**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

----o0o----



**BÁO CÁO BÀI TẬP LỚN**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**Đề tài:** **Quản lý nhân sự bằng danh sách liên kết.  
Cài đặt cấu trúc dữ liệu Double Linked List cho một dãy số nguyên.**

**Giảng viên hướng dẫn: TS. Phạm Xuân Tích**

**Sinh viên thực hiện: Lại Trường Giang – 231220753**

**Lớp: Công nghệ thông tin 3 – K64**

**Hà Nội, tháng 11 năm 2024**

MỤC LỤC

[1. GIỚI THIỆU 4](#_Toc182286880)

[1.1 Lời mở đầu 4](#_Toc182286881)

[1.2 Định nghĩa bài toán 4](#_Toc182286882)

[1.3 Các nhiệm vụ và tiến độ hoàn thành 5](#_Toc182286883)

[2. PHƯƠNG PHÁP LỰA CHỌN 6](#_Toc182286884)

[2.1 Cấu trúc dữ liệu Single Linked List : 6](#_Toc182286885)

[2.2 Cấu trúc dữ liệu Double Linked List : 6](#_Toc182286887)

[2.3 Vector 7](#_Toc182286889)

[3. TRIỂN KHAI CÀI ĐẶT 9](#_Toc182286891)

[3.1 Ngôn ngữ lập trình và thư viện 9](#_Toc182286892)

[3.2 Tổ chức chương trình và đóng gói 9](#_Toc182286893)

[4 PHÂN TÍCH CHƯƠNG TRÌNH 11](#_Toc182286894)

[4.1 Sử dụng Single Linked List để tạo chương trình quản lý nhân sự (Phần A)..11](#_Toc182286895)

[4.1.1 Xây dựng class cán bộ và hàm khởi tạo .. 11](#_Toc182286896)

[4.1.2 Quản lý danh sách nhân sự sử dụng **Single Linked List** và sắp xếp bằng cách sử dụng **Vector** 11](#_Toc182286897)

[4.2 Cài đặt cấu trúc dữ liệu Double Linked List (Phần B) 16](#_Toc182286898)

[5 KẾT QUẢ THỰC NGHIỆM 20](#_Toc182286899)

[5.1 Dữ liệu. 20](#_Toc182286900)

[5.2 Các kết quả. 22](#_Toc182286901)

[6 KẾT LUẬN 23](#_Toc182286902)

[6.1 Đánh giá mức độ hoàn thành . 24](#_Toc182286903)

[6.2 Bài học rút ra . 24](#_Toc182286905)

[6.3 Khó khăn với môn học](#_Toc182286907)

[7 LỜI CẢM ƠN 26](#_Toc182286908)

**TÀI LIỆU THAM KHẢO……………………………………………………..27**

**DANH SÁCH HÌNH ẢNH**

|  |  |  |
| --- | --- | --- |
| **Hình** | **Nội dung** | **Trang** |
| 1 | Minh họa cấu trúc Single Linked List | 7 |
| 2 | Minh họa cấu trúc Double Linked List | 8 |
| 3 | Minh họa Vector | 9 |
| 4 | In ra màn hình danh sách nhân sự | 22 |
| 5 | Thêm nhân sự mới | 23 |
| 6 | Tìm cán bộ có lương cao nhất | 23 |
| 7 | In ra màn hình kết quả | 24 |
|  |

# GIỚI THIỆU

## Lời mở đầu :

Trong lĩnh vực công nghệ thông tin, Cấu trúc dữ liệu và Giải thuật đóng vai trò quan trọng trong việc xây dựng nền tảng vững chắc cho quá trình thiết kế và phát triển các phần mềm, ứng dụng. Việc lựa chọn và sử dụng cấu trúc dữ liệu cùng thuật toán phù hợp không chỉ giúp tối ưu hóa hiệu suất mà còn nâng cao khả năng giải quyết các vấn đề phức tạp trong thực tế.

Báo cáo này được thực hiện nhằm trình bày các kiến thức căn bản về cấu trúc dữ liệu và các phương pháp giải thuật cơ bản. Đồng thời, thông qua báo cáo, em muốn thể hiện cách tiếp cận và áp dụng các kiến thức đã học vào phân tích và giải quyết một số bài toán tiêu biểu.

Trong quá trình thực hiện, dù đã cố gắng hoàn thiện, báo cáo vẫn khó tránh khỏi những thiếu sót. Chúng em mong nhận được ý kiến đóng góp từ thầy cô và các bạn để hoàn thiện hơn.

## Định nghĩa bài toán :

*Bài toán A: (Bài toán quản lý nhân sự)*

*Input:* Chương trình cho phép người dùng quản lý thông tin nhân sự công ty, bao gồm thêm, xóa, tìm kiếm, sắp xếp, và hiện thị danh sách nhân viên bằng cách sử dụng ***App Menu***.

*Ouput:* Chương trình giúp quản lý và thao tác với thông tin nhân sự một cách hiệu quả.

*Bài toán B: (Bài số 20 trong danh sách BTL)*

* Xây dựng bài toán bằng cách cài đặt cấu trúc dữ liệu Double List.
* Mô phỏng hoạt động của thuật toán :

*Input:* Dòng đầu chứa số nguyên dương n là số phần tử của dãy.

n dòng tiếp theo, mỗi dòng chứa một số nguyên.   
 Một số nguyên x để chèn vào danh sách sau khi sắp xếp.

