



# ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

## Τμήμα Πληροφορικής

### ΕΠΛ 232 – Προγραμματιστικές Τεχνικές και Εργαλεία

#### Άσκηση 4: Ομαδική Συγγραφή Βιβλιοθήκης (.a) και Πελάτη για Επεξεργασία Αρχείων Ήχου WAV

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

Υπεύθυνοι Εργαστηρίων: Παύλος Αντωνίου και Πύρρος Μπράτσкас

Ημερομηνία Ανάθεσης: Πέμπτη, 1 Νοεμβρίου 2018

Ημερομηνία Παράδοσης: Πέμπτη, 22 Νοεμβρίου 2018 (21 μέρες)

(να υποβληθεί σε ένα zip στο Moodle)

#### I. Στόχος Άσκησης

Στόχος αυτής της άσκησης είναι να σας επιτρέψει να βάλετε μαζί όλες τις γνώσεις και αρχές που αποκτήσατε μέσω του ΕΠΛ232. Πιο συγκεκριμένα, το θέμα της εργασίας αφορά τη συγγραφή μιας **βιβλιοθήκης επεξεργασίας αρχείων ήχου** τύπου **WAV** (*Waveform Audio File Format - WAV*) και η παρουσίαση των δυνατοτήτων της μέσω της υλοποίησης ενός **πελάτη**. Η υλοποίηση θα γίνει σε ομάδες των δυο (2) ατόμων, όπως αυτές έχουν ανακοινωθεί μέσω του Moodle. Για την υλοποίηση της άσκησης θα χρειαστεί να χρησιμοποιήσετε τα ακόλουθα στοιχεία:

#### Νέα Στοιχεία:

1. **Βιβλιοθήκη:** Καλείστε να υλοποιήσετε κάθε μια από τις λειτουργίες της βιβλιοθήκης, οι οποίες περιγράφονται στην ενότητα V αυτής της εκφώνησης, ως ένα ξεχωριστό module (.c) παρέχοντας τα πρότυπα συναρτήσεων σε ένα ενιαίο αρχείο κεφαλίδας (π.χ., δείτε σχετικό παράδειγμα `string.h` στη διάλεξη 16). Νοείται ότι η βιβλιοθήκη σας δύναται να έχει όσα άλλα modules (ζεύγη .c/.h) κρίνεται σκόπιμο. Εφαρμόστε το κατάλληλο επίπεδο απόκρυψης πληροφοριών με χρήση `PRIVATE` ή `PUBLIC`.
2. **Πελάτης:** Καλείστε να υλοποιήσετε ένα πελάτη ο οποίος θα συνδέεται στατικά με τη βιβλιοθήκη για να παρουσιάζει τις λειτουργίες της. Ο πελάτης θα ήταν καλό να αποτελείται από ένα μονάχα .c αρχείο (όλες οι υπόλοιπες λειτουργίες να ενσωματωθούν στην βιβλιοθήκη).
3. **Ανάπτυξη σε Ομάδες:** Για την ανάπτυξη της ομαδικής αυτής εργασίας πρέπει να χρησιμοποιηθεί το σύστημα εκδόσεων Subversion (SVN) του τμήματος μας. Τα μέλη των ομάδων, όπως αυτά θα ανακοινωθούν στο Moodle, θα πρέπει να συμβάλουν ισομερώς σε χρόνο και ουσιαστική δουλειά. Συμβουλευτείτε τη σχετική διάλεξη του μαθήματος για σύνδεση και χρήση του SVN. Κάθε άτομο μέλος ομάδας **πρέπει** να υποβάλλει τις αλλαγές του στο SVN σε τακτή βάση (π.χ., ημερησίως ή πολλαπλές φορές τη εβδομάδα).
4. **Σχεδίαση Προγράμματος:** Προτρέπεται όπως η βιβλιοθήκη σας σχεδιαστεί από πάνω προς τα κάτω, αποφασίζοντας δηλαδή τα αρχεία κεφαλίδας συλλογικά και υλοποιώντας κάθε συνάρτηση ατομικά. Η ορθότητα κάθε συνάρτησης της βιβλιοθήκης (το οποίο ονομάζεται module στη περίπτωση μας), θα πρέπει να εκλεχθεί και από τα δυο μέλη της ομάδας αλλά θα πρέπει να υλοποιείται από ένα μόνο μέλος της ομάδας (του οποίου το όνομα θα τοποθετηθεί και ως "Author" στο εν λόγω αρχείο).
5. **Έλεγχος Προγράμματος:** Το τελικό λογισμικό σας θα πρέπει να ελεγχθεί στατικά (κατά τη μεταγλώττιση και με τον debugger) αλλά και δυναμικά (με `valgrind` για διαρροή

μνήμης, profiler gprof για εύρεση συναρτήσεων που πρέπει να βελτιστοποιηθούν). Νοείται ότι κάθε module του προγράμματος σας θα συνοδεύεται από τους σχετικούς οδηγούς χρήσης για να γίνει το σχετικό white-box testing.

6. **GPL:** Για σκοπούς εξοικείωσης σας με την ορολογία του λογισμικού ανοικτού πηγαίου κώδικα, η βιβλιοθήκη σας θα πρέπει να κάνει χρήση του GPL προοιμίου (preamble) σε κάθε αρχείο πηγαίου κώδικα της βιβλιοθήκης. Ο πελάτης θα πρέπει να ανταποκρίνεται με το προοίμιο που θα δούμε στη διάλεξη 17, εάν δε δοθούν ορίσματα.
7. **Αξιολόγηση:** Η αξιολόγηση της άσκησης θα στηριχτεί στα αναλυτικά κριτήρια που θα παρουσιαστούν στο τέλος αυτής της εκφώνησης. Μεταξύ άλλων, βασικός στόχος της άσκησης είναι να αξιολογηθεί η δομή και οργάνωση της βιβλιοθήκης σας αλλά και επίδοση του συνολικού προγράμματος σας (π.χ., πόση ώρα θα χρειαστεί να διεκπεραιώσει μια ακολουθία από εντολές; πόσο αποδοτικά χρησιμοποιείται η μνήμη, κτλ.)
8. **Τελικός Διαγωνισμός:** Η τελική αξιολόγηση της άσκησης, στο τελευταίο εργαστήριο, δύναται να αναδείξει ένα μικρό αριθμό ομάδων οι οποίες ομάδες θα κληθούν να παρουσιάσουν το λογισμικό τους σε ένα τελικό διαγωνισμό (ημερομηνίες του τελικού διαγωνισμού θα ακολουθήσουν στη συνέχεια).

### Στοιχεία από Προηγούμενες Εργασίες:

1. Το πρόγραμμα πρέπει να μεταγλωττίζεται τόσο μέσω του eCclipse όσο και μέσω της γραμμής εντολής με το makefile. **Κάθε αντικείμενο (module)** πρέπει να συμπεριλαμβάνει τους **σχετικό οδηγό χρήσης (driver functions)**.
2. **Σχόλια** και οδηγό σχολίων με χρήση του **doxygen** αλλά και διάγραμμα εξαρτήσεων αντικειμένων με χρήση του graphviz. Για χρήση των διαγραμμάτων στο eCclipse του H/Y σας μπορείτε να χρησιμοποιήσετε το Eclox.

Οι λειτουργίες της βιβλιοθήκης σας και το αναμενόμενο αποτέλεσμα του πελάτη περιγράφονται αναλυτικότερα στην συνέχεια, αφού επεξηγηθεί πρώτα το πρόβλημα.

