Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #9

1 & 2 Ιουνίου 2023

Παναγιώτης Παύλου

c-programming-23@allos.gr

Αρχεία

Τα αρχεία ως πόροι – Μοντέλο περιεχομένων – Ειδικά αρχεία

Μοντελοποίηση περιεχομένων 1/2

Τα αρχεία αποθηκεύονται σε μνήμη μακροπρόθεσμης αποθήκευσης (π.χ. σε δίσκο) και είναι ακολουθίες από bytes, κατ'αναλογία με τη μνήμη RAM. Τα bytes είναι και πάλι αριθμημένα, αλλά για κάθε αρχείο ξεχωριστά, οπότε η αρίθμηση του πρώτου byte είναι πάντα μηδέν. Δηλαδή στην αρίθμηση μοιάζουν περισσότερο με τους δείκτες των πινάκων.

Στα αρχεία μεταφέρονται περιεχόμενα από και προς τη μνήμη.

Μοντελοποίηση περιεχομένων 2/2

Παρόλα αυτά διαφέρουν ως προς τον τρόπο της μετακίνησης των δεδομένων από και προς αυτά. Κάθε φορά που ξεκινά η χρήση ενός αρχείου (με το άνοιγμα) δημιουργείται γι' αυτή τη χρήση ένας **δρομέας**. Ο δρομέας αυτός υποδεικνύει το σημείο του αρχείου στο οποίο θα γίνει η **επόμενη** ανάγνωση δεδομένων από το αρχείο ή εγγραφή δεδομένων στο αρχείο. Η αρχική θέση του δρομέα συνήθως, αλλά όχι πάντα, είναι η μηδενική.

Τα αρχεία ως resources 1/4

Τα αρχεία λειτουργούν ως resources (πόροι), δηλαδή για να χρησιμοποιηθούν θα πρέπει, να ανοιχθεί (**open**) πρώτα και με το αποτέλεσμα του ανοίγματος να γίνει η χρήση του. Στο τέλος της χρήσης του να κλειστεί (**close**). Οι εντολές που σχετίζονται με τα αρχεία ξεκινούν με το πρόθεμα f. Όλες περιλαμβάνονται στο stdio.h

Το άνοιγμα γίνεται με την εντολή fopen που συντάσσεται ως εξής:

```
FILE *fopen(char *filename , char *mode)
```

Αυτή είτε επιστρέφει έναν pointer σε μία δομή `FILE`, είτε – ως ένδειξη λάθους – την τιμή `NULL`.

Η πρώτη παράμετρος είναι το όνομα του αρχείου (αν και στην πραγματικότητα είναι η διαδρομή – το path – του αρχείου) ενώ η δεύτερη είναι ένα κείμενο με ένα έως τρεις χαρακτήρες.

Τα αρχεία ως resources 2/4

Το filename είναι η διαδρομή, δηλαδή το όνομα του αρχείου μαζί με τον φάκελο στον οποίο βρίσκεται το αρχείο, και χωρίζονται με τον χαρακτήρα `/` ή τον `\`. Προσοχή όμως το `\` είναι ο ειδικός (escaping) χαρακτήρας μέσα στα εισαγωγικά (μονά ή διπλά), έτσι πρέπει να γράφεται ως `\\`.

Οι διαδρομές χωρίζονται σε απόλυτες (οι οποίες ξεκινούν και περιγράφουν τη θέση του αρχείου σε σχέση με τον δίσκο) και σχετικές (οι οποίες ξεκινούν από τον «τρέχοντα φάκελο» και ορίζουν τη θέση του αρχείου).

```
"a-file-name.txt"
"..\\some-folder\\other-file.txt"
"\\outter-folder\\inner-folder\\the-file.txt"
```

Σχηματικό παράδειγμα path

Τα αρχεία ως resources 3/4

Το mode παριστάνει τον σκοπό για τον οποίο ανοίγει το αρχείο βάσει του παρακάτω πίνακα:

mode	Χρήση του αρχείου	Το αρχείο υπάρχει ήδη; Ναι	Όχι	Περιεχόμενο
`r`	Ανάγνωση (read)	✓	Προκύπτει σφάλμα	Κείμενο
`w`	Εγγραφή (write)	Διαγράφονται τα περιεχόμενα	Δημιουργείται	Κείμενο
`a`	Προσάρτηση (append)	Προστίθενται στα υπάρχοντα	Δημιουργείται	Κείμενο
`r+`	Εγγραφή και ανάγνωση	✓	Προκύπτει σφάλμα	Κείμενο
`w+`	Εγγραφή και ανάγνωση	✓	✓	Κείμενο
`a+`	Προσάρτηση και ανάγνωση	✓	✓	Κείμενο

