

**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: Xây dựng và quản lý danh sách sinh viên.***

***Tìm hiểu cách cài đặt và ứng dụng lớp queue.***

***Giảng viên hướng dẫn:***

***TS. Phạm Xuân Tích***

***Sinh viên thực hiện:***

***Đinh Vũ Minh Quang - 231210877***

***Lớp:***

***Công nghệ thông tin 3 - K64***

**Hà Nội, tháng 11 năm 2024**

MỤC LỤC

[I. GIỚI THIỆU 4](#_Toc182035582)

[1. Lý do và động lực 4](#_Toc182035583)

[2. Định nghĩa bài toán 4](#_Toc182035584)

[3. Các nhiệm vụ và mức độ hoàn thành 5](#_Toc182035585)

[II. PHƯƠNG PHÁP LỰA CHỌN 6](#_Toc182035586)

[1. Cấu trúc dữ liệu 6](#_Toc182035587)

[Các thao tác trên Queue: 8](#_Toc182035588)

[III. TRIỂN KHAI CÀI ĐẶT 9](#_Toc182035589)

[1. Ngôn ngữ lập trình và thư viện 9](#_Toc182035590)

[2. Tổ chức chương trình và đóng gói 9](#_Toc182035591)

[IV. PHÂN TÍCH CHƯƠNG TRÌNH 9](#_Toc182035592)

[1. Viết chương trình quản lý sinh viên sử dụng cấu trúc list 9](#_Toc182035593)

[1.1 Lớp Student 9](#_Toc182035594)

[1.2 Lớp StudentList 10](#_Toc182035595)

[1.3 Lớp App 12](#_Toc182035596)

[2. Cài đặt cấu trúc dữ liệu queue 13](#_Toc182035597)

[2.1 Khởi tạo queue theo mảng 13](#_Toc182035598)

[2.2 Khởi tạo queue theo list 14](#_Toc182035599)

[3. Cài đặt cấu trúc queue cho bài toán 16](#_Toc182035600)

[V. KẾT QUẢ THỰC NGHIỆM 18](#_Toc182035601)

[1. Dữ liệu 18](#_Toc182035602)

[2. Các kết quả 18](#_Toc182035603)

[VI. KẾT LUẬN 20](#_Toc182035604)

[1. Đánh giá mức độ hoàn thành 20](#_Toc182035605)

[2. Bài học rút ra 20](#_Toc182035606)

[3. Khó khăn với môn học 21](#_Toc182035607)

[VII. LỜI CẢM ƠN 22](#_Toc182035608)

# GIỚI THIỆU

## Lý do và động lực

Để đáp ứng yêu cầu môn học cũng như mong muốn tìm hiểu, mở rộng kiến thức về ngôn ngữ lập trình nói chung và môn Cấu trúc dữ liệu và giải thuật nói riêng, em đã hướng tới đề tài với các tiêu chí:

* Hữu ích.
* Áp dụng được kiến thức đã học.
* Học hỏi được kiến thức mới.

## Định nghĩa bài toán

*Bài toán 1: Xây dựng chương trình quản lý sinh viên*

*Input:* Các thao tác:

* Thêm danh sách.
* Tìm sinh viên theo tên.
* Xoá sinh viên theo tên.
* Sắp xếp danh sách sinh viên theo điểm.
* Tìm sinh viên có điểm cao nhất.
* Tìm sinh viên có điểm thấp nhất.
* In ra danh sách sinh viên.

*Output:* Một danh sách sinh viên gồm tên sinh viên (name), mã sinh viên (ID), tuổi (age) và điểm (score).