*Ouput:* Danh sách theo chiều ngược (tức là danh sách giảm dần) sau khi đã xử lý toàn bộ yêu cầu.

## Các nhiệm vụ và tiến độ hoàn thành :

|  |  |
| --- | --- |
| Nhiệm vụ | Tiến độ hoàn thành |
| * Nghiên cứu lớp List * Lên ý tưởng và phân tích bài toán A. * Triển khai một cấu trúc dữ liệu danh sách liên kết đơn (Single Linked list). * Sử dụng list hoặc vector để tạo chương trình quản lý nhân sự hỗ trợ thao tác thêm, xóa, tìm kiếm, sắp xếp và hiển thị thông tin nhân viên. | Đã hoàn thành |
| * Tìm hiểu về Double Linked List. * Lên ý tưởng bài toán B. * Triển khai và cài đặt cấu trúc danh sách liên kết đôi (Double Linked List). * Mô phỏng cách hoạt động và ứng dụng vào giải bài toán đã cho. | Đã hoàn thành |
| * Làm báo cáo. * Lưu trữ code bằng Github. | Đã hoàn thành |

# PHƯƠNG PHÁP LỰA CHỌN

## Cấu trúc dữ liệu Single Linked List :

**Single Linked List** (Danh sách liên kết đơn) là một cấu trúc dữ liệu tuyến tính, trong đó các phần tử được lưu trữ dưới dạng các node. Mỗi node chứa hai thành phần:

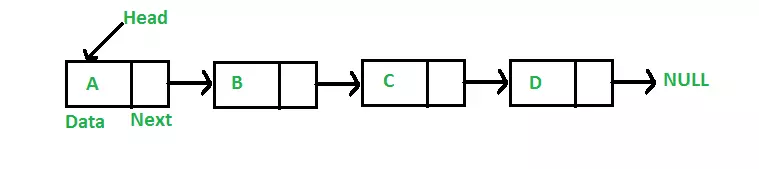
* **Giá trị**: Lưu trữ dữ liệu của node.
* **Con trỏ**: Trỏ đến node tiếp theo trong danh sách.

Trong Single Linked List, mỗi node chỉ biết đến node kế tiếp mà không có thông tin về node trước đó. Điều này giúp giảm thiểu bộ nhớ sử dụng so với danh sách liên kết đôi (Double Linked List) nhưng cũng có một số hạn chế, ví dụ như không thể duyệt ngược lại và chỉ có thể đi theo một chiều từ đầu đến cuối danh sách.

Các thao tác cơ bản trên Single Linked List bao gồm:

* **Thêm node** vào đầu hoặc cuối danh sách.
* **Xóa node** khỏi đầu, cuối, hoặc một vị trí cụ thể trong danh sách.
* **Duyệt qua danh sách** để tìm hoặc in các phần tử.

Single Linked List thường được sử dụng khi dữ liệu cần lưu trữ có kích thước thay đổi và không yêu cầu truy cập ngẫu nhiên hoặc duyệt ngược.



*Hình 1: Minh họa cấu trúc Single Linked List*

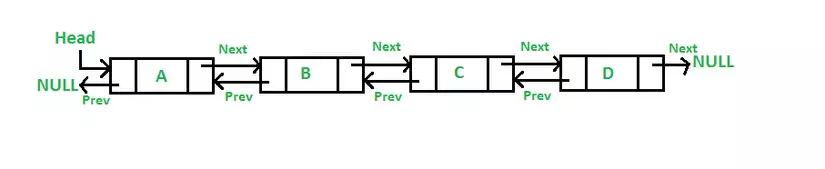
* 1. **Cấu trúc dữ liệu Double Linked List :**

**Double Linked List** (Danh sách liên kết đôi) là một biến thể mở rộng của Single Linked List, trong đó mỗi node chứa ba thành phần:

* **Giá trị**: Dữ liệu của node.
* **Con trỏ tới node trước đó**.
* **Con trỏ tới node tiếp theo**.

Điểm khác biệt chính của Double Linked List là mỗi node có thể tham chiếu cả node trước và node sau, cho phép duyệt danh sách theo cả hai chiều (xuôi và ngược). Nhờ tính năng này, việc thêm, xóa, hay tìm kiếm một node trong danh sách có thể được thực hiện một cách linh hoạt hơn, đặc biệt là khi cần duyệt hoặc thao tác từ giữa danh sách.

Tuy nhiên, Double Linked List yêu cầu nhiều bộ nhớ hơn so với Single Linked List vì cần lưu thêm một con trỏ cho mỗi node. Đây là lựa chọn tốt khi cần khả năng duyệt ngược và thường sử dụng trong các trường hợp cần lưu trữ và quản lý các dữ liệu không có kích thước cố định, như các ứng dụng cần xóa hoặc chèn phần tử linh hoạt.



*Hình 2. Minh họa cấu trúc Double Linked List*

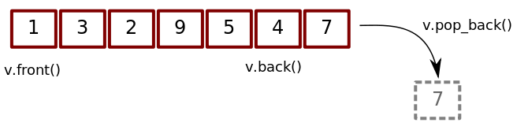
* 1. **Vector :**

**Vector** là một cấu trúc dữ liệu tuyến tính động được cung cấp bởi thư viện STL (Standard Template Library) của C++ và các ngôn ngữ lập trình khác cũng có các kiểu dữ liệu tương tự (như List trong Python). Vector hoạt động tương tự như một mảng động (dynamic array), có khả năng tự động điều chỉnh kích thước khi số phần tử thay đổi.

