Game: Hangman

3 - Phát triển chương trình

https://github.com/tqlong/advprogram

Nội dung

- Trò chơi Hangman
- Sơ đồ khối, mã giả và tư tưởng chia để trị
  - Hình dung các thành phần của chương trình
- Kỹ thuật:
  - Thao tác với xâu ký tự trong C++
  - Bắt đầu với hàm đơn giản, dần dần biến đổi và luôn có chương trình chạy được

Cùng chơi Hangman

- Trò chơi giữa bạn bè nổi tiếng
  - http://www.manythings.org/hmf/
- Luyện từ vựng tiếng Anh

Đối với người mới lập trình

- Mô-đun hóa chương trình
- Thao tác với xâu ký tự
- Xử lý logic của trò chơi (game logic)
- Vẽ hình đơn giản (text)

Hangman: Luật chơi

- Trò chơi giữa A (chủ trò) và B (người chơi)
- A nghĩ ra một từ tiếng Anh nhưng giấu
  - *secretWord*: Số vạch = số chữ cái trong từ
- B tìm cách đoán ra từ của A
  - Mỗi lần B đoán 1 chữ cái đúng, A ghi chữ cái đó lên các vạch tương ứng
  - Nếu B đoán sai, B mất 1 lượt đoán
- Số lượt ≈ số nét vẽ giá treo và thân người của B (so fun :-D)

Hangman: Luật chơi

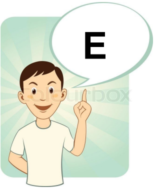
- Sai lần đầu: Vẽ chữ L ngược (giá treo cổ)
- Sai lần 2: Vẽ vòng tròn (đầu)
- Sai lần 3: Vẽ 1 vạch (thân người)
- Sai lần 4: Vẽ 1 vạch (tay trái)
- Sai lần 5: Vẽ 1 vạch (tay phải)
- Sai lần 6: Vẽ 1 vạch (chân trái)
- Sai lần 7: Vẽ 1 vạch (chân phải)

Đủ thân người → thua cuộc

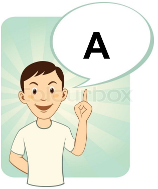
Ví dụ 1 ván chơi


```
 -------------
 |
 |
 |
 |
-----
```

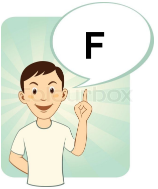
secretWord



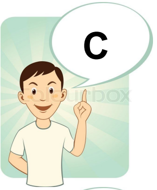

```
 -------------
 |           |
 |
 |
 |
-----
```



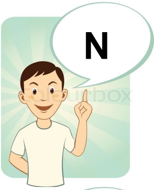
secretWord
-A---A-




```
 -------------
 |           |
 |           o
 |
 |
-----
```




```
 -------------
 |           |
 |           o
 |           |
 |
-----
```

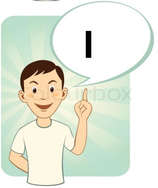


secretWord
-AN--AN


```
 -------------
 |           |
 |           o
 |          /|
 |
-----
```


```
 -------------
 |           |
 |           o
 |          /|\
 |
-----
```

secretWord
HAN--AN




```
 -------------
 |           |
 |           o
 |          /|\
 |          /
-----
```

secretWord
HANG-AN


```
 -------------
 |           |
 |           o
 |          /|\
 |          / \
-----
```