*Bài toán 2: (Bài số 33 trong danh sách BTL)*

*Input:* Các thao tác:

* Mô phỏng hàng đợi:
* Nhập n
* Thêm phần tử vào hàng đợi
* Xoá phần tử ở hàng đợi
* In ra phần tử ở hàng đợi
* Sắp xếp dãy số
* Nhập n
* Thêm phần tử vào hàng đợi

*Output:*

* Mô phỏng hàng đợi: In ra kết quả theo mong muốn
* Sắp xếp dãy số: In ra dãy số theo đề bài

## Các nhiệm vụ và mức độ hoàn thành

|  |  |
| --- | --- |
| Nhiệm vụ | Mức độ hoàn thành |
| * Tìm hiểu cấu trúc list * Lên ý tưởng bài toán 1 * Áp dụng vào bài toán Quản lý danh sách sinh viên. | Đã hoàn thành |
| * Tìm hiểu lớp queue * Lên ý tưởng bài toán 2 * Triển khai và cài đặt cấu trúc queue * Áp dụng vào bài toán | Đã hoàn thành |
| * Làm báo cáo * Lưu trữ code bằng Github | Đã hoàn thành |

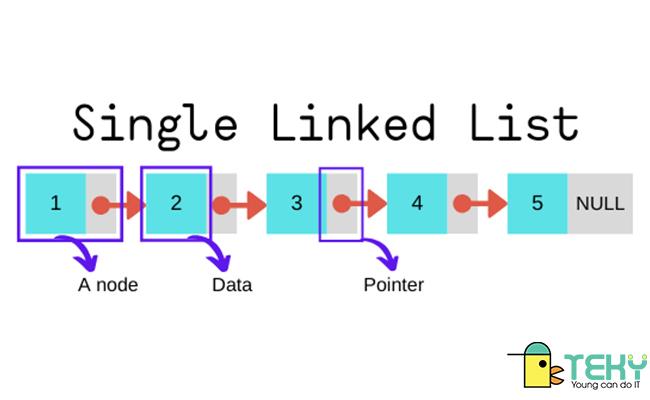
# PHƯƠNG PHÁP LỰA CHỌN

## Cấu trúc dữ liệu

* **LIST**

List trong C++ là một cấu trúc dữ liệu thuộc thư viện STL (Standard Template Library), cho phép lưu trữ các phần tử dưới dạng danh sách liên kết đôi (doubly linked list). Điểm đặc biệt của list là khả năng linh hoạt trong việc chèn và xóa các phần tử ở bất kỳ vị trí nào với độ phức tạp O(1), giúp tăng hiệu quả xử lý trong những tình huống cần thao tác với các vị trí khác nhau trong danh sách mà không cần dịch chuyển các phần tử.

Khác với mảng có kích thước cố định, list có thể tự động mở rộng hoặc thu hẹp khi các phần tử được thêm vào hoặc xóa đi, đồng thời cung cấp nhiều phương thức hữu ích như insert, erase, push\_back, pop\_back, giúp việc quản lý và thao tác dữ liệu trở nên linh hoạt hơn.



*Hình 1: Minh họa list và các phương thức cơ bản của nó.*

**Các thao tác cơ bản trên List:**

**Modifiers**

1. push\_back(): Thêm phần tử vào cuối danh sách.
2. push\_front(): Thêm phần tử vào đầu danh sách.
3. pop\_back(): Xóa phần tử cuối cùng trong danh sách.
4. pop\_front(): Xóa phần tử đầu tiên trong danh sách.
5. insert(): Chèn phần tử mới tại vị trí được chỉ định trong danh sách.
6. erase(): Xóa phần tử tại vị trí được chỉ định trong danh sách.
7. swap(): Hoán đổi nội dung của danh sách hiện tại với một danh sách khác cùng kiểu.
8. clear(): Xóa tất cả các phần tử trong danh sách.

**Iterators**

1. begin(): Đặt iterator tại phần tử đầu tiên của danh sách.
2. end(): Đặt iterator tại phần tử ngay sau phần tử cuối cùng trong danh sách.
3. rbegin(): Đặt reverse iterator tại phần tử cuối cùng và duyệt ngược về đầu danh sách.
4. rend(): Đặt reverse iterator tại vị trí trước phần tử đầu tiên của danh sách.
5. cbegin(): Đặt constant iterator tại phần tử đầu tiên của danh sách.
6. cend(): Đặt constant iterator tại phần tử cuối cùng của danh sách.
7. crbegin(): Đặt constant reverse iterator tại phần tử cuối cùng của danh sách.
8. crend(): Đặt constant reverse iterator tại phần tử đầu tiên của danh sách.

