**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

🙢 🕮 🙠



**MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**Đề tài: TRÌNH DUYỆT WEB**

**GVHD: Nguyễn Thủy An**

**Sinh viên thực hiện:**

**Nguyễn Bùi Minh Nhật 20161347**

**Nguyễn Hoàng Thanh Tùng 16110250**

**Tp. Hồ Chí Minh, tháng 11 năm 2022**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

---\*\*\*---

# **PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Họ và tên sinh viên: NGUYỄN BÙI MINH NHẬT – MSSV: 20161347

Họ và tên sinh viên: NGUYỄN HOÀNG THANH TÙNG – MSSV: 16110250

Ngành: Công nghệ thông tin

Tên đề tài: Trình duyệt web

Họ và tên Giáo viên hướng dẫn: Nguyễn Thủy An

**NHẬN XÉT**

1. Về nội dung đề tài & khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đánh giá loại:

5. Điểm: (Bằng chữ……………………. )

Tp. *Hồ Chí Minh*, ngày…,tháng…,năm 20…

Giáo viên hướng dẫn

*(Ký & ghi rõ họ tên)*

# **LỜI CẢM ƠN**

Một sinh viên dù thành công đến đâu, dù ít hay nhiều cũng luôn cần một giảng viên luôn đi cùng hỗ trợ và giúp đỡ để. Nhóm em xin được phép gửi lời cảm ơn chân thành này đến thầy Huỳnh Xuân Phụng, người đã hỗ trợ nhóm em trong suốt quá trình định chọn đề tài, hướng dẫn, nhận xét và góp ý cũng như cung cấp những lời khuyên giúp chúng em thực hiện đồ án tốt nhất. Nhờ có những chỉ dẫn của thầy đã giúp nhóm em hiểu rõ về kiến thức để thực hiện, các cách trình bày, các hướng đi, cũng như thực hiện hoàn chỉnh đồ án nên chúng em đã hoàn thành đúng tiến độ, và cả những kinh nghiệm và những kiến thức mới tụi em học hỏi được. Một lần nữa, nhóm em xin cảm ơn thầy.

Đồ án được thực hiện trong khoảng hai tháng, khoảng thời gian cũng vừa đủ để nhóm em hoàn tất được đồ án. Tuy nhiên, do tình hình dịch bệnh làm ảnh hưởng đến quá trình làm việc nhóm và do nhiều kiến thức mới, cũng như việc phân công thời gian làm việc không hợp lý nên công việc còn chưa được tối ưu gây ảnh hưởng đến đồ án, sẽ có nhiều sai sót là điều không thể tránh khỏi. Chúng em rất mong nhận được nhiều ý kiến của thầy đế giúp chúng em có thêm nhiều bài học, kinh nghiệm, cũng như kiến thức để củng cố nền tảng một cách vững chắc nhất. Chúng em xin chân thành cảm ơn !

**MỤC LỤC**

[**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN** 2](#_Toc120511525)

[**LỜI CẢM ƠN** 3](#_Toc120511526)

[**I. GIỚI THIỆU ĐỀ TÀI** 5](#_Toc120511527)

[**1. Lý do chọn đề tài** 5](#_Toc120511528)

[**2. Mục tiêu của đề tài** 5](#_Toc120511529)

[**3. Mô tả về đề tài** 5](#_Toc120511530)

[**4. Công cụ thực hiện đề tài** 5](#_Toc120511531)

[**II. CƠ SỞ LÝ THUYẾT** 6](#_Toc120511532)

[**1. Tầm quan trọng của Cấu trúc dữ liệu và giải thuật** 6](#_Toc120511533)

[**2. Tổng quan các cấu trúc dữ liệu sử dụng trong đồ án** 6](#_Toc120511534)

[**2.1. Cấu trúc danh sách liên kết đơn** 6](#_Toc120511535)

[**2.2. Cấu trúc danh sách liên kết kép** 8](#_Toc120511536)

[**III. CODE** 9](#_Toc120511537)

[**IV. Kết luận** 19](#_Toc120511538)

# **I. GIỚI THIỆU ĐỀ TÀI**

## **1. Lý do chọn đề tài**

Ngày nay, khi ngành công nghệ thông tin ngày càng phát triểm, khoa học máy tính không ngừng vươn tới những tìm tòi mới mẻ hơn. Mọi người chủ yếu làm việc đựa trên máy móc và thiết bị điện tử thì các phần mềm ứng dụng lại càng trở nên quan trọng và hữu ích hơn bao giờ hết. Tất cả thông tin hầu hết đều có cả trên những website Internet nên cần phải có các trình duyệt web để truy cập vào những website ấy. Vì vậy chúng em chọn đề tài trình duyệt web xây dựng các chức năng ở trên một trình duyệt web.

