Câu hỏi trong danh sách liên kết đơn:

1. Nếu q là node cuối cùng trong danh sách mà không gọi hàm add tail để thêm p vào sau q như hàm add\_after\_q thì chuyện gì sẽ xảy ra?
2. So sánh danh sách liên kết đơn với mảng cấp phát động thì cấu trúc dữ liệu nào đơn giản hơn, hiệu quả hơn, nhanh hơn về thời gian và độ phức tạp?
3. Ba câu lệnh

{

auto p=q->next;

q->next=p->next;

delete p;

}

Nếu thêm vào mã nguồn c++ thì có cần chỉnh sửa gì không? Và trường hợp đặc biệt khi phần tử cuối trong danh sách lại là phần tử mà ta muốn xóa và q là phần tử áp chót thì chúng ta thực hiện thao tác xóa như thế nào? Có cần thực hiện đúng 3 câu lệnh trên hay không hay sẽ có gì khác biệt?

1. Suy nghĩ và tìm hiểu trong danh sách liên kết đơn chúng ta thực hiện thao tác xóa phần tử cuối như thế nào?
2. Sau khi so sánh mảng cấp phát động và danh sách liên kết đơn ở câu hai thì rút ra câu hỏi lớn nhất trong danh sách liên kết đơn này là những ứng dụng thực tiễn nào phù hợp và cần thiết để sử dụng nó?

Trả lời:

1. Nếu bỏ hàm add\_tail thì chương trình vẫn chạy bình thường, hàm add\_after\_q vẫn hoạt động nhưng lúc này giá trị add\_tail không bị thay đổi sau hàm add\_after\_q thì có thêm vào ở vị trí cuối.
2. Bảng so sánh:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Bảng so sánh giữa mảng cấp phát động và danh sách liên kết đơn*** | | | | | |
|  | **Dynamic** | | **Link list** | |  |
|  | *Time* | *Memory* | *Time* | *Memory* |  |
| **Add\_head** | 0.122176 s | 8.5 MB | 0.0027515 s | 8.9 MB | n=10^4 |
| **Add\_tail** | 0.0004958 s | 8.6 MB | 0.0019516 s | 8.9 MB | n=10^4 |
| **Add\_middle** | 0.0001753 s | 8.7 MB | 1.9e-06 s | 9 MB | n=10^4 |
| **Remove\_head** | 3.58e-05 s | 8.6 MB | 4.9e-06 s | 8.9 MB | n=10^4 |
| **Remove\_tail** | 9e-07 s | 8.6 MB | 6.9e-06 s | 9 MB | n=10^4 |
| **Remove\_middle** | 1.09e^-05 s | 8 MB | 6.6e^-06s | 8.5 MB | n=10^4 |

Kết luận: Mảng cấp phát dộng cỏ thể truy cập được các phần tử trong mảng nên đơn giản hơn, thời gian chạy nhanh hơn, ít tốn bộ nhớ hơn. Tuy nhiên, đối với một số thao tác như thao tác ở vị trí đứng giữa thì mảng phức tạp hơn do phải dời vị trí các phần tử trong mảng,…. Còn danh sách liên kết đơn thì khi thao tác vào vị trí đầu hoặc giữa đều như nhau, không quá phức tạp như trong mảng nhưng cần chú ý các thao tác ở vị trí đầu và cuối và độ dài của danh sách có thể thay đổi linh hoạt hơn so với mảng cấp phát động.

1. Ba câu lệnh trên khi thêm vào mã nguồn C++ không cần chỉnh sửa gì. Và trường hợp đặc biệt khi phần tử cuối trong danh sách lại là phần tử mà ta muốn xóa và q là phần tử áp chót thì ta xử lý như sau:

if (q->next=l.tail){

delete l.tail;

q->next=NULL;

l.tail=q;

}

1. Thao tác xóa phần tử cuối như sau:

void delete\_tail(list &l){

if (l.head==NULL);

else if(l.head==l.tail){

delete l.head;

l.tail=l.head=NULL;

}

else{

auto q=l.head;

while(q->next->next!=NULL)

{q=q->next;}

delete l.tail;

p->next=NULL;

l.tai=q;

}

1. Sau bảng so sánh của câu 2 rút ra ứng dụng thực tiễn và cần thiết khi sử dụng danh sách liên kêt đơn. Tuy sử dụng danh sách liên kết đơn tốn nhiều thời gian và dung lượng bộ nhớ nhưng lại đòngv ai trò quan trọng đối với máy tính. Vì đôi khi bộ nhớ máy tính không có một vùng liên tiếp đủ rộng mà chỉ là những vùng lẻ tẻ nên ta có thể dung danh sách liên kết đơn để lưu trữ dữ liệu và tận dụng tối đa vùng nhớ.