`b` *ή binary*

Τα αρχεία ως resources 4/4

Το κλείσιμο ενός αρχείου γίνεται με την εντολή `fclose` που συντάσσεται ως εξής:

```
int fclose( FILE *handle )
```

Αυτή είτε επιστρέφει 0 εάν όλα είναι εντάξει, είτε μια ειδική τιμή την **`EOF`** που θα συναντήσουμε και αργότερα εάν υπάρχει πρόβλημα. Πρακτικά όμως σπάνια έχει νόημα να γίνει έλεγχος αυτής της τιμής.

Stdin/Stdout/Stderr

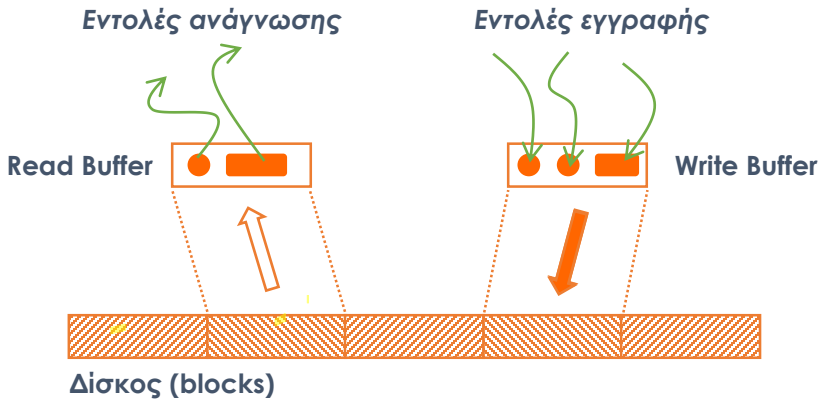
Εκτός από τα αρχεία που μπορεί να ανοίξουν με την `fopen` από τον κώδικα, υπάρχουν και τρία ειδικά αρχεία τα οποία είναι ανοιχτά κατά την έναρξη του κώδικα. Αυτά είναι:

`stdin`	το αρχείο αυτό αντιπροσωπεύει ό,τι δίνεται ως είσοδος στο πρόγραμμα, από το πληκτρολόγιο
`stdout`	το αρχείο αυτό αντιπροσωπεύει ότι δίνει ως έξοδο το πρόγραμμα στην οθόνη αλλά ως αποτέλεσμα της φυσιολογικής λειτουργίας του κώδικα
`stderr`	το αρχείο αυτό αντιπροσωπεύει πάλι την έξοδο από τον κώδικα προς την οθόνη, αλλά μόνο τα μηνύματα λάθους προς τον χρήστη

Τα λειτουργικά συστήματα παρέχουν τρόπους όπου τα αρχεία αυτά μπορούν να ανακατευθυνθούν σε άλλες πηγές (`stdin`) ή έξοδος (`stdout`,`stderr`), αλλά αυτό δεν διαφοροποιεί τον κώδικα που γράφεται. Είναι απλά μία δυνατότητα που δίνει το λειτουργικό σύστημα στον χρήστη που εκτελεί ένα πρόγραμμα.

Είσοδος/Έξοδος με buffer

Οι δίσκοι ανήκουν στα block storage devices, δηλαδή σε συσκευές αποθήκευσης όπου η πληροφορία (από/προς) μετακινείται σε τμήματα πληροφορίας προκαθορισμένου μεγέθους (τα blocks), τυπικά μεγέθους μερικών kB.



