

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH



BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI: CHƯƠNG TRÌNH GIỎ HÀNG

Giảng viên hướng dẫn:	Trần Thị Dung
Lớp:	CNTT.K60
Thành viên:	MSSV:
Lê Hùng Vỹ	6051071146
Nguyễn Hoàn Tín	6051071123
Nguyễn Cường Quốc	6051071098
Mã học phần:	CPM215.3

TP.Hồ Chí Minh, năm 2020

LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện tại môn Lập trình nâng cao trường Đại học Giao thông Vận tải – Phân hiệu tại thành phố Hồ Chí Minh em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành đề tài bài tập lớn của chúng em.

Chúng em xin gửi lời cảm ơn chân thành đến quý thầy, cô bộ môn Công nghệ thông tin trường Đại học Giao thông Vận tải – Phân hiệu tại thành phố Hồ Chí Minh đã quan tâm hướng dẫn truyền đạt học những kiến thức và kinh nghiệm cho chúng em trong suốt thời gian học tập vừa qua, và thực hiện bài tập lớn một cách tận tình và tâm huyết. Em xin chúc quý thầy cô thật nhiều sức khỏe và luôn đạt được thành công trong cuộc sống. Đặc biệt em xin cảm ơn cô Trần Thị Dung người đã trực tiếp hướng dẫn em và chỉ bảo em trong quá trình thực hiện đề tài bài tập lớn. Cô đã cùng em góp ý và xây dựng đề tài “Chương trình giỏ hàng”.

Sau một thời gian nỗ lực thực hiện thì đề tài cũng đã hoàn thành. Nhưng không sao tránh khỏi những sai sót do em còn chưa có nhiều kinh nghiệm thực tế. Em kính mong nhận được sự góp ý và nhận xét từ quý thầy, cô để em có thể hoàn thiện và hoàn thành tốt hơn cho đề tài của mình.

Lời sau cùng em một lần nữa kính chúc quý thầy, cô bộ môn Công nghệ thông tin Trường Đại học Giao thông Vận tải – Phân hiệu tại thành phố Hồ Chí Minh thật nhiều sức khỏe và thành công.

Tp. Hồ Chí Minh, ngày 20 tháng 6 năm 2020

MỤC LỤC

PHẦN MỘT: BÀI TẬP LỚN.....	4
I. Lý do chọn đề tài:.....	4
II. Giải thích code:	4
III. Hướng dẫn sử dụng:.....	6
PHẦN HAI: CƠ SỞ LÝ THUYẾT(Bổ sung)	9
I. Làm việc với tệp:.....	9
1. File văn bản – text files	9
2. File nhị phân – Binary files	9
3. Các thao tác làm việc với file:	10
II. Cấu trúc liên kết đơn.....	12
Cài đặt danh sách liên kết đơn:.....	13
III. Các thuật toán sắp xếp :.....	20
1. Sắp xếp đổi chỗ:	20
2. Sắp xếp chọn (selection sort):.....	21
3. Sắp xếp nổi bọt (bubble sort):	23
4. Sắp xếp hỗn hợp (cocktail shaker sort):	25
5. Sắp xếp chèn (insertion sort):	26
6. Chèn nhị phân (binary insertion sort):	27
7. Sắp xếp nhanh (quick sort):	27
8. Sắp xếp trộn (merge sort):	29
9. Sắp xếp vun đống (heap sort):.....	32
IV. Các thuật toán tìm kiếm :	34
1. Tìm kiếm tuyến tính:	34
2. Tìm kiếm nhị phân:	34

PHẦN MỘT: BÀI TẬP LỚN

I. Lý do chọn đề tài:

Trong xã hội ngày càng phát triển hiện nay, khoa học công nghệ là thứ không thể thiếu đối với mỗi quốc gia, doanh nghiệp, trường học hay mỗi cá nhân, đặc biệt là công nghệ thông tin. Với sự phát triển nhanh một cách không ngừng nghỉ như vậy của công nghệ thông tin đã giúp giải quyết các công việc học tập, nguyên cứu, quản lý thông tin,... một cách dễ dàng và tiện lợi. Thấy được tiềm năng đó các quốc gia, doanh nghiệp, trường học, các cá nhân, ... đã ứng dụng nó vào thực tiễn cuộc sống để giải quyết công việc, học tập, giải trí với những chiếc điện thoại thông minh nhỏ gọn.

