TRƯỜNG ĐẠI HỌC SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT 2025  
BÀI TẬP   
DANH SÁCH LIÊN KẾT

Sinh viên thực hiện: Huỳnh Hoàng Phong

MSSV: 3124411226

Giảng viên hướng dẫn: Đỗ Như Tài

Thành phố Hồ Chí Minh, tháng 03 năm 2025

**Bài 1.** Danh sách liên kết số nguyên

(a) Cài đặt CTDL ListInt dùng để chứa các số nguyên, trong đó:

• Sử dụng danh sách liên kết

• Cài đặt các phép toán: khởi tạo, hủy, thêm phần tử, xóa phần tử, thêm một danh sách phần tử, xuất danh sách

(b) Viết thành chương trình:

• Nhập 10 số nguyên và đưa vào danh sách

• In danh sách ra màn hình

• Nhập một số k và xóa số k trong danh sách

• In danh sách sau khi xóa phần tử

• Nhập 5 số nguyên vào một danh sách thứ hai

• Thêm danh sách thứ hai vào danh sách thứ nhất

• In danh sách thứ nhất ra màn hình

**Chương trình**

|  |
| --- |
| #include <iostream>  using namespace std;  struct Node {      int info;      Node\* next;  };  // Khởi tạo danh ds liên kết   void ListInt (Node\*& pHead){      pHead = nullptr;  }  //Hàm chèn 1 nút cuối danh sách liên kết  void Insert(Node\*& pHead, int n) {      Node\* newNode = new Node {n, nullptr};      if (!pHead) {          pHead = newNode;          return;      }      Node\* temp = pHead;     while (temp -> next ) {      temp = temp -> next;     }     temp -> next = newNode;  }  // Xóa 1 phần tử k trong danh sách  void Xoa(Node\*& pHead, int k){      if (!pHead) return;  // Nếu phần tử đầu tiên là k  if (pHead->info == k){      Node\* temp = pHead;      pHead = pHead-> next;      delete temp;      return;  }  Node\* temp = pHead;  while (temp-> next && temp-> next ->info != k ){      temp = temp -> next;  }  // tìm thấy k  if (temp -> next){      Node\* toDelete = temp-> next;      temp-> next = temp->next -> next;      delete toDelete;  }  }  // Hàm hiển thị danh sách liên kết  void PrintList (Node\* pHead){      if (!pHead){          cout <<"Ds rong \n";          return;      }      while (pHead){          cout << pHead -> info <<" " ;          pHead = pHead -> next;      }      cout << endl;  }   // Hàm thêm 1 danh sách liên kết thứ 2   void Merge(Node\*& pHead1, Node\*& pHead2) {      if (!pHead1) {          pHead1 = pHead2;      } else {          Node\* temp = pHead1;          while (temp->next) {              temp = temp->next;          }          temp->next = pHead2;      }      pHead2 = nullptr;  // Tránh giải phóng trùng lặp  }  // xóa toàn bộ danh sách  void ClearList (Node\*& pHead){      Node\* temp;      // lặp lại cho tới khi được xóa hết không gì trong danh sách      while (pHead){          temp = pHead;          pHead = pHead -> next;          delete temp;      } // Hết while  } // Hết ClearList  int main (){      Node\* list1 = nullptr;      Node\* list2 = nullptr;      ListInt (list1);      ListInt (list2);      // nhập 10 số nguyên từ bàn phím  cout <<"Nhap 10 so nguyen vao ds 1: \n";  for (int i = 0; i < 10; i++){      int x;      cin >> x;      Insert(list1, x);  }      // In ra danh sách màn hình  cout << "Danh sanh 1: ";  PrintList(list1);      // nhập số k từ bàn phím và xóa      int k;      cout << "Nhap so k can xoa: ";      cin >> k;      Xoa(list1, k);      // In ra ds sau khi xóa      cout <<"Danh sach sau khi xoa" << k << ": ";      PrintList(list1);      // nhập 5 số nguyên từ bàn phím vào 1 danh sách thứ 2      cout <<"Nhap 5 so nguyen cho ds 2: \n";      for (int i = 0; i < 5; i++){          int x;          cin >> x ;          Insert(list2, x);      }      // Thêm danh sách 2 vào chính danh sách 1  Merge(list1, list2);      // In ra danh sách 1 hoàn chỉnh sau khi đã xóa k và thêm danh sách 2 vào phần tử 1  cout << "Danh sach cuoi: ";  PrintList(list1);  // Giải phóng bộ nhớ  ClearList(list1);      return 0;  } |

|  |
| --- |
| /\*Nhap 10 so nguyen vao ds 1:  1 2 3 4 5 6 7 8 9 10  Danh sanh 1: 1 2 3 4 5 6 7 8 9 10  Nhap so k can xoa: 1  Danh sach sau khi xoa1: 2 3 4 5 6 7 8 9 10  Nhap 5 so nguyen cho ds 2:  1 2 3 4 5  Danh sach cuoi: 2 3 4 5 6 7 8 9 10 1 2 3 4 5  \*/ |

