LẬP TRÌNH ÂM THANH

Spring 2022- D19PT

Nhóm Bài tập lớn: SNDPROJSP2203		Bài: **PRJ01**	
Họ và tên:Đỗ Tuấn Dương	Mã sinhviên: B19DCPT035	Nhóm lớp: D19-147	**10**
Họ và tên : Nguyễn Thành Đô (NT)	Mã sinh viên: B19DCPT052	Nhóm lớp: D19-147	**10**
Họ và tên : Bùi Thị Mai	Mã sinh viên: B19DCPT154	Nhóm lớp: D19-147	**10**
Mức độ hoàn thành: **100%**			
Đánh giá cụ thể			

Hàm	Cài đặt	Script sử dụng
Class Wave		
+ Wave(): constructor	Dương	C++
+ Wave(string fileName): constructor	Dương	C++
+ copy(): Wave	Đô	C++
+ overload operator + (Wave wav): Wave	Đô	C++
+ overload operator * (int n): Wave	Mai	C++

+ overload operator * (Wave wav): Wave	Đô	C++
+ reverseWave(): Wave	Mai	C++
+ delayWave(int second): Wave	Đô	C++
+ echoWave(int times): Wave	Dương ,Đô	C++
+ tuongQuanCheo(Wave wav): constructor	Dương	C++
+ tíchChap(Wave wave): Wave	Mai	
+ setSampleRate(int Hz): void	Đô	C++
+ setChannel(int numberOfChannel): void	Mai	C++
+ play(): void	Đô	C++
+ getDurationAsSeconds(): int	Mai	C++
+ getChannel(): int	Đô	C++
+ getSampleCount(): unsigned long long	Mai	C++
+ getSamples(): vector<unsigned short>	Dương	C++
+ loadFromFile(string fileName): void	Mai	C++

+ getSource(): string	Mai	C++
+ getSampleRate(): int	Dương	C++
+ saveToFile(string fileName): void	Mai	C++
- toLow(vector<unsigned short> arr): vector<double>	Mai	C++
- toHigh(vector<double> arr): vector<unsigned short>	Đô	C++
GUI		
mainMenu(): void	Đô	C++
listenWave(Wave wav): void	Dương	C++
configureWave(Wave &wav): void	Đô	C++
getWaveInfo(const Wave &wav): void	Đô	C++
changeSourceFile(Wave &wav): void	Dương	C++
changeSampleRate(Wave &wav): void	Mai	C++
changeNumChannel(Wave &wav): void	Mai	C++

c_reverseSamples(Wave &wav): void	Đô	C++
c_MulSamplesWithConst(Wave &wav): void	Dương ,Đô	C++
c_echoSamples(Wave &wav): void	Dương	C++
c_delaySamples(Wave &wav): void	Đô	C++
selectWave(int &n): void	Mai	C++
c_addTwoWave(Wave &wav): void	Đô	C++
c_mulTwoWave(Wave &wav: void	Mai	C++
c_tichChap(Wave &wav): void	Đô	C++
c_tuongQuanCheo(Wave &wav): void	Mai	C++
writeToFile(Wave &wav): void	Dương	C++
selectWave(int &n): void	Đô	C++

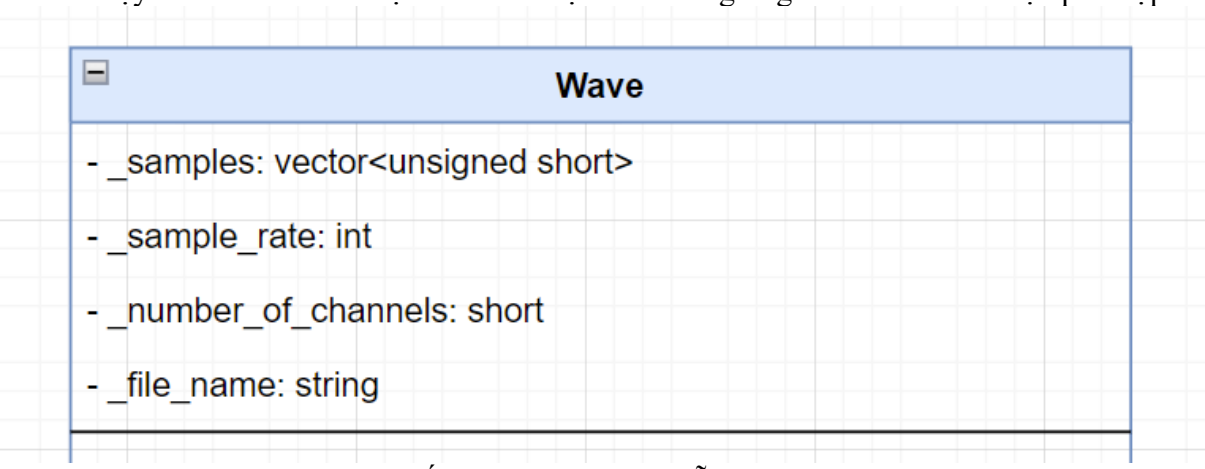
Báo cáo trình bày chi tiết:

- **Cách thiết kế xây dựng đối tượng**
- UML cho các lớp

- Các phần code chính trong cài đặt & thực hiện minh họa sử dụng kèm theo các giải thích
- Các kết quả (minh chứng: snapshot, hình vẽ, …)
- Kết luận
- Tài liệu tham khảo

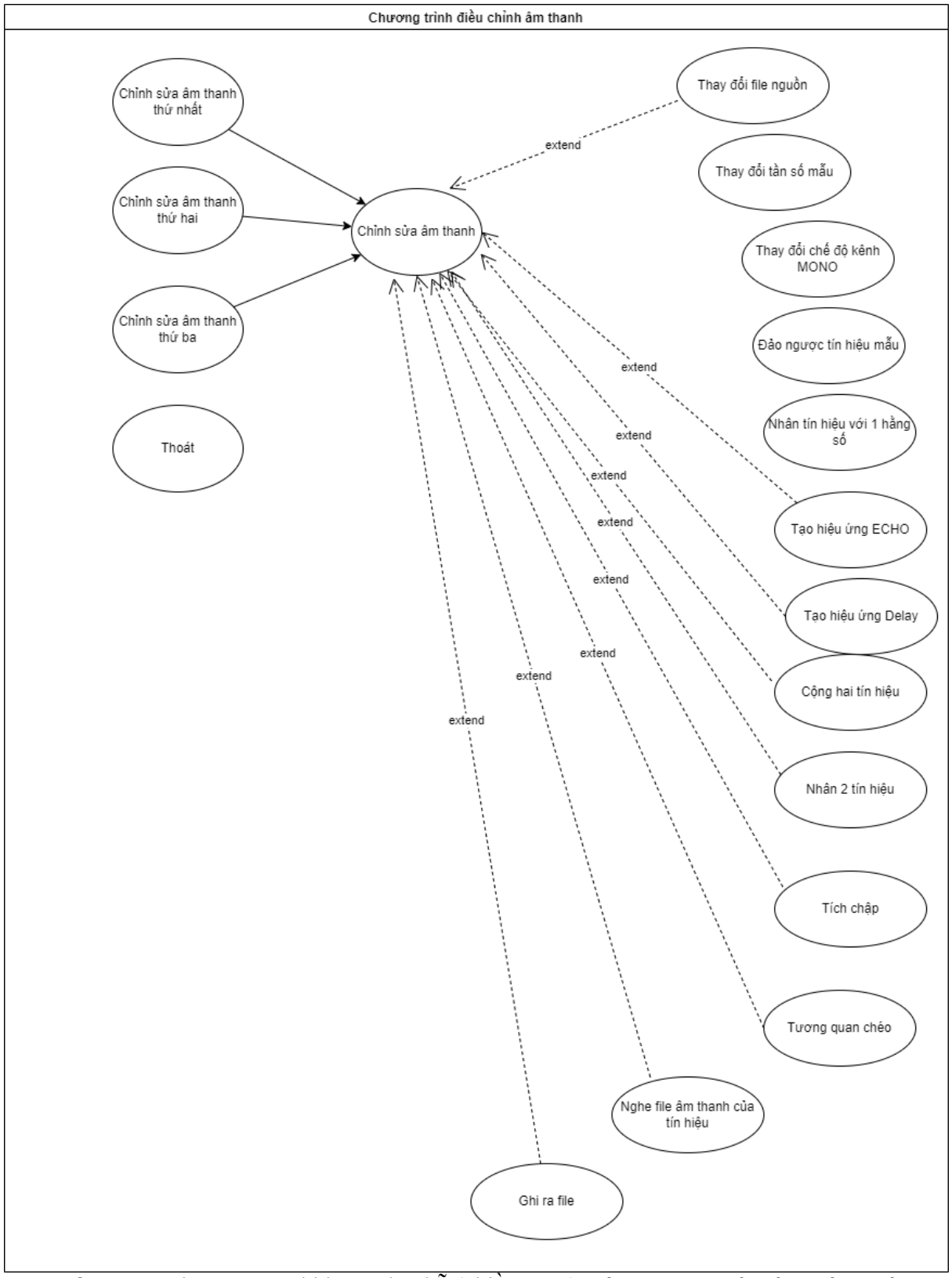
I.Cách thiết kế xây dựng đối tượng

- Một tín hiệu âm thanh ta sẽ cần các thông tin như là: Tần số lấy mẫu, Mảng các tín hiệu, Tên file nguồn lấy mẫu (nếu có), Số lượng kênh (ở file wav thì thường sẽ có 1 hoặc 2 kênh, MONO và STEREO).
- Vậy nên nhóm em sẽ đặt nó có 4 thuộc tính tương ứng với các kiểu dữ liệu phù hợp:



  - _samples: Mảng số nguyên chứa các mẫu.
  - _sample_rate: Tần số lấy mẫu.
  - _number_of_channels: Số lượng kênh của file wav.
  - _file_name: Tên của file nguồn load vào thực thể Wave.
- Ngoài ra sẽ có một số phương thức getter và setter để làm việc với các thuộc tính trên.
- Sử dụng các phương thức hoặc nạp chồng toán tử (Overload operator) để thực hiện:
  - Phép cộng 2 tín hiệu.
  - Phép nhân tín hiệu với 1 hằng số.
  - Phép nhân 2 tín hiệu.
  - Tạo hiệu ứng echo (thay đổi mảng tần số mẫu).
  - Tạo hiệu ứng Delay (thay đổi mảng tần số mẫu).
- Do để đa dạng cho mục đích sử dụng sau này, các phương thức không tác động thẳng vào **tín hiệu gốc** mà nó sẽ **tạo ra tín hiệu mới**.
- Nếu muốn **thay đổi tín hiệu gốc** thì thực hiện phép gán với chính phương thức của nó.
- Workflow: do file không đủ nên em xin gửi link github có kèm theo file pdf https://github.com/ThanhDo506/AudioProject.git

- Use case:

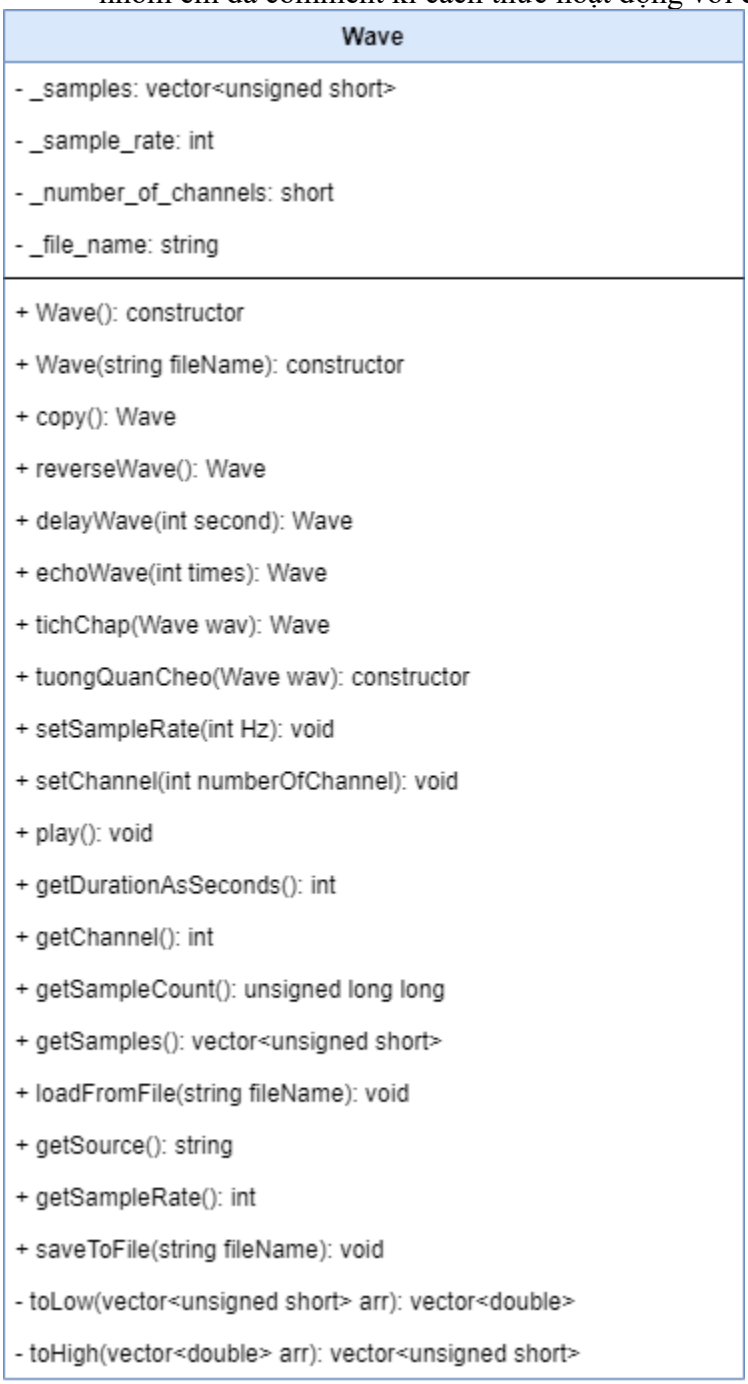


- **Work flow**: Do cho vào word không đủ chỗ (chiều rộng) nên em xin phép tách riêng thành 1 file PDF.

-

II. UML các lớp

- Ở đây nhóm chỉ xây dựng lớp Wave tượng trưng cho tín hiệu. (Các phương thức nhóm em đã comment kĩ cách thức hoạt động với các tham số và giá trị trả về).



III. Cài đặt

- sf::Int16 tức là biến int 16 bit tương đương với kiểu short

1.Phép cộng hai tín hiệu

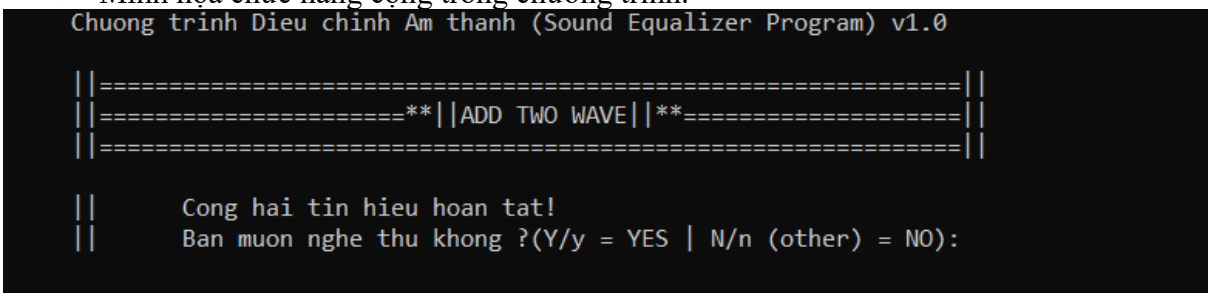
- Dưới đây là đoạn code nhóm em nạp chồng toán tử + để cộng hai tín hiệu Wave (Signal) kết quả trả về một object Wave (Signal) mới.
- Đoạn code trên từ **line 19 - 35** của file **Wave.cpp**

```cpp
// * Phép cộng 2 tín hiệu | Trộn tín hiệu.
// * @param other Obj Wave khác
// * @return new Wave
Wave operator+(const Wave &other) {
    Wave wav = other;
    vector<sf::Int16> w1 = this->_samples, w2 = other._samples;
    if (w1.size() < w2.size()) {
        w1.resize(w2.size(), 0);
    } else {
        w2.resize(w1.size(), 0);
    }
    for (unsigned long long i = 0; i < w1.size(); i++) {
        w1[i] += w2[i];
    }
    wav._samples = w1;
    return wav;
}
```

- Sau đây là hình ảnh minh họa thực hiện phép cộng hai tín hiệu trong chương trình chính (dấu gạch chân đỏ).
- Đoạn code từ line 495 - 519 của file GUI.cpp

```cpp
void c_addTwoWave(Wave &wav) {
    int n;
    char c;
    selectWave(n);
    system("cls");
    printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
    printf("\n");
    printf("\t||═══════════════════════════════════════════════════════════════||\n");
    printf("\t||═══════════════**||ADD TWO WAVE||**═══════════════||\n");
    printf("\t||═══════════════════════════════════════════════════════════════||\n");
    waves[0] = wav + waves[n];
    printf("\n\t||\tCong hai tin hieu hoan tat!");
    printf("\n\t||\tBan muon nghe thu khong ?(Y/y = YES | N/n (other) = NO): ");
    cin >> c;
    if (c == 'Y' || c == 'y') {
        printf("\n\t||\tPlaying ...\n");
        listenWave(waves[0]);
        printf("\n\t||\tDONE.\n");
    }
    printf("\n\t||\tBan co muon ghi ra file khong ?(Y/y = YES | N/n (other) = NO): ");
    cin >> c;
    if (c == 'Y' || c == 'y') {
        writeToFile(waves[0]);
    }
}
```

- Minh họa chức năng cộng trong chương trình.



2.Phép nhân hai tín hiệu

- Dưới đây là đoạn code nhóm em nạp chồng toán tử * để nhân hai tín hiệu Wave (Signal) kết quả trả về một object Wave (Signal) mới.
- Đoạn code trên từ **line 48 - 68** của file **Wave.cpp**

```cpp
// * Phép nhân hai tín hiệu.
// * @param wave
// * @return new Wave
Wave operator*(const Wave &other) {
    Wave wav = other;
    // * Biến đổi sang dạng thập phân trong khoảng -1 -> 1
    vector<double> v_wav = Wave::toLow(this->_samples),
                   v_other = Wave::toLow(other._samples);
    if (v_wav.size() < v_other.size()) {
        v_wav.resize(v_other.size(), 0);
    } else {
        v_other.resize(v_wav.size(), 0);
    }
    for (unsigned long long i = 0; i < v_wav.size(); i++) {
        v_wav[i] *= v_other[i];
    }
    // * Biến đổi ngược trở lại thành dạng số nguyên
    // * Trong khoảng từ -32767 -> 32767 (tức trong khoảng +- 2^15 - 1)
    wav._samples = toHigh(v_wav);
    return wav;
}
```