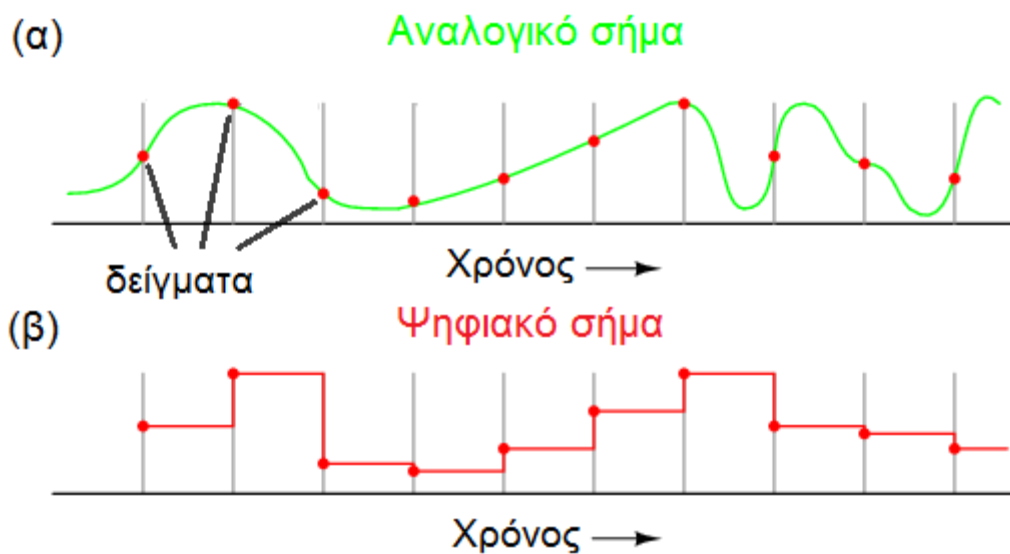
## II. Εισαγωγή

Γνωρίζουμε ότι τα ψηφιακά αρχεία ήχου σε ένα υπολογιστή (π.χ., WAV, MP3, WMA, κτλ.) αποθηκεύουν την πληροφορία σε δυαδική μορφή (παρά σε αρχεία χαρακτήρων ASCII που είδατε μέχρι στιγμής). Τα αρχεία ήχου μπορεί να βρίσκονται σε μη συμπίεσμένη (.WAV) ή συμπίεσμένη (.MP3, .WMA) μορφή. Τα συμπίεσμένα αρχεία έχουν συνήθως μικρότερο μέγεθος (σε bytes) από τα μη-συμπίεσμένα. Τα αρχεία με κατάληξη **.WAV** είναι μια ειδική κατηγορία αρχείων ήχου σε ασυμπίεστη μορφή που αναπτύχθηκε από την Microsoft και την IBM. Η απλή εσωτερική δομή των αρχείων αυτών τα καθιστούν κατάλληλα για την διδασκαλία βασικών αρχών που έχετε διδαχθεί στα πλαίσια του μαθήματος μας. Περισσότερες πληροφορίες για παιδαγωγικούς λόγους: <http://en.wikipedia.org/wiki/WAV>

Στη φύση, ο ήχος διαδίδεται μέσω κυμάτων τα οποία διαφέρουν στο ύψος. Ο ήχος είναι στην ουσία ένα αναλογικό σήμα με κάποια κυματομορφή όπως φαίνεται στην Εικόνα 1(α) με πράσινο χρώμα. Για να χρησιμοποιηθεί ένα *αναλογικό σήμα* σε έναν υπολογιστή – είτε για να αναπαραχθεί από κάποιο πρόγραμμα ήχου (π.χ. Windows Media Player) ή για να σταλεί μέσω του Διαδικτύου (Internet) – πρέπει να μετατραπεί σε *ψηφιακό σήμα* μέσω ενός μετατροπέα (analog to digital converter). Με άλλα λόγια ο ψηφιακός ήχος αποθηκεύεται σε bits με τιμές 1 ή 0 και όχι σε κύματα. Το ερώτημα είναι πως εκφράζουμε την κυματοειδή φύση του ήχου σε σειρές από bits;

Για την ψηφιοποίηση του ήχου, η διαδικασία που ακολουθείται είναι η *δειγματοληψία*. Δηλαδή λαμβάνονται περιοδικά κάποια δείγματα από το αναλογικό σήμα τα οποία αποτελούν το ψηφιακό σήμα, το οποίο φαίνεται στην Εικόνα 1(β) με κόκκινο χρώμα. Προφανώς, όσο περισσότερα δείγματα ήχου λαμβάνονται, τόσο πιο κοντά θα είναι ο ψηφιοποιημένος ήχος στον αρχικό αναλογικό, γιατί χρησιμοποιούνται περισσότερα σημεία για να περιγραφεί το σχήμα των της αναλογικής κυματομορφής. Έτσι έχουμε ψηφιακό ήχο υψηλότερης ποιότητας. Η *συχνότητα*

(ρυθμός) με την οποία λαμβάνονται τα σήματα ονομάζεται *συχνότητα δειγματοληψίας (sampling rate ή frequency)*. Στην πράξη τιμές της συχνότητας δειγματοληψίας είναι 11.025 kHz, 22.05 kHz και 44.1 kHz (KiloHertz: kilo = 1000, hertz = αριθμός ανά δευτερόλεπτο). Όταν για παράδειγμα έχουμε συχνότητα δειγματοληψίας ίση με 44.1 kHz αυτό σημαίνει ότι για να δημιουργήσουμε το ψηφιακό σήμα λαμβάνουμε 44100 δείγματα το δευτερόλεπτο από το αναλογικό σήμα. Κάθε ένα από τα δείγματα αυτά αποθηκεύεται σαν ένας δυαδικός αριθμός αποτελούμενος από bits. Στην πράξη, λαμβάνονται συνήθως δείγματα 8-bit (256 διαβαθμίσεις) ή 16-bit (65536 διαβαθμίσεις). Αν λαμβάνονται 16-bit δείγματα τότε το ψηφιακό σήμα έχει καλύτερη ποιότητα αλλά χρειάζεται περισσότερο χώρο για να αποθηκευτεί στο δίσκο και μεταδοθεί. Ένα αρχείο ήχου σε ψηφιακή μορφή μπορεί να έχει είτε ένα κανάλι ήχου (mono) ή δύο κανάλια ήχου (stereo). Το κάθε κανάλι ψηφιοποιείται (συνήθως) από διαφορετικό αναλογικό σήμα μέσω της πιο πάνω διαδικασίας.



Εικόνα 1: Αναλογικό και ψηφιακό σήμα

Το μέγεθος ενός αρχείου λοιπόν εξαρτάται (α) από το *πλήθος των καναλιών (number of channels)*, (β) το *πλήθος δειγμάτων* που λαμβάνονται κατά τη *δειγματοληψία (sample rate)*, και (γ) τον αριθμό των *bits ανά δείγμα (bits per sample)*. Για να κατανοήσουμε πόσο χώρο χρειαζόμαστε για να αποθηκεύσουμε ψηφιακό αρχείο ήχου (μουσική) διάρκειας 3 λεπτών (180 δευτερολέπτων) έχουμε το πιο κάτω παράδειγμα. Έστω ότι το ψηφιακό στερεοφωνικό (= 2 κανάλια) σήμα έχει προκύψει από δειγματοληψία στα 44.1 kHz (44100 δείγματα ανά δευτερόλεπτο) με μέγεθος κάθε δείγματος 16 bit. Αυτό σημαίνει ότι για ένα δευτερόλεπτο ήχου χρειαζόμαστε  $2 \text{ κανάλια} * 44100 \text{ δείγματα} * 16 \text{ bits} = 1411200 \text{ bits} = 176400 \text{ Bytes}$ . Εφόσον έχουμε 180 δευτερόλεπτα μουσικής αυτά θα καταλαμβάνουν χώρο στη μνήμη τουλάχιστον  $180 \text{ sec} * 176400 \text{ Bytes} = 31752000 \text{ Bytes}$  ή  $\sim 30.2 \text{ MBytes}$ . Το “τουλάχιστον” δηλώνει ότι πέραν από τα χρήσιμα δεδομένα (data) του αρχείου ήχου αποθηκεύεται επίσης στη μνήμη η επιπλέον πληροφορία της κεφαλίδας (header). Η κεφαλίδα αποθηκεύει διάφορες μέτα-πληροφορίες οι οποίες χαρακτηρίζουν τα bytes τα οποία ακολουθούν.

