

# Assessment

9 - Lớp = Dữ liệu + Hàm; Sắp xếp

<https://github.com/tqlong/advprogram>

# Nội dung

- Lớp (Class) = Dữ liệu + Hàm
  - Đóng gói mô-đun Guesser
- Đánh giá SimpleAI (assessment)
  - Máy chơi với máy
  - Đóng gói mô-đun Assessment
- Kỹ thuật
  - `class`
  - Hàm khởi tạo, danh sách khởi tạo
  - Hàm `public`, hàm `private`, hàm `const`
  - Sắp xếp với `sort`

# Đặt vấn đề: đánh giá SimpleAI

## Một số câu hỏi cho SimpleAI

- Các *tập từ vựng khác nhau* có cho kết quả khác nhau ?  
Nên chọn tập từ vựng nào ?
- Các thay đổi trong *thuật toán đoán kí tự* có thực sự giúp việc đoán từ chính xác hơn ?

## Cần có đánh giá định lượng (số hoá)

- Giúp trả lời rõ các câu hỏi trên
- Lựa chọn chương trình đoán từ chính xác hơn.

# Đặt vấn đề: đánh giá SimpleAI

Cần suy nghĩ về

- Cách đánh giá SimpleAI
  - Cách tính điểm
- Cách tổ chức chương trình
  - Đánh giá tự động trên tập từ vựng bất kỳ
  - Cho phép máy tự động chơi nhiều lần và ghi lại kết quả chơi (từ cần đoán, số lần đoán, số chữ cái đoán đúng ...)

# Chung và riêng

SimpleAI hiện có các mô-đun

- Giao diện, Util, Draw
- Guesser
  - Chương trình chính chỉ cần biết khai báo của **getNextGuess()** → **public**
  - Bản thân cài đặt của **getNextGuess()** và các hàm khác (độ thông minh của thuật toán), chương trình chính không cần biết → **private**

