# Bài 5. Danh sách động một chiều

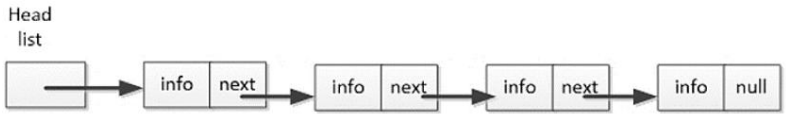
Mục tiêu: đạt đư ợc kiến thức và kỹ năng thực tế trong việc quản lý danh sách một chiều năng động.

## 1 Danh sách một chiều sử dụng ngôn ngữ C++

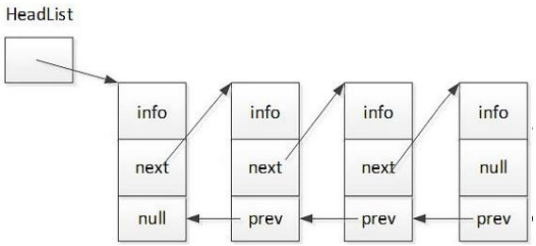
Danh sách tuyến tính liên kết đư ợc thiết kế để lư u trữ các chuỗi nhiều phần tử. Một ô (phần tử danh sách, nút) lư u trữ một giá trị (phần thông tin) và các kết nối (liên kết) với các ô danh sách khác.

Không có khái niệm về vị trí nút (chỉ mục). Để tổ chức các cấu trúc như vậy trong C++, cơ chế con trỏ đư ợc sử dụng.

Tùy thuộc vào số lư ợng kết nối trong một phần tử, có: danh sách một chiều (hoặc liên kết đơ n) và hai chiều (liên kết đôi) (tư ơ ng ứng, Hình 1.a và 1.b).




a) Danh sách liên kết đơ n




b) Danh sách liên kết đôi

Hình 1. Danh sách động tuyến tính

Cấu trúc dữ liệu tuyến tính: danh sách, ngăn xếp (phư ơ ng thức LIFO), hàng đợi (phư ơ ng thức FIFO), bộ bài.

Các cách biểu diễn cấu trúc: các kiểu tùy chỉnh (mảng và các lớp dựa trên con trỏ).

Các thao tác cơ bản trên cấu trúc dữ liệu:

- tìm phần tử mong muốn trong cấu trúc;
- đề cập đến phần tử mong muốn của cấu trúc;
- thêm một phần tử vào cấu trúc;
- Loại bỏ một phần tử khỏi cấu trúc

C++ có một cơ chế thuận tiện và mạnh mẽ để thực hiện động cấu trúc - con trỏ và kiểu cấu trúc (hoặc kiểu dữ liệu trừu tư ợng).

Một nơ i tốt để bắt đầu là mô tả kiểu dữ liệu cho nút danh sách (Liệt kê 1).

Liệt kê 1. Triển khai nút danh sách một chiều

```
struct Node {
  string val;
  Node* next;
  Node(string _val) : val(_val), next(nullptr){}
};
```

Ở đây trư ờng chuỗi mang tính thông tin (chuỗi dữ liệu hữu ích), tiếp theo là kết nối, Nút – hàm tạo của một phiên bản nút danh sách.

Tiếp theo chúng ta triển khai danh sách liên kết đơ n. Danh sách sẽ chứa: 1) một con trỏ tới nút đầu tiên; 2) con trỏ tới nút cuối cùng; 3) hàm tạo danh sách; 4) chức năng kiểm tra sự hiện diện của các nút trong danh sách; 5) chức năng thêm một phần tử vào cuối danh sách; 6) chức năng in danh sách; 7) chức năng tìm kiếm nút trong danh sách theo khóa; 8) chức năng xóa một nút theo khóa.

Trong đoạn mã sau, chúng tôi triển khai các điểm 1-3 (Liệt kê 2).

Liệt kê 2. Mô tả kiểu dữ liệu cho danh sách liên kết đơ n

```
struct list {
    Node* first;
    Node* last;
    list() : first(nullptr), last(nullptr) {}
};
```

Trong mã của hàm chính, bạn cần thêm một phiên bản tạo danh sách, ví dụ: danh sách l;

Hãy thêm một hàm phụ trợ vào cấu trúc danh sách để kiểm tra sự hiện diện của các nút trong danh sách - một hàm một dòng trong đó chúng ta cần kiểm tra xem nút đầu tiên có trống hay không (Liệt kê 3).