Για παράδειγμα ένα αρχείο ήχου θα αποθηκευτεί στη δευτερεύουσα μνήμη στη μορφή που υποδεικνύεται στην Εικόνα 2.



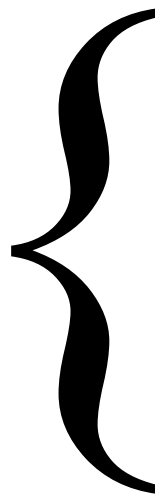
Εικόνα 2

Όπως βλέπουμε, τα δεδομένα (Data) ακολουθούν τις πληροφορίες της κεφαλίδας (Header). Όλοι οι τύποι δυαδικών αρχείων (.doc, .wav, .jpg, .pdf....) έχουν κάποιο είδος header. Συνεπώς με την ολοκλήρωση αυτής της εργασίας θα έχετε εκτιμήσει πολύ καλά τι χρειάζεται για να επεξεργαστείτε πραγματικά αρχεία δεδομένων σε ένα πρόγραμμα.

### III. Δομή του Header σε Αρχεία WAV

Ας δούμε τώρα αναλυτικότερα το Header μέρος ενός WAV αρχείου. Όπως φαίνεται στο σχήμα της Εικόνας 3, ένα αρχείο WAV διαιρείται σε 3 μέρη, όπου τα 2 πρώτα μέρη περιέχουν καθαρά συνοδευτικές πληροφορίες για το αρχείο ήχου (μέτα-πληροφορία) ενώ το τρίτο μέρος περιέχει τόσο μέτα-πληροφορίες όσο και δεδομένα (data). Τα δεδομένα δεν αποτελούν μέρος του Header και θα επεξηγηθούν στην επόμενη ενότητα.

Τα τρία μέρη RIFF CHUNK DESCRIPTOR, FMT SUB-CHUNK και DATA SUB-CHUNK παρουσιάζονται αναλυτικότερα στη συνέχεια. Το πρώτο μέρος περιέχει πληροφορίες για το **αρχείο** (π.χ., το μέγεθος του αρχείου σε bytes), το δεύτερο μέρος περιέχει πληροφορίες για τον **ήχο** (π.χ., αριθμός καναλιών, πλήθος δειγμάτων που λαμβάνονται κατά τη δειγματοληψία, κτλ) και το τρίτο μέρος περιέχει πληροφορίες για τα δεδομένα.



endian	File offset (bytes)	Field name	Field size (bytes)	
big	1 – 4	ChunkID	4	RIFF CHUNK
little	5 – 8	ChunkSize	4	
big	9 – 12	Format	4	
big	13 – 16	Subchunk1ID	4	FMT SUB-CHUNK
little	17 – 20	Subchunk1Size	4	
little	21 – 22	AudioFormat	2	
little	23 – 24	NumChannels	2	
little	25 – 28	SampleRate	4	
little	29 – 32	ByteRate	4	
little	33 – 34	BlockAlign	2	
little	35 – 36	BitsPerSample	2	
big	37 – 40	Subchunk2ID	4	DATA SUB-CHUNK
little	41 – 44	Subchunk2Size	4	
little		Data	Subchunk2 Size	

Εικόνα 3: Δομή ενός αρχείου WAV. Η αγκύλη υποδεικνύει το header του αρχείου

Τα Header ενός αρχείου WAV έχει συνολικό μέγεθος 44 bytes, όπως φαίνεται και στο πιο πάνω σχήμα μέσα σε έντονο περίγραμμα με την αγκύλη.

#### Endianness

Στο σημείο αυτό αξίζει να αναφερθούμε στην ορολογία big και little endian, όπως εμφανίζεται στη πρώτη στήλη, και η οποία σχετίζεται με το πώς θέλει ένας Η/Υ να αναπαριστά στη μνήμη δεδομένα που καταλαμβάνουν πολλαπλά bytes (π.χ., short, int, structs, κτλ., αλλά όχι char που καταλαμβάνει 1 byte μόνο). Κάποιες αρχιτεκτονικές Η/Υ λοιπόν, θέλουν να αναπαραστούν τα δεδομένα πολλαπλών bytes από αριστερά προς δεξιά (big endian) ενώ άλλες από δεξιά προς αριστερά (little endian). Για να αποφευχθεί αυτή η σύγχυση, το WAV πρότυπο (όπως φαίνεται στην εικόνα 3) δηλώνει ρητά σε τι endianness βρίσκεται το κάθε ένα από τα δεδομένα που εμπεριλαμβάνει.

**Για παράδειγμα**, θεωρήστε το πρώτο πεδίο **ChunkID** το οποίο είναι 4 bytes σε big endian. Το εν λόγω πεδίο καταγράφει την ακολουθία bytes "RIFF", όπως θα εξηγήσουμε σε λίγο, δηλαδή 'R' (=52, στο δεκαεξαδικό όπως δείχνει ο πίνακας ASCII στην ενότητα VII), 'I' (=49), 'F' (=46), 'F'

(=46). Αυτό σημαίνει ότι στο αρχείο θα βρούμε στα πρώτα 4 bytes την δεκαεξαδική ακολουθία 52-49-46-46 η οποία πρέπει να διαβαστεί από αριστερά στα δεξιά.

**Ως δεύτερο παράδειγμα**, θεωρήστε το επόμενο πεδίο **ChunkSize** το οποίο είναι 4 bytes σε little endian. Το εν λόγω πεδίο καταγράφει το μέγεθος του αρχείου σε bytes και έχει τιμή 171556 για το αρχείο Windows\_Error.wav που θα δούμε σε λίγο (και που παρουσιάζεται αποσπασματικά στο σχήμα που ακολουθεί). Αυτό σημαίνει ότι στο αρχείο θα βρούμε την δεκαεξαδική ακολουθία 24-9e-02-00 να ακολουθεί τους χαρακτήρες RIFF. Εάν διαβαστεί η δεκαεξαδική ακολουθία 24-9e-02-00 byte-byte από το τέλος προς την αρχή δίνεται η δεκαεξαδική ακολουθία 00-02-9e-24, το οποίο είναι η δεκαδική αναπαράσταση 171556 (δοκιμάστε το σε μια υπολογιστική!), και το οποίο υποδηλώνει το μέγεθος του αρχείου.

RIFF CHUNK DESCRIPTOR											
52	49	46	46	24	9e	02	00	57	41	56	45
R	I	F	F	ChunkSize=171556				W	A	V	E

Από το σχήμα 3 θα παρατηρήσετε ότι όλες οι αναπαραστάσεις ASCII χαρακτήρων στο header (π.χ., ChunkID="RIFF", Format="WAVE", SubChunkID="FMT" και SubChunk2ID="DATA"), βρίσκονται σε big-endian ενώ οι υπόλοιπες και τα data σε little endian.

### Περιγραφή Header

Οι επόμενοι πίνακες περιγράφουν αναλυτικά τα πεδία των headers. Εσείς θα κληθείτε να επεξεργαστείτε τα πεδία αυτά για να υλοποιήσετε τα ζητούμενα της άσκησης αυτής. Ειδικότερα θα πρέπει να ορίσετε τις κατάλληλες δομές (struct) λαμβάνοντας υπόψη θέματα ευθυγράμμισης μνήμης. Για ευκολία αναφοράς καλείστε να χρησιμοποιήσετε τις ακόλουθες δηλώσεις:

```
typedef unsigned char byte;
typedef unsigned short int word;
typedef unsigned int dword;
```

Η δομή ενός αρχείου WAV ξεκινά με το RIFF header:

#### RIFF CHUNK DESCRIPTOR:

Bytes	Όνομα	Χρήση
4	ChunkID	Contains the letters "RIFF" in ASCII form (big-endian form).
4	ChunkSize	Contains the size of the entire file in bytes minus 8 bytes for the two fields not included in this count: ChunkID and ChunkSize.
4	Format	Contains the letters "WAVE" (big-endian form).

