**ĐẠI HỌC ĐÀ NẴNG**
**ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**


# ĐỒ ÁN
# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

**Đề tài: Xây dựng thư viện có các thao tác của một danh sách liên kết. Viết chương trình mô phỏng hoạt động của danh sách liên kết**

**GIẢNG VIÊN HƯỚNG DẪN: Nguyễn Thị Minh Hỷ**

**SINH VIÊN THỰC HIỆN:**

**Nguyễn Xuân Tuấn LỚP: 18T1 NHÓM: 18N10.C**

**Nguyễn Anh Vũ LỚP: 18T1 NHÓM: 18N10.C**

**Đà Nẵng, 11-2020**

# LỜI MỞ ĐẦU

Giải thuật và lập trình là học phần quan trọng đối với mỗi lập trình viên. Học phần này được xem là nền tảng của lập trình máy tính. Nó là cơ sở vững chắc để giải quyết rất nhiều bài toán, đồng thời cung cấp cho chúng ta hiểu biết thêm về các giải thuật tác động lên dữ liệu, cũng như  cách tổ chức dữ liệu để giải quyết các bài toán sao cho hiệu quả và tối ưu nhất.

Sau khi học xong học phần lý thuyết, em đã nghiên cứu và thực hiện đồ án này như là một cách để củng cố và mở rộng kiến thức. Thông qua quá trình thực hiện đồ án, em đã nắm bắt được những kỹ thuật quan trọng của việc xây dựng cấu trúc dữ liệu và phân tích, thiết kế giải thuật sao cho tối ưu nhất.

Danh sách liên kết đơn là một cấu trúc dữ liệu cơ bản và có rất nhiều ứng dụng. Ý tưởng của danh sách liên kết đơn khá đơn giản. Tuy nhiên, để cài đặt cấu trúc danh sách liên kết đơn một cách ngắn gọn, hiệu quả thì không phải là điều hiển nhiên. Trong đồ án này em thực hiện xây dựng các thư viện cơ bản của một danh sách liên kết. Từ đó viết chương trình mô phỏng hoạt động của danh sách liên kết.

Em xin chân thành cám ơn cô Nguyễn Thị Minh Hỷ đã tận tình giúp đỡ, giải đáp các thắc mắc của em trong quá trình thực hiện đồ án này.

# MỤC LỤC

# DANH MỤC HÌNH VẼ

# 1. GIỚI THIỆU ĐỀ TÀI

Danh sách liên kết đơn là một cấu trúc dữ liệu cơ bản và có rất nhiều ứng dụng. Ý tưởng của danh sách liên kết đơn khá đơn giản. Tuy nhiên, để cài đặt cấu trúc danh sách liên kết đơn một cách ngắn gọn, hiệu quả thì không phải là điều hiển nhiên. Trong đồ án này em thực hiện xây dựng các thư viện cơ bản của một danh sách liên kết. Từ đó viết chương trình mô phỏng hoạt động của danh sách liên kết.

# 2. CƠ SỞ LÝ THUYẾT

## 2.1. Ý tưởng

Xây dựng thư viện có các thao tác của một danh sách liên kết. Viết chương trình mô phỏng hoạt động của danh sách liên kết

## 2.2. Cơ sở lý thuyết

### a. Khái niệm:

Danh sách liên kết đơn (Single Linked List) là một cấu trúc dữ liệu động, nó là một danh sách mà mỗi phần tử đều liên kết với phần tử đúng sau nó trong danh sách. Mỗi phần tử (được gọi là một node hay nút) trong danh sách liên kết đơn là một cấu trúc có hai thành phần:

Thành phần dữ liệu: lưu thông tin về bản thân phần tử đó.

Thành phần liên kết: lưu địa chỉ phần tử đứng sau trong danh sách, nếu phần tử đó là phần tử cuối cùng thì thành phần này bằng NULL.