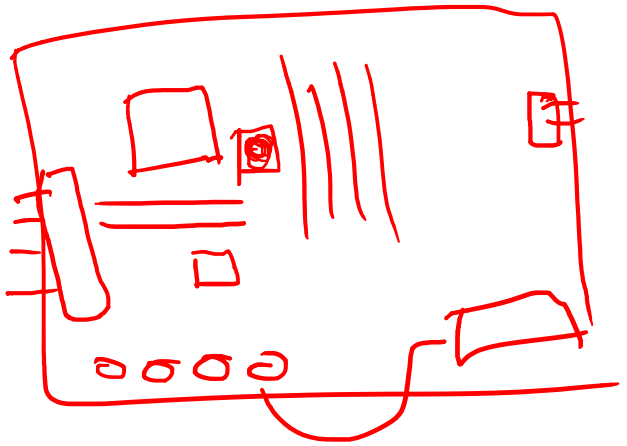
Αυτό δεν σημαίνει ότι ο κώδικάς μας πρέπει να διαβάζει όμως πληροφορίες ανά block. Αυτή η αποσύζευξη επιτυγχάνεται με τη χρήση ενδιάμεσης μνήμης από το σύστημα, από την οποία ο κώδικας μας διαβάζει ή γράφει και όταν συμπληρωθεί το block, τότε αυτό συγχρονίζεται με τον δίσκο.

Αυτό δημιουργεί πρόβλημα ενίοτε, καθώς τα περιεχόμενα ενός αρχείου μπορεί να μην ευσταθούν, οπότε (όταν χρειάζεται) θα πρέπει να μπορεί ο κώδικάς να ζητά την άμεση μεταφορά των περιεχομένων του buffer προς τον δίσκο.

Αυτό γίνεται με την εντολή:

```
int fflush( FILE *handle )
```

η οποία επιστρέφει αποτέλεσμα όπως και η `fclose`



Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-23@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Τονίζουμε : Μην στείλετε **ποτέ κώδικα ως εικόνα**, είναι παντελώς άχρηστος!

Αρχεία Κειμένου

Ανάγνωση και εγγραφή κειμένου από και προς αρχεία κειμένου

Μορφοποιημένη Έξοδος

Η μορφοποιημένη έξοδος προς κάποιο αρχείο γίνεται με την εντολή `fprintf` η οποία λειτουργεί ακριβώς όπως η `printf`, αλλά με ένα πρόσθετο όρισμα στην αρχή: το αρχείο στο οποίο θα γραφτεί το αποτέλεσμα (αντί για την οθόνη).

```
int fprintf( FILE *stream, char *format, ... )
```

Παρατηρήστε ότι η `fprintf(stdout, ...)` ταυτίζεται με την `printf(...)`.

Επίσης – αν και εκτός θέματος – μία άλλη παραλλαγή τους είναι η εντολή:

```
int sprintf( char *string, char *format, ... )
```

στην οποία, αντί το κείμενο που παράγεται να οδηγείται σε αρχείο, οδηγείται στη μνήμη που υποδεικνύει ο pointer `string` . Η δεσμευμένη μνήμη εκεί θα πρέπει να είναι αρκετά μεγάλη ώστε να χωράει το αποτέλεσμα.

Μορφοποιημένη Είσοδος

Η μορφοποιημένη είοδος προς κάποιο αρχείο γίνεται με την εντολή `fscanf` η οποία λειτουργεί ακριβώς όπως η `scanf`, αλλά με ένα πρόσθετο όρισμα στην αρχή: το αρχείο από το οποίο θα διαβαστούν τα δεδομένα (αντί για του πληκτρολογίου).

```
int fscanf( FILE *stream, char *format, ... )
```

Παρατηρήστε ότι η `fscanf(stdin, ...)` ταυτίζεται με την `scanf(...)`.

Επίσης – αν και εκτός θέματος – μία άλλη παραλλαγή τους είναι η εντολή:

```
int sscanf( char *string, const char *format, ... )
```

στην οποία αντί το κείμενο που περιέχει τα δεδομένα να προέρχεται από το αρχείο, προέρχεται από τη μνήμη που υποδεικνύει ο pointer `string`.

Είσοδος/Έξοδος ανά γραμμή

Στα αρχεία κειμένου, εφόσον έχουν ανοιχτεί σε κατάλληλο mode, μπορούν να γραφούν και να διαβαστούν ολόκληρες γραμμές με τις δύο παρακάτω εντολές:

```
char *fgets(char *str, int n, FILE *stream)
```

Αυτή διαβάζει μία ολόκληρη γραμμή με το πολύ `n` χαρακτήρες, μαζί με τον χαρακτήρα αλλαγής γραμμής. Επιστρέφει `NULL` είτε σε περίπτωση λάθους, είτε αν ολοκληρωθεί η ανάγνωση ενός αρχείου. Αλλιώς επιστρέφει την τιμή του `str`. Το κείμενο της γραμμής τοποθετείται στη μνήμη που υποδεικνύει ο pointer `str`, που θα πρέπει να έχει δεσμευμένα τουλάχιστον `n` bytes.

```
int fputs(char *str, FILE *stream)
```

Αυτή γράφει το κείμενο str στο αρχείο, και προσθέτει στο τέλος τον χαρακτήρα αλλαγής γραμμής. Σε περίπτωση λάθους, επιστρέφει την τιμή `EOF`.