Liệt kê 3. Kiểm tra các nút trong danh sách liên kết đơ n

```
bool is_empty() {
    return first == nullptr;
}
```

Hãy triển khai trong cấu trúc danh sách chức năng thêm một phần tử vào cuối danh sách. Có hai trư ờng hợp có thể xảy ra ở đây: a) danh sách trống và b) danh sách không trống.

Trong cả hai trư ờng hợp, cần phải tạo chính nút đó với giá trị đã chỉ định (đư ợc truyền cho hàm).

Đối với trư ờng hợp đầu tiên, bạn sẽ cần hàm đư ợc xác định trư ớc đó để kiểm tra sự hiện diện của các nút. Nếu danh sách trống, con trỏ tới nút đầu tiên và nút cuối cùng sẽ đư ợc hư ớng tới nút mới duy nhất.

Đối với trư ờng hợp thứ hai, bạn cần chỉ ra rằng nút mới nằm sau nút cuối cùng, sau đó chúng ta thay đổi giá trị của con trỏ về nút cuối cùng (Liệt kê 4).

Liệt kê 4. Thêm một nút vào cuối danh sách.

```
void push_back(string _val) {
    Node* p = new Node(_val);
    if (is_empty()) {
      first = p;
      last = p;
      return;
    }
    last->next = p;
    last = p;
}
```

Cái đó. Bạn có thể thêm các nút khác vào danh sách.

Trong cấu trúc danh sách có chức năng in toàn bộ danh sách (nếu danh sách không trống) Trỏ con trỏ p vào nút đầu tiên của danh sách và in giá trị của các nút cho đến khi con trỏ p trống. Tại mỗi lần lặp, chúng tôi chuyển hư ớng p tới nút tiếp theo (Liệt kê 5).

Liệt kê 5. Xuất danh sách ra bàn điều khiển.

```
void print() {
    if (is_empty()) return;
    Node* p = first;
    while (p) {
        cout << p->val << " ";
        p = p->next;
    }
    cout << endl;
}
```

Để tìm kiếm một nút trong danh sách theo giá trị khóa trong cấu trúc danh sách, chúng ta sẽ thêm hàm để duyệt danh sách cho đến khi con trỏ p trống và cho đến khi giá trị của nút p bằng khóa, sau đó chúng ta trả về nút đư ợc tìm thấy, nếu nó tồn tại (trang 6).

Liệt kê 6. Tìm một nút trong danh sách theo khóa.

```
Node* find(string _val) {
    Node* p = first;
    while (p && p->val != _val) p = p->next;
        return (p && p->val == _val) ? p : nullptr;
}
```

Liệt kê 7. Loại bỏ một nút khỏi danh sách bằng khóa.

```
void remove(string _val) {
    if (is_empty()) return;
    if (first->val == _val) {
        remove_first();
        return;
    }
    else if (last->val == _val) {
            remove_last();
        return;
    }
    Node* slow = first;
    Node* fast = first->next;
    while (fast && fast->val != _val) {
        fast = fast->next;
        slow = slow->next;
    }
    if (!fast) {
        cout << "This element does not exist" << endl;
        return;
    }
    slow->next = fast->next;
    delete fast;
}
```

Hãy thêm vào cấu trúc danh sách một hàm để loại bỏ một nút khỏi danh sách theo một chìa khóa.

Nếu danh sách không trống thì có thể xảy ra ba trư ờng hợp:

1) nút có giá trị mong muốn bằng nút đầu tiên;

2) nút có giá trị mong muốn bằng nút cuối cùng;

3) không phải trư ờng hợp đầu tiên và không phải trư ờng hợp thứ hai.

Trư ờng hợp đầu tiên: so sánh giá trị của nút đầu tiên với khóa, nếu giá trị khớp nhau thì chúng ta gọi hàm để xóa nút đầu tiên.

