

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**---------------🙦 🕮 🙤---------------

**BÁO CÁO ĐỒ ÁN**

***Đề tài báo cáo:***

**DANH SÁCH LIÊN KẾT ĐƠN**

***Giảng viên hướng dẫn:* Nguyễn Thiên Bảo**

***Lớp: 04CLC***

***Thành viên:***

Nguyễn Đức Anh 15110006

Lưu Đình Chuẩn 18110085

Phan Thị Thu Trang 18110217

Võ Hùng Tú 18110233

Đặng Thanh Tú 16110253

*Hồ Chí Minh, ngày 13 tháng 12 năm 2019*

Mục lục

[Mục lục 1](#_Toc27134914)

[CHƯƠNG 1: LÝ THUYẾT & MÔ PHỎNG 3](#_Toc27134915)

[1. Dữ liệu kiểu con trỏ 3](#_Toc27134916)

[1.1Giới thiệu chung và định nghĩa con trỏ: 3](#_Toc27134917)

[1.2. Khai báo: 4](#_Toc27134918)

[1.3. Sử dụng: 4](#_Toc27134919)

[2. Danh sách liên kết : 6](#_Toc27134920)

[2.1. Định nghĩa: 6](#_Toc27134921)

[2.2. Phân loại: 6](#_Toc27134922)

[2.3. Khai báo: 7](#_Toc27134923)

[2.4. Các thao tác cơ bản trên Danh sách liên kết đơn : 8](#_Toc27134924)

[2.4.1. Định nghĩa Node 8](#_Toc27134925)

[2.4.2. Khai báo một Node 8](#_Toc27134926)

[2.4.3. Chèn một Node P có giá trị x vào DSLK: 8](#_Toc27134927)

[2.5. Ưu điểm và nhược điểm của Danh sách liên kết đơn : 13](#_Toc27134931)

[CHƯƠNG 2: VÍ DỤ VỀ CÁC BÀI TOÁN CƠ BẢN TRÊN DSLK 13](#_Toc27134932)

[1. Nhập DSLK: 13](#_Toc27134933)

[3. Tính tổng các giá trị lẻ trong DSLK: 14](#_Toc27134934)

[4. Tính trung bình cộng các số nguyên tố trong DSLK: 15](#_Toc27134935)

[5. Đếm xem trong DSLK có bao nhiêu phần tử chẵn: 17](#_Toc27134936)

[6. Kiểm tra DSLK có chứa các số chẵn hay không? 17](#_Toc27134937)

[7. Tìm Node Q ở vị trí K trong DSLK: 18](#_Toc27134938)

[8. Chèn giá trị X vào vị trí K trong DSLK: 19](#_Toc27134939)

[9. Xoá một phần tử tại vị trí k trong DSLK: 20](#_Toc27134940)

[10. Xoá các số âm trong DSLK: 21](#_Toc27134941)

[CHƯƠNG 3: ỨNG DỤNG DANH SÁCH LIÊN KẾT ĐƠN 23](#_Toc27134942)

[1. Giới thiệu chung về phần mềm 23](#_Toc27134943)

[2. Phân tích thiết kế phần mềm: 23](#_Toc27134944)

[KẾT LUẬN 24](#_Toc27134945)

[DANH MỤC TÀI LIỆU THAM KHẢO 25](#_Toc27134946)

CHƯƠNG 1: LÝ THUYẾT & MÔ PHỎNG

## 1. Dữ liệu kiểu con trỏ

### 1.1Giới thiệu chung và định nghĩa con trỏ:

- Các biến được sử dụng từ trước đến nay đều là biến có kích thước và kiểu dữ liệu xác định, người ta gọi là biến tĩnh (static).

- Khi chạy chương trình, gặp những biến này, máy sẽ cung cấp lương bộ nhớ và địa chỉ cho các biến đó mà không cần biết các biến đó sử dụng luc nào hoặc thậm chí có được sử dụng hay không.

- Các biến tĩnh tồn tại trong suốt thời gian thực hiện chương trình. Vì vậy, nếu chạy một chương trình lớn trong khi máy lại hạn chế bộ nhớ thì sẽ xay ra tình trạng không đủ bộ nhớ.