## **2. Mục tiêu của đề tài**

Sử dụng danh sách liên kết đôi tạo danh sách URL cho thanh trình duyệt

Sử dụng danh sách liên kết đơn tạo danh sách BookMark và History

Sử dụng cấu trúc dữ liệu Hash Table để bật new tab và truy cập

Xây dựng các chức năng của một trình duyệt web:

+ Nhập URL

+ Nút tới (forward)

+ Nút lùi (previous)

+ BookMark

+ History

+ New tab

Tạo menu, xử kiện bàn phím

## **3. Mô tả về đề tài**

Ở thanh trình duyệt có nhập URL sẽ tạo nên một danh sách liên kết đôi, sử dụng nút tới và nút lùi để di chuyển URL hiện tại và khi nhập một URL mới thì URL mới sẽ thêm ngay sau URL hiện tại, nếu URL hiện tại đang ở giữa danh sách thì tất cả các URL đã nhập trước đó (các URL phía sau URL hiện tại) sẽ thay thế bằng URL mới được thêm vào.

History được tạo bằng danh sách liên kết đơn, khi nhập URL sẽ tự động thêm URL đó vào History, có thể vào xem và xóa URL trong History.

BookMark cũng được tạo bằng danh sách liên kết đơn, khi đang đứng ở URL hiện tại có nút thêm URL vào BookMark, có thể vào xem và xóa URL trong BookMark.

Bật nhiều new tab, truy cập vào một tab và sử dụng các chức năng trên.

## **4. Công cụ thực hiện đề tài**

Ngôn ngữ xây dựng code là C++

Cấu trúc danh sách liên kết đơn

Cấu trúc danh sách liên kết đôi

Cấu trúc dữ liệu Hash Table

# **II. CƠ SỞ LÝ THUYẾT**

## **1. Tầm quan trọng của Cấu trúc dữ liệu và giải thuật**

Cấu trúc dữ liệu(Data Structure) là cách lập trình để lưu trữ dữ liệu để dữ liệu có thể được sử dụng một cách hiệu quả. Hầu hết mọi ứng dụng doanh nghiệp đều sử dụng nhiều kiểu cấu trúc dữ liệu khác nhau theo cách này hay cách khác, vì nó mang lại nhiều lợi ích rất lớn không chỉ cho việc lưu trữ dữ liệu.

Thuật toán(Algorithms) là một thủ tục từng bước, xác định một tập hợp các lệnh được thực hiện theo một thứ tự nhất định để có được đầu ra mong muốn. Các thuật toán thường được tạo độc lập với các ngôn ngữ cơ bản, tức là một thuật toán có thể được triển khai bằng nhiều ngôn ngữ lập trình(C/C++, Java, Python, PHP…).

Cấu trúc dữ liệu và giải thuật(CTDL & GT) là sự kết hợp và áp dụng một hoặc nhiều cấu trúc dữ liệu nào đó vào một hoặc nhiều thuật toán nào đó để có được đầu ra mong muốn một cách tối ưu và tốt nhất khi dữ liệu có số lượng cực lớn.

Khi ngày càng có nhiều ứng dụng phức tạp và nhiều dữ liệu, có nhiều vấn đề phổ biến mà các ứng dụng ấy phải đối mặt như tìm kiếm dữ liệu, tìm kiếm một sản phẩm nào đó trong nhiều dữ liệu ngày càng lớn. Khi dữ liệu phát triển, việc tìm kiếm sẽ không còn nhạy nữa. Hơn nữa khi có nhiều khối dữ liệu nạp chồng lên nhau thì sẽ cần có một cách tổ chức sắp xếp sao cho ta có thể quản lí một cách dễ dàng nhất. Đồng thời đi kèm với đó là các thuật toán để lấy dữ liệu phục vụ cho việc xử lí thông tin. Với một cấu trúc dữ liệu đã chọn, sẽ có những giải thuật tương ứng, phù hợp. Khi cấu trúc dữ liệu thay đổi thường giải thuật cũng phải thay đổi theo để tránh việc xử lí gượng ép, thiếu tự nhiên trên một cấu trúc không phù hợp. Hơn nữa, một cấu trúc dữ liệu tốt sẽ giúp giải thuật xử lý trên đó có thể phát huy tác dụng tốt hơn, vừa nhanh vừa tiết kiệm tài nguyên, giải thuật cũng đơn giản và dễ hiểu hơn.