- Dưới đây là đoạn code sử dụng trong chương trình để minh họa phép nhân hai tín hiệu với nhau để tạo ra một tín hiệu mới.
- Line 540 - 565 file GUI.cpp

```
void c_mulTwoWave(Wave &wav) {
    int n;
    char c;
    selectWave(n);
    system("cls");
    printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
    printf("\n");
    printf("\t||═══════════════════════════════════════════════════════════════════════════||\n");
    printf("\t||═══════════════════════**||MUL TWO WAVE||**═══════════════════════||\n");
    printf("\t||═══════════════════════════════════════════════════════════════════════════||\n");
    waves[0] = wav * waves[n];
    printf("\n\t||\tNhan hai tin hieu hoan tat!");
    printf("\n\t||\tBan muon nghe thu khong ?(Y/y = YES | N/n (other) = NO): ");
    cin >> c;
    if (c == 'Y' || c == 'y') {
        printf("\n\t||\tPlaying ...\n");
        listenWave(waves[0]);
        printf("\n\t||\tDONE.\n");
    }
    printf("\n\t||\tBan co muon ghi ra file khong ?(Y/y = YES | N/n (other) = NO): ");
    cin >> c;
    if (c == 'Y' || c == 'y') {
        writeToFile(waves[0]);
    }
}
```

- Minh họa đoạn script trên chạy trong chương trình.

```
D:\AudioProject-master\build\program.exe
    Chuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0

    ||============================================================||
    ||======================**||MUL TWO WAVE||**====================||
    ||============================================================||

    ||      Nhan hai tin hieu hoan tat!
    ||      Ban muon nghe thu khong ?(Y/y = YES | N/n (other) = NO):
```

3. Nhân tín hiệu với hằng số

- Dưới đây là đoạn script nạp chồng toán tử * để nhân Wave (Signal) với một hằng số kết quả trả về một object Wave (Signal) mới.
- Đoạn code trên từ **line 37 - 47** của file **Wave.cpp**

```
// * Phép nhân tín hiệu với một hằng số.
// * @param number    hằng
// * @return new Wave
Wave operator*(const double &number) {
    Wave wav = this->copy();
    for (unsigned long long i = 0; i < _samples.size(); i++) {
        wav._samples.at(i) *= number;
    }
    return wav;
}
```

- Đoạn script thực hiện tính năng nhân với một hằng số, minh họa nhân tín hiệu với một hằng số.
- Line 394 - 435 file GUI.cpp

```
void c_MulSamplesWithConst(Wave &wav) {
    double n;
    bool check = true;
    char c;
    while(check){
        system("cls");
        printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
        printf("\n");
        printf("\t||═══════════════════════════════════════════════════════════════||\n");
        printf("\t||═══════════════**||MULTIPLE CONST||**═══════════════||\n");
        printf("\t||═══════════════════════════════════════════════════════════════||\n");
        printf("\n");
        printf("\n||\tBan muon tin hieu lap lai may lan (>1): ");
        cin >> n;
        if(n ≤ 0) {···
        } else if (n ≥ 5) {···
        printf("\t||\tDang xu li  ...\n");
        wav = wav * n;
        printf("\t||\tXu li thanh cong!\n");
        printf("\t");
        system("pause");
    }
}
```

4. Hiệu ứng echo

- Phương thức echoWave(int n) của class Wave / Signal cho phép lặp đoạn với n lần, giá trị trả ra là một tín hiệu mới với n lần lặp mẫu của tín hiệu cũ.
- Line 89 - 99 file Wave.cpp

```cpp
// * Tạo ra một echo Wave
// * @param n   Số lần lặp
// * @return  Wave
Wave echoWave(const int &n) {
    Wave wav = this->copy();
    for (int i = 1; i < n; i++) {
        wav._samples.insert(wav._samples.end(), this->_samples.begin(),
                            this->_samples.end());
    }
    return wav;
}
```

- Minh họa sử dụng phương thức trong chức năng tạo tín hiệu lặp
- Line 353 - 392 file GUI.cpp

```cpp
void c_echoSamples(Wave &wav) {
    int n;
    bool check = true;
    char c;
    while(check){
        system("cls");
        printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
        printf("\n");
        printf("\t||═══════════════════════════════════════════════════════════════||\n");
        printf("\t||═══════════════════**||MAKE ECHO SOUND||**═══════════════════||\n");
        printf("\t||═══════════════════════════════════════════════════════════════||\n");
        printf("\n");
        printf("\n||\tBan muon tin hieu lap lai may lan (>1): ");
        cin >> n;
        if(n ≤ 0) {···
        } else if (n ≥ 5) {···
        printf("\t||\tDang xu li ...\n");
        wav = wav.echoWave(n);
        printf("\t||\tXu li thanh cong!\n");
        printf("\t");
        check = false;
        system("pause");
    }
}
```

```
        Chuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0

        ||===============================================================||
        ||====================**||MAKE ECHO SOUND||**====================||
        ||===============================================================||


||      Ban muon tin hieu lap lai may lan (>1): _
```

5. Phép trễ tín hiệu

- Phương thức này trả về tín hiệu bị làm trễ thêm mili giây so với tín hiệu gốc.

- Line 101 - 109 file Wave.cpp

```
// * Tạo ra một Wave bị delay.
// * @param millisecond Số giây delay
// * @return  new Wave
Wave delayWave(const int &milliseconds) {
    Wave wav = this->copy();
    wav._samples.insert(wav._samples.begin(),
                        _sample_rate * 1.0f / 60 * milliseconds, 0);
    return wav;
}
```

- Minh họa sử dụng phương thức bằng chức năng làm trễ tín hiệu.
- Line 436 - 452 file GUI.cpp

```
void c_delaySamples(Wave &wav) {
    int n;
    system("cls");
    printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
    printf("\n");
    printf("\t||═══════════════════════════════════════════════════════════════||\n");
    printf("\t||═══════════════════**||MAKING DELAY||**═══════════════════||\n");
    printf("\t||═══════════════════════════════════════════════════════════════||\n");
    printf("\n");
    printf("\n||\tBan muon tin hieu delay nhieu s: ");
    cin >> n;
    printf("\t||\tDang xu li ...\n");
    wav = wav.delayWave(n);
    printf("\t||\tXu li thanh cong!\n");
    printf("\t");
    system("pause");
}
```

```
    Chuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0

    ||==============================================================||
    ||====================**||MAKING DELAY||**====================||
    ||==============================================================||


    Ban muon tin hieu delay nhieu s: 2
    ||      Dang xu li ...
    ||      Xu li thanh cong!
    Press any key to continue . . .
```

6. Phép tích chập

- Phương thức tích chập sẽ tạo ra một thực thể tín hiệu mới là tích chập của tín hiệu gốc với một tín hiệu khác.
- Line 111 - 136 file Wave.cpp

```
111     // * Tích chập của Wave này với Wave other
112     // * @param Wave    other
113     // * @return Wave là tích chập
        ...
114     Wave tichChap(const Wave &other) {
115         Wave wav;
116         vector<double> wav_t = Wave::toLow(this->_samples),
117                        wav_o = Wave::toLow(other._samples);
118         wav._samples.resize(wav_t.size() + wav_o.size() - 1, 0);
119         vector<vector<double>> matrix(wav_t.size());
120         for (int i = 0; i < matrix.size(); i++) {
121             matrix[i].resize(wav_o.size(), 0);
122             for (int j = 0; j < matrix[i].size(); j++) {
123                 matrix[i][j] = wav_t[i] * wav_o[j];
124             }
125         }
126         for (unsigned long long k = 0; k < wav._samples.size(); k++) {
127             for (int i = 0; i < wav_t.size(); i++) {
128                 for (int j = 0; j < wav_o.size(); j++) {
129                     if (i + j - 1 == k) {
130                         wav._samples[k] += matrix[i][j];
131                     }
132                 }
133             }
134         }
135         return wav;
136     }       You, March 15th, 2022 6:14pm • Nope …
```

- Minh họa sử dụng phương thức, rồi gán vào một tín hiệu mới.
- Line 567 - 592 file GUI.cpp

```
...
void c_tichChap(Wave &wav) {
    int n;
    char c;
    selectWave(n);
    system("cls");
    printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
    printf("\n");
    printf("\t||═══════════════════════════════════════════════════════════════════════════||\n");
    printf("\t||═══════════════════**||TICH CHAP||**═══════════════════||\n");
    printf("\t||═══════════════════════════════════════════════════════════════════════════||\n");
    printf("\n\t||\tDang xu li ...");
    waves[0] = wav.tichChap(waves[n]);
    printf("\n\t||\tTich chap hai tin hieu hoan tat!");
    printf("\n\t||\tBan muon nghe thu khong ?(Y/y = YES | N/n (other) = NO): ");
    cin >> c;
    if (c == 'Y' || c == 'y') {
        printf("\n\t||\tPlaying ...\n");
        listenWave(waves[0]);
        printf("\n\t||\tDONE.\n");
    }
    printf("\n\t||\tBan co muon ghi ra file khong ?(Y/y = YES | N/n (other) = NO): ");
    cin >> c;
    if (c == 'Y' || c == 'y') {
        writeToFile(waves[0]);
    }
}
```

7. Phép tương quan chéo

- Phương thức tích chập sẽ tạo ra một thực thể tín hiệu mới là tương quan chéo của tín hiệu gốc với một tín hiệu khác.
- Line 141 - 168 file Wave.cpp

```
// * Tương quan chéo giữa 2 Wave
// * @param Wave    other
// * @return Wave là tương quan chéo
Wave tuongQuanCheo(const Wave &other) {
    Wave wav;
    vector<double> wav_t = Wave::toLow(this->_samples),
                   wav_o = Wave::toLow(other._samples);
    reverse(wav_t.begin(), wav_t.end());

    wav._samples.resize(wav_t.size() + wav_o.size() - 1, 0);

    vector<vector<double>> matrix(wav_t.size());

    for (int i = 0; i < matrix.size(); i++) {
        matrix[i].resize(wav_o.size(), 0);
        for (int j = 0; j < matrix[i].size(); j++) {
            matrix[i][j] = wav_t[i] * wav_o[j];
        }
    }

    for (unsigned long long k = 0; k < wav._samples.size(); k++) {
        for (int i = 0; i < wav_t.size(); i++) {
            for (int j = 0; j < wav_o.size(); j++) {
                if (i + j - 1 == k) {
                    wav._samples[k] += matrix[i][j];
                }
            }
        }
    }
    return wav;
}
```

