

Διαδικαστικός Προγραμματισμός

Βασίλης Παλιουράς

Τι είναι ταχύτερο;

```
#include <stdio.h>
#include <time.h>
#define TIMES 1000000

typedef struct test {
    char data[100];
} Test;

void byvalue(Test a) {
    Test b;
    /* do something */
}

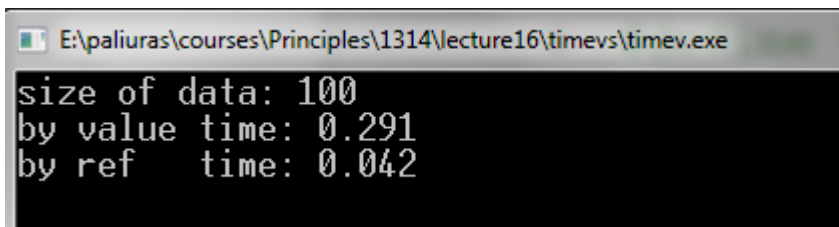
void byref(Test *a ) {
    Test b;
    /* do something */
}
```

```
int main( void) {
    Test a ;
    int i;
    clock_t start, stop;

    start = clock();
    printf("size of data: %d\n", sizeof (Test));
    for (i=0;i<TIMES; i++)
        byvalue(a);
    stop = clock();
    printf("by value time: %g\n",
        (double) (stop - start)/CLOCKS_PER_SEC);

    start = clock();
    for (i=0;i<TIMES; i++)
        byref(&a);
    stop = clock();
    printf("by ref  time: %g\n",
        (double) (stop - start)/CLOCKS_PER_SEC);

    return 0;
}
```



```
E:\paliuras\courses\Principles\1314\lecture16\timevs\timev.exe
size of data: 100
by value time: 0.291
by ref  time: 0.042
```

Κλήση με αξία και
Κλήση με αναφορά

Πού αποθηκεύονται παράμετροι και τοπικές μεταβλητές: stack

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

```
int f (int );  
int g (int );  
double h (double );  
int w (int, int);
```

```
int main(void) {
```

```
    f(1); ← Η f δεν καλεί την g
```

```
    g(1);
```

```
    h(1.0);
```

```
    w(1, 2);
```

```
    f(2);
```

```
    g(1);
```

```
    return 0;
```

```
}
```

Η f καλεί την g

```
int f(int a) {  
    int b = 1 ;  
    printf("function f: address of parameter %X\n", &a);  
    printf("\t\t address of local variable %X\n", &b);  
    if (a>1)  
        g(a);  
    return b + a;  
}
```

```
int g (int a) {
    int b = 1 ;
    int *c = &a;
    printf("function g: address of parameter %X\n", &a);
    printf("\t\t address of local variable b %X\n", &b);
    printf("\t\t address of local variable c %X\n", &c);
    return b + a ;
}
```

```
int w (int a, int c) {
    int b = 1 ;
    printf("function w: address of parameter %X\n", &a);
    printf("\t\t address of local variable %X\n", &b);
    return b + a + c;
}
```

```
double h (double a) {
    double b = 2.0 ;
    printf("function h: address of parameter %X\n", &a);
    printf("\t\t address of local variable %X\n", &b);
    return b + a;
}
```