## **2. Tổng quan các cấu trúc dữ liệu sử dụng trong đồ án**

### **2.1. Cấu trúc danh sách liên kết đơn**

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.



**Thêm phần tử vào danh sách**

- Thêm đầu: Để thêm node vào đầu danh sách, đầu tiên ta cần kiếm tra xem danh sách đó có rỗng hay không, nếu danh sách rỗng, ta chỉ cần gán head và tail của danh sách bằng node đó. Ngược lại nếu danh sách không rỗng, ta thực hiện trỏ thành phần liên kết vào head, sau đó gán lại head bằng node mới.

- Thêm cuối: Tương tự, để thêm node vào cuối danh sách, đầu tiên ta kiểm tra xem danh sách rỗng hay không, rỗng thì gán head và tail đều bằng node mới. Nếu không rỗng, ta thực hiện trỏ tail->next vào node mới, sau đó gán lại tail bằng node mới (vì bây giờ node mới thêm chính là tail).

- Thêm sau node bất kỳ: Để thêm một node p vào sau node q bất kỳ, đầu tiên ta cần kiếm tra xem node q có NULL hay không, nếu node q là NULL tức là danh sách rỗng, vậy thì ta sẽ thêm vào đầu danh sách. Nếu node q không NULL, tức là tồn tại trong danh sách, ta thực hiện trỏ p->next = q->next, sau đó q->next = p. Tiếp theo chúng ta kiểm tra xem node q trước đó có phải là node cuối hay không, nếu node q là node cuối thì thêm p vào, p sẽ thành node cuối nên ta gán lại tail = p.

**Xóa phần tử khỏi danh sách**

- Xóa đầu: Để xóa phần tử ở đầu danh sách, ta kiểm tra xem danh sách đó có rỗng hay không, nếu rỗng, ta không cần xóa, trả về kết quả là 0. Nếu danh sách không rỗng, ta thực hiện lưu node head lại, sau đó gán head bằng next của node head, sau đó xóa node head đi. Tiếp theo ta cần kiểm tra xem danh sách vừa bị xóa đi node head có rỗng hay không, nếu rỗng ta gán lại tail bằng NULL luôn sau đó trả về kết quả 1.

- Xóa ở sau node bất kỳ: Để xóa một node p sau node q bất kỳ, ta kiểm tra xem node q có NULL hay không, nếu node q NULL thì không tồn tại trong danh sách, do đó trả về 0, không xóa. Nếu node q khác NULL nhưng next của q là NULL, tức là p bằng NULL thì không xóa, trả về 0 (do sau q không có node nào cả, q là tail). Nếu node p tồn tại, ta thực hiện kiểm tra xem node p có phải là tail hay không, nếu node p là tail thì gán lại tail là q, tức là node trước đó để xóa node p đi.

**Duyệt và in danh sách**

Sau khi có các thao tác thêm, xóa, chúng ta có thể in ra danh sách để kiểm tra xem có hoạt động đúng hay không. Để in danh sách, ta duyệt từ đầu đến cuối danh sách và in ra trong lúc duyệt. Ta gán một node bằng head, sau đó kiểm tra xem node đó có NULL hay không, không thì in ra data của node đó, sau đó gán tiếp node đó bằng next của chính nó tức node đó bây giờ là node tiếp theo, cứ như vậy cho đến hết.

### **2.2. Cấu trúc danh sách liên kết đôi**

Danh sách liên kết đôi (Doubly Linked List) là một biến thể của Danh sách liên kết (Linked List), trong đó hoạt động duyệt qua các nút có thể được thực hiện theo hai chiều: về trước và về sau một cách dễ dàng khi so sánh với Danh sách liên kết đơn.



**Thêm phần tử vào danh sách**

Nguyên tắc thêm vào giống hệt danh sách liên kết đơn, chỉ có phần điều chỉnh lại liên kết là khác.