**Capacity**

1. size(): Trả về số lượng phần tử hiện có trong danh sách.
2. max\_size(): Trả về số phần tử tối đa mà danh sách có thể chứa.
3. resize(n): Thay đổi kích thước của danh sách để chứa đủ n phần tử. Nếu số phần tử hiện tại lớn hơn n, các phần tử dư sẽ bị xóa; nếu nhỏ hơn n, các phần tử mới sẽ được thêm vào.
4. empty(): Kiểm tra xem danh sách có rỗng hay không.

**Element Access**

1. front(): Trả về phần tử đầu tiên trong danh sách.
2. back(): Trả về phần tử cuối cùng trong danh sách.

* **QUEUE**

Queue là một cấu trúc dữ liệu trong C++ Standard Template Library (STL) dùng để lưu trữ và quản lý các đối tượng theo nguyên tắc "FIFO" (First In, First Out) - vào trước, ra trước. Điều này có nghĩa là phần tử đầu tiên được thêm vào queue sẽ là phần tử đầu tiên được lấy ra. Queue rất hữu ích trong các tình huống cần xử lý dữ liệu theo thứ tự, chẳng hạn như hàng đợi trong hệ thống xử lý đa nhiệm hoặc các thuật toán tìm kiếm đồ thị.

Mặc dù có chức năng tương tự nhau, queue và vector có một số điểm khác biệt. Queue không cho phép truy cập ngẫu nhiên vào các phần tử bên trong (chỉ có thể truy cập phần tử đầu hoặc cuối), và không thể thay đổi vị trí các phần tử như vector.

### Các thao tác trên Queue:

**Modifiers**

* 1. **push()**: Thêm một phần tử vào cuối queue.
  2. **pop()**: Loại bỏ phần tử ở đầu queue.
  3. **emplace()**: Tạo và thêm trực tiếp một phần tử vào cuối queue mà không cần sao chép hoặc di chuyển.
  4. **swap()**: Hoán đổi nội dung của queue hiện tại với một queue khác có cùng kiểu dữ liệu.

**Element access**

* 1. **front()**: Trả về một tham chiếu đến phần tử ở đầu queue.
  2. **back()**: Trả về một tham chiếu đến phần tử ở cuối queue.

**Capacity**

* 1. **size()**: Trả về số lượng phần tử có trong queue.
  2. **empty()**: Kiểm tra queue có rỗng hay không. Trả về true nếu rỗng, ngược lại trả về false.

# TRIỂN KHAI CÀI ĐẶT

## Ngôn ngữ lập trình và thư viện

* **Ngôn ngữ lập trình:** Sử dụng C++
* C ++ là một ngôn ngữ lập trình được phát triển bởi Bjarne Stroustrup vàonăm 1979 tại Bell Labs.
* Tính phổ biến: C++ là một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới.
* Tính thực thi nhanh
* Thư viện đầy đủ: Có rất nhiều tài nguyên cho người lập trình bằng C++
* Đa mô hình: C++ cho phép lập trình theo cấu trúc tuyến tính, hướng chức năng, hướng đối tượng đa dạng tùy theo nhu cầu người lập trình.
* **Thư viện:** Sử dụng 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: [BTL Cấu trúc dữ liệu giải thuật - GitHub](https://github.com/awithedino/BTL_CTDLGT))
* Bài báo cáo: Báo cáo Bài Tập Lớn.docx

