Simple AI

7 - Tìm kiếm và đếm

https://github.com/tqlong/advprogram

Nội dung

- Máy chơi Hangman
- Chương trình phức tạp → Mã giả + chia để trị
- AI = Dữ liệu + Tìm kiếm + Đếm (thống kê)
- Kỹ thuật:
  - Thư viện tập hợp **<set>**, thư viện ánh xạ **<map>**
  - Vòng lặp for trên vector, set, map
  - Tìm kiếm
    - Tìm kiếm thỏa mãn điều kiện
    - Tìm kiếm lớn nhất, nhỏ nhất
  - Đếm

Đặt vấn đề

Lập trình cho máy chơi trò Hangman:

- Người nghĩ từ
- Máy đoán các chữ cái
- Người trả lời các vị trí chữ cái đoán đúng

…

Người - chủ trò (host); Máy - người chơi (player)

Các thành phần

Giao diện tương tác (UI)

- Nhập số chữ cái của từ người chơi nghĩ (dễ)
- Hiển thị phán đoán, lịch sử phán đoán của máy và giá treo (đã làm)
- Nhập trả lời của người chơi

Lõi trí tuệ nhân tạo (AI core)

- Dựa vào các phán đoán đã đưa ra và ***secretWord*** hiện thời
  - Đưa ra phán đoán tiếp theo
  - Liệu máy tính có thể chơi Hangman giỏi hơn con người ?

Nhập trả lời của người chơi

Khi máy đưa ra phán đoán, người chơi trả lời bằng ***xâu mặt nạ*** (mask)

- Một xâu ký tự toàn dấu gạch ngang
- Chỉ hiển thị các vị trí đoán đúng

Ví dụ: người nghĩ từ "**hangman**"

máy đoán **p**, người trả lời **-------**

máy đoán tiếp **a**, người trả lời tiếp **-a---a-**

máy đoán tiếp **g**, người trả lời tiếp **-a-g-a-**

Tiện ích sinh xâu mặt nạ

```cpp
// genmask.cpp
// Mask generating tool for Hangman game
#include <iostream>
#include <cctype>
using namespace std;

int main(int argc, char* argv[])
{
    if (argc < 3) {
        cout << "Usage: genmask <word> <char>" << endl;
        return 1;
    }

    string word = argv[1];
    char guess = tolower(argv[2][0]);
    for (unsigned int i = 0; i < word.length(); i++)
        if (tolower(word[i]) != guess) word[i] = '-';
        else word[i] = guess;
    cout << word << endl;
    return 0;
}
```

Chuyển **word** sang mặt nạ, các ký tự khác **guess** biến thành dấu gạch ngang

Mã giả - chia để trị

```
wordLength = getUserWordLength();
secretWord = string(wordLength, '-');
incorrectGuess = 0;
previousGuesses = empty set of characters;
stop = false;
do {
    guess = getNextGuess(previousGuesses, secretWord);
    mask = getUserAnswer(guess);
    update(guess, mask, incorrectGuess, previousGuesses, secretWord, stop);
    render(incorrectGuess, previousGuesses, secretWord);
} while (!stop);
playAnimation(incorrectGuess == MAX_GUESSES, secretWord);
```

Trí tuệ nhân tạo (AI)

Lập trình nhóm

- Dự án phức tạp nhiều người
  - Mỗi người làm một phần
- Dự án này
  - Một người làm giao diện
  - Một người làm phần lõi AI (**getNextGuess**)
    - Đây là phần khó, chưa biết làm thế nào
      - Nếu đợi → làm chậm dự án
    - Cần một hàm **getNextGuess()** đơn giản để bên làm giao diện có thể phát triển độc lập
    - Đồng thời, bên làm AI có thể tìm cách cải tiến

Tạo Project

- Trong CodeBlocks tạo Project SimpleAI
- Tạo tệp guesser.h, guesser.cpp thêm vào Project
  - Thêm hàm **char getNextGuess(string, string)** vào guesser.*
  - **#include "guesser.h"** trong main.cpp

guesser.h

```
#pragma once

#include <string>
#include <set>

char getNextGuess(const std::set<char>& previousGuesses, const std::string& secretWord);
```

Giới thiệu thư viện <set>

- previousGuesses cần lưu tập hợp các chữ cái đã đoán
- **<set>**: tập hợp các giá trị cùng kiểu
  - set<int>: tập hợp (con) các số nguyên
  - set<char>: tập hợp các ký tự
  - set<string>: tập hợp các xâu ký tự
- Các phần tử trong tập hợp đảm bảo luôn khác nhau (!=)

