VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Kỹ Thuật Lập Trình (Cơ bản và nâng cao C++)

KTLT2 - HK242

TASK 2: 3.4 - 3.5

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 3/2025

Mục lục

1 Lý thuyết

- TASK 7 Nền móng OOP - Xây dựng thế giới đối tượng
- TASK 8 Kiến trúc sư OOP - Kiểm soát và tổ chức
- TASK 9 Nghệ thuật lập trình - Đa hình và Trừu tượng hóa
- TASK 10 Linked List

2 Bài Tập Lý thuyết

1. Thành phần nào dưới đây không phải là một phần của một nút (node) trong danh sách liên kết đơn?

   a) Giá trị dữ liệu (data)
   b) Con trỏ đến nút tiếp theo (next)
   c) Con trỏ đến nút trước đó (previous)
   d) Con trỏ đến nút đầu danh sách (head)

2. Thao tác nào dưới đây có độ phức tạp O(1) trong danh sách liên kết đơn?

   a) Thêm vào đầu danh sách
   b) Tìm kiếm một phần tử
   c) Thêm vào cuối danh sách (không có con trỏ tail)
   d) Xóa một phần tử ở vị trí bất kỳ

3. Trong danh sách liên kết đơn, nếu bạn muốn xóa nút cuối cùng thì cần phải...

   a) Đặt con trỏ của nút cuối trỏ về nullptr
   b) Duyệt danh sách đến phần tử kế cuối, đặt next của nó thành nullptr
   c) Đặt con trỏ head bằng nullptr
   d) Đặt con trỏ tail bằng nullptr

4. Giả sử bạn có đoạn mã sau:

```
Node* head = new Node(5);
head->next = new Node(10);
head->next->next = new Node(15);
```

   Giá trị của `head->next->next->data` là bao nhiêu?

   a) 5
   b) 10
   c) 15
   d) Lỗi biên dịch

5. Khi duyệt danh sách liên kết đơn, điều kiện dừng vòng lặp thường là...

   a) Khi con trỏ current trỏ đến nullptr
   b) Khi current->next == nullptr
   c) Khi current->data == -1
   d) Khi danh sách có số lượng phần tử chẵn

6. Câu nào sau đây về danh sách liên kết đơn là đúng?

   a) Chỉ có thể duyệt theo chiều từ cuối danh sách về đầu danh sách
   b) Cần sử dụng mảng để triển khai danh sách liên kết
   c) Có thể thêm phần tử vào đầu danh sách trong thời gian O(1)
   d) Không thể sử dụng con trỏ để quản lý danh sách liên kết

7. Giả sử bạn có danh sách liên kết đơn chứa các phần tử {3, 7, 9, 12}. Nếu thêm phần tử 5 vào đầu danh sách, kết quả sẽ là gì?

   a) {5, 3, 7, 9, 12}
   b) {3, 5, 7, 9, 12}
   c) {3, 7, 9, 12, 5}
   d) {5, 7, 9, 12, 3}

8. Nếu không giải phóng bộ nhớ sau khi sử dụng danh sách liên kết đơn, điều gì có thể xảy ra?

a) Gây lỗi biên dịch
b) Chương trình có thể bị chậm do sử dụng bộ nhớ không hiệu quả
c) Xuất hiện lỗi truy cập bộ nhớ (segmentation fault) ngay lập tức
d) Không có vấn đề gì

9. Phát biểu nào đúng về việc tìm kiếm phần tử trong danh sách liên kết đơn?

a) Luôn mất O(1) thời gian
b) Có thể tìm kiếm phần tử ở bất kỳ vị trí nào trong O(log n)
c) Cần duyệt từng phần tử theo thứ tự để tìm kiếm, nên mất O(n) thời gian
d) Tìm kiếm trong danh sách liên kết nhanh hơn tìm kiếm trong mảng

10. Khi thêm một đơn vị mới vào danh sách liên kết bằng addUnit(), đơn vị này sẽ được:

```
class Unit {
public:
    int id;
    string name;
    Unit(int _id, string _name) : id(_id), name(_name) {}
};

struct Node {
    Unit* data;
    Node* next;
    Node(Unit* u) : data(u), next(nullptr) {}
};

class UnitList {
public:
    Node* head;

    UnitList() : head(nullptr) {}

    void addUnit(Unit* u) {
        Node* newNode = new Node(u);
        newNode->next = head;
        head = newNode;
    }
};
```

a) Thêm vào cuối danh sách.
b) Thêm vào đầu danh sách.
c) Thêm vào giữa danh sách.
d) Ghi đè lên đơn vị cũ.