Είσοδος/Έξοδος ανά χαρακτήρα 1/2

Στα αρχεία κειμένου, εφόσον έχουν ανοιχτεί σε κατάλληλο mode, μπορούν να γραφούν και να διαβαστούν μεμονωμένοι χαρακτήρες με τις ακόλουθες εντολές:

```
int fgetc(FILE *stream)
```

η οποία επιστρέφει τον επόμενο χαρακτήρα του αρχείου ή `EOF` εάν ο δρομέας έχει φτάσει στο τέλος του αρχείου. Η τιμή που επιστρέφεται είναι `int` επειδή εκτός από τους 256 κωδικούς ενός χαρακτήρα char χρειάζεται να μπορεί να παρασταθεί μία τιμή ακόμα, η `EOF`.

```
int fputc(int char, FILE *stream)
```

η οποία γράφει έναν χαρακτήρα στο αρχείο και επιστρέφει είτε τον χαρακτήρα που έγραψε ή την τιμή `EOF` ως ένδειξη λάθους.

Είσοδος/Έξοδος ανά χαρακτήρα 2/2

Η χρήση του buffer μας επιτρέπει και την ακόλουθη «πολυτέλεια». Με την παρακάτω εντολή μπορεί να «επιστραφεί» ένας χαρακτήρας πίσω στον buffer ενός αρχείου που έχει ανοίξει για ανάγνωση.

```
int ungetc(int c, FILE *stream)
```

Επιστρέφει `EOF` σε περίπτωση σφάλματος ή τον χαρακτήρα `c` στον buffer ανάγνωσης (όχι στο ίδιο το αρχείο) και προσαρμόζει κατάλληλα τον δρομέα, ώστε η επόμενη εντολή ανάγνωσης να συνεχίσει από αυτόν. Δεν απαιτείται να έχει διαβαστεί ο χαρακτήρας (αυτός ή άλλος) προηγουμένως από αυτό το αρχείο.

Επίσης χρειάζεται **προσοχή** ότι εκτός από τις εντολές `fputc` και `fgetc`, υπάρχουν και οι παρόμοιες εντολές `putc` και `getc` τις οποίες όμως δεν πρέπει να χρησιμοποιείτε. Τις τεχνικές λεπτομέρειες μπορείτε να τις διαβάσετε σε αυτό το άρθρο, αλλά δεν μας αφορούν στα πλαίσια του μαθήματός μας.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-23@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Τονίζουμε : Μην στείλετε **ποτέ κώδικα ως εικόνα**, είναι παντελώς άχρηστος!

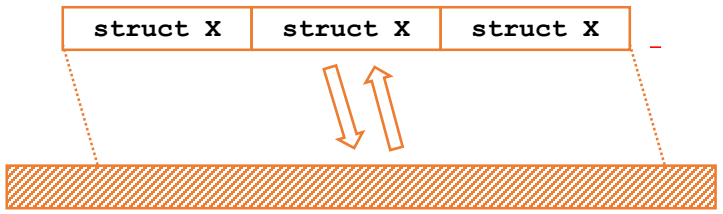
Binary Αρχεία

Ανάγνωση και εγγραφή δυαδικών δεδομένων από και προς αρχεία

Είσοδος/Έξοδος binary δεδομένων 1/2

Συχνά τα αρχεία δεν έχει νόημα να περιέχουν απλά κείμενο, για παράδειγμα αρχεία ήχου, εικόνες, βίντεο, συμπιεσμένα αρχεία και πολλά άλλα.

Η λογική της ανάγνωσης και εγγραφής binary δεδομένων από/σε αρχείο, είναι η μεταφορά τμημάτων του αρχείου σε μία περιοχή της μνήμης και αντίστροφα.



Η μεταφορά γίνεται σε ομάδες bytes οι οποίες ενδεχομένως δεν έχει νόημα να μεταφερθούν παρά μόνο ολόκληρες. Π.χ. ένας ολόκληρος `double` ή ένα ολόκληρο `struct`.