Trư ờng hợp thứ hai: so sánh giá trị của nút cuối cùng với khóa, nếu Các giá trị trùng nhau thì ta gọi hàm xóa nút cuối cùng.

Trư ờng hợp thứ ba:

Con trỏ chậm đư ợc tạo tới nút đầu tiên và con trỏ nhanh đến nút tiếp theo sau nút đầu tiên. Sau đó, miễn là nhanh không trống và miễn là giá trị của nút nhanh hiện tại không bằng khóa, tại mỗi lần lặp, chúng tôi chuyển hư ớng chậm và nhanh đến nút tiếp theo sau chúng. Nếu con trỏ nhanh trống thì chúng tôi sẽ báo lỗi, nếu không chúng tôi chỉ cần xóa nút nhanh.

Mã đư ợc mô tả cùng với các hàm trợ giúp để loại bỏ mã đầu tiên và nút cuối cùng trong danh sách, đư ợc hiển thị trong Liệt kê 7.

Đây là một trong những cách khả thi để thực hiện các thao tác cơ bản với danh sách liên kết đơ n. Các triển khai khác cũng có thể thực hiện đư ợc, bao gồm cả những triển khai dựa trên các kiểu dữ liệu trừu tư ợng trong khuôn khổ OOP.

Các hoạt động cơ bản với các loại cấu trúc tuyến tính khác (danh sách hai chiều, ngăn xếp, hàng đợi, bộ bài) đư ợc triển khai theo cách tư ơ ng tự, đư ợc điều chỉnh cho phù hợp với đặc thù của chúng.

## 2 nhiệm vụ

Thực hiện chư ơ ng trình giải bài toán biến thể bằng danh sách một chiều tuyến tính. Yêu cầu cho tất cả các tùy chọn

---

1. Phần thông tin của nút đư ợc xác định bởi tùy chọn

Phát triển các chức năng chèn nút mới trư ớc nút đầu tiên và xóa-2.
định vị một nút bằng khóa.
3. Triển khai khả năng a) tạo danh sách mới theo cách thủ công và b) sử dụng danh sách tạo sẵn để thử nghiệm các tác vụ của một phiên bản riêng lẻ.

Phát triển một hàm để xuất danh sách ra console. 4. 5. Phát triển
chức năng theo lựa chọn cá nhân. Nếu cần, bạn có thể thêm các chức năng phụ trợ bằng cách phân tách tác vụ.

6. Triển khai menu ngư ời dùng văn bản.
7. Trong chư ơ ng trình chính, kiểm tra từng chức năng
(điểm 2-5).

Lập báo cáo nhiệm vụ đã hoàn thành (dạng báo cáo sau các tùy chọn).

## 3 lựa chọn

| Số. Loại | thông tin<br>các bộ phận của đơ n vị | Hoạt động bổ sung |
|---|---|---|
| 1 | int | Cho hai danh sách tuyến tính một chiều L1 và L2. 1. Phát triển hàm tạo danh sách L bằng cách bao gồm<br>nó chứa mỗi phần tử có giá trị nằm trong ít nhất một trong các danh sách L1 và L2.<br>2. Phát triển chức năng loại bỏ tất cả các nút khỏi danh sách L1<br>ở các vị trí chẵn.<br>3. Phát triển hàm chèn vào danh sách L2 sau mỗi cặp nút một nút mới có giá trị bằng tổng giá trị của hai nút trư ớc đó. Nếu số nút trong danh sách ban đầu là số lẻ thì sau nút cuối cùng sẽ có một nút mới<br>không chèn nút |
| 2 | trôi nổi | Cho hai danh sách tuyến tính một chiều L1 và L2. 1. Phát triển hàm tạo danh sách L, gồm 5 |