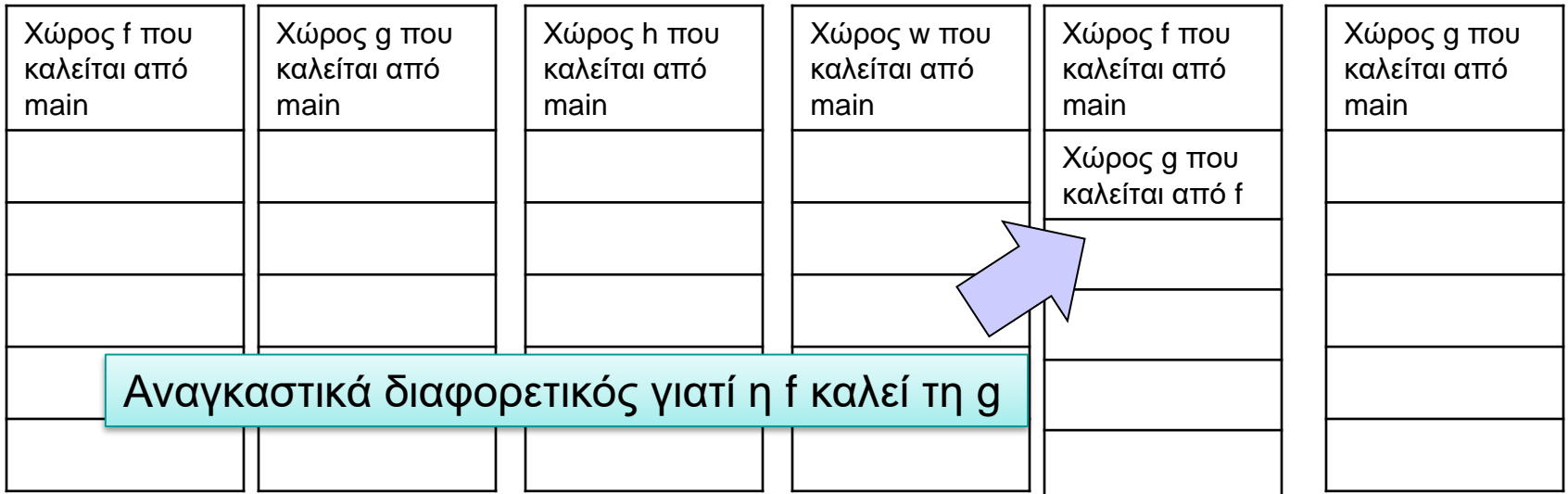
Εικόνα του stack κατά τη διάρκεια εκτέλεσης των συναρτήσεων

```

C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
function f: address of parameter 22FE40
            address of local variable 22FE2C
function g: address of parameter 22FE40
            address of local variable b 22FE2C
            address of local variable c 22FE20
function h: address of parameter 22FE40
            address of local variable 22FE28
function w: address of parameter 22FE40
            address of local variable 22FE2C
function f: address of parameter 22FE40
            address of local variable 22FE2C
function g: address of parameter 22FE00
            address of local variable b 22FDEC
            address of local variable c 22FDE0
function g: address of parameter 22FE40
            address of local variable b 22FE2C
            address of local variable c 22FE20

-----
Process exited with return value 0
Press any key to continue . . .
  
```

Κάθε φορά διατίθεται ο ίδιος χώρος μνήμης



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

```
int abc = 7;
```

```
int test (void);
```

```
int main(void) {
    int xyz ;
    printf("%d\n",test());
    printf("%d\n",test());
    printf("%d\n",test());
    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 ;
}
```

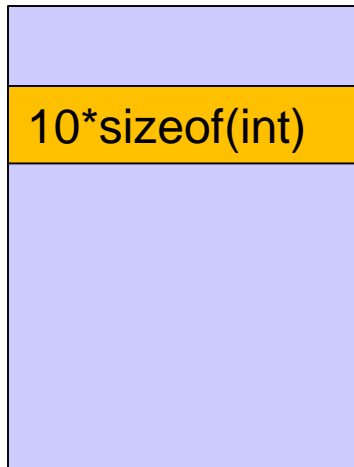
Περιοχές μνήμης



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;

```
#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;
}
```

```
#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;

}
```

```
#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;

}
```

```

#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!

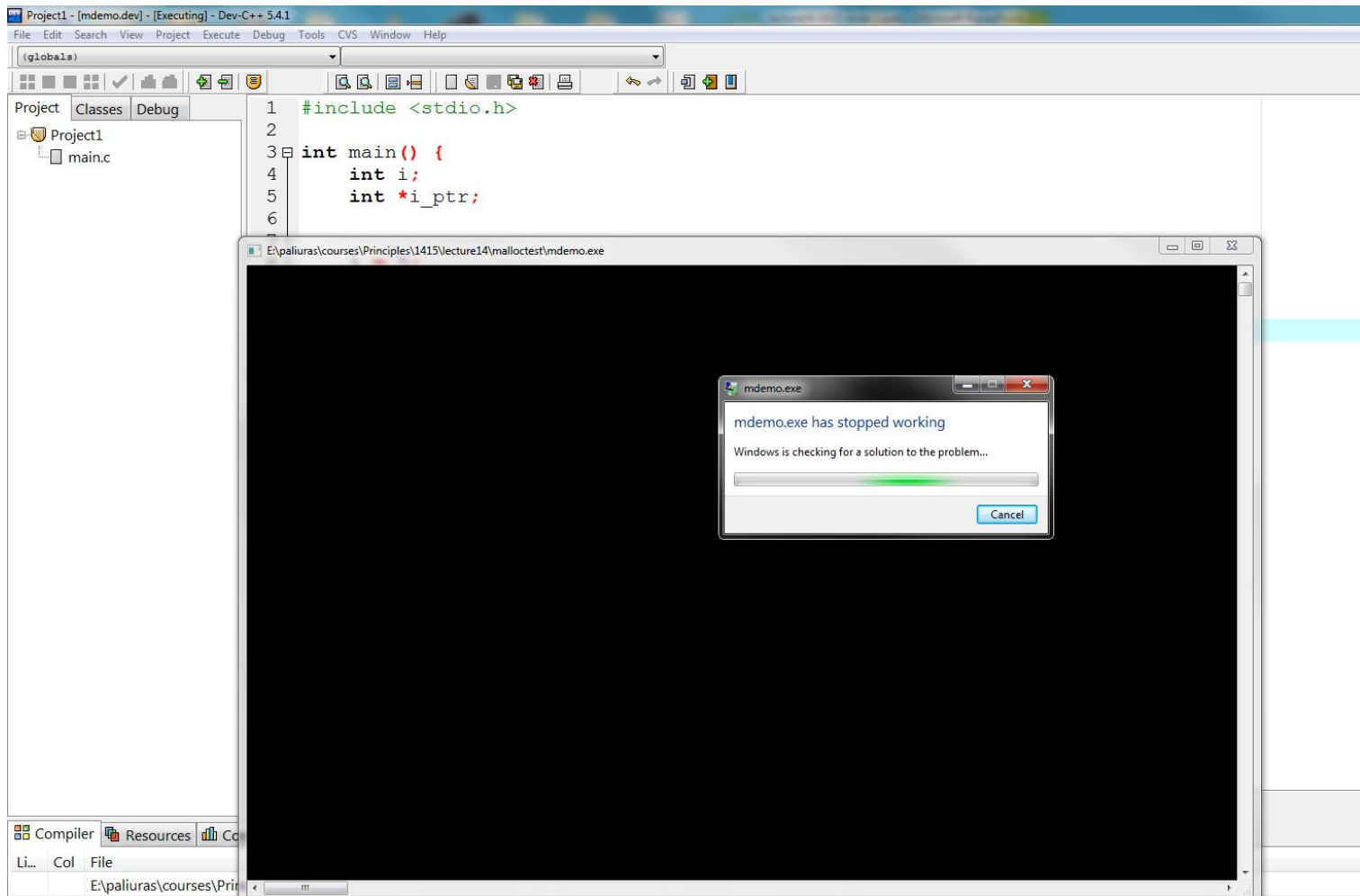
```
#include <stdio.h>
#include <stdlib.h>

