BÁO CÁO BÀI TẬP "TRÒ CHƠI: ĐOÁN TÊN CON VẬT"

Nguyễn Tấn Phát

20127588 – CLC10

1. Mô tả trò chơi.

- Người chơi, sẽ nghĩ trong đầu một con vật và máy tính sẽ có nhiệm vụ đoán tên con vật ấy là gì.
- Sau đó máy tính sẽ liên tục đưa ra những câu hỏi mà người chơi cần trả lời YES hay NO liên quan đến con vật này.
- Sau cùng máy tính sẽ đưa ra kết luận rằng đó là con vật nào.
    - Nếu đúng thì máy tính đã thắng (vì đã đoán đúng)
    - Nếu sai người chơi sẽ đưa ra câu trả lời tên con vật là gì và điểm khác nhau giữa con vật này và con vật mà máy đưa ra.

2. Hướng làm

- Sử dụng kiểu cấu trúc dữ liệu tương tự với Cây nhị phân(Binary Search Tree). Với node bên trái là sự lựa chọn CÓ(yes) và node bên phải là sự lựa chọn KHÔNG(no).
- Dữ liệu ban đầu sẽ được lấy từ file text – mỗi dòng của text sẽ là một node. Các node lá sẽ được đánh dấu là "#A", còn cái node còn lại được đánh dấu là "#Q". Từ đó ta load dữ liệu của file text vào Cây nhị phân (Binary Search Tree)
- Sau đó cho người chơi dưới dạng câu hỏi Yes, no – Question.
- Nếu kết quả ra sai (No) thì sẽ cho người chơi mở rộng thêm câu hỏi và câu trả lời bằng cách sửa node đó và thêm vào 2 con của nó 2 câu trả lời(answer).
- Cấu trúc dữ liệu một node bao gồm:
    - String data => Dữ liệu của 1 node (question or answer)
    - Bool isFullChild => Mục đích để kiểm tra xem node đã full chưa.
    - Node* left => Node trái, trả về khi đưa ra câu trả lời đúng(yes)
    - Node* right => Node phải, trả về khi đưa ra câu trả lời sai(no)

3. Hàm – Function:

```
bool canAddNode(Node* node)
```
-> Kiểm tra xem node đó có thể chèn vào nữa hay không

```
Node* add(Node* root, string s, bool& isAdded)
```

Add một node vào với dữ liệu là string s; biến Added kiểm tra xem đã được Add chưa.

Kiểm tra xem node truyền vào(root) có là NULL ➔ nếu là NULL thì khởi tạo root = new node{s, true, NULL, NULL}. Rồi sau đó kiểm tra xem (s) có phải là #Q – Question không. Nếu phải thì nó chưa full nên isFullChild = false. Rồi trả về và kết thúc hàm.

Kiểm tra nó fullChild chưa, nếu rồi thì trả về và kết thúc hàm.

Kiểm tra (root->left) có add được nữa hay không, nếu được thì đệ quy

```
root->left = add(root->left, s, isAdded);
```

Kiểm tra node(root) có thể add nữa được không, nếu không và node phải(root->right) vẫn có thể add thì ta sẽ đệ quy add vào node phải.

```
root->right = add(root->right, s, isAdded);
```

Nếu một node(root) không thể mà (root->left) và (root->right) đều không thể add thì node ấy sẽ có giá trị của isFullChild = true;

```
int Menu()
```

Hiển thị menu cho người dùng thấy.

Người dùng sẽ nhập sự lựa chọn của mình (1 – Start; 2 – Save; 3 – Exit)

Hàm sẽ trả về số mà người dùng lựa chọn.

```
Node* Play(Node* root, string& s)
```

Đầu tiên sẽ clear tất cả nhưng gì trên màn hình và bắt đầu trò chơi.

Temp = root; pre = temp;

Mục đích của node (pre) là để lưu lại node trước đó của node(temp)

Bắt đầu vòng lặp – vòng lặp sẽ không kết thúc nếu chưa về đến node rỗng (temp != NULL)

Máy đưa ra câu hỏi (yes/no) và người dùng sẽ nhập vào (y/n)

Nếu đúng(yes) thì temp = temp->left

Nếu không phải(no) thì temp = temp->right

Và cứ tiếp tục vòng lặp như vậy.

Hàm sẽ trả về node cuối cùng (lá) – Answer và lưu lại câu trả lời của người dùng vào (s) để kiểm tra xem máy đã đoán đúng hay sai.

```
void Expand(Node*& t)
```

Hàm sẽ được dùng nếu câu trả lời cuối cùng của người chơi vào (s) là n(no)

Máy sẽ hỏi "Con vật bạn nghĩ là gì?" rồi yêu cầu người chơi nhập vào. Ex "Hổ"

Máy sẽ hỏi "Điểm khác biệt giữa con vật của bạn và con vật của máy là gì" rồi yêu cầu người chơi nhập vào. Ex "Chúa sơn lâm"

Máy sẽ hỏi "Đặc điểm đó có phải của con vật của bạn không". Ex "Chúa sơn lâm có phải là đặc điểm của Hổ không?". Rồi yêu cầu người chơi nhập yes/no

Việc đầu tiên thì phải có được node mà hàm Play() trả về. Lưu lại data của nó vào một biến.

Thay data của node đó bằng điểm khác biệt mà người chơi đã nhập vào.

Nếu đặc điểm đó là của con vật mà người chơi nghĩ:

Node->left: Sẽ là con vật của người chơi nghĩ

Node->right: Sẽ là con vật mà máy đã đoán (sai)

Nếu không phải thì làm ngược lại.

```
void SaveGame(Node* root, ofstream &out)
```

In cây ra file theo cách duyệt tiền thứ tự

```
void SaveGame(Node* root, ofstream &out) {
    if (root != NULL) {
        out << root->data << endl;
        SaveGame(root->left, out);
        SaveGame(root->right, out);
    }
}
```

```
void Start(Node* &root, string &s)
```

```
void Start(Node* &root, string &s) {
    bool Check = true;
    int num;
    Node* temp = root;
    while (Check) {
        num = Menu();
        system("pause");
        if (num == 1) {
            temp = Play(root, s);
            if (s == "n")
                Expand(temp);
                system("pause");
//              #Start(root, s);
        }
        if (num == 2) {
            ofstream out;
            out.open("input.txt");
                SaveGame(root, out);
            out.close();
            system("pause");
        }
        if (num == 3) {
            Check = false;
        }
```

Biến Check để kết thúc vòng lặp khi người dùng muốn Exit (dừng trò chơi)

Biến num để lưu lại câu trả lời của người chơi ở hàm menu

nếu num = 1 – Bắt đầu trò chơi

Gọi hàm Play() và lưu node cuối vào temp

Nếu đoán sai (s = "n") thì gọi hàm mở rộngg Expand(temp)

Nếu num = 2 – Lưu

Thì mở file input.txt và xóa hết dữ liệu cũ

Và ghi mới lên đó bằng hàm SaveGame()

Nếu num – 3 - Thoát

Biến check ban đầu thành false

Rồi thoát khỏi vòng lặp