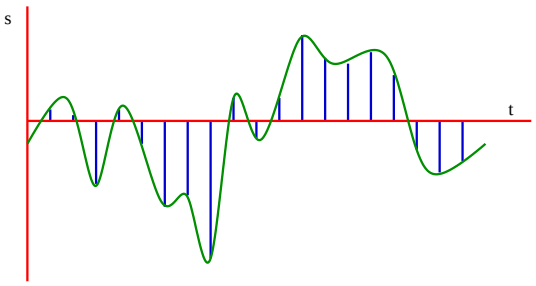
ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2020-21)

Εργασία 2

Ο ήχος είναι ένα αναλογικό σήμα, δηλαδή, από μαθηματική σκοπιά, μπορεί να θεωρηθεί σαν μία συνάρτηση ενός μεγέθους ως προς τον χρόνο. Το μέγεθος αυτό μπορεί να είναι η πυκνότητα του αέρα που διεγείρει το τύμπανο ενός αυτιού, το ηλεκτρικό σήμα που παράγεται από την κεφαλή ενός πικάπ όταν αυτή διατρέχει ένα αυλάκι ενός δίσκου βινυλίου, κλπ. Όμως, για να αποθηκευθεί ο ήχος σε ένα υπολογιστή, αυτό δεν μπορεί να γίνει στην αναλογική του μορφή, αλλά πρέπει να ψηφιοποιηθεί. Η ψηφιοποίηση αυτή γίνεται μέσω της καταγραφής της τιμής της συνάρτησης του σήματος σε διακριτές ισαπέχουσες χρονικές στιγμές, όπως φαίνεται στο διπλανό σχήμα. Οι τιμές της συνάρτησης μπορεί να θεωρηθεί ότι αντιστοιχούν στην ένταση του ήχου στις αντίστοιχες χρονικές στιγμές.



Υπό την προϋπόθεση ότι η καταγραφή των διαδοχικών τιμών της συνάρτησης είναι αρκετά πυκνή, παρότι δεν έχουμε πλέον στη διάθεσή μας το αρχικό αναλογικό σήμα, η αναπαραγωγή του ήχου από την ψηφιοποιημένη μορφή του μπορεί να είναι επαρκώς πιστή στην αρχική αναλογική εκδοχή του. Για παράδειγμα, η τυπική συχνότητα δειγματοληψίας σε ένα μουσικό CD είναι 44.1kHz, δηλαδή έχουν καταγραφεί 44100 δεδομένα για κάθε δευτερόλεπτο μουσικής.

Υπάρχουν διάφορα πρότυπα για την αποθήκευση ψηφιοποιημένων ήχων σε αρχεία, όπως, για παράδειγμα, wav, mp3, aac, m4a, wma, κλπ. Αντικείμενο αυτής της εργασίας είναι η επεξεργασία αρχείων wav. Πριν προχωρήσουμε στα ζητούμενα της εργασίας, θα πρέπει να γίνει λεπτομερής περιγραφή του format των αρχείων wav που θα πρέπει να είναι σε θέση να διαχειρίζεται το πρόγραμμα που θα γράψετε.

Η απλούστερη δυνατή εκδοχή του format των αρχείων wav, θα χρησιμοποιηθεί στα πλαίσια της εργασίας, περιγράφεται στη συνέχεια. Ένα αρχείο wav περιέχει κατά σειρά:

- 4 bytes με τους χαρακτήρες "`RIFF`" (αρχικά του Resource Interchange File Format, που είναι το πρότυπο που χρησιμοποιείται για τη δομή των wav αρχείων).
- 4 bytes, συμβολικά *SizeOfFile*, για το μέγεθος του αρχείου, που στην πραγματικότητα είναι το πλήθος των bytes που ακολουθούν μέχρι το τέλος του αρχείου. Ουσιαστικά, είναι το μέγεθος του αρχείου στον δίσκο μείον 8 bytes, δηλαδή αυτά που καταλαμβάνουν το τρέχον πεδίο και το προηγούμενο ("`RIFF`"). Σημειώνεται ότι η αναπαράσταση ακεραίων χωρίς πρόσημο γίνεται κατά σειρά από το λιγότερο σημαντικό byte προς το περισσότερο σημαντικό (πρότυπο little-endian). Δηλαδή, ο ακέραιος `0xA3870FB9` αναπαρίσταται κατά σειρά με τα bytes `0xB9`, `0x0F`, `0x87`, `0xA3`.
- 4 bytes με τους χαρακτήρες "`WAVE`", που υποδεικνύουν τη συγκεκριμένη υποκατηγορία αρχείων που ακολουθούν το πρότυπο RIFF.
- 4 bytes με τους χαρακτήρες "`fmt `" (προσοχή στον τελευταίο χαρακτήρα κενού διαστήματος), που υποδεικνύουν την έναρξη του τμήματος format (format chunk) του αρχείου.
- 4 bytes για τον χώρο που καταλαμβάνουν τα δεδομένα του τμήματος format που θα ακολουθήσουν. Στα αρχεία που θα χρησιμοποιηθούν στην εργασία, η τιμή αυτού του πεδίου θα είναι