Vector có một số tính chất nổi bật:

* **Truy cập ngẫu nhiên**: Các phần tử của Vector có thể truy cập bằng chỉ số (index) với thời gian truy cập O(1), giống như mảng.
* **Kích thước động**: Khi cần thêm hoặc xóa phần tử, Vector sẽ tự động điều chỉnh kích thước mà không cần lập trình viên quản lý bộ nhớ thủ công.
* **Hiệu suất**: Thêm phần tử vào cuối Vector rất nhanh với thời gian trung bình O(1). Tuy nhiên, việc chèn hoặc xóa phần tử ở giữa có thể chậm hơn do phải dời các phần tử khác.

Vector là lựa chọn phổ biến khi cần lưu trữ dữ liệu với kích thước thay đổi nhưng vẫn muốn giữ khả năng truy cập ngẫu nhiên nhanh chóng. Vector thích hợp cho các bài toán cần khả năng truy cập nhanh và dễ dàng thay đổi kích thước bộ nhớ.



## *3. Minh họa Vector*

# TRIỂN KHAI CÀI ĐẶT

## Ngôn ngữ lập trình và thư viện

* **Ngôn ngữ lập trình:** C++

**C++** là một ngôn ngữ lập trình mạnh mẽ, đa năng và hướng đối tượng, được phát triển bởi Bjarne Stroustrup vào đầu thập niên 1980, với mục tiêu mở rộng và nâng cao ngôn ngữ **C**. C++ thừa hưởng cú pháp và tốc độ của C nhưng bổ sung thêm các tính năng mới, đặc biệt là lập trình hướng đối tượng (OOP), giúp dễ dàng xây dựng các ứng dụng phức tạp hơn.

*Các đặc điểm nổi bật của C++:*

* **Hướng đối tượng**: C++ hỗ trợ các khái niệm lập trình hướng đối tượng như lớp (class), kế thừa (inheritance), đa hình (polymorphism), và đóng gói (encapsulation), giúp tổ chức mã nguồn dễ bảo trì, mở rộng.
* **Hiệu suất cao**: C++ là một ngôn ngữ biên dịch (compiled language), có khả năng tối ưu hóa và cho phép thao tác trực tiếp với bộ nhớ, giúp nó có hiệu suất cao, phù hợp với các ứng dụng đòi hỏi tốc độ như game, đồ họa và hệ thống nhúng.
* **Thư viện phong phú**: Thư viện chuẩn của C++ (STL - Standard Template Library) cung cấp các cấu trúc dữ liệu và thuật toán cơ bản như vector, danh sách liên kết, hàng đợi, ngăn xếp, v.v. Giúp lập trình viên xử lý các bài toán phức tạp dễ dàng hơn.
* **Đa dạng phong cách lập trình**: C++ hỗ trợ nhiều phong cách lập trình như lập trình thủ tục, lập trình hướng đối tượng, lập trình hàm (functional programming) và lập trình tổng quát (generic programming), giúp đáp ứng nhiều nhu cầu phát triển khác nhau.
* **Thư viện:** Sử dụng các thư viện trong C++

## Tổ chức chương trình và đóng gói

* **Tổ chức chương trình:**
* Viết chương trình C++ bằng DEVC++ hoặc Visual Studio Code, mỗi đoạn code xử lý một chức năng.
* Lưu trữ code bằng Github.
* **Đóng gói:**
* Các phần code và liên quan đến chương trình đóng gói ở Github (Link: https://github.com/LGiangzz05/btlDSA)
* Bài báo cáo: Báo cáo (Lại Trường Giang – 231220753).docx

# PHÂN TÍCH CHƯƠNG TRÌNH

# Sử dụng Single Linked List để tạo chương trình quản lý nhân sự : (Phần A)

### Xây dựng class cán bộ và hàm khởi tạo :

|  |
| --- |
| class CanBo {  private:  string maCanBo;  string hoTen;  string phongBan;  string chucVu;  float heSoLuong;  public:  CanBo(  string ma = "",  string ten = "",  string phong = "",  string chuc = "",  float hsl = 0.0  )  : maCanBo(ma), hoTen(ten), phongBan(phong), chucVu(chuc), heSoLuong(hsl) {}  float tinhLuong() const {  return heSoLuong \* 1350000;  }  string getMaCanBo() const { return maCanBo; }  string getHoTen() const { return hoTen; }  string getPhongBan() const { return phongBan; }  string getChucVu() const { return chucVu; }  float getHeSoLuong() const { return heSoLuong; }  }; |

### Quản lý danh sách nhân sự sử dụng **Single Linked List** và sắp xếp bằng cách sử dụng **Vector** :

Lớp này quản lý một danh sách liên kết các đối tượng Canbo. Đây là lớp điều hành, thực hiện các thao tác như đọc dữ liệu đầu vào từ file thêm, sử dụng **Vector** tạm thời để dễ sắp xếp, tìm kiếm cán bộ theo các tiêu chí khác nhau, và in danh sách cán bộ.

