KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2023 – 2024

ỨNG DỤNG DANH SÁCH LIÊN KẾT TRONG BÀI TOÁN QUẢN LÝ SẢN PHẨM

Giáo viên hướng dẫn:
ThS. Nguyễn Ngọc Đan Thanh

Sinh viên thực hiện:
Họ tên: Nguyễn Duy Khang
MSSV: 110121188
Lớp: DA21TTC

Trà Vinh, tháng 01 năm 2024

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2023 – 2024

ỨNG DỤNG DANH SÁCH LIÊN KẾT TRONG BÀI TOÁN QUẢN LÝ SẢN PHẨM

Giáo viên hướng dẫn:
ThS. Nguyễn Ngọc Đan Thanh

Sinh viên thực hiện:
Họ tên: Nguyễn Duy Khang
MSSV: 110121188
Lớp: DA21TTC

Trà Vinh, tháng 01 năm 2024

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

1. **Quá trình thực hiện:**
   - Nghiêm túc, báo cáo tiến độ theo lịch
   - Có khả năng nghiên cứu độc lập
2. **Về báo cáo**
   - Đúng mẫu
   - Đầy đủ các nội dung theo đề cương
3. **Về chương trình**
   - Chương trình cơ bản đáp ứng theo yêu cầu đã phân tích
   - Dữ liệu minh họa phong phú
4. **Kết luận:** Đạt mức Khá

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày ..... tháng …… năm ……

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Trà Vinh, ngày ..... tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Kính gửi: cô Nguyễn Ngọc Đan Thanh

Trước tiên, cho phép tôi được gửi lời cảm ơn chân thành nhất đến Cô Nguyễn Ngọc Đan Thanh đã tận tình giảng dạy, hướng dẫn và giúp đỡ tôi trong suốt quá trình thực hiện đồ án thực tập cơ sở ngành.

Cô đã dành nhiều thời gian quý báu của mình để chia sẻ những kiến thức bổ ích, hướng dẫn và giúp tôi hoàn thành đồ án cơ sở ngành một cách tốt nhất. Cô đã luôn động viên, khích lệ tôi trong suốt quá trình thực hiện đồ án này, giúp tôi vượt qua những khó khăn và hoàn thành đồ án đúng tiến độ đã đề ra.

Tôi cũng xin gửi lời cảm ơn đến tất cả các bạn đã giúp đỡ và hỗ trợ tôi khi gặp những vướng bận, khó khăn và trở ngại trong quá trình thực hiện đồ án.

Tôi xin hứa sẽ luôn tiếp tục nổ lực học tập và phấn đấu trong công việc để sau này trở thành một người thành công.

Trân trọng,

Nguyễn Duy Khang

MỤC LỤC

DANH MỤC HÌNH ẢNH – BẢNG BIỂU

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

Vấn đề nghiên cứu: tập trung vào việc nghiên cứu các nội dung của danh sách liên kết và ứng dụng vào bài toán quản lí sản phẩm, để cải thiện quá trình quản lý thông tin và các chức năng của sản phẩm.

Các hướng tiếp cận: tìm hiểu và phân tích cách sử dụng ngôn ngữ lập trình C và tìm hiểu về danh sách liên kết đơn. Ứng dụng vào chương trình để cài đặt các chức năng quản lý cơ bản thêm, xóa, tìm kiếm sản phẩm.

Cách giải quyết vấn đề: Tìm hiểu và đọc các tài liệu hướng dẫn, tài liệu tham khảo về danh sách liên kết đơn và ngôn ngữ C, để biết các cú pháp và cách sử dụng các câu lệnh cơ bản và thực hành nhiều về các câu lệnh cơ bản trong C để hiểu rõ hơn về cách chúng hoạt động và vận dụng vào chương trình.

Một số kết quả đạt được: chạy được chương trình danh sách liên kết quản lí sản phẩm, cài đặt và chạy được các chức năng trong bài toán quản lí sản phẩm. Có thể sử dụng các chức năng thêm sản phẩm, hiển thị sản phẩm, xóa sản phẩm, tìm kiếm sản phẩm và xuất thông tin sản phẩm ra file. Giao diện đơn giản, dễ sử dụng và đáp ứng các chức năng cơ bản.

MỞ ĐẦU

1. Lí do chọn đề tài

Công nghệ thông tin là một trong những ngành phát triển vượt bậc trong những năm gần đây. Ngày nay, với sự phát triển nhanh chóng của xã hội thì công nghệ thông tin được ứng dụng rất rộng rãi, hầu như ngành công nghệ thông tin đều đóng vai trò quan trọng trong tất cả các lĩnh vực và trở thành một phần thiết yếu trong cuộc sống hằng ngày. Công nghệ thông tin là ngành mà đòi hỏi ở người học một nền tảng kiến thức vững chắc, sự tư duy logic cao và phải hiểu biết sâu rộng trên nhiều lĩnh vực. Với tôi, hiện đang là sinh viên Công nghệ Thông tin cần phải có sự kiên trì, sự đầu tư và không ngừng học hỏi để nâng cao kiến thức. Việc ứng dụng công nghệ thông tin vào công tác quản lí là rất cần thiết. Do đó, đề tài mà tôi lựa chọn để thực hiện đồ án thực tập cơ sở ngành là: ***"Ứng dụng danh sách liên kết trong bài toán quản lý sản phẩm"***.

2. Mục đích của đề tài

Củng cố lại những kiến thức đã học về danh sách liên kết nói chung và các dụng danh sách liên kết trong đời sống nói riêng; Rèn luyện tốt kỹ năng lập trình trên ngôn ngữ C; Ứng dụng nội dung đã học để xây dựng bài toán danh sách liên kết quản lý sản phẩm.

3. Đối tượng nghiên cứu

Nghiên cứu danh sách liên kết trong bài toán quản lí sản phẩm trên đối tượng ngôn ngữ C; Thông tin các sản phẩm cần lưu trữ.

4. Phương pháp nghiên cứu

Phương pháp nghiên cứu lý thuyết: Tìm hiểu cấu trúc danh sách liên kết, phân loại và các ứng dụng.

Phương pháp nghiên cứu thực nghiệm: Cài đặt chương trình quản lý thông tin sản phẩm với cấu trúc danh sách liên kết.

5. Phạm vi nghiên cứu

Nghiên cứu trong phạm vi cài đặt chương trình quản lí sản phẩm bằng cách sử dụng danh sách liên kết đơn hoạt động trên ngôn ngữ lập trình C.

CHƯƠNG 1 TỔNG QUAN

Hiện nay, với việc quản lí sản phẩm là công việc cần phải làm hằng ngày của phòng quản lý. Công việc quản lí sản phẩm đòi hỏi phải có tính tỉ mỉ, cẩn thận trong khâu ghi chép các thông tin của sản phẩm như mã sản phẩm, tên sản phẩm, số lượng sản phẩm,... củng như công việc thống kê các sản phẩm cần phải rõ ràng và chính xác. Trước đây, công nghệ thông tin chưa phát triển mạnh mẽ, việc xử lý các công việc chủ yếu là bằng thủ công, như là ghi chép bằng bút, sổ sách. Chính vì vậy rất tốn công sức và khá nhiều thời gian. Ngày nay khi mà khoa học kỹ thuật phát triển, đặc biệt là sự bùng nổ của công nghệ thông tin thì việc quản lí sản phẩm sẽ dễ dàng hơn rất nhiều. Yêu cầu của bài toán quản lí sản phẩm là tạo ra chương trình có thể thực hiện các chức năng quản lí sản phẩm một cách dễ dàng, tiện lợi dựa trên sự trợ giúp của máy tính. Mọi công việc phải được đảm bảo đúng thao tác trên vùng dữ liệu chung để dễ dàng đảm bảo việc đồng bộ với nhau trong khâu quản lí.