πάντα 16 (`0x00000010`). Και εδώ εφαρμόζεται το πρότυπο little-endian, δηλαδή ο αριθμός αυτός αποθηκεύεται με την ακολουθία από bytes `0x10`, `0x00`, `0x00`, `0x00`.

- 2 bytes για τον τύπο του WAVE format. Για την εργασία, αυτός θα ισούται πάντα με 1 (`0x0001`) και θα αποθηκεύεται κατά little-endian, δηλαδή με τα bytes `0x01`, `0x00`.

- 2 bytes, συμβολικά *MonoStereo*, για το αν ο ήχος είναι μονοφωνικός (τιμή `0x0001`) ή στερεοφωνικός με δύο κανάλια (τιμή `0x0002`). Προσοχή στο little-endian, όπως και σε όλες τις καταχωρήσεις ακεραίων που ακολουθούν.

- 4 bytes, συμβολικά *SampleRate*, για το ρυθμό δειγματοληψίας, δηλαδή πόσες τιμές, ανά δευτερόλεπτο, της συνάρτησης του αναλογικού ήχου έχουν καταγραφεί στο αρχείο.

- 4 bytes, συμβολικά *BytesPerSec*, για το πλήθος bytes ανά δευτερόλεπτο ήχου, που είναι καταχωρημένα στο αρχείο.

- 2 bytes, συμβολικά *BlockAlign*, για το πλήθος των bytes που απαιτούνται για την καταχώρηση της πληροφορίας του ήχου σε μία χρονικά στιγμή, για όλα τα κανάλια. Σημειώστε ότι πάντοτε θα πρέπει να ισχύει ότι $BytesPerSec = SampleRate \times BlockAlign$.

- 2 bytes, συμβολικά *BitsPerSample*, για το πλήθος των bits που απαιτούνται για την καταχώρηση της πληροφορίας του ήχου σε μία χρονικά στιγμή, για ένα μόνο κανάλι. Στα πλαίσια της εργασίας, αυτή η τιμή θα είναι είτε 8 (`0x0008`), είτε 16 (`0x0010`). Για $BitsPerSample = 8$, η ένταση του ήχου είναι ένας ακέραιος χωρίς πρόσημο στο διάστημα `[0,255]`, ενώ για $BitsPerSample = 16$, η ένταση του ήχου είναι ένας ακέραιος με πρόσημο στο διάστημα `[-32768,32767]`. Σημειώστε ότι πάντοτε θα πρέπει να ισχύει ότι $BlockAlign = BitsPerSample/8 \times MonoStereo$.

- 4 bytes με τους χαρακτήρες "`data`", που υποδεικνύουν την έναρξη του τμήματος data (data chunk) του αρχείου.

- 4 bytes, συμβολικά *SizeOfData*, για τον χώρο που καταλαμβάνουν τα δεδομένα του τμήματος data που θα ακολουθήσουν.

- Τα ψηφιοποιημένα δεδομένα του ήχου με χρονική σειρά. Στην περίπτωση στερεοφωνικού ήχου με δύο κανάλια, για κάθε χρονική στιγμή, υπάρχει ένα ζευγάρι δεδομένων, με το πρώτο στοιχείο να αντιστοιχεί στο αριστερό κανάλι και το δεύτερο στοιχείο στο δεξιό κανάλι.

- Μετά το τέλος των δεδομένων ήχου, ενδεχομένως να υπάρχουν και άλλα τμήματα που προβλέπονται για την κατηγορία WAVE, με τα οποία όμως δεν θα ασχοληθούμε στην εργασία.

