Nội dung

* Các thao tác cơ bản với **danh sách liên kết đơn**
* Ví dụ xây dựng ứng dụng quản lý danh sách dinh viên dùng sanh sách liên kết đơn

Cấu trúc liên kết

* Các phần tử nằm rải rác trong bộ nhớ
* Các phần tử sẽ được cấp phát động bộ nhớ
* Phần tử trước sẽ lưu lại địa chỉ phần tử kế tiếp (thông qua con trỏ hoặc tham chiếu)
* Cần lưu trữ bao nhiêu phần tử thì cấp phát đủ ==> dùng bộ nhớ tối ưu hơn so với kiểu liên tiếp (mảng)
* Các phần tử móc nối thông qua biến con trỏ/tham chiếu
* Bạn không thể biết được địa chỉ của 1 phần tử bất kỳ 1 cách trực tiếp, cần đi từ phần tử ngay trước nó. Việc truy cập các phần tử là truy cập tuần tự O(n)
* Con trỏ/tham chiếu được coi là bộ nhớ lãng phí dưới góc độ người dùng, tuy nhiên nó bắt buộc phải có để duy trì được cấu trúc
* C/C++ khi xóa phần tử --> phải gọi lệnh giải phóng vùng nhớ cấp phát động
* Java/Python --> frameworks hoặc máy ảo tự giải phóng
* Lưu trữ lại địa chỉ phần tử đầu, nếu không cso địa chỉ phần tử đầu --> KHÔNG thể truy cập vào danh sách

Danh sách liên kết đơn

* Mỗi phần tử chỉ có 1 con trỏ để lưu trữ phần tử kế tiếp
* Kiểu dữ liệu của phần tử có thể là kiểu bất kỳ

Một số thao tác với danh sách liên kết đơn

* Thêm phần tử
  + Thêm vào cuối danh sách
  + Thêm vào đầu danh sách
  + Thêm vào vị trí bất kỳ

=======================

void push(Node\*\* head, int v)

{

Node \*q = makeNode(v);

q->next = \*head;

\*head = q;

}

======================

* Duyêt danh sách
  + In ra giá trị các phần tử trong danh sách
  + Tìm xem có phần tử nào giá trị bằng key trong danh sách

===============

// tim xem khoa key cos xuat hien trong danh sach

// neu co, tra ve dia chi phan tu chua khoa

// nguoc lai, tra ve NULL

Node \*findKey(Node \*head, int key)

{

Node \*p = head;

while(p!=NULL)

{

if(p->value==key) return p;

p=p->next;

}

return NULL;

}

========================

**Bài tập 1**. Viết hàm đếm xem khóa key xuất hiện bao nhiêu lần trong danh sách

**Bài tập 2**. Viết hàm in ra vị trí xuất hiện của khóa key đầu tiên trong danh sách

VD. Danh sách đầu vào 3-->5-->7-->10->7-->5

Và key = 7 thì in ra là 3

Tính vị trí đầu dãy từ 1

=======

Xóa toàn bộ khóa key ytong dãy

void removeAllKey(Node\*\* h, int v){

Node\* p;

// danh sach rong--> khong lam gi ca

if(\*h == NULL) return ;

// neu phan tu can xoa la dau danh sach

if((\*h)->value == v){

Node\* tmp = \*h; \*h = (\*h)->next;

free(tmp);

}

p = \*h;

do

{

// neu phan tu can xoa o giua hoac cuoi

// p tro toi truoc vi tri can xoa

while(p->next != NULL){

if(p->next->value == v) break;

p = p->next;

}

if(p->next != NULL){

Node\* q = p->next; p->next = q->next; free(q);

}

}

while(p->next != NULL);

}

==========

**Bài tập về nhà**

**Bài 1**. sửa lại bài toán quản lý thông tin sinh viên dùng danh sách liên kết đơn

Thông tin sinh viên gồm

* SHSV : char 15
* Họ tên
* Email

Thông tin sinh viên được lưu trữ trong file văn bản theo format sau

2

122334

Nguyen Van A

[A@gmail.com](mailto:A@gmail.com)

2323232

Le thi B

[B\_le@gmail.com](mailto:B_le@gmail.com)

==============

Dòng đầu là số lượng bản ghi trong file

Các dòng tiếp lần lượt là thông tin (mỗi trường trên 1 dòng)

Cần xây dựng menu với các chức năng như ví dụ trong slide

* Đọc danh sách từ file
* Bổ sung thêm thông tin sinh viên vào cuối danh sách (thông tin được nhập từ bàn phím)
* Ghi nội dung dan hsachs hiện tại ra file (theo format như trên)
* Tìm kiếm thông tin sinh viên theo SHSV

**Bài 2**. Đa thức bậc n nêu đầy đủ sẽ gồm n +1 số hạng, tuy nhiên thường đa thức bị khuyết

P = x100 + 10

Để tiết kiệm bộ nhớ, ta sẽ biểu diễn đa thức bằng danh sách liên kết đơn

1 phần tử gồm

* Hệ số
* Số mũ

(1, 100)-->(10,0)

Sắp xếp sao cho **hệ số mũ giảm dần**

P = 5x3 + 7x –10

Được biểu diễn là

(5,3) --> (7,1)-->(-10,0)

Struct Node

{

Double heso;

Double somu;

Struct Node \*next;

}

Hãy xây dựng chương trình

* Nhập vào đa thức bậc n từ bàn phím
* Tính giá trị của đa thỨc (yêu cầu nhập vào thêm giá trị của biến x)
* Tính tổng/hiệu của 2 đa thức (đa thức thứ 2 sẽ được nhập từ bàn phím)