**Xóa phần tử khỏi danh sách**

Nguyên tắc xóa cơ bản giống hệt như đối với danh sách liên kết đơn.

**Duyệt và in danh sách**

Nguyên tắc duyệt và in danh sách giống hệt với danh sách liên kết đơn.

### **2.3. Cấu trúc dữ liệu Hash Table**

Cấu trúc dữ liệu Hash Table là một cấu trúc dữ liệu lưu giữ dữ liệu theo cách thức liên hợp. Trong Hash Table, dữ liệu được lưu giữ trong định dạng mảng, trong đó các giá trị dữ liệu có giá trị chỉ mục riêng. Việc truy cập dữ liệu trở nên nhanh hơn nếu chúng ta biết chỉ mục của dữ liệu cần tìm.

Do đó, với loại cấu trúc dữ liệu Hash Table này thì các hoạt động chèn và hoạt động tìm kiếm sẽ diễn ra rất nhanh, bất chấp kích cỡ của dữ liệu là bao nhiêu. Hash Table sử dụng mảng như là một kho lưu giữ trung gian và sử dụng kỹ thuật Hash để tạo chỉ mục tại nơi phần tử được chèn vào.

**Kĩ thuật Hashing**

Hashing là một kỹ thuật để chuyển đổi một dãy các giá trị khóa (key) vào trong một dãy các giá trị chỉ mục (index) của một mảng.

**Hàm băm hash function**

Hàm băm là bất kỳ hàm nào có thể được sử dụng để ánh xạ tập dữ liệu có kích thước tùy ý thành tập dữ liệu có kích thước cố định và đưa vào bảng băm. Các giá trị được trả về bởi hàm băm được gọi là giá trị băm.

Một hàm băm được đánh giá tốt nếu nó đạt được các yêu cầu cơ bản sau:

+ Dễ tính toán: Nó phải dễ tính toán và bản thân nó không phải là một thuật toán.

+ Phân bố đồng đều: Nó cần phải phân phối đồng đều trên bảng băm, không xảy ra việc tập trung thành các cụm.

+ Ít va chạm: Va chạm xảy ra khi các cặp phần tử được ánh xạ tới cùng một giá trị băm.

**Kỹ thuật xử lí va chạm**

+ Separate chaining (open hashing)

Separate chaining là một kỹ thuật xử lý và chạm phổ biến nhất. Nó thường được cài đặt với danh sách liên kết. Để lưu giữ một phần tử trong bảng băm, bạn phải thêm nó vào một danh sách liên kết ứng với chỉ mục của nó. Nếu có sự va chạm xảy ra, các phần tử đó sẽ nằm cùng trong 1 danh sách liên kết.

+ Linear probing (open addressing or closed hashing)

Trong kỹ thuật xử lý va chạm này, chúng ta sẽ không dùng linklist để lưu trữ mà chỉ có bản thân array đó thôi.

Khi thêm vào bảng băm, nếu chỉ mục đó đã có phần tử rồi; Giá trị chỉ mục sẽ được tính toán lại theo cơ chế tuần tự. Giả sử rằng chỉ mục là chỉ số của mảng, khi đó, việc tính toán chỉ mục cho phần tử được tính theo cách sau:

index = index % hashTableSize

index = (index + 1) % hashTableSize

index = (index + 2) % hashTableSize

index = (index + 3) % hashTableSize

Và cứ thế theo cách như vậy chừng nào index thu được chưa có phần tử được sử dụng. Tất nhiên, không gian chỉ mục phải được đảm bảo để luôn có chỗ cho phần tử mới.

+ Quadratic Probing

Ý tưởng cũng khá giống Linear Probing, nhưng cách tính chỉ mục có khác đôi chút:

index = index % hashTableSize

index = (index + 12) % hashTableSize

index = (index + 22) % hashTableSize

index = (index + 32) % hashTableSize

Và cứ thế cho tới khi tìm được chỉ mục trống.

+ Double Hashing

Vẫn giống 2 kỹ thuật ngay phía trên, chỉ khác ở công thức tính khi xảy ra va chạm như sau:

index = (index + 1 \* indexH) % hashTableSize;

index = (index + 2 \* indexH) % hashTableSize;

Và cứ tiếp tục cho tới khi tìm được chỉ mục chưa được sử dụng.