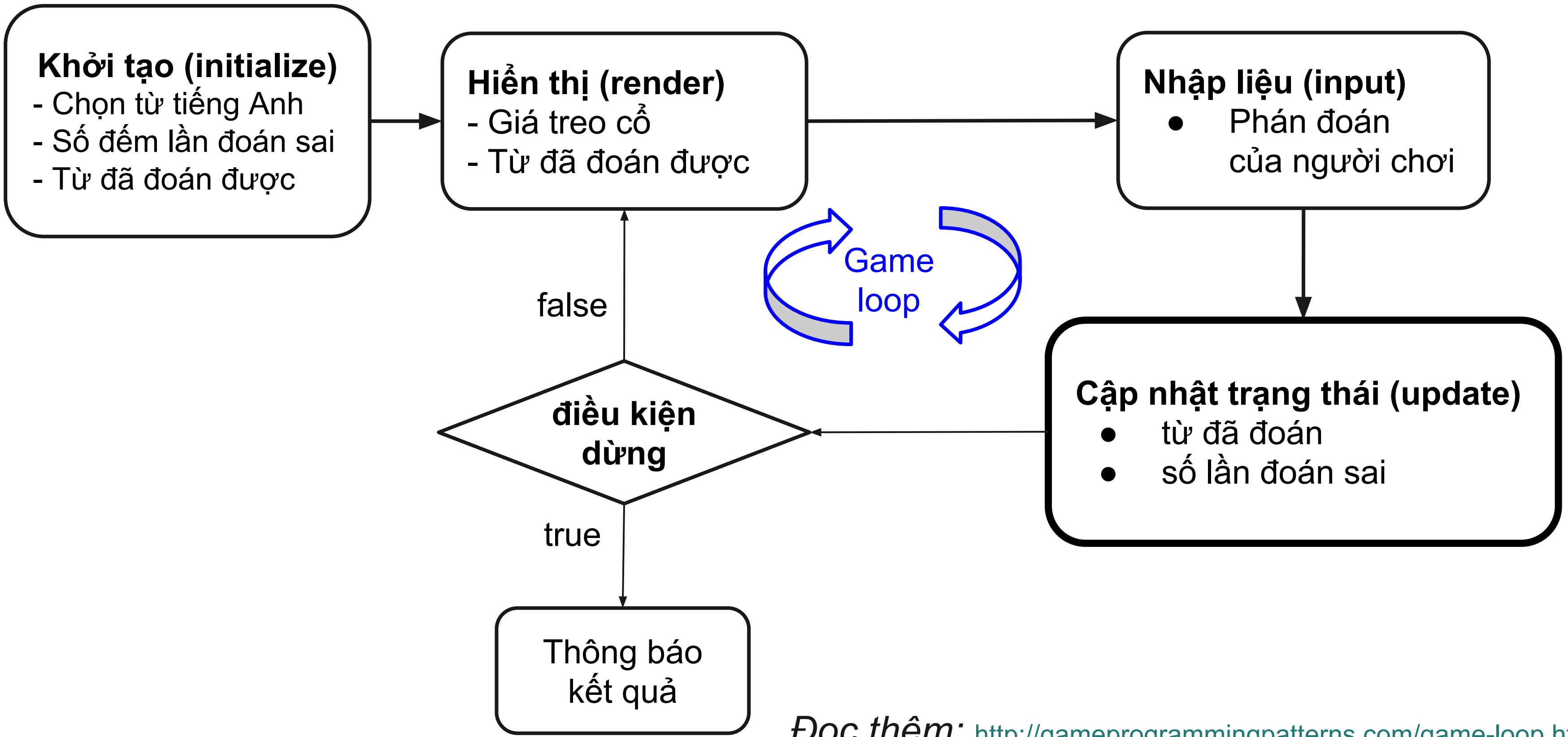
Lập trình trò chơi Hangman

Hãy lập trình trò chơi Hangman với máy là chủ trò

Cần *hình dung các tác vụ* của chương trình trước khi lập trình cụ thể

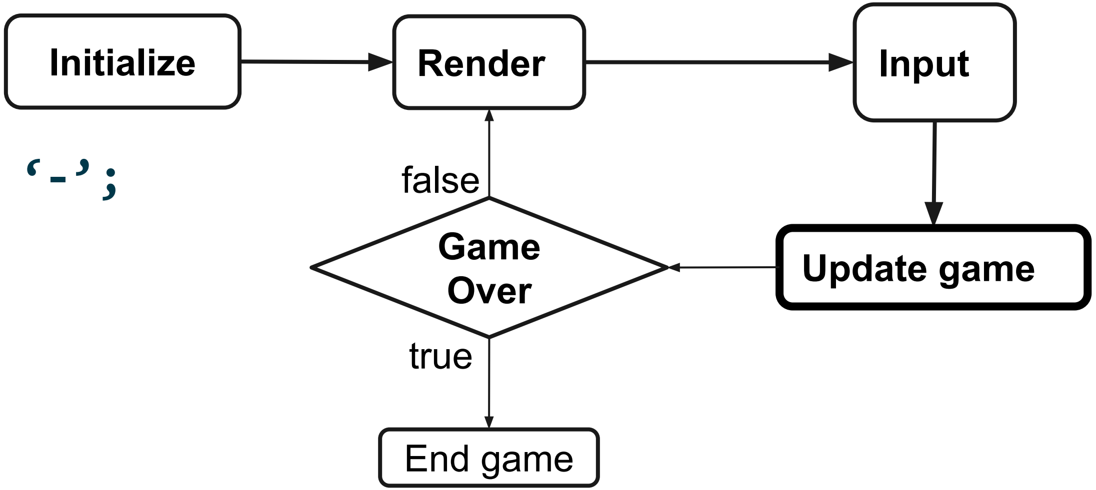
- Khởi tạo: máy nghĩ từ tiếng Anh, số đếm lần đoán sai, đúng
- Nhập liệu: phán đoán của người chơi
- Cập nhật: xử lý phán đoán và thay đổi trạng thái trò chơi
- Hiển thị trạng thái trò chơi: người trên giá treo và secretWord
- Thông báo kết quả trò chơi

Sơ đồ khối - quan hệ giữa các tác vụ



Đọc thêm: http://gameprogrammingpatterns.com/game-loop.html

Mã giả




```
choose word;
initialize guessedWord with ‘-’;
badGuessCount = 0;

do {
    render game;
    char guess = readAGuess;
    if (word contains guess) update guessedWord;
    else badGuessCount++;
} while (game not over);
display game result;
```

```
string word = chooseWord();
string guessedWord = string(word.length(), '-');
int badGuessCount = 0;

do {
    renderGame(guessedWord, badGuessCount);
    char guess = readAGuess();
    if (contains(word, guess))
        guessedWord = update(guessedWord, word, guess);
    else badGuessCount++;
} while (badGuessCount < 7 && word != guessedWord);

renderGame(guessedWord, badGuessCount);
if (badGuessCount < 7) cout << "Congratulations! You win!";