- Đối với những chương trình mà ta không dự đoán trước được kích thước của chúng ra sao thì sẽ xảy ra tình trạng:

+ Cấp dư, gây lãng phí bộ nhớ.

+ Cấp thiếu, chương trình không chạy được.

- Để khắc phục những nhược điểm trên, các ngôn ngữ lập trình cấp cao như Pascal, Turbo C, … thường sử dụng biến động (dynamic) vì có những đặc điểm sau:

+ Biến động được sinh ra trong quá trình thực hiện chương trình.

+ Biến động là loại biến có biến đổi được kích thước, vùng nhớ và địa chỉ của vùng nhớ được cấp phát lúc chạy chương trình.

+ Có thể giải phóng biến động sau kho đã sử dụng để tiết kiệm bộ nhớ.

- Thế nhưng biến động cũng có nhược điểm là không thể truy cập đến nó được bởi biến động không chứa địa chỉ nhất định.

- Để khắc phục nhược điểm này, người ta sử dụng một loại biến đặc biệt gọi là biến con trỏ (pointer).

- Biến con trỏ có các đặc điểm:

+ Không chứa dữ liệu nhưng chứa địa của dữ liệu tức địa chỉ của biến khác.

+ Kích thước của biến con trỏ không phụ thuộc vào đối tượng mà nó trỏ tới.

+ Con trỏ có cùng kiểu dữ liệu với kiểu dữ liệu của biến mà nó trỏ tới.

- Vậy: *“Biến con trỏ là loại biến chuyên dùng để chứa chứa địa chỉ của các biến động, giúp ta truy nhập đến biến động”.*

### 1.2. Khai báo:

- Cú pháp khai báo một con trỏ như sau

**<kiểu dữ liệu> \*<tên con trỏ>;**

Trong đó:

+ Kiểu dữ liệu: có thể là các kiểu dữ liệu cơ bản của C++, hoặc là kiểu dữ liệu có cấu trúc, hoặc là kiểu đối tượng do người dùng tự định nghĩa

+ Tên con trỏ: Tuân theo các quy tắc đặt tên biến của C++:

+ Chỉ được bắt đầu bằng một ký tự (chữ), hoặc dấu gạch dưới “\_”

+ Bắt đầu từ ký tự thứ hai có thể có kiểu ký tự số

+ Không có dấu trống (space bar) trong tên biến

+ Có phân biệt chữ hoa và chữ thường

+ Không giới hạn độ dài tên biến

Ví dụ: Để khai báo một biến con trỏ có kiểu là *int* và tên là *songuyen*, ta viết như sau:

**int \*songuyen;**

*Lưu ý:*

+ Có thể viết dấu con trỏ ”\*” ngay sau kiểu dữ liệu

Ví dụ:

**int \*songuyen;**

tương đương với

**int\* songuyen;**

### 1.3. Sử dụng:

- Con trỏ dùng để truy cập biến thông qua địa chỉ biến và chương trình tham khảo biến gián tiếp qua địa chỉ này.

- Con trỏ được sử dụng theo 2 cách:

+ Dùng con trỏ để lưu địa chỉ của biến: Lấy giá trị của biến do con trỏ trỏ đến để thao tác.

+ Dùng con trỏ để lưu địa chỉ của biến: Bản thân con trỏ sẽ được trỏ vào địa chỉ của một biến có cùng kiểu dữ liệu với nó. Cú pháp của phép gán như sau:

**<tên con trỏ> = &<tên biến>;**

*Lưu ý:*

+ Trong phép toán này, tên con trỏ không có dấu “\*”

Ví dụ:

**int x, \*px;**

**px = &x;**

Hai dòng lệnh trên sẽ cho con trỏ px có kiểu int trỏ vào địa chỉ của biến x có kiểu nguyên. Phép toán &<tên biến> sẽ cho địa chỉ của biến tương ứng.

- Lấy giá trị của biến do con trỏ trỏ đến: Phép lấy giá trị của biến do con trỏ trỏ đến được thực hiện bằng cách gọi tên:

**\*<tên con trỏ>;**

*Lưu ý:*

+Trong phép toán này phải có dấu con trỏ “\*”, nếu không có dấu con trỏ sẽ trở thành phép lấy địa chỉ của biến do con trỏ trỏ tới.