Giới thiệu thư viện <set>

- Các phép toán tập hợp:
  - s.insert('a'): thêm phần tử 'a' vào tập s
  - s.erase('a'): xóa phần tử 'a' khỏi tập s
  - s.find('a') != s.end(): phần tử 'a' thuộc tập s
  - s.find('a') == s.end(): phần tử 'a' không thuộc tập s
  - for (char c : s): duyệt các phần tử trong tập s
  - http://www.cplusplus.com/reference/set/set/

getNextGuess đơn giản

Chọn ngẫu nhiên 1 ký tự chưa đoán bao giờ

- Thêm util.* vào Project
- **#include "util.h"** trong guesser.cpp

guesser.cpp

```
#include <string>
#include "guesses.h"
#include "util.h"

using namespace std;

char getNextGuess(const set<char>& previousGuesses,
                  const string& secretWord)
{
    set<char> remainingChars =
        getRemainingChars(previousGuesses);
    if (remainingChars.size() == 0)
        return 0;
    else
        return selectRandomChar(remainingChars);
}
```

getRemainingChars()

Bắt đầu, remainChars = tập chữ cái từ a → z

sau đó xóa các chữ cái trong previousGuesses

```
set<char> getRemainingChars(const set<char>& previousGuesses)
{
    set<char> remainingChars;
    for (char c = 'a'; c <= 'z'; c++)
        remainingChars.insert(c);
    for (char c: previousGuesses)
        remainingChars.erase(c);
    return remainingChars;
}
```

selectRandomChar()

Google “c++ select random element from set”

http://stackoverflow.com/questions/3052788/how-to-select-a-random-element-in-stdset

```
char selectRandomChar(const set<char>& s) {
    int r = rand() % s.size();
    for (char c : s) {
        if (r-- == 0) return c;
    }
    return 0;
}
```

Lập trình giao diện

- Đã có lõi AI đơn giản
- Có thể phát triển giao diện riêng rẽ
  - Phát triển thêm từ code Hangman cũ
- Người làm AI tiếp tục tìm hiểu để cải tiến cách phán đoán (thuật toán)

main(): chuyển từ mã giả sang

Chia để trị

Viết mã lần lượt cho các hàm

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

    initialize(wordLength, secretWord, incorrectGuess, previousGuesses, stop);

    render(incorrectGuess, previousGuesses, secretWord);
    do {
        char guess = getNextGuess(previousGuesses, secretWord);
        string mask = getUserAnswer(guess);
        update(guess, mask, incorrectGuess, previousGuesses, secretWord, stop);
        render(incorrectGuess, previousGuesses, secretWord);
    } while (!stop);
    playAnimation(incorrectGuess == MAX_GUESSES, secretWord);
    return 0;
}
```

getUserWordLength()

Nhập độ dài từ người chơi nghĩ

```
int getUserWordLength()
{
    int wordLength;
    cout << endl << "Enter your word length: ";
    cin >> wordLength;
    return wordLength;
}
```

getUserAnswer()

Nhập (mặt nạ) trả lời của người chơi, chuyển qua chữ thường

```
string getUserAnswer(char guess)
{
    string answer;
    cout << endl << "I guess " << guess << ", please enter your mask: ";
    cin >> answer;
    transform(answer.begin(), answer.end(), answer.begin(), ::tolower);
    return answer;
}
```

initialize()

Khởi tạo các trạng thái của trò chơi

```
void initialize(int& wordLength, string& secretWord,
                int& incorrectGuess, set<char>& previousGuesses,
                bool& stop)
{
    wordLength = getUserWordLength();
    secretWord = string(wordLength, '-');
    incorrectGuess = 0;
    previousGuesses = set<char>();
    stop = false;
}
```

render()

Sử dụng lại các hàm trong draw.* (nhớ include)

for (char c: previousGuesses) in các phần tử

```
void render(int incorrectGuess, const set<char>& previousGuesses,
            const string& secretWord)
{
    clearScreen();
    cout << endl << "Incorrect guess = " << incorrectGuess
         << "   previous guesses = ";
    for (char c : previousGuesses)
        cout << c;
    cout << "   secretWord = " << secretWord << endl;
    cout << getDrawing(incorrectGuess) << endl;
}
```

playAnimation()

Sửa **playAnimation()** một ít cho phù hợp

```
void playAnimation(bool isLosing, const string& word)
{
    clearScreen();
    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();
    }
}
```

update(): viết như kể chuyện

```
void update(char guess, const string& mask,
            int& incorrectGuess, set<char>& previousGuesses,
            string& secretWord, bool& stop)
{
    Nếu mặt nạ không hợp lệ, báo lỗi (ném ngoại lệ)