Δείτε στη συνέχεια τα περιεχόμενα ενός νόμιμου αρχείου wav σε δεκαεξαδική μορφή:[1]

```
$ od -tx1 good.wav
0000000 52 49 46 46 85 00 00 00 57 41 56 45 66 6d 74 20
0000020 10 00 00 00 01 00 01 00 44 ac 00 00 88 58 01 00
0000040 02 00 10 00 64 61 74 61 58 00 00 00 00 00 d4 07
0000060 a1 0f 5e 17 04 1f 8a 26 ea 2d 1c 35 18 3c d7 42
0000100 54 49 86 4f 69 55 f6 5a 27 60 f8 64 63 69 64 6d
0000120 f7 70 18 74 c5 76 fa 78 b6 7a f6 7b b9 7c ff 7c
```

[1] Για να δείτε τα περιεχόμενα αρχείων με δυαδική κωδικοποίηση, μπορείτε να χρησιμοποιήσετε την εντολή `od` σε Unix-like συστήματα (Linux, MacOS, WSL, κλπ.). Για περιβάλλοντα Windows, υπάρχουν ελεύθερα διαθέσιμα προγράμματα που λειτουργούν παρόμοια με την εντολή `od` (π.χ. `https://www.hhdsoftware.com/free-hex-editor`).

```
0000140 c8 7c 12 7c e0 7a 33 79 0b 77 6c 74 58 71 d1 6d
0000160 dd 69 7e 65 b8 60 92 5b 0f 56 36 50 0c 4a 97 43
0000200 df 3c ea 35 45 78 74 72 61 44 61 74 61
0000215
$
```

Αντικείμενο της παρούσας εργασίας είναι να γράψετε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται `wavproc.c`), το οποίο θα διαβάζει με την `getchar` από την είσοδο δεδομένα ήχου που ακολουθούν το πρότυπο wav που περιγράφηκε προηγουμένως και θα εκτελούν κάποια επεξεργασία επάνω στα δεδομένα αυτά. Το ποια ακριβώς επεξεργασία θα εκτελεί το πρόγραμμά σας θα καθορίζεται από την τιμή μίας συμβολικής σταθεράς `MODE` που θα ορίσετε σε αυτό. Οι ζητούμενες επεξεργασίες, ανάλογα με την τιμή της `MODE` είναι οι εξής:

`#define MODE 1` (100% της βαθμολογίας της εργασίας)

Το πρόγραμμα θα πρέπει να ελέγχει αν τα δεδομένα που διαβάστηκαν ακολουθούν το πρότυπο wav που περιγράφηκε προηγουμένως και να εκτυπώνει τις σχετικές πληροφορίες του ήχου (χωρίς τα δεδομένα αυτά καθεαυτά). Ένα παράδειγμα εκτέλεσης είναι το εξής:[2]

```
$ ./wavproc < good.wav
size of file: 133
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 16
size of data chunk: 88
$
$ ./wavproc < 1MB.wav
size of file: 1073210
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 8000
bytes/sec: 32000
block alignment: 4
bits/sample: 16
size of data chunk: 1072948
$
$ ./wavproc < 2MB.wav
size of file: 2146158
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 16000
```

[2] Όλα τα αρχεία που φαίνονται στα παραδείγματα της εκφώνησης της εργασίας μπορείτε να τα κατεβάσετε από το `http://cgi.di.uoa.gr/~ip/hwfiles/wavproc`.

```
bytes/sec: 64000
block alignment: 4
bits/sample: 16
size of data chunk: 2145896
$
$ ./wavproc < 5MB.wav
size of file: 5226758
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 176400
block alignment: 4
bits/sample: 16
size of data chunk: 5226496
$
$ ./wavproc < 10MB.wav
size of file: 10406730
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 176400
block alignment: 4
bits/sample: 16
size of data chunk: 10406468
$
```

Οι εκτυπώσεις του προγράμματος θα πρέπει να γίνουν με τη συνάρτηση `fprintf` της πρότυπης βιβλιοθήκης εισόδου-εξόδου της C, αντί της `printf`. Ενώ η `printf` εκτυπώνει στο πρότυπο ρεύμα εξόδου (stdout), η `fprintf` μπορεί να εκτυπώσει σε οποιοδήποτε ρεύμα εξόδου, που μπορεί να αντιστοιχεί σε κάποιο αρχείο ή στο πρότυπο ρεύμα εξόδου για διαγνωστικά μηνύματα (stderr), στο οποίο τελικά θα πρέπει να γίνονται οι εκτυπώσεις από το πρόγραμμά σας. Ο τρόπος χρήσης της `fprintf`, για τη συγκεκριμένη περίπτωση, είναι ο εξής:

`fprintf(stderr,` <ίδια ακριβώς ορίσματα με αυτά της `printf`>`);`

Το πρόγραμμά σας πρέπει να είναι σε θέση να καταλαβαίνει λάθη που υπάρχουν στα δεδομένα ήχου που διαβάστηκαν και να σταματά στο πρώτο λάθος που βρίσκει εκτυπώνοντας το κατάλληλο διαγνωστικό μήνυμα. Παραδείγματα εκτέλεσης σε προβληματικά δεδομένα εισόδου είναι τα ακόλουθα:

```
$ ./wavproc < bad_riff.wav
Error! "RIFF" not found
$
$ ./wavproc < bad_wave.wav
size of file: 133
Error! "WAVE" not found
$
$ ./wavproc < bad_fmt.wav
size of file: 133
```

```
Error! "fmt " not found
$
$ ./wavproc < bad_sfc.wav
size of file: 133
size of format chunck: 18
Error! size of format chunck should be 16
$
$ ./wavproc < bad_wtf.wav
size of file: 133
size of format chunck: 16
WAVE type format: 2
Error! WAVE type format should be 1
$
$ ./wavproc < bad_ms.wav
size of file: 133
size of format chunck: 16
WAVE type format: 1
mono/stereo: 3
Error! mono/stereo should be 1 or 2
$
$ ./wavproc < bad_bys.wav
size of file: 133
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88202
block alignment: 2
Error! bytes/second should be sample rate x block alignment
$
$ ./wavproc < bad_bis.wav
size of file: 133
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 32
Error! bits/sample should be 8 or 16
$
$ ./wavproc < bad_ba.wav
size of file: 133
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 88200
block alignment: 2
```

```
bits/sample: 16
Error! block alignment should be bits per sample / 8 x mono/stereo
$
$ ./wavproc < bad_data.wav
size of file: 133
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 16
Error! "data" not found
$
$ ./wavproc < bad_insfd.wav
size of file: 133
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 16
size of data chunk: 88
Error! insufficient data
$
$ ./wavproc < bad_sf.wav
size of file: 128
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 16
size of data chunk: 88
Error! bad file size
$
```

`#define MODE 2` (Bonus βαθμολογία 10%)

Επεκτείνετε το πρόγραμμά σας, ώστε γι΄ αυτή την περίπτωση λειτουργίας, εκτός από τον έλεγχο ορθότητας των δεδομένων εισόδου, ως προς τη συμφωνία τους με το πρότυπο wav, να μεταφέρει στην έξοδο, με τη βοήθεια της συνάρτησης `putchar`, τα δεδομένα του ήχου, έχοντας υποδιπλασιάσει την ταχύτητα αναπαραγωγής του, χωρίς να αλλοιώσει τα δείγματα που αποτελούν τον ψηφιοποιημένο ήχο. Είναι αυτονόητο ότι ο ήχος στην έξοδο θα αναπαράγεται στον διπλάσιο χρόνο από τον αρχικό. Σκεφτείτε ποιες παράμετροι των δεδομένων εισόδου θα πρέπει να αλλάξουν και πώς, ώστε να επιτευχθεί το ζητούμενο. Σε κάθε περίπτωση, η έξοδος του προγράμματος πρέπει να είναι μία νόμιμη ακολουθία από δεδομένα που συνιστούν έναν ήχο με βάση το πρότυπο που έχει περιγραφεί. Ακόμα

και αν υπάρχουν στην είσοδο επιπλέον bytes μετά το τμήμα των δεδομένων του, θα πρέπει και αυτά να μεταφερθούν στην έξοδο, προφανώς αυτούσια.

