Tin học cơ sở 4

Buổi 8. Con trỏ

Bộ môn Khoa học máy tính - 2017

Nội dung buổi học

1. Con trỏ = địa chỉ biến
2. Tệp nhị phân
3. Dự án nhỏ: đọc tệp audio WAV

Biến và bộ nhớ

- Khi khai báo biến
  - Tên biến được gắn với một vùng nhớ chứa giá trị
- Khi sử dụng biến
  - Máy tìm địa chỉ của biến trong bộ nhớ
  - Đi tới địa chỉ đó và lấy giá trị của biến

Biến và bộ nhớ

- Toán tử &
  - Lấy địa chỉ của biến

```
cout << (&x) << endl; // địa chỉ
```

- Toán tử *
  - Lấy giá trị tại một địa chỉ

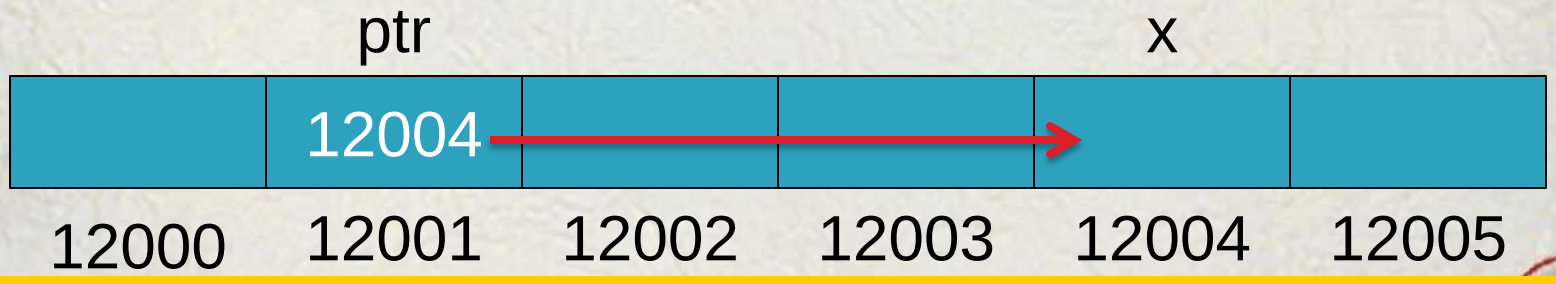
```
cout << *(&x) << endl;  // x
```

Con trỏ

- Lập trình hiệu năng cao (*high performance*)
  - Truyền địa chỉ biến (thay cho giá trị)
  - Xử lý cấu trúc dữ liệu phức tạp
    - ✓Ví dụ: dữ liệu ở các vùng nhớ khác nhau
  - Hàm không cần biết trước kiểu dữ liệu
    - ✓Chỉ cần biết địa chỉ của dữ liệu

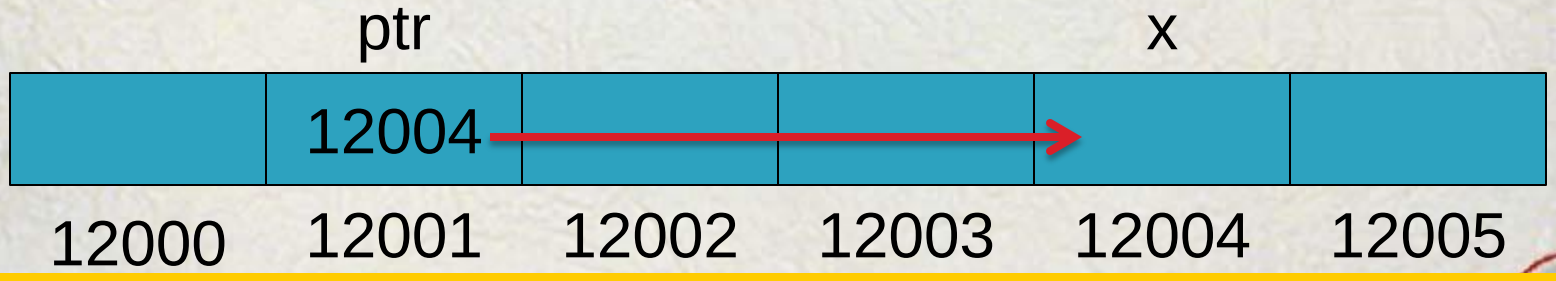
Con trỏ

- Bản chất của biến con trỏ
  - Giá trị là một số nguyên
  - Là địa chỉ bộ nhớ
  - Thường là địa chỉ của biến khác
    - ✓*Có thể là địa chỉ của một biến con trỏ khác*



