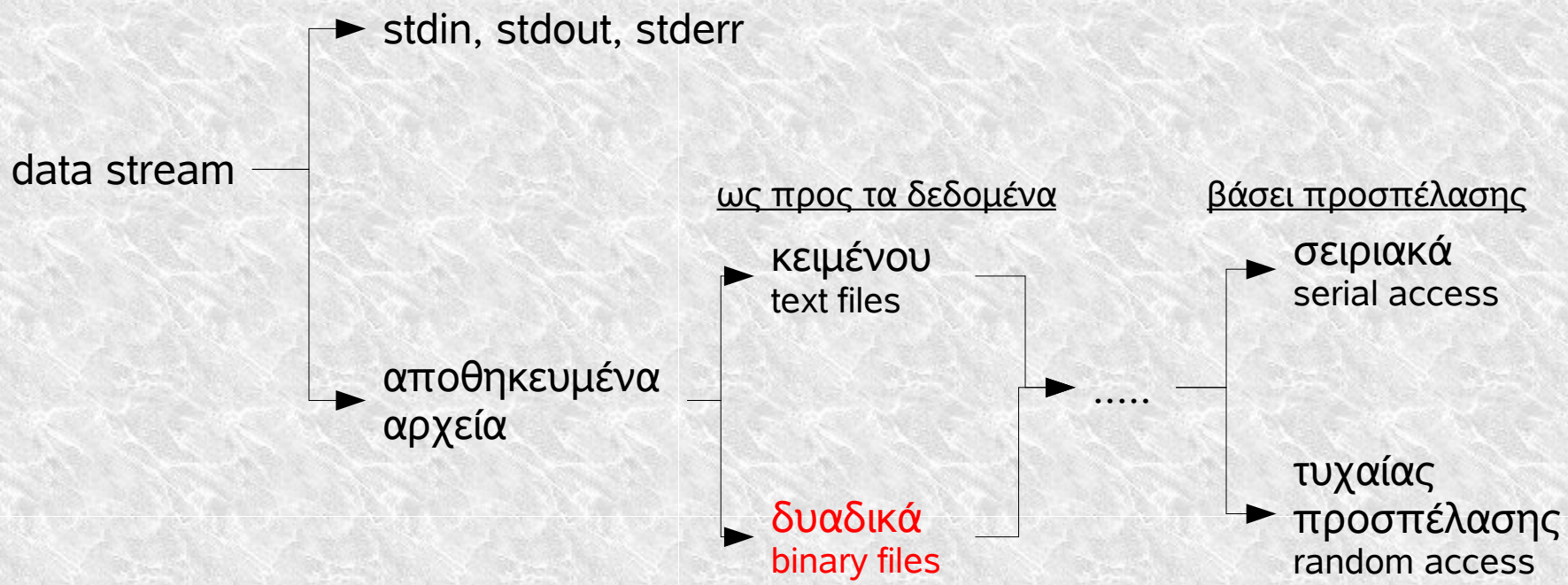
I
Αρχεία δεδομένων, μέρος δεύτερο: δυαδικά αρχεία
II
Δομές δεδομένων (struct)

Διδάσκων: Β. Ράπτης

Αρχεία

Το γενικό πλαίσιο: data stream (ρεύμα δεδομένων).



Αρχεία κειμένου: δεδομένα τύπου char
Δυαδικά: δεδομένα όπως ακριβώς αναπαρίστανται στη μνήμη του ΗΥ
Τρόπος προσπέλασης:

Δυαδικά Αρχεία

Έστω αρχείο κειμένου με τα παρακάτω περιεχόμενα:

```
123     255       0
209     100     107
 91     200     167
...
```