# PHÂN TÍCH CHƯƠNG TRÌNH

## Viết chương trình quản lý sinh viên sử dụng cấu trúc list

### 1.1 Lớp Student

|  |
| --- |
| class Student {  private:      string studentID;      string fullName;      int age;      double score;  public:      Student() : studentID(""), fullName(""), age(0), score(0.0) {}      ~Student() {}      friend istream& operator>>(istream& is, Student& s) {          cin.ignore();          cout << "Ma sinh vien: ";          getline(is, s.studentID);          cout << "Ho va ten: ";          getline(is, s.fullName);          cout << "Tuoi: ";          is >> s.age;          cout << "Diem so: ";          is >> s.score;          return is;      }      friend ostream& operator<<(ostream& os, const Student& s) {          os << '|' << setw(15) << s.studentID             << '|' << setw(20) << s.fullName             << '|' << setw(5) << s.age             << '|' << setw(8) << s.score << '|';          return os;      }      bool operator<(const Student& s) const {          return this->score < s.score;      }      string getName() const {          return fullName;      }      double getScore() const {          return score;      }  }; |

### 1.2 Lớp StudentList

|  |
| --- |
| class StudentList {  private:      list<Student> studentList;  public:      StudentList() {}      ~StudentList() {}      void input() {          int n;          cout << "Nhap so luong sinh vien: ";          cin >> n;          for (int i = 0; i < n; ++i) {              Student s;              cin >> s;              studentList.push\_back(s);          }      }      void output() {          cout << '|' << setw(15) << "Ma SV"               << '|' << setw(20) << "Ho va ten"               << '|' << setw(5) << "Tuoi"               << '|' << setw(8) << "Diem" << "|\n";          for (const auto& s : studentList) {              cout << s << endl;          }      }      void sortStudents() {          studentList.sort();          output();      }      void maxScore() {          if (studentList.empty()) {              cout << "Danh sach sinh vien rong!\n";              return;          }          float max = studentList.front().getScore();          for (const auto& s : studentList)              if (max < s.getScore())                  max = s.getScore();          cout << "Sinh vien co diem cao nhat:\n";          for (const auto& s : studentList)              if (max == s.getScore())                  cout << s;      }      void minScore() {          if (studentList.empty()) {              cout << "Danh sach sinh vien rong!\n";              return;          }          float min = studentList.front().getScore();          for (const auto& s : studentList)              if (min > s.getScore())                  min = s.getScore();          cout << "Sinh vien co diem thap nhat:\n";          for (const auto& s : studentList)              if (min == s.getScore())                  cout << s;      }      bool studentExists(const string& name) {          for (const auto& s : studentList)              if (s.getName() == name)                  return true;          return false;      }      void searchByName(const string& name) {          if (studentExists(name)) {              cout << "Thong tin sinh vien:\n";              for (const auto& s : studentList)                  if (s.getName() == name)                      cout << s;          } else {              cout << "Khong tim thay sinh vien co ten: " << name << endl;          }      }      void addStudent() {          Student s;          cin >> s;          studentList.push\_back(s);      }      void deleteStudent(const string& name) {          if (studentExists(name)) {              for (auto it = studentList.begin(); it != studentList.end();) {                  if (it->getName() == name)                      it = studentList.erase(it);                  else                      ++it;              }              cout << "Da xoa sinh vien co ten: " << name << endl;          } else {              cout << "Khong tim thay sinh vien co ten: " << name << endl;          }      }  }; |

### 1.3 Lớp App

|  |
| --- |
| class App {  private:      StudentList mylist;  public:      App() {          cout << "MENU\n"               << "1. Nhap danh sach sinh vien\n"               << "2. Hien thi danh sach sinh vien\n"               << "3. Tim sinh vien theo ten\n"               << "4. Xoa sinh vien theo ten\n"               << "5. Them sinh vien\n"               << "6. Sap xep sinh vien theo diem\n"               << "7. Tim sinh vien co diem cao nhat\n"               << "8. Tim sinh vien co diem thap nhat\n"               << "0. Thoat chuong trinh\n";      }      ~App() {}      void menu() {          int choice;          do {              cout << "Nhap lua chon cua ban: ";              cin >> choice;              switch (choice) {              case 0:                  cout << "Dang thoat chuong trinh..." << endl;                  break;              case 1:                  mylist.input();                  break;              case 2:                  mylist.output();                  break;              case 3: {                  string name;                  cout << "Nhap ten sinh vien can tim: ";                  cin.ignore();                  getline(cin, name);                  mylist.searchByName(name);                  break;              }              case 4: {                  string name;                  cout << "Nhap ten sinh vien can xoa: ";                  cin.ignore();                  getline(cin, name);                  mylist.deleteStudent(name);                  break;              }              case 5:                  mylist.addStudent();                  break;              case 6:                  mylist.sortStudents();                  break;              case 7:                  mylist.maxScore();                  break;              case 8:                  mylist.minScore();                  break;              default:                  cout << "Lua chon khong hop le, vui long chon lai!" << endl;                  break;              }          } while (choice != 0);      }  }; |