- Minh họa sử dụng phương thức, rồi gán vào một tín hiệu mới.
- Line 595 - 621 file GUI.cpp

```
void c_tuongQuanCheo(Wave &wav) {
    int n;
    char c;
    selectWave(n);
    system("cls");
    printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
    printf("\n");
    printf("\t||═══════════════════════════════════════════════════════════════════════════||\n");
    printf("\t||═══════════════**||TUONG QUAN CHEO||**═══════════════||\n");
    printf("\t||═══════════════════════════════════════════════════════════════════════════||\n");
    printf("\n\t||\tDang xu li ...");
    waves[0] = wav.tuongQuanCheo(waves[n]);
    printf("\n\t||\tThuc hien phep tuong quan cheo hai tin hieu hoan tat!");
    printf("\n\t||\tBan muon nghe thu khong ?(Y/y = YES | N/n (other) = NO): ");
    cin >> c;
    if (c == 'Y' || c == 'y') {
        printf("\n\t||\tPlaying ...\n");
        listenWave(waves[0]);
        printf("\n\t||\tDONE.\n");
    }
    printf("\n\t||\tBan co muon ghi ra file khong ?(Y/y = YES | N/n (other) = NO): ");
    cin >> c;
    if (c == 'Y' || c == 'y') {
        writeToFile(waves[0]);
    }
}
```

8. Phép đảo tín hiệu

- Phương thức tạo ra một thực thể tín hiệu mới với tín hiệu gốc bị đảo ngược.
- Line 82 - 87 file Wave.cpp

```
// * Trả về một object Wave với Mẫu bị đảo ngược.
// * @return new Wave
Wave reverseWave() {
    Wave wav = this->copy();
    std::reverse(wav._samples.begin(), wav._samples.end());
    // wav._samples.reserve(wav._samples.size());
    return wav;
}
```

- Minh họa cho phương thức thông qua chức năng đảo ngược tín hiệu.
- Line 337 - 351 file GUI.cpp

```
void c_reverseSamples(Wave &wav) {
    system("cls");
    printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
    printf("\n");
    printf("\t||═══════════════════════════════════════════════════════════════||\n");
    printf("\t||═══════════════**||REVERSE SAMPLES||**═══════════════════════||\n");
    printf("\t||═══════════════════════════════════════════════════════════════||\n");
    printf("\n");
    printf("\t||\tDang xu li ...\n");
    wav = wav.reverseWave();
    printf("\n\tDao nguoc tin hieu thanh cong!\n");
    printf("\t");

    system("pause");
}
```

9. Thay đổi file nguồn

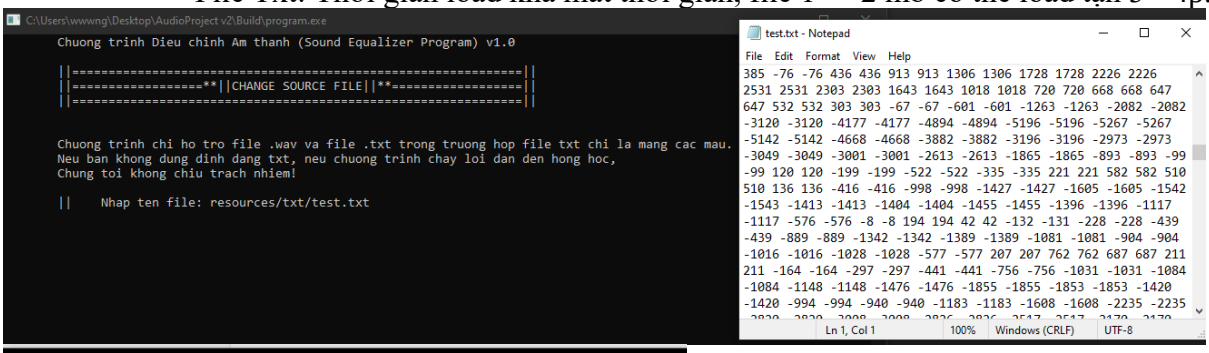
- Là phương thức giúp tải tần số mẫu từ file **txt** (chỉ chứa mảng các số biểu diễn các mẫu của tín hiệu) hoặc file **wav**.
- Line 221 - 244 file Wave.cpp

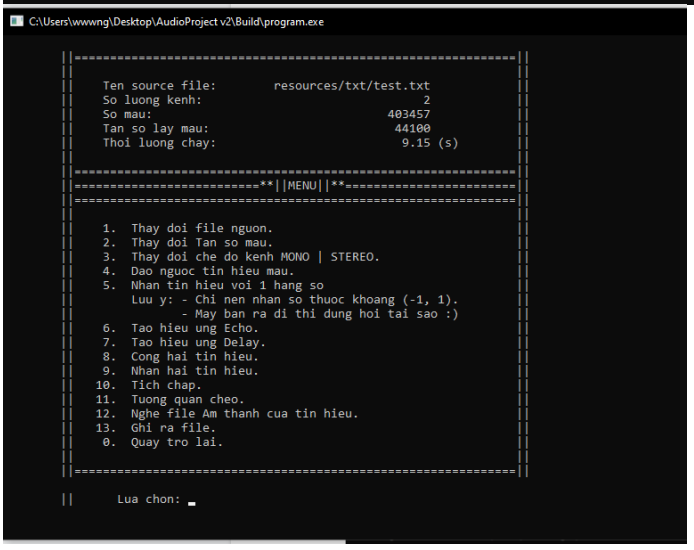
```
// * Tải dữ liệu từ file WAV, TXT vào obj
// * Riêng file TXT chỉ chứa các biến số
...
void loadFromFile(const string &fileName) {
    if (fileName.substr(fileName.size() - 4) == ".txt") {
        std::ifstream file(fileName);
        // Nếu không mở được file thì thông báo rồi kết thúc
        if (!file.is_open()) {
            throw std::runtime_error("Error opening file!");
        }
        double s;
        while (!file.eof()) {
            file >> s;
            _samples.push_back((sf::Int16)s);
        }
        file.close();
        _number_of_channels = 2;
        _sample_rate = 44100;
    } else if (fileName.substr(fileName.size() - 4) == ".wav") {
        sf::SoundBuffer _soundBuffer;
        _soundBuffer.loadFromFile(fileName);
        loadBufferToChannels(_soundBuffer);
    } else {
        throw std::invalid_argument("Can read WAV and TXT file!");
    }
    _file_name = fileName;
}
```

- Minh họa bằng chức năng nhập nguồn file tín hiệu.
- Line 226 - 264 file GUI.cpp

```cpp
void changeSourceFile(Wave &wav) {
    std::string fileName;
    bool check = true;
    while(check) {
        system("cls");
        printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
        printf("\n");
        printf("\t||═══════════════════════════════════════════════════════════════||\n");
        printf("\t||═══════════════**||CHANGE SOURCE FILE||**═══════════════════════||\n");
        printf("\t||═══════════════════════════════════════════════════════════════||\n");
        printf("\n");
        printf("\n\tChuong trinh chi ho tro file .wav va file .txt trong truong hop file txt chi la mang cac mau.");
        printf("\n\tNeu ban khong dung dinh dang txt, neu chuong trinh chay loi dan den hong hoc,");
        printf("\n\tChung toi khong chiu trach nhiem!\n");
        printf("\n");
        printf("\t||    Nhap ten file: ");
        cin >> fileName;
        if(fileName.substr(fileName.length() - 4) == ".txt" ||
           fileName.substr(fileName.length() - 4) == ".wav") {
            try {
                wav.loadFromFile(fileName);
            } catch (const std::exception &e) {
                std::cerr << e.what() << '\n';
            }
            system("pause");
            check = false;
        } else {···
    }
}
```

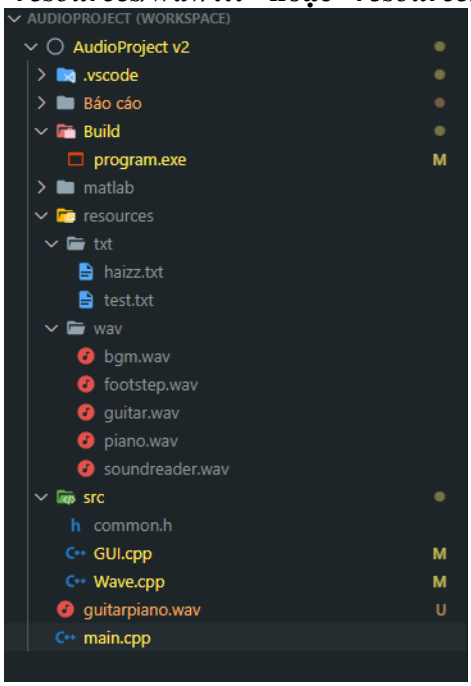
- Minh họa chương trình
    - File Txt: Thời gian load khá mất thời gian, file 1 -> 2 mb có thể load tận 3 - 4p.



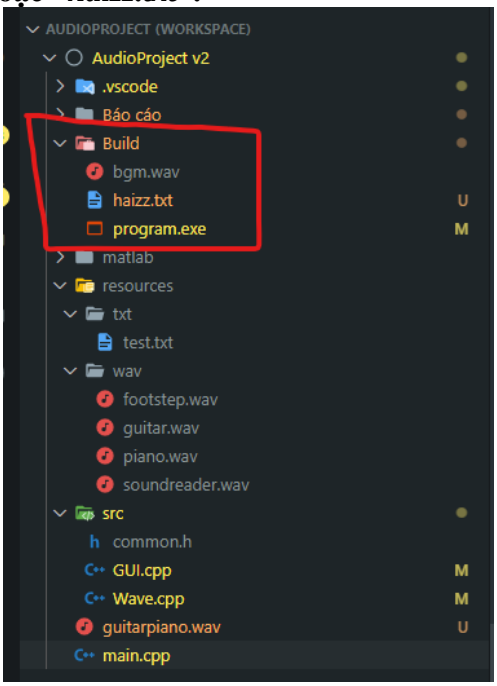


- file wav thì tương tự vậy ta gõ đuôi file .wav , thời gian tải sẽ nhanh hơn.

