BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÁO CÁO THỰC TẬP CƠ SỞ**

**TÌM HIỂU CẤU TRÚC FILE ÂM THANH (WAV), THỂ HIỆN ĐỒ HỌA DẠNG SÓNG VÀ PHỔ VÀ CÁC THÔNG TIN HEADER**

**Giảng viên hướng dẫn: Nguyễn Thị Hương Lý**

**Sinh viên thực hiện: Mạnh Văn Hiệp**

**Mã số sinh viên: 61133631**

Khánh Hòa – 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÁO CÁO THỰC TẬP CƠ SỞ**

**TÌM HIỂU CẤU TRÚC FILE ÂM THANH (WAV), THỂ HIỆN ĐỒ HỌA DẠNG SÓNG VÀ PHỔ VÀ CÁC THÔNG TIN HEADER**

GVHD: Nguyễn Thị Hương Lý

SVTH: Mạnh Văn Hiệp

MSSV: 61133631

Khánh Hòa – Tháng 01/2022

# LỜI CẢM ƠN

Để hoàn thành bài báo cáo thực tập cơ sở này, trước hết em xin gửi đến quý thầy, cô Khoa Công nghệ Thông tin - Trường Đại học Nha Trang lời cảm ơn chân thành.

Em xin gửi đến cô Nguyễn Thị Hương Lý, người đã tận tình hướng dẫn, giúp đỡ em hoàn thành bài báo cáo này lời cảm ơn sâu sắc nhất.

Trong quá trình nghiên cứu thực hiện đề tài, cũng như là trong quá trình làm bài báo cáo thực tập cơ sở, khó tránh khỏi sai sót, rất mong các thầy, cô bỏ qua. Đồng thời do kiến thức cũng như kinh nghiệm thực tiễn của bản thân còn hạn chế nên đề tài, bài báo cáo cơ sở này khó thể không tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp từ quý thầy, cô để em học hỏi thêm được nhiều kinh nghiệm, cũng như kỹ năng cần thiết.

Em xin chân thành cảm ơn!

# MỤC LỤC

[LỜI CẢM ƠN i](#_Toc93392449)

[MỤC LỤC ii](#_Toc93392450)

[PHẦN MỞ ĐẦU 1](#_Toc93392451)

[1. Lý do chọn đề tài: 1](#_Toc93392452)

[2. Mục tiêu của đề tài. 1](#_Toc93392453)

[3. Phạm vi nghiên cứu. 1](#_Toc93392454)

[4. Phương pháp nghiên cứu. 1](#_Toc93392455)

[Chương 1. TỔNG QUAN VỀ VẤN ĐỀ NGHIÊN CỨU 2](#_Toc93392456)

[1.1 CƠ SỞ LÝ THUYẾT 2](#_Toc93392457)

[1.1.1 Tổng quan về file âm thanh (wav) 2](#_Toc93392458)

[1.1.1.1 Mô tả 3](#_Toc93392459)

[1.1.1.2 Chi tiết kỹ thuật 4](#_Toc93392460)

[1.1.1.3 Cách mở file WAV, WAVE 5](#_Toc93392461)

[1.1.1.4 WAV – Waveform Audio File Format 6](#_Toc93392462)

[1.1.2 Biến đổi Fourier 9](#_Toc93392519)

[1.1.2.1 Ứng dụng 9](#_Toc93392520)

[1.1.2.2 Biến đổi Fourier rời rạc 10](#_Toc93392521)

[1.2 CÔNG CỤ VÀ NGÔN NGỮ LẬP TRÌNH ĐƯỢC SỬ DỤNG 12](#_Toc93392522)

[1.2.1 Tổng quan về VISUAL STUDIO 2019 12](#_Toc93392523)

[1.2.2 Ngôn ngữ C++ 13](#_Toc93392524)

[1.2.3 Ngôn ngữ C# 13](#_Toc93392525)

[Chương 2. Đặc tả bài toán 15](#_Toc93392526)

[2.1 BÀI TOÁN 15](#_Toc93392527)

[2.2 Hướng xử lý và các thuật toán để thực hiện các yêu cầu: 15](#_Toc93392528)

[Mức 1: Đọc và thể hiện các thông tin header của file wav, hiển thị dạng sóng 15](#_Toc93392529)

[Đọc thông tin header file wav 15](#_Toc93392530)

[Hiển thị dạng sóng 20](#_Toc93392531)

[Mức 2: Phân tích phổ và hiển thị 24](#_Toc93392532)

[Chương 3. CÀI ĐẶT CHƯƠNG TRÌNH 26](#_Toc93392533)

[3.1 Code chương trình 26](#_Toc93392534)

[3.1.1 Mức 1 26](#_Toc93392535)

[3.1.1.1 Đọc thông tin header file wav 26](#_Toc93392536)

[3.1.1.2 Hiển thị dạng sóng 30](#_Toc93392537)

[3.1.2 Mức 2 32](#_Toc93392538)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 37](#_Toc93392539)

[4.1 KẾT QUẢ ĐẠT ĐƯỢC 37](#_Toc93392540)

[4.2 ƯU ĐIỂM 37](#_Toc93392541)

[4.3 HẠN CHẾ 37](#_Toc93392542)

[4.4 HƯỚNG PHÁT TRIỂN 37](#_Toc93392543)

[4.5 KẾT LUẬN 37](#_Toc93392544)

[TÀI LIỆU THAM KHẢO 38](#_Toc93392545)

# PHẦN MỞ ĐẦU

1. **Lý do chọn đề tài:**

**Công nghệ thông tin** là một trong những ngành phát triển vượt bậc trong những năm gần đây. Ngày nay với sự phát triển nhanh chóng của xã hội thì công nghệ thông tin được ứng dụng rộng rãi ở hầu hết tất cả các lĩnh vực và ngày càng đóng vai trò quan trọng, trở thành một phần thiết yếu trong đời sống hằng ngày. Công nghệ thông tin là một ngành đòi hỏi ở người học một nền tảng kiến thức vững chắc, sự tư duy logic cao, hiểu biết sâu rộng trên nhiều lĩnh vực. Với chúng em hiện đang là những sinh viên công nghệ thông tin cần phải có sự đầu tư, không ngừng học hỏi để nâng cao kiến thức. Do đó để củng cố lại kiến thức đã học, đề tài mà em chọn để thực hiện là: **Tìm hiểu cấu trúc file âm thanh (wav), thể hiện đồ họa dạng sóng và phổ và các thông tin header.**

1. **Mục tiêu của đề tài.**
   * Củng cố lại kiến thức đã học về các cấu trúc điều khiển, ôn tập lại kiến thức về phần mảng, chuỗi cũng như kiểu dữ liệu cấu trúc...
   * Rèn luyện kỹ năng lập trình trên ngôn ngữ C++, C#.
   * Ứng dụng lý thuyết đã học giải quyết bài toán ứng dụng cụ thể.
2. **Phạm vi nghiên cứu.**
   * Đọc thông tin header file wav, hiện thị dạng song.
   * Lý thuyết về biến đổi Fourier, sử dụng Fourier để phân tích phổ
3. **Phương pháp nghiên cứu.**
   * Tìm kiếm và nghiên cứu trên mạng Internet.

# TỔNG QUAN VỀ VẤN ĐỀ NGHIÊN CỨU

## CƠ SỞ LÝ THUYẾT

### Tổng quan về file âm thanh (wav)

WAV, được biết đến với WAVE (Waveform Audio File Format), là một tập hợp con của Microsoft’s Resource Interchange File Format (RIFF) để lưu trữ các tệp âm thanh kỹ thuật số. Định dạng này không áp dụng bất kỳ quá trình nén nào cho dòng bit và lưu trữ các bản ghi âm với tốc độ lấy mẫu và tốc độ bit khác nhau. Nó đã và đang là một trong những định dạng tiêu chuẩn cho đĩa CD âm thanh. Các tệp sóng có kích thước lớn hơn so với các định dạng tệp âm thanh mới như MP3 , sử dụng tính năng nén mất dữ liệu để giảm kích thước tệp trong khi vẫn giữ nguyên chất lượng âm thanh. Tuy nhiên, các tệp WAV có thể được nén bằng  Audio Compression Manager (ACM) codecs . Có một số API và ứng dụng có sẵn có thể chuyển đổi tệp WAV sang các định dạng tệp âm thanh phổ biến khác.