Xã hội đang trong giai đoạn 4.0 các công việc của con người được thực hiện và giải quyết nhanh hơn nhờ vào thiết bị máy móc hiện đại, các chương trình giúp quản lí công việc. Hầu như trong mọi việc đều có sự góp mặt của công nghệ 4.0. Đặc biệt là các chương trình ứng dụng trong công việc tính toán, quản lí, liên lạc... phát triển vô cùng mạnh mẽ.

Nhận thấy được tiềm năng đó nhóm em đã tìm hiểu và thực hiện đề tài **“quản lí giỏ hàng”**. Nhằm giúp đỡ người sử dụng đặc biệt là thu ngân tiện lợi trong việc tổng hợp sản phẩm và giá cả chỉ với một vài bước đơn giản. Ứng dụng sử dụng dễ dàng với mọi người kể cả là một người không rành về lập trình.

II. Giải thích code:

Struct `giohang_st` là phần định nghĩa kiểu dữ liệu với tên là `giohang_st` bao gồm `tenkhachhang` với kiểu dữ liệu `char` cho phép nhập tên người dùng, `diachi` với kiểu dữ liệu `char` cho phép nhập địa chỉ người dùng, `sosanpham` với kiểu dữ liệu `int` cho phép nhập số sản phẩm đã thêm vào, `dssp` với kiểu dữ liệu `st_sp*` cho phép người dùng nhập vào 1 số thông tin tên sản phẩm, giá, số lượng.

*`st_sp` là tên của một struct bao gồm:

- `tensp`: nhập tên sản phẩm
- `soluong`: nhập số lượng sản phẩm
- `gia`: nhập giá thành của sản phẩm

Ta gọi `giohang` dùng để tạo giỏ hàng nhập thông tin cá nhân của người dùng sử dụng hàm void với tham chiếu là con trỏ `a` kiểu dữ liệu `giohang_st` trong hàm sử dụng các lệnh xuất `printf()` để xuất thông báo nhập thông tin của người dùng

bao gồm tên và địa chỉ. Tiếp theo sử dụng hàm `gets(a->tenkhachhang)` và `gets(a->tenkhachhang)` để nhập chuỗi ký tự tên và địa chỉ người dùng, `fflush` đứng trước `gets(stdin)` dùng để xóa ký tự đệm. Cuối cùng `a->sosanpham=0` để khởi tạo số sản phẩm bằng 0.

`Chonhang` dùng để nhập sản phẩm của người dùng sử dụng hàm `void` với con trỏ `a` kiểu dữ liệu `giohang_st` trong hàm sử dụng các lệnh xuất `printf` để xuất thông báo nhập tên sản phẩm, nhập số lượng sản phẩm, nhập giá sản phẩm. Tiếp theo sử dụng hàm `gets(a->dssp[a->sosanpham-1].tensp)`, `scanf("%d", &a->dssp[a->sosanpham-1].soluong)` và `scanf("%d", &a->dssp[a->sosanpham-1].gia)` để nhập tên sản phẩm, số lượng sản phẩm và giá của sản phẩm, `fflush(stdin)` đứng trước `gets()` và `scanf` dùng để xóa ký tự đệm.

`Sapxep` dùng để sắp xếp tên sản phẩm theo chiều A-Z sử dụng vòng lặp `for` và `if` để thực hiện thuật toán với ý tưởng thuật toán đổi chỗ để hoán vị 2 sản phẩm.

`xemchitietgiohang` dùng để hiển thị toàn bộ thông tin đã nhập trước đó sử dụng hàm `void` với tham trị `a` kiểu dữ liệu `giohang_st` sử dụng vòng lặp `for` để xuất toàn bộ thông tin đã nhập trước đó và `*Tonggia` lên màn hình console.

* `Tonggia` với kiểu dữ liệu `float` được gán trị = 0 và được thêm vào trong vòng lặp `for` để thực hiện tính tổng giá tiền sản phẩm với công thức: `tonggia+= (soluong*gia)`

`loaibomotmathang` dùng để loại bỏ 1 một sản phẩm sử dụng hàm `void` với tham chiếu con trỏ `a` kiểu dữ liệu `giohang_st`, dùng vòng lặp `for` để làm thuật toán tìm kiếm sản phẩm cần loại bỏ và `if else` để phân chia trường hợp giỏ hàng có sản phẩm hay không có sản phẩm để code chạy theo từng trường hợp tương ứng.