Con trỏ

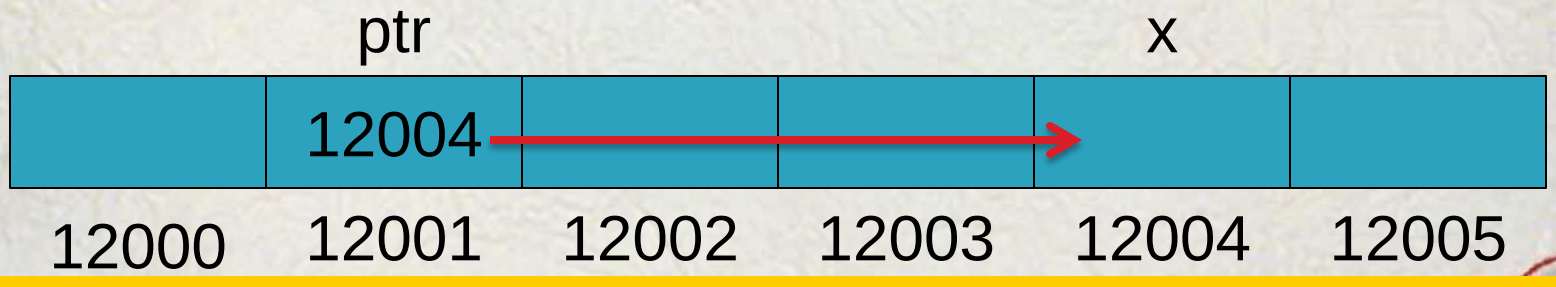
```
&x == ptr
*ptr == x
&(*ptr) == ptr
*(&x) == x
```



Khai báo con trỏ

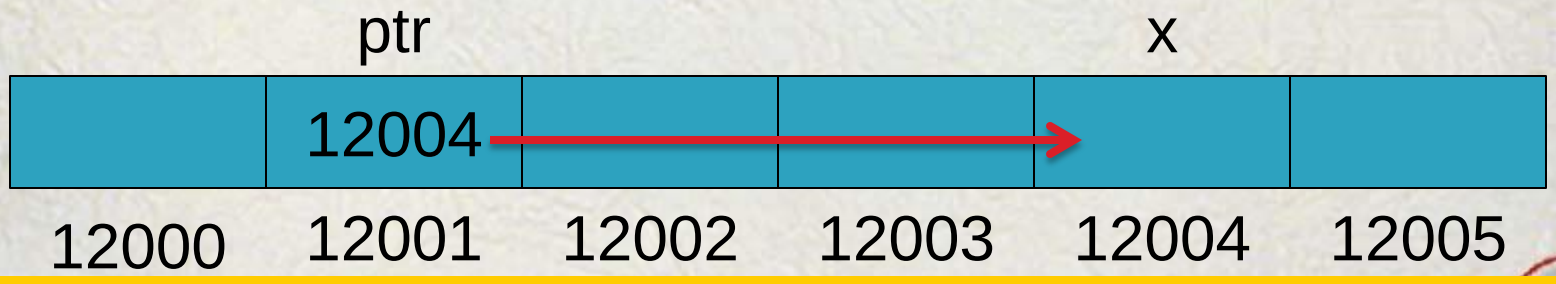
Cú pháp

```
<kiểu giá trị> *<tên con trỏ>;
<kiểu giá trị> *<tên con trỏ>
    = <địa chỉ>;
```



Khai báo con trỏ

```
int x;
int *ptr = &x;
string line;
string *pline = &line;
```