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

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

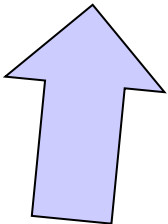
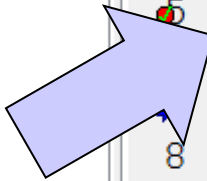
    return EXIT_SUCCESS;
}
```

(τα warnings προειδοποιούν!!!)



i_ptr = (int *) 0x1f001e

i = 0

Δήλωση δείκτη
Χωρίς αρχικοποίηση:
«τυχαία» τιμή

```

1  #include <stdio.h>
2
3  int main() {
4      int i;
5      int *i_ptr;
6
7      i = 5;
8      *i_ptr = -6;
9      printf("%d\n", i);
10
11     return 0;
12 }
```

Debug Add watch Next line Next instruction Stop Execution Modify watch Into line Into instruction View CPU window Remove watch Continue Skip function

Send command to GDB: display i

->->prompt

(globals)

Project Classes Debug

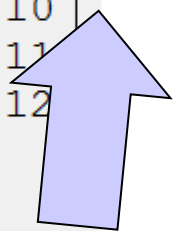
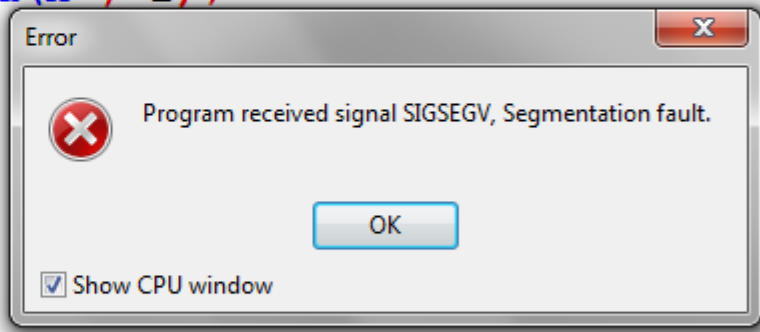
- i_ptr = (int *) 0x1f001e
- i = 5

main.c

```

1  #include <stdio.h>
2
3  int main() {
4      int i;
5      int *i_ptr;
6
7      i = 5;
8      *i_ptr = -6;
9      printf("%d\n", i);
10
11     return 0;
12

```



Λόγω της μη αρχικοποίησης,
Segmentation fault όταν προσπαθεί να γράψει σε «τυχαία» θέση μνήμης

Compiler Resources Compile Log Debug Find Results Close

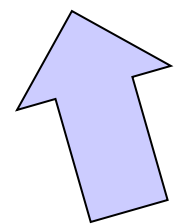
Debug Add watch Next line Next instruction Send command to GDB: next
 Stop Execution Modify watch Into line Into instruction ->-> prompt
 Show CPU window Remove watch Continue Skip function

Project Classes Debug

- i_ptr = (int *) 0x1
- i = 0

main.c