Στη συνέχεια ακολουθούν δύο subchunks: "fmt " and "data". Το "fmt " subchunk περιγράφει τη μορφή των δεδομένων ήχου όπως φαίνεται πιο κάτω:

#### FMT SUB-CHUNK:

Bytes	Όνομα	Χρήση
4	Subchunk1ID	Contains the letters "fmt " (big-endian form).
4	Subchunk1Size	16 for PCM. This is the size of the rest of the Subchunk which follows this number.
2	AudioFormat	PCM = 1 (i.e. Linear quantization) Values other than 1 indicate some form of compression.
2	NumChannels	Mono = 1, Stereo = 2, etc.
4	SampleRate	8000, 44100, etc.
4	ByteRate	== SampleRate * NumChannels * BitsPerSample/8



2	BlockAlign	$== \text{NumChannels} * \text{BitsPerSample}/8$ The number of bytes for one sample including all channels.
2	BitsPerSample	8 bits = 8, 16 bits = 16, etc.

Το "data" subchunk περιέχει το μέγεθος των δεδομένων (σε bytes) και τα πραγματικά δεδομένα ήχου:

#### DATA SUB-CHUNK:

Bytes	Όνομα	Χρήση
4	Subchunk2ID	Contains the letters "data" (big-endian form).
4	Subchunk2Size	$== \text{NumSamples} * \text{NumChannels} * \text{BitsPerSample}/8$ . This is the number of bytes in the data.
*	Data	The actual sound data.

## IV. Δομή του DATA σε Αρχεία WAV

Τα data (δηλαδή τα πραγματικά bytes ενός αρχείου ήχου) αποθηκεύονται μέσα στο αρχείο ανά δείγμα. Για κάθε δείγμα αποθηκεύεται το αριστερό και το δεξί κανάλι. Μπορείτε να δείτε το πιο κάτω παράδειγμα που απεικονίζει (αποσπασματικά) τα περιεχόμενα του αρχείου από το αρχείο Windows\_error.wav (κεφαλίδα και μερικά δεδομένα) σε δεκαεξαδική αναπαράσταση.

RIFF CHUNK DESCRIPTOR												FMT SUB-CHUNK											
52	49	46	46	24	9e	02	00	57	41	56	45	66	6d	74	20	10	00	00	00	01	00	02	00
R	I	F	F	ChunkSize = 171556				W	A	V	E	f	m	t		SubChunk1Size=16				AudioFormat = 1 (PCM)		NumChannels = 2	

												DATA SUB-CHUNK											
44	ac	00	00	10	b1	02	00	04	00	10	00	64	61	74	61	00	9e	02	00	00	00	00	00
SampleRate=44100				ByteRate=176400				BlockAlign=4				BitsPerSample=16				SubChunk2Size=171520				Left Channel		Right Channel	

LEFT	RIGHT	LEFT	RIGHT	LEFT	RIGHT	LEFT	RIGHT	LEFT	RIGHT	LEFT	RIGHT	LEFT	RIGHT
00	00	00	00	00	00	00	00	00	00	00	00	00	00
Sample 2		Sample 3		Sample 4		Sample 5		Sample 6		Sample ...			

Η πιο κάτω εκτέλεση της εντολής `hexdump` δείχνει το περιεχόμενο του **Windows\_Error.wav** αρχείου σε bytes: η **πρώτη στήλη** δείχνει τη διεύθυνση της γραμμής σε δεκαεξαδική αναπαράσταση, η **δεύτερη στήλη** δείχνει ένα-ένα τα bytes του αρχείου (για χάρη παρουσίασης κάθε γραμμή περιέχει 16 bytes) και η **τρίτη στήλη** δείχνει κάθε ένα από τα 16 bytes της γραμμής ως να ήταν ASCII χαρακτήρες (στη περίπτωση μας οι χαρακτήρες αντιστοιχούν στις λέξεις RIFF, WAVE, fmt και data είναι πραγματικά ASCII χαρακτήρες, τα υπόλοιπα είναι ακολουθίες δυάδων byte, word, dword).

```
$ hexdump -C Windows_Error.wav | head
```

ΣΤΗΛΗ 1	ΣΤΗΛΗ 2	ΣΤΗΛΗ 3
00000000	52 49 46 46 24 9e 02 00 57 41 56 45 66 6d 74 20	RIFF\$...WAVEfmt
00000010	10 00 00 00 01 00 02 00 44 ac 00 00 10 b1 02 00	.....D.....
00000020	04 00 10 00 64 61 74 61 00 9e 02 00 00 00 00 00	....data.....
00000030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
*		
000006a0	00 00 00 00 00 00 00 00 00 00 ff ff 00 00 fe ff	.....
000006b0	00 00 fc ff 00 00 fa ff 00 00 fd ff 00 00 09 00	.....
000006c0	00 00 26 00 00 00 4f 00 00 00 82 00 00 00 bb 00	...&...O.....
000006d0	00 00 f6 00 00 00 2e 01 00 00 65 01 00 00 95 01	.....e.....
000006e0	00 00 bd 01 00 00 db 01 00 00 e9 01 00 00 e4 01	.....

Από την πιο πάνω έξοδο της `hexdump` παρατηρούμε τα ακόλουθα: **α)** τα πρώτα 44 bytes, τα οποία είναι σκιασμένα με γκρίζο, για χάρη παρουσίασης, αναφέρονται στο header της εικόνας. **β)** Το header ακολουθείται από πολλές γραμμές με bytes, τα οποία ανά δύο αναφέρονται στο αριστερό και δεξί κανάλι αντίστοιχα, πλάτους 2 bytes το καθένα. Συνολικά, τα bytes των δεδομένων δίνονται από τη μεταβλητή `SubChunk2Size` της οποίας η τιμή δίνεται από τα 4 τελευταία σκιασμένα bytes (little endian μορφή).  $00\ 02\ 9e\ 00_{16} = 171520_{10}$ . Λόγω του ότι όλα τα bytes των δεδομένων του αρχείου από τη γραμμή 00000030 μέχρι τη γραμμή 000006a0 είναι τα ίδια, δεν εκτυπώνονται αλλά εκτυπώνεται ένα \*. Επίσης η πιο πάνω εντολή δίδει τις 10 πρώτες γραμμές δεδομένων (λόγω της εντολής `head`). Αν αφαιρέσουμε την εντολή `| head` τότε θα εκτυπωθούν όλα τα δεδομένα του αρχείου `Windows_Error.wav`.

## VI. Ζητούμενα Άσκησης

Έστω ότι `wavlib.a` είναι η βιβλιοθήκη που θα δημιουργήσετε και `wavengine.c` ο πελάτης, τότε `wavengine` είναι το εκτελέσιμο αρχείο που παράγεται συνδέοντας το `wavlib.a` με το `wavengine.c` κατά την μεταγλώττιση (υποδειγματικά αρχεία ήχου WAV έχουν αναρτηθεί στο `as4-supplementary.zip`). Πιο κάτω περιγράφεται η αναμενόμενη λειτουργία της βιβλιοθήκης κάνοντας αναφορά στις λειτουργίες της μέσω υποδειγματικών εκτελέσεων του πελάτη.

### A) Πελάτης

Η γενική μορφή εκτέλεσης του πελάτη είναι η ακόλουθη:

```
$ ./wavengine <-option> sound1.wav [ sound2.wav sound3.wav ... ]
```

όπου `option` σχετίζεται με μια επί μέρους λειτουργία της βιβλιοθήκης (π.χ., `-list` εκτυπώνει το header, που θα δούμε σε λίγο) και στη συνέχεια ακολουθεί ένας απροσδιόριστος αριθμός από ονόματα αρχείων. Σημειώστε ότι το `./wavengine <-option> *.wav` θα εφαρμόσει ένα `option` σε όλα τα αρχεία wav του καταλόγου.

Ενδεχόμενα Λάθη (μεταξύ άλλων)

- Δεν ορίζεται `option` ή δεν δίνεται το όνομα τουλάχιστο ενός αρχείου ήχου: Πρέπει να εκτυπώνεται το GPL προοίμιο και στη συνέχεια να δίνεται το σχετικό μήνυμα λάθους.
- Εάν ένα αρχείο ήχου δεν είναι .wav ή δεν είναι PCM: το αρχείο πρέπει να αγνοείται.