Để đáp ứng các yêu cầu trên của bài toán quản lí sản phẩm. Trước tiên, ta cần phải tìm hiểu và đánh giá chi tiết về hệ thống danh sách liên kết hiện tại, phân tích sự tương tác giữa các yếu tố trong danh sách liên kết. Thu thập tất cả dữ liệu có liên quan đến bài toán quản lí sản phẩm để đảm rằng dữ liệu trong danh sách liên kết là chính xác, đầy đủ và đáng tin cậy.

Phải nghiên cứu các phương pháp thu thập dữ liệu hiệu quả và đảm bảo sự liên tục trong quá trình thực hiện. Xây dựng và tích hợp hệ thống thông tin quản lí sản phẩm để theo dõi và quản lí thông tin sản phẩm một cách hiệu quả. Xây dựng chiến lực tương tác với khách hàng thông minh khi sử dụng dữ liệu và công nghệ để cá nhân hóa trải nghiệm của khách hàng. Xây dựng và thực hiện các chương trình kiểm soát chất lượng để đảm bảo sản phẩm đáp ứng các yêu cầu chất lượng, tối ưu hóa quy trình đảm bảo an toàn sản phẩm. Thiết lập các chỉ số hiệu suất quan trọng, đánh giá hiệu suất thường xuyên và áp dụng các phương pháp tối ưu hóa liên tục để duy trì và cải thiện quá trình quản lí sản phẩm.

CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT

2.1 Tổng quan về ngôn ngữ C

2.1.1 Khái niệm

C là một ngôn ngữ lập trình phổ biến nhất thế giới, là ngôn ngữ đơn giản và linh hoạt khi sử dụng. Nó là một ngôn ngữ lập trình có cấu trúc độc lập và được sử dụng rộng rãi để viết các ứng dụng, hệ điều hành như Windows và nhiều chương trình phức tạp khác như Oracle database, Git, Python Interpreter,…

Ngoài ra, rất nhiều lập trình viên khi học lập trình C đều ví C là "ngôn ngữ mẹ". Bởi C là cơ sở, nền tảng cho các ngôn ngữ khác và nếu lập trình viên học lập trình C giỏi thì các ngôn ngữ khác như C++, C#, Java đều có thể chinh phục dễ dàng.

2.1.2 Đặc điểm của ngôn ngữ C

Tính cô đọng (compact): C chỉ có 32 từ khóa chuẩn và 40 toán tử chuẩn, nhưng hầu hết đều được biểu diễn bằng những chuỗi ký tự ngắn gọn.

Tính cấu trúc (structured): C có một tập hợp những chỉ thị của lập trình như cấu trúc lựa chọn, lặp… Từ đó các chương trình viết bằng C được tổ chức rõ ràng, dễ hiểu.

Tính tương thích (compatible): C có bộ tiền xử lý và một thư viện chuẩn vô cùng phong phú nên khi chuyển từ máy tính này sang máy tính khác các chương trình viết bằng C vẫn hoàn toàn tương thích.

Tính linh động (flexible): C là một ngôn ngữ rất uyển chuyển và cú pháp, chấp nhận nhiều cách thể hiện, có thể thu gọn kích thước của các mã lệnh làm chương trình chạy nhanh hơn.

Biên dịch (compile): C cho phép biên dịch nhiều tập tin chương trình riêng rẽ thành các tập tin đối tượng (object) và liên kết (link) các đối tượng đó lại với nhau thành một chương trình có thể thực thi được (executable) thống nhất.

2.1.3 Một số chức năng

Một ngôn ngữ cốt lõi đơn giản, với các chức năng quan trọng chẳng hạn như là những hàm hay việc xử lý tập tin sẽ được cung cấp bởi các bộ thư viện các thủ tục.

Tập trung trên mẫu hình lập trình thủ tục, với các phương tiện lập trình theo kiểu cấu trúc.

Một hệ thống kiểu đơn giản nhằm loại bỏ nhiều phép toán không có ý nghĩa thực dụng.

Dùng ngôn ngữ tiền xử lý, tức là các câu lệnh tiền xử lý C, cho các nhiệm vụ như là định nghĩa các macro và hàm chứa nhiều tập tin mã nguồn (bằng cách dùng câu lệnh tiền xử lý dạng #include chẳng hạn).

Mức thấp của ngôn ngữ cho phép dùng tới bộ nhớ máy tính qua việc sử dụng kiểu dữ liệu pointer.

Số lượng từ khóa rất nhỏ gọn.

Các tham số được đưa vào các hàm bằng giá trị, không bằng địa chỉ.

Hàm các con trỏ cho phép hình thành một nền tảng ban đầu cho tính đóng và tính đa hình.

Hỗ trợ các bản ghi hay các kiểu dữ liệu kết hợp do người dùng từ khóa định nghĩa struct cho phép các dữ liệu liên hệ nhau có thể được tập hợp lại và được điều chỉnh như là toàn bộ.

2.1.4 Một số lệnh cơ bản thường gặp

#include <stdio.h>	Lệnh này bao gồm tệp tiêu đề đầu ra đầu vào tiêu chuẩn (stdio.h) từ thư viện C trước khi biên dịch chương trình C
int main()	Đây là hàm chính từ nơi bắt đầu thực thi chương trình C.
{	Cho biết phần đầu của hàm chính.
/*_some_comments_*/	Bất cứ điều gì được viết bên trong lệnh này "/*….. */" bên trong chương trình C, nó sẽ không được xem xét để biên dịch và thực thi.
printf("Hello_World! ");	Lệnh này in kết quả ra màn hình
getch();	Lệnh này được sử dụng cho bất kỳ dữ liệu nhập ký tự nào từ bàn phím.
return 0;	Lệnh này được sử dụng để kết thúc chương trình C (chức năng chính) và nó trả về 0.
}	Nó được sử dụng để biểu thị sự kết thúc của chức năng chính

Bảng 1. Một số lệnh trong ngôn ngữ C

2.1.5 Ứng dụng chính của ngôn ngữ C

Ngôn ngữ C được áp dụng rộng rãi trong các hệ thống Nhúng.

C được sử dụng để phát triển System Apps.

C còn được sử dụng rộng rãi để phát triển các ứng dụng máy tính để bàn.

C được sử dụng để phát triển các phần mềm ứng dụng nổi tiếng như adobe, trình duyệt Chromium của Google, MySQL,…

C cũng được sử dụng để phát triển các hệ điều hành OSX của Apple, Windows của Microsoft và Symbian được phát triển bằng ngôn ngữ C.

Ngoài các phần mềm, hệ điều hành C còn được sử dụng để phát triển máy tính bàn, sản xuất trình biên dịch và sử dụng rộng rãi trong các ứng dụng IOT.

2.2 Tổng quan về danh sách liên kết

2.2.1 Khái niệm danh sách liên kết

Danh sách liên kết (Linked List) là một dãy các cấu trúc dữ liệu được kết nối với nhau thông qua các liên kết (Link). Hiểu đơn giản hơn danh sách liên kết là một cấu trúc dữ liệu bao gồm các nút (Node) tạo thành một chuỗi. Mỗi nút bao gồm dữ liệu ở nút đó và tham chiếu đến nút kế tiếp trong chuỗi.

Thuật ngữ trong danh sách liên kết

Link: mỗi liên kết của danh sách được liên kết có thể lưu trữ một dữ liệu được gọi là một phần tử.