```
1 #include <stdio.h>
2
3 int main() {
4     int i;
5     int *i_ptr;
6
7     i_ptr = &i;
8     i = 5;
9     *i_ptr = -6;
10    printf("%d\n", i);
11
12    return 0;
13 }
```



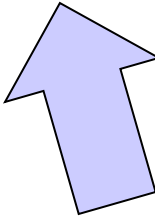
Αρχικοποιώ το δείκτη στη θέση της μεταβλητής i

Debug Add watch Next line Next instruction Stop Execution Modify watch Into line Into instruction View CPU window Remove watch Continue Skip function

Send command to GDB: display i
->->prompt

Project Classes Debug

- i_ptr = (int *) 0x22fe54
- i = 0

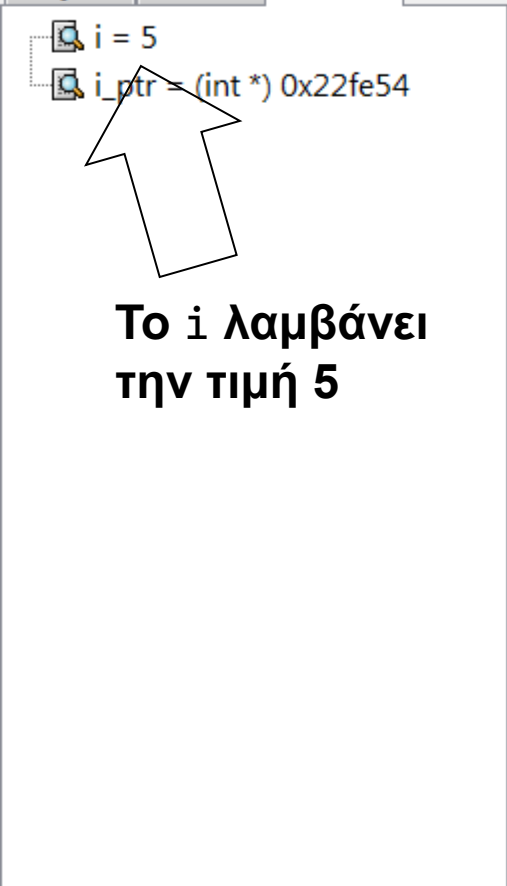


Ο δείκτης λαμβάνει ως τιμή τη θέση της i

```
1 #include <stdio.h>
2
3 int main() {
4     int i;
5     int *i_ptr;
6
7     i_ptr = &i;
8     i = 5;
9     *i_ptr = -6;
10    printf("%d\n", i);
11
12    return 0;
13 }
```

Debug Add watch Next line Next instruction Stop Execution Modify watch Into line Into instruction View CPU window Continue Skip function

Send command to GDB: next
->->prompt



Το i λαμβάνει την τιμή 5

```
1 #include <stdio.h>
2
3 int main() {
4     int i;
5     int *i_ptr;
6
7     i_ptr = &i;
8     i = 5;
9     *i_ptr = -6;
10    printf("%d\n", i);
11
12    return 0;
13 }
```

Debug Stop Execution

Send command to GDB: next

->->prompt



Project Classes Debug

i_ptr = (int *) 0x22fe54
i = -6

Το i λαμβάνει την τιμή 6 γιατί ο i_ptr δείχνει στη θέση του i

```

1  #include <stdio.h>
2
3  int main() {
4      int i;
5      int *i_ptr;
6
7      i_ptr = &i;
8      i = 5;
9      *i_ptr = -6;
10     printf("%d\n", i);
11
12     return 0;
13 }
```

main.c

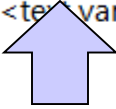
Debug Add watch Next line Next instruction Stop Execution Modify watch Into line Into instruction View CPU window Continue Skip function

Send command to GDB: next
->->prompt



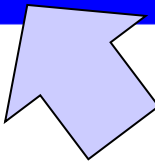


Project Classes Debug

- i = 0
- i_ptr = (int *) 0x4c21b0
- printf = {<type> variable, no del



Ο χώρος που διατίθεται από τη malloc δεν ανήκει στο stack