else cout << "You lost. The correct word is " << word;
```

```
string word = chooseWord();
string guessedWord = string(word.length(), '-');
int badGuessCount = 0;

do {
    renderGame(guessedWord, badGuessCount);
    char guess = readAGuess();
    if (contains(word, guess))
        guessedWord = update(guessedWord, word, guess);
    else badGuessCount++;
} while (badGuessCount < 7 && word != guessedWord);

renderGame(guessedWord, badGuessCount);
if (badGuessCount < 7) cout << "Congratulations! You win!";
else cout << "You lost. The correct word is " << word;
```

Các logic đủ đơn giản để đặt tại câu chuyện chính

Chia để trị

- Sơ đồ khối và mã giả
  - Chuyển hóa từ ngôn ngữ đời thường sang ngôn ngữ gần máy hơn
  - Cấu trúc chung của chương trình cơ bản đã rõ
  - Tách các thành phần tương đối độc lập thành hàm
- Xây dựng, cài đặt từng thành phần / hàm
  - Thử nghiệm các kỹ thuật
  - Kiểm tra, chạy thử
  - Ráp nối

Các vấn đề kĩ thuật tồn đọng

- **Choose word**: chọn ra một từ ngẫu nhiên từ đâu?
  - Hardcode? Hơi mất công nếu muốn có nhiều lựa chọn
  - file? Cần học về ra vào dữ liệu với file
- **Render game**: vẽ màn hình game với giá treo cổ như thế nào?
  - Đồ họa? Chưa học thư viện
  - Text? Vẫn mất thì giờ vẽ và chỉnh
- *Quyết định thế nào?*

Làm gì trước?