File đuôi có dạng .WAV hoặc .WAVE là các file Waveform Audio được phát triển bởi sự hợp tác của Microsoft và IBM vào những năm 1991.

Mặc dù các file WAV thường không được nén nhưng chúng ta vẫn có thể nén chúng nếu muốn, phù hợp để sử dụng trên các hệ thống Windows, đối với MacOS vẫn có thể sử dụngnhưng không phổ biến bằng.

WAV thường được dùng cho công việc chỉnh sửa âm nhạc, trò chơi điện tử, không được dùng trong chia sẻ vì kích thước tệp khá lớn. Về bản chất, WAV khá giống với các tệp AIFF và 8SVX, cả hai đều được sử dụng phổ biến.



Hình 1: WAV là định dạng file chứa âm thanh của Windows

#### ****Mô tả****

WAV chứa các tác phẩm âm nhạc, bản ghi âm giọng nói và các hiệu ứng âm thanh. Có thể phát được trong QuickTime, Windows Media Player và một số phần mềm hỗ trợ khác.

WAV có kích thước lớn hơn MP3, bạn sẽ rất tốn nhiều thời gian cho việc chia sẻ file qua Internet.

Một trong những ưu điểm chính của WAV được liên kết với việc sử dụng Điều chế mã xung tuyến tính (LPCM) để lưu trữ luồng âm thanh.



Hình 2: File WAV thường dùng trong phần mềm chỉnh sửa âm thanh

#### ****Chi tiết kỹ thuật****

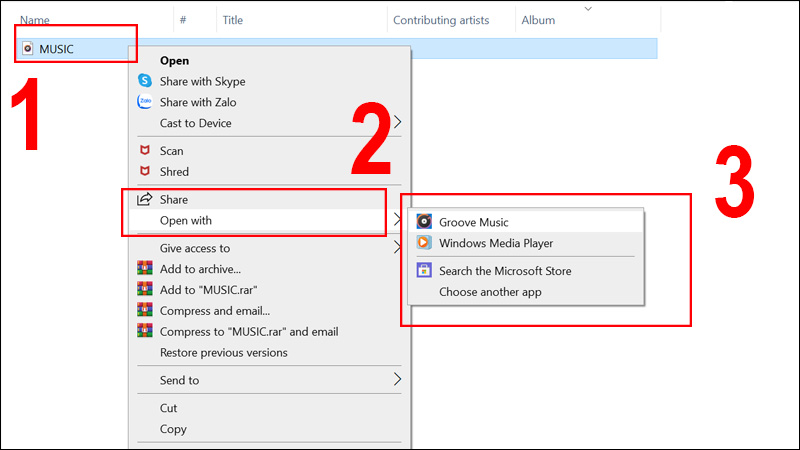
* WAV được ghi thành 2 loại 8 bit và 16 bit cho mỗi mẫu.
* Một tùy chọn tiêu chuẩn cho CD Audio là một dòng âm thanh 16 bit cho mỗi mẫu và tần số lấy mẫu là 44,1 KHz.
* Một giây của âm thanh tương ứng với 88 Kb bộ nhớ trong.
* Các tệp WAV có thể lưu trữ siêu dữ liệu trong đoạn INFO và chúng cũng bao gồm các danh sách IFF được tích hợp.
* Trong một số trường hợp, định dạng chuẩn có thể được sử dụng để phát sóng.



Hình 3: WAV được phát triển bởi Microsoft và IBM

#### ****Cách mở file WAV, WAVE****

Nhấn chuột phải vào file cần mở > **Open with**>Chọn phần mềm bạn muốn dùng là hoàn tất.



Hình 4: Chọn file muốn mở với Open with

#### WAV – Waveform Audio File Format

**Kích thước bit là gì?**  
 Kích thước bit xác định lượng thông tin có thể được lưu trữ trong một tệp. Đối với hầu hết các mục đích ngày nay, kích thước bit phải là 16 bit. Các tệp 8 bit nhỏ hơn (1/2 kích thước), nhưng có độ phân giải thấp hơn.

Kích thước bit liên quan đến biên độ. Trong các bản ghi 8 bit, có sẵn tổng cộng 256 (0 đến 255) mức biên độ. Trong 16 bit, có sẵn tổng cộng 65,536 (-32768 đến 32767) mức biên độ. Độ phân giải của tệp càng lớn thì phạm vi động thực tế của tệp càng lớn. CD-Audio sử dụng các mẫu 16 bit.

**Sample Rate là gì?**  
 Sample rate là số lượng mẫu trong một giây. CD-Audio có tỷ lệ mẫu là 44.100. Điều này có nghĩa là 1 giây âm thanh có 44.100 mẫu. Băng DAT có tỷ lệ mẫu là 48.000.

Khi nhìn vào đáp ứng tần số, tần số cao nhất có thể được coi là 1/2 sample rate.

**Channels là gì?**  
 Channels là số phần tử ghi riêng biệt trong dữ liệu. Đối với một ví dụ nhanh thực tế, channel là đơn âm và hai channels là âm thanh nổi. Trong tài liệu này, cả bản ghi kênh đơn và kênh đôi sẽ được thảo luận.

**Dữ liệu là gì?**  
 Dữ liệu là các mẫu riêng lẻ. Một mẫu riêng lẻ là kích thước bit nhân với số kênh. Ví dụ, một kênh đơn (kênh đơn), ghi tám bit có kích thước mẫu riêng là 8 bit. Bản ghi mười sáu bit đơn nguyên có kích thước mẫu riêng là 16 bit. Bản ghi âm thanh nổi mười sáu bit có kích thước mẫu riêng lẻ là 32 bit.

Các mẫu được đặt từ đầu đến cuối để tạo thành dữ liệu. Vì vậy, ví dụ: nếu bạn có bốn mẫu (s1, s2, s3, s4) thì dữ liệu sẽ giống như sau: s1s2s3s4.

**Tiêu đề là gì?**  
 Tiêu đề là phần đầu của tệp WAV (RIFF). Tiêu đề được sử dụng để cung cấp thông số kỹ thuật về loại tệp, tỷ lệ mẫu, kích thước mẫu và kích thước bit của tệp, cũng như độ dài tổng thể của nó.

* Tiêu đề của tệp WAV (RIFF) dài 44 byte và có định dạng sau:

|  |  |  |
| --- | --- | --- |
| Vị trí | Giá trị mẫu | Sự miêu tả |
| 1 - 4 | "RIFF" | Đánh dấu tệp là tệp riff. Mỗi ký tự dài 1 byte. |
| 5 - 8 | File size (integer) | Kích thước của tệp tổng thể - 8 byte, tính bằng byte (số nguyên 32 bit). Thông thường, bạn sẽ điền vào thông tin này sau khi tạo. |
| 9 - 12 | “WAVE” | Tiêu đề Loại Tệp. Đối với mục đích của chúng tôi, nó luôn luôn bằng "WAVE". |
| 13 - 16 | “fmt" | Định dạng điểm đánh dấu chunk. Bao gồm null ở cuối. |
| 17 - 20 | 16 | Độ dài của dữ liệu định dạng như được liệt kê ở trên. |
| 21 - 22 | 1 | Loại định dạng (1 is PCM) - 2 byte integer. |
| 23 - 24 | 2 | Số kênh - 2 byte integer. |
| 25 – 28 | 44100 | Tỷ lệ mẫu - 32 byte integer. Common values là 44100 (CD), 48000 (DAT). Tốc độ lấy mẫu = Số lượng mẫu mỗi giây hoặc Hertz. |
| 29 - 32 | 176400 | (Sample Rate \* BitsPerSample \* Channels) / 8. |
| 33 – 34 | 4 | (BitsPerSample \* Channels) / 8.1 - 8 bit mono2 - 8 bit stereo/16 bit mono4 - 16 bit stereo |
| 35 – 36 | 16 | Số bit trên mỗi mẫu |
| 37 – 40 | “data” | Tiêu đề phân đoạn "data". Đánh dấu phần bắt đầu của phần dữ liệu. |
| 41 – 44 | File size (data) | Kích thước của phần dữ liệu. |
| Các giá trị mẫu được đưa ra ở trên đối với nguồn âm thanh nổi 16 bit. | | |