Ví dụ:

**int x=12, y, \*px;**

**px = &y;**

**\*px = x;**

Quy trình diễn ra như sau:

*int x=12, y, \*px;* x=12 y=0 px 🡪 NULL

*px = &y;* x=12 y=0 🡨 px

*\*px = x;* x=12 y=x=12🡨 px

Con trỏ px vẫn trỏ tới địa chỉ biến y và giá trị biến y sẽ là 12.

2. Danh sách liên kết :

2.1. Định nghĩa:

- Danh sách liên kết là danh sách mà các phần tử (Node) được liên kết với nhau nhờ vào vùng liên kết của chúng.

- Như vậy, mỗi Node có 2 thành phần là phần Info dùng để chứa dữ liệu cần xử lý và phần liên kết dùng để liên kết với các Node khác.

### 2.2. Phân loại:

- Có nhiều loại Danh sách liên kết như:

+ Danh sách liên kết đơn.

+ Danh sách liên kết đôi/kép.

+ Danh sách đa liên kết.

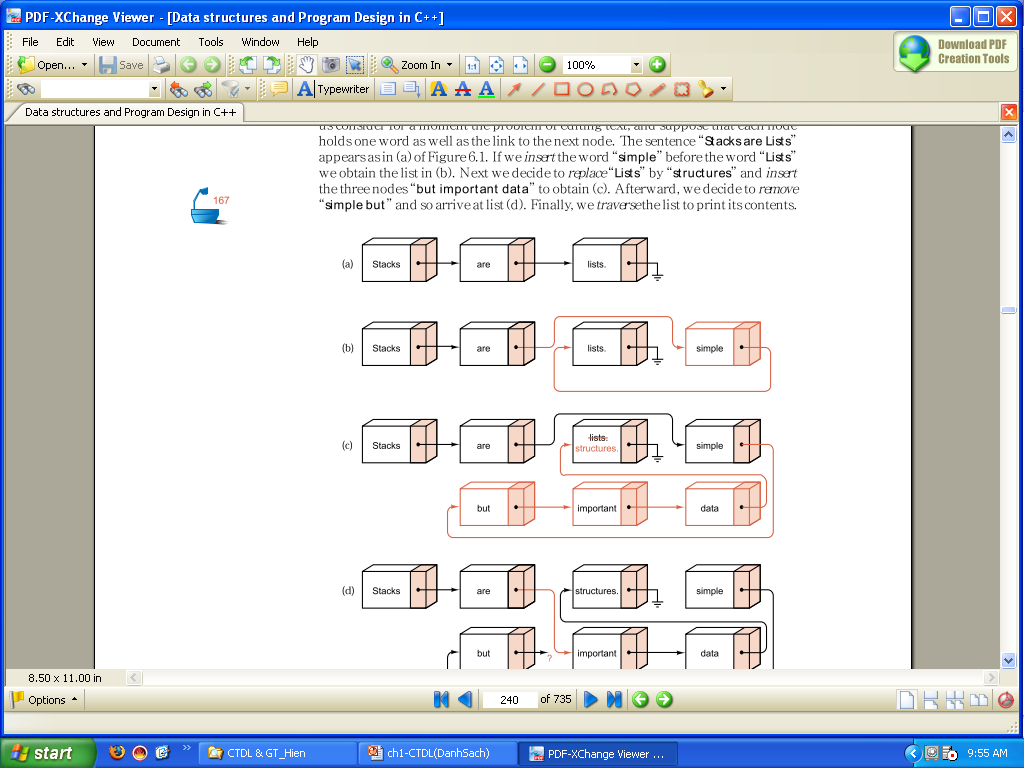
+ Danh sách liên kết vòng.

+ …

- Mỗi loại danh sách liên kết sẽ có cách biểu diễn các phần tử (cấu trúc dư liệu) riêng và các thao tác trên đó. Trong nội dung bản báo cáo này chỉ đề cập đến loại danh sách liên kết cơ bản nhất là Danh sách liên kết đơn.