- *Thử nghiệm các kỹ thuật*
- *Kiểm tra, chạy thử*
- *Ráp nối*

Hai cách tiếp cận:

1. Thử các kĩ thuật trước khi lắp ghép vào chương trình chính
2. Chạy chương trình với phiên bản tối thiểu để test logic trước khi nâng cấp về giao diện, hiệu năng

Kế hoạch

Mục tiêu: nhanh chóng có game chơi được, nâng cấp dần chất lượng

Các phiên bản:

0.1 Phiên bản tối thiểu dùng để test logic chính của game: chooseWord luôn trả về một từ, renderGame hiển thị thông tin tối thiểu đủ chơi

0.2 ChooseWord chọn ngẫu nhiên trong một danh sách hardcode

1.0 RenderGame vẽ được giá treo cổ

2.0 ChooseWord chọn từ trong file *(để các bài sau)*

3.0 RenderGame dùng thư viện đồ họa *(để tự làm sau)*

Phiên bản 0.1

Giao diện tối thiểu

Từ được chọn cố định

Tập trung vào logic chính của game

```
string chooseWord()
{
    return "book";
}

void renderGame(string guessedWord, int badGue
{
    cout << guessedWord << endl;
    cout << "Number of wrong guesses: " << bad
}
```

Number of wrong guesses: 0
Your guess: **a**

Number of wrong guesses: 1
Your guess: **b**
b---
Number of wrong guesses: 1
Your guess: **e**
b---
Number of wrong guesses: 2
Your guess: **o**
boo-
Number of wrong guesses: 2
Your guess: **k**
book
Number of wrong guesses: 2
Congratulations! You win!

Thao tác với từ

- Chương trình cần thao tác và xử lý từ và chuỗi kí tự. Ví dụ:
    - Cần kiểm tra xem “book” có chứa kí tự ‘o’
    - Update(“----”, “book”, ‘o’) cần biến “----” thành “-oo-”
- Các lựa chọn kiểu dữ liệu cho từ:
    - Mảng char
    - Kiểu **string** (*tự tra tài liệu*)
      string ~ Mảng các kí tự + Các hàm tiện ích
- Lựa chọn của ta: string

string

- Khai báo giống các kiểu cơ bản
- Có thể là *kết quả trả về của hàm*
- Có nhiều thao tác xâu kí tự được cài đặt sẵn
- Lập trình viên không phải *lo cấp phát bộ nhớ*

http://www.cplusplus.com/reference/string/string/

```
string greeting = "hello"
string name = "world!";

cout << greeting << " " << name << endl;
cout << "First char: " << greeting[0];

greeting[0] = 'H';
cout << greeting + " " + name << endl;

cout << name.size() << endl;  // 6

size_t pos = name.find("or"); // 1

//sub string starting at pos
string found = name.substr(pos);
cout << found << endl;      //  orld!
```

Xử lý luật chơi (game logic)

Trạng thái trò chơi tại mỗi lượt chơi (lượt đoán):

- **char guess**: phán đoán của người chơi
- **string word**: từ tiếng Anh được máy chọn từ đầu
- **string guessedWord**: các vạch (chữ cái chưa đoán được) và các chữ cái đã đoán được
- **int badGuessCount**: số lần đoán sai

Cần cập nhật **guessedWord, badGuessCount** theo luật chơi, kiểm tra thắng / thua

Xử lý luật chơi (game logic)

- Kiểm tra thắng thua - dễ
  - Thua: `badGuessCount == 7`
  - Đã đoán xong: `word == guessedWord`
  - Chưa đoán xong: `word != guessedWord`

(ở kiểu string, các phép so sánh `==` và `!=` kiểm tra nội dung hai chuỗi kí tự nằm trong hai biến string)