*Có thể tách guesser và dữ liệu liên quan thành mô-đun riêng*

# Phân tích chức năng của guesser

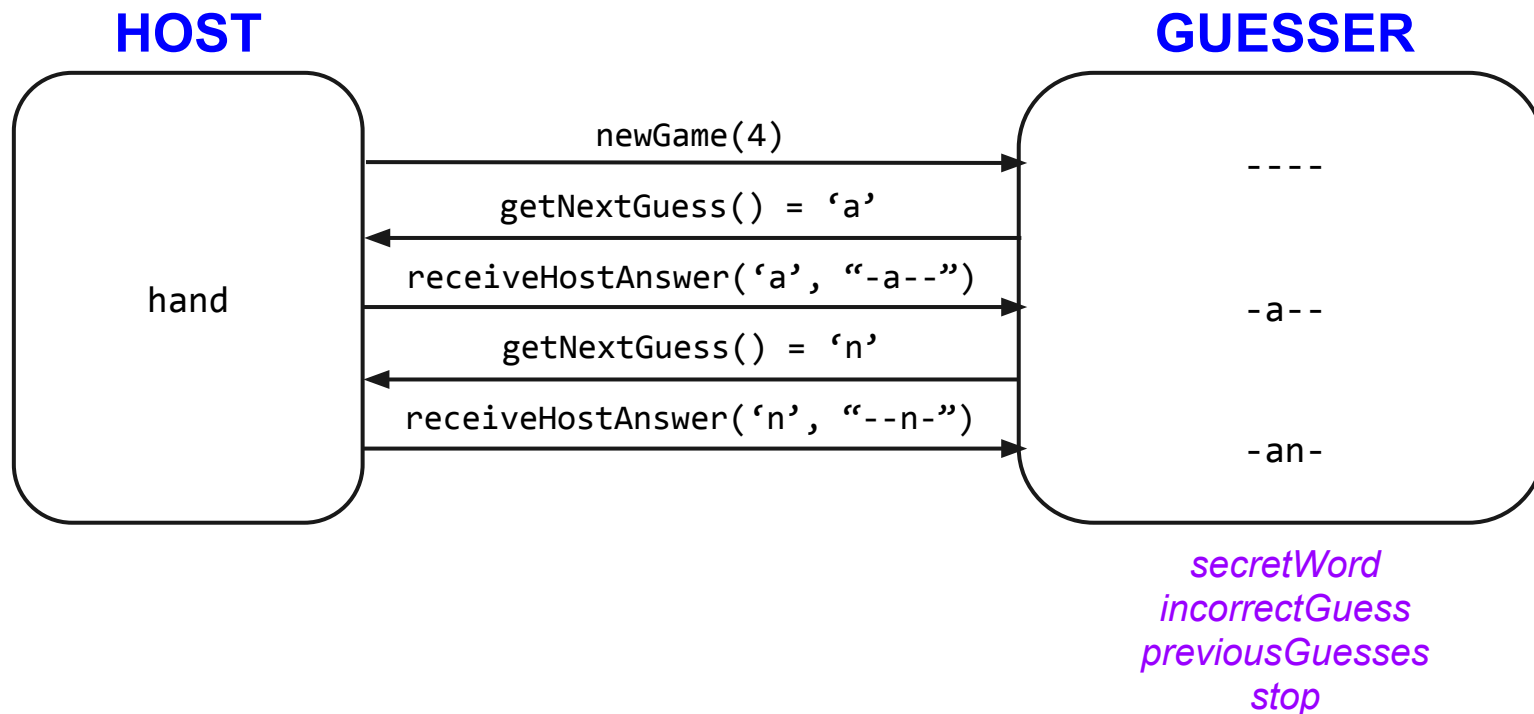
## Các chức năng

- Bắt đầu ván chơi **newGame()** với độ dài từ cho trước
- Đoán ký tự tiếp theo **getNextGuess()**
- Nhận trả lời của chủ trò **receiveHostAnswer()**

Phía ngoài (chủ trò, hệ thống) chỉ cần biết các chức năng này của guesser còn bên trong

- guesser đoán thế nào không cần biết
- guesser quản lý dữ liệu thế nào không cần biết

# Phân tích chức năng của guesser



# Class

- C++ hỗ trợ cơ chế đóng gói hàm và dữ liệu
- Ví dụ: trong tệp **MyClass.h**

```
class MyClass {
```

```
private:
```

```
    int value;
```

```
    bool checkNewValue(int newValue);
```

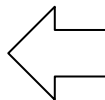
```
public:
```

```
    MyClass();
```

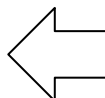
```
    void setValue(int newValue);
```

```
    int getValue();
```

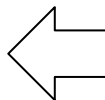
```
};
```



Các đối tượng thuộc lớp MyClass sẽ có dữ liệu kiểu nguyên value



hàm dành riêng cho các đối tượng của lớp, bên ngoài không dùng được



các hàm “của công”, bên ngoài có thể gọi được



# Cài đặt hàm trong MyClass.cpp

Thêm phạm vi **MyClass::** vào trước tên hàm

```
#include "MyClass.h"

MyClass::MyClass()
{
    value = 0;
}

bool MyClass::checkNewValue(int newValue)
{
    return newValue >= 0;
}
```

```
void MyClass::setValue(int newValue)
{
    if (checkNewValue(newValue))
        value = newValue;
}

int MyClass::getValue()
{
    return value;
}
```

# Sử dụng lớp

```
MyClass obj; // gọi hàm khởi tạo MyClass::MyClass()
cout << obj.getValue() << endl; // 0
obj.setValue(5);
cout << obj.getValue() << endl; // 5
obj.setValue(-5);
cout << obj.getValue() << endl; // 5, không thay đổi
cout << obj.checkNewValue(50); // lỗi biên dịch
cout << obj.value << endl; // lỗi biên dịch
```

## Ở bên ngoài

- chỉ cần biết các hàm `setValue()`, `getValue()`
- không cần biết `checkNewValue()`
  - Là xử lý nội tại bên trong `MyClass`