Có thể tìm thấy ví dụ về nội dung của tiêu đề cho tệp wave

|  |  |  |  |
| --- | --- | --- | --- |
| ****Name**** | ****offset**** | ****Size**** | ****Value**** |
| ChunkID | 0 | 4 | “RIFF” |
| ChunkSize | 4 | 4 | 405040 |
| Format | 8 | 4 | “WAVE” |
| Subchunk1 ID | 12 | 4 | “fmt” |
| Subchunk1 Size | 16 | 4 | 16 |
| Audio Format | 20 | 2 | 1 |
| Num Channels | 22 | 2 | 2 |
| Sample Rate | 24 | 4 | 22050 |
| Byte Rate | 28 | 4 | 88200 |
| Block Align | 32 | 2 | 4 |
| Bits per Sample | 34 | 2 | 16 |
| Subchunk2 ID | 36 | 4 | “data” |
| Subchunk 2 Size | 40 | 4 | 405004 |

### Biến đổi Fourier

**Biến đổi Fourier** hay **chuyển hóa Fourier**, được đặt tên theo nhà toán học người Pháp Joseph Fourier, là phép biến đổi một hàm số hoặc một tín hiệu theo miền thời gian sang miền tần số. Chẳng hạn như một bản nhạc có thể được phân tích dựa trên tần số của nó.

#### Ứng dụng

Biến đổi Fourier có rất nhiều ứng dụng khoa học, ví dụ như trong vật lý, số học, xử lý tín hiệu, xác suất, thống kê, mật mã, âm học, hải dương học, quang học, hình học và rất nhiều lĩnh vực khác. Trong xử lý tín hiệu và các ngành liên quan, biến đổi Fourier thường được nghĩ đến như sự chuyển đổi tín hiệu thành các thành phần biên độ và tần số. Sự ứng dụng rộng rãi của biến đổi Fourier bắt nguồn từ những tính chất hữu dụng của biến đổi này:

* Tính tuyến tính: {\displaystyle {\mathfrak {F}}[a.f+b.g]=a.{\mathfrak {F}}[f]+b.{\mathfrak {F}}[g]} 
* Tồn tại biến đổi nghịch đảo, và thực tế là biến đổi Fourier nghịch đảo gần như có cùng dạng với biến đổi thuận.
* Những hàm số sin cơ sở là các hàm riêng của phép vi phân, có nghĩa là khai triển này biến những phương trình vi phân tuyến tính với các hệ số không đổi thành các phương trình đại số cơ bản. Ví dụ, trong một hệ vật lý tuyến tính không phụ thuộc thời gian, tần số là một đại lượng không đổi, do đó những thành phần tần số khác nhau có thể được tính toán một cách độc lập.
* Theo định lý tích tổng chập, biến đổi Fourier chuyển một tích tổng chập phức tạp thành một tích đại số đơn giản.
* Biến đổi Fourier rời rạc có thể được tính toán một cách nhanh chóng bằng máy tính nhờ thuật toán FFT (*fast Fourier transform*).
* Theo định lý Parseval-Plancherel, năng lượng của tín hiệu (tích phân của bình phương giá trị tuyệt đối của hàm) không đổi sau biến đổi Fourier.

#### Biến đổi Fourier rời rạc

Trong toán học, phép **biến đổi Fourier rời rạc (DFT)**, đôi khi còn được gọi là biến đổi Fourier hữu hạn biến đổi Fourier hữu hạn, là một biến đổi trong giải thích Fourier cho các tín hiệu thời gian rời rạc. Đầu vào của biến đổi này là một chuỗi hữu hạn các số thực hoặc số phức, làm biến đổi này là một công cụ lý tưởng để xử lý thông tin trên các máy tính. Đặc biệt, biến đổi này được sử dụng rộng rãi trong xử lý tín hiệu và các ngành liên quan đến phân tích tần số chứa trong một tín hiệu, để giải phương trình đạo hàm riêng, và để làm các phép như tích chập. Biến đổi này có thể được tính nhanh bởi thuật toán biến đổi Fourier nhanh (FFT).

##### Định nghĩa

Dãy của *N* số phức:{\displaystyle x\_{0},...,x\_{N-1}} x0, …, xN-1 được biến đổi thành chuỗi của *N* số phức *X*0, ..., *XN*−1 bởi công thức sau đây:

Text, letter

Description automatically generated

với *e* là cơ số của logarit tự nhiên, {\displaystyle i\,}i là đơn vị ảo ({\displaystyle i^{2}=-1}i2 = -1), và π là pi. Phép biến đổi đôi khi được ký hiệu bởi {\displaystyle {\mathcal {F}}}

Phép **biến đổi Fourier rời rạc ngược (IDFT)** được cho bởi công thức sau

Text, letter

Description automatically generated

Những phương trình này có thể được mô tả đơn giản như sau: các số phức *Xk* đại diện cho biên độ và pha ở các bước sóng khác nhau của "tín hiệu vào" *xn*. Phép biến đổi DFT tính các giá trị *Xk* từ các giá trị *xn*, trong khi IDFT tính *xn* bằng tổng của các sóng thành phần {\displaystyle {\frac {1}{N}}X\_{k}e^{{\frac {2\pi i}{N}}kn}} Text

Description automatically generated với tần số *k / N*. Khi viết các phương trình dưới dạng như trên, ta đã sử dụng công thức Euler để biểu diễn các hàm lượng giác dưới dạng lũy thừa số phức để biến đổi được dễ dàng. Khi viết *Xk* dưới dạng tọa độ cực, ta thu được biên độ *Ak / N* và pha *φk* từ modulus và argument của *Xk*:

Text, letter

Description automatically generated

trong đó atan2 là dạng hai đối số của hàm arctan. Cần ghi chú rằng các thừa số chuẩn hóa của DFT và IDFT (ở đây là 1 và 1/*N*) và dấu của các số mũ chỉ là quy ước, và có thể khác nhau trong các tài liệu khác nhau. Điều kiện duy nhất cho các quy ước này là DFT và IDFT có dấu ngược nhau ở các số mũ và tích của hai thừa số chuẩn hóa phải là 1/*N*.

##### Ứng dụng

DFT có nhiều ứng dụng rộng rãi trong nhiều ngành khác nhau. Tất cả các ứng dụng của DFT đều dựa trên một tính chất quan trọng là DFT và IDFT đều có thể được tính nhanh chóng bằng thuật toán biến đổi Fourier nhanh.

**Phân tích phổ**

Khi sử dụng DFT để phân tích phổ, dãy {x\_n} thường đại diện cho một dãy hữu hạn các mẫu tại các thời điểm cách đều nhau của một tín hiệu x(t), trong đó t để chỉ thời gian. Việc chuyển từ thời gian liên tục sang mẫu (thời gian rời rạc) chuyển biến đổi Fourier liên tục của x(t) thành biến đổi Fourier thời gian rời rạc (DTFT), và thường gây ra hiệu ứng răng cưa. Việc chọn lựa tần số lấy mẫu thích hợp (xem tần số Nyquist) là vô cùng quan trọng cho việc giảm thiểu hiệu ứng này.