Xử lý luật chơi (game logic)

- Cập nhật **guessedWord, badGuessCount** theo luật chơi. Tiếp tục cách tiếp cận top-down

```
if (contains(word, guess))
    Update //sửa guessedWord
else badGuessCount++;
```

- Hàm update sửa guessedWord để hiện các kí tự đã đoán được

  ‘----’ (‘book’) thành ‘-oo-’ nếu vừa đoán ‘o’

  ‘-oo-’ (‘book’) thành ‘-ook’ nếu vừa đoán ‘k’

update(guessedWord, word, guess)

Đầu vào (tham số):

- **char guess**: phán đoán của người chơi
- **string guessedWord**: các vạch (chữ cái chưa đoán được) và các chữ cái đã đoán được
- **string word**: từ máy chọn từ đầu

Đầu ra: Xâu **guessedWord** mới, hiển thị các vị trí **guess** xuất hiện trong **word**

```
update("-------", "HANGMAN", 'A') trả về "-A---A-"
update("-A---A-", "HANGMAN", 'P') trả về "-A---A-"
update("-A---A-", "HANGMAN", 'H') trả về "HA---A-"
```

update(guessedWord, word, guess)

Duyệt lần lượt các ký tự của **word**: Nếu ký tự đó bằng **guess** thì thay thế vào vị trí tương ứng (cùng chỉ số) trong **guessedWord**

```
string update(string guessedWord, string word, char guess)
{
   for (int i = word.length() - 1; i >= 0; i--) {
        if (word[i] == guess) {
            guessedWord[i] = guess;
        }
    }
    return guessedWord;
}
```

update(guessedWord, word, guess)

Chú ý hàm length() lấy độ dài của string,
cách đọc và ghi giá trị của một ký tự trong string

```
string update(string guessedWord, string word, char guess)
{
    for (int i = word.length() - 1; i >= 0; i--) {
        if (word[i] == guess) {
            guessedWord[i] = guess;
        }
    }
    return guessedWord;
}
```

Hàm contains(word, guess)

Đầu vào (tham số):

- **char guess**: phán đoán của người dùng
- **string word**: từ máy chọn từ đầu

Đầu ra: giá trị kiểu **bool**: **true** nếu **ch** xuất hiện trong **word**, là **false** nếu ngược lại

`contains("HANGMAN",'A')` trả về `true`

`contains("HANGMAN",'P')` trả về `false`

Gợi ý: hàm **`s.find_first_of(c)`** trả về chỉ số của vị trí đầu tiên của c trong string s, trả về hằng số `string::npos` nếu không tìm thấy

Hoàn thành phiên bản 0.1

- Test được luật chơi
- Do chooseWord cố định nên khi chạy ta biết đang đoán từ nào
  - dễ dàng tạo các trường hợp đoán sai/đúng để test badGuessCount và renderGame

- Nên refactor (cải tiến, làm sạch code) trước khi đi tiếp

Dùng hằng để tránh magic number

```
do {
...
} while (badGuessCount < 7 && word != guessed
...
if (badGuessCount < 7) cout << "Congratulations! You win!";
...
```

Magic number:
7 là gì vậy?

```
const int MAX_BAD_GUESSES = 7;
...
} while (badGuessCount < MAX_BAD_GUESSES && word !=
...
if (badGuessCount < MAX_BAD_GUESSES) ...
```

- Dễ hiểu hơn
- Dễ dàng sửa giá trị khi cần

Phiên bản 0.2

Có thể chọn từ ngẫu nhiên từ một danh sách cố định trong code (hardcode)

string chooseWord()

- Danh sách từ vựng lưu trong mảng
- *Chọn từ* ngẫu nhiên ⇔ *Chọn chỉ số* ngẫu nhiên trong mảng đó
- Các kỹ thuật cần thiết
  - Dữ liệu về từ vựng
    http://www.manythings.org/vocabulary/lists/l/
    http://www.manythings.org/vocabulary/lists/l/words.php?f=ogden-picturable (200 từ)
  - Mảng các **string** (*array of string*)
    http://stackoverflow.com/questions/9626722/c-string-array-initialization

string chooseWord()