# Sử dụng lớp

```
MyClass another_obj; // gọi hàm khởi tạo MyClass::MyClass()
cout << another_obj.getValue() << endl; // 0
another_obj.setValue(7);
cout << another_obj.getValue() << endl; // 7
cout << obj.getValue() << endl; // 5, obj chứa value khác another_obj
another_obj.setValue(-3);
cout << another_obj.getValue() << endl; // 7, không thay đổi
```

Các biến cùng kiểu **MyClass** gọi là *đối tượng* thuộc lớp **MyClass**

# Cấu trúc lại SimpleAI (refactor)

Mục đích: *đóng gói hàm và dữ liệu* của guesser bằng Class

- Dữ liệu của guesser
  - secretWord
  - incorrectGuess
  - previousGuesses
  - stop
  - wordList

# Chức năng của guesser

- Khởi tạo: đọc từ vựng
- Bắt đầu ván chơi: `newGame(wordLength)`
- Đoán chữ cái: `getNextGuess()`
  - Dựa vào `previousGuesses`, `secretWord`
  - Các dữ liệu này là dữ liệu nội tại của guesser
- Gửi trả lời của chủ trò (host)
  - `receiveHostAnswer(guess, mask)`
  - Nhận trả lời và cập nhật các dữ liệu trong guesser
- Ngoài ra, còn các chức năng “của riêng”

# Xây dựng lớp Guesser

Trong tệp **guesser.h**, khai báo lớp Guesser, trước tiên đưa vào các dữ liệu cần thiết

```
class Guesser
{
private:
    std::string secretWord;
    int incorrectGuess;
    std::set<char> previousGuesses;
    bool stop;
    std::vector<std::string> wordList;
};
```

# Hàm khởi tạo

Có tên trùng với tên lớp **Guesser()**

guesser.h

```
...  
public:  
    Guesser();  
};
```

guesser.cpp

```
Guesser::Guesser()
```

```
{
```

```
    wordList =
```

```
        readWordListFromFile(  
            
```

```
            "data/Ogden_Picturable_200.txt"
```

```
        );
```

```
}
```

khởi tạo vốn từ  
vựng của đối tượng  
thay thế cho biến  
static của hàm cũ

# Hàm newGame()

Thay hàm `initialize()` cũ, khởi tạo các biến

`secretWord`, `previousGuesses`, `incorrectGuess`, `stop`

guesser.h

```
...  
public:  
    Guesser();  
    void newGame(int wordLength);  
};
```

guesser.cpp

```
void Guesser::newGame(int wordLength)  
{  
    secretWord = string(wordLength, '-');  
    incorrectGuess = 0;  
    previousGuesses = set<char>();  
    stop = false;  
}
```



mã giống hệt `initialize()`  
nhưng đây là dữ liệu của lớp



# Hàm receiveHostAnswer()

Chuyển hóa từ `update()` cũ

Không cần truyền `secretWord` (tại sao?)

```
public:
...
    void receiveHostAnswer(char guess,
                           const std::string& mask);
};
```

guesser.h

# Hàm receiveHostAnswer()

```
void Guesser::receiveHostAnswer(char guess, const std::string& mask)
{
    if (!isGoodMask(guess, mask, secretWord))
        throw invalid_argument("mistake entering answer");

    previousGuesses.insert(guess);
    if (isAllDash(mask)) {
        incorrectGuess ++;
        if (incorrectGuess == MAX_GUESSES) stop = true;
    } else {
        updateSecretWord(mask, secretWord);
        if (isAllNotDash(secretWord)) stop = true;
    }
} // sao chép nguyên xi hàm update(), bỏ đi tham số secretWord
// báo lỗi biên dịch isGoodMask, MAX_GUESSES, updateSecretWord
```

guesser.cpp

# Hàm receiveHostAnswer()

- `isGoodMask()`, `updateSecretWord()`: là chức năng “của riêng” Guesser → đặt trong private
- `MAX_GUESSES`: tùy vào ý định của người viết có muốn bên ngoài nhìn thấy giá trị hằng số này

...