|  |
| --- |
| class DanhSachCanBo {  private:  struct Node {  CanBo data;  Node\* next;  Node(CanBo cb) : data(cb), next(nullptr) {}  };  Node\* head;    void inThongTinCanBo(const CanBo& cb) const {  cout << "Ma CB: " << cb.getMaCanBo() << ", Ho Ten: " << cb.getHoTen()  << ", Phong Ban: " << cb.getPhongBan() << ", Chuc Vu: " << cb.getChucVu()  << ", He So Luong: " << cb.getHeSoLuong() << ", Luong: " << cb.tinhLuong()  << endl;  };  public:  DanhSachCanBo() : head(nullptr) {}  void taoDanhSachTuFile() {  ifstream file("input.txt");  if (!file.is\_open()) {  cerr << "Khong the mo file" << endl;  return;  }  int n;  file >> n;  file.ignore();  for (int i = 0; i < n; ++i) {  string ma, hoTen, phongBan, chucVu;  float heSoLuong;  getline(file, ma);  getline(file, hoTen);  getline(file, phongBan);  getline(file, chucVu);  file >> heSoLuong;  file.ignore();  CanBo cb(ma, hoTen, phongBan, chucVu, heSoLuong);  themCanBo(cb);  }  file.close();  }  void themCanBo(const CanBo& cb, int viTri = -1) {  Node\* newNode = new Node(cb);  if (!head || viTri == 0) {  newNode->next = head;  head = newNode;  return;  }  Node\* curr = head;  int pos = 1;  while (curr->next && pos < viTri) {  curr = curr->next;  pos++;  }  newNode->next = curr->next;  curr->next = newNode;  }  void inCanBoHeSoLuongLonHon44() const {  Node\* curr = head;  while (curr) {  if (curr->data.getHeSoLuong() >= 4.4) {  inThongTinCanBo(curr->data);  }  curr = curr->next;  }  }  void timCanBoTheoChucVu(const string& chucVu) const {  Node\* curr = head;  while (curr) {  if (curr->data.getChucVu() == chucVu) {  inThongTinCanBo(curr->data);  }  curr = curr->next;  }  }  void timCanBoTheoHeSoVaPhongBan(float heSo, const string& phongBan) const {  Node\* curr = head;  while (curr) {  if (curr->data.getHeSoLuong() == heSo && curr->data.getPhongBan() == phongBan) {  inThongTinCanBo(curr->data);  }  curr = curr->next;  }  }  void sapXepTheoTen() {  vector<CanBo> canBos;  Node\* curr = head;  while (curr) {  canBos.push\_back(curr->data);  curr = curr->next;  }  sort(canBos.begin(), canBos.end(), [](const CanBo& a, const CanBo& b) {  return a.getHoTen() < b.getHoTen();  });  curr = head;  for (const auto& cb : canBos) {  curr->data = cb;  curr = curr->next;  }  }  void timCanBoLuongCaoNhat() const {  if (!head) {  cout << "Danh sach trong.\n";  return;  }  Node\* curr = head;  Node\* maxNode = head;  while (curr) {  if (curr->data.tinhLuong() > maxNode->data.tinhLuong()) {  maxNode = curr;  }  curr = curr->next;  }  cout << "Can bo co luong cao nhat:\n";  inThongTinCanBo(maxNode->data);  }  void timCanBoLuongThapNhat() const {  if (!head) {  cout << "Danh sach trong.\n";  return;  }  Node\* curr = head;  Node\* minNode = head;  while (curr) {  if (curr->data.tinhLuong() < minNode->data.tinhLuong()) {  minNode = curr;  }  curr = curr->next;  }  cout << "Can bo co luong thap nhat:\n";  inThongTinCanBo(minNode->data);  }  void inDanhSach() const {  Node\* curr = head;  while (curr) {  inThongTinCanBo(curr->data);  curr = curr->next;  }  }  }; |

* + 1. Xây dựng lớp App dùng để chứa Menu và chạy ứng dụng :

|  |
| --- |
| class App {  public:  void luachon() {  DanhSachCanBo ds;  int choice;  do {  cout << "Menu:\n";  cout << "1. Tao danh sach can bo tu file\n";  cout << "2. In danh sach can bo\n";  cout << "3. Them can bo\n";  cout << "4. Tinh luong can bo\n";  cout << "5. In can bo co he so luong >= 4.4\n";  cout << "6. Tim can bo theo chuc vu\n";  cout << "7. Tim can bo theo he so luong va phong ban\n";  cout << "8. Sap xep can bo theo ten\n";  cout << "9. Tim can bo co luong cao nhat\n";  cout << "10. Tim can bo co luong thap nhat\n";  cout << "0. Thoat\n";  cout << "Nhap lua chon: ";  cin >> choice;  switch (choice) {  case 1:  ds.taoDanhSachTuFile();  break;  case 2:  ds.inDanhSach();  break;  case 3: {  CanBo cb;  string ma, ten, phong, chuc;  float hsl;  cout << "Nhap ma can bo: ";  cin.ignore(); getline(cin, ma);  cout << "Nhap ho ten: "; getline(cin, ten);  cout << "Nhap phong ban: "; getline(cin, phong);  cout << "Nhap chuc vu: "; getline(cin, chuc);  cout << "Nhap he so luong: "; cin >> hsl;  cb = CanBo(ma, ten, phong, chuc, hsl);  int viTri;  cout << "Nhap vi tri can them (-1 de them vao cuoi): ";  cin >> viTri;  ds.themCanBo(cb, viTri);  break;  }  case 4:  cout << "Luong cua cac can bo:\n";  ds.inDanhSach();  break;  case 5:  ds.inCanBoHeSoLuongLonHon44();  break;  case 6: {  string chucVu;  cout << "Nhap chuc vu can tim: ";  cin.ignore(); getline(cin, chucVu);  ds.timCanBoTheoChucVu(chucVu);  break;  }  case 7: {  float heSo;  string phongBan;  cout << "Nhap he so luong: ";  cin >> heSo;  cout << "Nhap phong ban: ";  cin.ignore(); getline(cin, phongBan);  ds.timCanBoTheoHeSoVaPhongBan(heSo, phongBan);  break;  }  case 8:  ds.sapXepTheoTen();  break;  case 9:  ds.timCanBoLuongCaoNhat();  break;  case 10:  ds.timCanBoLuongThapNhat();  break;  case 0:  cout << "Thoat chuong trinh.\n";  break;  default:  cout << "Lua chon khong hop le!\n";  }  } while (choice != 0);  }  }; |