    Thêm guess vào previousGuesses (các ký tự đã đoán)
    Nếu mặt nạ toàn dấu gạch ngang
      tăng incorrectGuess
      nếu incorrectGuess == MAX_GUESSES (7), stop = true
    Ngược lại
      cập nhật secretWord dựa vào mặt nạ
      nếu secretWord không còn dấu gạch ngang, stop = true
}
```

update(): viết như kể chuyện

```
void update(char guess, const string& mask,
            int& incorrectGuess, set<char>& previousGuesses,
            string& secretWord, bool& stop)
{
    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;
    }
}
```

isAllDash(): trong util.*

Kiểm tra toàn bộ chữ cái là dấu gạch ngang

```
bool isAllDash(const string& s)
{
    for (unsigned int i = 0; i < s.length(); i++)
        if (s[i] != '-') return false;
    return true;
}
```

isAllDash(): trong util.*

Kiểm tra toàn bộ chữ cái là dấu gạch ngang

```
bool isAllDash(const string& s)
{
    for (char c : s)
        if (c != '-') return false;
    return true;
}
```

isAllNotDash(): trong util.*

Kiểm tra toàn bộ chữ cái không là dấu gạch ngang

```
bool isAllNotDash(const string& s)
{
    for (unsigned int i = 0; i < s.length(); i++)
        if (s[i] == '-') return false;
    return true;
}
```

isAllNotDash(): trong util.*

Kiểm tra toàn bộ chữ cái không là dấu gạch ngang

```
bool isAllNotDash(const string& s)
{
    for (char c : s)
        if (c == '-') return false;
    return true;
}
```

updateSecretWord()

Hiển thị các chữ cái trong mặt nạ (mask)

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

isGoodMask()

```
bool isGoodMask(char guess, const string& mask,
                const string& secretWord)
{
    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;
}
```

mặt nạ hợp lệ

độ dài phải bằng nhau

nếu **mask[i]** là chữ cái thì **mask[i]** phải bằng **guess** và **secretWord[i]** phải là dấu gạch hoặc phải bằng **mask[i]**

Sửa hàm main() bắt ngoại lệ

```
do {
    char guess = getNextGuess(previousGuesses, secretWord);
    if (guess == 0) {
        cout << "I give up, hang me" << endl;
        return 0;
    }
    do {
        try {
            string mask = getUserAnswer(guess);
            update(guess, mask, incorrectGuess, previousGuesses, secretWord, stop);
            break;
        } catch (invalid_argument e) {
            cout << "Invalid mask, try again" << endl;
        }
    } while (true);
    render(incorrectGuess, previousGuesses, secretWord);
} while (!stop);
```

Đến đây, mỗi người có thể làm phần của mình độc lập

Nội dung

- Máy chơi Hangman
- Chương trình phức tạp → Mã giả + chia để trị
- **AI = Dữ liệu + Tìm kiếm + Đếm (thống kê)**
- Kỹ thuật:
  - Thư viện tập hợp **<set>**, ánh xạ **<map>**
  - Vòng lặp for trên vector, set, map
  - Tìm kiếm
    - Tìm kiếm thỏa mãn điều kiện
    - Tìm kiếm lớn nhất, nhỏ nhất
  - Đếm

Simple AI

getNextGuess() hiện thời

- may rủi, có tỉ lệ thua cao
- đơn giản, dễ cài đặt

→ dùng làm *thuật toán tạm thời* để phát triển độc lập các thành phần của chương trình

Simple AI

Cải tiến **getNextGuess()** như con người chơi

- B0: Chuẩn bị vốn từ vựng tiếng Anh
- B1: Thử đoán các nguyên âm **a, e, i, o, u**
  - Khi còn chưa đoán đúng
- B2: Sau khi đoán đúng một số vị trí
  - Lọc trong vốn từ vựng ra các từ
    - Có độ dài phù hợp
    - Có các chữ cái đã đoán đúng vị trí
  - Chọn chữ cái có khả năng cao nhất để đoán tiếp

B0: Chuẩn bị vốn từ vựng tiếng Anh

Sử dụng lại hàm đọc danh sách từ **readWordListFromFile()**

Sử dụng static chỉ đọc file 1 lần

Đổi file = đổi từ vựng

guesser.cpp

```
char getNextGuess(const set<char>& previousGuesses,
                  const string& secretWord)
{
    static vector<string> wordList =
        readWordListFromFile("data/Ogden_Picturable_200.txt");
    set<char> remainingChars = getRemainingChars(previousGuesses);