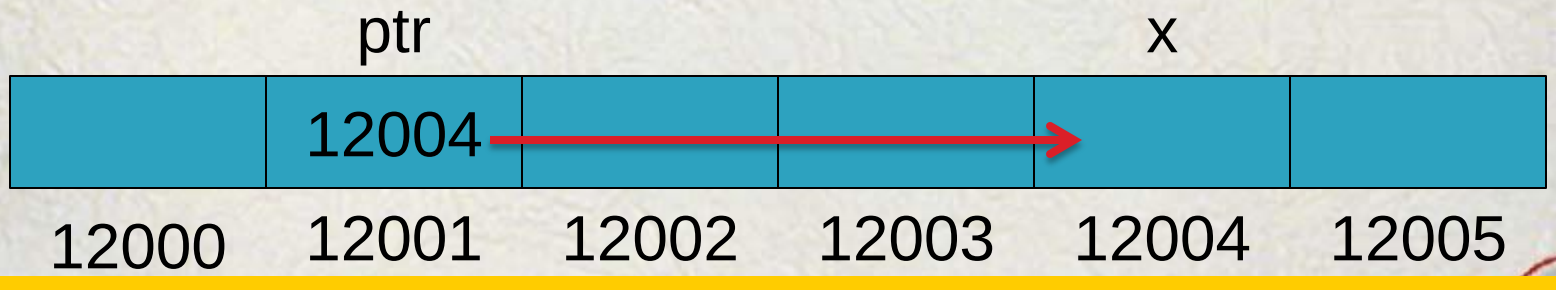
Sử dụng con trỏ

- Đọc giá trị tại địa chỉ

```
cout << *ptr << endl;
```

- Sửa giá trị tại địa chỉ

```
*ptr = 5; // x == 5
```



Truyền tham số bằng địa chỉ

```
void squareValue(int x) {
   x = x * x;
}

void squareByPtr(int *ptr) {
   *ptr = *ptr * *ptr;
}

int main() {
   int x = 5;
   squareValue(x);
   cout << x << endl; // 5
   squareByPtr(&x);
   cout << x << endl; // 25
}
```

Không làm thay đổi giá trị truyền vào hàm

Thay đổi giá trị do con trỏ chỉ tới

```
void swapByValue(int a, int b) {
   int tmp = a;
   a = b;
   b = tmp;
}

void swapByPtr(int *a, int *b) {
   int tmp = *a;
   *a = *b;
   *b = tmp;
}

int main() {
   int x = 5, y = 6;
   swapByValue(x, y);
   cout << x << " " << y << endl; // 5 6
   swapByPtr(&x, &y);
   cout << x << " " << y << endl; // 6 5
}
```

Không làm thay đổi giá trị truyền vào hàm

Thay đổi giá trị do con trỏ chỉ tới

Từ khóa const với con trỏ

```
const int *ptr = &x;
```

Không được sửa giá trị ptr trỏ đến (*ptr)

```
int * const ptr = &x;
```

Được sửa *ptr nhưng không được sửa ptr

```
const int* const ptr = &x;
```

Không được sửa cả *ptr và ptr

Con trỏ và mảng

- Tên mảng tĩnh là con trỏ đến phần tử đầu tiên của mảng

```
string str[10];
string *p = str;
cout << *p << endl; // s[0]
```

- Truyền mảng bản chất là truyền con trỏ

Các phép toán trên con trỏ

```
string str[10];
string *p = str;
p++; // p → str[1]
p += 2; // p → str[3]
p--; // p → str[2]
cout << (p – str) << endl; // 2
cout << *(p + 3) << endl; // str[5]
cout << p[0] << endl; // str[2]
cout << p[2] << endl; // str[4]
```

Con trỏ đến con trỏ

- Khai báo

```
int x;
int *ptr = &x;
int **pptr = &ptr;
cout << *pptr; // địa chỉ của x
cout << **pptr; // giá trị của x
```

Tệp nhị phân

- Là tệp KHÔNG phải tệp văn bản
- Nội dung là một chuỗi byte (8 bit)
- Công dụng:
  - Chứa ảnh, video âm thanh, dữ liệu nén
  - Lưu nội dung một vùng bộ nhớ
  - … hoặc bất kì nội dung nào
- Xem/sửa nội dung: Hex Editors (Frhed)

Lưu bộ nhớ xuống tệp nhị phân

- Hàm write của ofstream

```
ostream& write (const char* s,
  streamsize n);
```

  - s: địa chỉ bộ nhớ cần lưu
  - n: kích thước (tính bằng byte)

Lưu bộ nhớ xuống tệp nhị phân

```
int num[100];
ofstream outFile("number.bin", ios::binary);

outFile.write( (char*) num, 100*sizeof(int) );
```

Chuyển kiểu con trỏ int* sang char*

Tổng số byte của vùng nhớ cần lưu

Đọc tệp nhị phân lên bộ nhớ

- Hàm read của ifstream

```
istream& read(char* s,
  streamsize n);
```

  - s: địa chỉ bộ nhớ để chứa dữ liệu
  - n: kích thước (tính bằng byte)