`ghitepnhiphan` dùng để ghi toàn thông tin sau khi đã thêm bớt trước đó vào tệp `cart.dat` sử dụng hàm `void` với tham trị `a` kiểu dữ liệu `giohang_st` và tham chiếu `p` kiểu dữ liệu `FILE`, `f p = fopen("cart.dat", "wb")` để mở tệp với tên `cart.dat` và `wb` xác định tệp là ghi vào tệp kiểu nhị phân, `fwrite` để ghi thông tin tên khách hàng, địa chỉ, số sản phẩm vào file, còn danh sách sản phẩm sử dụng vòng lặp `for` để ghi vào file, `fclose` để đóng file lại sau ghi.

`Menu()` sử dụng hàm `void`, dùng `switch case` để tạo ra giao diện với 7 case (trường hợp):

- ✓ case 1 : Tạo giỏ hàng gọi hàm `taogiohang` thực hiện các chức năng được ghi sẵn.
- ✓ case 2: Chọn sản phẩm gọi hàm `chonhang` thực hiện các chức năng được ghi sẵn.

- ✓ case 3: xem chi tiet gio hang sử dụng hàm if else để đưa ra 2 trường giỏ hàng có số sản phẩm = 0 sẽ xuất thông báo giỏ trống trong khi trường số sản phẩm > 0 code hàm sapxep và hàm xemchitietgiohang được gọi thực hiện chức năng được định sẵn .
- ✓ case 4: loại bỏ 1 mặt hàng gọi hàm loaibomotmathang để thực chức năng được định sẵn.
- ✓ case 5: Ghi toan bo noi dung gio hang vào tệp nhị phân cart.dat gọi hàm ghitepnhiphan để thực hiện chức năng được định sẵn.
- ✓ case 6 : Thoat sử dụng switch case để thực hiện 2 lựa chọn 1 YES để thoát và 2 NO để quay lại menu.

Hướng dẫn sử dụng:

Bước 1: Để thiết lập một thống kê các sản phẩm khách hàng đã chọn, trước tiên bạn cần tạo một giỏ hàng có đầy thông tin về khách hàng bao gồm: Tên khách hàng, địa chỉ khách hàng. Bạn nhấn phím 1, sau đó nhấn "enter"(giỏ hàng của khách đã được thiết lập)

```

=====M E N U=====
| 1. Tao gio hang
| 2. Chon san pham
| 3. Xem chi tiet gio hang
| 4. Loai bo 1 mat hang
| 5. Ghi toan bo noi dung gio hang vào tệp nhị phân cart.dat
| 6. Thoat
=====

Ban chon: 1

Ban da chon Tao gio hang
-----
Nhap ten khách hàng:
Hung Vy
Nhap địa chỉ:
Quan Phu nhuan

***GIO CUA BAN DA DUOC KHOI TAO***

-----
nhấn ENTER để chọn lại menu

```

Bước 2: Sau khi đã thiết lập được một giỏ hàng, cần phải có đầy đủ thông tin về sản phẩm, số lượng và đơn giá của những sản phẩm mà khách đã chọn nhấn phím 2 và "enter". Một bảng thống kê sẽ xuất hiện, việc còn lại của bạn là phải nhập thông tin như bảng thống kê đã yêu cầu. Nếu có từ 2 sản phẩm trở lên bạn chỉ cần "enter" và thực hiện lại bước 2.

```
=====M E N U=====
|1. Tao gio hang
|2. Chon san pham
|3. Xem chi tiet gio hang
|4. Loai bo 1 mat hang
|5. Ghi toan bo noi dung gio hang vao tep nhi phan cart.dat
|6. Thoat
=====

Ban chon: 2
Ban da chon 'Chon hang'
Nhap ten san pham: Sua TH

Nhap so luong: 3

Nhap gia: 7500

***NHAP SAN PHAM THANH CONG***

nhan ENTER de chon lai menu
_

=====M E N U=====
|1. Tao gio hang
|2. Chon san pham
|3. Xem chi tiet gio hang
|4. Loai bo 1 mat hang
|5. Ghi toan bo noi dung gio hang vao tep nhi phan cart.dat
|6. Thoat
=====

Ban chon: 2
Ban da chon 'Chon hang'
Nhap ten san pham: Banh poca

Nhap so luong: 2

Nhap gia: 5000

***NHAP SAN PHAM THANH CONG***

nhan ENTER de chon lai menu
_
```