Σημειώνεται ότι επεκτείνετε σταδιακά το ίδιο πηγαίο αρχείο `wavproc.c`, αλλά μπορείτε να παράγετε κάθε φορά διαφορετικό εκτελέσιμο, ανάλογα με την τιμή στην οποία έχει αντιστοιχηθεί η συμβολική σταθερά `MODE`. Το εκτελέσιμο `wavpoc` που χρησιμοποιήθηκε στα παραδείγματα εκτέλεσης της προηγούμενης ενότητας είχε προκύψει από μεταγλώττιση με την τιμή 1 για τη σταθερά. Στα παραδείγματα εκτέλεσης που φαίνονται στη συνέχεια, το εκτελέσιμο `wavproc2` προέκυψε από τη μεταγλώττιση του ίδιο πηγαίου αρχείου, αλλά με την τιμή 2 στη συμβολική σταθερά. Αντίστοιχη λογική ισχύει και για τις περιπτώσεις λειτουργίας του προγράμματος στις επόμενες ενότητες. Παραδείγματα εκτέλεσης για την περίπτωση υποδιπλασιασμού της ταχύτητας αναπαραγωγής του ήχου:[3]

```
$ ./wavproc2 < LaughEvil.wav > LaughEvil2.wav
size of file: 329266
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 16
size of data chunk: 329230
$
$ ./wavproc2 < ThePinkPantherTheme.wav > ThePinkPantherTheme2.wav
size of file: 26032440
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 176400
block alignment: 4
bits/sample: 16
size of data chunk: 26032404
$
$ ./wavproc2 < la.wav > la2.wav
size of file: 88236
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 16
```

[3]Στις ενότητες που ακολουθούν, η έξοδος του προγράμματος ανακατευθύνεται σε ένα νέο αρχείο ήχου. Αν θέλετε να αναπαραγάγετε αρχεία ήχου, ο εύκολος τρόπος σε κάθε υπολογιστή και λειτουργικό σύστημα είναι να κάνετε διπλό κλικ στο αρχείο. (Σχεδόν) πάντοτε υπάρχει για τα αρχεία ήχου μία εφαρμογή που είναι συνυφασμένη με αυτά, ώστε με διπλό κλικ στα αρχεία να γίνεται η αναπαραγωγή που θέλουμε. Σε κάποια συστήματα, η αναπαραγωγή αρχείων ήχου μπορεί να γίνει και με κατάλληλη εντολή στο τερματικό, για παράδειγμα την `play` για Linux συστήματα, ή την `afplay` για MacOS. Μάλιστα, σε `Linux` μπορεί να αποφευχθεί η δημιουργία αρχείου με το τελικό αποτέλεσμα, αν απλώς κάνουμε σωλήνωση. Για παράδειγμα, "`./wavproc2 la.wav | play -`". Ή, ακόμα πιο απλά, για ένα απομακρυσμένο αρχείο ήχου: "`curl -s http://cgi.di.uoa.gr/~ip/hwfiles/wavproc/la.wav | ./wavproc2 | play -`".

```
size of data chunk: 88200
$
$ ./wavproc2 < 8bitstereo.wav > 8bitstereo2.wav
size of file: 176436
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 8
size of data chunk: 176400
$
```

#define MODE 3 **(Bonus βαθμολογία 5%)**

Υλοποιήστε αντίστοιχη λειτουργία με αυτήν της προηγούμενης ενότητας, όπου ο ήχος στην είσοδο θα μεταφέρεται στην έξοδο με διπλασιασμένη την ταχύτητα αναπαραγωγής του. Προφανώς, ο χρόνος αναπαραγωγής θα υποδιπλασιασθεί, αφού τα δείγματα που αποτελούν τον ψηφιοποιημένο ήχο θα πρέπει και πάλι να μεταφερθούν αναλλοίωτα. Παραδείγματα εκτέλεσης για το `wavproc3`:

```
$ ./wavproc3 < LaughSarc.wav > LaughSarc3.wav
size of file: 1541540
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 48000
bytes/sec: 192000
block alignment: 4
bits/sample: 16
size of data chunk: 1541504
$
$ ./wavproc3 < ThePinkPantherTheme.wav > ThePinkPantherTheme3.wav
size of file: 26032440
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 176400
block alignment: 4
bits/sample: 16
size of data chunk: 26032404
$
$ ./wavproc3 < narc.wav > narc3.wav
size of file: 4956196
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
```

```
bytes/sec: 176400
block alignment: 4
bits/sample: 16
size of data chunk: 4956160
$
$ ./wavproc3 < 8bitmono.wav > 8bitmono3.wav
size of file: 44136
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 44100
block alignment: 1
bits/sample: 8
size of data chunk: 44100
$
```