Άρα:
10 bytes ανά αριθμό

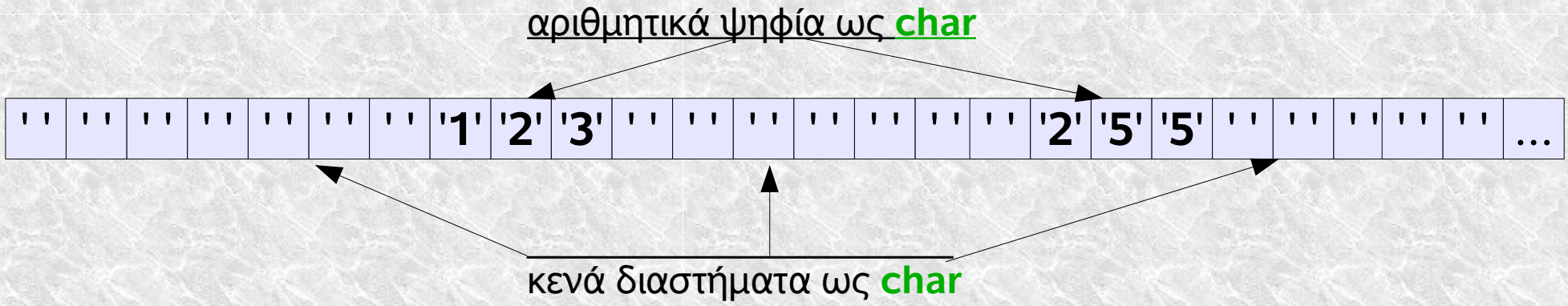
3 X 10 + 1 = 31 bytes
ανά εγγραφή (το 1 byte για '\n')

Πιθανός τρόπος δημιουργίας:

```
int n1, n2, n3;
FILE *f;
...
fprintf(f, "%10d%10d%10d\n", n1, n2, n3);
```

← αποθηκεύει ως char, ανεξαρτήτως τύπου!

Αποθήκευση στη μνήμη:



Δυαδικά Αρχεία

Τι μνήμη καταλαμβάνουν τα ίδια δεδομένα ως int;

123	255	0
209	100	107
91	200	167
...		

int = 4 bytes

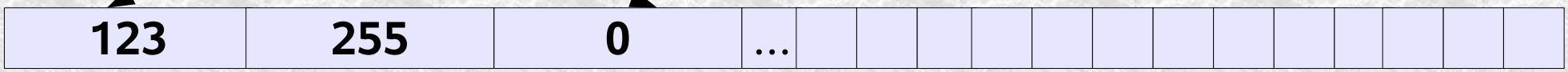
12 bytes = ανά εγγραφή
(+1 για '\n', αλλά όχι απαραίτητα)

Και πώς γράφονται/διαβάζονται;

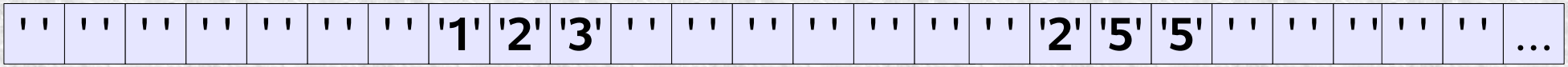
Απάντηση:
fwrite(void *, size_t, int, FILE *);
fread (void *, size_t, int, FILE *);

Αποθήκευση στη μνήμη:

αριθμοί ως **int**



Σύγκρινε:



Δυαδικά Αρχεία – Δημιουργία

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

int main(int argc, char **argv)
{
    int n = 0, iarr[3];
    char filename[20];
    FILE *fp, *gp;

    strcpy(filename, argv[1]);
    fp = fopen(filename, "r");
    gp = fopen("txt.bin", "wb");
    while(fscanf(fp, "%10d%10d%10d\n", iarr,iarr+1,iarr+2) > 0) {
        fwrite ( iarr, sizeof ( iarr[0] ), 3, gp );
        n++;
    }
    printf("number of records read: %d\n", n);
    fclose(gp);
    fclose(fp);

    return 0;
}
```

Εκτέλεση

```
% cc bin.c
% ./a.out test.txt
number of records read: 4
% more txt.bin
{[][]dk[][][][][]
%
```

Δυαδικά Αρχεία – Προσπέλαση

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

int main(int argc, char **argv)
{
    int n, iarr[3];
    FILE *fp;

    n = atoi(argv[1]);
    fp = fopen("txt.bin", "r");
    while(n-- > 0) {
        fread ( iarr, sizeof ( iarr[0] ), 3, fp );
        printf( "%10d%10d%10d\n", iarr[0], iarr[1], iarr[2]);
    }
    fclose(fp);

    return 0;
}
```

Εκτέλεση

```
% cc read.c
% ./a.out 4
       123       255         0
       209       100       107
        91       200       167
    567199       176       224
%
```

Σύγκριση

```
% ls -l test.txt txt.bin
-rw-r--r-- 1 brapt users 124 2008-12-01 15:51 test.txt
-rw-r--r-- 1 brapt users  48 2008-12-01 18:35 txt.bin
%
```

Δυαδικά Αρχεία – Προσπέλαση (2)