*- Danh sách liên kết đơn là danh sách mà mỗi Node của nó chỉ có một trường liên kết***.**

- Các hình dưới mô tả một danh sách liên kết đơn

****

### 2.3. Khai báo:

- Mỗi Node của danh sách có thể biểu diễn như sau:

|  |  |
| --- | --- |
| **Info** | **Next** |

- Trong đó:

+ Info: gồm các trường chứa nội dung thông tin của phần tử.

+ Next: chứa địa chỉ của Node tiếp theo hay còn gọi là vùng liên kết.

***Định nghĩa Node:***

**struct Node**

**{**

**int Info;**

**Node \*Next;**

**};**

- Danh sách liên kết đơn có thể hình dung qua hình vẽ sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***first*** | A |  |  | B |  |  | C |  |  | D |  |  | NULL |

-  ***first*** là con trỏ chứa địa chỉ Node đầu danh sách.

- Danh sách rỗng thì first=NULL.  
- Khi làm việc trên danh sách liên kết đơn chúng ta cần có những chú ý sau:

+ Danh sách luôn có con trỏ trỏ đầu danh sách: con trỏ first.

+ Danh sách luôn có một giá trị báo kết thúc danh sách: giá trị NULL.

+ Trường Next của mỗi Node chỉ chứa địa chỉ Node ở sau nó, trừ Node cuối cùng.

+ Trường Next của Node cuối cùng chứa giá trị NULL.

+ Không tách danh sách thành hai danh sách con nếu danh sách phần sau chưa có con trỏ trỏ tới.

+ Khởi tạo danh sách rỗng: first = NULL;

### 2.4. Các thao tác cơ bản trên Danh sách liên kết đơn :

### 2.4.1. Định nghĩa Node

struct Node

{

int Info; // khai báo trường Info là kiểu int

Node \*Next; // khai báo con trỏ Next kiểu Node (vì nó chứa địa chỉ trỏ tới 1 Node khác có 2 trường)

};

### 2.4.2. Khai báo một Node

Node \*MakeNode(int x) // Hàm MakeNode tạo ra một con trỏ kiểu Node nên có kiểu trả về là Node

{

Node \*P; // khai báo biến con trỏ P kiểu Node

P=new Node; // cấp phát một Node cho con trỏ P

P🡪Info=x; // truy cập vào trường Info gán giá trị là x

P🡪Next=NULL; // truy cập vào trường Next, trỏ tới NULL

return P; // trả về kiểu Node cho con trỏ P

}

|  |  |  |  |
| --- | --- | --- | --- |
| x |  |  | NULL |

### 

### 2.4.3. Chèn một Node P có giá trị x vào DSLK:

### Chèn Node P có chứa giá trị x vào đầu danh sách liên kết

*- Dữ liệu:*

+ Đầu vào : Giá trị x, DSLK

+ Đầu ra : DSLK.

*- Thuật toán:*

+ Bước 1: Nhập giá trị x

+ Bước 2: Khởi tạo Node P có giá trị x

+ Bước 3: So sánh first với NULL .Nếu first bằng NULL thì thực hiện bước 4, nếu không thì qua bước 5.

+ Bước 4: Gán first = P. Thực hiện bước 6

+ Bước 5: Gán P🡪Next=first, first=P.

+ Bước 6: Kết thúc hàm.

*- Chương trình:*

void PushTop(int x,Node \*&first)

{

Node \*P;

P=MakeNode(x);

if(first= = NULL)

first=P;

else

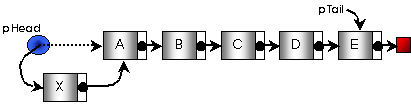
{

P🡪Next=first;

First=P;

}

}



### Chèn Node P có chứa giá trị x vào vị trí bất kỳ trên danh sách liên kết

*- Dữ liệu:*

+ Đầu vào: Giá trị x, vị trí k , DSLK

+ Đầu ra: DSLK.

*- Thuật toán:*

+ Bước 1: Nhập giá trị x và vị trí k.

+ Bước 2: Khởi tạo Node P có giá trị x, chạy hàm Tìm Node Q có vị trí k.

+ Bước 3: So sánh Q với NULL .Nếu Q bằng NULL thì thực hiện bước 4, nếu không thì qua bước 5.

+ Bước 4: Kết luận không chèn.

+ Bước 5: Khai báo Node R=first.