### b. Đặc điểm:

Do danh sách liên kết đơn là một cấu trúc dữ liệu động, được tạo nên nhờ việc cấp phát động nên nó có một số đặc điểm sau đây:

- Được cấp phát bộ nhớ khi chạy chương trình
- Có thể thay đổi kích thước qua việc thêm, xóa phần tử
- Kích thước tối đa phụ thuộc vào bộ nhớ khả dụng của RAM
- Các phần tử được lưu trữ ngẫu nhiên (không liên tiếp) trong RAM

Và do tính liên kết của phần tử đầu và phần tử đứng sau nó trong danh sách liên kết đơn, nó có các đặc điểm sau:

- Chỉ cần nắm được phần tử đầu và cuối là có thể quản lý được danh sách
- Truy cập tới phần tử ngẫu nhiên phải duyệt từ đầu đến vị trí đó
- Chỉ có thể tìm kiếm tuyến tính một phần tử

# 3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## 3.1. Phát biểu bài toán

Xây dựng thư viện có các thao tác của một danh sách liên kết. Viết chương trình mô phỏng hoạt động của danh sách liên kết.

## 3.2. Cấu trúc dữ liệu

***Các hàm:***

```
void pushNode(int value);     // them Node vao dau day
void appendNode(int value);  //them Node vao cuoi day
void addNode(int value, int position);  //them Node o vi tri position
void deleteFirstNode();  //xoa Node dau tien
void deleteNode(int position);  //xoa Node o vi tri position
int getNode(int position); //lay data o vi tri position
int searchNode(int value); //tim gia tri data trong day
int getLength(); //kiem tra do dai cua List
```

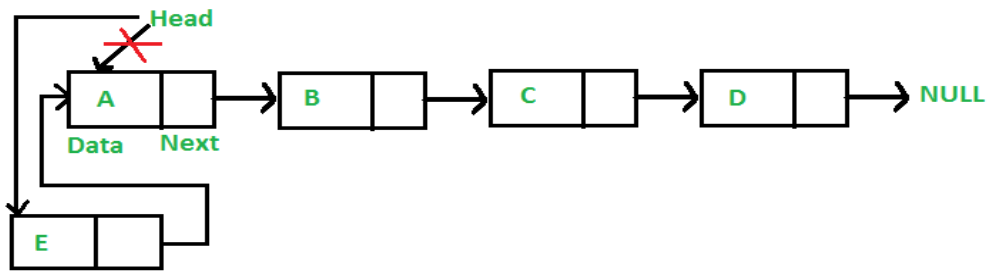
## 3.3. Thuật toán

Trình bày các thuật toán và phân tích độ phức tạp của các thuật toán.

**a. Khai báo danh sách liên kết đơn:**

```
typedef  struct LinkedList {
    int data;
    LinkedList * next;
}* node;
```

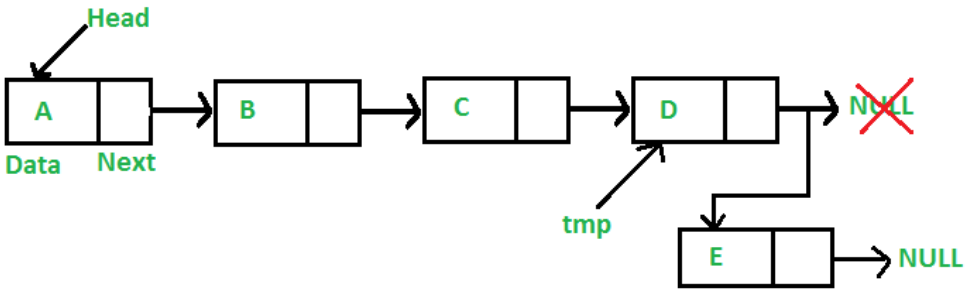
**c. Thêm 1 Node vào đầu danh sách:**




Hình 1:Mô phỏng thao tác thêm 1 Node vào đầu danh sách

```
void List::pushNode(int value)
{
    node newNode = new LinkedList;
    newNode->data = value;
    newNode->next = headNode; //chi den Node dau tien
    headNode = newNode; // chuyen Node moi thanh Node dau tien
    /*khong can check headNode
     * vi neu headNode == null thi newNode->next = headNode = null
     * headNode = newNode => headNode->next = null
     */
    length++;
}
```