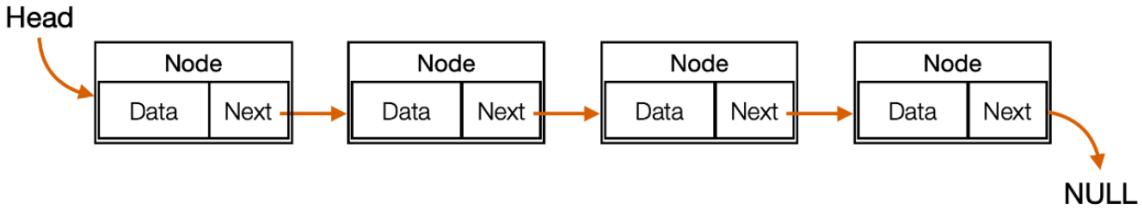
Next: mỗi liên kết của danh sách được liên kết chứa một liên kết đến liên kết tiếp theo.

Linked List: danh sách được liên kết chưa liên kết kết nối với liên kết đầu tiên.

```
struct Node{
    int data;
    struct Node* next;
};
```

Biểu diễn Danh sách liên kết

Danh sách được liên kết có thể là một chuỗi các nút, nơi mọi nút đều trỏ đến nút tiếp theo



Hình 2. 1. Danh sách liên kết

Danh sách được liên kết chứa một phần tử liên kết được gọi là đầu tiên.

Mỗi liên kết mang trường dữ liệu và một trường liên kết được gọi là tiếp theo.

Mỗi liên kết được liên kết với liên kết tiếp theo của nó bằng các sử dụng liên kết tiếp theo của nó.

Liên kết cuối cùng mang một liên kết là NULL để đánh dấu phần cuối của danh sách.

2.2.2 Các thao tác cơ bản trong danh sách liên kết

Chèn: thêm một phần tử vào đầu danh sách.

Duyệt: duyệt qua lần lượt các phần tử trong danh sách.

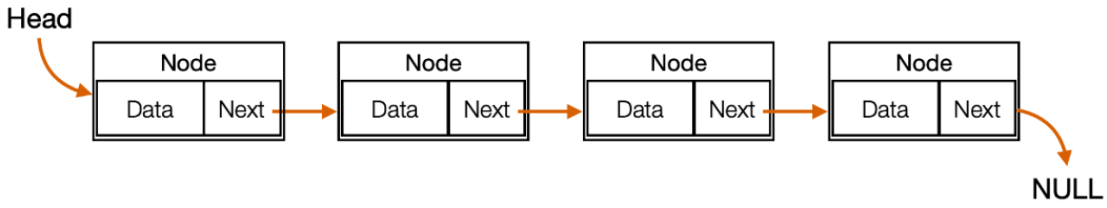
Xóa: xóa một phần tử ở đầu danh sách hoặc xóa một phần tử bằng cách sử dụng khóa đã cho.

Tìm kiếm: tìm kiếm một phần tử bằng cách sử dụng khóa đã cho.

2.2.3 Các loại danh sách liên kết

Danh sách liên kết đơn

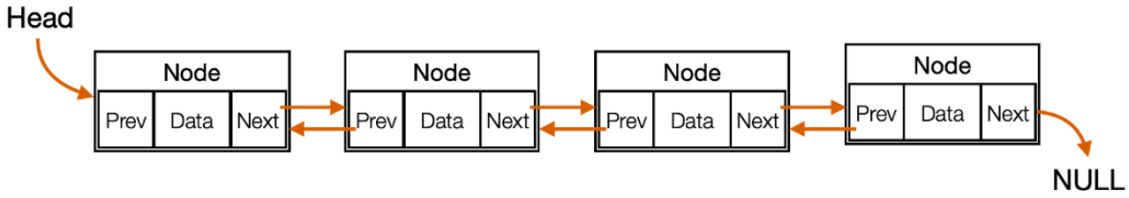
Danh sách liên kết đơn chứa các nút có trường dữ liệu (data) cũng như trường tiếp theo (next), trỏ đến nút tiếp theo trong chuỗi các nút. Các thao tác có thể được thực hiện trên các danh sách được liên kết đơn bao gồm chèn, xóa và duyệt.



Hình 2. 2. Danh sách liên kết đơn

Danh sách liên kết kép

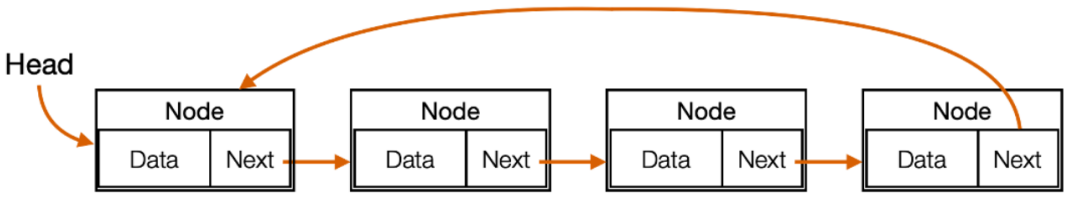
Trong danh sách liên kết kép, mỗi nút chứa, ngoài liên kết nút tiếp theo, một trường liên kết nút thứ hai trỏ đến nút trước đó trong chuỗi các nút. Hai liên kết có thể được gọi là chuyển tiếp (forward) và quay lại (backwards), hoặc tiếp theo (next) và trước (prev/previous).



Hình 2. 3. Danh sách liên kết kép

Danh sách liên kết móc vòng

Trong nút cuối cùng của danh sách, trường liên kết thường chứa tham chiếu rỗng, một giá trị đặc biệt được sử dụng để chỉ ra việc thiếu các nút tiếp theo. Một quy ước ít phổ biến hơn là làm cho nó trỏ đến nút đầu tiên của danh sách; trong trường hợp đó, danh sách được cho là vòng tròn hoặc liên kết móc vòng; nếu không, nó được cho là mở hoặc tuyến tính. Nó là một danh sách mà con trỏ cuối cùng trỏ đến nút đầu tiên.



Hình 2. 4. Danh sách liên kết móc vòng

2.3 Triển khai danh sách liên kết

2.3.1 Danh sách liên kết đơn

2.3.1.1 Khái niệm 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 đứng sau nó trong danh sách. Mỗi phần tử (Node) 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 thành phần đứng sau trong danh sách, nếu phần tử đó là phần tử cuối cùng thì thành phần này bằng NULL.

2.3.1.2 Đặc điểm của danh sách liên kết đơn

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

- 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 năng 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, có các đặc điểm sau:

- Chỉ cần nắm được phần tử đầu và cuối là có thể quản lí 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.