```
1 #include <stdio.h>
2
3 int main() {
4     int i;
5     int *i_ptr;
6
7     i_ptr = malloc (1 *sizeof(int));
8     i = 5;
9     *i_ptr = -6;
10    printf("%d\n", i);
11    printf("%d\n", *i_ptr);
12    return 0;
13 }
```



Χρησιμοποιώ malloc
Για να δεσμεύσω χώρο
Για ένα ακέραιο.

main.c

Compiler Resources Compile Log Debug Find Results Close

Debug Add watch Next line Next instruction Stop Execution Modify watch Into line Into instruction View CPU window Remove watch Continue Skip function

Send command to GDB: next

->->prompt

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 = $4 \times 2^{30} = 4 \text{ G}$

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.

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

int main(void) {

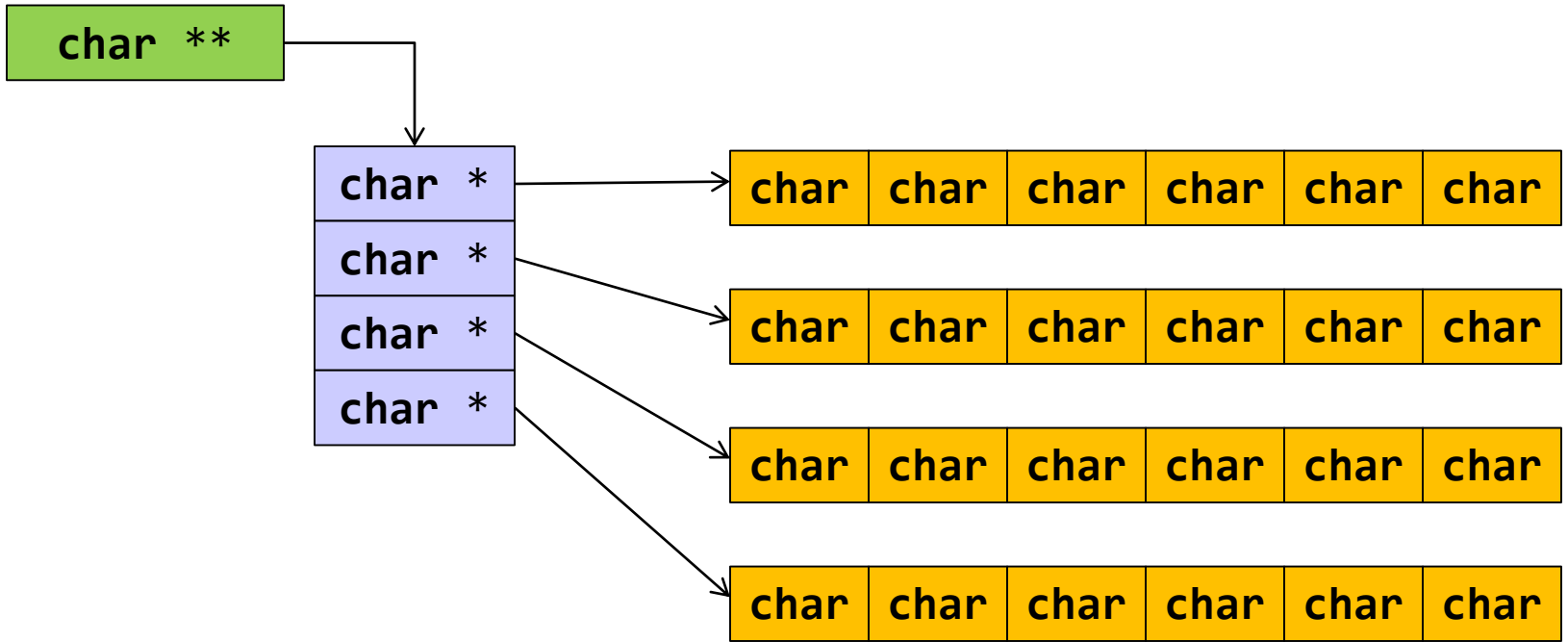
    int *datatable = NULL;
    int d;
    int numbers = 0;
    int i;

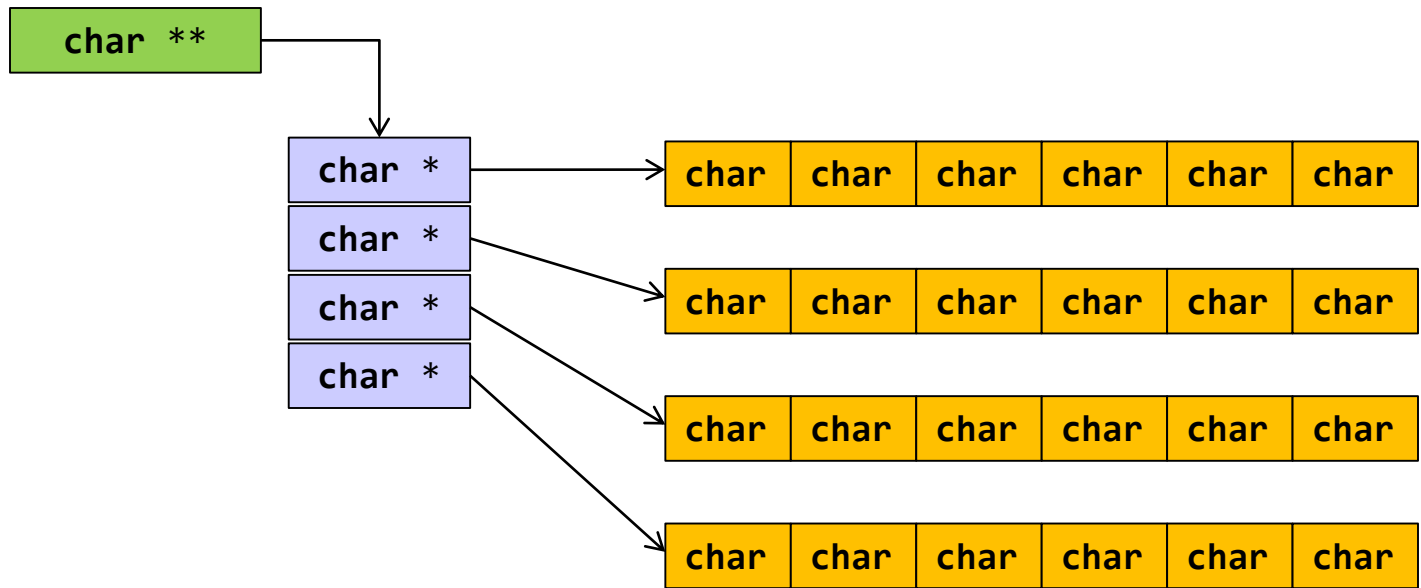
    while (scanf("%d", &d), d>0) {
        numbers ++;
        datatable = realloc(datatable, numbers*sizeof(int));
        datatable[numbers - 1] = d;
    }

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

    free(datatable);

    return EXIT_SUCCESS;
}
```