**private:**

```
bool isGoodMask(char guess, const std::string& mask);
```

```
void updateSecretWord(const std::string& mask);
```

**public:**

```
const int MAX_GUESSES = 7;
```

...

```
};
```

guesser.h

# Hàm Guesser::isGoodMask()

Không cần tham số **secretWord** nữa

```
bool Guesser::isGoodMask(char guess, const string& mask)
{
    if (mask.length() != secretWord.length()) return false;
    for (unsigned int i = 0; i < secretWord.length(); i++)
        if (mask[i] != '-') {
            if (mask[i] != guess)
                return false;
            if (secretWord[i] != '-' && secretWord[i] != mask[i])
                return false;
        }
    return true;
}
```

guesser.cpp

# Guesser::updateSecretWord()

Sửa dữ liệu **secretWord**

guesser.cpp

```
void Guesser::updateSecretWord(const string& mask)
{
    for (unsigned int i = 0; i < secretWord.length(); i++)
        if (mask[i] != '-')
            secretWord[i] = mask[i];
}
```

# Hàm getNextGuess()

Chuyển hóa từ **update()** cũ

Không cần truyền **previousGuesses**, **secretWord**

```
public:
```

```
...
```

```
    char getNextGuess();
```

```
};
```

guesser.h

# Hàm getNextGuess()

guesser.cpp

```
char Guesser::getNextGuess()
{
    set<char> remainingChars = getRemainingChars(previousGuesses);
    if (remainingChars.size() == 0)
        return 0;

    if (isAllDash(secretWord))
        return getVowelGuess(remainingChars);

    vector<string> filteredWordList =
        getSuitableWords(wordList, secretWord, remainingChars);
    map<char, int> occurrenceCount =
        getOccurrenceCount(remainingChars, filteredWordList);
    return getMaxOccurenceChar(remainingChars, occurrenceCount);
}
```

# Các hàm tiện ích

Đưa các hàm

- `getRemainingChars()`
- `getVowelGuess()`
- `selectRandomChar()`
- `getOccurenceCount()`
- `getMaxOccurenceChar()`
- `isSuitableWord()`
- `getSuitableWords()`

vào phần `private`



# Các hàm tiện ích

**private:**

```
...
std::set<char> getRemainingChars(const std::set<char>& previousGuesses);
char getVowelGuess(const std::set<char>& remainingChars);
char selectRandomChar(const std::set<char>& s);
std::map<char, int> getOccurenceCount(const std::set<char>& remainingChars,
                                     const std::vector<std::string>& wordList);
char getMaxOccurenceChar(const std::set<char>& remainingChars,
                        const std::map<char, int>& count);
bool isSuitableWord(const std::string& word,
                  const std::string& secretWord, const std::set<char>& remainingChars);
std::vector<std::string> getSuitableWords(
    const std::vector<std::string>& wordList,
    const std::string& secretWord,
    const std::set<char>& remainingChars);
```

- Thêm `Guesser::` vào trước cài đặt các hàm này trong `guesser.cpp`
- Xóa khai báo và cài đặt của hàm `getNextGuess()` cũ khỏi `guesser.*`

# Sử dụng lớp Guesser

Lúc này

- `guesser.*` chỉ còn khai báo và cài đặt của lớp `Guesser`
- Nếu dịch, sẽ thấy báo lỗi không tìm thấy hàm `getNextGuess()` ở `main.cpp`
- Cần sử dụng lớp `Guesser` ở `main.cpp`

# Sử dụng lớp Guesser

- Loại bỏ các biến ở đầu hàm `main()`, thay thế bằng đối tượng `guesser`
- Khởi động trò chơi bằng `newGame()`

```
int wordLength;  
string secretWord;  
int incorrectGuess;  
set<char> previousGuesses;  
bool stop;  
  
initialize(wordLength, secretWord,  
           incorrectGuess, previousGuesses, stop);
```



```
Guesser guesser;  
guesser.newGame(getUserWordLength());
```