2.3.1.3 Cài đặt danh sách liên kết đơn

Tạo node

Danh sách liên kết đơn được tạo thành từ nhiều node, do đó, chúng ta sẽ cùng đi từ node trước. Một node gồm hai thành phần là thành phần dữ liệu và thành phần liên kết. Thành phần dữ liệu có thể là kiểu dữ liệu có sẵn hoặc bạn tự định nghĩa (struct hay class…). Thành phần liên kết là địa chỉ con trỏ, con trỏ này trỏ đến node tiếp theo; do đó, con trỏ này là con trỏ trỏ vào một node.

```
struct Node{
     int data;
     Node* next;
};
```

Để tạo một node mới, ta thực hiện cấp phát động cho node mới, khởi tạo giá trị ban đầu và trả về địa chỉ của node mới được cấp phát.

```
Node* CreateNode(int init_data){
     Node* node = new Node;
     node -> data = init_data;
     node -> next = NULL;
     return node;
}
```

Tạo danh sách liên kết đơn

Thành phần tạo nên danh sách liên kết đơn là node, tiếp theo cần quản lý danh sách bằng cách biết được phần tử đầu và cuối. Vì mỗi phần tử đều liên kết với phần tử kế vậy nên chỉ cần biết phần tử đầu và cuối là có thể quản lý được danh sách này. Vậy ta cần tạo một cấu trúc lưu trữ địa chỉ phần tử đầu (Head) và phần tử cuối (hay phần tử đuôi Tail).

```
struct LinkedList {
     Node* head;
     Node* tail;
};
```

Khi mới tạo danh sách, danh sách sẽ không có phần tử nào, do đó head và tail không trỏ vào đâu cả, ta sẽ gán chúng bằng NULL. Cần xây dựng hàm tạo danh sách như sau:

```
void CreateList (LinkedList&1)
{
```

```
    l.head = NULL;
    l.tail = NULL;
}
```

Để tạo một danh sách, ta cần các thao tác sau:

```
LinkedList list;
CreateList;   //Gán head và tail bằng NULL
```

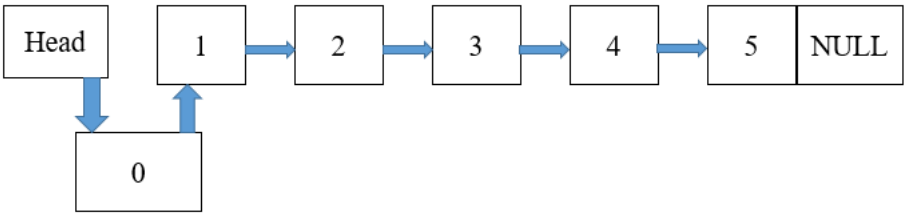
Thêm phần tử vào danh sách

Thêm vào đầu

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

```
void AddHead (LinkedList& l, Node* node){
    if (l.head == NULL){
        l.head = node;
        l.tail = node;
    }
    else
    {
        node -> next = l.head;
        l.head = node;
    }
}
```

Ví dụ thêm phần tử vào đầu danh sách liên kết đơn



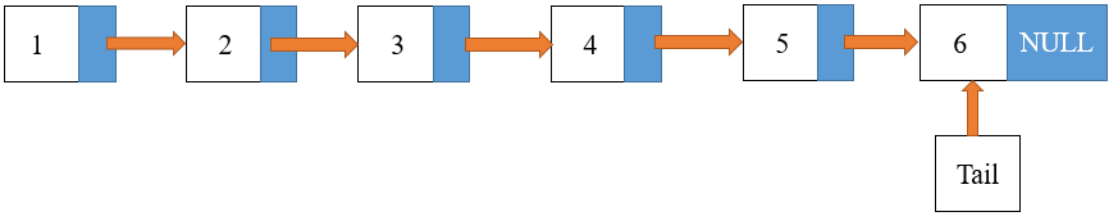
Hình 2. 5. Thêm phần tử vào đầu danh sách

Thêm vào cuối

Để thêm node vào cuối danh sách, đầu tiên 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).

```
void AllTail (LinkedList& l, Node* node)
{
	if (l.head == NULL)
	{
		l.head = node;
		l.tail = node;
	}
	else
	{
		l.tail -> next = node;
		l.tail = node;
	}
}
```

Ví dụ thêm phần tử vào cuối danh sách liên kết đơn



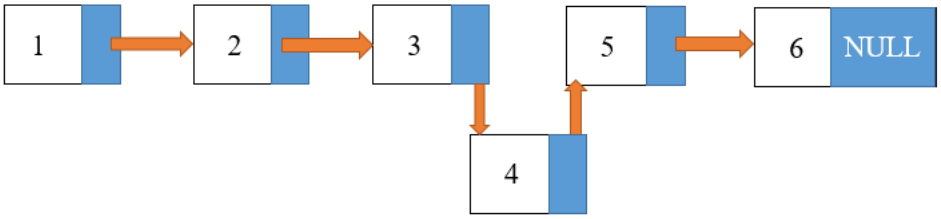
Hình 2. 6. Thêm phần tử vào cuối danh sách

Thêm vào sau node bất kỳ

Để thêm một node p vào sau node q bất kỳ, đầu tiên cần kiểm tra xem node q có NULL hay không, nếu node q là NULL tức là danh sách rỗng, vậy thì 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 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.

```
void InsertAfterQ (LinkedList& l, Node* p, Node* q)
{
     if (q != NULL)
     {
          p -> next = q -> next;
          q -> next = p;
          if (l.tail == q)
               l.tail = p;
     }
     else
          AllHead(l, p)
}
```

Ví dụ thêm phần tử vào sau nút Q trong danh sách liên kết đơn



Hình 2. 7. Thêm phần tử sau nút Q

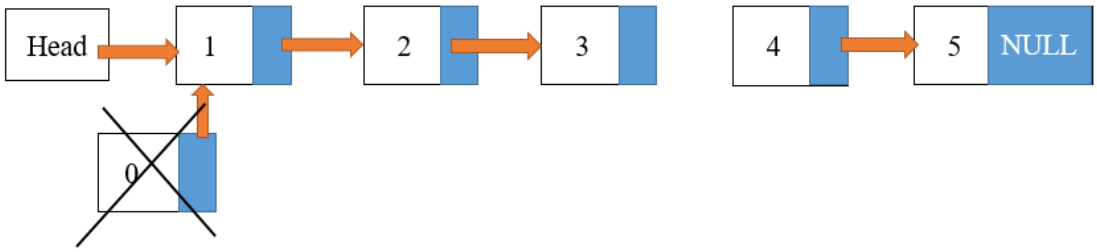
Xóa phần tử khỏi danh sách

Xóa ở 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.

```
int RemoveHead (LinkedList& l, int& x){
     if (l.head != NULL){
          Node* node = l.head;
          x = node -> data;  //Lưu giá trị của node head
lại
          l.head = node -> next;
          delete node;       //Hủy node head đi
          if   (l.head == NULL)
               l.tail = NULL;
          return 1;
     }
     return 0;
}
```

Ví dụ xóa phần tử đầu danh sách liên kết đơn



Hình 2. 8. Xóa phần tử đầu danh sách

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.

```
int RemoveAfterQ (LinkedList& l, Node* q, int& x){
     if (q != NULL){
          Node* p = q -> next;
          if (p != NULL){
               if (l.tail == p)
                    l.tail = q;
               q -> next = p -> next;
               x = p -> data;
               delete p;
               return 1;
          }
          return 0;
     }    return 0;}
```

Duyệt danh sách và in

Sau khi có các thao tác thêm, xóa, 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.

```
void PrintfList (LinkedList l){
     if (l.head != NULL)
     {
          Node* node = l.head;
          while (node != NULL)
          {
               printf (" node -> data");
               node = node -> next;   //chuyển sang node
tiếp theo
          }
     }
}
```

Lấy giá trị node bất kỳ

Để lấy giá trị phần tử trong danh sách, ta thực hiện duyệt tương tự như khi in phần tử. Ta sẽ tạo một biến đếm để biết vị trí hiện tại, duyệt qua các node cho đến khi node bằng NULL hoặc biến đếm bằng với vị trí node cần lấy. Kiểm tra xem nếu node khác NULL và biến đếm bằng vị trí cần lấy, ta sẽ trả về địa chỉ của node đó, ngược lại trả về NULL (danh sách rỗng hoặc là vị trí cần lấy nằm ngoài phạm vi của danh sách).

```
Node* GetNode (LinkedList& l, int index){
     Node* node = l.head;
     int i = 0;
```

```
    while (node != NULL && i != index)
    {
        node = node -> next;
        i++;
    }
    if (i == index && node != NULL)
        return node;
    return NULL;
}
```