### B) Βιβλιοθήκη

#### Λειτουργία 1: Εξαγωγή Μέτα-Πληροφοριών (Listing)

Το όρισμα `-list` αφορά τη εκτύπωση των στοιχείων του header ενός προσδιορισμένο αρχείου ήχου σε μορφή που ακολουθεί.

```
$ ./wavengine -list Windows_Error.wav Windows_Shutdown.wav
```

```
RIFF_CHUNK_HEADER
=====
chunkID: RIFF
chunkSize: 171556
format: WAVE

fmt_subchunk1_header
=====
subChunk1ID: fmt
subChunk1Size: 16
audioFormat: 1
numChannels: 2
sampleRate: 44100
byteRate: 176400
blockAlign: 4
bitsPerSample: 16

DATA_subchunk1_header
```

```

=====
subChunk2ID: data
subChunk2Size: 171520

*****
RIFF_CHUNK_HEADER
=====
chunkID: RIFF
chunkSize: 169436
format: WAVE

FMT_SUBCHUNK_HEADER
=====
subChunk1ID: fmt
subChunk1Size: 16
audioFormat: 1
numChannels: 2
sampleRate: 44100
byteRate: 176400
blockAlign: 4
bitsPerSample: 16

DATA_SUBCHUNK_HEADER
=====
subChunk2ID: data
subChunk2Size: 169400

```

## Λειτουργία 2: Μετατροπή από Στερεοφωνικό σε Μονοφωνικό (Stereo to Mono)

Το όρισμα `-mono` μετατρέπει το αρχείο ήχου από στερεοφωνικό (ύπαρξη 2 καναλιών) σε μονοφωνικό (ύπαρξη μόνο ενός καναλιού). Αυτό μπορεί να γίνει με διαγραφή του δεξιού ή του αριστερού καναλιού (στην άσκηση να αφαιρείται το δεξί κανάλι). Το αποτέλεσμα αποθηκεύεται σε νέο αρχείο με όνομα `new-sound1.wav`, όπου `sound1.wav` είναι το αρχικό όνομα του αρχείου ήχου. Σημειώστε ότι ο χρήστης έχει και πάλι τη δυνατότητα να εισάγει ένα απροσδιόριστο αριθμό από ονόματα αρχείων όπως και στην επιλογή `list`, π.χ.,

```
$ ./wavengine -mono sound1.wav sound2.wav
```

Η πιο πάνω εντολή δίνει σαν είσοδο 2 αρχεία ήχου για διαγραφή του δεξιού καναλιού και οπότε θα πρέπει να παραχθούν 2 αρχεία με μονοφωνικό ήχο στην έξοδο.

## Λειτουργία 3: Μίξη 2 αρχείων ήχου (Mix 2 tracks)

Το όρισμα `-mix` συνενώνει 2 αρχεία ήχου σε ένα νέο αρχείο ήχου. Το νέο αρχείο ήχου θα αποθηκεύεται σε νέο αρχείο με όνομα `mix-<filename1>-<filename2>.wav` όπου το `<filename1>` και `<filename2>` είναι τα ονόματα των 2 αρχείων που θα συνενωθούν. Το νέο αρχείο ήχου θα έχει σαν δεξιό κανάλι (`right channel`) το δεξιό κανάλι του πρώτου αρχείου και σαν αριστερό κανάλι (`left channel`) το αριστερό κανάλι του δεύτερου αρχείου. Η διάρκεια του νέου αρχείου θα είναι ίση με τη διάρκεια του μικρότερου από τα 2 αρχεία εισόδου. Παρακάτω φαίνεται ένα παράδειγμα μίξης 2 αρχείων ήχου:

```
$ ./wavengine -mix sound1.wav sound2.wav
```

## Λειτουργία 4: Τεμαχισμός Κομματιού Ήχου (Chop track)

Το όρισμα `-chop` τεμαχίζει ένα αρχείο ήχου από μια χρονική στιγμή σε μια άλλη. Το αποτέλεσμα αποθηκεύεται σε νέο αρχείο με όνομα `chopped-sound1.wav`, όπου `sound1.wav` είναι το αρχικό όνομα του αρχείου ήχου. Παρακάτω φαίνεται ένα παράδειγμα τεμαχισμού ενός αρχείου ήχου από το 2<sup>ο</sup> στο 4<sup>ο</sup> δευτερόλεπτο:

```
$ ./wavengine -chop sound1.wav 2 4
```



Να γίνονται οι κατάλληλοι έλεγχοι για τις χρονικές στιγμές εισόδου αν βρίσκονται εντός της διάρκειας του αρχείου εισόδου.

### Λειτουργία 5: Αντιστροφή Αρχείου Ήχου (Reverse track)

Το όρισμα `-reverse` αντιστρέφει τα δεδομένα ενός αρχείο ήχου. Το αποτέλεσμα αποθηκεύεται σε νέο αρχείο με όνομα `reverse-sound1.wav`, όπου `sound1.wav` είναι το αρχικό όνομα του αρχείου ήχου. Σημειώστε ότι ο χρήστης έχει και πάλι τη δυνατότητα να εισάγει ένα απροσδιόριστο αριθμό από ονόματα αρχείων όπως και στην επιλογή `list`.

```
$./wavengine -reverse sound1.wav sound2.wav
```

### Λειτουργία 6: Έλεγχος ομοιότητας αρχείων ήχου (Track Similarity)

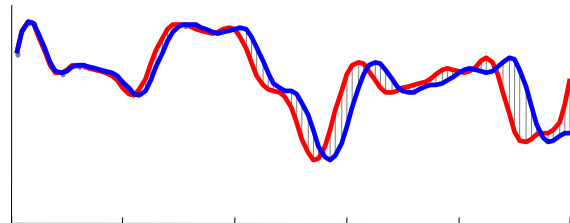
Το όρισμα `-similarity` θα ελέγχει την ομοιότητα **ανά byte** των δοθέντων αρχείων ήχου (μόνο πάνω στα `data`, όχι το `header`). Ο έλεγχος ομοιότητας θα γίνεται με βάση δύο γνωστούς αλγόριθμους: (α) την τεχνική ταύτισης του Ευκλείδη (Euclidean matching) και (β) την τεχνική ταύτισης μέγιστης κοινής υπό-ακολουθίας (Longest Common Subsequence (LCSS) matching). Η διαδικασία σύγκρισης αρχείων ήχου χρησιμοποιείται στην φωνητική αναγνώριση (speech recognition).

#### Τεχνική Ταύτισης Ευκλείδη

Στην τεχνική ταύτισης του Ευκλείδη οι δύο εμπλεκόμενες ακολουθίες συγκρίνονται «παράλληλα», στοιχείο προς στοιχείο. Ταυτόχρονα υπολογίζεται η ευκλείδεια απόσταση (euclidean distance). Η Ευκλείδεια απόσταση είναι η πιο ευρέως διαδεδομένη μετρική απόστασης ακολουθιών που ορίζει ομοιότητα/διαφοροποίηση μεταξύ των ακολουθιών *A* και *B*:

$$d_{EUCLIDEAN}(A, B) = \sqrt{\sum_{i=1}^n |A[i] - B[i]|^2}$$

Εξίσωση 1



Η τεχνική ταύτισης του Ευκλείδη είναι αρκετά γρήγορη αλλά η χειρότερη από πλευράς ακρίβειας. Ένα μειονέκτημα είναι ότι η τεχνική δεν βρίσκει τη μέγιστη υπό-ακολουθία (την υπό-ακολουθία με το μέγιστο μήκος) μεταξύ δύο ακολουθιών. Για παράδειγμα δείτε πιο κάτω τη σύγκριση μεταξύ δύο 8-byte ακολουθιών (οι δύο ακολουθίες φαίνονται σε δεκαεξαδική μορφή):