Κάθε φορά μπορεί να μεταφερθεί ένα ολόκληρο πλήθος από τέτοιες ομάδες (π.χ. `double`) με μία εντολή.

Είσοδος/Έξοδος binary δεδομένων 2/2

Οι εντολές ανάγνωσης και εγγραφής binary δεδομένων σε αρχείο είναι:

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

η οποία αντιγράφει από το αρχείο `stream` , `nmemb` φορες `size` bytes και τα βάζει στη μνήμη που δείχνει ο `ptr`. Επιστρέφει το πλήθος των τμημάτων που διαβάστηκαν, άρα έχει τιμές από `0` μέχρι `nmemb`.

$$0 \leq returnValue \leq nmemb$$

```
size_t fwrite(void *ptr, size_t size, size_t nmemb, FILE *stream)
```

η οποία λειτουργεί όμοια με την `fread`, απλά μετακινεί τα δεδομένα προς την αντίθετη κατεύθυνση.

int x = 1000000000;

Υπέρ & Κατά

Ως προς την αποθήκευση των αρχείων σε μορφή κειμένου ή δυαδική, μπορούμε να παρατηρήσουμε τα ακόλουθα.

	Κείμενο	Δυαδική
Μπορούν να περιέχουν κείμενο	Ναι	Ναι
Απαιτούμενος χώρος σε bytes	Μεγαλύτερος	Μικρότερος
Ταχύτητα ανάγνωσης/εγγραφής	Μικρότερη	Μεγαλύτερη
Έλεγχος περιεχομένων από άνθρωπο	Εύκολος	Δύσκολος
Απόκρυψη/Κρυπτογράφηση πληροφορίας	Περιορισμένη	Αυξημένη
Debugging του κώδικα που χειρίζεται τα αρχεία	Εύκολο	Δύσκολο

Σήμερα τα περισσότερα αρχεία εκτός από αυτά που έχουν εκ φύσεως binary πληροφορία (=ήχος, εικόνα, βίντεο) αποθηκεύονται σε αρχεία κειμένου. Και για να αντιμετωπιστεί το ζήτημα του αυξημένου χώρου που απαιτείται, συμπιέζονται σε κάποια μορφή, συνήθως zip.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-23@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Τονίζουμε : Μην στείλετε **ποτέ κώδικα ως εικόνα**, είναι παντελώς άχρηστος!

Διαχείριση Αρχείων

Μετακίνηση δρομέα, Διαχείριση ολόκληρου αρχείου

Μετακίνηση του δρομέα

Ο δρομέας του αρχείου, εκτός από την αυτόματη μετακίνηση μπορεί να μετακινηθεί και με την εντολή:

```
int fseek(FILE *stream, long int offset, int whence)
```

όπου μετακινείται ο δρομέας του αρχείου `stream` κατά `offset` bytes μετρώντας από τη θέση `whence`, η οποία μπορεί να είναι:

`SEEK_SET`	Η θέση offset μετρά από την αρχή του αρχείου
`SEEK_CUR`	Η θέση offset μετρά από την τρέχουσα θέση του δρομέα
`SEEK_END`	Η θέση offset μετρά από το τέλος του αρχείου και έχει νόημα όταν είναι αρνητική

Μηδενικό αποτέλεσμά σημαίνει επιτυχή εκτέλεσή της `fseek`

Αντίθετα εάν απλά χρειαζόμαστε να δούμε ποια είναι η θέση του δρομέα ενός αρχείου χρησιμοποιούμε την εντολή:

```
long int ftell(FILE *stream)
```

Η οποία επιστρέφει τη θέση το δρομέα ή `-1` εάν προέκυψε σφάλμα κατά την εκτέλεσή της.

Διαχείριση αρχείων

Μέχρι στιγμής έχουμε δει τη χρήση και τη δημιουργία αρχείων. Για υπάρχοντα αρχεία συχνά είναι χρήσιμες δύο ακόμα δυνατότητες:

```
int rename(const char *old_filename, const char *new_filename)
```

η οποία μετονομάζει ένα αρχείο χωρίς να το ανοίξει. Αυτό μπορεί να χρησιμοποιηθεί και για τη μετακίνηση ενός αρχείου από έναν φάκελο σε έναν άλλο.

```
int remove(const char *filename)
```

η οποία διαγράφει ένα αρχείο από τον δίσκο βάσει του path.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-23@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Τονίζουμε : Μην στείλετε **ποτέ κώδικα ως εικόνα**, είναι παντελώς άχρηστος!