**b. Thêm 1 Node vào cuối danh sách:**



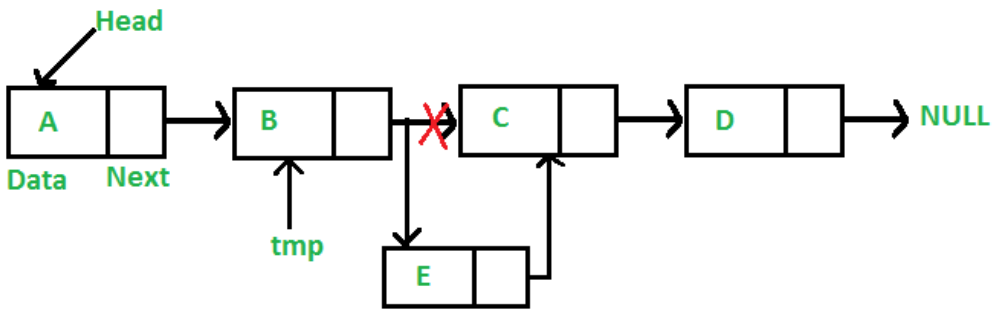

Hình 2. Mô phỏng thao tác thêm 1 Node vào cuối danh sách

```
void List::appendNode(int value)
{
    node newNode = new LinkedList;
    newNode->next = nullptr;
    newNode->data = value;
    currentNode = headNode;

    if (headNode != nullptr)
        /*kiem tra co Node dau hay chua
         *neu co thi duyet toi Node cuoi roi tro next den newNode
```

```
        *neu khong thi Node dau la newNode
        *neu khong kiem tra thi vdu headNode = null
        *check currentNode->next = headNode->next = null->next k co => loi
        */
    {
        while (currentNode->next != nullptr)
        {
            currentNode = currentNode->next;
        }
        currentNode->next = newNode;
    }
    else
    {
        headNode = newNode;
    }
    length++;
}
```

**c. Thêm 1 Node vào vị trí bất kỳ:**




Hình 3. Mô phỏng thao tác thêm 1 Node vào vị trí bất kỳ

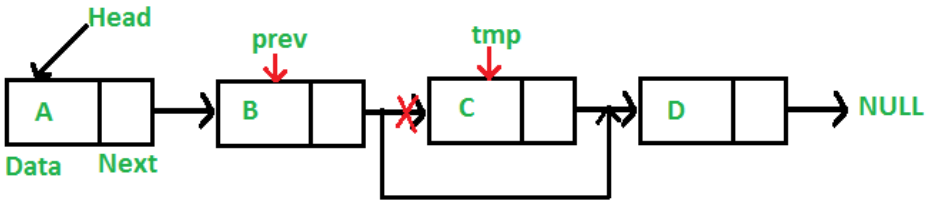
```
void List::addNode(int value, int position)
{
    if (position < 1 || position > length + 1)
    {
```

```
        cout << "invalid position" << endl;
    }
    else if (position == 1)
    {
        pushNode(value);
    }
    else if (position == length + 1)
    {
        appendNode(value);
    }
    else
    {
        node newNode = new LinkedList;
        newNode->data = value;
        currentNode = headNode;
        int i = 1;
        /*Tim den Node dung truoc vi tri position roi them newNode sau Node do
         *->position -1
         */
        while (i != position - 1)
        {
            currentNode = currentNode->next;
            i++;
        }
        tempNode = currentNode->next;
        currentNode->next = newNode;
        newNode->next = tempNode;
        length++;
    }
}
```

**d. Xoá Node đầu tiên trong danh sách liên kết:**

```
void List::deleteFirstNode()
{
   currentNode = headNode;
   headNode = headNode->next;
   delete currentNode;
   length--;
}
```