+ Bước 6: So sánh R🡪Next với Q. Nếu khác thì thực hiện bước 7, nếu không thì qua bước 8.

+ Bước 7: Gán R=R🡪Next. Quay lại bước 6.

+ Bước 8: Gán R🡪Next=P và P🡪Next=Q.

+ Bước 9: Kết thúc hàm.

*- Chương trình:*

void Insert(int x,int k,Node \*&first)

{

cout<<"\nNhap gia tri can chen:";

cin>>x;

cout<<"\nNhap vi tri can chen:";

cin>>k;

Node \*P=MakeNode(x);

Node \*Q=FindQ(k,first);

if(Q==NULL)

cout<<"\nDSLK rong.Khong chen";

else

{

if(Q==first)

{

P->Next=first;

first=P;

}

else

{

Node \*R=first;

while(R->Next!=Q)

R=R->Next;

R->Next=P;

P->Next=Q;

}

}

}



### Chèn Node P có chứa giá trị x vào cuối danh sách liên kết

*- Dữ liệu:*

+ Đầu vào: Giá trị x, DSLK

+ Đầu ra: DSLK.

*- Thuật toán:*

+ Bước 1: Nhập giá trị x

+ Bước 2: Khởi tạo Node P có giá trị x

+ Bước 3: So sánh first với NULL .Nếu first bằng NULL thì thực hiện bước 4, nếu không thì qua bước 5.

+ Bước 4: Gán first = P. Thực hiện bước

+Bước 5: Khai báo Node R=first.

+ Bước 6: So sánh R🡪Next với NULL. Nếu khác NULL thì qua bước 7, nếu bằng NULL thì qua bước 8.

+ Bước 7: Gán R=R🡪Next. Quay lại bước 6.

+ Bước 8: Gán R🡪Next=P và P🡪Next=NULL.

+ Bước 9: Kết thúc hàm.

*- Chương trình*

void PushEnd(int x,Node \*&first)

{

Node \*P;

P=MakeNode(x);

if(first= = NULL)

first=P;

else

{

Node \*R=first;

while(R🡪Next!=NULL)

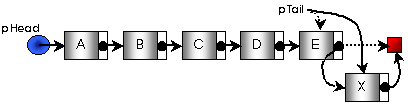
R=R🡪Next;

R🡪Next=P;

P🡪Next=NULL;

}

}



### 2.5. Ưu điểm và nhược điểm của Danh sách liên kết đơn :

* Ưu điểm
  + Tận dụng được không gian bộ nhớ nhỏ để lưu trữ từng nút.
  + Việc thêm, xóa phần tử trong danh sách liên kết là dễ dàng, chỉ cần thay đổi mối liên kết của các phần tử với nhau.
* Nhược điểm
  + Mật độ sử dụng bộ nhớ của danh sách liên kết không tối ưu tuyệt đối (<100%)
  + Việc truy xuất và tìm kiếm các phần tử trong danh sách liên kết mất nhiều thời gian vì phải duyệt tuần tự qua các phần tử trong danh sách.
  + Bộ nhớ cần nhiều vì phải lưu thêm phần tử liên kết, nếu vùng dữ liệu là lớn thì tỷ lệ mức sử dụng bộ nhớ là cao.

# CHƯƠNG 2: VÍ DỤ VỀ CÁC BÀI TOÁN CƠ BẢN TRÊN DSLK

1. Nhập DSLK:

*- Dữ liệu:*

+ Đầu vào : x, first

+ Đầu ra : DSLK

*- Thuật toán:*

+ Bước 1: Nhập số phần tử của DSLK cần tạo.

+ Bước 2: Khai báo con chạy i=1

+ Bước 3: So sánh con chạy i với n. Nếu i<=n thì chuyển sang bước 4, nếu không qua bước 6.

+ Bước 4: Nhập giá trị x, thực thi hàm chèn một giá trị mới vào đầu DSLK PushTop.

+ Bước 5: Giá trị i tăng thêm 1. Quay trở lại bước 3.

+ Bước 6: Kết thúc hàm.