Tìm kiếm phần tử trong danh sách

Tìm kiếm phần tử cũng như là duyệt danh sách, nếu như chưa tìm thấy thì tiếp tục duyệt. Sau khi kết thúc duyệt, chỉ cần kiểm tra xem node duyệt có bằng NULL hay không, nếu không tức là đã tìm thấy, ta sẽ trả về địa chỉ của node đó.

```
Node* Search(LinkedList l, int x){
    Node* node = l.head;
    while (node != NULL && node -> data != x)
        node = node -> next;
    if (node != NULL)
        return node;
    return NULL;
}
```

Đếm số phần tử duyệt của danh sách

Đếm số phần tử của danh sách như là duyệt từ đầu đến cuối và đếm số node.

```
int Length(LinkedList l){
    int i = 0;
    Node* node = l.head;
```

```
     while (node != NULL)
     {
          i++;
          node = node  -> next;
     }
     return i;
}
```

Xóa danh sách

Để xóa danh sách, ta cần hủy tất cả các node duyệt và hủy từng node. Ở đây sẽ dùng lại hàm RemoveHead. Đầu tiên, ta gán một node bằng head, kiểm tra nếu node đó khác NULL thì gọi RemoveHead và gán lại node bằng head tiếp, cứ lặp như vậy cho đến khi node đó NULL thì thôi. Sau khi xóa hết tất cả phần tử thì gán lại tail bằng NULL.

```
void DestroyList(LinkedList& l){
     int x;
     Node* node = l.head;
     while (node != NULL)
     {
          RemoveHead(l, x);
          node = l.head;
     }
     l.tail = NULL;
}
```

2.3.2 Danh sách liên kết kép

2.3.2.1 Khái niệm danh sách liên kết kép

Danh sách liên kết kép là nơi mỗi phần tử (nút) trong danh sách không chỉ lưu giữ giá trị dữ liệu mà còn liên kết với cả phần tử trước và phần tử sau nó trong danh sách.

2.3.2.2 Đặc điểm của danh sách liên kết kép

Liên kết hai chiều: mỗi nút chứa hai con trỏ, một trỏ đến nút phía trước và một trỏ đến nút phía sau. Điều này cho phép việc điều hướng qua lại giữa các phần tử cả ở phía trước và phía sau một cách hiệu quả.

Được truy cập ngược lại: dễ dàng duyệt danh sách theo cả hai hướng, từ đầu đến cuối và từ cuối về đầu

Chèn và xóa linh hoạt: danh sách liên kết kép cho phép chèn và xóa phần tử ở cả hai đầu.

Thực hiện các thao tác phức tạp một cách dễ dàng: việc có thể điều hướng cả hai chiều giúp thực hiện các thao tác phức tạp như lập danh sách, hoán đổi các phần tử,...

2.3.2.3 Các thao tác trên danh sách liên kết kép

Tạo một Node mới

```
Node *createNewNode (int data){
     Node *newNode = (Node *) malloc (sizeof(Node));
     newNode -> data = data;
     newNode -> next = NULL;
     return newNode;
}
```

Chèn một phần tử vào danh sách

Chèn vào đầu danh sách

```
void AddHead (Node *Head, int data)
```

```
{
     Node *newNode = createNewNode(data);
     newNode -> next = *Head;
     *Head = newNode;
}
```

Chèn vào cuối danh sách

```
void AddTail (Node *Head, int data){
     if (*Head = NULL){
          *Head = newNode;
          return;
     }
     Node *Tail = *Head;
     while (Tail -> next != NULL){
          Tail = Tail -> next;
     }
     Tail -> next = newNode;
}
```

Chèn vào danh sách sau một phần tử q

```
void AddAfter (Node *prevNode, int data)
{
     if (prevNode = NULL)
{
          printf ("\nKhong the tim thay nut da cho.");
          return;
     }
     Node *newNode = createNewNode(data);
```

```
     newNode -> next = prevNode -> next;
     prevNode -> next = newNode;
}
```

Chèn vào danh sách trước một phần tử q

```
void AddBefore ( Node *Head, Node *nextNode, int data)
{
     if (nextNode = NULL)
{
          printf ("\nKhong the tim thay nut da cho.");
          return;
     }
     Node *newNode = createNewNode(data);
     Node *q = *Head;
     while (q != NULL && q -> next != nextNode)
{
          q = q -> next;
     }
     if (q = NULL)
     {
          printf (\nKhong the tim thay nut da cho.");
          free(newNode);
          return;
     }
     newNode -> next = nextNode;
     q -> next = newNode;
}
```

Xóa phần tử khỏi danh sách

Xóa một Node xác định

```
void RemoveNode (Node *Head, Node *nodeRemove) {
     if (nodeRemove = NULL || *Head = NULL){
          return;
     }
     if (*Head = nodeRemove){
          *Head = nodeRemove -> next;
          free(nodeRemove);
          return;
     }
     Node *q = *Head;
     while (q != NULL && q -> next != nodeRemove)   {
          q = q -> next;
     }
     if (q = NULL) {
          return;
     }
     q -> next = nodeRemove -> next;
     free(nodeRemove);
}
```

Xóa danh sách liên kết

```
void RemoveLinkedList (Node *Head){
     Node *q = *Head;
     Node *next;
     while ( q != NULL)
     {
          next = q -> next;
          free(q);
          q = next;
     }
     *Head = NULL;
}
```

Duyệt danh sách liên kết và in ra các giá trị của các Node

```
void printLinkedList(Node *Head){
     Node *q = Head;
     while (q != NULL)
     {
          printf("%d", q -> data);
          q = q -> next;
     }
     printf("\n");
}
```

CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU

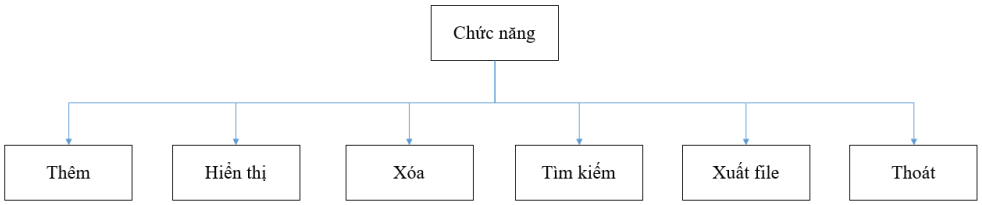
3.1 Mô tả bài toán

Bài toán quản lý sản phẩm là một trong những thách thức quan trọng trong lĩnh vực kinh doanh, bán lẻ và quản lý kho hàng. Công việc quản lí sản phẩm đòi hỏi tính tỉ mỉ, cẩn thận trong khâu ghi chép các thông tin của sản phẩm như mã sản phẩm, tên sản phẩm, giá sản phẩm, số lượng sản phẩm,.... Trước đây công nghệ thông tin chưa phát triển mạnh mẽ, các công việc được xử lý thủ công, chủ yếu được ghi lại bằng bút, sổ sách; chính vì vậy rất tốn nhiều công sức và khá nhiều thời gian. Ngày nay khi mà khoa học kỹ thuật phát triển, đặc biệt là sự bùng nổ công nghệ thông tin thì việc quản lí sản phẩm sẽ dễ dàng hơn. Xuất phát từ nhu cầu đó, mà bài toán quản lý sản phẩm bằng danh sách liên kết được ra đời. Yêu cầu của bài toán là tạo ra chương trình có thể thực hiện các thao tác quản lý sản phẩm một các dễ dàng, tiện lợi dựa trên sự trợ giúp của máy tính. Mọi công việc phải được thao tác trên một vùng dữ liệu chung để đảm bảo việc đồng bộ với nhau trong việc quản lý sản phẩm.