# Hàm render()

Có lỗi: hàm `render()` dùng tham số là dữ liệu của `guesser`, có 2 cách sửa

- Đưa `render()` vào Guesser
  - Cố định cách vẽ của trò chơi
- Tạo hàm lấy dữ liệu của Guesser
  - Mềm dẻo hơn, cho phép `main()` vẽ theo ý mình
  - Sẽ làm theo cách này để lấy các dữ liệu
    - `incorrectGuess, previousGuesses`
    - `secretWord, stop`

# Getters

Các hàm này ngắn, có thể viết ngay trong **guesser.h**

- Chỉ lấy dữ liệu, không sửa dữ liệu (**const**)

**public:**

guesser.h

...

```
int getIncorrectGuess() const { return incorrectGuess; }  
std::set<char> getPreviousGuesses() const { return previousGuesses; }  
bool isStop() const { return stop; }  
std::string getSecretWord() const { return secretWord; }
```

# Hàm render()

Thay tham số là **const Guesser&**

```
void render(const Guesser& guesser)
{
    clearScreen();
    cout << endl << "Incorrect guess = " << guesser.getIncorrectGuess()
        << " previous guesses = ";
    for (char c : guesser.getPreviousGuesses())
        cout << c;
    cout << " secretWord = " << guesser.getSecretWord() << endl;
    cout << getDrawing(guesser.getIncorrectGuess()) << endl;
}
```

Gọi hàm **render(guesser)** để vẽ

# Sử dụng lớp Guesser

```
char guess = getNextGuess(previousGuesses, secretWord);
```



```
char guess = guesser.getNextGuess();
```

```
update(guess, mask, incorrectGuess, previousGuesses, secretWord, stop);
```



```
guesser.receiveHostAnswer(guess, mask);
```

```
!stop
```



```
!guesser.isStop()
```

# playAnimation()

```
void playAnimation(const Guesser& guesser)
{
    clearScreen();
    bool isLosing = guesser.getIncorrectGuess() == guesser.MAX_GUESSES;
    const string& word = guesser.getSecretWord();
    while (true) {
        if (isLosing)
            cout << endl << "I lost :(. My best word is: " << word << endl;
        else
            cout << endl << "Haha, I win :D. The word is: " << word << endl;
        cout << (isLosing ? getNextHangman() : getNextStandingman());
        this_thread::sleep_for(chrono::milliseconds(500));
        clearScreen();
    }
}
```

Gọi hàm:  
`playAnimation(guesser)`



# Hoàn thành đóng gói

- Xóa các hàm `initialize`, `updateSecretWord`, `update`, `isGoodMask` khỏi `main.cpp`
- Chương trình trong `main.cpp` chỉ còn
  - Nhập liệu: `getUserWordLength`, `getUserAnswer`
  - Hiển thị: `render`, `playAnimation`
  - Vòng lặp chính sử dụng `guesser`
    - Không cần biết chi tiết `guesser` đoán như thế nào
    - Chỉ giao tiếp thông qua
      - `newGame`, `getNextGuess`, `receiveHostAnswer`
      - Các getters

# Hangman 4.0

- Đóng gói hàm và dữ liệu của **Guesser**
- Phân biệt các hàm **private** “của riêng” **Guesser** và các hàm **public** có thể gọi từ bên ngoài
- Giao tiếp thông qua các hàm **public**
- Sử dụng getters (với từ khóa **const**) để lấy dữ liệu

<https://github.com/tqlong/advprogram/archive/aacb5c9fbfde9876b586bab9aa409994dffc856c.zip>

# Nội dung

- Lớp (Class) = Dữ liệu + Hàm
  - Đóng gói một mô-đun
- **Đánh giá SimpleAI**
  - Máy chơi với máy
- **Kỹ thuật**
  - Class
  - Hàm khởi tạo
  - Hàm public, hàm private, hàm const

# Đánh giá SimpleAI

- Lớp Guesser đã đóng gói 1 thuật toán đoán ký tự tiếp theo
- Ta cần đánh giá thuật toán này tốt hay xấu
  - Trên tập từ vựng nào ?
  - Độ đo là gì ?
    - Số lần đoán sai
    - Do mỗi từ có số lần đoán sai khác nhau  
→ lấy trung bình cộng số lần sai trên tập từ vựng làm độ đo
    - Trung bình cộng số lần sai càng nhỏ càng tốt

