

# Game: Guess It

2 - Hàm

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

# Nội dung

- **Game: Đoán số (Guess It)**
- Chuyển hoá vấn đề thành chương trình
- Kỹ thuật:
  - Sinh số ngẫu nhiên
  - Vòng lặp, điều kiện vòng lặp
  - Mô-đun hóa chương trình bằng hàm

# Đoán số: Luật chơi

- Hai người: chủ trò - A, người chơi - B.
- Người A chọn số bất kỳ từ 1-100
- Người B đoán con số này
  - Nếu đúng, người B thắng.
  - Nếu sai, người A sẽ trả lời con số người B đoán là lớn hơn hay nhỏ hơn. Người B tiếp tục đoán số.
- Cho trẻ em học Toán
- Giúp hiểu thuật toán quan trọng:  
*Tìm kiếm nhị phân (Binary Search)*



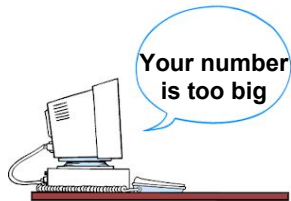
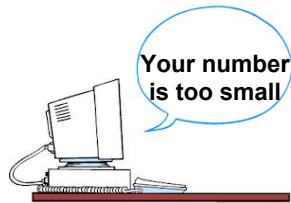
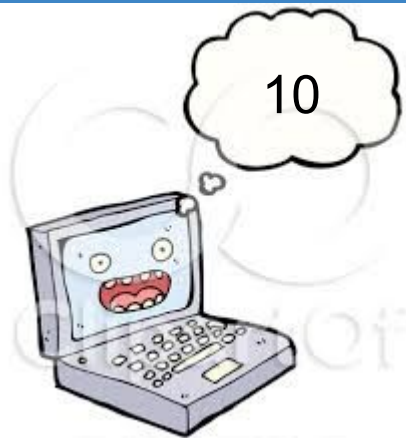
GUESS IT

# Đoán số: Chương trình

- Giữa người (B) và máy (chủ trò - A).
- Máy "nghĩ" ra một con số từ 1 đến 100
- Người chơi đoán con số này
  - Nếu đúng, người chơi thắng cuộc. Nếu sai, máy sẽ trả lời con số người chơi đoán lớn hơn hay nhỏ hơn con số của máy để người chơi tiếp tục đoán số
- Cách khác: người (chủ trò - A) và máy (đoán - B).



# Ví dụ một lần chơi



GUESS IT

# Nội dung

- Game: Đoán số (Guess It)
- **Chuyển hoá thành chương trình**
- Kỹ thuật:
  - Sinh số ngẫu nhiên
  - Vòng lặp, điều kiện vòng lặp
  - Mô-đun hóa chương trình bằng hàm

# Mô tả thành các bước (bằng lời)

- Máy tính nghĩ số
- Nhập con số người chơi đoán
- Máy chọn câu trả lời phù hợp
- Lặp lại nếu người chơi chưa đoán đúng

# Mô tả thành các bước (gần máy)

- B1: Máy tính nghĩ số
- B2: Nhập con số người chơi đoán
- B3: Máy chọn câu trả lời phù hợp.
- B4:  
Nếu người chơi đoán sai. Quay lại B2.  
Nếu người chơi đoán đúng. Chuyển tới B5.
- B5: Kết thúc.



# Chương trình (mã giả, gần máy)

```
randomNumber = generateRandomNumber(); // B1
while (true) {
    number = getPlayerGuess(); // B2
    printAnswer(number, randomNumber); // B3
    if (number == randomNumber) break; // B4
    // else continue;
}
// B5
```

# Chương trình (ngôn ngữ C++)

```
int randomNumber = generateRandomNumber();  
do {  
    int number = getPlayerGuess();  
    printAnswer(number, randomNumber);  
} while (number != randomNumber);
```

# Tách hàm (mô-đun hóa)

Tìm cách đặt tên cho từng bước

- Nghĩ số: **int generateRandomNumber()**
- Nhập con số người dùng đoán:  
**int getPlayerGuess()**
- In câu trả lời phù hợp:  
**void printAnswer(int number, int randomNumber)**

*Viết chương trình như kể một câu chuyện*

- Tên biến = cụm danh từ
- Tên hàm = cụm động từ

# Sao lại tiếng Anh ? Khó thể :(

- Tiếng Anh là ngôn ngữ của Công nghệ thông tin (IT - Information Technology):
  - Từ khoá ngôn ngữ lập trình
  - Tài liệu, sách vở tiếng Anh nhiều
  - Tìm kiếm trên Internet; Trao đổi với người các nước (không chỉ Anh, Mỹ, Úc)
- Viết phần mềm cho thế giới: out source hay đưa “app” của mình lên Internet.
- Học thêm tiếng Anh