## CÔNG CỤ VÀ NGÔN NGỮ LẬP TRÌNH ĐƯỢC SỬ DỤNG

### Tổng quan về VISUAL STUDIO 2019

* **Microsoft Visual Studio** là một môi trường phát triển tích hợp (IDE) từ Microsoft. Microsoft Visual Studio còn được gọi là "Trình soạn thảo mã nhiều người sử dụng nhất thế giới ", được dùng để lập trình C++ và C# là chính. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.
* **Ưu điểm:**
* Đa nền tảng.
* Đa ngôn ngữ lập trình
* Hỗ trợ website.
* Kho tiện ích mở rộng phong phú.
* Lưu trữ phân cấp.
* Kho lưu trữ an toàn.
* **Hạn chế:**
* Tuy có giao diện đẹp, nhưng khá khó sử dụng cho người mới học lập trình. Vì khá nhiều bước thiết lập và cài đặt.
* Vì Visual Studio là một IDE nặng nên cần sử dụng nhiều tài nguyên để khởi động và vận hành.

### Ngôn ngữ C++

C++ là một loại ngôn ngữ lập trình bậc trung (middle-level). Đây là ngôn ngữ lập trình đa năng được tạo ra bởi Bjarne Stroustrup như một phần mở rộng của ngôn ngữ lập trình C, hoặc "C với các lớp Class", Ngôn ngữ đã được mở rộng đáng kể theo thời gian và C ++ hiện đại có các tính năng: lập trình tổng quát, lập trình hướng đối tượng, lập trình thủ tục, ngôn ngữ đa mẫu hình tự do có kiểu tĩnh, dữ liệu trừu tượng, và lập trình đa hình, ngoài ra còn có thêm các tính năng, công cụ để thao tác với bộ nhớ cấp thấp. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ thương mại ưa thích và phổ biến của lập trình viên.

* **Ưu điểm:**
* Là ngôn ngữ lập trình hướng đối tượng.
* Cho phép thoải mái quản lý vùng nhớ
* Là một ngôn ngữ low-level dễ dàng giao tiếp với phần cứng.
* **Hạn chế:**
* Tràn vùng nhớ.
* OOP trong C++ khá phức tạp và khó hiểu.
* Con trỏ là một cái gì đó rất … khó.

### Ngôn ngữ C#

C# (hay C sharp) là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000. C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.

**C#** (**C Sharp**, đọc là *"xi-sáp"*) là một ngôn ngữ lập trình hướng đối tượng đa năng, mạnh mẽ được phát triển bởi Microsoft, C# là phần khởi đầu cho kế hoạch .NET của họ. Tên của ngôn ngữ bao gồm ký tự thăng theo Microsoft nhưng theo ECMA là C#, chỉ bao gồm dấu số thường. Microsoft phát triển C# dựa trên C++ và Java. C# được miêu tả là ngôn ngữ có được sự cân bằng giữa C++, Visual Basic, Delphi và Java.

**Ưu điểm:**

* C Sharp là ngôn ngữ đơn giản.
* Ngôn ngữ đa nền tảng, an toàn và hiệu quả.
* Là ngôn ngữ hiện đại thông dụng, ít từ khóa dễ hiểu.
* Là ngôn ngữ lập trình thuần hướng đối tượng
* **Hạn chế:**
* Nhược điểm lớn nhất của C# là chỉ chạy trên nền Windows và có cài .NET Framework. Thao tác đối với phần cứng yếu hơn so với ngôn ngữ khác, hầu hết phải dựa vào windows.

# Đặc tả bài toán

## BÀI TOÁN

Vận dụng kiến thức lập trình C, C#. Biết làm việc với file nhị phân. Tìm hiểu biến đổi Fourier để phân tích phổ.

* 1. **Mức 1:** Đọc và thể hiện các thông tin header của file wav, hiển thị dạng sóng.
  2. **Mức 2:** Phân tích phổ và hiển thị.

## Hướng xử lý và các thuật toán để thực hiện các yêu cầu:

### Mức 1: Đọc và thể hiện các thông tin header của file wav, hiển thị dạng sóng

#### Đọc thông tin header file wav

##### **Hướng xử lý**

* Tìm hiểu tệp WAV là gì.
* Định dạng tệp WAV (\*.wav).
* Tìm hiểu thông tin header file wav có những gì.

##### **Chi tiết chương trình**

Khai báo thư viện C++ và khai báo hằng số \_CRT\_SECURE\_NO\_DEPRECATE

* Các hàm đọc dữ liệu như scanf, fscanf, gets, ...v.v là **không an toàn** bởi vì nó có thể gây **tràn bộ nhớ đệm** khi đọc dữ liệu nhập từ người dùng nên ở đây ta khai báo hằng số \_CRT\_SECURE\_NO\_DEPRECATE để bỏ qua lỗi này.
* Thư viện **cstdint** đảm bảo rằng biến (variable) kiểu số nguyên mà bạn chọn chiếm cùng kích thước vùng nhớ trên mọi kiến trúc hệ thống.

|  |
| --- |
| *#define \_CRT\_SECURE\_NO\_DEPRECATE*  *#include <iostream>*  *#include <string>*  *#include <fstream>*  *#include <cstdint>*  *using namespace std;* |

Ta dung kiểu struct để lưu trữ các biến thông tin file header wav. Ở đây ta dung typedef struct để định nghĩa lại kiểu struct bằng tên wav\_hdr.

|  |
| --- |
| *typedef struct WAV\_HEADER*  *{*  */\* RIFF Chunk Descriptor \*/*  *uint8\_t RIFF[4]; // RIFF Header Magic header*  *uint32\_t ChunkSize; // RIFF Chunk Size*  *uint8\_t WAVE[4]; // WAVE Header*  */\* "fmt" sub-chunk \*/*  *uint8\_t fmt[4]; // FMT header*  *uint32\_t Subchunk1Size; // Size of the fmt chunk*  *uint16\_t AudioFormat; // Audio format 1=PCM,6=mulaw,7=alaw, 257=IBM Mu-Law, 258=IBM A-Law, 259=ADPCM*  *uint16\_t NumOfChan; // Number of channels 1=Mono 2=Sterio*  *uint32\_t SamplesPerSec; // Sampling Frequency in Hz*  *uint32\_t bytesPerSec; // bytes per second*  *uint16\_t blockAlign; // 2=16-bit mono, 4=16-bit stereo*  *uint16\_t bitsPerSample; // Number of bits per sample*  */\* "data" sub-chunk \*/*  *uint8\_t Subchunk2ID[4]; // "data" string*  *uint32\_t Subchunk2Size; // Sampled data length*  *} wav\_hdr;* |

Sử dụng hàm getFileSize để lấy kích thược tệp.

* Hàm **int fseek**(FILE \*\_Stream, long \_Offset, int \_Origin) thiết lập vị trí file của Stream tới offset đã cho. Tham số **offset** xác định số byte để tìm kiếm từ vị trí **where** đã cho.
* **stream** − Đây là con trỏ tới một đối tượng FILE mà nhận diện Stream.
* **offset** − Đây là số byte để offset từ đó.
* **whence** − Đây là vị trí từ đó offset được thêm vào. Nó được xác định bởi một trong số các hằng sau: (ở bài báo cáo này sử dụng hằng SEEK\_SET và SEEK\_END)
* SEEK\_SET miêu tả phần đầu file.
* SEEK\_CUR miêu tả vị trí hiện tại của con trỏ file.
* SEEK\_END miêu tả phần cuối file.
* Hàm **ftell**(FILE \*\_Stream) trả về vị trí file hiện tại của steam được chỉ định. Chúng ta có thể sử dụng hàm ftell() để lấy tổng kích thước của một file sau khi di chuyển con trỏ tệp ở cuối tệp. Chúng ta có thể sử dụng hằng số SEEK\_END để di chuyển con trỏ file đến cuối tập tin.

