دليل شامل - BookNode شرح ملفات

🎯 الفكرة العامة

وكل كتاب ممكن، (Linked List) في قائمة مترابطة (Node) الملفات دي بتمثل نظام كتب في لعبة أو تطبيق تفاعلي. كل كتاب عبارة عن نقطة يودي لكتاب تاني أو اتنين حسب صعوبته.

📂 ملف BookNode.h (ملف الهيدر)

1 الأنواع الأساسية (Enums)

BookType - نوع الكتاب

```cpp
enum BookType {
    ENTRANCE,      // كتاب البداية - أول كتاب يبدأ منه اللاعب
    INTERMEDIATE,  // كتاب وسطي - كتاب عادي في النص
    FINAL          // الكتاب الأخير - نهاية المغامرة
};
```

الاستخدام: يحدد مكان الكتاب في السلسلة (بداية، وسط، نهاية).

BookDifficulty - مستوى الصعوبة

```cpp
enum BookDifficulty {
    EASY,  // سهل - يتيح للاعب اختيار مسارين
    HARD   // صعب - مسار واحد فقط
};
```

الاستخدام: يحدد عدد الخيارات المتاحة للاعب بعد حل ألغاز الكتاب.

---

2 هيكل اللغز (Clue Structure)

```cpp
struct Clue {
    string problem;   // نص السؤال أو اللغز
    string solution;  // الإجابة الصحيحة
};
```

مثال عملي:

```cpp
Clue puzzle;
puzzle.problem = "ما هو الشيء الذي يمشي بلا أرجل؟";
puzzle.solution = "الوقت";
```

---

3 هيكل الكتاب (BookNode Structure)

المتغيرات الأساسية

```cpp
int id;                 // رقم تعريف الكتاب (مثل: 1, 2, 3)
BookType type;          // نوع الكتاب (ENTRANCE, INTERMEDIATE, FINAL)
BookDifficulty difficulty; // مستوى الصعوبة (EASY, HARD)
```

متغيرات الألغاز

```cpp
Clue* clues;     // مؤشر لمصفوفة ديناميكية من الألغاز
int clueCount;   // عدد الألغاز في الكتاب
```

ملاحظة: استخدام المؤشر لأن عدد الألغاز متغير لكل كتاب.

مؤشرات الربط (Linked List Pointers)

```cpp
BookNode* next1;  // المؤشر للكتاب التالي (المسار الأول)
BookNode* next2;  // المؤشر للكتاب التالي (المسار الثاني - للكتب السهلة فقط)
```

رسم توضيحي:

```
كتاب سهل (EASY):
[كتاب #1] --next1--> [كتاب #2]
        \--next2--> [كتاب #3]

كتاب صعب (HARD):
[كتاب #4] --next1--> [كتاب #5]
        \--next2--> [NULL]
```

متغيرات الحالة

```cpp
bool visited;      // هل زار اللاعب هذا الكتاب؟ (true/false)
bool puzzleSolved; // هل حل اللاعب ألغاز الكتاب؟ (true/false)
```

---

4 الدوال (Functions)

دوال البناء (Constructors)

```cpp
BookNode();  // بناء افتراضي - يعطي قيم أولية
BookNode(int bookId, BookType bookType, BookDifficulty bookDiff);  // بناء بمعاملات
```

المدمر (Destructor)

```cpp
~BookNode();  // ينظف الذاكرة المحجوزة للألغاز
```

دوال العرض والتحقق

```cpp
void displayInfo();          // يطبع معلومات الكتاب على الشاشة
bool hasTwoPaths();          // يتحقق: هل الكتاب له مسارين؟
string getTypeString();      // يحول نوع الكتاب لنص
string getDifficultyString(); // يحول الصعوبة لنص
```

دوال الألغاز

```cpp
void setupClues(int count);    // يجهز مساحة لعدد معين من الألغاز
void addClue(int index, string problem, string solution);  // يضيف لغز في موقع محدد
Clue getClue(int index);       // يرجع لغز من موقع محدد
```

📝 ملف BookNode.cpp (ملف التنفيذ)

1 البناء الافتراضي

```cpp
BookNode::BookNode() {
    id = 0;                    // رقم افتراضي
    type = INTERMEDIATE;       // نوع وسطي افتراضياً
    difficulty = EASY;         // صعوبة سهلة افتراضياً
    clues = nullptr;           // لا توجد ألغاز في البداية
    clueCount = 0;             // عدد الألغاز صفر
    next1 = nullptr;           // لا يوجد كتاب تالي
    next2 = nullptr;           // لا يوجد مسار ثاني
    visited = false;           // لم يُزر بعد
    puzzleSolved = false;      // لم تُحل الألغاز بعد
}
```

الاستخدام:

```cpp
BookNode book1;  // ينشئ كتاب بقيم افتراضية
```

---

2 البناء بمعاملات

```cpp
BookNode::BookNode(int bookId, BookType bookType, BookDifficulty bookDiff) {
    id = bookId;               // نأخذ الرقم من المعاملات
    type = bookType;           // نأخذ النوع
    difficulty = bookDiff;     // نأخذ الصعوبة
    // باقي المتغيرات نفس البناء الافتراضي
}
```

الاستخدام:

```cpp
BookNode book2(5, ENTRANCE, HARD);  // كتاب رقم 5، مدخل، صعب
```

---

3 المدمر (Destructor)

```cpp
BookNode::~BookNode() {
    if (clues != nullptr) {    // لو في ألغاز محجوزة
        delete[] clues;        // امسح المصفوفة من الذاكرة
        clues = nullptr;       // اجعل المؤشر يشير لـ NULL
    }
}
```

أهميته: يمنع تسرب الذاكرة (Memory Leak) عند حذف الكتاب.