LƯU Ý: Nếu bạn build từ file main.cpp tức lúc đó bạn phải ghi đầy đủ đường dẫn (cùng cấp với file *main.cpp*) như ví dụ trên với cây thư mục (được biểu thị bằng ảnh dưới) thì phải ghi rõ "*resources/wav/…*" hoặc "*resources/txt/…*".



- **Trong trường hợp chạy file "program.exe" từ thư mục "Build" thì bạn nên ném hẳn file wav hay text vô rồi gõ hẳn tên file wav và text đó mà không cần ghi rõ đường dẫn.**
- **Ví dụ như trường hợp dưới đây thì mình trong chương trình sẽ ghi là "*bgm.wav*" hoặc "*haizz.txt*".**



10. Thay đổi tần số lấy mẫu.

- Phương thức này làm thay đổi tần số lấy mẫu, nếu ta muốn làm tăng nhanh hoặc chậm đi tín hiệu âm thanh.
- Line 170 - 174 file Wave.cpp

```
// * Đặt tần số lấy mẫu.
// * @param Hz    Tần số.
void setSampleRate(const int& Hz) {
    _sample_rate = Hz;
}
```

- Minh họa bằng chức năng thay đổi tần số lấy mẫu.
- Line 268 - 282 file GUI.cpp

```
You, yesterday | 1 author (You)
void changeSampleRate(Wave &wav) {
    int value;
    system("cls");
    printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
    printf("\n");
    printf("\t||═══════════════**||CHANGE SAMPLE RATE||**═══════════════||\n");
    printf("\t||═══════════════════════════════════════════════════════||\n");
    printf("\n");
    printf("\t||    Nhap tan so: ");
    cin >> value;
    wav.setSampleRate(value);
    printf("\n\t||    Tan so lay mau hien tai: %8d Hz.\n", wav.getSampleRate());
    system("pause");
    return;
}
```

11. Nghe file âm thanh

- Phương thức play để nghe thử tín hiệu âm thanh, sử dụng script của thư viện SFML/Audio để ghi các mẫu vào một biến Âm thanh
- Line 183 - 192 file Wave.cpp

```
// * Nghe file wav.
...
void play() {
    sf::SoundBuffer sb;
    sb.loadFromSamples(&_samples[0], _samples.size(), 1, _sample_rate);
    sf::Sound sound;
    sound.setBuffer(sb);
    sound.play();
    // * Phải cho chương trình ngủ thì biến sound mới tồn tại
    // * nếu không khi đó tức chương trình con của hàm kết thúc
    // * đồng nghĩa việc biến sound bị hủy khi chưa phát được hết âm thanh
    sleep(getDurationAsSeconds());
}
```

- Chức năng nghe file âm thanh của chương trình
- Line 316 - 336 file GUI.cpp

```
void listenWave(Wave &wav) {
    bool check = true;
    char c;
    while (check) {
        system("cls");
        printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
        printf("\n");
        getWaveInfo(wav);
        printf("\tPlaying ...\n");
        wav.play();
        printf("\tDa ket thuc!\n");
        printf("\tBan co muon nghe tiep khong? (Y/y = YES | N/n (other) = NO):");
        cin >> c;
        if (c == 'Y' || c == 'y') {
            check = true;
        } else {
            check = false;
        }
    }
}
```

12. Viết ra file

- Tương tự đọc file, hiện chưa phát triển ghi mẫu ra file text, nhóm mới chỉ sử dụng script của thư viện SFML để ghi ra file .wav
- Line 258 - 262 file Wave.cpp

```
void saveToFile(const string &fileName) const {
    sf::SoundBuffer sb;
    sb.loadFromSamples(&_samples[0], _samples.size(), _number_of_channels, _sample_rate);
    sb.saveToFile(fileName);
}
```

- Chức năng ghi tín hiệu ra file.
- Line 512 - 538 file GUI.cpp

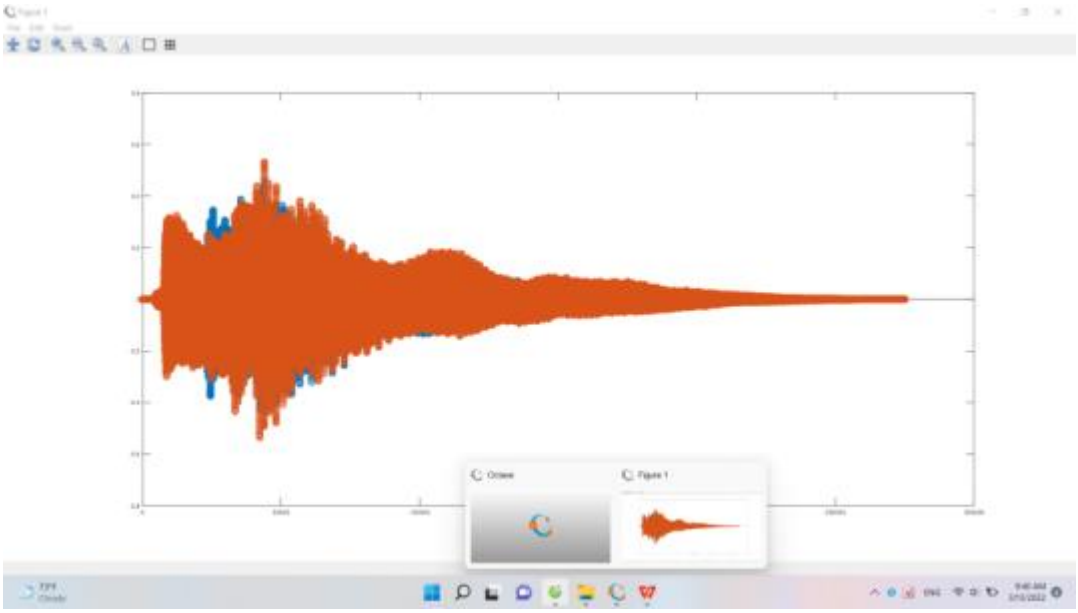
```
void writeToFile(Wave &wav) {
    system("cls");
    printf("\tChuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0\n");
    printf("\n");
    printf("\t||═══════════════════════════════════════════════════════════════════════════||\n");
    printf("\t||═══════════════════**||WRITE TO FILE||**═══════════════════════════════════||\n");
    printf("\t||═══════════════════════════════════════════════════════════════════════════||\n");
    printf("\n");
    std::string name;
    printf("\n\t||\tTen file (Khong can ghi duoi file, luu duoi dang wav): ");
    cin >> name;
    name.append(".wav");
    printf("\n\t||\tWriting %s ...\n", name.c_str());
    wav.saveToFile(name);
    printf("\n\t||\tGhi file hoan tat\n");
    printf("\n\t||\t");
    system("pause");
}
```

```cpp
224
225     // * Tải dữ liệu từ file WAV, TXT vào obj
226     // * Riêng file TXT chỉ chứa các biến số
        You, 2 hours ago | 1 author (You)
227     void loadFromFile(const string &fileName) {
228         if (fileName.substr(fileName.size() - 4) == ".txt") {
229             std::ifstream file(fileName);
230             // Nếu không mở được file thì thông báo rồi kết thúc
231             if (!file.is_open()) {
232                 throw std::runtime_error("Error opening file!");
233             }
234             double s;
235             while (!file.eof()) {
236                 file >> s;
237                 _samples.push_back((sf::Int16)s);
238             }
239             file.close();
240             _number_of_channels = 2;
241             _sample_rate = 44100;
242         } else if (fileName.substr(fileName.size() - 4) == ".wav") {
243             sf::SoundBuffer _soundBuffer;
244             _soundBuffer.loadFromFile(fileName);
245             loadBufferToChannels(_soundBuffer);
246         } else {
247             throw std::invalid_argument("Can read WAV and TXT file!");
248         }
249         _file_name = fileName;
250     }
251             You, March 15th, 2022 6:14pm • None
```

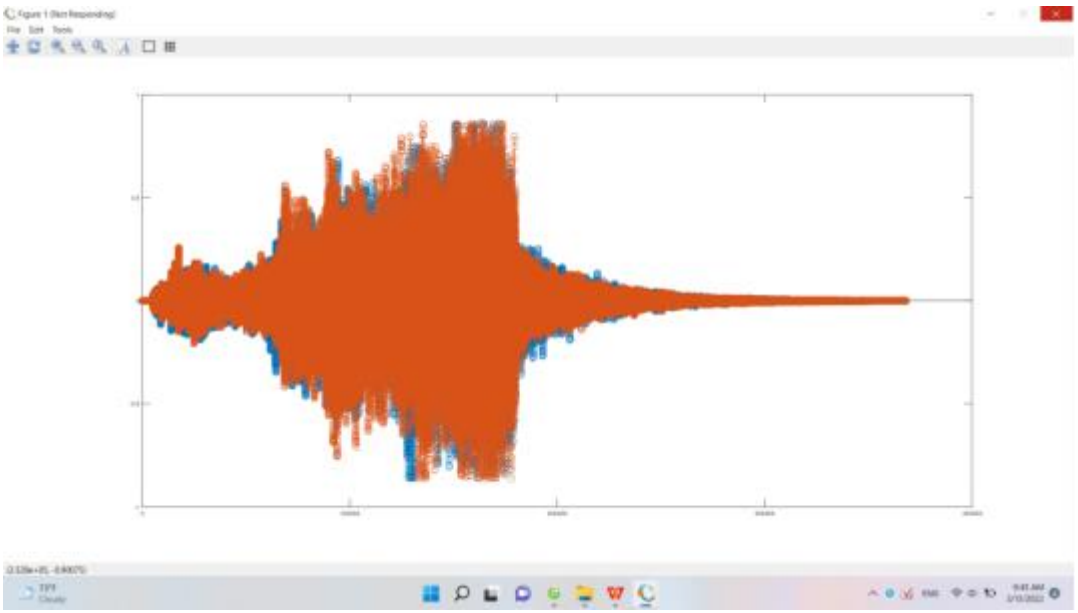
```
C:\Users\wwwng\Desktop\AudioProject v2\Build\program.exe
Chuong trinh Dieu chinh Am thanh (Sound Equalizer Program) v1.0

||============================================================||
||==================**||CHANGE SOURCE FILE||**==================||
||============================================================||


Chuong trinh chi ho tro file .wav va file .txt trong truong hop file txt chi la mang cac mau.
Neu ban khong dung dinh dang txt, neu chuong trinh chay loi dan den hong hoc,
Chung toi khong chiu trach nhiem!

||    Nhap ten file:
```