*A*={11 bc ff 22 b3 68 96 78}

| | | ● ● ● ● ●

*B*={11 bc ff 2b 2f b3 f8 96}

Σύμφωνα με τη τεχνική της Ευκλείδειας απόστασης (εξίσωση 1), οι πράσινες γραμμές υποδηλώνουν επιτυχή ταύτιση, ενώ οι κόκκινες αποτυχημένη ταύτιση. Οπότε η υπό-ακολουθία που υπολογίζει η τεχνική της ταύτισης του Ευκλείδη είναι 11 bc ff και έχει μήκος 3 bytes. Η υπό-ακολουθία αυτή δεν είναι η μέγιστη. Η μέγιστη υπό-ακολουθία μεταξύ *A* και *B* έχει μήκος 5 bytes (11 bc ff b3 96) όπως φαίνεται πιο κάτω:

*A*={11 bc ff 22 b3 68 96 78}

| | | ● \ ● \ ●

*B*={11 bc ff 2b 2f b3 f8 96}

**Εάν τα αρχεία έχουν διαφορετικές παραμέτρους (channels, bitspersample, κτλ.) μπορείτε να δώσετε κάποιο μήνυμα λάθους ή να τα ευθυγραμμίσετε έναντι επιπλέον μονάδων.**

### Τεχνική Ταύτισης Μέγιστης Κοινής Υπό-ακολουθίας

Η μέγιστη υπό-ακολουθία μεταξύ δύο ακολουθιών μπορεί να υπολογιστεί με την τεχνική ταύτισης μέγιστης κοινής υπό-ακολουθίας (Longest Common SubSequence – LCSS – Matching):

$$LCSS(A, B) = \begin{cases} 0 & \\ 1 + LCSS(Tail(A), Tail(B)) & \\ \max(LCSS(Tail(A), B), LCSS(A, Tail(B))) & \end{cases} \quad \text{Εξίσωση 2}$$

Αντί του αναδρομικού τύπου υπολογισμού του LCSS (που είναι αρκετά αργός λόγω της αναδρομικής του φύσεως) μπορείτε να χρησιμοποιήσετε την μη-αναδρομική προσέγγιση (δυναμικού προγραμματισμού) που περιγράφεται εδώ: <http://goo.gl/MzZSj>

Για την εύρεση της **απόστασης κατά LCSS** δύο ακολουθιών A και B,  $d_{LCSS}(A, B)$ , δίνεται ο πιο κάτω τύπος:

$$d_{LCSS}(A, B) = 1 - LCSS(A, B) / \min(|A|, |B|) \quad \text{Εξίσωση 3}$$

όπου  $\min(|A|, |B|)$  είναι το μήκος (σε στοιχεία) της μικρότερης ακολουθίας μεταξύ A και B. **Η απόσταση αυτή είναι η ελάχιστη.** Στο πιο πάνω παράδειγμα, οι δύο ακολουθίες A και B έχουν το ίδιο μήκος 8, οπότε  $\min(|A|, |B|) = 8$ .

Σημειώστε ότι κατά τη κλήση του ορίσματος `-similarity` ο χρήστης θα πρέπει να δίνει τουλάχιστον δύο αρχεία ήχου σαν είσοδο. Αν εισάγονται δύο αρχεία ήχου το πρόγραμμα θα τυπώνει τις αποστάσεις κατά Ευκλείδη και LCSS του δεύτερου αρχείου ήχου εν σχέση με το πρώτο αρχείο ήχου. Αν δίνονται πάνω από δύο αρχεία ήχου, τότε το πρόγραμμα θα τυπώνει τις αποστάσεις κατά Ευκλείδη και LCSS των αρχείων από το δεύτερο έως το τελευταίο εν σχέση με το πρώτο αρχείο ήχου ξεχωριστά. Αν για παράδειγμα έχουμε την πιο κάτω εντολή:

```
$. /wavengine -similarity sound1.wav sound2.wav sound3.wav
```

θα πρέπει να τυπώνονται οι αποστάσεις κατά Ευκλείδη και LCSS του `sound2.wav` με το `sound1.wav` και του `sound3.wav` με το `sound1.wav`.

Για μεγάλα αρχεία μια τέτοια λειτουργία μπορεί να απαιτεί ιδιαίτερα πολύ χρόνο (π.χ., 15 seconds πιο κάτω για το `Windows\Startup.wav` που έχει μέγεθος 41 KB)

```
time ./wavengine -similarity Windows\Startup.wav Windows\Startup.wav
```

```
Euclidean distance: 0.000
LCSS distance      : 0.000
```

```
real 0m15.359s
user 0m1.706s
sys 0m1.268s
```

### Λειτουργία 7: Κρυπτογράφηση ενός κειμένου μέσα σε αρχείο ήχου

Η λειτουργία αυτή κρύβει ένα κείμενο μέσα σε ένα αρχείο ήχου wav. Η διαδικασία κρυπτογράφησης περιγράφεται αναλυτικά πιο κάτω και απαιτεί γνώσεις τεχνικές χαμηλού επιπέδου προγραμματισμού με δυαδικούς τελεστές που θα **διδασθούμε στις διαλέξεις 18-19**.

Έστω ένα μήνυμα  $m$  (για σκοπούς επεξήγησης μια συμβολοσειράς) το οποίο θέλουμε να κρύψουμε σε ένα αρχείο ήχου και  $m_i$  το  $i$ -οστό byte αυτού του μηνύματος, και  $[m_i]_j$  το  $j$ -οστό bit αυτού του  $i$ -οστού byte (τα bytes είναι αριθμημένα ξεκινώντας από το 0, και τα bits αριθμούνται από το 7 για το πιο σημαντικό bit, στο 0 για το λιγότερο σημαντικό bit). Είναι εύκολο τότε να ορίσουμε μια σειρά  $u_n$  η οποία περιέχει τα ακόλουθα bits του μηνύματος  $m$ :

$$u_n = \begin{cases} \left\lfloor \frac{m_n}{8} \right\rfloor_{(7-(n \% 8))} & \text{εάν } n \in [0 \dots 8 \times \text{strlen}(m)] \\ 0 & \text{εάν } n \notin [0 \dots 8 \times \text{strlen}(m)] \end{cases}$$

Προσθέστε μια συνάρτηση `int getBit(char *m, int n);` στο πρόγραμμα η οποία υπολογίζει το  $u_n$  και να επιστρέφει είτε 0 ή 1 ανάλογα με τη τιμή του bit που ζητείται.

Παράδειγμα: Αν  $m = "ca"$   $\Rightarrow m_0 = c, m_1 = a$ . Στον πίνακα `ascii` ο χαρακτήρας  $c = 99_{10} = 01100011_2$  ο χαρακτήρας  $a = 97_{10} = 01100001_2$  κτλ. Οπότε έχουμε  $[m_0]_7=0, [m_0]_6=1, [m_0]_5=1, [m_0]_4=0, [m_0]_3=0, [m_0]_2=0, [m_0]_1=1, [m_0]_0=1, [m_1]_7=0, [m_1]_6=1, [m_1]_5=1, [m_1]_4=0, [m_1]_3=0, [m_1]_2=0, [m_1]_1=0, [m_1]_0=1$ . Οπότε η τιμή του  $u_0 = [m_0]_7=0$  ενώ  $u_9 = [m_1]_6=1$ .

**Για την κωδικοποίηση πρέπει να κρύψουμε την σειρά των bits  $u_n$  μέσα στα bytes του DATA SUB-CHUNK των αρχείων WAV.**

### Πίνακας μετάθεσης

Για να **MHN** τοποθετήσουμε το μήνυμα  $m$  σε συνεχόμενα bytes του αρχείου ήχου `wav` (εφόσον κάτι τέτοιο θα αποκάλυπτε εύκολα το κρυφό μήνυμα μέσα στο αρχείο `wav`), θα καθορίσουμε μια συνάρτηση ικανή για να κατασκευάσει μεταθέσεις (permutations) βασιζόμενα σε μια παράμετρο που θα ονομάζεται κλειδί συστήματος (*system-key-integer*), το οποίο πρέπει να ορίσετε στο πρόγραμμά σας ως σταθερά, δηλ., ένας ακέραιος της επιθυμίας σας. Η μετάθεση θα μας επιτρέψει να γνωρίζουμε ποια είναι τα bytes του αρχείου ήχου που θα αλλάξουμε, και με ποια σειρά. Αυτή η συνάρτηση  $f$  χρησιμοποιεί ένα πίνακα  $N$  ακέραιων το  $i$ -οστό στοιχείο του οποίου έχει την τιμή  $f(i)$ .