## Cài đặt cấu trúc dữ liệu queue

### 2.1 Khởi tạo queue theo mảng

|  |
| --- |
| #include <bits/stdc++.h>  #ifndef QUEUE\_\_CPP  #define QUEUE\_\_CPP  using namespace std;  template<class T>  class Queue {    private:      int num, cap, first, last;      T\* arr;    public:      Queue() {          num = cap = first = last = 0;          arr = nullptr;      }      ~Queue() {          if (arr)              delete[] arr;      }      int size() {          return num;      }      bool empty() {          return num == 0;      }      T& front() {          return arr[first];      }      T& back() {          return (last == 0 ? arr[cap - 1] : arr[last - 1]);      }      void pop() {          first = (first + 1) % cap;          num--;      }      void push(T x) {          if (num == cap) {              cap = (1 + cap) \* 2;              T\* temp = new T[cap];              for (int i = 0, j = first; i < num; j++, i++) {                  temp[i] = arr[j % cap];              }              if (arr)                  delete[] arr;              arr = temp;              first = 0;              last = num;          }          arr[last] = x;          last = (last + 1) % cap;          num++;      }  };  #endif |

### 2.2 Khởi tạo queue theo list

|  |
| --- |
| #ifndef QUEUE2\_\_CPP  #define QUEUE2\_\_CPP  #include <bits/stdc++.h>  template<class T>  class Node {      T elem;      Node<T>\* next;    public:      Node(T x, Node<T>\* a = nullptr) {          elem = x;          next = a;      }      T& getElem() {          return elem;      }      void setNext(Node\* a) {          next = a;      }      Node<T>\* getNext() {          return next;      }  };  template<class T>  class Queue {      int num;      Node<T>\* first;      Node<T>\* last;    public:      Queue() {          num = 0;          first = last = nullptr;      }      bool empty() {          return num == 0;      }      int size() {          return num;      }      T& front() {          return first->getElem();      }      T& back() {          return last->getElem();      }      void push(T x) {          if (num == 0) {              first = last = new Node<T>(x);          } else {              last->setNext(new Node<T>(x));              last = last->getNext();          }          num++;      }      void pop() {          if (num == 0) return;          first = first->getNext();          num--;      }  };  #endif |

## Cài đặt cấu trúc queue cho bài toán

|  |
| --- |
| #include "queueList.cpp"  using namespace std;  void dao\_nguoc(Queue<int>& q) {      int s = q.size();      Queue<int> temp;      for (int i = 0; i < s; i++) {          for (int j = 0; j < q.size() - 1; j++) {              int t = q.front();              q.pop();              q.push(t);          }          temp.push(q.front());          q.pop();      }      while (!temp.empty()) {          q.push(temp.front());          temp.pop();      }  }  int main() {      Queue<int> B;      long long n;      cout << "Nhap so luong so can them: ";      cin >> n;      while (n < 0 || n > 105 \* 2) {          cout << "Moi ban nhap lai:" << endl;          cout << "Nhap so luong so can them: ";          cin >> n;      }      int a;      for (int i = 0; i < n; i++) {          cin >> a;          while (a < 0 || a > 109) {              cout << "Nhap lai so can them: ";              cin >> a;          }          B.push(a);      }      dao\_nguoc(B);      while (B.size()) {          cout << B.front() << " ";          B.pop();      }      return 0;  } |