|  |
| --- |
| *int getFileSize(FILE\* inFile);*  *int main(int argc, char\* argv[]) { … }*  *int getFileSize(FILE\* inFile)*  *{*  *int fileSize = 0;*  *fseek(inFile, 0, SEEK\_END); // đặt con trỏ ghi tại vị trí thứ 0 bắt đầu từ phần cuối file*  *fileSize = ftell(inFile); // lấy tổng kích thước của một file sau khi di chuyển con trỏ file ở cuối file*  *fseek(inFile, 0, SEEK\_SET); // đặt lại con trỏ ghi tại vị trí thứ 0 bắt đầu từ phần đầu file*  *return fileSize;*  *}* |

Xây dựng hàm main

* Hàm main có thể có hai tham số, argc và argv. argc là một tham số nguyên (int) và đó là số lượng đối số được truyền cho chương trình.
* Tên chương trình luôn là đối số đầu tiên, do đó, sẽ có ít nhất một đối số cho một chương trình và giá trị tối thiểu argc sẽ là một đối số. Nhưng nếu một chương trình có hai đối số thì giá trị của argc sẽ là ba.
* Tham số argv trỏ đến một mảng chuỗi và được gọi là vectơ đối số . Nó là một chuỗi một chiều của các đối số hàm.
* Toán tử **sizeof** cung cấp thông tin về bao nhiêu byte bộ nhớ mà đối số của nó chiếm (hoặc có thể chiếm).
* **cin.get** (string\_name, size) được sử dụng để truy cập mảng ký tự. Nó bao gồm các ký tự khoảng trắng. Nói chung, cin với toán tử trích xuất (>>) kết thúc khi tìm thấy khoảng trắng. Tuy nhiên, cin.get () đọc một chuỗi có khoảng trắng.
* **c\_str** ()trả về một con trỏ đến một mảng có chứa một chuỗi ký tự được kết thúc bằng rỗng (tức là chuỗi C) đại diện cho giá trị hiện tại của đối tượng chuỗi.
* int **fprintf** (FILE \* stream, const char \* format, ...); Hàm fprintf() ghi chuỗi được trỏ đến theo định dạng vào luồng dòng. Định dạng chuỗi có thể chứa các chỉ định dạng bắt đầu bằng % được thay thế bằng các giá trị của các biến được chuyển đến fprintf() dưới dạng các đối số bổ sung.
* **stderr** có thể được sử dụng làm đối số cho bất kỳ hàm nào nhận đối số kiểu FILE \* mong đợi một luồng đầu ra, như fputs hoặc fprintf.
* **size\_t** thực chất cũng là một kiểu dữ liệu không dấu mà giá trị của nó phụ thuộc vào từng chương trình cụ thể chứ không phải được cố định sẵn như các kiểu dữ liệu không dấu khác. size\_t rất linh hoạt, có thể tự mở rộng miền giá trị ra để đáp ứng con số lớn hơn.
* **fread** (void \* ptr, size\_t size, size\_t count, FILE \* stream); Tổng số byte được đọc nếu thành công.
* **uint8\_t**, **uint16\_t**, **uint32\_t** giống với **size\_t**. **int8\_t** kiểu dữ liệu có dấu.
* **getFileSize** (FILE \*inFile) Truy xuất kích thước của tệp được chỉ định, tính bằng byte.

|  |
| --- |
| *int main(int argc, char\* argv[])*  *{*  *wav\_hdr wavHeader;*  *int headerSize = sizeof(wav\_hdr), filelength = 0;*  *const char\* filePath;*  *string input;*  *if (argc <= 1)*  *{*  *cout << "Input wave file name: ";*  *cin >> input;*  *cin.get();*  *filePath = input.c\_str();*  *}*  *else*  *{*  *filePath = argv[1];*  *cout << "Input wave file name: " << filePath << endl;*  *}*  *FILE\* wavFile = fopen(filePath, "r"); // mở file wav ở chế độ đọc (r)*  *if (wavFile == nullptr)*  *{*  *fprintf(stderr, "Unable to open wave file: %s\n", filePath);*  *return 1;*  *}*  *//Read the header*  *size\_t bytesRead = fread(&wavHeader, 1, headerSize, wavFile);*  *cout << "Header Read " << bytesRead << " bytes." << endl; // độ dài header file wav (44 bytes)*  *if (bytesRead > 0)*  *{*  *//Read the data*  *uint16\_t bytesPerSample = wavHeader.bitsPerSample / 8; //Number of bytes per sample*  *if (bytesPerSample != 0)uint64\_t numSamples = wavHeader.ChunkSize / bytesPerSample; //How many samples are in the wav file?*  *static const uint16\_t BUFFER\_SIZE = 4096;*  *int8\_t\* buffer = new int8\_t[BUFFER\_SIZE];*  *while ((bytesRead = fread(buffer, sizeof buffer[0], BUFFER\_SIZE / (sizeof buffer[0]), wavFile)) > 0)*  *{*  */\*\* DO SOMETHING WITH THE WAVE DATA HERE \*\*/*  *cout << "Read " << bytesRead << " bytes." << endl;*  *}*  *delete[] buffer;*  *buffer = nullptr;*  *filelength = getFileSize(wavFile); // tính kích thước tệp*  *cout << "File is :" << filelength << " bytes." << endl;*  *cout << "RIFF header :" << wavHeader.RIFF[0] << wavHeader.RIFF[1] << wavHeader.RIFF[2] << wavHeader.RIFF[3] << endl;*  *cout << "WAVE header :" << wavHeader.WAVE[0] << wavHeader.WAVE[1] << wavHeader.WAVE[2] << wavHeader.WAVE[3] << endl;*  *cout << "FMT :" << wavHeader.fmt[0] << wavHeader.fmt[1] << wavHeader.fmt[2] << wavHeader.fmt[3] << endl;*  *cout << "Data size :" << wavHeader.ChunkSize << endl;*  *// Hiển thị Tỷ lệ lấy mẫu từ header*  *cout << "Sampling Rate :" << wavHeader.SamplesPerSec << endl; // Mẫu mỗi giây*  *cout << "Number of bits used :" << wavHeader.bitsPerSample << endl; // Số lượng bits được sử dụng*  *cout << "Number of channels :" << wavHeader.NumOfChan << endl; // Số kênh*  *cout << "Number of bytes per second :" << wavHeader.bytesPerSec << endl; // Số byte mỗi giây*  *cout << "Data length :" << wavHeader.Subchunk2Size << endl; // Độ dài dữ liệu*  *cout << "Audio Format :" << wavHeader.AudioFormat << endl; // Định dạng âm thanh*  *// Audio format 1=PCM,6=mulaw,7=alaw, 257=IBM Mu-Law, 258=IBM A-Law, 259=ADPCM*  *cout << "Block align :" << wavHeader.blockAlign << endl; // Căn chỉnh khối*  *cout << "Data string :" << wavHeader.Subchunk2ID[0] << wavHeader.Subchunk2ID[1] << wavHeader.Subchunk2ID[2] << wavHeader.Subchunk2ID[3] << endl;*  *// Chuỗi dữ liệu*  *}*  *fclose(wavFile); // đóng file*  *return 0;*  *}* |

#### Hiển thị dạng sóng

##### **Hướng xử lý**

* Thể hiện giao diện bằng winform.
* Sử dụng Chart và thư viện NAudio để thể hiện sóng âm thanh.

##### **Chi tiết chương trình**

Xử lý giao diện bằng winform

Thêm thư viện NAudio về máy

* Vào Solution Explorer chuột phải vào tên file chọn Manage NuGet Packages.

A picture containing text, screenshot, computer, computer

Description automatically generated

* Bấm Browse nhập NAudio và bấm install.

Graphical user interface, text, application, email

Description automatically generated

* Graphical user interface, application

  Description automatically generatedKéo thả MenuStrip trong Toolbox vào form chính.

Graphical user interface, application

Description automatically generated

* Hiển thị WAV bằng WaveViewer trong trong Toolbox bằng cách kéo thả vào form chính. Sau đó vào Properties kéo tới Dock trong Layout chọn Fill.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* Nhập thông tin vào MenuStrip.