- Thử với số từ nhỏ

```
const string WORD_LIST[] = {"dog", "cat", "human"};
string chooseWord()
{
    int randomIndex = rand() % 3;
    return WORD_LIST[randomIndex];
}
```

string chooseWord()

- Tổng quát hóa số lượng từ (không thể mỗi lần sửa danh sách lại phải sửa cả số từ)
    - Kĩ thuật tìm số phần tử của mảng

```
const string WORD_LIST[] = {"dog", "cat", "human"};
const int WORD_COUNT = sizeof(WORD_LIST) / sizeof(string);
string chooseWord()
{
    int randomIndex = rand() % WORD_COUNT;
    return WORD_LIST[randomIndex];
}
```

Kích thước mảng / kích thước một phần tử

string chooseWord() : 200 từ

- Thay danh sách từ
- Đã có srand()?
- Đến đây ta đã cho chương trình *chọn ngẫu nhiên 1 từ*

- Hoàn thành phiên bản 0.2

```
string WORD_LIST[] = {
        "angle", "ant", "apple", "arch", "arm", "army",
        "baby", "bag", "ball", "band", "basin", "basket", "bath", "bed", "bee", "bell", "berry",
        "bird", "blade", "board", "boat", "bone", "book", "boot", "bottle", "box", "boy",
        "brain", "brake", "branch", "brick", "bridge", "brush", "bucket", "bulb", "button",
        "cake", "camera", "card",  "cart", "carriage", "cat", "chain", "cheese", "chest",
        "chin", "church", "circle", "clock", "cloud", "coat", "collar", "comb", "cord",
        "cow", "cup", "curtain", "cushion",
        "dog", "door", "drain", "drawer", "dress", "drop", "ear", "egg", "engine", "eye",
        "face", "farm", "feather", "finger", "fish", "flag", "floor", "fly",
        "foot", "fork", "fowl", "frame", "garden", "girl", "glove", "goat", "gun",
        "hair", "hammer", "hand", "hat", "head", "heart", "hook", "horn", "horse",
        "hospital", "house", "island", "jewel",  "kettle", "key", "knee", "knife", "knot",
        "leaf", "leg", "library", "line", "lip", "lock",
        "map", "match", "monkey", "moon", "mouth", "muscle",
        "nail", "neck", "needle", "nerve", "net", "nose", "nut",
        "office", "orange", "oven", "parcel", "pen", "pencil", "picture", "pig", "pin",
        "pipe", "plane", "plate", "plow", "pocket", "pot", "potato", "prison", "pump",
        "rail", "rat", "receipt", "ring", "rod", "roof", "root",
        "sail", "school", "scissors", "screw", "seed", "sheep", "shelf", "ship", "shirt",
        "shoe", "skin", "skirt", "snake", "sock", "spade", "sponge", "spoon", "spring",
        "square", "stamp", "star", "station", "stem", "stick", "stocking", "stomach",
        "store", "street", "sun", "table", "tail", "thread", "throat", "thumb", "ticket",
        "toe", "tongue", "tooth", "town", "train", "tray", "tree", "trousers", "umbrella",
        "wall", "watch", "wheel", "whip", "whistle", "window", "wire", "wing", "worm",
    };
```

