Διαδικαστικός Προγραμματισμός

Βασίλης Παλιουράς

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;
}
```

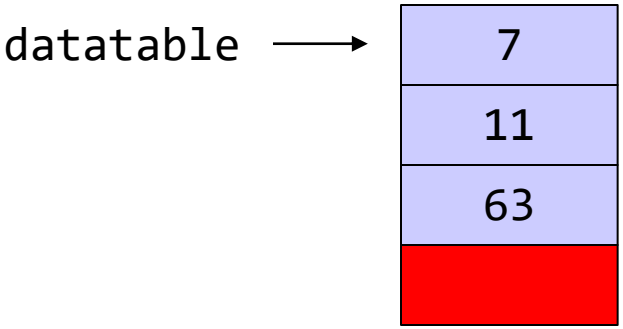
Αρχικοποίηση σε NULL

Αυξανόμενο μέγεθος πίνακα στο heap

Διεύθυνση πρώτης θέσης του νέου block

Το d καταχωρείται στην τελευταία θέση του block


datatable
NULL

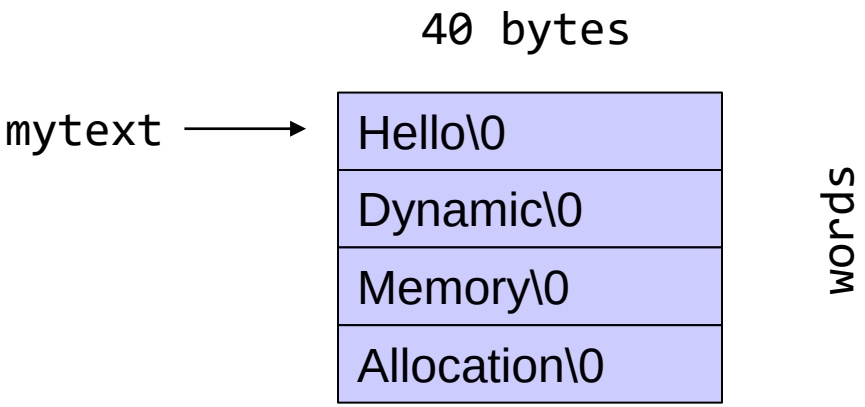

datatable
7
11
63


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;
}
```

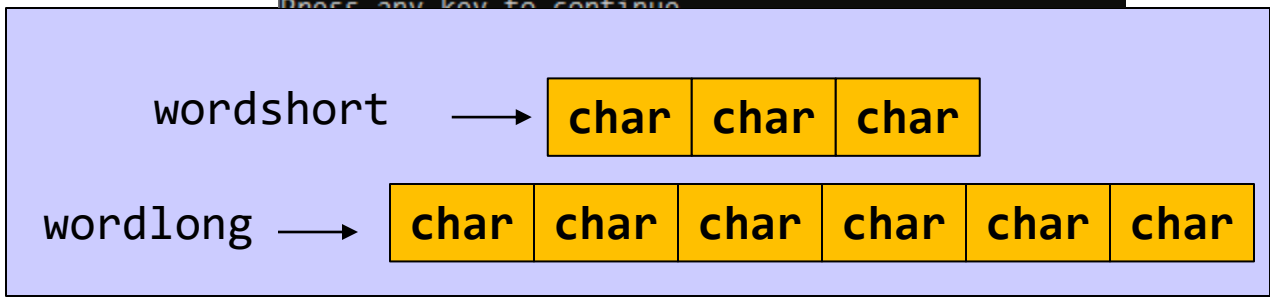
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 40

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

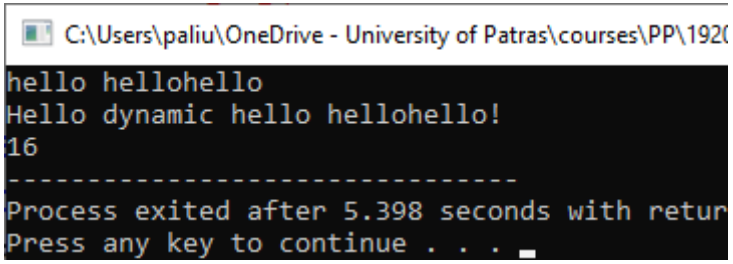

```c
#include <stdio.h>
#include <stdlib.h>
#define N 40

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;

}
```