Graphical user interface, application, table

Description automatically generated

* Hiển thị dạng biểu đồ bằng cách thêm Chart trong Toolbox vào form chính. Sau đó vào Properties kéo tới Dock trong Layout chọn Fill.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* Trong Properties kéo đến phần Chart bấm vào dấu 3 chấm bên cạnh (Collection) nhấn remove rồi ok. Tương tự với Series.

Graphical user interface, application

Description automatically generated

Xử lý code

Nhấp đúp chuột vào Open Wave trong form

Tạo code thực hiện việc mở file với phần đuôi mở rộng là .wav

* **OpenFileDialog** cho phép người dùng duyệt các thư mục trên máy tính hoặc thư mục trên bất kỳ máy tính nào trên mạng LAN.

|  |
| --- |
| *OpenFileDialog open = new OpenFileDialog();*  *open.Filter = "Wave File (\*.wav)|\*.wav;";*  *if (open.ShowDialog() != DialogResult.OK) return;* |

Tùy chỉnh lại cấu trúc của Chart1 để hiển thị biểu đồ song WAV

|  |
| --- |
| *chart1.Series.Add("wave");*  *chart1.Series["wave"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.FastLine;*  *chart1.Series["wave"].ChartArea = "ChartArea1";* |

Thực hiện trả về dấu phẩy động từ việc chuyển đổi các byte

* **open.Filter** Lấy hoặc đặt chuỗi bộ lọc tên tệp hiện tại, chuỗi này xác định các lựa chọn xuất hiện trong hộp "Lưu dưới dạng tệp" hoặc "Loại tệp" trong hộp thoại. Ở đây ta lấy định dạng là .wav.
* Phương thức **BitConverter.ToSingle** () trong C # được sử dụng để trả về một số dấu phẩy động có độ chính xác đơn được chuyển đổi từ bốn byte tại một vị trí xác định trong một mảng byte.

|  |
| --- |
| *byte[] buffer = new byte[16384];*  *int read = 0;*  *while (wave.Position < wave.Length)*  *{*  *read = wave.Read(buffer, 0, 16384);*  *for (int i = 0; i < read / 4; i++)*  *{*  *chart1.Series["wave"].Points.Add(BitConverter.ToSingle(buffer, i \* 4));*  *}*  *}* |

### Mức 2: Phân tích phổ và hiển thị

##### **Hướng xử lý**

* Tìm hiểu về FFT.
* Thể hiện phổ âm thanh lên màn hình bằng winform

##### **Chi tiết chương trình**

Xử lý giao diện bằng winform

* Thêm thư viện NAudio và Accord về máy (làm tương tự với thêm NAudio trong phần chi tiết chương trình hiển thị dạng sóng)

Giao diện chính

Table

Description automatically generated

* Ở đây ta thấy có 2 biểu đồ là của Microphone PCM Data và Microphone FFT Data
* Microphone PCM Data. PCM là một đại diện kỹ thuật số của một tín hiệu analog mà sẽ đưa các mẫu biên độ của tín hiệu analog đều đặn. Các dữ liệu tương tự được lấy mẫu được thay đổi thành, và sau đó được đại diện bởi, dữ liệu nhị phân.
* Trục hoành biểu thị cho thời gian với đơn vị đo là ms.
* Trục tung biểu thị cho biên độ.
* LPCM là từ viết tắt của Linear pulse-code modulation (tạm dịch: Điều mã xung tuyến tính), một phương pháp mã hóa tín hiệu âm thanh kỹ thuật số.
* Microphone FFT Data. Biến đổi Fourier nhanh (FFT) là một thuật toán rất hiệu quả để tính toán Biến đổi Fourier rời rạc (DFT) và Biến đổi ngược. Có nhiều loại thuật toán FFT khác nhau sử dụng các kiến thức từ nhiều mảng khác nhau của toán học, từ số phức tới lý thuyết nhóm và lý thuyết số.
* Trục hoành biểu thị cho tần số với đơn vị đo là Hz.
* Trục hoành biểu thị cho độ lớn âm thanh.
* RAW là từ dùng để chỉ định dạng âm thanh được lưu trữ thô và không nén trên bộ nhớ.
* Ở mục View có phần Auto-Scale. Tự động thay đổi tỷ lệ, còn được hiểu là tự động mở rộng quy mô hoặc tự động điều chỉnh tỷ lệ, và đôi khi còn được gọi là tự động mở rộng tỷ lệ, là một phương pháp được sử dụng trong điện toán đám mây để điều chỉnh động lượng tài nguyên tính toán trong một trang trại máy chủ - thường được đo bằng số lượng máy chủ đang hoạt động - tự động dựa trên tải trọng của trang trại.
* Ở mục Help ta có Info Message. Chức năng này cho biết các phím bấm chuột tác động lên biểu đồ như thế nào.
* Bấm giữ chuột trái lên biểu đồ để di chuyển biểu đồ.
* Bấm giữ chuột phải lên biểu đồ để phóng to thu nhỏ biểu đồ.
* Bấm chuột giữa lên biểu đồ để trục tự động biểu đồ.
* Bấm đúp chuột trên biểu đồ để thống kê đồ thị biểu đồ.

