Hello dynamic %s %s!\n", wordshort, wordlong);
        printf("%d %d", (int) sizeof wordshort, (int) sizeof wordlong);
        free(wordshort);
        free(wordlong);

        return EXIT_SUCCESS;

}
```

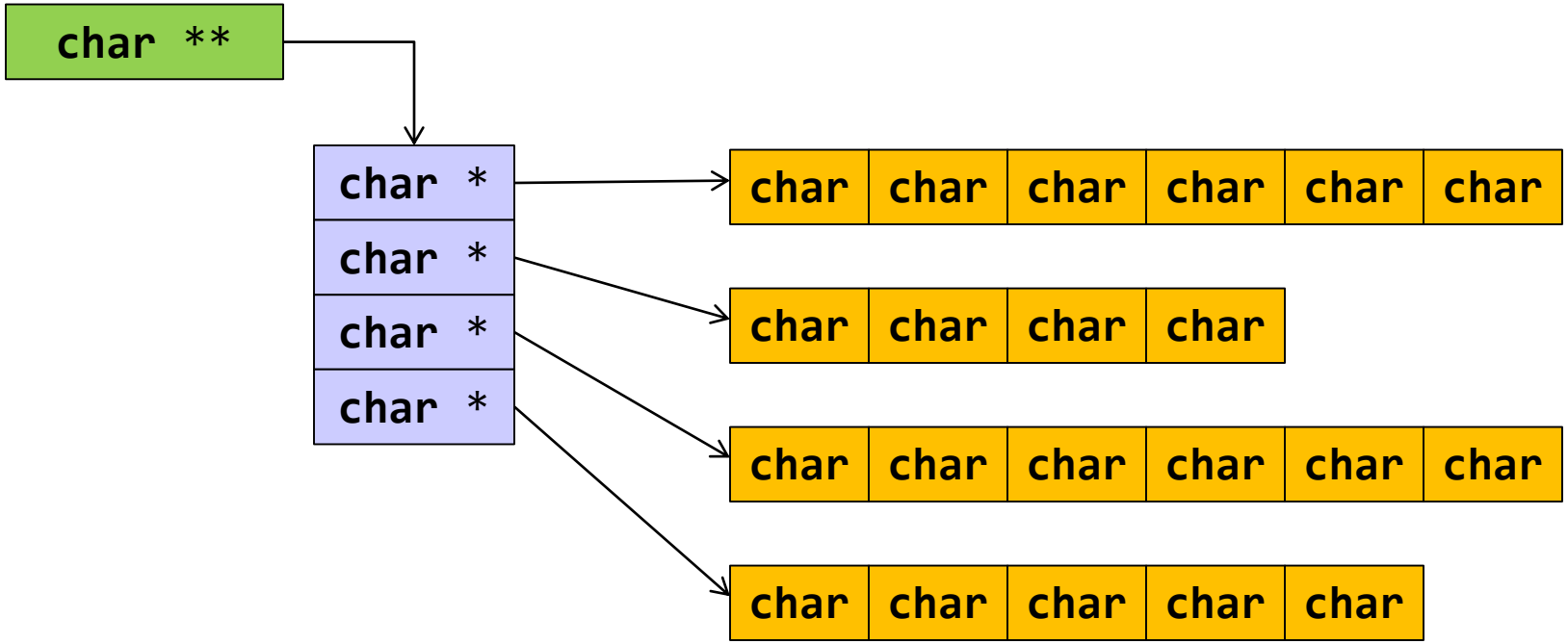
```
Select C:\Users\paliu\OneDrive - University of Patras\courses\PP\1920\rem
hello
hellohellohellohello
Hello dynamic hello hellohellohellohello!
8 8
--------------------------------
Process exited after 7.312 seconds with return value
Press any key to continue
```

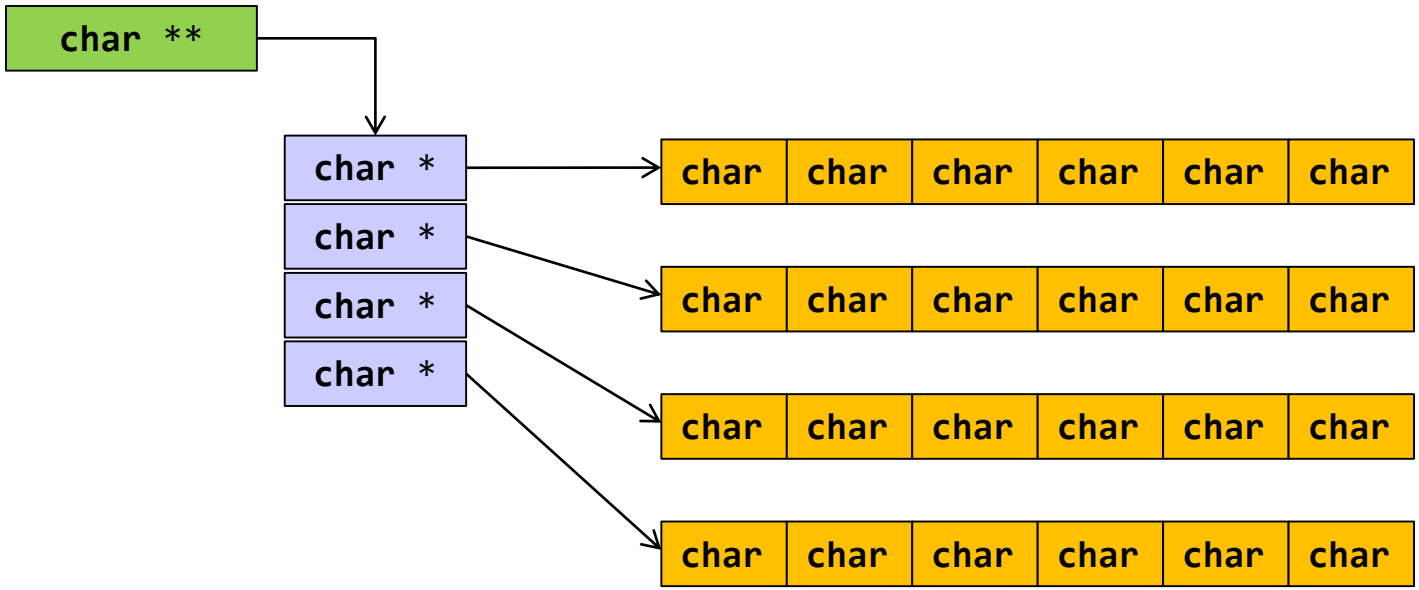