Code đầy đủ có thể xem đầy đủ ở Github.

Link: https://github.com/LGiangzz05/btlDSA/blob/main/BTLphana(2).cpp

## Cài đặt cấu trúc dữ liệu Double Linked List (Phần B) :

Với cấu trúc dữ liệu Double Linked List, chúng ta cài đặt bằng 2 bước sau :

* + 1. Khởi tạo cấu trúc Node :

Một Node trong danh sách liên kết đôi bao gồm Previous Pointer, Data và Next Pointer.

|  |
| --- |
| struct Node {  float data;  Node\* next;  Node\* prev;  Node(float value) : data(value), next(nullptr), prev(nullptr) {}  }; |

* + 1. Khởi tạo thêm một class DoubleList :

#### Lớp này được sử dụng để quản lý một danh sách các phần tử mà mỗi phần tử có thể được truy cập từ hai chiều: chiều xuôi (từ đầu đến cuối) và chiều ngược (từ cuối về đầu). Cài đặt các bộ duyệt xuôi và ngược có các thao tác cơ bản thêm bớt ở 2 đầu, chèn và xóa theo bộ lặp, ngoài ra còn có các thao tác bổ sung theo yêu cầu đề bài.

|  |
| --- |
| class DoubleList {  private:  Node\* head;  Node\* tail;  public:  DoubleList() : head(nullptr), tail(nullptr) {}  // Them phan tu vào dau danh sach  void push\_front(float value) {  Node\* newNode = new Node(value);  newNode->next = head;  if (head) head->prev = newNode;  head = newNode;  if (!tail) tail = head;  }  // Them phan tu vao cuoi danh sach  void push\_back(float value) {  Node\* newNode = new Node(value);  newNode->prev = tail;  if (tail) tail->next = newNode;  tail = newNode;  if (!head) head = tail;  }  // Xoa phan tu dau danh sach  void pop\_front() {  if (!head) return;  Node\* temp = head;  head = head->next;  if (head) head->prev = nullptr;  else tail = nullptr;  delete temp;  }  // Xoa phan tu cuoi danh sach  void pop\_back() {  if (!tail) return;  Node\* temp = tail;  tail = tail->prev;  if (tail) tail->next = nullptr;  else head = nullptr;  delete temp;  }  // Xoa phan tu trung nhau, giu lai phan tu dau tien  void remove\_duplicates() {  unordered\_set<float> seen;  Node\* current = head;  while (current) {  if (seen.count(current->data)) {  Node\* Delete = current;  if (Delete->prev) Delete->prev->next = Delete->next;  if (Delete->next) Delete->next->prev = Delete->prev;  if (Delete == head) head = Delete->next;  if (Delete == tail) tail = Delete->prev;  current = current->next;  delete Delete;  } else {  seen.insert(current->data);  current = current->next;  }  }  }  // Sap xep danh sach tang dan  void sort() {  if (!head) return;  for (Node\* i = head; i->next; i = i->next) {  for (Node\* j = i->next; j; j = j->next) {  if (i->data > j->data) {  swap(i->data, j->data);  }  }  }  }  // Chen phan tu vào danh sach da sap xep  void insert\_sorted(float value) {  Node\* newNode = new Node(value);  if (!head || head->data >= value) {  push\_front(value);  return;  }  Node\* current = head;  while (current->next && current->next->data < value) {  current = current->next;  }  newNode->next = current->next;  if (current->next) current->next->prev = newNode;  newNode->prev = current;  current->next = newNode;  if (!newNode->next) tail = newNode;  }  // Ap dung ham fun vao tung phan tu trong danh sach  void transform(function<float(float)> fun) {  Node\* current = head;  while (current) {  current->data = fun(current->data);  current = current->next;  }  }  // Xuat danh sach theo chieu nguoc lai  void print\_reverse() const {  Node\* current = tail;  while (current) {  cout << current->data << " ";  current = current->prev;  }  cout << endl;  }  // Xuat danh sach theo chieu xuoi  void print\_forward() const {  Node\* current = head;  while (current) {  cout << current->data << " ";  current = current->next;  }  cout << endl;  }  }; |