# CÀI ĐẶT CHƯƠNG TRÌNH

## Code chương trình

### Mức 1

#### Đọc thông tin header file wav

##### **Code**

|  |
| --- |
| *#define \_CRT\_SECURE\_NO\_DEPRECATE*  *#include <iostream>*  *#include <string>*  *#include <fstream>*  *#include <cstdint>*  *using namespace std;*  *typedef struct WAV\_HEADER*  *{*  */\* RIFF Chunk Descriptor \*/*  *uint8\_t RIFF[4]; // RIFF Header Magic header*  *uint32\_t ChunkSize; // RIFF Chunk Size*  *uint8\_t WAVE[4]; // WAVE Header*  */\* "fmt" sub-chunk \*/*  *uint8\_t fmt[4]; // FMT header*  *uint32\_t Subchunk1Size; // Size of the fmt chunk*  *uint16\_t AudioFormat; // Audio format 1=PCM,6=mulaw,7=alaw, 257=IBM Mu-Law, 258=IBM A-Law, 259=ADPCM*  *uint16\_t NumOfChan; // Number of channels 1=Mono 2=Sterio*  *uint32\_t SamplesPerSec; // Sampling Frequency in Hz*  *uint32\_t bytesPerSec; // bytes per second*  *uint16\_t blockAlign; // 2=16-bit mono, 4=16-bit stereo*  *uint16\_t bitsPerSample; // Number of bits per sample*  */\* "data" sub-chunk \*/*  *uint8\_t Subchunk2ID[4]; // "data" string*  *uint32\_t Subchunk2Size; // Sampled data length*  *} wav\_hdr;*  *int getFileSize(FILE\* inFile);*  *int main(int argc, char\* argv[])*  *{*  *wav\_hdr wavHeader;*  *int headerSize = sizeof(wav\_hdr), filelength = 0;*  *const char\* filePath;*  *string input;*  *if (argc <= 1)*  *{*  *cout << "Input wave file name: ";*  *cin >> input;*  *cin.get();*  *filePath = input.c\_str();*  *}*  *else*  *{*  *filePath = argv[1];*  *cout << "Input wave file name: " << filePath << endl;*  *}*  *FILE\* wavFile = fopen(filePath, "r"); // mở file wav ở chế độ đọc (r)*  *if (wavFile == nullptr)*  *{*  *fprintf(stderr, "Unable to open wave file: %s\n", filePath);*  *return 1;*  *}*  *//Read the header*  *size\_t bytesRead = fread(&wavHeader, 1, headerSize, wavFile);*  *cout << "Header Read " << bytesRead << " bytes." << endl; // độ dài header file wav (44 bytes)*  *if (bytesRead > 0)*  *{*  *//Read the data*  *uint16\_t bytesPerSample = wavHeader.bitsPerSample / 8; //Number of bytes per sample*  *if (bytesPerSample != 0)uint64\_t numSamples = wavHeader.ChunkSize / bytesPerSample; //How many samples are in the wav file?*  *static const uint16\_t BUFFER\_SIZE = 4096;*  *int8\_t\* buffer = new int8\_t[BUFFER\_SIZE];*  *while ((bytesRead = fread(buffer, sizeof buffer[0], BUFFER\_SIZE / (sizeof buffer[0]), wavFile)) > 0)*  *{*  */\*\* DO SOMETHING WITH THE WAVE DATA HERE \*\*/*  *cout << "Read " << bytesRead << " bytes." << endl;*  *}*  *delete[] buffer;*  *buffer = nullptr;*  *filelength = getFileSize(wavFile); // tính kích thước tệp*  *cout << "File is :" << filelength << " bytes." << endl;*  *cout << "RIFF header :" << wavHeader.RIFF[0] << wavHeader.RIFF[1] << wavHeader.RIFF[2] << wavHeader.RIFF[3] << endl;*  *cout << "WAVE header :" << wavHeader.WAVE[0] << wavHeader.WAVE[1] << wavHeader.WAVE[2] << wavHeader.WAVE[3] << endl;*  *cout << "FMT :" << wavHeader.fmt[0] << wavHeader.fmt[1] << wavHeader.fmt[2] << wavHeader.fmt[3] << endl;*  *cout << "Data size :" << wavHeader.ChunkSize << endl;*  *// Hiển thị Tỷ lệ lấy mẫu từ header*  *cout << "Sampling Rate :" << wavHeader.SamplesPerSec << endl; // Mẫu mỗi giây*  *cout << "Number of bits used :" << wavHeader.bitsPerSample << endl; // Số lượng bits được sử dụng*  *cout << "Number of channels :" << wavHeader.NumOfChan << endl; // Số kênh*  *cout << "Number of bytes per second :" << wavHeader.bytesPerSec << endl; // Số byte mỗi giây*  *cout << "Data length :" << wavHeader.Subchunk2Size << endl; // Độ dài dữ liệu*  *cout << "Audio Format :" << wavHeader.AudioFormat << endl; // Định dạng âm thanh*  *// Audio format 1=PCM,6=mulaw,7=alaw, 257=IBM Mu-Law, 258=IBM A-Law, 259=ADPCM*  *cout << "Block align :" << wavHeader.blockAlign << endl; // Căn chỉnh khối*  *cout << "Data string :" << wavHeader.Subchunk2ID[0] << wavHeader.Subchunk2ID[1] << wavHeader.Subchunk2ID[2] << wavHeader.Subchunk2ID[3] << endl;*  *// Chuỗi dữ liệu*  *}*  *fclose(wavFile); // đóng file*  *return 0;*  *}*  *// find the file size*  *int getFileSize(FILE\* inFile)*  *{*  *int fileSize = 0;*  *fseek(inFile, 0, SEEK\_END); // đặt con trỏ ghi tại vị trí thứ 0 bắt đầu từ phần cuối file*  *fileSize = ftell(inFile); // lấy tổng kích thước của một file sau khi di chuyển con trỏ tệp ở cuối tệp*  *fseek(inFile, 0, SEEK\_SET); // đặt lại con trỏ ghi tại vị trí thứ 0 bắt đầu từ phần đầu file*  *return fileSize;*  *}* |

##### **Kết quả chạy thử chương trình**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

#### Hiển thị dạng sóng

##### **Code**

|  |
| --- |
| *using System;*  *using System.Collections.Generic;*  *using System.ComponentModel;*  *using System.Data;*  *using System.Drawing;*  *using System.Linq;*  *using System.Text;*  *using System.Threading.Tasks;*  *using System.Windows.Forms;*  *namespace Plotting\_Audio\_Waveforms*  *{*  *public partial class Form1 : Form*  *{*  *public Form1()*  *{*  *InitializeComponent();*  *}*  *private void openWaveToolStripMenuItem\_Click(object sender, EventArgs e)*  *{*  *OpenFileDialog open = new OpenFileDialog();*  *open.Filter = "Wave File (\*.wav)|\*.wav;";*  *if (open.ShowDialog() != DialogResult.OK) return;*  *chart1.Series.Add("wave");*  *chart1.Series["wave"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.FastLine;*  *chart1.Series["wave"].ChartArea = "ChartArea1";*  *NAudio.Wave.WaveChannel32 wave = new NAudio.Wave.WaveChannel32(new NAudio.Wave.WaveFileReader(open.FileName));*  *byte[] buffer = new byte[16384];*  *int read = 0;*  *while (wave.Position < wave.Length)*  *{*  *read = wave.Read(buffer, 0, 16384);*  *for (int i = 0; i < read / 4; i++)*  *{*  *chart1.Series["wave"].Points.Add(BitConverter.ToSingle(buffer, i \* 4));*  *}*  *}*  *}*  *}*  *}* |

##### **Kết quả chạy thử chương trình với bài APM\_Adobe\_Going\_Home\_v3.wav**

**Chart, histogram

Description automatically generated**

##### **Kết quả chạy file wav trên phần mềm Audacity**

**Chart

Description automatically generated with medium confidence**

Qua phần mềm ta thấy được trục tung là biên độ và trục hoành là thời gian phát của âm thanh.

### Mức 2

Phân tích phổ và hiển thị.