*- Sơ đồ khối:*



## 3. Tính tổng các giá trị lẻ trong DSLK:

*- Dữ liệu:*

+ Đầu vào: DSLK

+ Đầu ra: Tổng các giá trị lẽ

*- Thuật toán:*

+ Bước 1: Khai báo Node R=first, Tong Le=0.

+ Bước 2: So sánh R với NULL. Nếu R khác NULL thì thực hiện bước 3, nếu không thì qua bước 6.

+ Bước 3: So sánh R🡪Info có chia hết cho 2 không, nếu không thì thực hiện bước 4, nếu đúng thì thực hiện bước 5.

+ Bước 4: TongLe bằng TongLe cộng với giá trị R🡪Info.

+ Bước 5: Gán R=R🡪Next. Quay lại bước 2.

+ Bước 6: Kết thúc hàm.

*- Sơ đồ khối:*



## 4. Tính trung bình cộng các số nguyên tố trong DSLK:

*- Dữ liệu:*

+ Đầu vào: DSLK

+ Đầu ra: Trung bình cộng các số nguyên tố.

*- Thuật toán:*

+ Bước 1: Khai báo Node R=first, Tong=0, DemSNT=0, TBC.

+ Bước 2: So sánh R với NULL. Nếu R khác NULL thì thực hiện bước 3, nếu không thì qua bước 14.

+ Bước 3: Gán R🡪Info=x

+ Bước 4: So sánh x với 2. Nếu x=2 thì chuyển qua bước 5, nếu không qua bước 9.

+ Bước 5: Gán CheckNumber=1

+ Bước 6: So sánh CheckNumber với 1. Nếu CheckNumber =1 thì chuyển sang bước 7, nếu không qua bước 8.

+ Bước 7: DemSNT tăng thêm 1, Tong = Tong + x

+ Bước 8: Gán R=R🡪Next. Quay lại bước 2.

+ Bước 9: Khai báo con chạy i=2:

+ Bước 10: So sánh I với x. Nếu i<x thì qua bước 11, nếu không phải thì quay lại bước 6.

+ Bước 11: So sánh xem x có chia hết cho I hay không. Nếu sai thì qua bước 12, nếu đúng thì qua bước 13.

+ Bước 12: Gán CheckNumber=1, i tăng thêm 1. Quay lại bước 10.

+ Bước 13: CheckNumber=0. Quay lại bước 6.

+ Bước 14: TBC bằng Tong chia DemSNT

+ Bước 15: Kết thúc hàm.

*- Sơ đồ khối:*



## 5. Đếm xem trong DSLK có bao nhiêu phần tử chẵn:

*- Dữ liệu:*

+ Đầu vào: DSLK

+ Đầu ra: Số phần tử chẵn.

*- Thuật toán:*

+ Bước 1: Khai bào Node R=first. Dem=0

+ Bước 2: So sánh R với NULL. Nếu R khác NULL thì thực hiện bước 3, nếu không thì qua bước 6.

+ Bước 3: So sánh xem R🡪Info có chia hết cho 2 hay không. Nếu có thì qua bước 4, nếu không chuyển sang bước 5.

+ Bước 4: Dem tăng thêm 1.

+ Bước 5: Gán R=R🡪Next. Quay lại bước 2.

+ Bước 6: Kết thúc hàm.

*- Sơ đồ khối:*



## 6. Kiểm tra DSLK có chứa các số chẵn hay không?

*- Dữ liệu:*

Đầu vào : DSLK

Đầu ra : Kết luận DSLK có chẵn hay không.

*- Thuật toán:*

+ Bước 1: Khai báo biến Check. R=first.

+Bước 2: So sánh R với NULL. Nếu R khác NULL thì thực hiện bước 3, nếu không thì qua bước 7.

+ Bước 3: So sánh xem R🡪Info có chia hết cho 2 hay không, nếu có thì qua bước 4, nếu không qua bước 6:

+Bước 4: Gán Check=1

+ Bước 5: Gán R=R🡪Next. Quay lại bước 2

+ Bước 6: Gán Check=0

+ Bước 7: Nếu Check=1 kết luận DSLK chẵn, nếu không kết luận DSLK lẽ

+ Bước 8: Kết thúc hàm.

*- Sơ đồ khối:*



## 7. Tìm Node Q ở vị trí K trong DSLK:

*- Dữ liệu:*

+ Đầu vào: Vị trí k , DSLK

+ Đầu ra: Node Q.