    // TODO: make a guess (B1, B2)
    ...
}
```

B1: ban đầu đoán nguyên âm

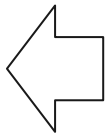
Nếu secretWord toàn dấu gạch, chọn 1 nguyên âm chưa đoán trong **a, e, i, o, u** để đoán

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

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

...
```

getVowelGuess(): tìm nguyên âm

Trả về 0 nếu không tìm thấy nguyên âm

```
char getVowelGuess(const set<char>& remainingChars)
{
    char vowel[] = {'a', 'e', 'i', 'o', 'u'};
    for (int i = 0; i < 5; i++) {
        if (remainingChars.find(vowel[i]) != remainingChars.end())
            return vowel[i];
    }
    return 0;
}
```

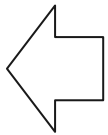


Kiểm tra vowel[i] có nằm trong remainingChars

getVowelGuess(): tìm nguyên âm

Vòng lặp for trên vector

```
char getVowelGuess(const set<char>& remainingChars)
{
    char vowel[] = {'a', 'e', 'i', 'o', 'u'};
    for (char c : vowel) {
        if (remainingChars.find(c) != remainingChars.end())
            return c;
    }
    return 0;
}
```



Kiểm tra c
có nằm trong
remainingChars

getVowelGuess(): thứ tự tìm

Đoán nguyên âm có tần suất xuất hiện cao trước

- Google: letter frequency
  - https://en.wikipedia.org/wiki/Letter_frequency
  - Thứ tự: **e, a, o, i, u**

```
char vowel[] = {'e', 'a', 'o', 'i', 'u'};
```

- Cũng có thể tính tần suất các ký tự trong từ vựng của mình (sẽ làm)

B2: lọc từ và chọn chữ cái

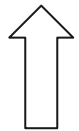
- Lọc từ trong từ vựng
  - Độ dài phải bằng **secretWord**
  - Có các chữ cái ở vị trí giống **secretWord** (trừ dấu gạch ngang)
  - Còn lại chỉ chứa các ký tự trong **remainingChars**

```
vector<string> filteredWordList =
    getSuitableWords(wordList, secretWord, remainingChars);
```

getSuitableWords()

Duyệt mảng **wordList** để *tìm* các từ phù hợp

```
vector<string> getSuitableWords(const vector<string>& wordList,
                                const string& secretWord,
                                const set<char>& remainingChars)
{
    vector<string> result;
    for (unsigned int i = 0; i < wordList.size(); i++)
        if (isSuitableWord(wordList[i], secretWord, remainingChars))
            result.push_back(wordList[i]);
    return result;
}
```

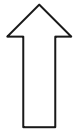


thỏa mãn điều kiện thì đưa vào kết quả

getSuitableWords()

Dùng lệnh for mới

```
vector<string> getSuitableWords(const vector<string>& wordList,
                                const string& secretWord,
                                const set<char>& remainingChars)
{
    vector<string> result;
    for (const string& word : wordList)
        if (isSuitableWord(word, secretWord, remainingChars))
            result.push_back(word);
    return result;
}
```

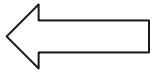


thỏa mãn điều kiện thì đưa vào kết quả

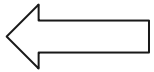
isSuitableWord()

- Có độ dài bằng **secretWord**
- Các chữ cái ở **secretWord** hiện đúng vị trí trong **word**
- Các chữ cái còn lại nằm trong **remainingChars**

```
bool isSuitableWord(const string& word, const string& secretWord,
                    const set<char>& remainingChars)
{
    if (word.length() != secretWord.length()) return false;
    for (unsigned int i = 0; i < word.length(); i++) {
        if (secretWord[i] != '-') {
            if (tolower(word[i]) != tolower(secretWord[i])) return false;
        }
        else if (remainingChars.find(word[i]) == remainingChars.end())
             return false;
    }
    return true;
}
```



các chữ cái đã hiện

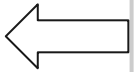


các chữ cái chưa hiện

getSuitableWords()

Bài tập: tách các bộ lọc riêng

```
vector<string> getSuitableWords(const vector<string>& wordList,
                                const string& secretWord,
                                const set<char>& remainingChars)
{
    vector<string> result;
    result = filterWordListByLen(secretWord.length(), wordList);
    result = filterWordListByMask(secretWord, result);
    result = filterWordListByRemainingChars(
                remainingChars, secretWord, result);
    return result;
}
```



làm các hàm này

B2: lọc từ và chọn chữ cái

- Chọn chữ cái
  - Đếm số lần xuất hiện các chữ cái (chưa đoán)
  - Chọn chữ cái có số lần xuất hiện cao nhất
- Chức năng *auto-complete* của Google
  - Chọn từ / ngữ *phù hợp* có số lần xuất hiện cao nhất

```
occurenceCount = getOccurenceCount(remainingChars, filteredWordList);
return getMaxOccurenceChar(remainingChars, occurenceCount);
```