# Đánh giá SimpleAI - mã giả

```
testWordList = readWordListFromFile(testFile)
sum = 0
for (word : testWordList) {
    run guesser until stop to guess word
        using generated masks for host answers
    add guesser.getIncorrectGuess() to sum
}
return sum / testWordList.size()
```

# Hàm main() mới

```
int main(int argc, char* argv[])
{
    string testFile = argc > 1 ? argv[1] : "data/Ogden_Picturable_200.txt";
    vector<string> testWordList = readWordListFromFile(testFile);

    double totalGuess = 0;
    for (const string& word : testWordList) {
        Guesser guesser;
        guesser.newGame(word.length());
        do {
            char guess = guesser.getNextGuess();
            if (guess == 0) { // guesser chịu thua
                totalGuess += guesser.MAX_GUESSES;
                break;
            }
        }
```

Chuyển hàm `main()` cũ  
thành hàm `playHangman()`

- Lưu lại code chơi  
Hangman cũ để sau này  
có thể cần dùng lại hoặc  
tham khảo cách dùng  
`Guesser`

# Hàm main() mới

```
        guesser.receiveHostAnswer(guess, getMask(guess, word));
        if (guesser.isStop()) totalGuess += guesser.getIncorrectGuess();
    } while (!guesser.isStop());
}
cout << "For testFile " << testFile
      << ", average number of guesses = " << totalGuess / testWordList.size()
      << endl;
return 0;
}
```

Máy chơi với máy - sinh mặt nạ từ **guess** và **word**

Chạy thử sẽ thấy  
con số 1.885 trên  
bộ từ vựng sẵn có

```
string getMask(char guess, const string& word)
{
    string mask(word.length(), '-');
    for (unsigned int i = 0; i < word.length(); i++)
        if (tolower(word[i]) == guess) mask[i] = guess;
    return mask;
}
```

# Đánh giá trên nhiều bộ từ vựng

- Guesser hiện đang dùng từ vựng
  - Ogden\_Picturable\_200.txt
  - Xem hàm khởi tạo Guesser::Guesser()
- Để tăng “trí tuệ” của Guesser
  - Cho phép khởi tạo **wordList** với bộ từ vựng khác
  - Dùng hàm khởi tạo có tham số là tên tệp

```
public:
```

```
...
```

```
Guesser(const std::string&  
        wordFile);
```

guesser.h

```
Guesser::Guesser(const string& wordFile)  
{  
    wordList = readWordListFromFile(wordFile);  
}
```