11. Giả sử danh sách UnitList đã có nhiều phần tử. Đoạn code sau thực hiện thao tác gì?

```
void removeFirst() {
    if (head == nullptr) return;
    Node* temp = head;
    head = head->next;
    delete temp;
}
```

a) Xóa phần tử cuối danh sách.
b) Xóa phần tử đầu danh sách.
c) Xóa tất cả phần tử trong danh sách.
d) Không làm gì cả.

12. Giả sử ta có một danh sách liên kết chứa các đơn vị quân sự (Unit). Unit là lớp cơ sở, và có các lớp dẫn xuất như Infantry (bộ binh) và Tank (xe tăng).

```
class Unit {
public:
    virtual void attack() {
        cout << "Unit attacks!" << endl;
    }
    virtual ~Unit() {}
};

class Infantry : public Unit {
public:
    void attack() override {
        cout << "Infantry charges with rifle!" << endl;
    }
};

class Tank : public Unit {
public:
    void attack() override {
        cout << "Tank fires cannon!" << endl;
    }
};

struct Node {
    Unit* data;
    Node* next;
    Node(Unit* u) : data(u), next(nullptr) {}
};
```

Khi duyệt danh sách và gọi attack() trên mỗi Unit, điều gì xảy ra?

```
void attackAll(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        current->data->attack();
        current = current->next;
    }
}
```

a) Tất cả các đơn vị đều in "Unit attacks!", vì gọi hàm của lớp Unit.
b) Gọi đúng hàm attack() của từng loại đơn vị, nhờ tính đa hình.
c) Chương trình gặp lỗi biên dịch, vì Unit* không thể trỏ đến lớp dẫn xuất.
d) Chương trình gặp lỗi runtime, vì attack() không được khai báo virtual.

3 Câu hỏi thiết kế cơ bản(CS, CE, DS)

1. Vì sao nên dùng một lớp **Army** thay vì chỉ lưu danh sách đơn vị quân sự đơn giản?
2. Chọn **mảng động (vector)** hay **danh sách liên kết** để quản lý đơn vị quân sự? Vì sao?
3. Làm thế nào để xóa một đơn vị khỏi danh sách mà không gây rò rỉ bộ nhớ?
4. Nếu muốn tìm kiếm nhanh một đơn vị theo **ID**, nên chọn cấu trúc dữ liệu nào?
5. Làm sao để tính tổng sức mạnh tấn công của toàn bộ quân đội một cách hiệu quả?

4 Câu hỏi thiết kế trung bình(CS, CE)

1. Làm sao để hỗ trợ nhiều loại đơn vị khác nhau (bộ binh, xe tăng) mà không cần sửa code cũ?
2. Nếu muốn thêm cơ chế nâng cấp đơn vị quân sự (tăng cấp, nâng cấp vũ khí), cần thay đổi thiết kế ra sao?
3. Nếu muốn sắp xếp danh sách đơn vị theo sức mạnh tấn công, chọn thuật toán nào?
4. Làm thế nào để thiết kế phương thức **merge(Army other)** để hợp nhất hai quân đội?
5. Nếu muốn hỗ trợ **undo/redo** khi thay đổi danh sách đơn vị, có thể dùng phương pháp nào?

5 Câu hỏi thiết kế mở rộng (CS)

1. Nếu muốn mô phỏng trận chiến giữa hai **Army**, cần thiết kế phương thức **battle(Army enemy)** như thế nào?
2. Nếu lưu dữ liệu quân đội vào file để tải lại sau, chọn **JSON, XML, hay Binary**? Vì sao?
3. Nếu cần hỗ trợ **đa luồng**, mỗi đơn vị hoạt động độc lập, làm sao để đảm bảo đồng bộ dữ liệu?
4. Nếu cần lưu lịch sử trận chiến giữa các Army nhưng vẫn đảm bảo hiệu suất, nên lưu trữ dữ liệu theo cách nào?
5. Nếu muốn thiết kế hệ thống **đa người chơi (multiplayer)** cho game chiến thuật, làm sao để mở rộng kiến trúc hợp lý?