**DANH SÁCH LIÊN KẾT**

# MỤC TIÊU

Hoàn tất bài thực hành này, sinh viên có thể:

* Hiểu được các thành phần của danh sách liên kết.
* Thành thạo các thao tác trên danh sách liên kết: thêm phần tử, xóa phần tử, duyệt danh sách liên kết.
* Áp dụng cấu trúc dữ liệu danh sách liên kết vào việc giải quyết một số bài toán đơn giản. Thời gian thực hành: **từ 120 phút đến 400 phút**

# TÓM TẮT

Danh sách liên kết là cấu trúc dữ liệu dùng để lưu trữ một danh sách (tập hợp hữu hạn) dữ liệu. Điểm đặc biệt của cấu trúc này là khả năng chứa của nó **động** (có thể mở rộng và thu hẹp dễ dàng). Có các loại danh sách liên kết:

* Danh sách liên kết đơn
* Danh sách liên kết kép
* Danh sách liên kết vòng

Mỗi danh sách liên kết là tập hợp các phần tử (element) chứa thông tin lưu trữ của dữ liệu. Giữa các phần tử có một hoặc nhiều liên kết để đảm bảo danh sách liên kết có thể giữ các phần tử này một cách chặt chẽ. *Ví dụ 1:*



Phần tử có một liên kết Phần tử có hai liên kết Phần tử rỗng

*Ví dụ 2:*

Danh sách liên kết đơn

Danh sách liên kết kép



Danh sách liên kết vòng

Trong mỗi phần tử của danh sách liên kết, thông tin liên kết là vô cùng quan trọng. Chỉ cần một xử lý không cẩn thận có thể làm mất phần liên kết này thì danh sách liên kết sẽ bị ‘**gãy**’ từ phần tử đó (không thể truy xuất tiếp các phần tử từ phần tử đó trở về trước hoặc trở về sau).

Các thao tác cơ bản trên danh sách liên kết:

* Thêm phần tử: vào đầu danh sách liên kết, vào cuối danh sách liên kết, vào trước/sau một phần tử trên danh sách liên kết.
* Xóa phần tử: ở đầu danh sách liên kết, ở cuối danh sách liên kết, một phần tử trên danh sách liên kết.
* Duyệt danh sách liên kết: để có thể đi được hết các phần tử trên danh sách liên kết.

# NỘI DUNG THỰC HÀNH

## Cơ bản

Sinh viên đọc kỹ phát biểu bài tập và thực hiện theo hướng dẫn:

*Tổ chức một danh sách liên kết đơn trong đó mỗi phần tử chứa thông tin dữ liệu nguyên.*

***Phân tích***

* Danh sách liên kết đơn gồm mỗi phần tử chứa dữ liệu nguyên. Thông tin của mỗi phần tử được khai báo theo ngôn ngữ C/C++ như sau:
* Thao tác cần thực hiện: **thêm** phần tử nguyên **vào đầu** danh sách liên kết (**First**), **in** các phần tử của danh sách liên kết (**Travel**), **loại bỏ tất cả** các phần tử trên danh sách liên kết (**RemoveAll**).

***Chương trình mẫu***

|  |
| --- |
| /// File: element.h  #ifndef ELEMENT\_H  #define ELEMENT\_H  class element  {  private:  int data;  element \*pointer;  public:  element();  element(int);  virtual ~element();  int Getdata() { return data; }  void Setdata(int val) { data = val; }  element \* Getpointer() { return pointer; }  void Setpointer(element\* val) { pointer = val; }  protected:  };  #endif // ELEMENT\_H |
| /// File: element.cpp  #include "element.h"  element::element()  {  //ctor  this->data=0;  this->pointer=nullptr;  }  element::element(int data)  {  //ctor  this->data=data;  this->pointer=nullptr;  }  element::~element()  {  //dtor  }  /// File: linkedlist.h  #ifndef LINKEDLIST\_H  #define LINKEDLIST\_H  #include "element.h"  class linkedlist  {  private:  element\* head;  element\* tail;  public:  linkedlist();  virtual ~linkedlist();  element\* Gethead() { return head; }  void Sethead(element\* val) { head = val; }  element\* Gettail() { return tail; }  void Settail(element\* val) { tail = val; }  void InsertFirst(element\*);  void InsertTail(element\*);  void DeleteFirst();  void Travel();  protected:  };  #endif // LINKEDLIST\_H  /// File: linkedlist.cpp  #include "linkedlist.h"  #include <iostream>  using namespace std;  linkedlist::linkedlist()  {  //ctor  this->head=nullptr;  this->tail=nullptr;  }  linkedlist::~linkedlist()  {  //dtor  }  void linkedlist::InsertFirst(element\* e){  if(this->head==nullptr)  this->head=this->tail=e;  else{  e->Setpointer(this->head);  this->head=e;  }  }  void linkedlist::InsertTail(element\*e){  if(this->head==nullptr)  this->head=this->tail=e;  else{  this->tail->Setpointer(e);  this->tail=e;  }  }  void linkedlist::Travel(){  element\* p=this->head;  while(p!=nullptr){  cout<<p->Getdata()<<"\t";  p=p->Getpointer();  }  }  void linkedlist::DeleteFirst(){  element\*p=this->head;  this->head=this->head->Getpointer();  delete p;  }  ///File: main.cpp  #include <iostream>  #include "linkedlist.h"  using namespace std;  int main()  {  linkedlist \*list\_=new linkedlist();  element \*e;  e=new element(9);  list\_->InsertTail(e);  e=new element(10);  list\_->InsertTail(e);  e=new element(8);  list\_->InsertTail(e);  list\_->Travel();  list\_->DeleteFirst();  cout<<"\n";  list\_->Travel();  return 0;  } |