**Độ khó của thuật toán.**

**1. Chèn một nút vào cuối danh sách liên kết**

* **Thời gian**: O(n (Duyệt đến cuối danh sách trước khi chèn)
* **Không gian**: O(1
* **Ghi chú**: Nếu chèn nhiều phần tử liên tiếp, nên dùng con trỏ tail để tối ưu xuống O(1)

**2. Xóa một phần tử có giá trị kkk**

* **Thời gian**: O(n) (Duyệt qua danh sách để tìm và xóa)
* **Không gian**: O(1)
* **Ghi chú**: Nếu phần tử đầu bị xóa, cần cập nhật lại con trỏ đầu.

**3. In danh sách liên kết**

* **Thời gian**: O(n) (Duyệt toàn bộ danh sách)
* **Không gian**: O(1)

**4. Gộp hai danh sách liên kết**

* **Thời gian**: O(n) (Duyệt đến cuối danh sách 1 rồi gán danh sách 2)
* **Không gian**: O(1)
* **Ghi chú**: Nếu danh sách 1 rỗng, chỉ cần trỏ đến danh sách 2.

**5. Xóa toàn bộ danh sách**

* **Thời gian**: O(n) (Duyệt từng phần tử và giải phóng bộ nhớ)
* **Không gian**: O(1)

**Tổng kết**

* **Tất cả các thao tác đều có độ phức tạp tối đa O(n), không có thuật toán nào vượt quá O(n)**
* **Chèn cuối và gộp danh sách có thể tối ưu xuống O(1) nếu sử dụng con trỏ tail**.
* **Xóa phần tử có thể chậm nếu phần tử cần xóa nằm cuối danh sách**.

**Bài 2.** Danh sách liên kết sinh viên

(a) Cài đặt CTDL SinhVien để quản lý một sinh viên gồm có: họ tên (50 ký tự), địa chỉ (70 ký tự), lớp (10 ký tự),

khóa (số nguyên). Cài đặt 4 hàm so sánh 2 sinh viên theo từng tiêu chí. Viết hàm nhập một sinh viên, hàm xuất

một sinh viên.

(b) Cài đặt CTDL ListSV dùng để chứa các sinh viên, trong đó:

• Sử dụng danh sách liên kết

• Cài đặt các phép toán: khởi tạo, hủy, thêm phần tử, xóa phần tử, thêm một danh sách phần tử, xuất danh sách, sắp

xếp danh sách sử dụng selection sort và con trỏ hàm so sánh.

(c) Viết thành chương trình:

• Nhập 10 sinh viên và đưa vào danh sách

• In danh sách ra màn hình

• Xóa sinh viên có tên “Nguyen Van Teo” trong danh sách

• Xóa sinh viên có địa chỉ “Nguyen Van Cu” trong danh sách

• Thêm sinh viên có tên “Tran Thi Mo”, địa chỉ “25 Hong Bang”, lớp “TT0901”, khóa 2009 vào danh sách.  
  