# KẾT QUẢ THỰC NGHIỆM

## Dữ liệu

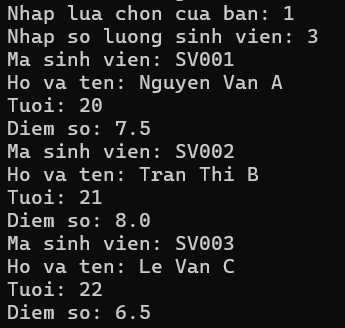
* Phần 1: Dữ liệu được nhập từ bàn phím và in ra màn hình.
* Phần 2 (Bài 33): Dữ liệu được nhập từ bàn phím và in ra màn hình

## Các kết quả

* **Các chức năng trong chương trình:**
* Phần A:
* Thêm danh sách sinh viên lưu vào StudentList.
* Đọc dữ liệu trong StudentList và in ra màn hình.
* Tìm và xóa sinh viên theo tên.
* Thêm một sinh viên mới vào StudentList.
* Tìm sinh viên có điểm cao nhất, sinh viên có điểm thấp nhất.
* Phần B (Bài 33):
* Đọc dữ liệu được nhập từ bàn phím và in ra màn hình.
* Thực hiện các thao tác: thêm, xoá dữ liệu trong hàng đợi, sắp xếp lại hàng đợi.

**Kết quả 1:**

Tính năng 1: Thêm danh sách sinh viên



*Hình 2. Nhập danh sách sinh viên từ bàn phím*

Tính năng 2: In danh sách sinh viên ra màn hình

A black screen with white text

Description automatically generated

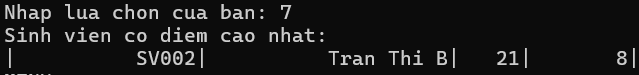
*Hình 3. Chương trình in danh sách sinh viên ra màn hình*

Một vài tính năng khác:

A screen shot of a black and white screen

Description automatically generated

*Hình 4. Tính năng sắp xếp sinh viên theo điểm số*

**

*Hình 5. Tính năng tìm sinh viên có điểm cao nhất*

*A black background with white text

Description automatically generated*

*A black screen with white text

Description automatically generated*

*Hình 6 + 7. Xoá sinh viên và kết quả sau khi xoá*

* **Kết quả 2:**

Chương trình đọc dữ liệu từ input, và mô phỏng lại cách hoạt động của queue, và giải bài toán theo đề bài.

A screenshot of a black screen

Description automatically generated

*Hình 8. Thực hiện mô phỏng cách hoạt động của queue*

A black background with white text

Description automatically generated

*Hình 9. Ví dụ bài toán*

# KẾT LUẬN

## Đánh giá mức độ hoàn thành

* Hoàn thành đúng với mục tiêu ban đầu đề ra.
* Đã ứng dụng được kiến thức của môn học, kết hợp với một số kiến thức tự tìm hiểu để làm sản phẩm.
* Cần nghiên cứu thêm kiến thức để hình thức sản phẩm trở thành một web hay app đặc thù.

## Bài học rút ra

* 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 khoa 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

* Để học tốt môn Cấu trúc dữ liệu và giải thuật cần nắm vững kiến thức môn Kỹ thuật lập trình và Hướng đối tượng.
* Cần hiểu bản chất.
* Lượng kiến thức lớn.

# LỜI CẢM ƠN

Cấu trúc dữ liệu và giải thuật là môn học rất quan trọng đối với sinh viên khoa Công nghệ thông tin. Em xin chân thành cảm ơn sự chỉ dạy nhiệt tình và những ví dụ minh họa của thầy giúp em học tập. Cảm ơn thầy vì đã giao một bài tập lớn để em có được cơ hội thử thách bản thân vào một project không chỉ đơn giản là code mà còn là tinh thần làm việc, tìm hiểu những thứ mới mẻ.

Cuối cùng, sản phẩm em vẫn còn nhiều thiếu sót do thiếu nhiều kinh nghiệm cũng như kỹ năng, mong thầy nhận xét và góp ý để em hoàn thiện hơn.