Bài toán quản lý sản phẩm là chương trình quản lý hồ sơ, thông tin của sản phẩm. Chương trình có thể thực hiện các công việc thêm mới sản phẩm, tìm kiếm sản phẩm cần tìm, xóa sản phẩm, hiển thị thông tin sản phẩm,...

Chương trình được viết bằng ngôn ngữ C và dựa trên cấu trúc lưu trữ của danh sách liên kết đơn.

3.2 Đặc tả các yêu cầu chức năng



Hình 3. 1. Sơ đồ các chức năng

3.3 Thiết kế hệ thống

3.3.1 Thiết kế cấu trúc dữ liệu

Cấu trúc thương hiệu

```
struct THUONG_HIEU
{
	char tenth[50];
	int namthanhlap;
};
```

Mô tả chi tiết cấu trúc

STT	*Thuộc tính*	*Diễn giải*	*Kiểu dữ liệu*
1	tenth	Tên thương hiệu	char
2	namthanhlap	Năm thành lập	int

Cấu trúc danh mục

```
struct DANH_MUC
{
	int madm;
	char tendm[50];
};
```

Mô tả chi tiết cấu trúc

STT	*Thuộc tính*	*Diễn giải*	*Kiểu dữ liệu*
1	madm	Mã danh mục	int
2	tendm	Tên danh mục	char

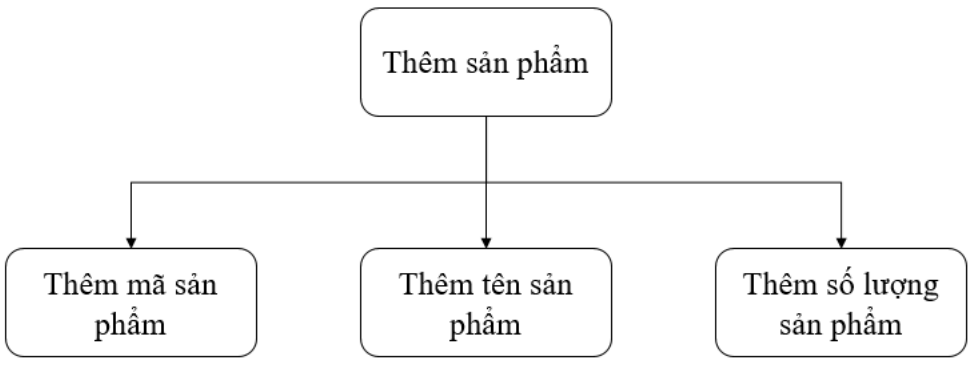
Cấu trúc sản phẩm

```
struct SAN_PHAM
{
	int masp;
	char tensp[50];
	float giaban;
	char ngaysx[50];
	char hansd[50]
};
```

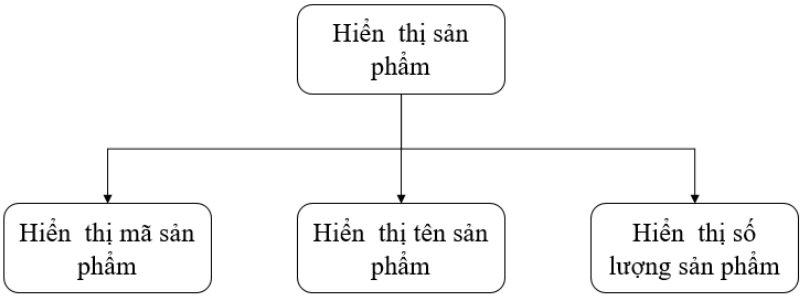
Mô tả chi tiết cấu trúc

STT	*Thuộc tính*	*Diễn giải*	*Kiểu dữ liệu*
1	masp	Mã sản phẩm	int
2	tensp	Tên sản phẩm	char
3	giaban	Giá bán	float
4	ngaysx	Ngày sản xuất	char
5	hansd	Hạn sử dụng	char

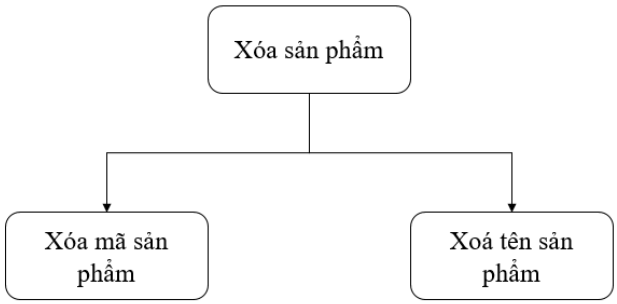
3.3.2 Thiết kế xử lý



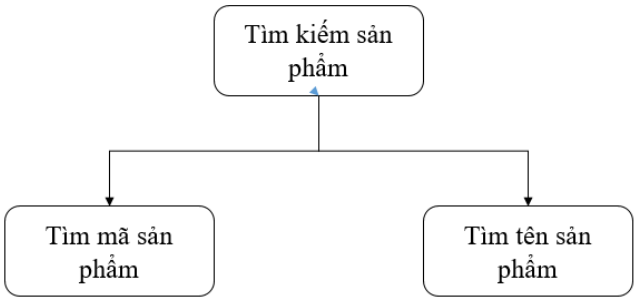
Hình 3. 2. Sơ đồ chức năng thêm sản phẩm



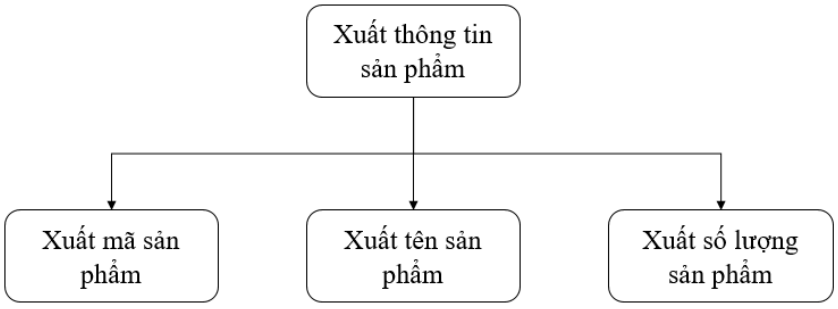
Hình 3. 3. Sơ đồ chức năng hiển thị sản phẩm



Hình 3. 4. Sơ đồ chức năng xóa sản phẩm



Hình 3. 5. Sơ đồ chức năng tìm kiếm sản phẩm



Hình 3. 6. Sơ đồ xuất thông tin sản phẩm ra file

CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU

Trình bày các kết quả đạt được sau quá trình thực hiện đồ án. Có thể đánh giá về hiệu năng, trải nghiệm người dùng, hoặc trình bày các giao diện chức năng của nghiên cứu ở phần này.

4.1.1 Dữ liệu thử nghiệm

Dữ liệu về danh sách liên kết đơn trong bài toán quản lí sản phẩm bằng ngôn ngữ lập trình C.

Dữ liệu được tham khảo tại trang:

https://songdaymooncake.com/cac-san-pham-cua-kinh-do/, mailto:http://www.bibica.com.vn/#:~:text=M%E1%BB%97i%20n%C4%83m%20c%C3%B4ng%20ty%20c%C3%B3,kh%C3%A1%20m%E1%BA%A1nh%20tr%C3%AAn%20th%E1%BB%8B%20tr%C6%B0%E1%BB%9Dng.