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

```

#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;
}

```

realloc

```
#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

```
#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 ηου μέλους> <όνομα ηου μέλους>;  
} <λίστα ονομάτων μεταβλητών> ;
```

Ορισμός τύπου **struct** address

```
struct address {  
    char street[20];  
    int number;  
    int code;  
    char city[20];  
};
```

Παράδειγμα α

```
struct  
    address {  
        char  
            street[20];  
        int number;  
        int code;  
        char city[20];  
    } myaddress;
```

συνδυασμένος
ορισμός τύπου
struct address
και δήλωση
μεταβλητής
τύπου **struct**
address

```
struct address  
myaddress ;
```

χρήση τύπου **struct** address

όνομα μεταβλητής τύπου
struct address

Αρχικοποίηση μεταβλητών

```
struct address {  
    char street[20];  
    int number;  
    int code;  
    char city[20];  
} ;
```

```
struct address myaddress = {"Anthewn", 1, 123,  
    "Patra" };
```

typedef

```
#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;
}
```

```
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 ;
}
```

Ένθεση Δομών

```
struct person {  
    char firstname[20];  
    char surname[20];  
    int age;  
    struct address homeaddress;  
    struct address bussinessaddress;  
} ;
```

Δείκτες ως μέλη δομών

```
struct person {  
    char firstname[20];  
    char surname[20];  
    int age;  
    struct address homeaddress;  
    struct address bussinessaddress;  
    struct person *next_person;  
} ;
```

Δείκτης σε δομή τύπου person

ΣΥΝΟΠΤΙΚΑ ΟΙ ΔΟΜΕΣ (1)

- Ορισμός δομής \Rightarrow ορισμός τύπου

```
struct test {  
    int a;  
    char d[10]; };  
struct test mytest;
```

- Επιστρέφονται και περνούν κατ' αξία από συναρτήσεις.

```
struct test dosomething(int a, struct test b) {  
    <κώδικας>  
}
```

- Βοηθάει το **typedef**

```
typedef struct test Test;  
Test dosomething(int a, Test b) {  
    <κώδικας>  
}
```

Συνοπτικά οι δομές (2)

- Αναφερόμαστε με dot notation σε μέλος μιας μεταβλητής τύπου δομής