| | | |
|---|---|---|
| | | nó chứa từng phần tử có giá trị đư ợc bao gồm đồng thời trong cả hai danh sách L1 và L2.<br>2. Phát triển hàm xóa một nút trong danh sách L2 nằm trư ớc nút chứa giá trị âm. Và cứ thế cho tất cả các nút chứa giá trị âm.<br>3. Phát triển hàm chèn một nút mới có giá trị cho trư ớc trư ớc mỗi nút trong danh sách L1 chứa giá trị lẻ. |
| 3 | ký tự | Cho hai danh sách tuyến tính một chiều L1 và L2. 1. Phát triển hàm tạo danh sách L bằng cách bao gồm nó chứa từng phần tử có giá trị nằm trong danh sách L1 và không nằm trong danh sách L2.<br>2. Phát triển chức năng xóa danh sách con của danh sách L1 đư ợc chỉ định bởi một phạm vi vị trí. Ví dụ, từ ba thứ hai.<br>3. Phát triển hàm sắp xếp các giá trị của danh sách L2, sắp xếp chúng theo thứ tự tăng dần. |
| 4 | int | Cho một danh sách tuyến tính một chiều L1<br>1. Phát triển hàm định dạng lại danh sách L1, viết lại một phần của nó vào đầu danh sách, bắt đầu từ một vị trí nhất định (số vị trí đư ợc truyền cho hàm).<br>2. Phát triển hàm chèn một nút vào danh sách không tăng dần. Tạo một danh sách như vậy L2.<br>3. Phát triển hàm loại bỏ tất cả các phần tử lặp lại khỏi L2 Có giá trị, để lại một trong số chúng. |
| 5 | ký tự | Cho hai danh sách tuyến tính một chiều L1 và L2 với yếu tố câu cá.<br>1. Phát triển chức năng kiểm tra sự bằng nhau của danh sách L1 và L2.<br>2. Phát triển một hàm chèn phần cuối cùng phần tử thứ n của danh sách L2.<br>3. Phát triển chức năng loại bỏ khỏi danh sách L2, các nút, đồng nắm giữ các giá trị kỹ thuật số. |
| 6 | gấp đôi | Cho danh sách một chiều tuyến tính L<br>1. Phát triển hàm chèn trư ớc nút cuối cùng loại bỏ hai đơ n vị mới.<br>2. Xóa giá trị âm đầu tiên khỏi danh sách L nếu nó có mặt trong danh sách.<br>3. Tìm giá trị lớn nhất trong danh sách L và di chuyển nút của nó đến cuối danh sách. |
| 7 | int | Cho danh sách một chiều tuyến tính L<br>1. Xây dựng hàm kiểm tra L có trong danh sách hay không hai phần tử giống hệt nhau. |

| | | |
|---|---|---|
| | | 2. Phát triển một hàm loại bỏ cực đại giá trị nhỏ.<br>3. Phát triển hàm chèn một giá trị mới vào danh sách L trư ớc mỗi nút ở vị trí chẵn. |
| 8 | trôi nổi | Cho danh sách một chiều tuyến tính L<br>1. Phát triển hàm chuyển k nút đầu tiên sang đồng ròng của danh sách.<br>2. Phát triển một hàm hoán đổi các nút với giá trị tối đa và tối thiểu.<br>3. Phát triển hàm loại bỏ nút áp chót danh sách. |
| 9 | ký tự | Cho một danh sách một chiều tuyến tính L chứa văn bản. Mỗi nút có một ký hiệu. Các từ đư ợc phân tách bằng một mẫu phế liệu.<br>1. Phát triển một hàm tìm từ cuối cùng và tìm lại đặt nó ở đầu danh sách.<br>2. Phát triển một hàm loại bỏ từ thứ hai.<br>3. Phát triển hàm thay thế từ thứ k bằng một từ mới. Độ dài của từ mới có thể lớn hơ n độ dài của từ thứ k từ. |
| 10 | ký tự | Cho danh sách một chiều tuyến tính L<br>1. Phát triển hàm xác định trong danh sách L dài nhất một chuỗi khác của các ký tự giống hệt nhau.<br>2. Phát triển một hàm trong mỗi chuỗi Chỉ còn lại một biểu tư ợng giống hệt nhau.<br>3. Phát triển hàm tạo danh sách mới từ các chữ số của danh sách ban đầu, chèn một phần tử vào danh sách mới theo thứ tự tăng dần của các chữ số. Danh sách mới không thể chứa số trùng lặp. |
| 11 | int | Cho một danh sách tuyến tính một chiều L, thông tin một phần trong đó chứa các số có một chữ số và hai chữ số.<br>1. Phát triển hàm tạo mảng A gồm 10 con trỏ tới một phần tử danh sách và bao gồm phần tử khối trong danh sách siva có chỉ số i, các số trong danh sách L bắt đầu bằng chữ số bằng i. Bao gồm ở cuối danh sách. Các số có một chữ số đư ợc bao gồm trong danh sách mảng ở chỉ số 0.<br>2. Xây dựng hàm xóa danh sách L.<br>3. Phát triển hàm tạo danh sách L, bao gồm trong đó các danh sách của mảng A tuần tự từ danh sách có chỉ số 0 đến danh sách có chỉ số 9. |
| 12 | int | Cho một danh sách tuyến tính một chiều L1, thông tin Phần của nó chứa các số có một và hai chữ số, đư ợc sắp xếp theo thứ tự tăng dần của chữ số cao nhất. |