- Số byte thực sự đọc được – hàm `gcount()`

Đọc tệp nhị phân lên bộ nhớ

```
int numCopy[100];
ifstream inFile("number.bin", ios::binary);

inFile.read( (char*) numCopy, 100*sizeof(int) );
```

Chuyển kiểu con trỏ int* sang char*

Tổng số byte của vùng nhớ cần đọc

Sao chép file

```
const int SIZE = 1 << 16;
char buffer[SIZE];
ifstream inFile("number.bin", ios::binary);
ofstream outFile("number.copy", ios::binary);

do {
  inFile.read(buffer, SIZE);
  int realReadBytes = inFile.gcount();
  outFile.write(buffer, realReadBytes);
} while (inFile);
```

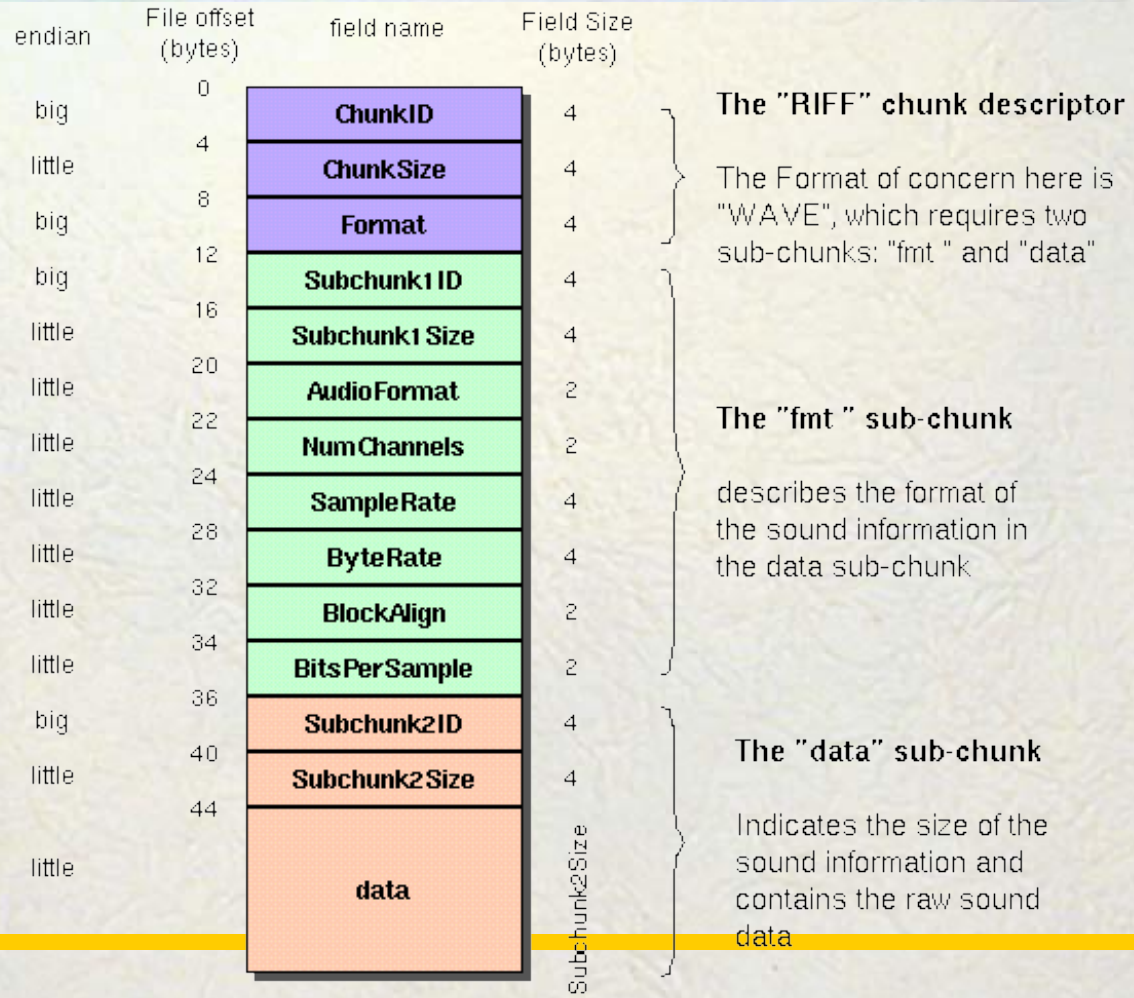
Khai báo bộ đệm để lưu tạm dữ liệu

Lấy số lượng byte thực sự đọc được

Dự án nhỏ: Đọc tệp .WAV

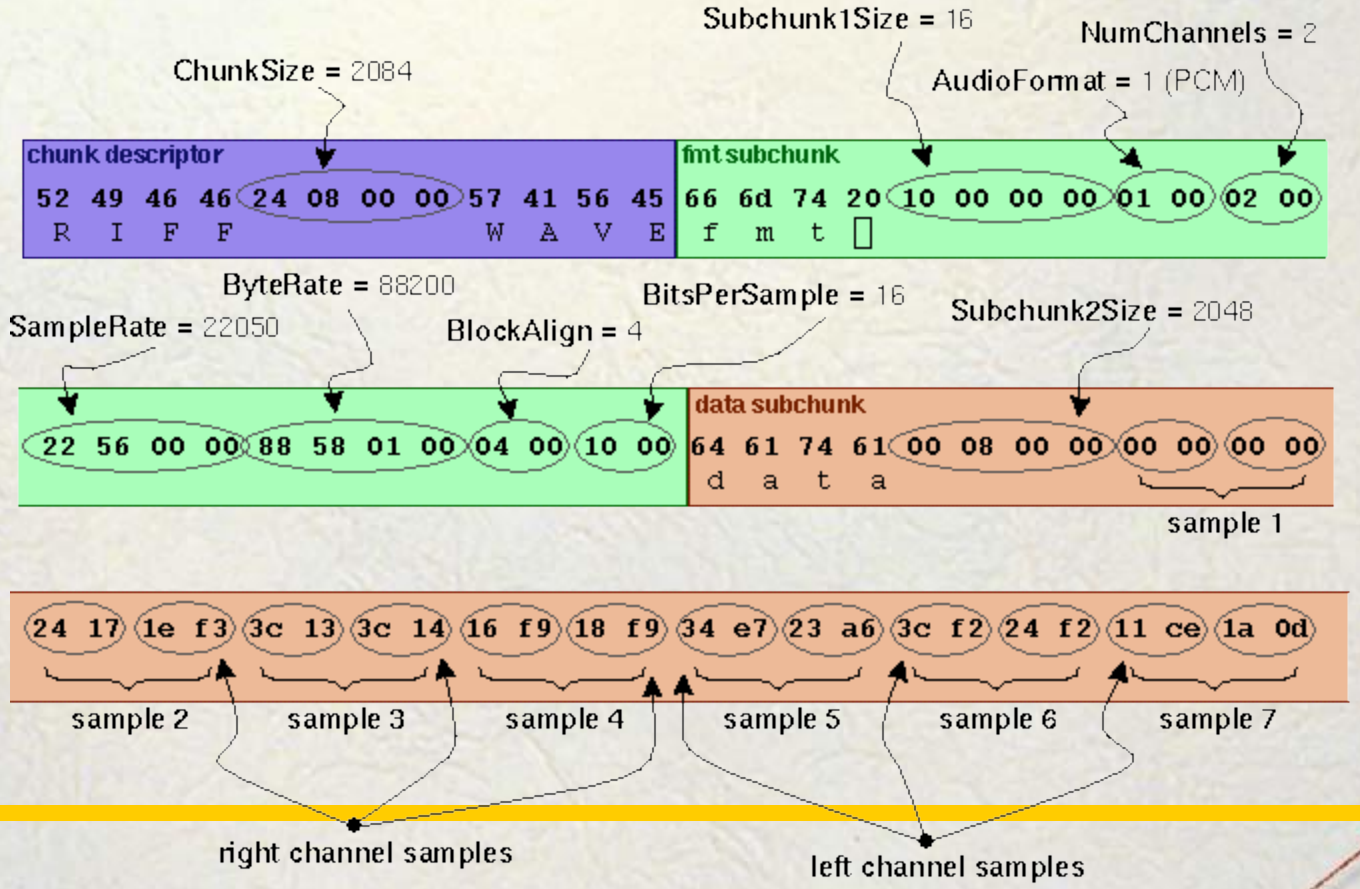
- Tệp .WAV: lưu dữ liệu âm thanh
  - Trường hợp không nén dữ liệu
  - Định dạng gồm header + data
  - http://soundfile.sapp.org/doc/WaveFormat/
  - Sử dụng Frhed để xem file .WAV