```
struct test {  
    int a;  
    char d[10]; };
```

ΤΥΠΟΣ

Test mytest;

mytest.a = 5;

strcpy(mytest.d, "hello");

μεταβλητή

μέλος

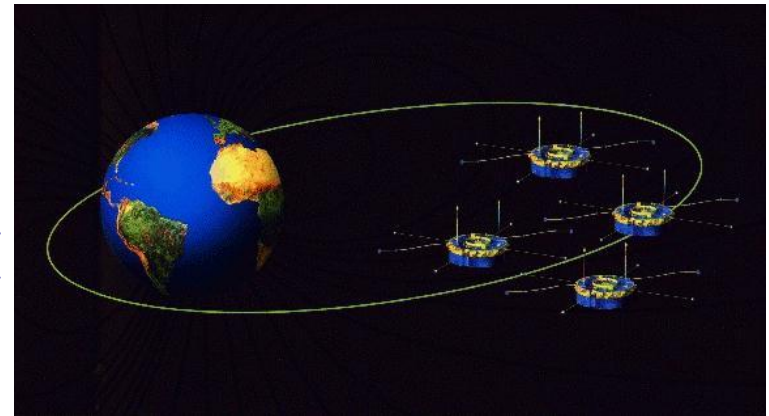
- Ένα μέλος μιας μεταβλητής τύπου μίας δομής έχει ως τύπο τον τύπο του μέλους.

Σύστημα τύπων:

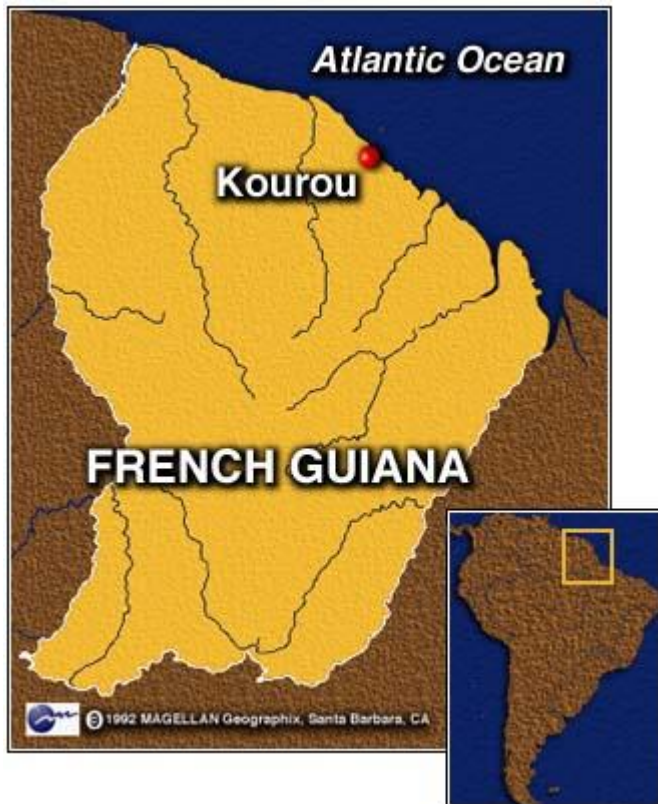
Ένα βήμα στην ανάπτυξη λογισμικού

- Καθορισμός συστήματος τύπων οι οποίοι χαρακτηρίζουν τα δεδομένα του προβλήματος
 - Επιβάλλει κανόνες στη χρήση των δεδομένων
 - Αυτοματοποιεί τον λογικό έλεγχο
- Για να αξιοποιήσουμε αυτή τη δυνατότητα
 - ορίζουμε κατάλληλους τύπους
 - γράφουμε κατάλληλο κώδικα

4 Ιουνίου 1996, Κουρού, Γαλλ. Γουιιάνα



Η πρώτη πτήση του Ariane 5G με φορτίο τέσσερεις δορυφόρους Cluster.

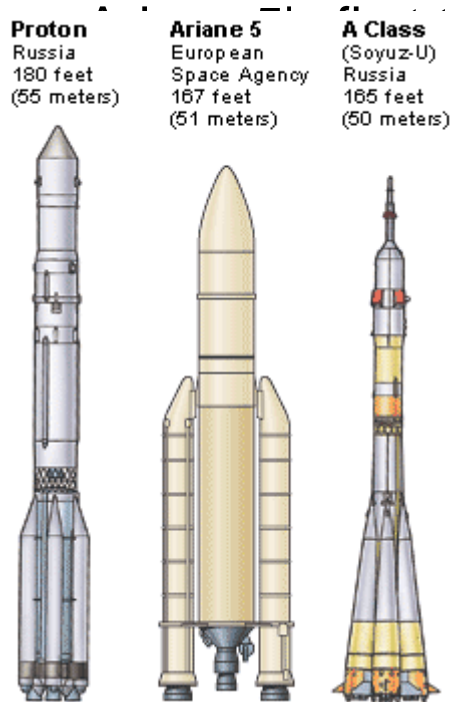


Ariane takes off Thursday
(Ariane TV)



Smoke from the explosion
June 4, 1996 (AP Photo)

Ariane 5 Flight 501



First flight ([Ariane 5 Flight 501](#)) on [4 June 1996](#) rocket self-destructing 37 seconds after launch malfunction in the control software, which was the most expensive [computer bugs](#) in history.

Conversion from 64-bit [floating point](#) to 16-bit [signed integer](#) caused a processor trap (operand error).

The constant number had a value too large to be a 16-bit signed integer.

These conversions had led to the disabling of the software handler (in [Ada](#) code) for this trap, although other conversions of comparable variables in the code remained protected.

Κώδικας σε ADA