Tên thương hiệu	Năm thành lập
Kinh đô	1993
Bibica	1999
Orion – Vina	1956

Bảng 2. Dữ liệu thương hiệu

Mã danh mục	Tên danh mục
1	Nhãn hàng bánh AFC
2	Nhãn hàng bánh kẹo Cosy
3	Nhãn hàng bánh Solite
4	Nhãn hàng Danisa
5	Nhãn hàng Goody Gold
6	Nhãn hàng An
7	Nhãn hàng Migita

Bảng 3. Dữ liệu danh mục

Mã sản phẩm	Tên sản phẩm	Giá bán	Ngày sản xuất	Hạn sử dụng
1	Bánh quy bơ thập cẩm Cosy	98.000.000	22/11/2023	22/11/2024
2	Bộ quà tết Orion An Tết Phúc	125.000.000	05/12/2023	05/06/2024
3	Bánh quy bơ Danisa	180.000.000	02/01/2024	02/06/2024

4	Bánh bông lan Orion C'est Bon	22.000.000	27/12/2023	27/03/2024
5	Hộp quà Socola Cadbuly Dairy	98.000.000	05/01/2024	05/05/2024
6	AFC bánh Cracker	42.300.000	28/12/2023	28/04/2024
7	Bánh quế Cosy	12.650.000	03/01/2024	03/03/2024
8	Hộp quà tết bánh Oreo Kinh Đô	148.000.000	29/12/2023	29/06/2024
9	Bánh bông lan tròn Solite	46.500.000	01/11/2023	01/07/2024
10	Kẹo cứng Migita hương me	10.500.000	18/12/2023	18/06/2024
11	Kẹo dẻo hương trái cây Alpenliebe	19.000.000	10/12/2023	10/04/2024
12	Bánh gạo khoai tây Orion An	22.000.000	03/01/2024	03/06/2024
13	Bánh quy bơ Danisa	133.000.000	25/12/2023	25/06/2024

Bảng 4. Dữ liệu sản phẩm

4.1.2 Kết quả thực nghiệm

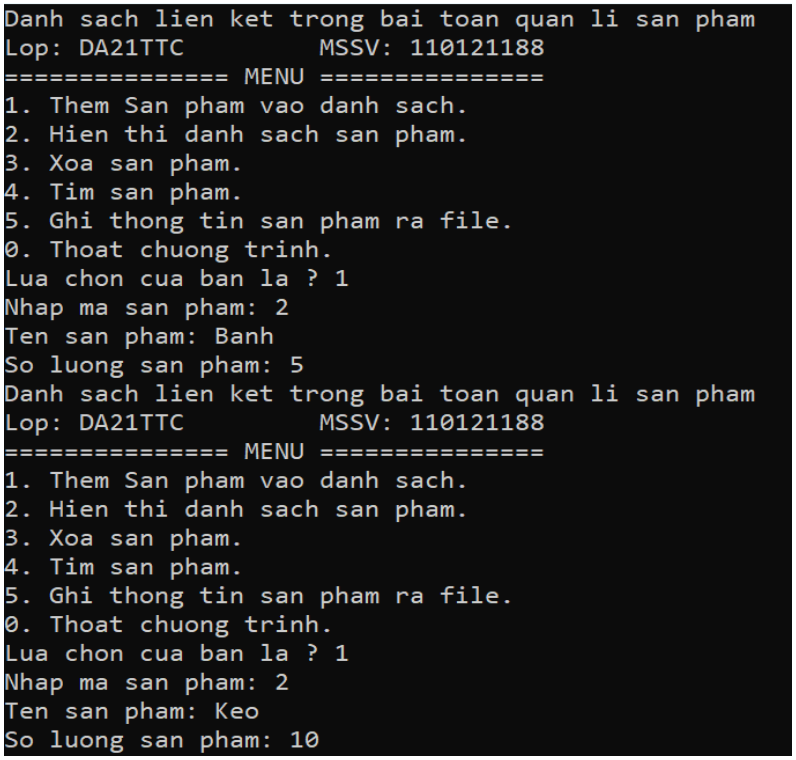
Màn hình chính

```
Danh sach lien ket trong bai toan quan li san pham
Lop: DA21TTC          MSSV: 110121188
=============== MENU ===============
1. Them San pham vao danh sach.
2. Hien thi danh sach san pham.
3. Xoa san pham.
4. Tim san pham.
5. Ghi thong tin san pham ra file.
0. Thoat chuong trinh.
Lua chon cua ban la ?
```

Hình 3. 7. Màn hình chính

Mô tả: Sau khi chạy chương trình thì danh sách liên kết trong bài toán quản lí sản phẩm sẽ hiển thị ra các chức năng có trong bài toán quản lí gồm các chức năng thêm sản phẩm, hiển thị sản phẩm, xóa sản phẩm, tìm kiếm sản phẩm, ghi thông tin sản phẩm ra file và thoát chương trình. Người dùng có thể lựa chọn các chức năng mà mình muốn.

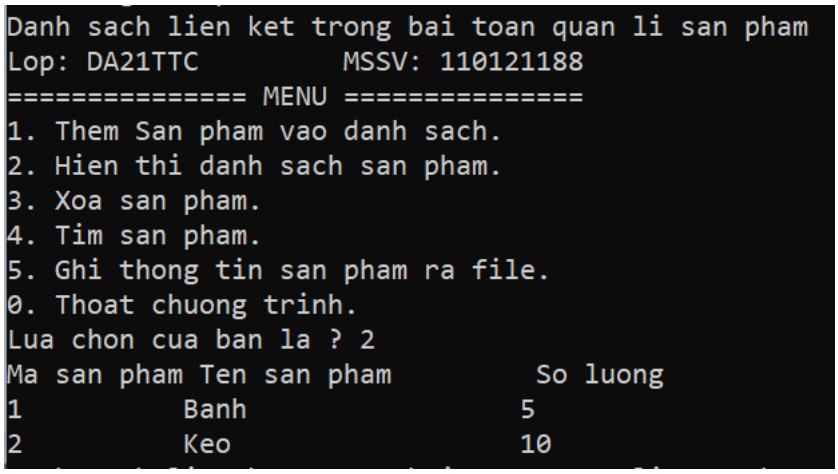
4.1.2.1 Chức năng thêm sản phẩm vào danh sách



Hình 3. 8. Chức năng thêm sản phẩm

Mô tả chức năng: Khi hệ thống hiển thị lên danh sách các chức năng, khi bạn nhập vào số 1 thì sẽ vào chức năng thêm sản phẩm, bạn cần phải nhập mã sản phẩm, tên sản phẩm và số lượng sản phẩm bạn muốn đặt. Khi nhập xong thì các dữ liệu thì hệ thống sẽ quay lại Menu lúc đầu và nếu bạn cần nhập sản phẩm khác thì sẽ làm như lúc đầu.

4.1.2.2 Chức năng hiển thị danh sách sản phẩm



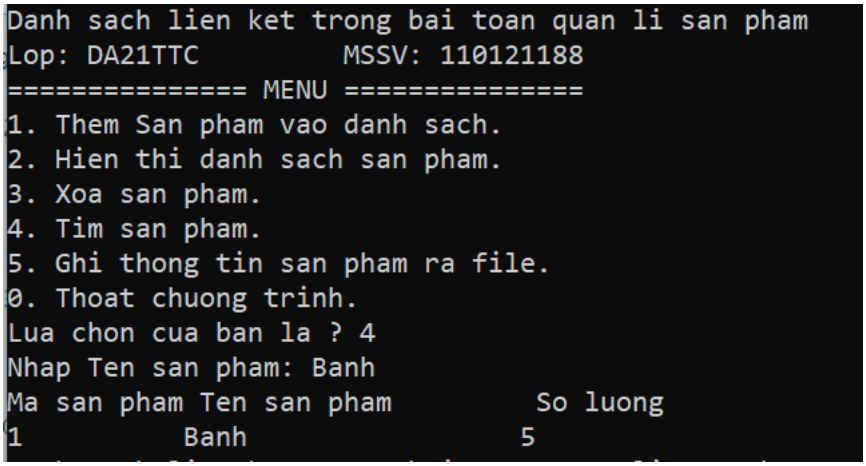
Hình 3. 9. Chức năng hiển thị sản phẩm

Mô tả chức năng: Khi hệ thống hiển thị lên danh sách các chức năng, sau khi bạn đã thêm các sản phẩm vào danh sách. Bạn muốn kiểm tra xem các sản phẩm mình đã thêm, bạn sẽ nhập vào 2 thì hệ thống sẽ hiển thị lên thông tin các sản phẩm mà bạn đã thêm vào.