**Các hoạt động cơ bản trên hash table**

Dưới đây là một số hoạt động cơ bản có thể được thực hiện trên cấu trúc dữ liệu Hash Table.

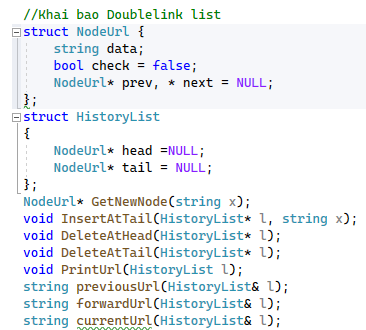
+ Hoạt động tìm kiếm: tìm kiếm một phần tử trong cấu trúc dữ liệu Hash Table.

+ Hoạt động chèn: chèn một phần tử vào trong cấu trúc dữ liệu Hash Table.

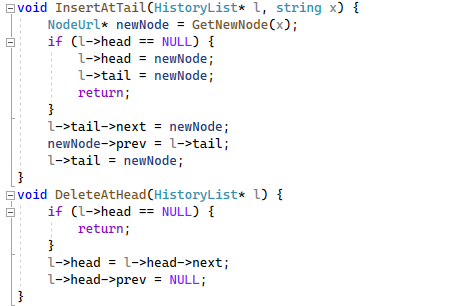
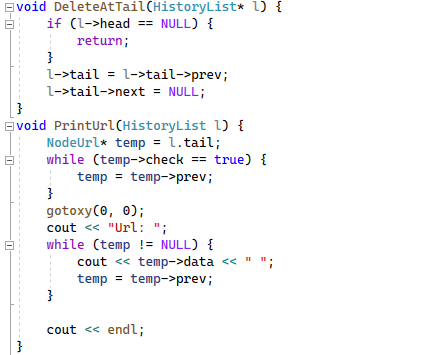
+ Hoạt động xóa: xóa một phần tử từ cấu trúc dữ liệu Hash Table.

# **III. CODE**

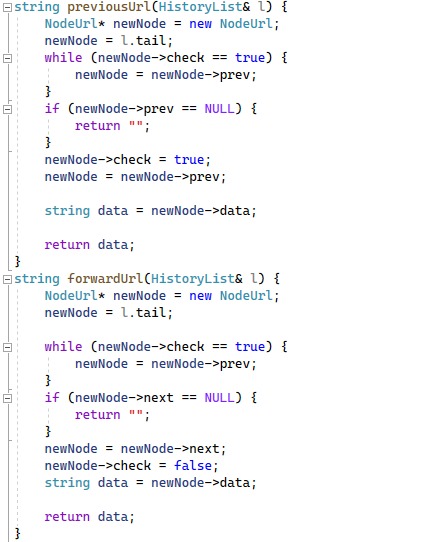
Khai báo danh sách liên kết đôi



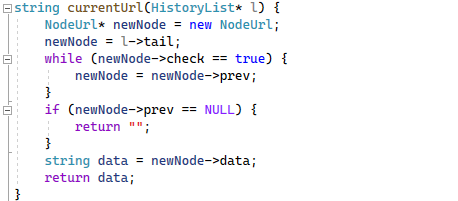
Các hàm chức năng của danh sách liên kết đôi gồm **InsertAtTail**, **DeleteAtHead**, **DeleteAtTail**, **PrintUrl**

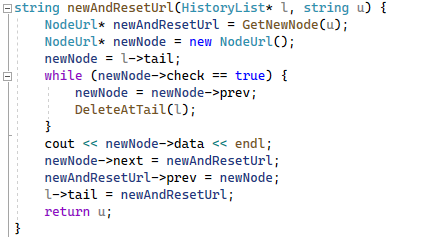
Các hàm **forwardUrl**, **previousUrl** cho chức năng nút tới, nút lùi



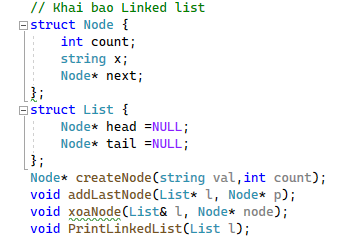
Hàm **currentUrl** để thêm Url hiện tại vào BookMark