III. Giải thích và minh họa các kết quả

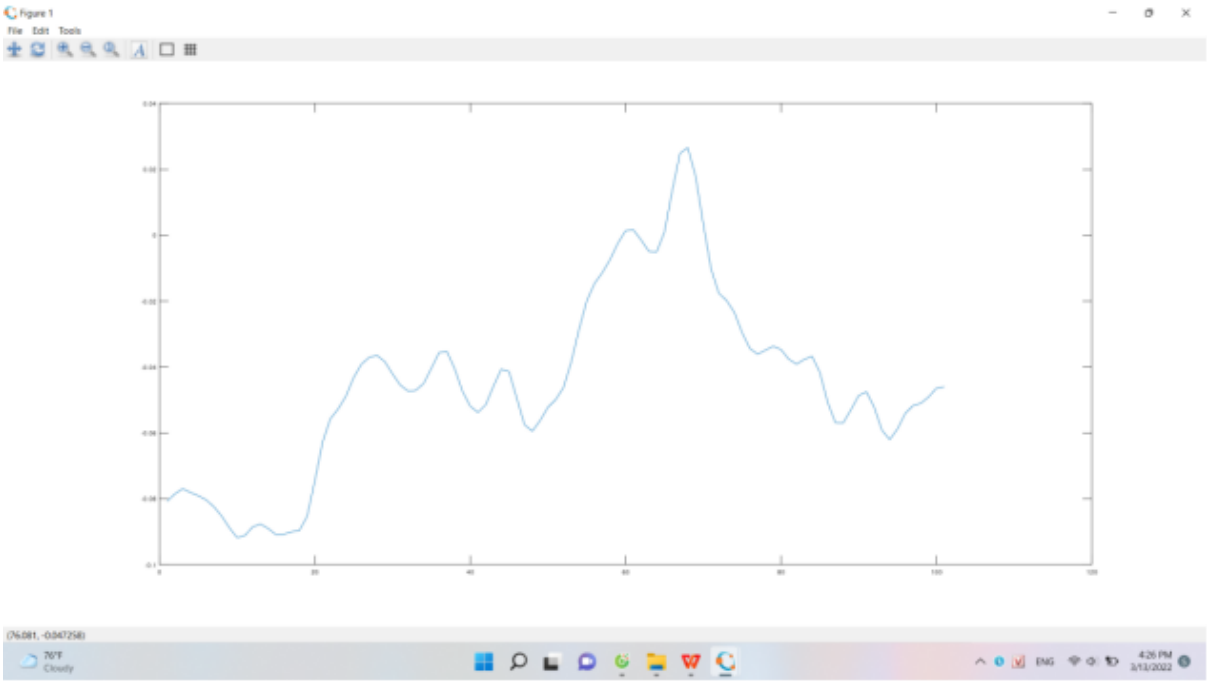
- Minh họa tín hiệu piano :



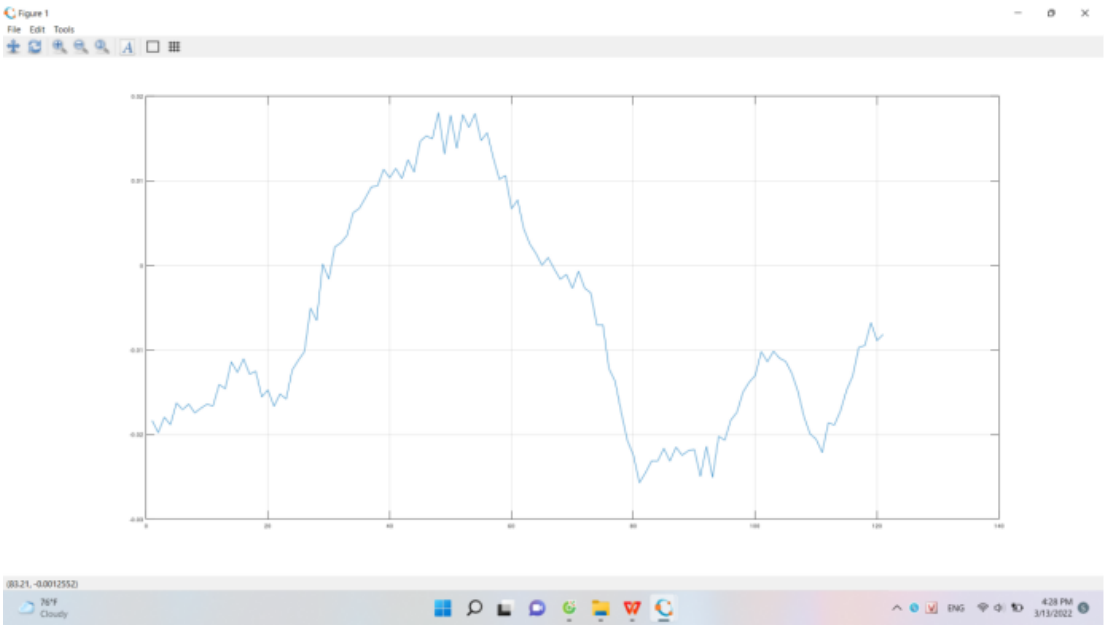
- Minh họa tín hiệu guitar :



- Lấy 1 đoạn tín hiệu trong tín hiệu piano



- Lấy 1 đoạn tín hiệu trong tín hiệu guitar



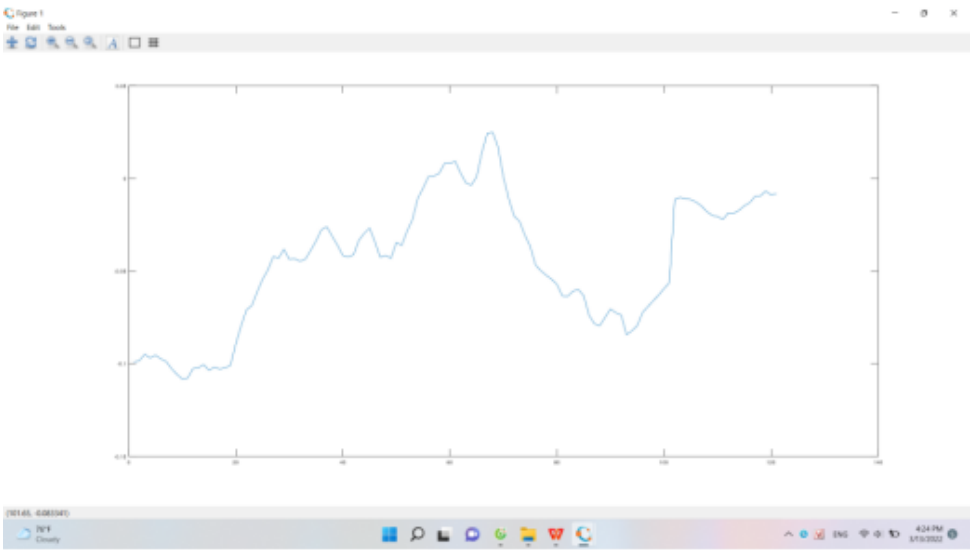
*** Các hiệu ứng với tín hiệu:**

1. Phép cộng 2 tín hiệu (Hiệu ứng trộn):

So sánh độ dài hai tín hiệu: Nếu tín hiệu nào ngắn hơn thì chèn thêm các tín hiệu '0' vào

```
x1len=length(x1);
y1len=length(y1);
if(x1len<y1len)
x1=[x1 zeros(1,y1len-x1len)];
endif;

if(x1len>y1len)
y1=[y1; zeros(1,x1len-y1len)];
endif;
z= x1+y1;
plot(z);
```

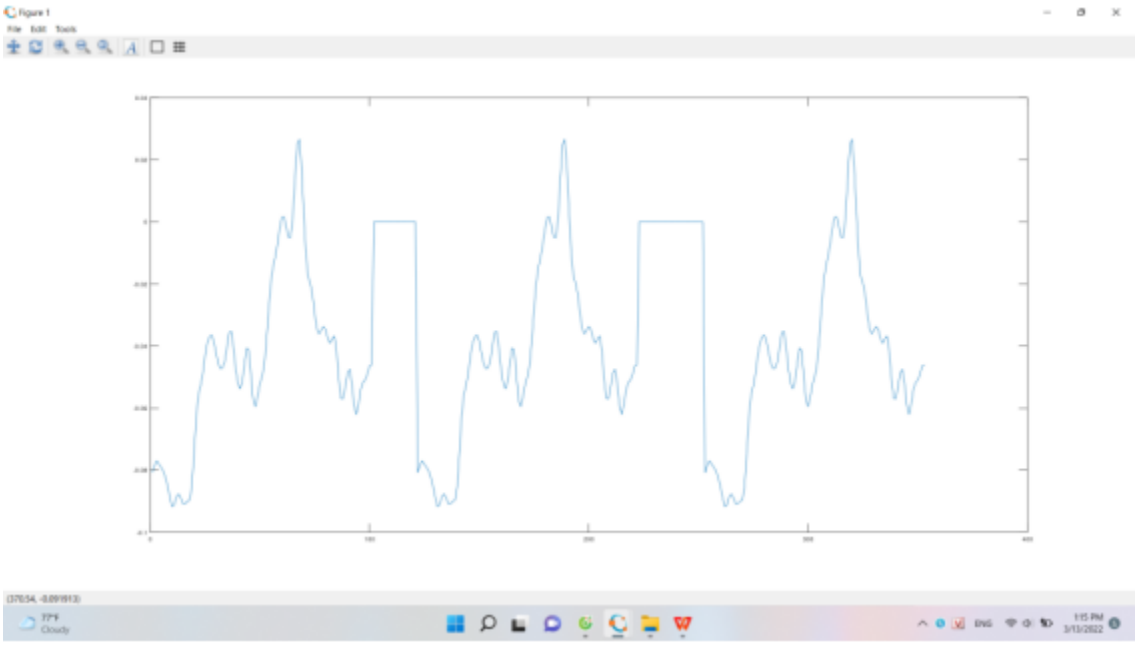
Cộng hai tín hiệu có độ dài bằng nhau: z = x1+y1