* + 1. Xây dựng hàm main để thực hiện các thao tác theo yêu cầu đề bài :

|  |
| --- |
| int main() {  DoubleList list;  int n;  float x;  // Nhap danh sach so nguyen  cout << "Nhap so luong phan tu: ";  cin >> n;  cout << "Nhap cac phan tu: ";  for (int i = 0; i < n; i++) {  cin >> x;  list.push\_back(x);  }  // Xoa phan tu trung nhau  cout<< "Day sau khi xoa cac phan tu trung nhau:";  list.remove\_duplicates();  list.print\_forward();  // Bien doi tung phan tu: f(x) = (x^2 - 7x + 3) / 100  cout<< "Day sau khi bien doi tung phan tu theo f(x):";  list.transform([](float x) { return ((x\*x - 7\*x + 3)/100); });  list.print\_forward();  // Xoa phan tu trung nhau sau khi bien doi  list.remove\_duplicates();  // Sap xep danh sach tang dan  cout << "Day sau khi bien doi va sap xep tang dan:";  list.sort();  list.print\_forward();  // Nhap x và chèn vào danh sách do dãy van tang dan  cout << "Nhap phan tu can chen: ";  cin >> x;  list.insert\_sorted(x);  // Xuat danh sách theo chieu nguoc lai  cout << "Danh sach cuoi cung theo thu tu giam dan: ";  list.print\_reverse();  return 0;  } |

Code đầy đủ có thể xem ở Github.

Link: https://github.com/LGiangzz05/btlDSA/blob/main/BTLphanb.cpp

# KẾT QUẢ THỰC NGHIỆM

## Dữ liệu :

* Ở phần A,bộ dữ liệu thông tin **nhân sự** được lưu trữ trong file input.txt. Chương trình sẽ đọc dữ liệu từ file input.txt và ghi kết quả ra màn hình. Dữ liệu sẽ bao gồm các **các bộ** với Mã cán bộ, Họ đệm, tên cán bộ, Phòng ban, Chức vụ, Hệ số lương.
* Ở phần B, bộ dữ liệu đầu vào được nhập từ bàn phím.

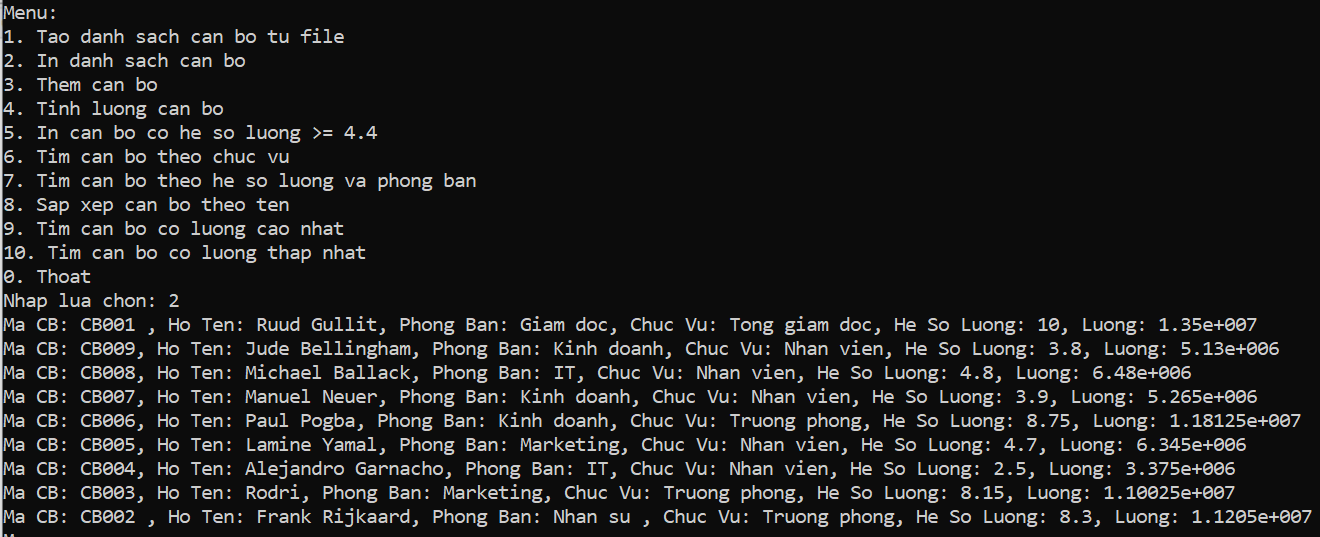
* Các chức năng trong chương trình:
* Bài toán A:
* Đọc bộ dữ liệu cán bộ từ file input.txt, in ra thông tin các cán bộ theo dạng list
* Thực hiện các thao tác quản lý nhân sự, bao gồm:
  + Tạo danh sách cán bộ.
  + Thêm 1 cán bộ vào danh sách, vị trí thêm vào do người dùng chọn, nếu không chọn thì thêm vào cuối.
  + Tính lương cho nhân viên, biết rằng: Lương = Hệ số lương \* 1350000
  + In lên màn hình tất cả cán bộ có hệ số lương >= 4.4
  + Tìm và in danh sách cán bộ theo Chức vụ
  + Tìm và in danh sách cán bộ theo hệ số lương và phòng ban
  + Sắp xếp danh sách cán bộ theo thứ tự của tên.
  + Tìm cán bộ có lương cao nhất, lương thấp nhất.
* Bài toán B:
* Xây dựng cấu trúc dữ liệu Double Linked List.
* Cài đặt các bộ duyệt xuôi và ngược có các thao tác cơ bản thêm bớt ở 2 đầu, chèn và xóa theo bộ lặp ngoài ra còn có các thao tác bổ sung sau:
* Xóa hết các phần tử xuất hiện từ lần 2 trở lên ra khỏi list để mỗi phần tử chỉ xuất hiện 1 lần
* Sắp xếp danh sách tăng dần
* Chèn thêm 1 phần tử vào danh sách đã sắp tăng để được danh sách sắp tăng dần
* Tác động 1 hàm fun(x) vào từng phần tử x của list để biến đổi list