guesser.cpp



# Thử dùng các bộ từ vựng

- Thay thế khai báo

`Guesser guesser;`

Bằng khai báo

`Guesser guesser("data/dictionary.txt");`

Download ở

<https://github.com/mrdziuban/Hangman/blob/master/dictionary.txt>

- Thử nghiệm xong, dùng tham số dòng lệnh để nhập tên tệp từ vựng

# Assessment 1.0

<https://github.com/tqlong/advprogram/archive/faf729381c8ecb47578a170cad022102841e53b1.zip>

```
int main(int argc, char* argv[])
{
    string testFile = argc > 1 ? argv[1] : "data/Ogden_Picturable_200.txt";
    string dictFile = argc > 2 ? argv[2] : "data/dictionary.txt";

    ...

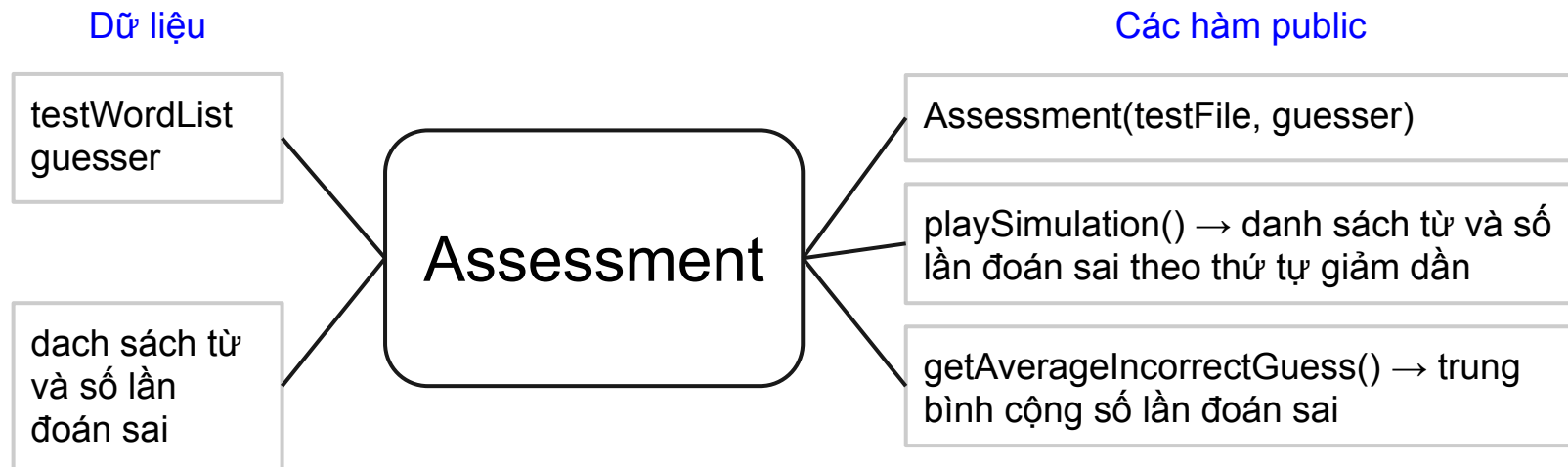
    for (const string& word : testWordList) {
        Guesser guesser(dictFile);
        ...
    }
}
```

# Tiếp tục cấu trúc và tối ưu mã

Trong phần này

- Đóng gói các đoạn mã đánh giá thành lớp **Assessment** vào **assessment**.\*
- Cho phép liệt kê các từ theo thứ tự giảm dần số lần đoán sai
  - Biết từ nào khó đoán
- Cải tiến tốc độ của **guesser**

# Dữ liệu và hàm của Assessment



# Hàm khởi tạo Assessment()

```
#pragma once
```

assessment.h

```
#include <string>
#include <vector>
#include "guesser.h"
```

```
class Assessment
```

```
{
```

```
    std::vector<std::string> testWordList;
```

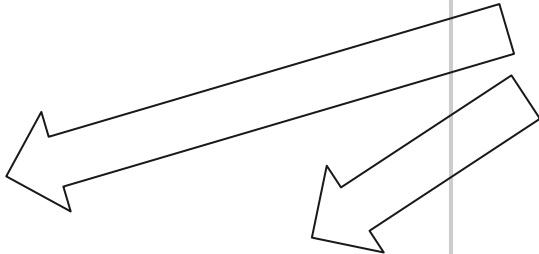
```
    Guesser& guesser;
```

```
public:
```

```
    Assessment(const std::string& testFile, Guesser& guesser_);
```

```
};
```

sẽ đưa  
tham chiếu  
vào phần  
dữ liệu



# Hàm khởi tạo Assessment()

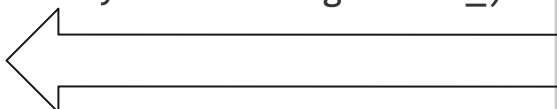
```
#include "assessment.h"  
#include "util.h"
```

```
using namespace std;
```

```
Assessment::Assessment(const string& testFile, Guesser& guesser_)  
    : guesser(guesser_)  
{  
    testWordList = readWordListFromFile(testFile);  
}
```

assessment.cpp

danh sách  
khởi tạo  
dữ liệu lớp



# playSimulation(): máy chơi với máy

- Cần lưu số lần đoán sai mỗi từ
  - Định nghĩa `struct WordCount` gồm từ và số đếm
  - `struct` giống `class` nhưng mặc định là `public`

```
struct WordCount
{
    std::string word;
    int count;
    WordCount(
        const std::string& word_,
        int count_)
        : word(word_),
          count(count_) {}
};
```