**e. Xoá 1 Node trong danh sách liên kết:**




Hình 4. Mô phỏng thao tác xoá 1 Node ở vị trí bất kỳ

```
void List::deleteNode(int position)
{
   if (position < 1 || position > length)
   {
      cout << "Invalid position" << endl;
   }
   else
   {
      if (headNode != nullptr) //check co Node nao chua
      {
         if (position == 1)// xoa Node dau tien
         {
            deleteFirstNode();
         }
```

```
            else
            {
                currentNode = headNode;
                int i = 1;
                while (i != position - 1) //Tim den Node dung truoc Node can xoa
                {
                    currentNode = currentNode->next;
                    i++;
                }
                tempNode = currentNode->next; //tempNode la Node can xoa
                currentNode->next = tempNode->next;
                delete tempNode;
                length--;
            }
        }
        else
        {
            cout << "Empty list" << endl;
        }
    }
}
```

**f. Lấy giá trị ở vị trí bất kỳ**

```
int List::getNode(int position)
{
    if (position < 1 || position > length)
    {
        return -1;
    }
    else
    {
```

```
        int i = 1;
        currentNode = headNode;
        while (i != position)
        {
            currentNode = currentNode->next;
            i++;
        }
        return currentNode->data;
    }
}
```

**g. Tìm kiếm trong danh sách liên kết:**

```
int List::searchNode(int value)
{
    int position = 1;
    for (currentNode = headNode; currentNode != nullptr; currentNode = currentNode->next)
    {
        if (currentNode->data == value)
        {
            return position;
        }
        position++;
    }
    return -1; //khong tim thay
}
```

# 4. CHƯƠNG TRÌNH VÀ KẾT QUẢ

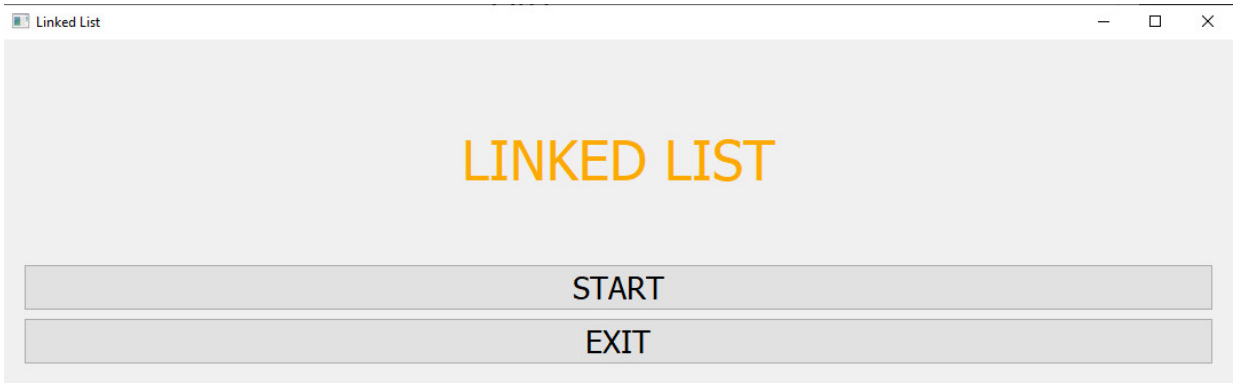
## 4.1. Ngôn ngữ cài đặt:

Chương trình được viết bằng ngôn ngữ lập trình C++ được thiết kế GUI bằng QT Framework, một ngôn ngữ phổ biến và được sử dụng rộng rãi trên toàn thế giới.

Ngoài ra, có nhiều tài liệu được viết bằng ngôn ngữ C++ nên ta có thể dễ dàng tìm kiếm và tham khảo.