*- Thuật toán:*

+ Bước 1: Khai báo Q=first. Con chạy i=1.

+ Bước 2: So sánh Q với NULL và I với k. Nếu Q khác NULL đồng thời i<k thì thực hiện bước 3, nếu không thì qua bước 4.

+ Bước 3: Q=Q🡪Next. i tăng thêm 1. Quay lại bước 2.

+ Bước 4: Kết thúc hàm.

*- Sơ đồ khối:*



## 8. Chèn giá trị X vào vị trí K trong DSLK:

*- Dữ liệu:*

+ Đầu vào : Giá trị x, vị trí k , DSLK

+ Đầu ra : DSLK.

*- Thuật toán:*

+ Bước 1: Nhập giá trị x và vị trí k.

+ Bước 2: Khởi tạo Node P có giá trị x, chạy hàm Tìm Node Q có vị trí k.

+ Bước 3: So sánh Q với NULL .Nếu Q bằng NULL thì thực hiện bước 4, nếu không thì qua bước 5.

+ Bước 4: Kết luận không chèn.

+ Bước 5: Khai báo Node R=first.

+ Bước 6: So sánh R🡪Next với Q. Nếu khác thì thực hiện bước 7, nếu không thì qua bước 8.

+ Bước 7: Gán R=R🡪Next. Quay lại bước 6.

+ Bước 8: Gán R🡪Next=P và P🡪Next=Q.

+ Bước 9: Kết thúc hàm.

*- Sơ đồ khối:*



## 9. Xoá một phần tử tại vị trí k trong DSLK:

*- Dữ liệu:*

+ Đầu vào: vị trí k, DSLK

+ Đầu ra: DSLK

*-Thuật toán:*

+ Bước 1: Nhập vị trí k.

+ Bước 2: Khởi chạy hàm Tìm Node Q có vị trí k.

+ Bước 3: So sánh Q với NULL .Nếu Q bằng NULL thì thực hiện bước 4, nếu không thì qua bước 5.

+ Bước 4: Kết luận không xóa.

+ Bước 5: Khai báo Node R=first.

+ Bước 6: So sánh R🡪Next với Q. Nếu khác thì thực hiện bước 7, nếu không thì qua bước 8.

+ Bước 7: Gán R=R🡪Next. Quay lại bước 6.

+ Bước 8: Gán R🡪Next=Q🡪Next. xóa Node Q.

+ Bước 9: Kết thúc hàm.

*- Sơ đồ khối:*



## 10. Xoá các số âm trong DSLK:

*- Dữ liệu:*

+ Đầu vào: DSLK

+ Đầu ra: DSLK

*- Thuật toán:*

+ Bước 1: Khai báo Node P=first, con chạy k=1

+ Bước 2: So sánh P với NULL .Nếu P bằng NULL thì thực hiện bước 3, nếu không thì qua bước 4.

+ Bước 3: Kết luận không xóa được

+ Bước 4: So sánh P với NULL .Nếu P khác NULL thì thực hiện bước 5, nếu không thì qua bước 8.

+ Bước 5: So sánh xem P🡪Info có âm hay không. Nếu có thực hiện bước 6, nếu không thực hiện bước 7.

+ Bước 6: Chạy hàm xóa Node tại vị trí k. Gán P=P🡪Next. Quay lại bước 4

+ Bước 7: Gán P=P🡪Next, k tăng thêm 1. Quay lại bước 4.

+ Bước 8: Kết thúc hàm.