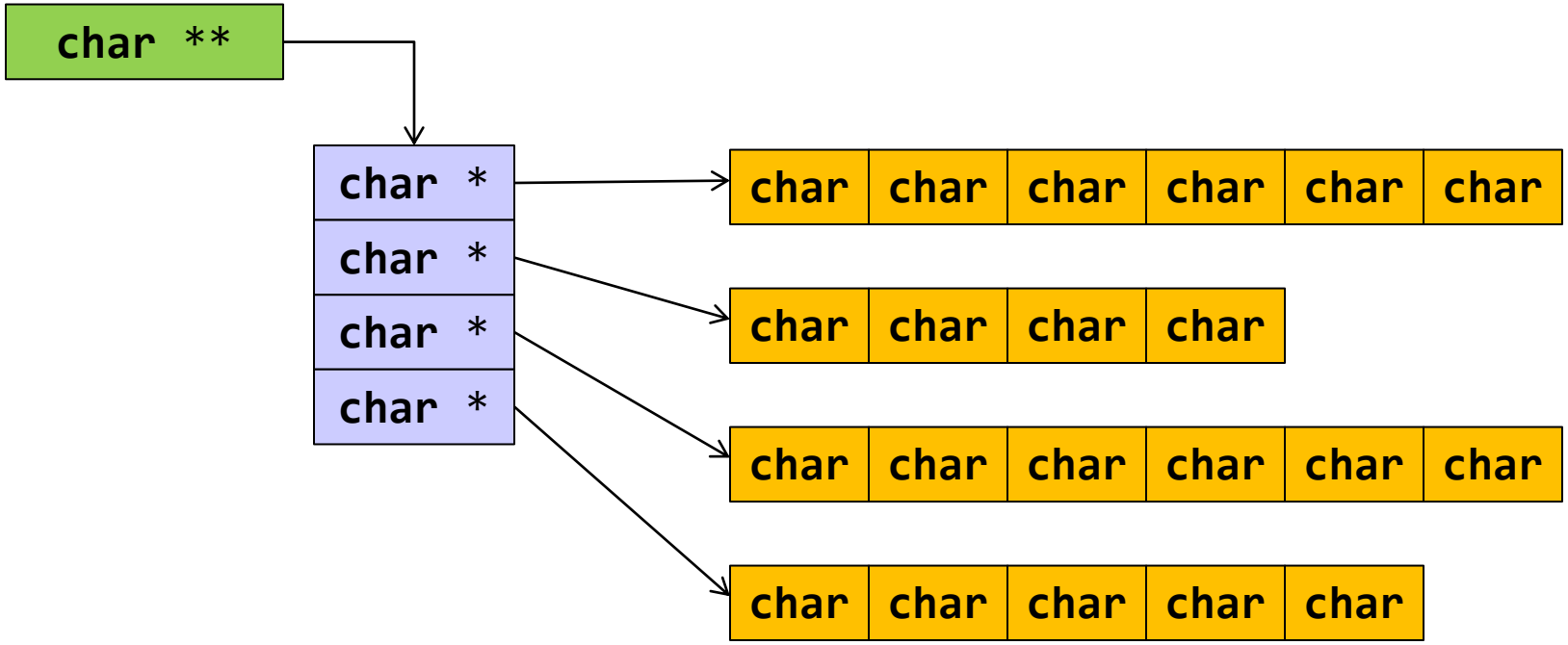
```
Select C:\Users\paliu\OneDrive - University of Patras\courses\PP\1920\rem
hello
hellohellohellohello
Hello dynamic hello hellohellohellohello!
8 8
--------------------------------
Process exited after 7.312 seconds with return value
Press any key to continue
```