### Ορισμός συνάρτησης μετάθεσης:

Έστω  $K = [0 \dots N-1]$  ένα σύνολο ακέραιων αποθηκευμένα σε βοηθητικό πίνακα permutation, τότε:

$$f : K \longrightarrow K$$

$$\forall (x, y) \in K^2, f(x) = f(y) \iff x = y$$

Για να επιτύχουμε το στόχο μας, ορίζουμε την συνάρτηση  $f$  ως την πιο απλή μετάθεση:

$$f(x) = x.$$

Δηλαδή, στην  $i$ -οστή θέση του πίνακα θα δώσουμε την τιμή  $i$ . Μετά θα αλλάξουμε τυχαία τα στοιχεία του πίνακα χρησιμοποιώντας τον παρακάτω αλγόριθμο:

Κάνε  $N$  φορές

Διάλεξε «τυχαία» (με βάση το *system-key-integer*) δυο ακέραιους  $i$  και  $j$

Μετάτρεψε τους στο διάστημα  $[0 \dots N - 1]$

Αντάλλαξε τα στοιχεία στις θέσεις  $i$  και  $j$  του πίνακα

Το πρόβλημα είναι ότι πρέπει να δημιουργήσουμε ξανά τον ίδιο πίνακα της συνάρτησης μετάθεσης ανά πάσα στιγμή εάν το χρειαστούμε. Για να κάνετε αυτό αρκεί να χρησιμοποιήσετε την συνάρτηση `srand(<system-key-integer>)` και μετά να καλέσετε την συνάρτηση `rand()` η οποία επιστρέφει ψευδο-τυχαίους αριθμούς. Το πρότυπο της συνάρτησης θα είναι:

```
int* createPermutationFunction(int N, unsigned int systemkey);
```

## Εισαγωγή μηνύματος στο αρχείο ήχου

Η εισαγωγή των bits του μηνύματος  $m$  στο αρχείο  $wav$  μπορεί να γίνει ως εξής:

Για κάθε  $i$  από 0 στο  $(1 + \text{strlen}(m)) \times 8$

- υπολόγισε  $u = \text{getBit}(m, i)$
- υπολόγισε  $x = \text{permutation}[i]$ , το οποίο είναι το  $x$ -οστό byte των samples
- διέγραψε το bit μικρότερου βάρους του  $x$ -οστού byte του πίνακα των samples
- αντικατέστησε αυτό το bit που διαγράφηκε με την τιμή  $u$ .

Η τοποθέτηση  $(1 + \text{strlen}(m)) \times 8$  bits στο αρχείο ήχου, επιτρέπει την απόκρυψη όλων των bits του μηνύματος  $m$  (δηλαδή  $\text{strlen}(m) \times 8$  bits) συν το μηδέν (NUL character) για το τέλος της συμβολοσειράς.

Για τη λειτουργία αυτή έχουμε την πιο κάτω εντολή:

```
$/wavengine -encodeText sound1.wav inputText.txt
```

Το αποτέλεσμα της εκτέλεσης αυτής της λειτουργίας είναι η δημιουργία ενός αρχείου ήχου με το όνομα `new-sound1.wav` το οποίο είναι ένα αρχείο wav με ενσωματωμένο μέσα της το κείμενο του αρχείου `inputText.txt`. Το όνομα του δημιουργείται από το όνομα του αρχείου κάλυμμα προσθέτοντας το πρόθεμα "new-".

## Λειτουργία 8: Αποκρυπτογράφηση/Επανάκτηση μυστικού κειμένου από ένα αρχείο ήχου

Εάν ακολουθηθεί η αντιστροφή διαδικασία που περιγράφηκε πιο πάνω, με προηγούμενη γνώση το `<system-key-integer>`, τότε θα οδηγηθούμε από ένα κρυπτογραφημένο αρχείο ήχου στο αποκρυπτογραφημένο κείμενο που είναι το ίδιο το αρχικό κείμενο  $m$ . Για τη λειτουργία αυτή έχουμε την πιο κάτω εντολή:

```
$/wavengine -decodeText encryptedSound.wav msgLength output.txt
```

Το αποτέλεσμα της εκτέλεσης αυτής της λειτουργίας είναι η δημιουργία ενός αρχείου κειμένου με το μήνυμα που έχει ανακτηθεί από το κωδικοποιημένο αρχείο ήχου `encryptedSound.wav`. Για να υλοποιηθεί αυτή η λειτουργία, πρέπει να κάνετε την αντίθετη διαδικασία που κάνατε για την λειτουργία 7. Για να πάρετε το τελικό κείμενο, πρέπει να γνωρίζετε το μήκος του σε χαρακτήρες, το οποίο δίνεται στη γραμμή εντολής από την παράμετρο `msgLength` (εφόσον απαιτείται από την `createPermutationFunction` που πρέπει να εκτελέσετε εδώ επίσης). Το μήνυμα πρέπει να είναι το ίδιο ως προς τη μορφή του (ίδιοι χαρακτήρες) με το μήνυμα που κωδικοποιήθηκε στην λειτουργία 7.

## Λειτουργία 9: Επιπρόσθετη Λειτουργία

Σκεφτείτε μια ή περισσότερες απλές επιπρόσθετες λειτουργίες που θα έκαναν την βιβλιοθήκη σας πιο ενδιαφέρουσα. Οι λειτουργίες αυτές μπορεί να προκύψουν από αναζήτηση στο WWW. Οι λειτουργίες αυτές δύναται να λάβουν μέχρι 10 επιπρόσθετες μονάδες εάν υλοποιηθούν, παρουσιαστούν και τεκμηριωθούν με σχόλια.

**Οι επιπλέον αυτές λειτουργίες ΔΕΝ θα ληφθούν υπόψη για τον τελικό διαγωνισμό, ο οποίος θα στηριχτεί αποκλειστικά στις λειτουργίες 1 με 8.**

## VII. Υποδειγματικά Αρχεία & Ο Πίνακας ASCII

### as4-supplementary.zip (Μέγεθος σε bytes και Όνομα Αρχείου)

```

24 Windows Navigation Start.wav
40 Windows Feed Discovered.wav
48 Windows Information Bar.wav
48 Windows Ringout.wav
56 Windows Menu Command.wav
80 Windows Startup.wav
144 Speech Misrecognition.wav
144 Windows Default.wav
160 Windows User Account Control.wav
168 Windows Pop-up Blocked.wav
176 Windows Hardware Fail.wav
176 ringout.wav
192 Windows Hardware Remove.wav
192 Windows Recycle.wav
200 Windows Hardware Insert.wav
224 chord.wav
224 recycle.wav
232 ir_begin.wav
248 ir_end.wav
256 Speech Sleep.wav
256 Windows Minimize.wav
296 Speech On.wav
304 Speech Disambiguation.wav
304 Windows Restore.wav
312 Windows Critical Stop.wav
312 Windows Logon Sound.wav
336 Windows Error.wav
336 Windows Shutdown.wav
336 Windows_Error.wav
336 Windows_Shutdown.wav
352 Windows Balloon.wav
352 Windows Battery Low.wav
352 ir_inter.wav
376 Speech Off.wav
376 Windows Ding.wav
376 Windows Logoff Sound.wav
392 Windows Ringin.wav
424 Windows Battery Critical.wav
424 chimes.wav
448 Windows Notify.wav
448 notify.wav
472 Windows Exclamation.wav
560 tada.wav
696 Windows Print complete.wav
2368 piano.wav