# Guess It 1.1

```
#include <iostream>
#include <cstdlib>

using namespace std;

int generateRandomNumber();
int getPlayerGuess();
void printAnswer(int number, int randomNumber);

int main()
{
    int randomNumber = generateRandomNumber();
    int number;

    do {
        number = getPlayerGuess();
        printAnswer(number, randomNumber);
    } while (number != randomNumber);

    return 0;
}
```

# Nội dung

- Game: Đoán số (Guess It)
- Chuyển hoá thành chương trình
- **Kỹ thuật:**
  - Sinh số ngẫu nhiên
  - Vòng lặp, điều kiện vòng lặp
  - Mô-đun hóa chương trình bằng hàm

# Máy tính nghĩ số

- Máy tính không thể thật sự “ngẫu nhiên”
- Sinh số “giả ngẫu nhiên” - pseudo random
  - Tìm kiếm Google: “C++ random”
  - Hàm **rand()** trong **<cstdlib>**
  - Hằng RAND\_MAX



```
v1 = rand() % 100;           // v1 in the range 0 to 99  
v2 = rand() % 100 + 1;      // v2 in the range 1 to 100  
v3 = rand() % 30 + 1985;    // v3 in the range 1985-2014
```

```
int randomNumber = rand() % 100 + 1;
```



# Guess It 1.1

```
using namespace std;

int generateRandomNumber();
int getPlayerGuess();
void printAnswer(int number, int randomNumber);

int main()
{
    int randomNumber = generateRandomNumber();
    int number;

    do {
        number = getPlayerGuess();
        printAnswer(number, randomNumber);
    } while (number != randomNumber);
}
```

```
int generateRandomNumber()
{
    return rand() % 100 + 1;
}
```



# Nhập con số người chơi đoán

- Quá dễ

```
int number;  
cout << endl << "Enter your number: ";  
cin >> number;
```

# Guess It 1.1

```
int main()
{
    int randomNumber = generateRandomNumber();
    int number;

    do {
        number = getPlayerGuess();
        printAnswer(number, randomNumber);
    } while (number != randomNumber);

    return 0;
}
```

```
int getPlayerGuess()
{
    int number;
    cout << endl << "Enter your number between 1 and 100: ";
    cin >> number;
    return number;
}
```

# Máy tính chọn câu trả lời

- Lựa chọn bằng **if ... else if ... else** liên tiếp

```
if (number > randomNumber) {  
    cout << "Your number is too big." << endl;  
} else if (number < randomNumber) {  
    cout << "Your number is too small." << endl;  
} else {  
    cout << "Congratulation! You win." << endl;  
}
```

# Guess It 1.1

```
int main()
{
    int randomNumber = generateRandomNumber();
    int number;

    do {
        number = getPlayerGuess();
        printAnswer(number, randomNumber);
    } while (number != randomNumber);
}
```

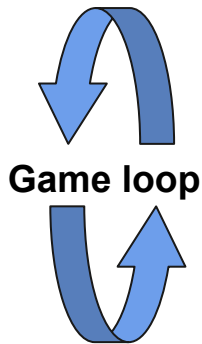
```
void printAnswer(int number, int randomNumber)
{
    if (number > randomNumber) {
        cout << "Your number is too big." << endl;
    } else if (number < randomNumber) {
        cout << "Your number is too small." << endl;
    } else {
        cout << "Congratulation! You win." << endl;
    }
}
```

# Guess It 1.1

```
void printAnswer(int number, int randomNumber)
{
    if (number > randomNumber) {
        cout << "Your number is too big." << endl;
    } else if (number < randomNumber) {
        cout << "Your number is too small." << endl;
    } else {
        cout << "Congratulation! You win." << endl;
    }
}
```

# Lặp lại (Game loop)

- Nếu người chơi đoán sai, lặp lại bước nhập
- Cần hỏi người chơi ít nhất 1 lần
- Vòng lặp **do ... while**



```
do {  
    // Nhập số người chơi đoán  
    // In câu trả lời phù hợp  
} while (number != randomNumber);
```

# Kết quả

```
C:\software\cygwin64\home\doe\advprogram\lec2-guessit\GuessIt.exe
```

```
Enter your number between 1 and 100: 50
```

```
Your number is too big.
```

```
Enter your number between 1 and 100: 25
```

```
Your number is too small.
```

```
Enter your number between 1 and 100: 42
```

```
Congratulation! You win.
```

# Thực hành

- Hiển thị số lần đoán của người chơi
- Điểm của người chơi =  $100 - \text{số lần đoán}$
- Điểm của người chơi =  
 $10000 - 2 \times 100 - 2 \times 99 - \dots$   
tùy theo số lần người chơi đoán
- Cho phép chơi nhiều ván
  - Hỏi người chơi có muốn chơi tiếp không ?
  - Gợi ý: đưa toàn bộ mã trong hàm **main()** vào một hàm **playGuessIt()**



# Cho phép chơi nhiều ván

- Bạn có nhận ra mỗi lần chạy chương trình, máy “nghĩ” lại cùng một con số
- Tìm kiếm Google: “C++ random repeat”
- Câu lệnh: **srand(time(0));**
- Khởi tạo “hạt giống” cho hàm **rand()** bằng thời gian bắt đầu chạy chương trình
  - Lưu ý: chỉ cần gọi **srand()** một lần.
- Mỗi lần chạy, chương trình dùng một hạt giống khác nhau : thời gian hiện hành