***Yêu cầu***

1. Biên dịch đoạn chương trình nêu trên.
2. Cho biết kết quả in ra màn hình:
3. Nêu nhận xét ngắn gọn mối liên hệ giữa thứ tự nhập dữ liệu vào với thứ tự in dữ liệu ra màn hình.
4. Vẽ hình danh sách liên kết theo dữ liệu được nhập ở câu 2.
5. Viết hàm RemoveAll xóa tất cả các phần tử trong DSLK

## Áp dụng – Nâng cao

1. Bổ sung chương trình mẫu cho phép tính **tổng giá trị** các phần tử trên danh sách liên kết đơn gồm các giá trị nguyên.

Gợi ý: tham khảo hàm Travel để viết hàm **SumList**.

1. Bổ sung chương trình mẫu cho phép tìm **giá trị nguyên lớn nhất** trong số các phần tử nguyên trên danh sách liên kết đơn gồm các giá trị nguyên.

Gợi ý: tham khảo hàm Travel để viết hàm **MaxList**.

1. Bổ sung chương trình mẫu cho phép tính **số lượng các phần tử** của danh sách liên kết đơn gồm các giá trị nguyên.

Gợi ý: tham khảo hàm Travel để viết hàm **CountList**.

1. Bổ sung chương trình mẫu cho phép **thêm vào cuối** danh sách liên kết đơn một giá trị nguyên.

Gợi ý: tham khảo hàm InsertFirst để viết hàm Insert**Tail**.

1. Bổ sung chương trình mẫu cho phép **xóa phần tử đầu** danh sách liên kết đơn.
2. Bổ sung chương trình mẫu cho phép **xóa phần tử cuối** danh sách liên kết đơn.
3. Bổ sung chương trình mẫu cho biết **số lượng các phần tử** trên danh sách liên kết đơn có giá trị trùng với giá trị ***x*** được cho trước.

Gợi ý: tham khảo thao tác duyệt danh sách liên kết trong hàm **Travel**.

1. Bổ sung chương trình mẫu cho phép tạo một danh sách liên kết đơn gồm các phần tử mang giá trị nguyên trong đó không có cặp phần tử nào mang giá trị giống nhau.

Gợi ý: sử dụng hàm InsertHead hoặc InsertTail có bổ sung thao tác kiểm tra phần tử giống nhau.

9. Cho sẵn một danh sách liên kết đơn gồm các phần tử mang giá trị nguyên và một giá trị nguyên ***x***. Hãy tách danh sách liên kết đã cho thành 2 danh sách liên kết: một danh sách gồm các phần tử có giá trị nhỏ hơn giá trị ***x*** và một danh sách gồm các phần tử có giá trị lớn hơn giá trị ***x***.

# BÀI TẬP THÊM

1. Đề xuất cấu trúc dữ liệu thích hợp để biểu diễn đa thức (anxn + an-1xn-1+..+ a1x + a0) bằng danh sách liên kết (đơn hoặc kép). Cài đặt các thao tác trên danh sách liên kết đơn biểu diễn đa thức: a. In đa thức
   1. Rút gọn đa thức
   2. Cộng hai đa thức
   3. Nhân hai đa thức
2. Thông tin của một quyển sách trong thư việc gồm các thông tin:
   * Tên sách (chuỗi)
   * Tác giả (chuỗi, tối đa 5 tác giả)
   * Nhà xuất bản (chuỗi)
   * Năm xuất bản (số nguyên)
   1. Hãy tạo danh sách liên kết (đơn hoặc kép) chứa thông tin các quyển sách có trong thư viện (được nhập từ bàn phím).
   2. Cho biết số lượng các quyển sách của một tác giả bất kỳ (nhập từ bàn phím).
   3. Trong năm YYYY (nhập từ bàn phím), nhà xuất bản ABC (nhập từ bàn phím) đã phát hành những quyển sách nào.