

# Mystery Library Puzzle Game

## نظرة عامة

اللاعب بيننتقل من كتاب لكتاب، يحل ألغاز، لحد ما يوصل لكتاب النهائي **Linked Lists**. لعبه ألغاز مبنية على

## 🎮 كيف اللعبة بتشتعل؟

### 1 البداية

- اللاعب بيختار واحدة من 4 بوابات دخول
- كل بوابة عندها كتاب بلغز واحد

### 2 المسار

- بعد كل كتاب، اللاعب بيننتقل لكتاب التالي
- عندما مسارين - اللاعب يختار: (EASY) الكتب السهلة
- عندما مسار واحد بس: (HARD) الكتب الصعبة

### 3 الألغاز

- كل كتاب عنده ألغاز لازم تحل
- عندك 3 محاولات لكل لغز
- لو فشلت = Game Over

### 4 النهاية

- لما توصل لكتاب النهائي وتحل اللغز = تفوز 

## 📁 بنية المشروع

Mystery-Library/

```
|  
|--- BookNode.h      #تعريف الكتاب (Node)  
|--- BookNode.cpp    #تنفيذ دوال الكتاب  
|  
|--- Utils.h        #دوال مساعدة  
|--- Utils.cpp       #معالجة النصوص والألغاز
```

```

├── Library.h      تعریف المکتبة #
├── Library.cpp    بناء وربط الكتب #

|
├── Game.h         تعریف اللعبة #
├── Game.cpp       منطق اللعبة #

|
├── main.cpp       نقطه البداية #

|
└── data/
    └── puzzles.txt  ملف الألغاز #

```

## 🔍 شرح كل ملف

### 📘 BookNode (الكتاب)

كل كتاب عنده Node في الـ Linked List. هو الـ

```

cpp

struct BookNode {
    int id;           رقم الكتاب // نوعه (مدخل/وسطي/نهائي) //
    BookType type;   نوعه (مدخل/وسطي/نهائي) //
    BookDifficulty difficulty; // صعوبته (سهل/صعب)

    Clue* clues;    الألغاز //
    int clueCount;  عددهم //

    BookNode* next1; // المسار الأول
    BookNode* next2; // المسار الثاني (للسهل فقط)

    bool visited;   زرناه؟ //
    bool puzzleSolved; // حلينا الألغازه؟ //

};


```

### الدوال المهمة:

- `setupClues(count)` - تجهيز مكان للألغاز
- `addClue(index, problem, solution)` - إضافة لغز
- `(hasTwoPaths()` - هل عندنا مسارين؟

## Utils (الدوال المساعدة)

:دوال علشان نسهل الشغل

```
cpp
معالجة النصوص //
string cleanInput(string text); تضييف وتصغير الحروف //

تحميل الألغاز //
vector<Puzzle> loadPuzzlesFromFile(string filename);

أرقام عشوائية //
int randomNumber(int min, int max);
```

## Library (المكتبة)

المكتبة بتعمل كل الشغل الثقيل:

```
cpp
class Library {
private:
    BookNode* entranceBooks[4];    // 4 كتب مدخل
    vector<BookNode*> middleBooks; // 10-15 كتاب وسطي
    BookNode* finalBook;           // الكتاب النهائي

public:
    Library(vector<Puzzle> puzzles); // بناء المكتبة
    BookNode** getEntranceBooks();   // جلب المدخل
};
```

كيف بتشتفق:

1. `createEntranceBooks()` - تنشئ 4 كتب مدخل
2. `createMiddleBooks()` - تنشئ 10-15 كتاب وسطي
3. `createFinalBook()` - تنشئ الكتاب النهائي
4. `linkAllBooks()` - تربط كل الكتب ببعضها

:الربط

- المدخل → كتب وسطية عشوائية
- الكتب الوسطية → كتب وسطية تانية أو الكتاب النهائي

- آخر 3 كتب → الكتاب النهائي
- 

## Game (اللعبة)

دي اللي بتدير اللعبة:

```
cpp

class Game {
private:
    Library* library;
    BookNode* currentBook;

public:
    void start(); // تشغيل اللعبة
};
```