# Tổng kết

- *Viết chương trình như kể một câu chuyện*
- Chia nhỏ các bước thành hàm và lệnh
  - Mô-đun hóa bằng hàm
  - Truyền tham số bằng giá trị
- Tìm kiếm, tra cứu kỹ thuật lập trình
  - Không thể thiếu tiếng Anh
- Sinh số (giả) ngẫu nhiên
- Lựa chọn bằng **if ... else if ... else**
- Vòng lặp **do ... while**

# Máy chơi Guess It

- Đặt vấn đề: đảo vai trò người và máy
  - Người làm chủ trò, nghĩ số từ 1 đến 100
  - Máy đoán số
  - Người “thông báo” cho máy giá trị máy đoán lớn hơn, nhỏ hơn hay đúng bằng giá trị cần tìm
- Có nhiều cách chơi (thuật toán đoán)
  - Đoán ngẫu nhiên
  - Đoán tuần tự từ 1 đến 100 (hoặc ngược lại)
  - Đoán “đại” một số, nhận trả lời của người chơi để phán đoán lần sau nên đoán thế nào

# Thuật toán chung

B1: select a number  $X$  in  $[1, 100]$

B2: ask for host's answer on  $X$

B3: if  $X$  is right, exit (win)

else goto B1

# Thuật toán chung

```
int X, answer;  
do {  
    X = selectNumber(1, 100);  
    answer = getHostAnswer(X);  
    if (answer == '=')  
        cout << "Your number is " << X << endl;  
} while (answer != '=');
```

tùy thuộc vào hàm `selectNumber()` mà ta có các cách đoán (thuật toán đoán) khác nhau

```
char getHostAnswer(int X)  
{  
    char answer;  
    cout << "Is " << X  
        << " your number? ";  
    cin >> answer;  
    return answer;  
}
```

Quy ước trả lời:

answer có thể là >, <, =

>: X lớn hơn số cần đoán

<: X nhỏ hơn số cần đoán

=: X bằng số cần đoán

# Đoán ngẫu nhiên (may rủi)

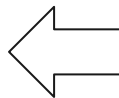
```
int selectNumber(int low, int high)
{
    return rand() % (high-low+1) + low;
}
```

Rõ ràng không thể biết lúc nào thuật toán này đoán được số cần tìm

# Tìm kiếm tuần tự (chắc ăn)

```
int selectNumber(int low, int high)
{
    return low;
}
...
int X, answer, low = 1;
do {
    X = selectNumber(low, 100);
    answer = getHostAnswer(X);
    if (answer != '=') low++;
} while (answer != '=');
```

- Lần lượt đoán các số từ 1 đến 100
- Thuật toán chắc chắn đoán ra số cần tìm
- Nếu số cần đoán là 100 thì cần 100 lần đoán



Nhích cận dưới  
của lần đoán sau  
lên 1 đơn vị

# Đoán đại rồi chỉnh khoảng tin cậy

```
int selectNumber(int low, int high)
{
    return rand() % (high-low+1) + low;
}
```

...

```
int X, answer, low = 1, high = 100;
```

```
do {
    X = selectNumber(low, high);
    answer = getHostAnswer(X);
    if (answer == '>') high = X-1; // X lớn hơn nên giảm high
    if (answer == '<') low = X+1; // X nhỏ hơn nên tăng low
} while (answer != '=');
```

- Tuy may rủi nhưng chắc chắn đoán ra số cần tìm (*tại sao*)
- Nếu số X chỉ tăng hoặc giảm 1 đơn vị so với lần lặp trước → *giống tìm kiếm tuần tự* (không may)

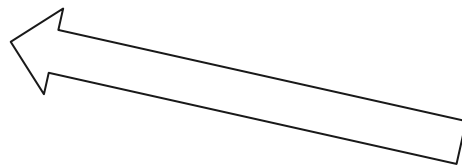


# Cải tiến khoảng tin cậy

- Thuật toán chia đôi (tìm kiếm nhị phân)
  - Chọn số **X** là điểm giữa khoảng **[low, high]**
  - Mỗi lần đoán (sai), kích thước khoảng tin cậy giảm ít nhất 1 nửa
  - Số lần đoán tối đa  $\approx \log_2 100 = 7$

# Thuật toán chia đôi

```
int selectNumber(int low, int high)
{
    return (low+high) / 2;
}
...
```



```
int X, answer, low = 1, high = 100;
```

```
do {
    X = selectNumber(low, high);
    answer = getHostAnswer(X);
    if (answer == '>') high = X-1; // X lớn hơn nên giảm high
    if (answer == '<') low  = X+1; // X nhỏ hơn nên tăng low
} while (answer != '=');
```

chắc chắn giảm kích thước  
khoảng tin cậy ít nhất một nửa