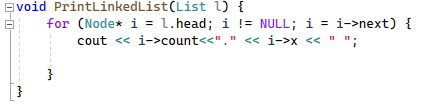
Hàm **newAndResetUrl** cho việc nhập Url mới và xóa các Url phía sau Url hiện tại

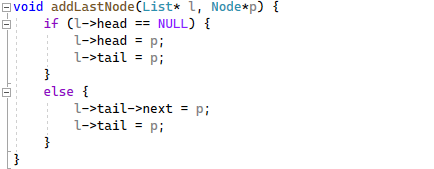


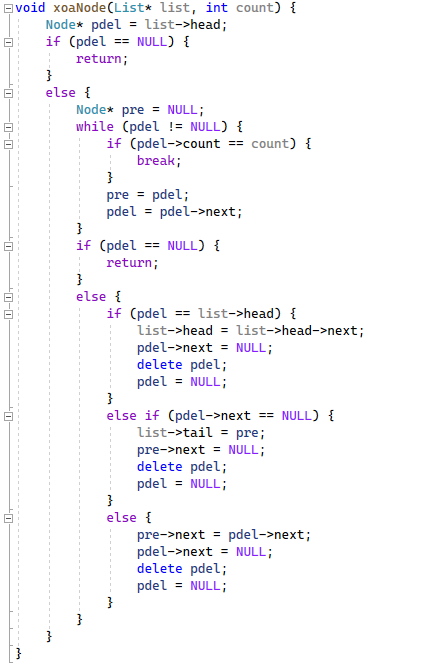
Khai báo danh sách liên kết đơn



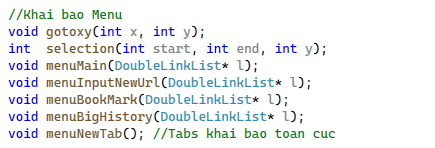
Các hàm chức năng của danh sách liên kết đơn gồm **addLastNode**, **xoaNode** và **PrintLinkedList**



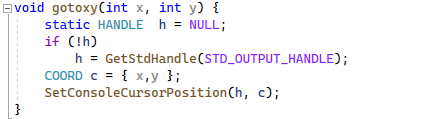


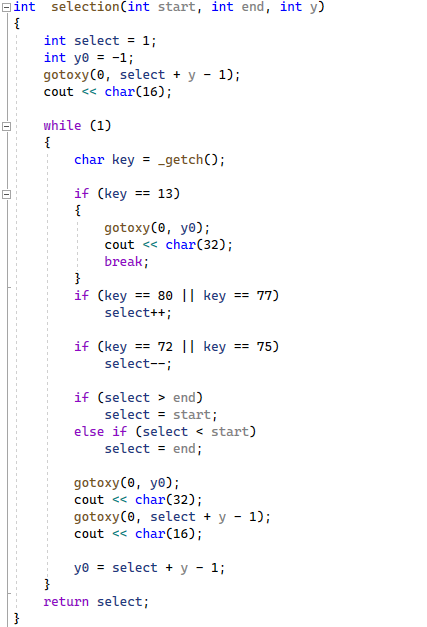


Khai báo Menu

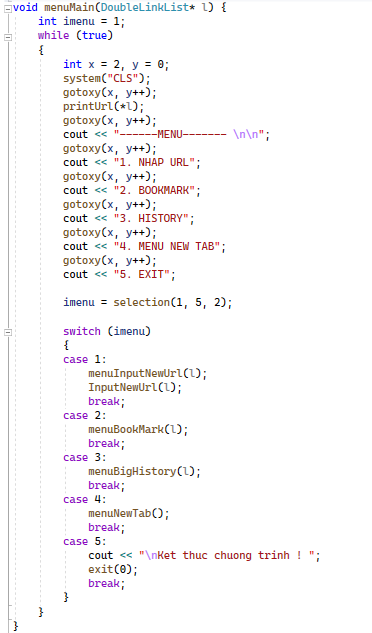


Hàm **gotoxy** và hàm **selection** dùng cho việc tạo menu và xử kiện bàn phím

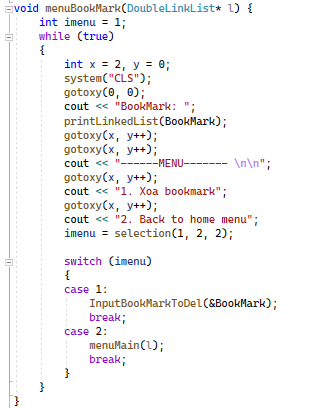




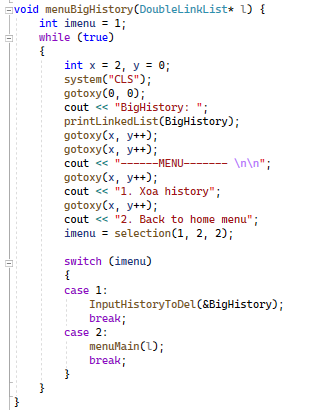
Hàm **menuMain** tạo menu chính với các option nhập Url, xem BookMark và xem History