4.1.2.3 Chức năng xóa sản phẩm

Mô tả chức năng: Khi hệ thống hiển thị lên danh sách các chức năng, sau khi bạn đã thêm các sản phẩm vào danh sách và đã hiển thị thông tin các sản phẩm. Khi bạn muốn xóa một sản phẩm bất kỳ thì bạn sẽ nhập vào số 3, hệ thống sẽ hỏi sản phẩm bạn muốn xóa có id là gì? và bạn chỉ cần nhập mã sản phẩm bạn muốn xóa thì hệ thống sẽ xóa sản phẩm đó.

4.1.2.4 Chức năng tìm kiếm sản phẩm



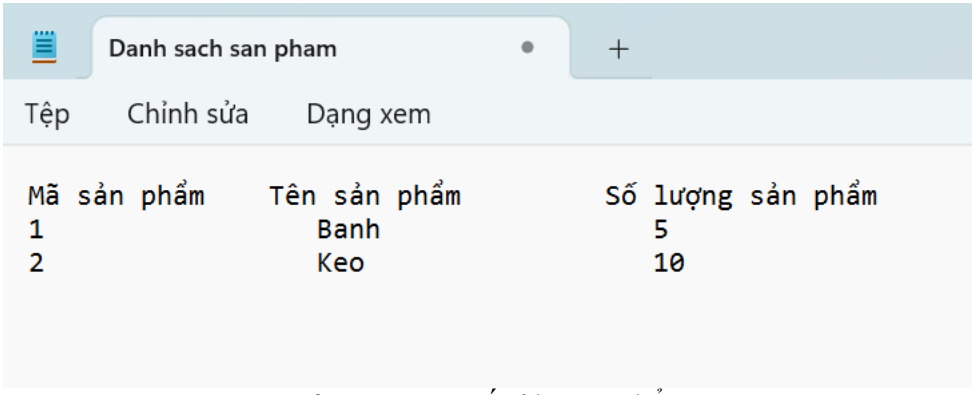
Hình 3. 10. Chức năng tìm sản phẩm

Mô tả chức năng: Khi hệ thống hiển thị lên danh sách các chức năng, sau khi bạn đã thêm các sản phẩm vào danh sách và đã hiển thị thông tin các sản phẩm. Bạn muốn tìm kiếm một sản phẩm bất kỳ thì bạn sẽ chọn vào 4, khi đó hệ thống sẽ kêu bạn nhập tên sản phẩm cần tìm, khi bạn nhập xong thì sản phẩm bạn tìm kiếm sẽ hiển thị lên trong hệ thống.

4.1.2.5 Chức năng ghi thông tin sản phẩm ra file

```
Danh sach lien ket trong bai toan quan li san pham
Lop: DA21TTC         MSSV: 110121188
=============== MENU ===============
1. Them San pham vao danh sach.
2. Hien thi danh sach san pham.
3. Xoa san pham.
4. Tim san pham.
5. Ghi thong tin san pham ra file.
0. Thoat chuong trinh.
Lua chon cua ban la ? 5
```

Hình 3. 11. Ghi thông tin sản phẩm ra file



Hình 3. 12. Xuất file sản phẩm

Mô tả chức năng: Khi hệ thống hiển thị lên danh sách các chức năng, sau khi đã nhập xong các thông tin sản phẩm và cần xuất thông tin ra file thì bạn sẽ nhập vào 5 để ghi thông tin sản phẩm ra file. Khi nhập xong thì sản phẩm sẽ được xuất ra một file khác và lưu thông tin sản phẩm lại.

4.1.2.6 Chức năng thoát chương trình

```
Danh sach lien ket trong bai toan quan li san pham
Lop: DA21TTC         MSSV: 110121188
=============== MENU ===============
1. Them San pham vao danh sach.
2. Hien thi danh sach san pham.
3. Xoa san pham.
4. Tim san pham.
5. Ghi thong tin san pham ra file.
0. Thoat chuong trinh.
Lua chon cua ban la ? 0

--------------------------------
Process exited after 1756 seconds with return value 0
Press any key to continue . . .
```

Hình 3. 13. Chức năng thoát chương trình

Mô tả chức năng: Khi hệ thống hiển thị lên danh sách các chức năng, sau khi đã nhập xong các thông tin sản phẩm và đã xuất thông tin sản phẩm ra file rồi. Bạn muốn tắt chương trình hiện tại thì bạn sẽ nhập vào 0 để thoát ra khỏi chương trình.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận: Trình bày những kết quả đạt được, những đóng góp mới và những đề xuất mới. Phần kết luận cần ngắn gọn, không có lời bàn và bình luận thêm.

Hướng phát triển: Kiến nghị về những hướng nghiên cứu tiếp theo.

5.1. Kết luận

- Phân tích, tìm hiểu các chức năng của bài toán quản lí sản phẩm: chức năng thêm, hiển thị, xóa, tìm kiếm, xuất file và thoát khỏi chương trình.

- Phân tích được dữ liệu cho hệ thống

- Xây dựng được chương trình quản lý sản phẩm

- Liên kết sản phẩm có thể giúp theo dõi thông tin về nguồn gốc, nhà cung cấp, và các thông tin liên quan đến chuỗi cung ứng, từ đó giúp quản lý dễ dàng tối ưu hóa quá trình sản xuất và cung ứng.

- Tạo được một nền tảng để phát triển thành một danh sách liên kết sản phẩm hoàn chỉnh.

- Khi có một danh sách cụ thể với liên kết đến thông tin chi tiết về từng sản phẩm, người quản lý có thể dễ dàng tìm kiếm và truy cập thông tin cần thiết mà không mất nhiều thời gian.

5.2. Hướng phát triển

Để chương trình có thể hoạt động hiệu quả hơn cần dữ liệu phải được lưu trữ bằng các hệ quản trị cơ sở dữ liệu có tính bảo mật cao hơn.

Khả năng xử lý được tất cả các chức năng, các lỗi ngoài ý muốn tốt hơn của chương trình và dùng các thao tác của chương trình.

Thêm các chức năng mới để đáp ứng điều kiện cho quản lí sản phẩm hoàn chỉnh.

Để đáp ứng tốt với nhu cầu thực tế thì cần phải nghiên cứu thêm nhu cầu của khách hàng để phát triển hệ thống một cách linh hoạt và đơn giản hơn để có thể sử dụng tốt hơn.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] L. V. Bằng, Hệ thống quản lí sinh viên, 2013.

[2] R. Hac, Ngôn ngữ C, 2021.

[3] D. T. Hiếu, Danh sách liên kết đơn (Singly Linked List), 2021.

[4] L. V. Hoàng, Giới thiệu ngôn ngữ C, 2021.

[5] N. M. Phúc, Danh sách liên kết đơn - Quản lí sinh viên, 2021.

[6] L. N. Quang, Quản lí sinh viên bằng danh sách liên kết đơn, 2023.

[7] T. Triệu, Tạo menu quản lí các chức năng, 2019.