#define MODE 4 **(Bonus βαθμολογία 15%)**

Επεκτείνετε το πρόγραμμά σας, ώστε γι΄ αυτή την περίπτωση λειτουργίας, εκτός από τον έλεγχο ορθότητας των δεδομένων εισόδου, ως προς τη συμφωνία τους με το πρότυπο wav, να μεταφέρει στην έξοδο, με τη βοήθεια της συνάρτησης `putchar`, τα δεδομένα μόνο του αριστερού καναλιού του ήχου, στην περίπτωση που ήχος στην είσοδο είναι στερεοφωνικός. Αν ο ήχος στην είσοδο είναι μονοφωνικός, να μην προκύπτει σφάλμα, αλλά απλώς να μεταφέρεται στην έξοδο το μοναδικό κανάλι της εισόδου. Σκεφτείτε ποιες παράμετροι των δεδομένων εισόδου θα πρέπει να αλλάξουν και πώς και ποια από τα bytes του τμήματος δεδομένων της εισόδου θα πρέπει να μεταφερθούν στην έξοδο, ώστε να επιτευχθεί το ζητούμενο. Σε κάθε περίπτωση, η έξοδος του προγράμματος πρέπει να είναι μία νόμιμη ακολουθία από δεδομένα που συνιστούν έναν ήχο με βάση το πρότυπο που έχει περιγραφεί. Ακόμα και αν υπάρχουν στην είσοδο επιπλέον bytes μετά το τμήμα των δεδομένων του, θα πρέπει και αυτά να μεταφερθούν στην έξοδο, προφανώς αυτούσια.

Παραδείγματα εκτέλεσης για την περίπτωση εξαγωγής του αριστερού καναλιού από στερεοφωνικό ήχο:

```
$  ./wavproc4 < 8bitstereo.wav > 8bitstereo4.wav
size of file: 176436
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 8
size of data chunk: 176400
$
$ ./wavproc4 < CaliforniaDreamin.wav > CaliforniaDreamin4.wav
size of file: 30231108
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
```

```
sample rate: 48000
bytes/sec: 192000
block alignment: 4
bits/sample: 16
size of data chunk: 30230972
$
```

#define MODE 5 (Bonus βαθμολογία 5%)

Υλοποιήστε αντίστοιχη λειτουργία με αυτήν της προηγούμενης ενότητας, όπου στην έξοδο θα μεταφέρεται το δεξί κανάλι της εισόδου, στην περίπτωση στερεοφωνικού ήχου στην είσοδο, ή αυτούσιο το μοναδικό κανάλι, στην περίπτωση μονοφωνικού ήχου. Παραδείγματα εκτέλεσης για το `wavproc5`:

```
$ ./wavproc5 < 8bitstereo.wav > 8bitstereo5.wav
size of file: 176436
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 8
size of data chunk: 176400
$
$ ./wavproc5 < CaliforniaDreamin.wav > CaliforniaDreamin5.wav
size of file: 30231108
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 48000
bytes/sec: 192000
block alignment: 4
bits/sample: 16
size of data chunk: 30230972
$
```

#define MODE 6 (Bonus βαθμολογία 15%)

Επεκτείνετε το πρόγραμμά σας, ώστε γι΄ αυτή την περίπτωση λειτουργίας, εκτός από τον έλεγχο ορθότητας των δεδομένων εισόδου, ως προς τη συμφωνία τους με το πρότυπο wav, να μεταφέρει στην έξοδο, με τη βοήθεια της συνάρτησης `putchar`, τον ήχο της εισόδου στην έξοδο, αλλά με μειωμένη την έντασή του στο 1/8 της αρχικής. Παραδείγματα εκτέλεσης για το `wavproc6`:

```
$ ./wavproc6 < 8bitmono.wav > 8bitmono6.wav
size of file: 44136
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
```

```
sample rate: 44100
bytes/sec: 44100
block alignment: 1
bits/sample: 8
size of data chunk: 44100
$
$ ./wavproc6 < 8bitstereo.wav > 8bitstereo6.wav
size of file: 176436
size of format chunck: 16
WAVE type format: 1
mono/stereo: 2
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 8
size of data chunk: 176400
$
$ ./wavproc6 < la.wav > la6.wav
size of file: 88236
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 16
size of data chunk: 88200
$
```

`#define MODE 7` (Bonus βαθμολογία 15%)