Thư viện <map>

- Mỗi chữ cái cần lưu số lần xuất hiện
  - Ánh xạ từ ký tự (char) ra số nguyên (int)
  - Trong C++: map<char, int>
  - Thư viện <map>: http://www.cplusplus.com/reference/map/map/

Thư viện <map>

- Các thao tác với map:
  - map['a']=1: cho 'a' ánh xạ tới 1
    - 'a' gọi là *key*, 1 là *value*
  - map['a']: lấy giá trị ánh xạ tới bởi 'a'
  - map.insert('a', 1): tương tự như trên

Duyệt các phần tử của map

- Mỗi phần tử của map có dạng
  - struct pair { first, second }
  - first là key
  - second là value
- Duyệt qua map

```
for (auto p: my_map)
    cout << p.first << p.second << endl;
```

getOccurenceCount()

- Khởi tạo **count** là map<char, int>
  - *Khởi tạo count[c] = 0, với mọi ký tự c trong remainingChars*
- Duyệt qua các từ, với mỗi từ
  - Duyệt qua từng chữ cái
  - Tăng số đếm tương ứng trong **count** thêm 1

getOccurenceCount()

Tăng số đếm các ký tự trong danh sách từ

```
map<char, int> getOccurenceCount(const set<char>& remainingChars,
                                 const vector<string>& wordList)
{
    map<char, int> count;
    for (char c: remainingChars) count[c] = 0;

    for (unsigned int i = 0; i < wordList.size(); i++) {
        const string& word = wordList[i];
        for (unsigned int j = 0; j < word.length(); j++)
            if (count.find(word[j]) != count.end())
                count[word[j]]++;
    }
    return count;
}
```

getOccurenceCount()

Chuyển hết qua lệnh for mới

```
map<char, int> getOccurenceCount(const set<char>& remainingChars,
                                 const vector<string>& wordList)
{
    map<char, int> count;
    for (char c: remainingChars) count[c] = 0;

    for (const string& word : wordList) {
        for (char c : word)
            if (count.find(c) != count.end())
                count[c]++;
    }
    return count;
}
```

getMaxOccurenceChar()

Duyệt các cặp (key, value) trong **count**

Nếu **value > best_count** thì gán **best** bằng **c** và **best_count** bằng **value**

```
char getMaxOccurenceChar(const set<char>& remainingChars,
                         const map<char, int>& count)
{
    char best = 0;
    int best_count = 0;
    for (auto p : count)
        if (p.second > best_count) {
            best = p.first;
            best_count = p.second;
        }
    return best;
}
```

Simple AI 1.0

https://github.com/tqlong/advprogram/archive/9bc6614903304407ddee771d30cad02cf5051ecb.zip

```
char getNextGuess(const set<char>& previousGuesses, const string& secretWord)
{
    static vector<string> wordList = readWordListFromFile("data/Ogden_Picturable_200.txt");
    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> occurenceCount = getOccurenceCount(remainingChars, filteredWordList);
    return getMaxOccurenceChar(remainingChars, occurenceCount);
} // chỉ có hàm này được khai báo ở guesser.h, các hàm khác chỉ nằm trong guesser.cpp
```

Bài tập

- Dùng chương trình chuyển các nguồn sau
  https://github.com/dwyl/english-words/blob/master/words.txt
  https://github.com/mrdziuban/Hangman/blob/master/dictionary.txt
  http://stackoverflow.com/questions/4456446/dictionary-text-file
  http://www.gwicks.net/dictionaries.htm
  để **readWordListFromFile()** có thể đọc được, thử chơi với các tập từ vựng mới
- Có nhận xét gì về khả năng của SimpleAI khi thay đổi từ vựng

Bài tập

- Có bỏ phần **getVowelGuess()** được không ?
- Xử lý trường hợp từ cần đoán không có trong từ vựng, tức là

filteredWordList.size() == 0

- Kết thúc 1 ván chơi, hỏi có muốn chơi tiếp ? Cho chơi tiếp nếu người chơi đồng ý.
- Nếu không đoán được từ (**guess == 0**) hoặc thua cuộc, hỏi từ của người chơi rồi thêm vào vốn từ vựng và ghi xuống file (giúp lần chơi sau)