## Các kết quả:

* **Kết quả phần A :**

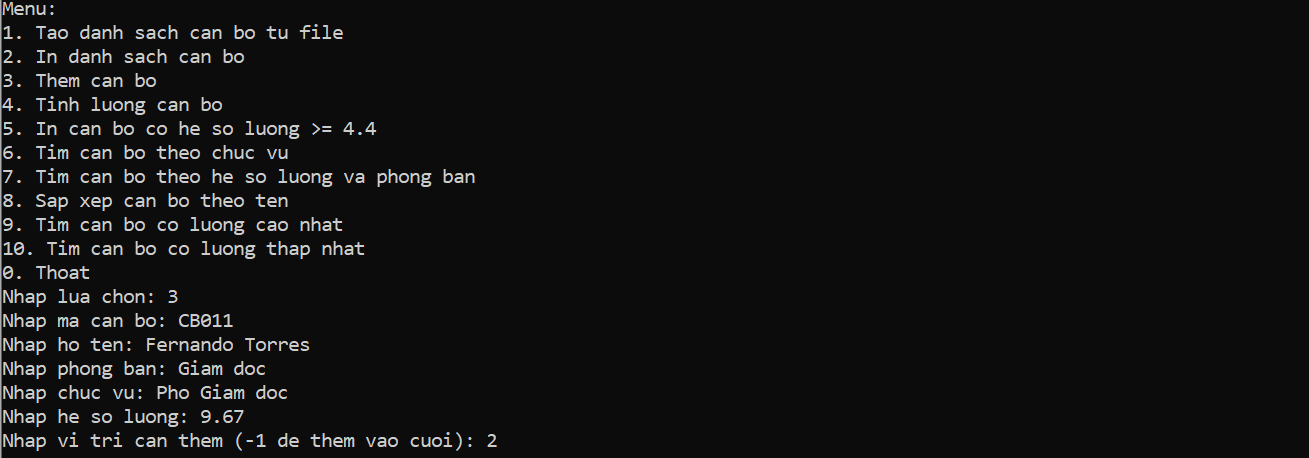
Tính năng 1: Đọc file input.

Tính năng 2: In danh sách cán bộ :

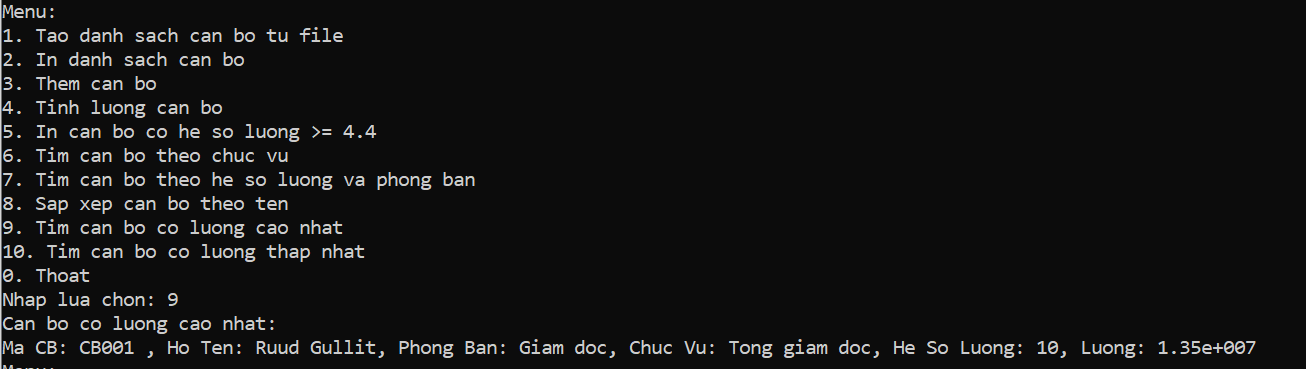


Hình 4. In ra màn hình danh sách nhân sự

Tính năng 3: Thực hiện các thao tác quản lý :



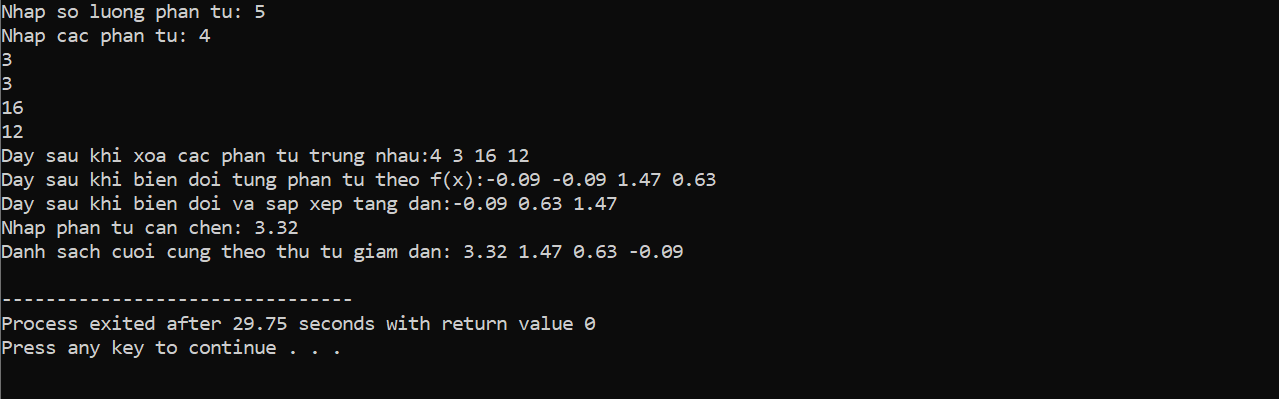
Hình 5 Thêm nhân sự mới



*Hình 6 Tìm cán bộ có lương cao nhất*

* **Kết quả phần B:**

In ra màn hình dãy sắp xếp theo thứ tự giảm dần sau khi đã thực hiện các thao tác theo yêu cầu đề bài :