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

int main(int argc, char **argv)
{
    int iarr[3];
    FILE *fp;

    fp = fopen("txt.bin", "r");
    while( !feof(fp) ) {
        fread ( iarr, sizeof ( iarr[0] ), 3, fp );
        printf( "%10d%10d%10d\n", iarr[0], iarr[1], iarr[2]);
    }
    fclose(fp);

    return 0;
}
```

Εκτέλεση

```
% cc read.c
% ./a.out 4
       123       255         0
       209       100       107
        91       200       167
    567199       176       224
    567199       176       224
%
```

feof αντί EOF

Συνήθως, EOF = -1, δηλαδή αποδεκτή τιμή για δυαδικά αρχεία!

Σύγκριση

```
% ls -l test.txt txt.bin
-rw-r--r-- 1 brapt users 124 2008-12-01 15:51 test.txt
-rw-r--r-- 1 brapt users  48 2008-12-01 18:35 txt.bin
%
```

Δυαδικά Αρχεία

Συνοψίζοντας...

Δυαδικά αρχεία : περιέχουν δεδομένα *όπως ακριβώς* παριστάνονται στη μνήμη (όχι ως χαρακτήρες)

Δημιουργία και προσπέλαση: fwrite και fread, αντίστοιχα

Μπορεί να καταλαμβάνουν λιγότερη μνήμη από αντίστοιχα κειμένου

Αλλά, πρέπει να ξέρουμε ακριβώς τη δομή τους για να τα διαβάσουμε πάλι!

Έλεγχος τερματισμού: **feof**(FILE *) καλύτερα αντί για **EOF**

Δομές (struct)

Πίνακες: πολλές μεταβλητές *ίδιου* τύπου

Ερώτηση: ομαδοποίηση ***διαφορετικών*** τύπων;

Π.χ. οντότητα "Φοιτητής" με εξής ιδιότητες:

Αριθμός μητρώου (int)
Ονοματεπώνυμο (char)
Εξάμηνο (int)
Μέσος όρος βαθμών εξαμήνου (float)

Οντότητα "Πόλη" με ιδιότητες:

Όνομα πόλης (char)
Πληθυσμός της πόλης (int)
κλπ

Δομές (**struct**): ομαδοποίηση μεταβλητών *διαφορετικού* τύπου

Επιτρέπουν να ορίσουμε νέους τύπους μεταβλητών

Δομές (struct)

Η οντότητα “φοιτητής” ως δομή

```
struct Student
{
    int AM;
    char *name;
    char *fname;
    int semester;
    float grade;
};
```

Ορίζουμε μία δομή με όνομα student
και μέλη:
AM, τύπου int, για τον αριθμό μητρώου
name, τύπου char, για το όνομα
name, τύπου char, για το επώνυμο
semester, τύπου int, για το εξάμηνο
grade, τύπου float, για το μέσο όρο των βαθμών
τελειώνουμε με ;

Χρήση:

```
struct Student someone;
someone.AM = 25;
someone.name = "Tade";
someone.fname = "Tadopoulos";
someone.semester = 1;
someone.grade = 8.3;
...
struct Student *sptr;
sptr = &someone;
sptr->AM = 30;
```

struct Student = νέος (παράγωγος) τύπος
μπορώ να ορίσω νέες μεταβλητές
χρήση τελεστή μέλους “τελεία” (δομή.μέλος)

μπορώ να ορίσω και δείκτες σε δομές

τελεστής μέλους “βέλος” για δείκτες σε δομές.

Δομές (struct)

Οι παράγωγοι τύποι (δομές, struct) μπορούν να χρησιμοποιηθούν όπως κάθε τύπος.

Π.χ. πίνακες

```
struct Student pliroforiki[100];
...
for(i = 0; i < 100; i++) {
        printf("\nΑριθμός μητρώου : "); scanf("%d", &pliroforiki[i].AM);
        printf("\n;Όνομα              : "); scanf("%s", pliroforiki[i].name); getchar();
        printf("\nΕπώνυμο            : "); scanf("%s", pliroforiki[i].fname); getchar();
        printf("\nΕξάμηνο            : ");  scanf("%d", &pliroforiki[i].semester);
        printf("\nΜέσος όρος         : ");  scanf("%f", &pliroforiki[i].grade);
}
...
for(i = 0; i < 100; i++) {
        printf("\nΑριθμός μητρώου : %d", pliroforiki[i].AM);
        printf("\n;Όνομα              : %s", pliroforiki[i].name);
        printf("\nΕπώνυμο            : %s", pliroforiki[i].fname);
        printf("\nΕξάμηνο            : %d", pliroforiki[i].semester);
        printf("\nΜέσος όρος         : "%f", pliroforiki[i].grade);
}
```