Εφαρμογές

Ας εφαρμόσουμε επιτέλους τη θεωρία

Εφαρμογή εμφάνισης γραμμών

Γράψτε ένα κώδικα ο οποίος θα ανοίγει ένα αρχείο κειμένου και θα εκτυπώνει μόνο τις γραμμές του, οι οποίες περιέχουν ένα άλλο συγκεκριμένο string.

Το όνομα του αρχείου και το λεκτικό που αναζητείται θα είναι τα δύο ορίσματα της συνάρτησης.

```
void showLines(char *fileName, char *word);
```

Παράδειγμα αποθήκευσης δομών

Η αποθήκευση των δομών και γενικότερα πληροφορίας η οποία περιέχει pointers χρειάζεται προσοχή. Γι'αυτό ζητάμε να γραφτεί ένα πρόγραμμα, στο οποίο:

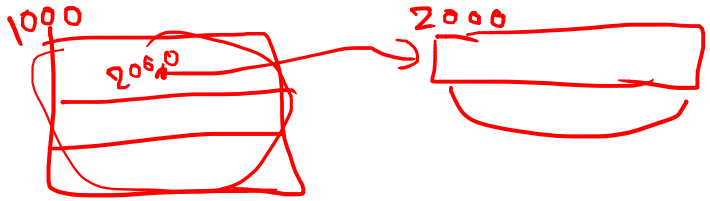
1. Θεωρήστε τη δομή που περιγράφει το SuperString από την προηγούμενη εργασία.
2. Γράψτε μία συνάρτηση που να αποθηκεύει μια τέτοια δομή σε ένα binary αρχείο.

   ```
   bool ssWriteToFile(SuperString *t, FILE *fout);
   ```

3. Γράψτε μία συνάρτηση που να διαβάζει τα binary δεδομένα από το αρχείο και να "γεμίζει" μια τέτοια δεδομένη δομή που τα περιέχει.

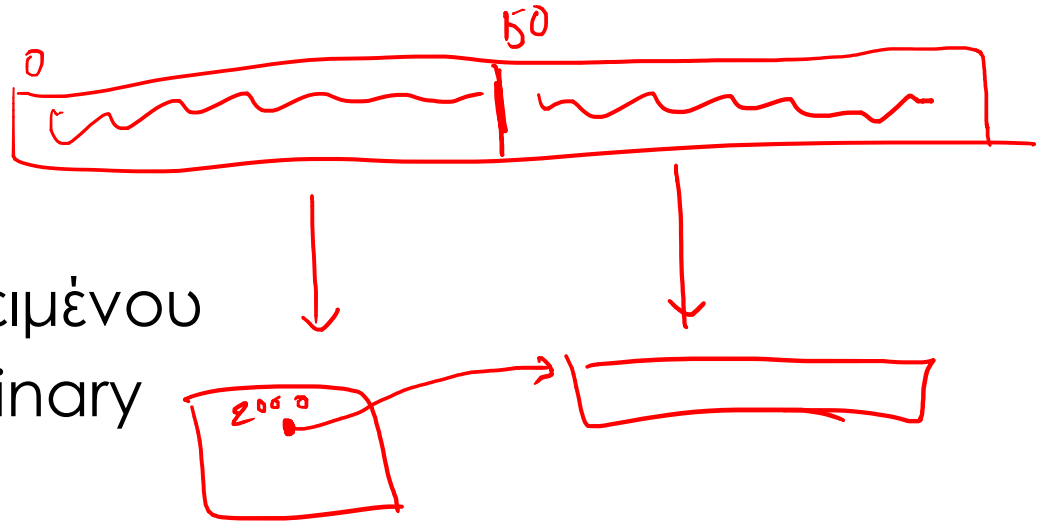
   ```
   bool ssReadFromFile(SuperString *t, FILE *fin);
   ```

Σημαντικά σημεία



Μετά από τη σημερινή διάλεξη θα πρέπει να γνωρίζετε:

- Τη λογική λειτουργίας των αρχείων
- Το άνοιγμα/κλείσιμο αρχείων
- Τα paths των αρχείων
- Ανάγνωση και εγγραφή σε μορφή κειμένου
- Ανάγνωση και εγγραφή σε μορφή binary
- Μετακίνηση του δρομέα
- Βασική διαχείριση αρχείων



Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-22@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!