*Hình 7 In ra màn hình kết quả*

# KẾT LUẬN

## Đánh giá mức độ hoàn thành :

## Qua quá trình nghiên cứu và thực hiện báo cáo môn Cấu trúc dữ liệu và Giải thuật, em đã hoàn thành các yêu cầu đề ra, bao gồm việc tìm hiểu lý thuyết về các cấu trúc dữ liệu cơ bản, ứng dụng thuật toán vào giải quyết các bài toán, và triển khai một số ví dụ minh họa. Nhờ đó, chúng em có cơ hội áp dụng kiến thức vào thực tế và củng cố hiểu biết về những khái niệm trọng tâm của môn học.

## Bài học rút ra :

## Thực hiện báo cáo này giúp em rèn luyện tư duy giải thuật, nhận thức rõ tầm quan trọng của việc lựa chọn cấu trúc dữ liệu và thuật toán phù hợp. Em cũng hiểu thêm về cách tối ưu hóa thuật toán để đạt hiệu quả cao trong quá trình xử lý dữ liệu. Qua đó, em tích lũy được nhiều kinh nghiệm quý giá, không chỉ về mặt lý thuyết mà còn về kỹ năng quản lý thời gian và giải quyết vấn đề. Từ đó, em đã nhận được 3 bài học như sau :

* Môn học Cấu trúc dữ liệu và giải thuật là vô cùng quan trọng, là phần kiến thức không thể thiếu đối với sinh viên ngành Công nghệ thông tin.
* Cần sử dụng kiến thức đã học để áp dụng vào một việc nào đó để giúp nắm vững kiến thức và tìm ra những điều mới mẻ.
* Sau khi hoàn thành sản phẩm, cần nghĩ thêm để phát triển sản phẩm, khắc phục những hạn chế của sản phẩm.

## Khó khăn với môn học : Mặc dù đạt được những kết quả nhất định, em đã gặp phải nhiều khó khăn, đặc biệt là khi làm quen với các cấu trúc dữ liệu phức tạp và thuật toán có tính toán phức tạp. Việc hiểu rõ các nguyên lý của giải thuật và áp dụng chúng vào bài toán thực tế đòi hỏi nhiều thời gian nghiên cứu và thực hành.

* Để học tốt môn Cấu trúc dữ liệu và giải thuật, cần có sự chuẩn bị tốt từ các môn học nền tảng như Kỹ thuật lập trình và Lập trình hướng đối tượng, vì đây là cơ sở giúp hiểu và áp dụng các thuật toán một cách hiệu quả.
* Lượng kiến thức trong môn học này rất lớn, cần thời gian để nắm vững và áp dụng vào các bài toán thực tế.

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến Ths. Phạm Xuân Tích – giảng viên bộ môn Cấu trúc dữ liệu và Giải thuật, khoa Công nghệ thông tin, Trường Đại học Giao thông Vận tải. Thầy đã tận tình hướng dẫn, truyền đạt kiến thức và kỹ năng, giúp chúng em nắm vững những khái niệm nền tảng và ứng dụng các thuật toán một cách hiệu quả.

Bên cạnh đó, em cũng xin cảm ơn thầy và các bạn sinh viên trong Lớp CNTT3.K64 - Khoa Công nghệ thông tin đã hỗ trợ và tạo điều kiện tốt nhất cho em trong quá trình học tập và thực hiện báo cáo. Sự giúp đỡ và động viên của mọi người đã là nguồn động lực lớn giúp em hoàn thành báo cáo này.  
  
Sản phẩm của em vẫn còn nhiều thiếu sót do thiếu kinh nghiệm và kỹ năng thực tiễn. Em mong nhận được những nhận xét và góp ý từ thầy để có thể hoàn thiện hơn trong tương lai. Em hy vọng rằng những kiến thức và kỹ năng thu được từ môn học sẽ là nền tảng vững chắc cho quá trình học tập và làm việc sau này.

Em xin trân trọng cảm ơn và kính chúc thầy dồi dào sức khỏe, đạt nhiều thành công trong sự nghiệp giảng dạy và nghiên cứu.

*Trân trọng,*

*Lại Trường Giang*

**TÀI LIỆU THAM KHẢO:**

[1]. Đỗ Trung Kiên (Chủ biên). (2022). Sách Giáo trình Cấu trúc dữ liệu và Giải thuật**.** *Nhà xuất bản Đại học Sư phạm.* 

[2]. Michael T.Goodrich, Roberto Tamassia, David Mount. (2020). Data Structures and Algothrithms in C++. *John Wiley & Sons, Inc*.

[3]. Linked List Data Structure. Last Update 10 21, 2024 from [*https://www.geeksforgeeks.org/linked-list-data-structure*](https://www.geeksforgeeks.org/linked-list-data-structure)*/.*