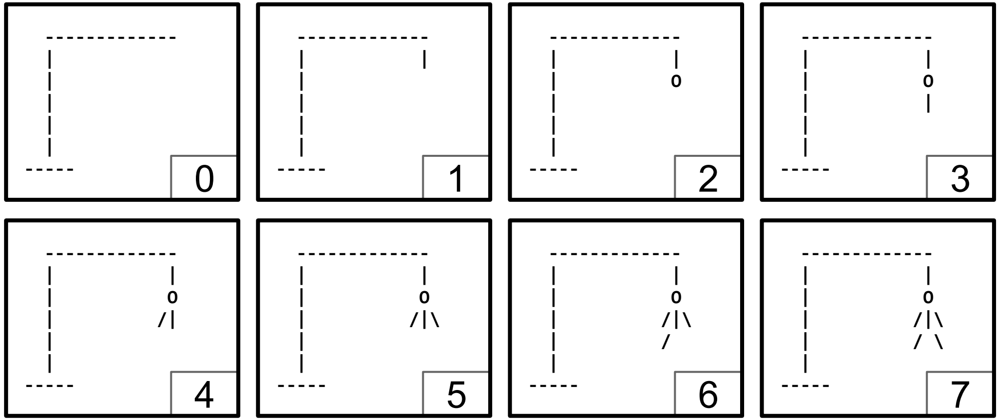
Phiên bản 1.0

Vẽ giá treo cổ bằng text

```
  --------------
  |            |
  |            O
  |           /|\
  |           / \
  |
-----
```

Hiển thị giá treo cổ

- Bản chất là 1 đoạn văn bản có nhiều dòng
  - 1 Hình vẽ ⇔ 1 **string** (xuống dòng bằng ký tự **\n**)
- Nếu lưu các hình vẽ trong mảng **string**
  - **badGuessCount** tương ứng với *chỉ số mảng*



renderGame()

- Luôn bắt đầu từ đơn giản để chạy thử

```
const string FIGURE[] = {
  "fig0", "fig1", "fig2", "fig3", "fig4", "fig5", "fig6", "fig7"
    };
void renderGame(string guessedWord, int badGuessCount)
{
    cout << FIGURE[badGuessCount] << endl;
    cout << guessedWord << endl;
    cout << "Number of wrong guesses: " << badGuessCount << endl;
}
```

renderGame

- Sau đó đưa hình vẽ thật vào biến **figure**

```
const string FIGURE[] = {
        "   -------------    \n"
        "   |                \n"
        "   |                \n"
        "   |                \n"
        "   |                \n"
        "   |     \n"
        " -----   \n",

        "   -------------    \n"
        "   |           |    \n"
        "   |                \n"
        "   |                \n"
        "   |                \n"
        "   |     \n"
        " -----   \n",

        "   -------------    \n"
        "   |           |    \n"
        "   |           O    \n"
        "   |                \n"
        "   |                \n"
        "   |     \n"
        " -----   \n",

        "   -------------    \n"
        "   |           |    \n"
        "   |           O    \n"
        "   |           |    \n"
        "   |                \n"
        "   |     \n"
        " -----   \n",

        "   -------------    \n"
        "   |           |    \n"
        "   |           O    \n"
        "   |          /|    \n"
        "   |                \n"
        "   |     \n"
        " -----   \n",

        "   -------------    \n"
        "   |           |    \n"
        "   |           O    \n"
        "   |          /|\\  \n"
        "   |                \n"
        "   |     \n"
        " -----   \n",

        "   -------------    \n"
        "   |           |    \n"
        "   |           O    \n"
        "   |          /|\\  \n"
        "   |          /     \n"
        "   |     \n"
        " -----   \n",

        "   -------------    \n"
        "   |           |    \n"
        "   |           O    \n"
        "   |          /|\\  \n"
        "   |          / \\  \n"
        "   |     \n"
        " -----   \n",
};
```

Ta đã hoàn thành phiên bản 1.0

Tổng kết

- Viết chương trình
  - Bắt đầu đơn giản
  - Dần dần thay đổi theo yêu cầu
  - Luôn có chương trình chạy được
- Sơ đồ khối, mã giả, tư tưởng chia để trị
- string, thao tác với xâu kí tự
  - Khởi tạo xâu, duyệt ký tự, nối/cộng xâu
- Dùng hằng
  - thay vì magic number
  - cho các giá trị không đổi