assessment.h

```
class Assessment
{
private:
    ...
    std::vector< WordCount > wordIncorrectGuess;
public:
    ...
    void playSimulation();
};
```

assessment.h

↑  
lưu số lần đoán  
sai mỗi từ

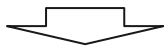
# playSimulation(): máy chơi với máy

```
void Assessment::playSimulation()
```

assessment.cpp

```
{  
    wordIncorrectGuess.clear();  
    for (const string& word : testWordList) {  
        guesser.newGame(word.length());  
        do {  
            char guess = guesser.getNextGuess();  
            if (guess == 0) {  
                wordIncorrectGuess.push_back(WordCount(word, guesser.MAX_GUESSES));  
                break;  
            }  
            guesser.receiveHostAnswer(guess, getMask(guess, word));  
            if (guesser.isStop())  
                wordIncorrectGuess.push_back(WordCount(word, guesser.getIncorrectGuess()));  
        } while (!guesser.isStop());  
    }  
}
```

dùng hàm khởi tạo  
để tạo đối tượng  
WordCount



Đưa hàm getMask  
thành private trong  
Assessment





# playSimulation(): sắp xếp kết quả

- Sử dụng hàm **sort** trong **<algorithm>**
- Cần làm hàm so sánh hai **WordCount**
  - <http://stackoverflow.com/questions/4892680/sorting-a-vector-of-structs>

```
#include <algorithm>
bool greaterWordCount(const WordCount& a, const WordCount& b)
{
    return a.count > b.count;
}
void Assessment::playSimulation()
{
    wordIncorrectGuess.clear();
    for (const string& word : testWordList) { ... }
    sort(wordIncorrectGuess.begin(), wordIncorrectGuess.end(), greaterWordCount);
}
```

assessment.cpp



Sửa thành nhỏ hơn <  
để sắp xếp tăng dần

# getAverageIncorrectGuess()

Dùng vòng lặp **for** duyệt **wordIncorrectGuess**

assessment.h

```
public:  
    ...  
    double getAverageIncorrectGuess();
```

assessment.cpp

```
double Assessment::getAverageIncorrectGuess()  
{  
    double totalGuess = 0;  
    for (const WordCount& p : wordIncorrectGuess)  
        totalGuess += p.count;  
    return totalGuess / wordIncorrectGuess.size();  
}
```

# Assessment 2.0

<https://github.com/tqlong/advprogram/archive/cf6e81dbcd38b1960225fbe4881145e90a0d81bc.zip>

```
#include <iostream>
#include "guesser.h"
#include "assessment.h"
using namespace std;
```



Bỏ hết các  
#include thừa, xóa  
các hàm không  
còn cần thiết

```
int main(int argc, char* argv[])
{
    string testFile = argc > 1 ? argv[1] : "data/Ogden_Picturable_200.txt";
    string dictFile = argc > 2 ? argv[2] : "data/dictionary.txt";
    Guesser guesser(dictFile);
    Assessment assessment(testFile, guesser);

    assessment.playSimulation();
    cout << "Using dictFile " << dictFile << endl
         << "on testFile " << testFile << endl
         << "average #incorrect guesses = " << assessment.getAverageIncorrectGuess()
         << endl;
    return 0;
}
```



Sử dụng hai mô-đun  
đã đóng gói

# Bài tập

- Sử dụng `map` thay cho `vector` để lưu số lần đoán sai mỗi từ
  - `map<string, int> wordIncorrectGuess;`
- Cải tiến tốc độ của Guesser
  - Mỗi lần lọc từ, dùng `vector<string>` sẽ chậm
  - Thay thế bằng `vector<int>` các chỉ số từ hợp lệ
  - Mỗi lần chỉ lọc trên các từ hợp lệ của lần đoán trước
    - Không lọc lại từ danh sách từ ban đầu