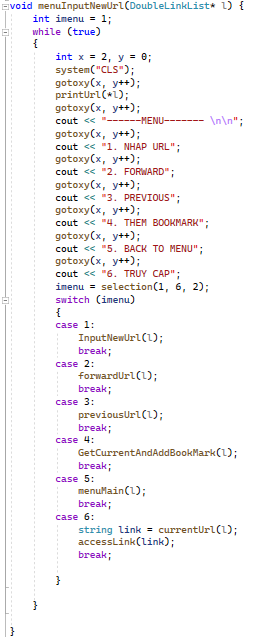
Hàm **menuBookMark** tạo menu chức năng thêm, xóa BookMark.



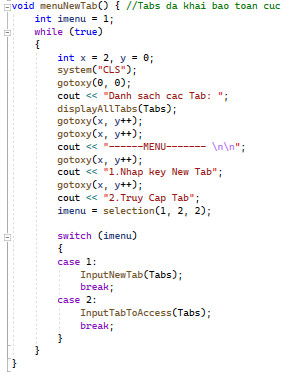
Hàm **menuBigHistory** tạo menu chức năng xóa History



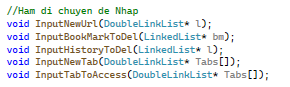
Hàm **menuInputNewURL** tạo menu với các option nhập Url, Forward, Previous, thêm BookMark



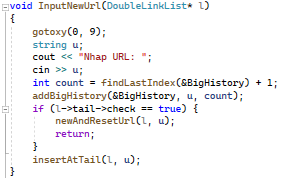
Hàm **menuNewTab** tạo menu với các option nhập key new tab, truy cập tab để bật và truy cập các tab khác



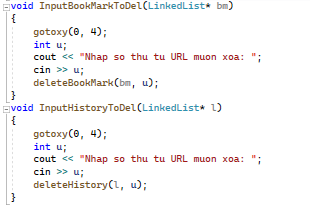
Khai báo các hàm di chuyển để nhập



Hàm **InputNewUrl** chức năng nhập Url



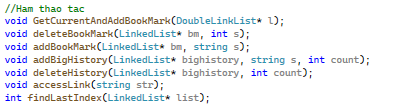
Hàm **InputBookMarkToDel** và **InputHistoryToDel** chức năng xóa BookMark và History



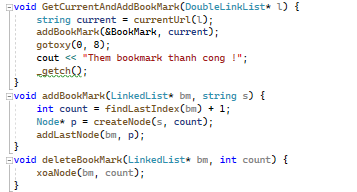
Hàm **InputNewTab** và **InputTabToAccess** chức năng bật new tab và truy cập vào tab



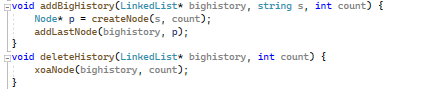
Khai báo các hàm thao tác để thực hiện các thao tác của chức năng



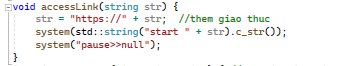
Hàm **GetCurrentAndAdDBookMark** gọi hàm thực thi việc thêm Url hiện tại vào BookMark, hàm **addBookMark** và **deleteBookMark** thực thi việc thêm và xóa BookMark



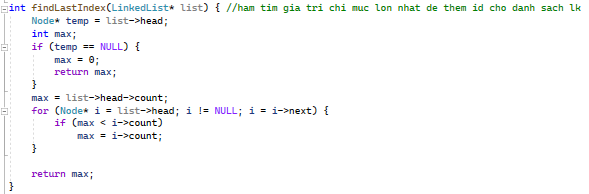
Hàm **addBigHistory** thực thi việc thêm vào History khi nhập Url, hàm **deleteHistory** thực thi việc xóa History