---

4 دالة displayInfo()

تطبع كل معلومات الكتاب بشكل منظم:

```cpp
void BookNode::displayInfo() {
    // طباعة المعلومات الأساسية
    cout << "Book #" << id << endl;
    cout << "Type: " << getTypeString() << endl;
    cout << "Difficulty: " << getDifficultyString() << endl;

    // طباعة قائمة الألغاز
    for (int i = 0; i < clueCount; i++) {
        cout << (i + 1) << ". " << clues[i].problem << endl;
    }

    // طباعة المسارات المتاحة
    cout << "Path 1: " << (next1 != nullptr ? "Available" : "None") << endl;
}
```

مثال خرج:

```
==================================
Book #3
Type: INTERMEDIATE
Difficulty: EASY
Visited: No
Solved: No

Puzzles in this book:
  1. ما الذي له عين ولا يرى؟
  2. أنا أسير بلا قدمين

Available Paths:
  Path 1: Available
  Path 2: Available
==================================
```

---

5 دالة hasTwoPaths()

```cpp
bool BookNode::hasTwoPaths() {
    // فقط الكتب السهلة لها مسارين
    return (difficulty == EASY && next2 != nullptr);
}
```

الاستخدام:

```cpp
if (currentBook->hasTwoPaths()) {
    cout << "اختر المسار (1 أو 2): ";
}
```

---

6 دوال التحويل للنص

```cpp
```

```cpp
string BookNode::getTypeString() {
    if (type == ENTRANCE) return "ENTRANCE";
    if (type == INTERMEDIATE) return "INTERMEDIATE";
    if (type == FINAL) return "FINAL";
    return "UNKNOWN";
}
```

الفائدة: تحول الأرقام (enum values) لنص قابل للقراءة.

---

7 دالة setupClues()

```cpp
void BookNode::setupClues(int count) {
    // لو في ألغاز قديمة، امسحها
    if (clues != nullptr) {
        delete[] clues;
    }

    // احجز مساحة للألغاز الجديدة
    clueCount = count;
    clues = new Clue[count];

    // فضّي كل الألغاز
    for (int i = 0; i < count; i++) {
        clues[i].problem = "";
        clues[i].solution = "";
    }
}
```

مثال استخدام:

```cpp
BookNode book;
book.setupClues(3);  // جهز مكان لـ 3 ألغاز
```

---

8 دالة addClue()

```cpp
```

```cpp
void BookNode::addClue(int index, string problem, string solution) {
    // تأكد من صحة الـ index
    if (index >= 0 && index < clueCount) {
        clues[index].problem = problem;
        clues[index].solution = solution;
    }
}
```

مثال كامل:

```cpp
BookNode book;
book.setupClues(2);
book.addClue(0, "ما هو الشيء الذي يكتب ولا يقرأ؟", "القلم");
book.addClue(1, "له رأس ولا عين له؟", "الدبوس");
```

---

9 دالة getClue()

```cpp
Clue BookNode::getClue(int index) {
    if (index >= 0 && index < clueCount) {
        return clues[index];  // ارجع اللغز
    }

    // غلط، ارجع لغز فاضي لو الـ index
    Clue empty;
    empty.problem = "";
    empty.solution = "";
    return empty;
}
```

الاستخدام:

```cpp
Clue puzzle = book.getClue(0);
cout << puzzle.problem << endl;
string userAnswer;
cin >> userAnswer;
if (userAnswer == puzzle.solution) {
    cout << "صحيح!" << endl;
}
```

🎮 مثال تطبيق كامل

```cpp
// إنشاء ثلاث كتب
BookNode* book1 = new BookNode(1, ENTRANCE, EASY);
BookNode* book2 = new BookNode(2, INTERMEDIATE, HARD);
BookNode* book3 = new BookNode(3, FINAL, EASY);

// ربط الكتب مع بعض
book1->next1 = book2;
book1->next2 = book3;  // الكتاب السهل له مسارين
book2->next1 = book3;

// إضافة ألغاز للكتاب الأول
book1->setupClues(2);
book1->addClue(0, "لغز 1", "حل 1");
book1->addClue(1, "لغز 2", "حل 2");

// بدء اللعبة
BookNode* current = book1;
current->visited = true;
current->displayInfo();

// بعد حل الألغاز
current->puzzleSolved = true;
if (current->hasTwoPaths()) {
    // دع اللاعب يختار
}

// التنظيف
delete book1;
delete book2;
delete book3;
```

🔑 نقاط مهمة

1. **إدارة الذاكرة**: الكود يستخدم `new` و `delete` بشكل صحيح لتجنب تسرب الذاكرة
2. **التحقق من الصحة**: كل دالة تتحقق من صحة المدخلات قبل الاستخدام
3. **المرونة**: النظام يسمح بعدد متغير من الألغاز لكل كتاب
4. **التنظيم**: الكود مقسم بشكل منطقي بين الهيدر والتنفيذ

5. **التعليقات**: الكود معلّق بالعربية لسهولة الفهم

استخدامات محتملة

- **لعبة مغامرات نصية**: حيث كل كتاب محطة في القصة
- **نظام تعليمي**: كل كتاب درس بألغازه
- **متاهة تفاعلية**: المسارات المختلفة تمثل خيارات اللاعب
- **شجرة قرارات**: لبناء نظام قرارات متفرع