*- Sơ đồ khối:*



CHƯƠNG 3: ỨNG DỤNG DANH SÁCH LIÊN KẾT ĐƠN

1. Giới thiệu chung về phần mềm

Quản lý sinh viên là công việc hằng ngày mà phòng quản lý sinh viên. Công việc quản lý sinh viên đòi hỏi tính tỉ mỉ, cẩn thận trong từng khâu ghi chép các thông tin cá nhân của sinh viên : mã số sinh viên, tên sinh viên, ngày tháng năm sinh, khóa, lớp…cũng như công việc thống kê kết quả học tập của sinh viên cần pahir rõ ràng và chính xác cao. Trước đây công nghệ thông tin chưa phát triển mạnh mẽ, các công việc được xử lý thủ công, chủ yếu là ghi chép bằng tay, sổ sách chính vì vậy rất tốn thời gian và công sức. Ngày nay khi mà khoa học kỹ thuật phát triển, đặc biệt là sự bùng nổ công nghệ thông tin việc quản lý sinh viên trở lên dễ dàng hơn. Xuất phát từ nhu câu đó bài toán Quản Lý Sinh Viên ra đời. Quản lý sinh viên là chương trình quản lý hồ sơ, điểm học tập cuarsinh viên trong quá trình theo học tai trường. Chương trình có thẻ thực hiện các công việc thêm mới sinh viên, tìm kiếm sinh viên theo một điều kiện nào đó, xóa sinh viên, liệt kê danh sách. Mọi công việc phải được thao tác trên một vùng dữ liệu chung để đảm bảo việc đồng bộ với nhau trong khâu quản lý sinh viên. Chương trình được viết bằng gôn ngữ C++ và dực trên cấu trức lưu trữ của danh sách liên kết đơn.

2. Phân tích thiết kế phần mềm:

Để cho việc quản lý sinh viên dễ dàng, chặt chẽ chương trình Quản Lý Sinh Viên sẽ gồm có :

* Msv (mã số sinh viên) : có kiểu dữ liệu int, mỗi sinh viên có một mã số riêng, không trùng lặp.
* HoTen (họ và tên) : có kiểu dữ liệu char, độ dài ký tự 40, họ và tên sinh viên có thể trùng nhau.
* CTDL (điểm cấu trúc dữ liệu) : có kiểu float, đây là kết quả học tập cấu trúc dữ liệu của sinh viên được nhập từ giáo viên.
* OOP (lập trình hướng đối tượng) : có kiểu float, đây là kết quả học tập môn lập trình hướng đối tượng của sinh viên được nhập từ giáo viên.
* Ngay, thang nam : có kiểu int, là ngày tháng năm sinh của sinh viên.
* TK : là điểm tổng các môn học.

KẾT LUẬN

- Các kết quả đạt được:

+ Nẵm vững lý thuyết về con trỏ và cách vận dụng

+ Hiểu về Danh sách liên kết đơn và các thao tác cơ bản trên Danh sách liên kết đơn

+ Nắm được các thao tác khi thực hiện phân tích thuật toán và thiết kế một chương trình vận dụng danh sách liên kết đơn để giải các bài toán cơ bản.

+ Nắm được cách viết một chương trình trên Turbo C++ IDE

- Hạn chế:

+ Do thời gian nghiên cứu và kiến thức còn hạn chế nên các lý thuyết và phương pháp để giải các bài toán trong đồ án này chưa hiệu quả, chưa sử dụng phương pháp khác để giải bài toán, còn mất nhiều công đoạn để giải quyết bài toán.

+ Mới chỉ đề cập đến một số bài toán cơ bản và đơn giản, chưa làm được các bài toán đặc biệt và phức tạp.

- Hướng phát triển:

+ Tìm hiểu thêm các cách khác nhau để cùng giải quyết một bài toán và từ đó lựa chọn phương pháp tối ưu nhất.

+ Tìm hiểu sâu hơn về các loại Danh sách liên kết còn lại.

+ Tìm hiểu các bải toán phức tạp hơn như dùng Danh sách liên kết để quản lý điểm của sinh viên…v…v.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Lưu Văn Hiền-Huỳnh Bá Diệu-Nguyễn Thị Bảo Trang, Giới thiệu cấu trúc dữ liệu và giải thuật, Khoa CNTT-DTU,08/2009.

[2] Dương Trần Đức, Giáo trình cấu trúc dữ liệu và giải thuật, Học viện Công nghệ Bưu chính viễn thông, 2010.

[3] Trần Hạnh Nhi-Dương Anh Đức, Giáo trình cấu trúc dữ liệu và giải thuật, NXB Đại học Quốc gia Hồ Chí Minh, 2010.

[4] <https://daynhauhoc.com/t/giai-dap-ve-code-quan-ly-sinh-vien-bang-danh-sach-lien-ket/24461>.