ملخص هياكل البيانات

القوائم المترابطة، المكدسات، الطوابير

١. المصفوفة مقابل القائمة المترابطة

العملية	المصفوفة	القائمة المترابطة
الوصول لعنصر برقمه	O(1)	O(n)
الإضافة في البداية	O(n)	O(1)
الإضافة في النهاية	O(1) إن وُجد مكان	O(n)، أو O(1) مع مؤشر للذيل
الحذف من الوسط	O(n)	O(1) بعد الوصول للعقدة
الذاكرة	متجاورة وحجمها ثابت	متفرقة وتنمو عند الحاجة

٢. القائمة المترابطة الأحادية

كل عقدة تحوي البيانات ومؤشراً للعقدة التالية. آخر عقدة مؤشرها فارغ.

```
struct Node {
    int   data;
    Node* next;
};

// الإضافة في بداية القائمة
void pushFront(Node*& head, int value) {
    Node* n = new Node{value, head};
    head = n;
}
```

خطأ شائع: تغيير head قبل ربط العقدة الجديدة بالقائمة، فتضيع بقية العقد.

٣. المكدس (Stack)

مبدؤه: **آخر من يدخل أول من يخرج** (LIFO). العمليات: push للإضافة، pop للحذف، top لقراءة القمة، وكلها O(1).

- استخداماته: التراجع في المحررات، استدعاء الدوال، فحص توازن الأقواس، تحويل التعابير الحسابية.

٤. الطابور (Queue)

مبدؤه: **أول من يدخل أول من يخرج** (FIFO). العمليات: enqueue للإضافة من الخلف، dequeue للحذف من الأمام.

- استخداماته: جدولة المهام، طابور الطباعة، البحث بالعرض في الأشجار والبيانات.
- الطابور الدائري يحل مشكلة ضياع الخانات الفارغة في بداية المصفوفة.

٥. مثال: فحص توازن الأقواس

```
bool balanced(const string& s) {
    stack<char> st;
    for (char c : s) {
        if (c == '(' || c == '[' || c == '{') st.push(c);
        else if (c == ')' || c == ']' || c == '}') {
            if (st.empty()) return false;
            char open = st.top(); st.pop();
            if ((c == ')' && open != '(') ||
                (c == ']' && open != '[') ||
                (c == '}' && open != '{')) return false;
        }
    }
    return st.empty();
}
```

> للمراجعة: تتبّع المكدس يدوياً للتعبير)}[()]{ واذكر لماذا يُرفض.