##### **Code Form1.cs**

|  |
| --- |
| *using System;*  *using System.Collections.Generic;*  *using System.ComponentModel;*  *using System.Data;*  *using System.Drawing;*  *using System.Linq;*  *using System.Text;*  *using System.Threading.Tasks;*  *using System.Windows.Forms;*  *using NAudio.Wave;*  *using NAudio.CoreAudioApi;*  *namespace ScottPlotMicrophoneFFT*  *{*  *public partial class Form1 : Form*  *{*  *// MICROPHONE ANALYSIS SETTINGS*  *private int RATE = 44100; // sample rate of the sound card*  *private int BUFFERSIZE = (int)Math.Pow(2, 11); // must be a multiple of 2*  *// prepare class objects*  *public BufferedWaveProvider bwp;*  *public Form1()*  *{*  *InitializeComponent();*  *SetupGraphLabels();*  *StartListeningToMicrophone();*  *timerReplot.Enabled = true;*  *}*  *void AudioDataAvailable(object sender, WaveInEventArgs e)*  *{*  *bwp.AddSamples(e.Buffer, 0, e.BytesRecorded);*  *}*  *private void Form1\_Load(object sender, EventArgs e)*  *{*  *}*  *public void SetupGraphLabels()*  *{*  *scottPlotUC1.fig.labelTitle = "Microphone PCM Data";*  *scottPlotUC1.fig.labelY = "Amplitude (PCM)";*  *scottPlotUC1.fig.labelX = "Time (ms)";*  *scottPlotUC1.Redraw();*  *scottPlotUC2.fig.labelTitle = "Microphone FFT Data";*  *scottPlotUC2.fig.labelY = "Power (raw)";*  *scottPlotUC2.fig.labelX = "Frequency (Hz)";*  *scottPlotUC2.Redraw();*  *}*  *public void StartListeningToMicrophone(int audioDeviceNumber = 0)*  *{*  *WaveIn wi = new WaveIn();*  *wi.DeviceNumber = audioDeviceNumber;*  *wi.WaveFormat = new NAudio.Wave.WaveFormat(RATE, 1);*  *wi.BufferMilliseconds = (int)((double)BUFFERSIZE / (double)RATE \* 1000.0);*  *wi.DataAvailable += new EventHandler<WaveInEventArgs>(AudioDataAvailable);*  *bwp = new BufferedWaveProvider(wi.WaveFormat);*  *bwp.BufferLength = BUFFERSIZE \* 2;*  *bwp.DiscardOnBufferOverflow = true;*  *try*  *{*  *wi.StartRecording();*  *}*  *catch*  *{*  *string msg = "Could not record from audio device!\n\n";*  *msg += "Is your microphone plugged in?\n";*  *msg += "Is it set as your default recording device?";*  *MessageBox.Show(msg, "ERROR");*  *}*  *}*  *private void Timer\_Tick(object sender, EventArgs e)*  *{*  *// turn off the timer, take as long as we need to plot, then turn the timer back on*  *timerReplot.Enabled = false;*  *PlotLatestData();*  *timerReplot.Enabled = true;*  *}*  *public int numberOfDraws = 0;*  *public bool needsAutoScaling = true;*  *public void PlotLatestData()*  *{*  *// check the incoming microphone audio*  *int frameSize = BUFFERSIZE;*  *var audioBytes = new byte[frameSize];*  *bwp.Read(audioBytes, 0, frameSize);*  *// return if there's nothing new to plot*  *if (audioBytes.Length == 0)*  *return;*  *if (audioBytes[frameSize - 2] == 0)*  *return;*  *// incoming data is 16-bit (2 bytes per audio point)*  *int BYTES\_PER\_POINT = 2;*  *// create a (32-bit) int array ready to fill with the 16-bit data*  *int graphPointCount = audioBytes.Length / BYTES\_PER\_POINT;*  *// create double arrays to hold the data we will graph*  *double[] pcm = new double[graphPointCount];*  *double[] fft = new double[graphPointCount];*  *double[] fftReal = new double[graphPointCount/2];*    *// populate Xs and Ys with double data*  *for (int i = 0; i < graphPointCount; i++)*  *{*  *// read the int16 from the two bytes*  *Int16 val = BitConverter.ToInt16(audioBytes, i \* 2);*  *// store the value in Ys as a percent (+/- 100% = 200%)*  *pcm[i] = (double)(val) / Math.Pow(2,16) \* 200.0;*  *}*  *// calculate the full FFT*  *fft = FFT(pcm);*  *// determine horizontal axis units for graphs*  *double pcmPointSpacingMs = RATE / 1000;*  *double fftMaxFreq = RATE / 2;*  *double fftPointSpacingHz = fftMaxFreq / graphPointCount;*  *// just keep the real half (the other half imaginary)*  *Array.Copy(fft, fftReal, fftReal.Length);*    *// plot the Xs and Ys for both graphs*  *scottPlotUC1.Clear();*  *scottPlotUC1.PlotSignal(pcm, pcmPointSpacingMs, Color.Blue);*  *scottPlotUC2.Clear();*  *scottPlotUC2.PlotSignal(fftReal, fftPointSpacingHz, Color.Blue);*  *// optionally adjust the scale to automatically fit the data*  *if (needsAutoScaling)*  *{*  *scottPlotUC1.AxisAuto();*  *scottPlotUC2.AxisAuto();*  *needsAutoScaling = false;*  *}*  *//scottPlotUC1.PlotSignal(Ys, RATE);*  *numberOfDraws += 1;*  *lblStatus.Text = $"Analyzed and graphed PCM and FFT data {numberOfDraws} times";*  *// this reduces flicker and helps keep the program responsive*  *Application.DoEvents();*  *}*  *private void autoScaleToolStripMenuItem\_Click(object sender, EventArgs e)*  *{*  *needsAutoScaling = true;*  *}*  *private void infoMessageToolStripMenuItem\_Click(object sender, EventArgs e)*  *{*  *string msg = "";*  *msg += "left-click-drag to pan\n";*  *msg += "right-click-drag to zoom\n";*  *msg += "middle-click to auto-axis\n";*  *msg += "double-click for graphing stats\n";*  *MessageBox.Show(msg);*  *}*    *public double[] FFT(double[] data)*  *{*  *double[] fft = new double[data.Length];*  *System.Numerics.Complex[] fftComplex = new System.Numerics.Complex[data.Length];*  *for (int i = 0; i < data.Length; i++)*  *fftComplex[i] = new System.Numerics.Complex(data[i], 0.0);*  *Accord.Math.FourierTransform.FFT(fftComplex, Accord.Math.FourierTransform.Direction.Forward);*  *for (int i = 0; i < data.Length; i++)*  *fft[i] = fftComplex[i].Magnitude;*  *return fft;*  *}*  *}*  *}* |

##### **Kết quả chạy thử chương trình**

A screenshot of a computer

Description automatically generated with medium confidence

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## KẾT QUẢ ĐẠT ĐƯỢC

* + Đọc được thông tin header của file wav, hiển thị dạng sóng.
  + Hiển thị phổ âm thanh.

## ƯU ĐIỂM

* + Chương trình dễ sử dụng, giao diện thân thiện.
  + Bài toán hiển thị phổ âm thanh có thể làm tiền đề để giải quyết các bài toán khó hơn, phức tạp hơn

## HẠN CHẾ

* Giao diện hệ thống chưa được tối ưu cho trải nghiệm người dùng.
* Code chương trình còn dài, còn nhiều chỗ chưa được tối ưu.

## HƯỚNG PHÁT TRIỂN

* Hoàn thiện thêm những chức năng hệ thống hiện có, cải thiện về giao diện, trải nghiệm người.

## KẾT LUẬN

Ngày nay, cuộc cách mạng công nghiệp 4.0 và vai trò của Internet ngày càng được khẳng định và không thể thiếu trong quản lý và chia sẻ thông tin của một quốc gia hay địa phương. Cùng với sự phát triển mạnh mẽ của hệ thống mạng toàn cầu – Internet và nhu cầu chia sẻ, tra cứu thông tin trên Internet.

Định dạng tệp WAV không nén, nên nó hoạt động tốt hơn về chất lượng âm thanh và tốc độ bit, không giới hạn về độ sâu bit.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Wikipedia tiếng việt, "C Sharp (ngôn ngữ lập trình)," 07 01 2021. [Online]. Available: https://vi.wikipedia.org/wiki/C\_Sharp\_(ng%C3%B4n\_ng%E1%BB%AF\_l%E1%BA%ADp\_  tr%C3%ACnh). [Accessed 07 01 2021]. |
| [2] | Wikipedia tiếng việt, “Biến đổi Fourier,” 7 1 2022. [Trực tuyến]. Available: https://vi.wikipedia.org/wiki/Bi%E1%BA%BFn\_%C4%91%E1%BB%95i\_Fourier. [Đã truy  cập 7 1 2022]. |
| [3] | Wikipedia tiếng việt, "Biến đổi Fourier rời rạc," 7 1 2022. [Online]. Available: https://vi.wikipedia.org/wiki/Bi%E1%BA%BFn\_%C4%91%E1%BB%95i\_Fourier\_r%E1%  BB%9Di\_r%E1%BA%A1c. [Accessed 7 1 2022]. |
| [4] | Wikipedia, “Microsoft Visual Studio,” 7 1 2022. [Trực tuyến]. Available: https://vi.wikipedia.org/wiki/Microsoft\_Visual\_Studio. [Đã truy cập 7 1 2022]. |
| [5] | Wikipedia tiếng việt, "C++," 7 1 2022. [Online]. Available: https://vi.wikipedia.org/wiki/C%2B%2B. [Accessed 7 1 2022]. |
| [6] | GiawaVideos, "C# Audio Tutorial 10 - Plotting Audio Waveforms," [Online]. Available: https://www.youtube.com/watch?v=ZnFoVuOVrUQ. |
| [7] | S. Harden, “Realtime Microphone Audio FFT Graph with C#,” [Trực tuyến]. Available: https://www.youtube.com/watch?v=q9cRZuosrOs. |
| [8] | FILEFORMAT, "WAV - Waveform Audio File Format," [Online]. Available: https://docs.fileformat.com/audio/wav/. |
| [9] | stackoverflow, "C++ Reading the Data part of a WAV file," [Online]. Available: https://stackoverflow.com/questions/13660777/c-reading-the-data-part-of-a-wav-file. |
| [10] | O. S. Code, "Reading the header of a wav file - One Step! Code," [Online]. Available: https://onestepcode.com/read-wav-header/. |
| [11] | PLANETCALC, "Audio file waveform - PLANETCALC Online calculators," [Online]. Available: https://planetcalc.com/8627/. |