## 4.2. Kết quả

### *4.2.1. Giao diện chính của chương trình*

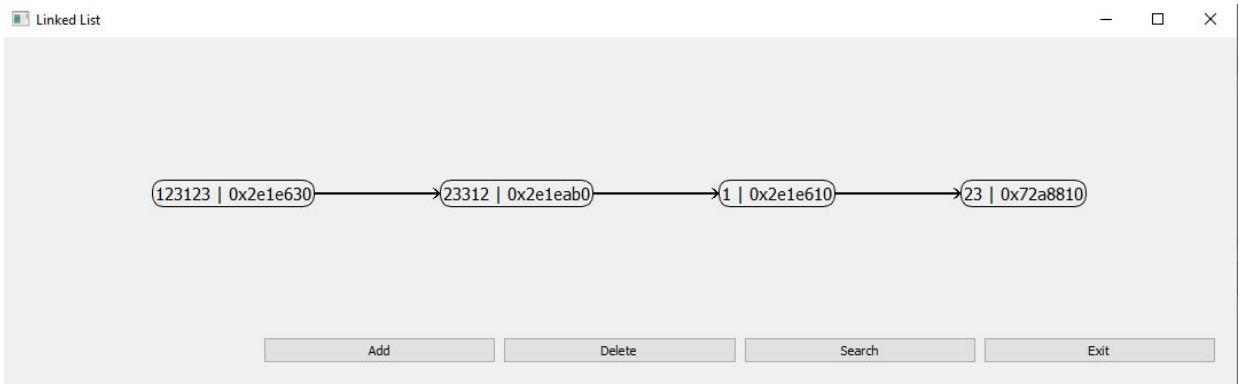



Hình 5. Giao diện của chương trình

### *4.2.2. Kết quả thực thi của chương trình*

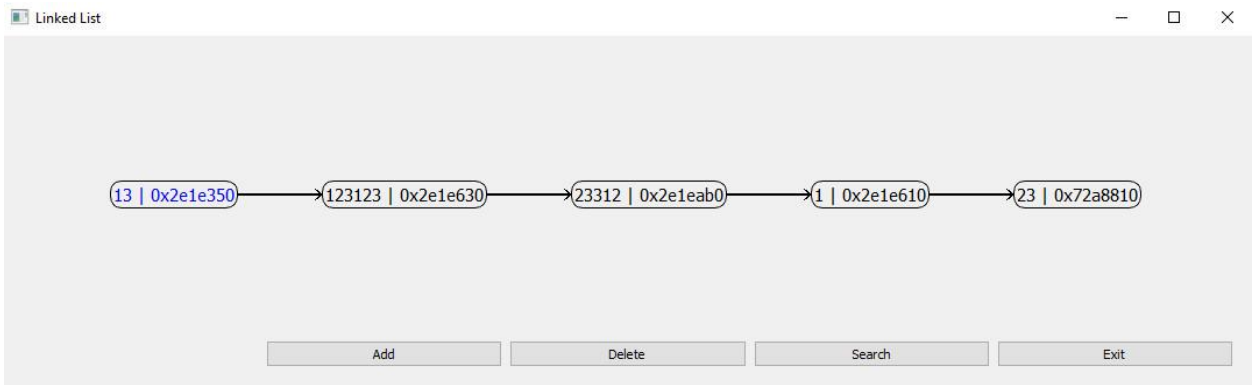
Mô tả kết quả thực hiện chương trình.