**Chương trình**

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  struct SinhVien {      char HoTen[50];      char DiaChi[70];      char Lop[10];      int Khoa;  };  struct Node {      SinhVien data;      Node\* next;  };  // Khởi tạo danh sách liên kết  void KhoiTaoDanhSach(Node\*& pHead) {      pHead = nullptr;  }  // Nhập thông tin của sinh viên  void ThongTin(SinhVien& sv) {      cout << " Ho va ten: ";      cin.ignore();      cin.getline(sv.HoTen, 50);      cout << " dia chi: ";      cin.getline(sv.DiaChi, 70);      cout <<" lop: ";      cin.getline(sv.Lop, 10);      cout <<" Khoa: ";      cin >> sv.Khoa;      cin.ignore();  }  // In thông tin sinh viên  void InThongTin(SinhVien sv) {      cout << sv.HoTen << " , " << sv.DiaChi << " , " << sv.Lop << " , " << sv.Khoa << endl;  }  // Thêm sinh viên vào danh sách  void Them(Node\*& pHead, SinhVien sv) {      Node\* newNode = new Node;      newNode->data = sv;      newNode->next = pHead;      pHead = newNode;  }  // Xuất danh sách sinh viên  void Xuat(Node\* pHead) {      if (!pHead) {          cout << " Ds rong \n";          return;      }      Node\* temp = pHead;      while (temp) {          InThongTin(temp->data);          temp = temp->next;      }  }  // Xóa sinh viên theo tên  void Xoa1(Node\*& pHead, const char\* ten) {      Node\* temp = pHead;      Node\* prev = nullptr;      while (temp) {          if (strcmp(temp->data.HoTen, ten) == 0) {              if (prev == nullptr) {                  pHead = temp->next;              } else {                  prev->next = temp->next;              }              delete temp;              return;          }          prev = temp;          temp = temp->next;      }  }  // Xóa sinh viên theo địa chỉ  void Xoa2(Node\*& pHead, const char\* diachi) {      Node\* temp = pHead;      Node\* prev = nullptr;      while (temp) {          if (strcmp(temp->data.DiaChi, diachi) == 0) {              if (prev == nullptr) {                  pHead = temp->next;              } else {                  prev->next = temp->next;              }              delete temp;              return;          }          prev = temp;          temp = temp->next;      }  }  // So sánh theo họ tên  bool SoSanh(SinhVien a, SinhVien b) {      return strcmp(a.HoTen, b.HoTen) < 0;  }  // Sắp xếp sinh viên bằng selection sort  void SelectionSort(Node\*& pHead, bool (\*cmp)(SinhVien, SinhVien)) {      for (Node\* p = pHead; p; p = p->next) {          Node\* minNode = p;          for (Node\* q = p->next; q; q = q->next) {              if (cmp(q->data, minNode->data)) {                  minNode = q;              }          }          if (minNode != p) {              swap(p->data, minNode->data);          }      }  }  // Xóa toàn bộ danh sách  void ClearList(Node\*& pHead) {      Node\* temp;      while (pHead) {          temp = pHead;          pHead = pHead->next;          delete temp;      }  }  int main() {      Node\* DanhSach;      KhoiTaoDanhSach(DanhSach);      // Nhập 10 sinh viên vào danh sách      cout << "Nhap 10 sinh vien: \n";      for (int i = 0; i < 10; i++) {          SinhVien sv;          ThongTin(sv);          Them(DanhSach, sv);      }      // In danh sách ra màn hình      cout << "\nDanh sach sinh vien vua nhap:\n";      Xuat(DanhSach);      // Xóa sinh viên có tên "Nguyen Van Teo"      Xoa1(DanhSach, "Nguyen Van Teo");      // Xóa sinh viên có địa chỉ "Nguyen Van Cu"      Xoa2(DanhSach, "Nguyen Van Cu");      // Thêm sinh viên mới      SinhVien sv = {"Tran Thi Mo", "25 Hong Bang", "TT0901", 2009};      Them(DanhSach, sv);      cout << "\nDanh sach sau khi them/xoa:\n";      Xuat(DanhSach);      // Sắp xếp danh sách theo tên      SelectionSort(DanhSach, SoSanh);      cout << "\nDanh sach sau khi sap xep theo ten:\n";      Xuat(DanhSach);      // Giải phóng bộ nhớ      ClearList(DanhSach);      return 0;  } |

#### **Độ khó thuật toán 1. Thêm sinh viên vào danh sách liên kết**

* **Thời gian**: O(1) (Thêm vào đầu danh sách)
* **Không gian**: O(1) (Chỉ thêm một node mới)
* **Ghi chú**: Rất nhanh vì chỉ cần cập nhật con trỏ đầu danh sách.

**2. Xuất danh sách sinh viên**

* **Thời gian**: O(n) (Duyệt qua toàn bộ danh sách)
* **Không gian**: O(1)
* **Ghi chú**: Duyệt từng phần tử và in ra, tốc độ tuyến tính.

**3. Xóa sinh viên (theo tên hoặc địa chỉ)**

* **Thời gian**: O(n) (Tìm phần tử cần xóa trong danh sách)
* **Không gian**: O(1)
* **Ghi chú**: Nếu phần tử cần xóa ở đầu danh sách, thao tác sẽ nhanh hơn.

**4. Sắp xếp sinh viên bằng Selection Sort**

* **Thời gian**: O(n^2) (Duyệt từng phần tử và tìm phần tử nhỏ nhất)
* **Không gian**: O(1) (Không dùng mảng phụ, chỉ hoán đổi dữ liệu)
* **Ghi chú**: Không hiệu quả với danh sách dài, tốt hơn nên dùng Merge Sort O(nlogn).

**5. Xóa toàn bộ danh sách**

* **Thời gian**: O(n)(Duyệt danh sách và giải phóng từng node)
* **Không gian**: O(1)
* **Ghi chú**: Giải phóng bộ nhớ từng phần tử, tránh rò rỉ bộ nhớ.

**Tổng kết**

* **Thêm, xóa, xuất danh sách** có độ phức tạp **tuyến tính O(n) hoặc tốt hơn**.
* **Sắp xếp Selection Sort** có độ phức tạp **bình phương O(n^2)**, không phù hợp với danh sách lớn.