Định dạng .WAV



Định dạng .WAV

```
52 49 46 46 24 08 00 00 57 41 56 45 66 6d 74 20 10 00 00 00 01 00 02 00
22 56 00 00 88 58 01 00 04 00 10 00 64 61 74 61 00 08 00 00 00 00 00 00
24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d
```



Mở file WAV

```
ifstream wavFile("Alarm01.wav", ios::binary);
if (!wavFile) {
  cout << "No file exists" << endl;
  return 0;
}
```

Đọc 4 byte đầu RIFF

```
char chunkID[5] = {0,0,0,0,0};
wavFile.read(chunkID, 4);
cout << chunkID << endl;
```

Đọc 4 byte chỉ kích thước tệp

```
unsigned long chunkSize;
wavFile.read((char*)&chunkSize, 4);
cout << "chunkSize = " << chunkSize << endl;
```

- Tạo hàm đọc số nguyên 4 byte

```
unsigned long readLong(ifstream* inFile) {
    unsigned long result;
    inFile->read((char*)&result, 4);
    return result;
}
```

```
unsigned long chunkSize = readLong(&wavFile);
cout << "chunkSize = " << chunkSize << endl;
```

Đọc các xâu WAVE và fmt

```
char format[5] = {0,0,0,0,0};
wavFile.read(format, 4);
cout << format << endl;          // WAVE

char subChunk1ID[5] = {0,0,0,0,0};
wavFile.read(subChunk1ID, 4);
cout << subChunk1ID << endl;   // fmt
```

Đọc các số còn lại của header

```
unsigned short readLong(ifstream* inFile) {
    unsigned short result;
    inFile->read((char*)&result, 2);
    return result;
}
```

```
unsigned long subChunk1Size = readLong(&wavFile);
cout << "numChannels = " << numChannels << endl;

unsigned long sampleRate = readLong(&wavFile);
cout << "sampleRate = " << sampleRate << endl;

unsigned long byteRate = readLong(&wavFile);
cout << "byteRate = " << byteRate << endl;

unsigned short blockSize = readShort(&wavFile);
cout << "blockSize = " << blockSize << endl;

unsigned short bitPerSample = readShort(&wavFile);
cout << "bitPerSample = " << bitPerSample << endl;
```

Tạo hàm đọc số nguyên 2 byte

Đọc phần dữ liệu

```
char subChunk2ID[5] = {0,0,0,0,0};
wavFile.read(subChunk2ID, 4);
cout << subChunk2ID << endl;  // data

unsigned long subChunk2Size = readLong(&wavFile);
cout << "subChunk2Size = " << subChunk2Size << endl;

unsigned long n_sample =
    subChunk2Size * 8 / bitPerSample;
short data[n_sample];
wavFile.read((char*)data, n_sample);
cout << "gcount() = " << wavFile.gcount() << endl;
```

Vẽ sóng âm

- Thông số kỹ thuật
  - numChannels = 2 kênh (stereo)
  - blockSize = 4 (4 byte cho 2 kênh)
    → mỗi kênh (trái, phải) có 2 byte = short
    → trong mảng data, short thứ tự chẵn là kênh âm bên trái (left), short thứ tự lẻ là kênh âm bên phải (right)

Vẽ sóng âm

- Mỗi mẫu (sample) trong mảng data
  - Là 1 số nguyên từ -2^15 đến 2^15-1
  - Cần hàm vẽ 1 mẫu: tính độ lớn cần hiển thị

```
string getSignalString(int signal, int absMax, int maxLen) {
    int len = (double) (signal > 0 ? signal : -signal)
                / absMax * maxLen;
    if (signal > 0) {
        return string(maxLen, ' ') + "-"
               + string(len, '*') + string(maxLen-len, ' ');
    } else {
        return string(maxLen-len, ' ') + string(len, '*')
               + "-" + string(maxLen, ' ');
    }
}
```

Vẽ sóng âm

```
int absMax = 0;
for (int i = 0; i < n_sample; i++) {
  int abs = data[i] > 0 ? data[i] : -data[i];
  if (absMax < abs) absMax = abs;
}
cout << "absMax = " << absMax << endl;

for (int i = 0; i < n_sample; i+=2) {
  string left  = getSignalString(data[i], absMax, 8);
  string right = getSignalString(data[i+1], absMax, 8);
  cout << left << "   " << right << endl;
}
```