```c
#include <stdio.h>
#include <stdlib.h>

int main (void ) {

        char * words[2];

        words[0] = malloc( 20 * sizeof (char));
        words[1] = malloc( 60 * sizeof (char));

        scanf("%s", words[0]);
        scanf("%s", words[1]);

        printf("Hello dynamic %s %s!\n", words[0], words[1]);
        printf("%d", (int) sizeof words);
        free(words[0]);
        free(words[1]);

        return EXIT_SUCCESS;

}
```



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

int main (void ) {

        char ** words;

        words = malloc ( 2 * sizeof (char *));

        words[0] = malloc( 20 * sizeof (char));
        words[1] = malloc( 60 * sizeof (char));

        scanf("%s", words[0]);
        scanf("%s", words[1]);

        printf("Hello dynamic %s %s!\n", words[0], words[1]);
        printf("%d", (int) sizeof words);
        free(words[0]);
        free(words[1]);
        free(words);

        return EXIT_SUCCESS;
}
```


char **
char *
char *
char *
char *
char char char char char char
char char char char
char char char char char char
char char char char char



- Δημιούργησε χώρο για τη νέα λέξη
- Δημιούργησε χώρο για τη διεύθυνση της νέας λέξης
- Αποθήκευσε τη διεύθυνση της νέας λέξης

realloc

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 100
int main( void ) {
   char **mytext = NULL;
   int words = 0;
   char word[CHARS] = "";
   int i;

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

   for (i=0; i<words; i++)
        printf("%s\n", mytext[i]);
   /* Think about free !!!*/
   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;
        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;
}
```

Διαχείριση πίνακα χαρακτήρων μεταβλητού μεγέθους

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

char * getname(void) ;

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

  name = getname();

  printf("name : %s at %p\n", name, name);
  printf("size of name %d chars\n", (int) strlen(name));
  printf("other: %s at %p\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: %p\n", more);
  }

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

  return more;
}
```