```
...  
declare  
vertical_veloc_sensor: float;  
horizontal_veloc_sensor: float;  
vertical_veloc_bias: integer;  
horizontal_veloc_bias: integer;  
...  
begin  
declare  
pragma suppress(numeric_error, horizontal_veloc_bias);  
  begin  
    sensor_get(vertical_veloc_sensor);  
    sensor_get(horizontal_veloc_sensor);  
    vertical_veloc_bias := integer(vertical_veloc_sensor);  
    horizontal_veloc_bias := integer(horizontal_veloc_sensor);  
  ...  
  exception  
    when numeric_error => calculate_vertical_veloc();  
    when others => use_irs1();  
  end;  
end irs2;
```

Date & Time (UTC)	Flight	Model	Serial number	Payload	Result
04.06.1996 12:34:06	V-89	Ariane-5G	501	Cluster	Failure
30.10.1997 13:43:00	V-101	Ariane-5G	502	MaqSat H & TEAMSAT, MaqSat B, YES	Partial failure
21.10.1998 16:37:21	V-112	Ariane-5G	503	MaqSat 3, ARD	Success
10.12.1999 14:32:07	V-119	Ariane-5G	504	XMM-Newton	Success
21.03.2000 23:28:19	V-128	Ariane-5G	505	INSAT 3B , AsiaStar	Success
14.09.2000 22:54:07	V-130	Ariane-5G	506	Astra 2B, GE 7	Success
16.11.2000 01:07:07	V-135	Ariane-5G	507	PAS 1R, Amsat P3D , STRV 1C, STRV 1D	Success
20.12.2000 00:26:00	V-138	Ariane-5G	508	Astra 2D , GE 8 (Aurora 3), LDREX	Success
08.03.2001 22:51:00	V-140	Ariane-5G	509	Eurobird 1 , BSat 2a	Success
12.07.2001 22:58:00	V-142	Ariane-5G	510	Artemis, BSat 2b	Partial failure
01.03.2002 01:07:59	V-145	Ariane-5G	511	Envisat	Success
05.07.2002 23:22:00	V-153	Ariane-5G	512	Stellat 5, N-Star c	Success
28.08.2002 22:45:00	V-155	Ariane-5G	513	Atlantic Bird 1, MSG 1, MFD	Success
11.12.2002 22:22:00	V-157	Ariane-5ECA	517	Hot Bird 7, Stentor, MFD A, MFD B	Failure
09.04.2003 22:52:19	V-160	Ariane-5G	514	Insat 3A, Galaxy 12	Success
11.06.2003 22:38:15	V-161	Ariane-5G	515	Optus C1 , BSat 2c	Success
27.09.2003 23:14:46	V-162	Ariane-5G	516	Insat 3E, eBird 1, SMART-1	Success
02.03.2004 07:17:44	V-158	Ariane-5G+	518	Rosetta	Success
18.07.2004 00:44:00	V-163	Ariane-5G+	519	Anik F2	Success
18.12.2004 16:26:00	V-165	Ariane-5G+	520	Helios 2A, Essaim 1, 2, 3 and 4, PARASOL , Nanosat 01	Success
12.02.2005 21:03:00	V-164	Ariane-5ECA	521	XTAR-EUR, Maqsat B2, Sloshtat	Success
11.08.2005 08:20:00	V-166	Ariane-5GS	523	Thaicom 4-iPStar 1	Success
13.10.2005 22:32:00	V-168	Ariane-5GS	524	Syracuse 3A, Galaxy 15	Success
16.11.2005 23:46:00	V-167	Ariane-5ECA	522	Spaceway F2 , Telkom 2	Success
21.12.2005 22:33:00	V-169	Ariane-5GS	525	Insat 4A , MSG 2 , MFD C	Success
11.03.2006 22:32:50	V-170	Ariane-5ECA	527	Spainsat, MFD C, MFD C, Hot Bird 7A	Success
26.05.2006 21:08:50	V-171	Ariane-5ECA	529	Satmex 6, Thaicom 5	Success
11.08.2006 22:15:00	V-172	Ariane-5ECA	531	JCSat 10, Syracuse 3B	Success
13.10.2006 20:56:00	V-173	Ariane-5ECA	533	DirecTV-9S, Optus D1 , LDREX-2	Success
08.12.2006 22:08:00	V-174	Ariane-5ECA	534	WildBlue 1 , AMC 18	Success
11.03.2007 22:03	V-175	Ariane-5ECA	535	Skynet-5A, Insat-4B	Success
04.05.2007 22:29	V-176	Ariane-5ECA	536	Astra 1L, Galaxy 17	Success

Τύποι Δεδομένων

- Καθορίζουν σύνολο ιδιοτήτων μεταβλητών
 - πχ. το άθροισμα το ακεραίων είναι ακέραιος
- Αξιοποιούνται από ένα μηχανισμό ελέγχου
 - Τι γίνεται αν συνδυάσω σε μια έκφραση δεδομένα διαφορετικού τύπου;
 - Είναι σωστές οι διεπαφές των δομικών στοιχείων;
 - Βοηθάει στην αποφυγή λογικών λαθών στον κώδικα.