## Flow (اللعبة):

```
showIntro()      عرض المقدمة // ↓
chooseEntrance() اختيار بوابة // ↓
while (currentBook != null) {
    solvePuzzles()   حل الألغاز // ↓
    if (FINAL)       وصلنا للنهاية؟ //
        WIN!
    else
        chooseNextPath() // نكمل
}
```

## puzzles.txt (ملف الألغاز)

الشكل:

[Category Name]

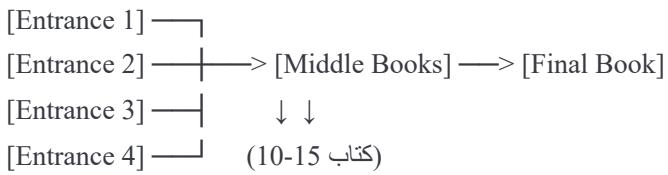
Riddle: What has keys but no locks? | Answer: keyboard

Riddle: I have cities but no houses. What am I? | Answer: map

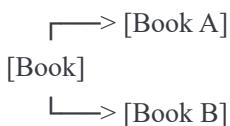
[Math]

Riddle: What is  $2 + 2$ ? | Answer: 4

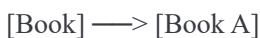
## Linked List Structure



الكتب السهلة (EASY):



الكتب الصعبة (HARD):



### نصائح لفهم

1. ابدأ من main.cpp

- الأساسي flow شوف الـ
- تحميل الغاز → بناء مكتبة → تشغيل لعبة

2. افهم BookNode

- الأساسي building block هو الـ
- كل كتاب = Node في Linked List

3. شوف Library::linkAllBooks()

- هنا بنربط الكتب بعضها
- هنا السحر بيحصل □

4. اتبع Game::start()

- هنا اللعبة الفعلية
- بسيط: حل → انتقل → حل → انتقل

## ⚙️ Compilation

```
bash  
g++ -o game main.cpp BookNode.cpp Library.cpp Game.cpp Utils.cpp  
./game
```

أو باستخدام Makefile:

```
bash  
make  
./game
```

## 🎓 مفاهيم مهمة تعلمتها

### ✓ Linked Lists

- كل كتاب عنده مؤشرات للكتب التالية
- `next1` و `next2` هم الـ links

### ✓ Dynamic Memory

- استخدام `new` و `delete`
- إدارة الذاكرة في Constructor و Destructor

### ✓ Pointers

- `BookNode*` = مؤشر لكتاب
- `BookNode**` = مؤشر لمصفوفة مؤشرات

### ✓ Enums

- `BookType` و `BookDifficulty`
- بدل استخدام strings

### ✓ Structs

- `BookNode` و `Clue` و `Puzzle`

- تنظيم البيانات
- 

## Common Issues

مشكلة: "Can't open puzzles.txt"

الحل: تأكد إن الملف موجود في `(data/puzzles.txt)`

مشكلة: Memory Leaks

الحل: المدمر بيمسح كل الكتب تلقائياً `(~Library())`

مشكلة: Invalid input

الحل: في كل مكان `(cin.fail())` checks

---

## تحسينات ممكنة

- إضافة نظام Hints
  - للألغاز Timer
  - حفظ التقدم
  - Difficulty levels مختلفة
  - Sound effects
  - رسومات ASCII art
- 

## الخلاصة

المشروع ده عبارة عن:

1. Linked List من الكتب

2. كل كتاب عنده أغاز

3. اللاعب ينتقل من كتاب لكتاب

4. يحل الألغاز

5. لحد ما يفوز أو يخسر

عندما تعليقات توضح إيه اللي بيحصل الكود مكتوب بشكل بسيط وواضح للمبتدئين. كل

---