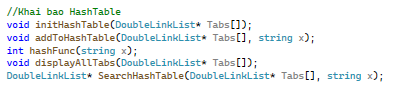
Hàm **accessLink** thực thi việc truy cập Url



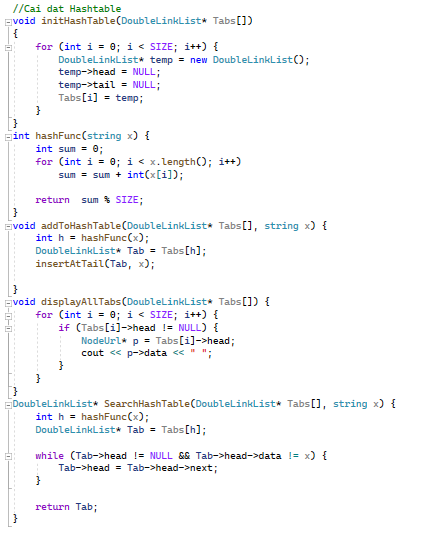
Hàm **findLastIndex** tìm giá trị chỉ mục lớn nhất trong danh sách để thêm id



Khai báo các hàm HashTable



Các hàm chức năng cho HashTable gồm **initHashTable**, **hashFunc**, **addToHashTable**, **displayAllTabs**, **SearchHashTable**



# **IV. Kết luận**

## **1. Bảng kế hoạch thực hiện đồ án**

|  |  |  |
| --- | --- | --- |
| STT | Thời gian | Công Việc |
| 1 | Tuần 11  31/10/2022 đến 7/11/2022 | Tìm hiểu về khái niệm, các đặc điểm, và thành phần cấu tạo nên URL.  Xem xét những cấu trúc có thể sử dụng được trong đồ án.  Phân công kế hoạch và tiến độ cần thực hiện. |
| 2 | Tuần 12  7/11/2022 đến 14/11/2022 | Xây dựng cấu trúc URL bằng danh sách liên kết đôi và cấu trúc History, BookMark bằng danh sách liên kết đơn.  Xây dựng các chức năng cho các danh sách |
| 3 | Tuần 13  14/11/2022 đến 21/11/2022 | Xây dựng cấu trúc dữ liệu Hash Table cho việc bật new tab.  Xây dựng menu và hoàn thiện các chức năng của một trang web. |
| 4 | Tuần 14  21/11/2021 đến 28/11/2021 | Hoàn thành báo cáo và chuẩn bị thuyết trình |

## **2. Kết quả đạt được**

Hiểu rõ kiến thức về các cấu trúc dữ liệu như danh sách liên kết đơn, danh sách liên kết đôi và Hash Table.

Xây dựng được và vận dụng được các cấu trúc dữ liệu vào trong bài toán cụ thể, lựu chọn đúng cấu trúc dữ liệu phù hợp nhất với yêu cầu đề bài.

Nâng cao tinh thần làm việc nhóm, phân công hợp lý để hoàn thành công việc hiệu quả nhất.

Từng thành viên đã có thể trình bày tự tin những ý tưởng và những thuật toán đã làm trước thầy/cô và toàn bộ lớp.

Hoàn thành một bài tập lớn bằng chính kiến thức đã học và những kiến thức tìm hiểu thêm.

## **3. Ưu điểm, nhược điểm và hướng phát triển**

Ưu điểm:

+ Đồ án xây dựng và kết hợp giữa cấu trúc danh sách liên kết đơn và danh sách liên kết đôi để tạo nên các chức năng của một trình duyệt web.

+ Sử dụng Hash Table để bật và truy cập các tab khác nhau hoạt động riêng biệt trên một trình duyệt web.

+ Các URL được liên kết và hiện ra rõ ràng giúp người dùng phân biệt rõ các mối liên kết giữa các đường đẫn URL.

Nhược điểm:

+ Giao diện còn sơ sài vì chỉ hiển thị trên console.

+ Chức năng còn vẫn khá ít và hạn chế.

Hướng phát triển:

+ Kết nối dữ liệu với SQL Server để lưu dữ lựu liên kết giữa các URL để có thể sử dụng và tái sử dụng.

+ Nâng cấp giao diện bằng đồ hoạ.

+ Thêm nhiều chức năng đáp ứng với yêu cầu.