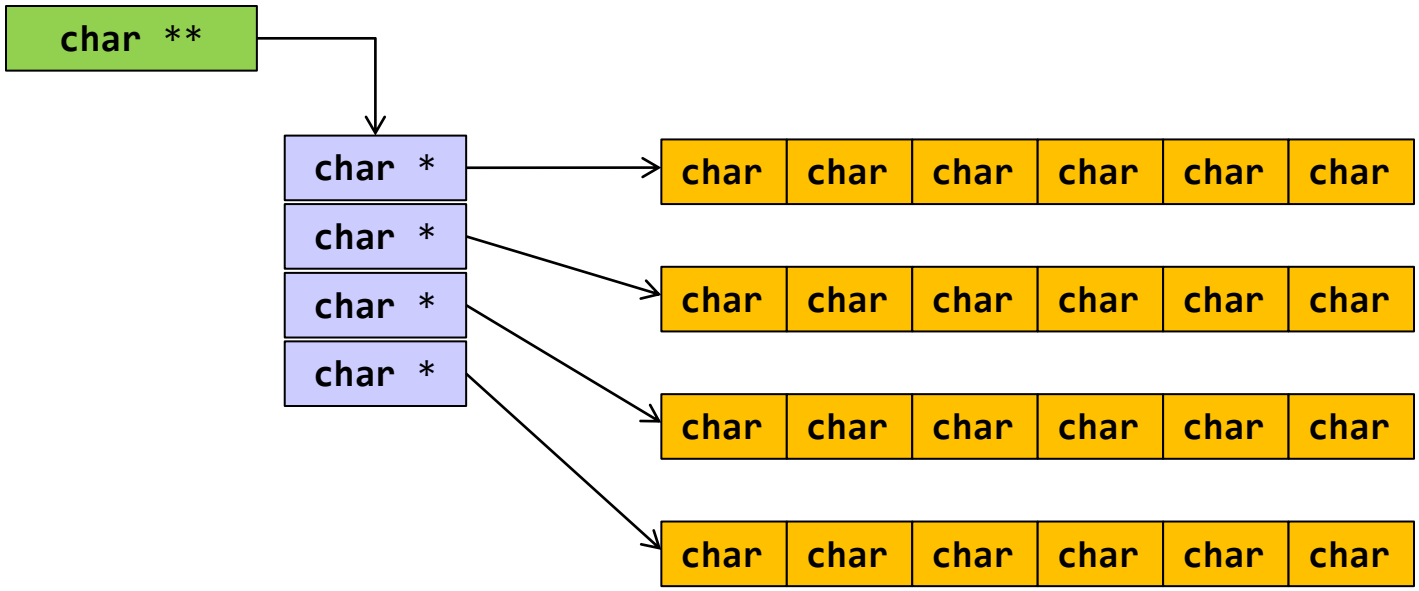


```c
#include <stdio.h>
#include <stdlib.h>

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;
}
```

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> ;
```

Ορισμός τύπου **struct** address

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

Παράδειγμα

```
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 ;
```

όνομα μεταβλητής τύπου **struct** address

χρήση 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

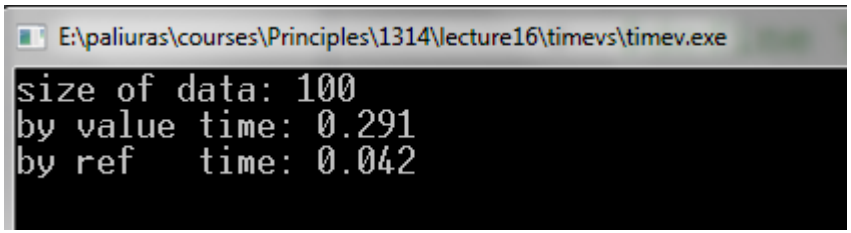
```c
#include <stdio.h>
#include <string.h>

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 0;
}
```

```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 ;
}
```

Τι είναι ταχύτερο;

```c
#include <stdio.h>
#include <time.h>
#define TIMES 10000000

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

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

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

```c
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;
}
```



Κλήση με αξία και
Κλήση με αναφορά

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;
} ;
```

Συνοπτικά οι δομές (1)

- Ορισμός δομής ⇒ 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]; };
Test mytest;
mytest.a = 5;
strcpy(mytest.d,"hello");
```

μεταβλητή

μέλος

τύπος

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

```c
#include <stdio.h>
#include <stdlib.h>
struct data {
          int * data_ptr ;
          int len;
};
typedef struct data Data;
Data createdata(void);
void printdata(Data);

int main ( void ) {

  Data mydata;

  mydata = createdata();
  printdata(mydata);
  /* how to free ?*/
  return 0;
}
```

```c
Data createdata (void) {
          Data data ;
          int x;
          int i;
          scanf("%d", &x);
          data.data_ptr = malloc(x*sizeof (int));
          data.len = x;
          for (i=0;i<x;i++) data.data_ptr[i] = 2*i;
          return data;
}

void printdata(Data data) {
 int i;
 for (i=0;i<data.len;i++) printf("%d\n", data.data_ptr[i]);
 return ;
}
```