2. Hiệu ứng echo:

Tạo hiệu ứng echo cho tín hiệu x1 bằng cách lặp lại tín hiệu x1 và giữa các khoảng lặp đó chèn thêm các tín hiệu '0'

```
x1=[ x1 zeros(1,20) x1 zeros(1, 30) x1 ];
```



3.Nhân hai tín hiệu

So sánh độ dài hai tín hiệu: Nếu tín hiệu nào ngắn hơn thì chèn thêm các tín hiệu ‘0’ vào

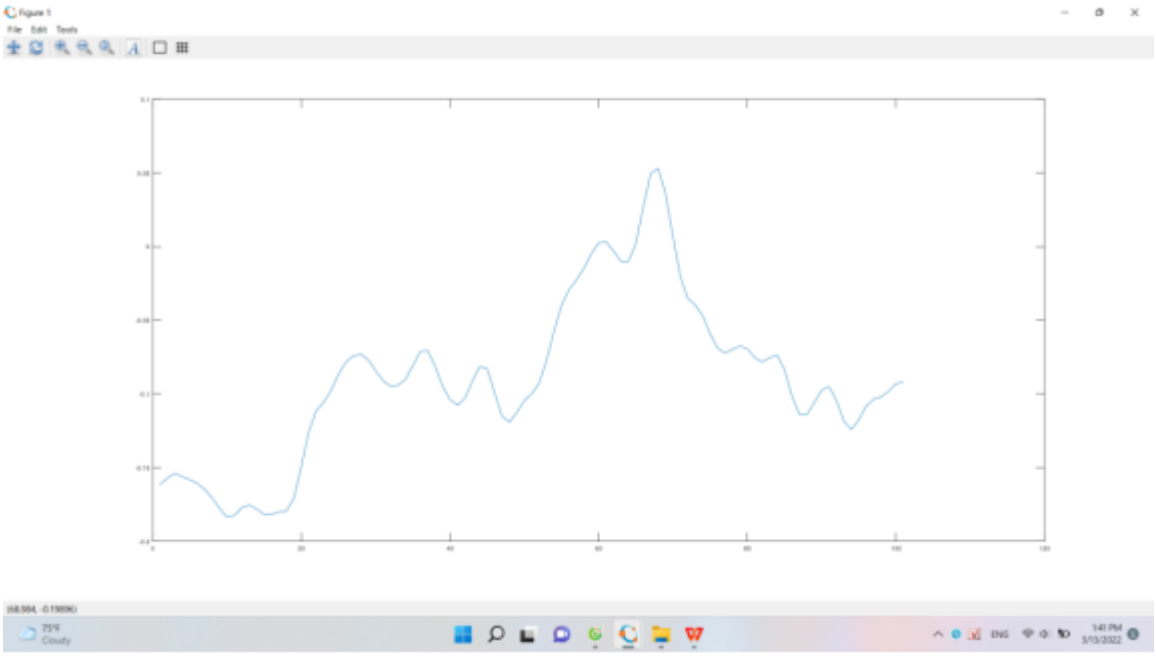
```
x1len=length(x1);
y1len=length(y1);
if(x1len<y1len)
x1=[x1 zeros(1,y1len-x1len)];
endif;

if(x1len>y1len)
y1=[y1; zeros(1,x1len-y1len)];
endif;

for i=1:length(y1)
  z(i)= x1(i)*y1(i);
endfor;
```

Nhân hai tín hiệu có độ dài bằng nhau: (nhân từng giá trị tương ứng): z= x1*y1

4.Nhân 1 tín hiệu với hằng số (điều chỉnh độ lớn/ nhỏ âm lượng

Nhân tín hiệu x1 với 2 (nhân từng giá trị của x1 với 2) : Âm lượng của x1 tăng lên gấp 2 lần : x1 (n)= 2*x1 (n)

```
x1=x1*2;
```

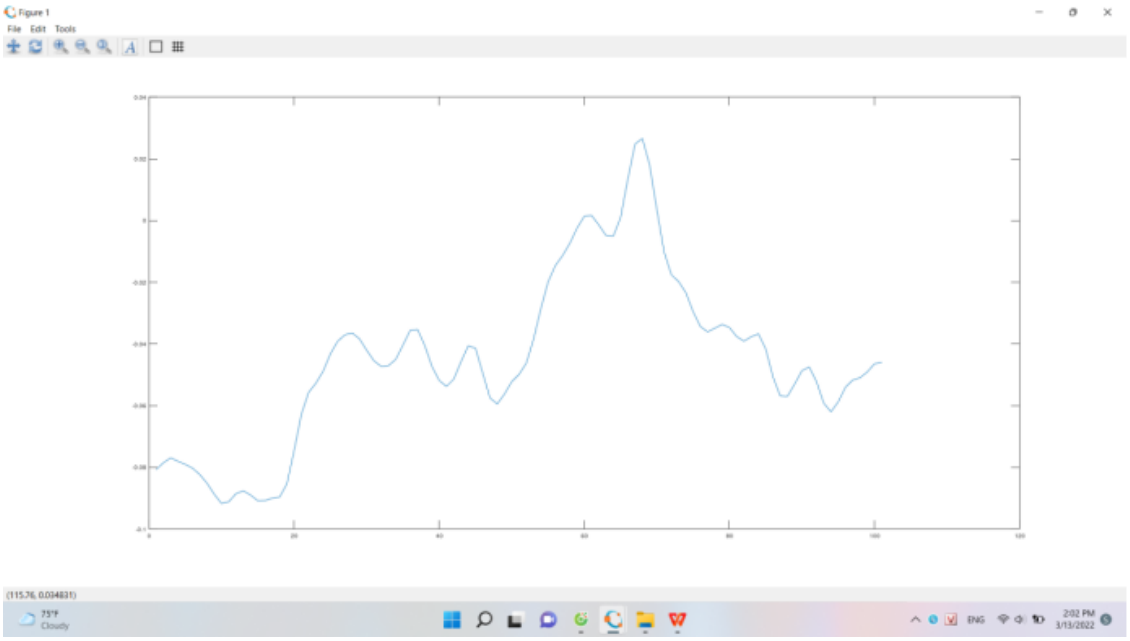


5.Phép đảo thời gian

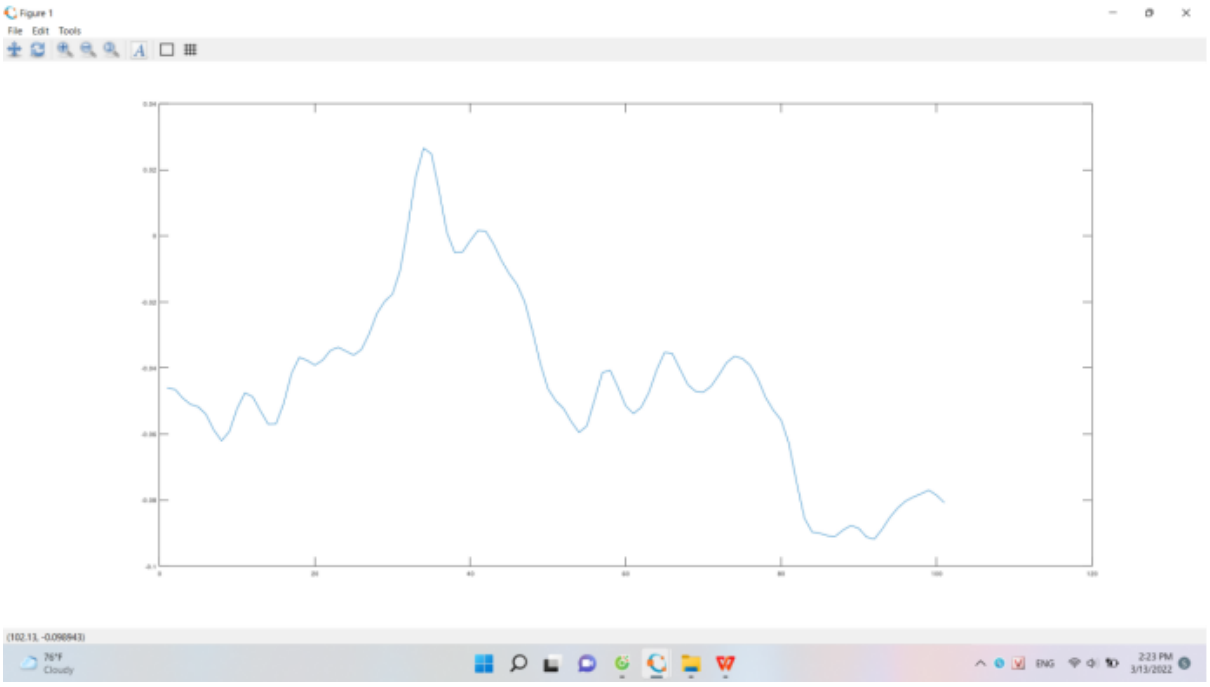
Đảo tín hiệu x1 qua trục tung : z(n)= x1(-n)

```
len=length(x1);
z  ( len ,1);
for i=1:len
   z(len-i+1)=x1(i);
endfor;
```