```

### Πίνακας ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	ε#32;	Space	64	40	100	ε#64;	@	96	60	140	ε#96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	ε#33;	!	65	41	101	ε#65;	A	97	61	141	ε#97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	ε#34;	"	66	42	102	ε#66;	B	98	62	142	ε#98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	ε#35;	#	67	43	103	ε#67;	C	99	63	143	ε#99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	ε#36;	\$	68	44	104	ε#68;	D	100	64	144	ε#100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	ε#37;	%	69	45	105	ε#69;	E	101	65	145	ε#101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	ε#38;	&	70	46	106	ε#70;	F	102	66	146	ε#102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	ε#39;	'	71	47	107	ε#71;	G	103	67	147	ε#103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	ε#40;	(	72	48	110	ε#72;	H	104	68	150	ε#104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	ε#41;	)	73	49	111	ε#73;	I	105	69	151	ε#105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	ε#42;	*	74	4A	112	ε#74;	J	106	6A	152	ε#106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	ε#43;	+	75	4B	113	ε#75;	K	107	6B	153	ε#107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	ε#44;	,	76	4C	114	ε#76;	L	108	6C	154	ε#108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	ε#45;	-	77	4D	115	ε#77;	M	109	6D	155	ε#109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	ε#46;	.	78	4E	116	ε#78;	N	110	6E	156	ε#110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	ε#47;	/	79	4F	117	ε#79;	O	111	6F	157	ε#111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	ε#48;	0	80	50	120	ε#80;	P	112	70	160	ε#112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	ε#49;	1	81	51	121	ε#81;	Q	113	71	161	ε#113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	ε#50;	2	82	52	122	ε#82;	R	114	72	162	ε#114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	ε#51;	3	83	53	123	ε#83;	S	115	73	163	ε#115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	ε#52;	4	84	54	124	ε#84;	T	116	74	164	ε#116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	ε#53;	5	85	55	125	ε#85;	U	117	75	165	ε#117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	ε#54;	6	86	56	126	ε#86;	V	118	76	166	ε#118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	ε#55;	7	87	57	127	ε#87;	W	119	77	167	ε#119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	ε#56;	8	88	58	130	ε#88;	X	120	78	170	ε#120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	ε#57;	9	89	59	131	ε#89;	Y	121	79	171	ε#121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	ε#58;	:	90	5A	132	ε#90;	Z	122	7A	172	ε#122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	ε#59;	;	91	5B	133	ε#91;	[	123	7B	173	ε#123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	ε#60;	<	92	5C	134	ε#92;	\	124	7C	174	ε#124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	ε#61;	=	93	5D	135	ε#93;	]	125	7D	175	ε#125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	ε#62;	>	94	5E	136	ε#94;	^	126	7E	176	ε#126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	ε#63;	?	95	5F	137	ε#95;	_	127	7F	177	ε#127;	DEL

Source: [www.LookupTables.com](http://www.LookupTables.com)

## VIII. Γενικές Οδηγίες

Το πρόγραμμα σας θα πρέπει να συμβαδίζει με το πρότυπο ISO C, να περιλαμβάνει εύστοχα και περιεκτικά σχόλια, να έχει καλή στοίχιση και το όνομα κάθε μεταβλητής, σταθεράς, ή συνάρτησης να είναι ενδεικτικό του ρόλου της. **Να χρησιμοποιήσετε το λογισμικό τεκμηρίωσης doxygen** έτσι ώστε να μπορούμε να μετατρέψουμε τα σχόλια του προγράμματός σας σε HTML αρχεία και να τα δούμε με ένα browser. Η συστηματική αντιμετώπιση της λύσης ενός προβλήματος περιλαμβάνει στο παρόν στάδιο τη διάσπαση του προβλήματος σε μικρότερα ανεξάρτητα προβλήματα που κατά κανόνα κωδικοποιούμε σε ξεχωριστές συναρτήσεις. Για αυτό τον λόγο σας καλούμε να κάνετε χρήση συναρτήσεων και άλλων τεχνικών δομημένου προγραμματισμού που διδαχτήκατε στο ΕΠΛ131. Επίσης, σας θυμίζουμε ότι κατά την διάρκεια της εκτέλεσης του προγράμματος σας αυτό θα πρέπει να δίνει τα κατάλληλα μηνύματα σε περίπτωση λάθους. Τέλος το πρόγραμμα σας θα πρέπει να μεταγλωττίζεται στις μηχανές του εργαστηρίου. **Παρακαλώ όπως μελετηθούν ξανά οι κανόνες υποβολής εργασιών όπως αυτοί ορίζονται στο συμβόλαιο του μαθήματος.** Επίσης να ακολουθήσετε τα πιο κάτω βήματα όταν υποβάλετε την άσκηση σας στο Moodle:

- Δημιουργείτε ένα κατάλογο με το όνομά σας π.χ PavlosAntoniou/ χωρίς να αφήνετε κενά στο όνομα του καταλόγου.
- Βάλτε μέσα στον κατάλογο αυτό όλα τα αρχεία της εργασίας που πρέπει να υποβάλετε.
- Συμπίεστε (zip) τον κατάλογο (και όχι τα αρχεία ξεχωριστά) χρησιμοποιώντας την εντολή  
`zip -r PavlosAntoniou.zip PavlosAntoniou/`
- Ανεβάστε στο Moodle το συμπιεσμένο αρχείο π.χ. PavlosAntoniou.zip

## Εξέταση Εργασίας και Τελικός Διαγωνισμός

Κατά την διάρκεια του τελευταίου εργαστηρίου θα γίνει η εξέταση της εργασίας κατά το οποίο θα πρέπει να γίνει η επίδειξη της σχεδίασης και υλοποίησης της βιβλιοθήκης και του πελάτη σας απ' όλα τα μέλη της ομάδας. Τυχούσα παράληψη παρουσίασης της εργασίας ενδέχεται να συνεπάγεται τον μηδενισμό της εργασίας. Στοιχεία τα οποία ληφθούν υπόψη στην αξιολόγηση της εργασίας σας περιλαμβάνουν: ορθότητα λειτουργίας, στοιχεία επίδοσης: π.χ., ελαχιστοποίηση του χρόνου απόκρισης σε αιτήσεις, το οποίο ορίζεται ως το διάστημα μεταξύ της χρονικής στιγμής που γίνεται η αίτηση και της στιγμής που διεκπεραιώνεται μια λειτουργία, σχεδίαση μονάδων λογισμικού και της βιβλιοθήκης ευρύτερα, σωστή διαχείριση μνήμης (μηδενικές διαρροές μνήμης μέσω valgrind), σωστά μηνύματα λάθους και GPL προοίμιο όπου ορίζει η εκφώνηση.

## IX. Κριτήρια αξιολόγησης:

Υλοποίηση Πελάτη	8
Λειτουργία 1 Βιβλιοθήκης (List)	10
Λειτουργία 2 Βιβλιοθήκης (Stereo to Mono)	7
Λειτουργία 3 Βιβλιοθήκης (Mix)	7
Λειτουργία 4 Βιβλιοθήκης (Chop)	7
Λειτουργία 5 Βιβλιοθήκης (Reverse)	6
Λειτουργία 6 Βιβλιοθήκης (Similarity)	10
Λειτουργία 7 Βιβλιοθήκης (Encode)	12
Λειτουργία 8 Βιβλιοθήκης (Decode)	8
Γενική εικόνα (στοιχισμένος και ευανάγνωστος κώδικας, εύστοχα ονόματα μεταβλητών, σταθερών και συναρτήσεων, σχολιασμός)	10
GPL Προοίμια, Σχεδίαση Προγράμματος, Χρήση SVN, gprof report, valgrind report	15
Επιπλέον Λειτουργίες	10
Σύνολο	110

Παρακαλώ όπως μελετηθούν ξανά οι κανόνες υποβολής εργασιών όπως αυτοί ορίζονται στο συμβόλαιο του μαθήματος.

**Καλή Επιτυχία!**