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

int main (void ) {
int i;
int num;
FILE *data ;

        if ((data = fopen ("dataout.txt", "wt") )==NULL) {
                printf("Cannot create file.\n");
                exit(1);
                }
        for ( i = 0; i < 10 ; i++)
                fprintf (data, "number i: %d\n", i);
        fclose(data);

        if((data = fopen ("numbers.txt", "rt"))==NULL) {
                printf("Cannot read from file.\n");
                exit(1);
                };
        while (fscanf(data, "%d", &num)!=EOF)
                printf("%d ", num);
        fclose(data);
}
```

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

int main(void)
{
  int n;
  FILE * pFile;
  char buffer [27];

  pFile = fopen ("test.txt","w+");

  for ( n='A' ; n<= 'Z'; n++)
    fputc ( n, pFile);

  rewind (pFile);
  fread (buffer,1,26,pFile);
  fclose (pFile);
  buffer[26]='\0';
  printf("%s\n", buffer);

  return 0;
}
```

fwrite ( )

```
size_t fwrite ( const void * ptr, size_t size, size_t count, FILE *stream );
```

- Γράφει περιοχή μνήμης στη ροή stream
- Παράμετροι
  - ptr Δείκτης σε πίνακα στοιχείων προς εγγραφή.
  - size μέγεθος κάθε στοιχείου σε bytes
  - count πλήθος στοιχείων καθένα μεγέθους *size* bytes.
  - stream Δείκτης σε FILE

```
size_t fread ( void * ptr, size_t size, size_t count, FILE * stream );
```

- Παράμετροι
  - Ptr Δείκτης σε μπλοκ μνήμης ελάχιστου μεγέθους (*size***count*) bytes.
  - Size Μέγεθος σε bytes κάθε στοιχείου προς ανάγνωση.
  - Count Πλήθος στοιχείων καθένα μεγέθους *size* bytes.
  - Stream Δείκτης σε FILE

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

typedef struct data {
        int a;
        char name[10];
        } Data;

int main(void) {

Data mydata[] = {{5, "red"}, {10, "black"}, {11,"yellow"}};
Data mycopy[10];
FILE *testio;
int i;

testio = fopen ("mydata.bin", "wb");
fwrite (mydata, sizeof (Data), 3, testio);
fclose(testio);

testio = fopen ("mydata.bin", "rb");
fread(mycopy, sizeof (Data), 3, testio);
fclose(testio);

for (i=0;i<3;i++)
    printf("value: %d\nname: %s\n", mycopy[i].a, mycopy[i].name);

return 0;
}
```

- `int fseek ( FILE * stream, long int offset, int origin );`

- Παράμετροι
  - Δείκτης σε FILE .
  - Offset πλήθος bytes για πρόσθεση στο *origin*.
  - Origin τρέχουσα θέση στην οποία προστίθεται το *offset*.
  - Μπορεί να είναι μια από τις σταθερές:
    - SEEK_SET Αρχή αρχείου
    - SEEK_CUR Τρέχουσα θέση
    - SEEK_END Τέλος αρχείου

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

typedef struct data{
        int a;
        char name[10];
        } Data;

int main(void)
{
  FILE * testio;
  long filesize;
  Data * mycopy;
  size_t result;
  int numberofelems;
  int i;

  testio = fopen ( "mydata.bin" , "rb" );
  if (testio == NULL)
     printf("failed to open\n");
  else
      printf("stream open\n");

  fseek(testio, 0 , SEEK_END);
  filesize = ftell (testio);
  rewind (testio);

  mycopy = (Data *) malloc (sizeof (char)*filesize);

  result = fread(mycopy,1,filesize,testio);
  if (result != filesize)
    {
          printf("Reading error");
          exit(1);
    }

  fclose (testio);

  numberofelems=(filesize)/(sizeof(Data));
  printf("Data elements read: %d\n", numberofelems);

  for (i=0;i<numberofelems;i++)
    printf( "value: %d\nname: %s\n",
          mycopy[i].a, mycopy[i].name);


  free (mycopy);

  return 0;
}
```