| | | |
|---|---|---|
| | | 1. Phát triển hàm xóa một nút tại một vị trí nhất định các mục của danh sách L1.<br>2. Phát triển hàm tạo danh sách L2 mới bằng cách chèn các phần tử của danh sách L1 vào đó, sắp xếp chúng theo thứ tự tăng dần của chữ số có nghĩa nhỏ nhất. Bằng cách xóa nút đã di chuyển khỏi danh sách L1. 3. Phát triển hàm xác định danh sách L2 sắp xếp theo thứ tự tăng dần. |
| 13 | int | Cho một mảng gồm n con trỏ tới đầu danh sách. Cấu trúc của nút danh sách chứa một khóa (phần thông tin của nút) và một liên kết đến nút tiếp theo.<br>1. Phát triển hàm chèn khóa được truyền dưới dạng tham số vào danh sách thứ i của mảng. Chỉ số i được xác định theo quy tắc: i=key%n. Một số phần tử mảng có thể vẫn còn nullptr.<br>2. Phát triển hàm xóa giá trị khóa khỏi danh sách.<br>3. Phát triển hàm tìm nút có giá trị khóa và trả về con trỏ tới nút tìm thấy. |
| 14 | int | Cho danh sách một chiều tuyến tính L<br>1. Phát triển hàm tạo hai danh sách mới từ giá trị của các nút trong danh sách L: L1 – từ các phần tử dương của mảng L; L2 – từ các phần tử còn lại của danh sách L.<br>2. Phát triển chức năng loại bỏ mọi tiêu cực các yếu tố có giá trị.<br>3. Phát triển hàm trong danh sách L1 là nút có giá trị lớn nhất với một giá trị nhất định được đặt trước nút đầu tiên. |
| 15 | int | Cho một danh sách tuyến tính một chiều L có các nút được sắp xếp theo thứ tự tăng dần phù hợp với các giá trị của phần thông tin của nút.<br>1. Phát triển hàm chèn một giá trị mới vào danh sách L trong khi vẫn giữ nguyên thứ tự của danh sách.<br>2. Phát triển hàm loại bỏ khỏi danh sách L tất cả các nút có giá trị lớn hơn giá trị nhất định.<br>3. Phát triển hàm tạo danh sách L1 mới từ giá trị của các nút trong danh sách L, sao cho trong danh sách L2, các nút được sắp xếp theo thứ tự giá trị giảm dần. |
| 16 | ký tự | Cho hai danh sách tuyến tính một chiều L1 và L2.<br>1. Phát triển hàm chèn vào danh sách L1 sau một nút có giá trị X cho trước tất cả các nút của danh sách L2, nếu nút X nằm trong danh sách L1.<br>2. Phát triển một hàm, từ danh sách L2, loại bỏ tất cả các nút có giá trị không phải là số. |