Tín hiệu x1 ban đầu:



Tín hiệu x1 khi bị đảo ngược :



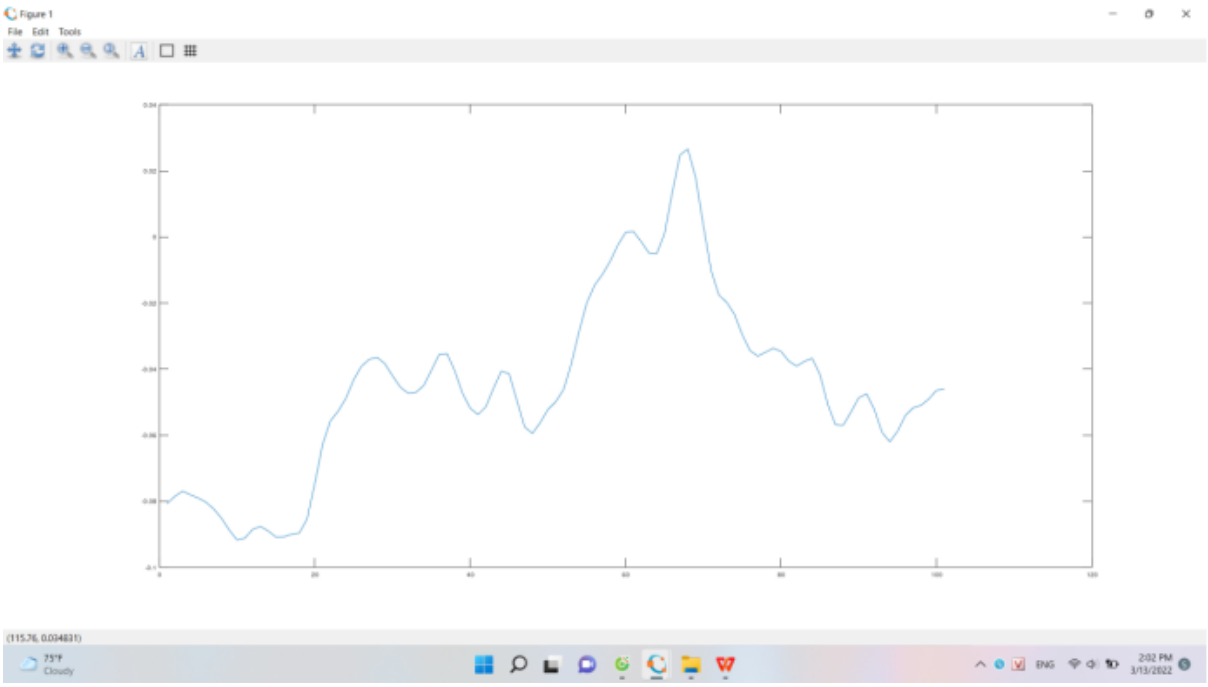
6. Phép trễ tín hiệu:

Dịch chuyển tín hiệu x1 sang phải 20 đơn vị thời gian (tín hiệu x1 bị chậm đi 20 đơn vị thời gian)

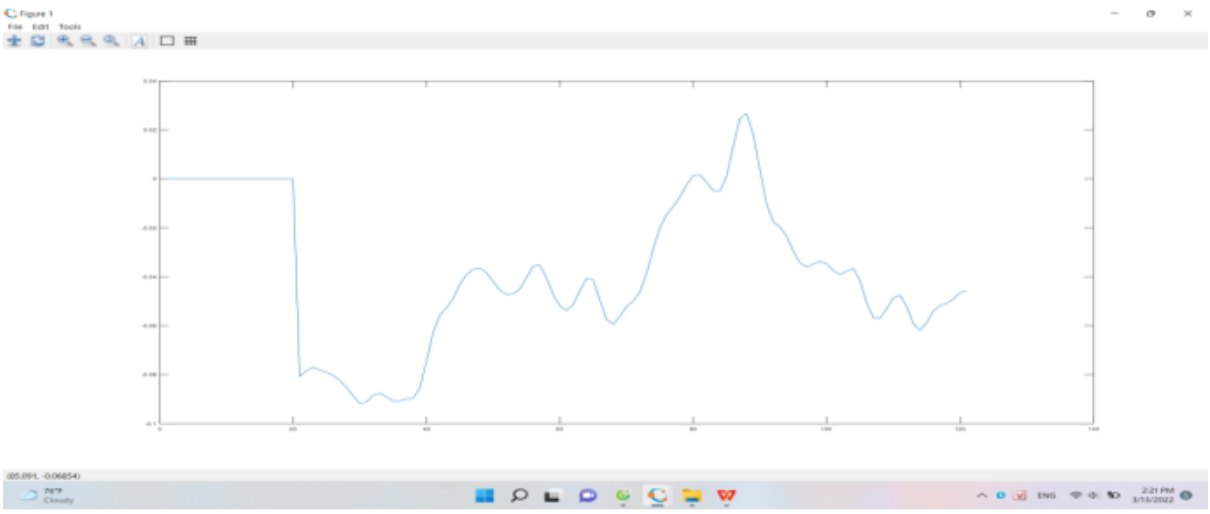
: Chèn các giá trị ‘0’ vào trước tín hiệu x1 : x1(n)= x1(n-20)

```
x1= [ zeros(1,20) x1    ];
```

Tín hiệu x1 ban đầu:



Tín hiệu x1 khi bị trễ 20 đơn vị thời gian:



7. Tích chập

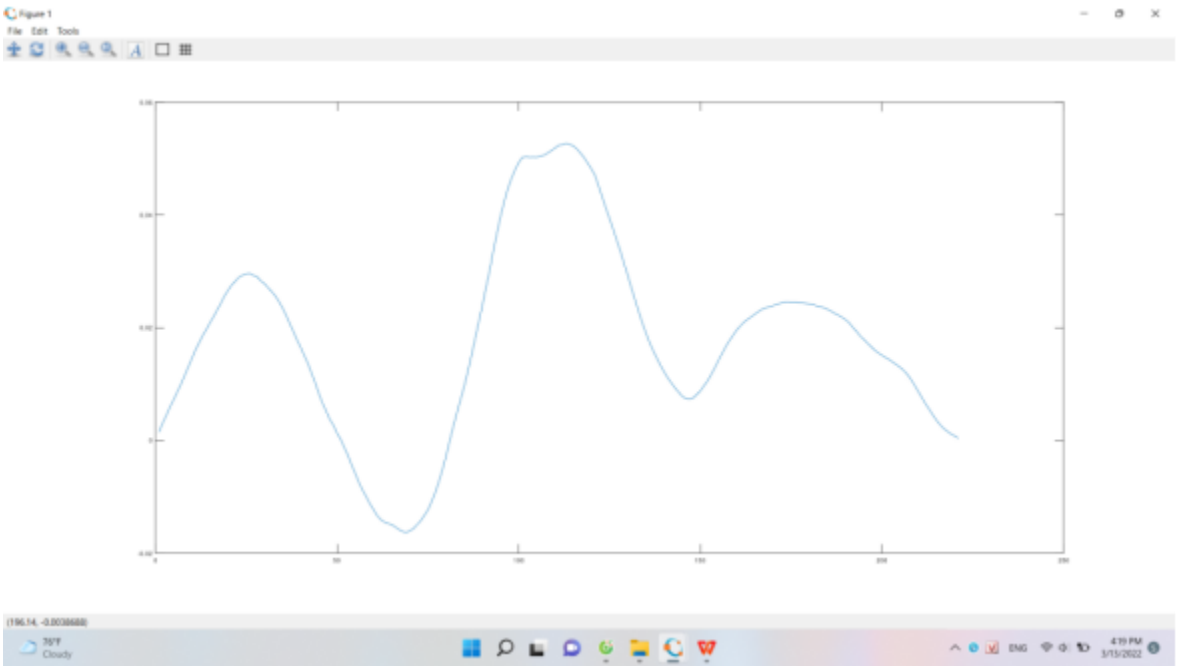
Tính tích chập bằng cách tạo bảng:

```
  x1len=length(x1);
y1len=length(y1);



for i=1:x1len
  for j= 1:y1len
    z(i,j) = x1(i)*y1(j);
  endfor;
 endfor;

 a= zeros(1,x1len+y1len-1);

 for k= 1: x1len+y1len-1
   for i=1:x1len
      for j= 1:y1len
          if(i+j-1==k)
              a(k)= a(k)+z(i,j);
          endif;
       endfor;
   endfor;
 endfor;
plot(a)
```

-Tạo ra ma trận z từ 2 tín hiệu x1 và y1

- Tính tổng các giá trị trên các đường chéo song song của ma trận

- Các giá trị đó tạo thành vector a: a là tích chập của 2 tín hiệu x1 và y1



8. Tương quan chéo

Tính tương quan chéo bằng cách tạo bảng:

```
  x1len=length(x1);
y1len=length(y1);


c= ( x1len ,1);
for i=1:x1len
  c(x1len-i+1)=x1(i);
endfor;
x1=c;

for i=1:x1len
  for j= 1:y1len
    z(i,j) = x1(i)*y1(j);
  endfor;
 endfor;

 a= zeros(1,x1len+y1len-1);

 for k= 1: x1len+y1len-1
   for i=1:x1len
      for j= 1:y1len
          if(i+j-1==k)
              a(k)= a(k)+z(i,j);
          endif;
       endfor;
   endfor;
 endfor;
```

- Đảo ngược x1 hoặc y1 trước khi tính

-Tạo ra ma trận z từ 2 tín hiệu x1 và y1

- Tính tổng các giá trị trên các đường chéo song song

- Các giá trị đó tạo thành vector a: a là tích chập của 2 tín hiệu x1 và y1