**a. Tạo một danh sách liên kết gồm n Node:**



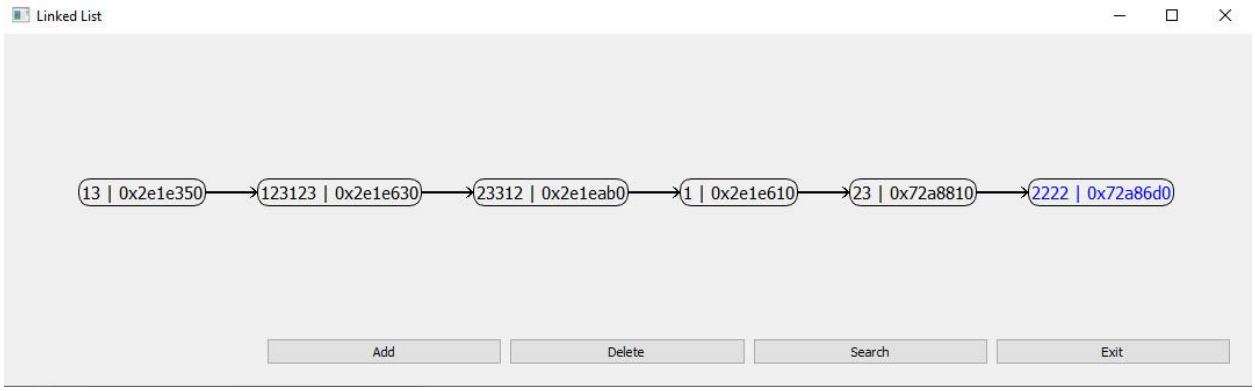

Hình 6. Giao diện sau khi tạo danh sách liên kết có n Node

**b. Chèn 1 Node vào đầu danh sách:**




Hình 7. Giao diện sau khi chèn 1 Node vào đầu danh sách

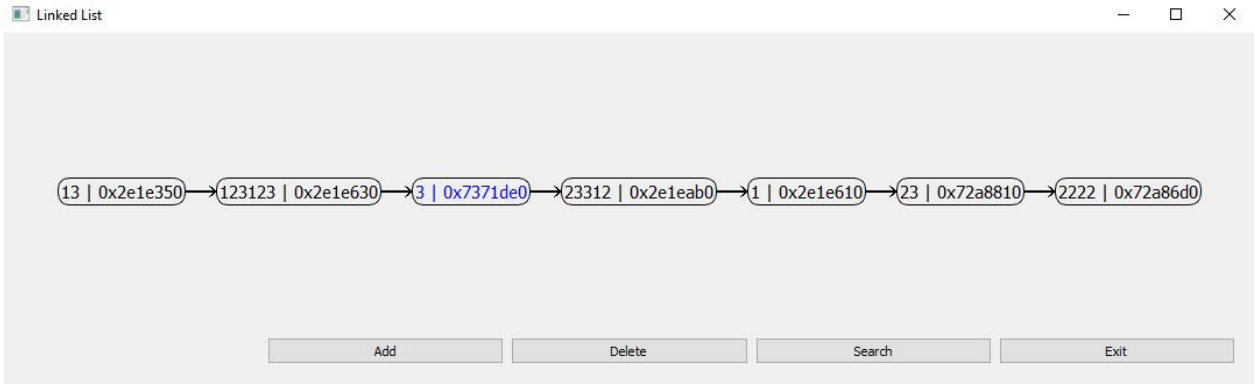
### c. Chèn 1 Node vào cuối danh sách:




Hình 8. Giao diện sau khi chèn 1 Node vào cuối danh sách

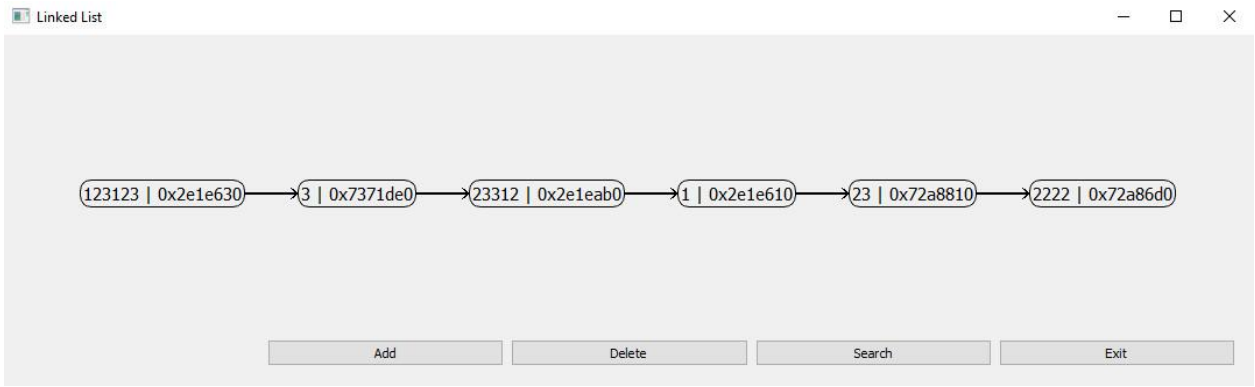
### d. Chèn 1 Node vào vị trí bất kỳ:

VD: Chèn Node ở vị trí thứ 3




Hình 9. Giao diện sau khi chèn 1 Node vào vị trí bất kỳ

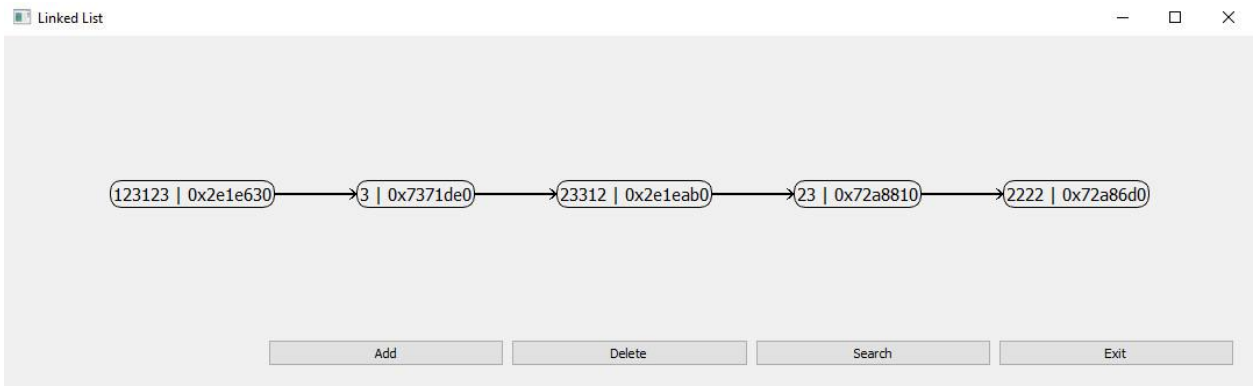
### e. Xoá Node ở đầu danh sách:




Hình 10. Giao diện sau khi xoá Node đầu danh sách

### f. Xoá Node ở vị trí bất kỳ:

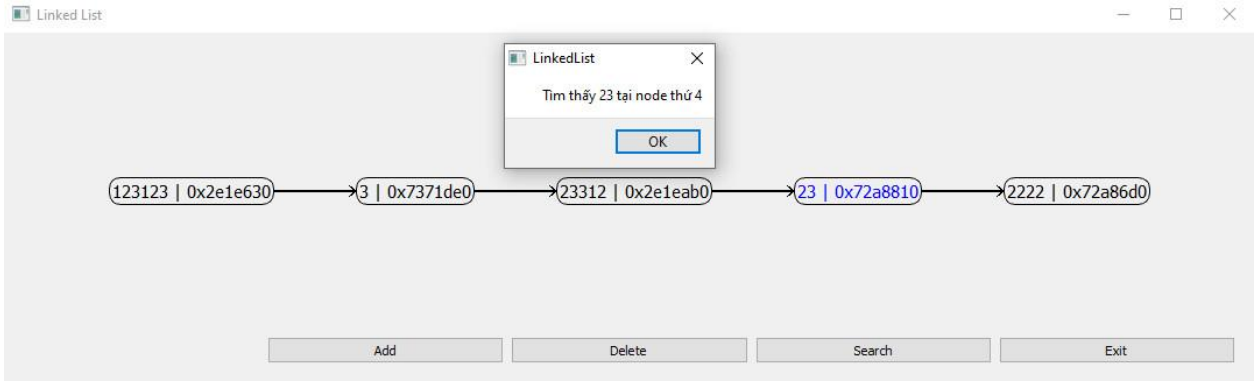
VD: Xoá Node ở vị trí thứ 4




Hình 11. Giao diện sau khi xoá 1 Node ở vị trí bất kỳ

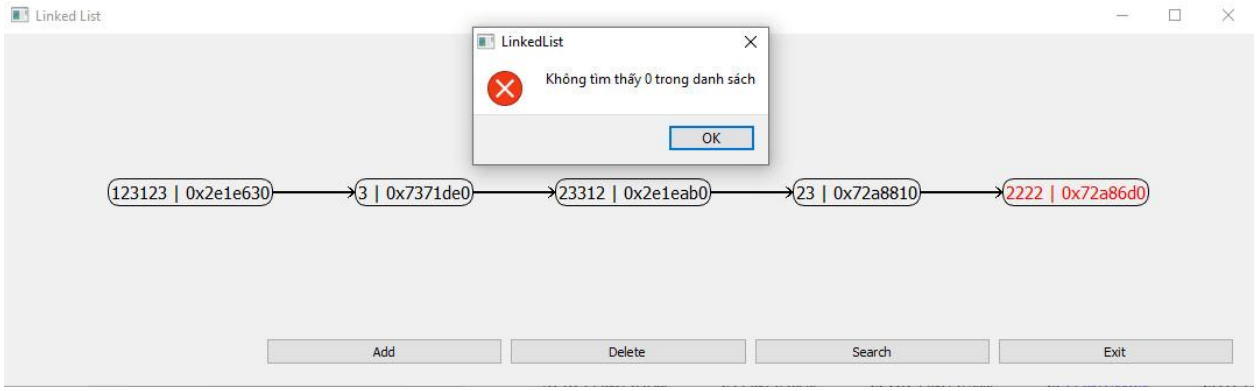
## g. Tìm kiếm trong danh sách:

VD: Tìm giá trị 23 trong danh sách




Hình 12. Giao diện sau khi tìm kiếm 1 giá trị trong danh sách

VD: Tìm giá trị 0 trong danh sách




Hình 13. Giao diện sau khi tìm kiếm 1 giá trị trong danh sách (không tìm thấy)

# 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

### a. Ưu điểm:

- Danh sách liên kết là một cấu trúc dữ liệu động, có thể tăng lên hoặc bớt đi, cấp phát hoặc thu hồi bộ nhớ trong khi chương trình đang chạy.
- Thao tác chèn và xoá các nút thực hiện một cách dễ dàng.
- Giảm thời gian truy cập và có thể mở trộng mà không cần bộ nhớ được chỉ định.

### b. Nhược điểm:

- Truy xuất tuần tự: với mảng ta có thể truy xuất đến phần tử bất kì một cách dễ dàng bằng toán tử móc vuông ([ ]), với danh sách liên kết đơn thì không đơn giản như vậy, để đến được 1 phần tử trong danh sách liên kết đơn, bắt buộc phải đi từ phần tử đầu tiên, lần theo các mối liên kết để đến được phần tử cần truy xuất. Chi phí để thực hiện công việc này là tuyến tính
- Thao tác phức tạp: Không giống với mảng, để xây dựng danh sách liên kết đơn bạn phải va chạm nhiều với con trỏ và công việc này đòi hỏi bạn phải có một sự cẩn trọng nhất định để những lỗi không mong muốn không xảy ra.

## 5.2 Hướng phát triển:

- Phát triển thêm phần giao diện cho chương trình

# TÀI LIỆU THAM KHẢO

[1] Linked List Data Structure: https://www.geeksforgeeks.org/data-structures/linked-list/

[2] Wikipedia, https://en.wikipedia.org/wiki/Linked_list/

[3] Phan Thanh Tao, “Giáo Trình Cấu Trúc Dữ Liệu” , Đại học Bách Khoa Đà Nẵng