| | | |
|---|---|---|
| | | 3. Phát triển một hàm từ các chữ số của danh sách L2 tạo thành một chuỗi Số lượng độ sâu bit cho phép thấp. |
| 17 | <độ, hệ số<br><br>ent><br><int, thực> | Một đa thức tuyến tính bậc n được biểu diễn trong chương trình dưới dạng danh sách một chiều tuyến tính. Mỗi nút thứ i của danh sách lưu trữ thông tin về số hạng thứ i của đa thức. Do đó, phần thông tin của một nút bao gồm hai giá trị: bậc của thuật ngữ và hệ số của bậc này. Nếu số hạng thứ i trong đa<br>thành viên bị thiếu, nút không được tạo.<br>1. Phát triển hàm tạo danh sách bằng cách chuyển<br>thành một đa thức dưới dạng tham số: nó được biểu diễn bằng một mảng các hệ số và lũy thừa của chúng.<br>2. Phát triển hàm in đa thức và tiền<br>đặt nó dưới dạng một biểu thức.<br>3. Phát triển hàm tính giá trị của đa thức khi biết giá trị của x. Sử dụng thuật toán Horner trong tính toán. |

## 4 Mẫu báo cáo bài tập

Điều kiện, yêu cầu công việc theo phương án

1. Tuyên bố vấn đề
2. Xác định danh sách các thao tác trên danh sách được xác định trong tiến trình trong quá trình nghiên cứu các vấn đề của nhiệm vụ bổ sung.

   2.1 Xác định cấu trúc của nút danh sách một chiều theo tùy chọn.

   2.2 Draw (vẽ) cho từng thao tác của danh sách kết quả
   quá trình thực hiện một thao tác trên một phương tiện đơn hướng hiện có danh sách chữ ký.

   2.3 Vẽ cấu trúc dữ liệu sẽ được sử dụng trong
   hoạt động.

   2.4 Đưa ra thuật toán thực hiện thao tác

   2.5 Cung cấp bảng test để test từng thao tác
3. Gửi mã chương trình
4. Trình bày kết quả chạy thử chương trình: ảnh chụp màn hình chương trình đã hoàn thành hiểu rõ từng thao tác. 5.

Đưa ra kết luận về kiến thức, kỹ năng đã tiếp thu

6. Danh sách các nguồn thông tin đã được sử dụng
   khi hoàn thành một nhiệm vụ.

# 5 Câu hỏi bảo mật

1) Hãy cho chúng tôi biết về ba cấp độ trình bày dữ liệu trong phần mềm
hệ thống.

2) Điều gì quyết định kiểu dữ liệu?

3) Cấu trúc dữ liệu xác định điều gì?

4) Hãy cho biết cấu trúc lư u trữ dữ liệu trong công nghệ máy tính
logies.

5) Xác định cấu trúc dữ liệu tuyến tính.

6) Xác định cấu trúc dữ liệu danh sách tuyến tính.

7) Xác định cấu trúc dữ liệu ngăn xếp.

8) Xác định cấu trúc dữ liệu hàng đợi.

9) Ngăn xếp khác với cấu trúc dữ liệu danh sách tuyến tính như thế nào?

10) Nên sử dụng loại danh sách tuyến tính nào tốt hơ n nếu bạn cần xuất chuỗi đã nhập
ngư ợc lại?

11) Xác định độ phức tạp của thuật toán chèn một phần tử vào
vị trí thứ i: a) mảng; b) danh sách tuyến tính.

12) Xác định độ phức tạp của thuật toán thực hiện thao tác loại bỏ một phần tử khỏi
vị trí thứ i: a) mảng; b) danh sách tuyến tính.

13) Bản chất thủ thuật của Wirth khi thực hiện thao tác loại bỏ phần tử là gì?
cảnh sát khỏi danh sách?

14) Xác định cấu trúc nút danh sách một chiều.

15) Thực hiện thuật toán suy luận một chiều tuyến tính
danh sách.

16) Cung cấp một đoạn mã chư ơ ng trình bằng ngôn ngữ thực thi C++
thao tác di chuyển phần tử cuối cùng về đầu danh sách.

17) Hành động nào là dư thừa trong đoạn mã sau? Ở đâu
một nút mới đư ợc chèn vào?

```
nút cấu trúc {
        thông tin int;
 Nút*tiếp theo;
};
Nút typedef *Danh sách;
Danh sách L=Danh sách mới;
void InsertToList(List LL, int x){
    Danh sách q=Nút mới; q->thông tin=x; q->tiếp theo=0;
    if (LL==nullptr) LL->next=q;
    khác
      q->next=LL->next;
            LL->tiếp theo=q;
}
```