Επεκτείνετε το πρόγραμμά σας, ώστε γι΄ αυτή την περίπτωση λειτουργίας, να μην διαβάζει δεδομένα από την είσοδο, αλλά να παράγει στην έξοδο δεδομένα τύπου wav, με βάση τον εξής μαθηματικό τύπο:[4]

$$f(t) = mv \cdot \sin(2\pi f_c t - m_i \cdot \sin(2\pi f_m t))$$

Για το σκοπό αυτό, ορίστε μία συνάρτηση C με πρωτότυπο το

```
void mysound(int dur, int sr, double fm, double fc, double mi, double mv);
```

η οποία να καλείται με κατάλληλες τιμές στα ορίσματά της από τη `main` συνάρτηση (όταν το `MODE` είναι `7`), ώστε να παράγει τον κατάλληλο ήχο με βάση τον μαθηματικό τύπο που δόθηκε. Το όρισμα `dur` είναι η διάρκεια του ήχου σε δευτερόλεπτα, το `sr` είναι το *SampleRate* για την ψηφιοποίηση του ήχου και τα ορίσματα `fm`, `fc`, `mi`, `mv` είναι οι παράμετροι που φαίνονται στον τύπο (για το π μπορείτε να χρησιμοποιήσετε τη συμβολική σταθερά `M_PI`, που είναι ορισμένη στο `math.h`). Ο ήχος που θα παραχθεί να είναι μονοφωνικός ($MonoStereo = 1$) και να αναπαρίσταται με 2 bytes ανά δείγμα ($BitsPerSample = 16$).

Καλέστε τη συνάρτηση από τη `main` με ορίσματα κατά σειρά `dur = 3`, `sr = 44100`, `fm = 2.0`, `fc = 1500.0`, `mi = 100.0`, `mv = 30000.0`. Τι ήχος παράγεται; Ενδεικτική εκτέλεση:

[4] Το π είναι το γνωστό 3.14... και με το sin συμβολίζεται η συνάρτηση του ημιτόνου.

```
$ ./wavproc7 > mysound.wav
$
$ ./wavproc < mysound.wav
size of file: 264636
size of format chunck: 16
WAVE type format: 1
mono/stereo: 1
sample rate: 44100
bytes/sec: 88200
block alignment: 2
bits/sample: 16
size of data chunk: 264600
$
```

Σημειώσεις/Απαγορεύσεις:

1. Στο πρόγραμμα που θα παραδώσετε, να έχετε βάλει στην αρχή σε σχόλιο ποιες από τις περιπτώσεις `MODE` έχετε υλοποιήσει. Επίσης, να έχετε σε σχόλιο τον ορισμό της συμβολικής σταθεράς `MODE`.

2. Για να συγκρίνετε δύο αρχεία, μπορείτε να χρησιμοποιήσετε σε Unix-like συστήματα (Linux, MacOS, WSL, κλπ.) την εντολή `cmp`.

3. Στην εργασία αυτή απαγορεύονται αυστηρά η χρήση πινάκων (συμπεριλαμβανομένων και των συμβολοσειρών), δεικτών και συναρτήσεων της μαθηματικής βιβλιοθήκης της C, εκτός από τη συνάρτηση του ημιτόνου (`sin`). Επίσης, απαγορεύεται η χρήση συναρτήσεων της βιβλιοθήκης εισόδου-εξόδου της C που διαχειρίζονται αρχεία.

4. Στην περίπτωση που αναπτύξετε τον κώδικα της εργασίας σε Windows μέσω του Dev-C++, επειδή υπάρχει ένα πρόβλημα στο περιβάλλον αυτό με τη διαχείριση της εισόδου και της εξόδου, θα πρέπει αμέσως μετά τη γραμμή `#include <stdio.h>` να προσθέσετε και τις:

   ```
   #ifdef WIN32
   #include <io.h>
   #include <fcntl.h>
   #endif
   ```

   Και στην αρχή της `main()` συνάρτησης του προγράμματος, αμέσως μετά τις δηλώσεις μεταβλητών, θα πρέπει να προσθέσετε και τις γραμμές:

   ```
   #ifdef WIN32
   _setmode (_fileno (stdout), O_BINARY);
   _setmode (_fileno (stdin), O_BINARY);
   #endif
   ```

5. Η παράδοση της εργασίας αυτής συνίσταται στην υποβολή του πηγαίου αρχείου `wavproc.c` μέσω του e-class του μαθήματος (επιλογή "Εργασίες").