Bước 3: Để có thể cho khách hàng xem chi tiết về giỏ hàng và những mặt hàng, số lượng hàng hóa họ đã mua nhấn phím 3 và "enter". Một bảng thống kê đầy đủ về hàng hóa, số lượng, đơn giá của các mặt hàng. Ngoài ra còn có thêm tính năng tính tổng tiền giúp cho việc tính tiền trở nên nhanh hơn.

```
=====M E N U=====
|1. Tao gio hang
|2. Chon san pham
|3. Xem chi tiet gio hang
|4. Loai bo 1 mat hang
|5. Ghi toan bo noi dung gio hang vao tep nhi phan cart.dat
|6. Thoat
=====

Ban chon: 3
Ban da chon xem chi tiet gio hang
*****CHI TIET GIO HANG*****
| Ten Khach hang : Hung Vy
| Dia chi: Quan Phu nhuan
| So san pham: 2
|-----|
|STT |Ten san pham |So luong |Gia |
|-----|
|1 |Banh poca |2 |5000 |
|-----|
|2 |Sua TH |3 |7500 |
|-----|
|| Tong gia: 32500 ||
*****
```

Bước 4: Nếu như khách trả lại một mặt hàng hay bớt đi số lượng hàng vì một lý do nào đó, bạn chọn 4 và "enter". điều bạn cần làm lúc này là loại bỏ mặt hàng mà khách không cần nữa, bạn chỉ cần nhập đúng số thứ tự của mặt hàng cần được loại bỏ. Nếu muốn kiểm tra lại hàng hóa đã được loại bỏ hay chưa bạn chỉ cần lặp lại bước 3.

```
=====M E N U=====
|1. Tao gio hang
|2. Chon san pham
|3. Xem chi tiet gio hang
|4. Loai bo 1 mat hang
|5. Ghi toan bo noi dung gio hang vao tep nhi phan cart.dat
|6. Thoat
=====

Ban chon: 4
Ban da chon Loai bo 1 mat hang
Can loai bo san pham thu may: 1
***LOAI BO SAN PHAM THANH CONG***

nhan ENTER de chon lai menu
_
```

*Nếu bạn muốn xuất thông tin vào file để lưu trữ bạn chọn phím 5 để ghi thông tin vào file cart.dat.


```
=====M E N U=====
|1. Tao gio hang
|2. Chon san pham
|3. Xem chi tiet gio hang
|4. Loai bo 1 mat hang
|5. Ghi toan bo noi dung gio hang vao tep nhi phan cart.dat
|6. Thoat
=====
Ban chon: 5
Ban da chon Ghi toan bo noi dung gio hang vao tep nhi phan cart.dat'
***GHI VAO THANH CONG!***

nhan ENTER de chon lai menu
```

Cuối cùng nếu bạn muốn thoát chương trình nhấn 6 , lúc này màn hình sẽ xuất hiện 2 thông tin:
Thứ nhất là quay lại menu để tiếp tục công việc. Để tiếp tục bạn nhấn 1.
Thứ hai là kết thúc chương trình làm việc. Để kết thúc bạn nhấn 2.

PHẦN HAI: CƠ SỞ LÝ THUYẾT(Bổ sung)

I. Làm việc với tệp:

Có hai kiểu file khác nhau cần phân biệt: file văn bản và file nhị phân.

Một tệp đại diện cho một chuỗi byte, bất kể đó là tệp văn bản hay tệp nhị phân. Ngôn ngữ lập trình C cung cấp quyền truy cập vào các chức năng cấp cao cũng như các cuộc gọi cấp thấp (cấp độ hệ điều hành) để xử lý tệp trên thiết bị lưu trữ của bạn.

1. File văn bản – text files

File văn bản là file thường có đuôi là “.txt”. Những file này bạn có thể dễ dàng tạo ra bằng cách dùng các text editor thông dụng như Notepad, Notepad++, Sublime Text,...

Khi bạn mở các file này bằng các text editor nói trên, bạn sẽ thấy được văn bản ngay và có thể dễ dàng thao tác sửa, xóa, thêm nội dung của file này.

Kiểu file này thuận tiện cho chúng ta trong việc sử dụng hàng ngày, nhưng nó sẽ kém bảo mật và cần nhiều bộ nhớ để lưu trữ hơn.

2. File nhị phân – Binary files

File nhị phân thường có đuôi mở rộng là “.bin”.

Thay vì lưu trữ dưới dạng văn bản thuần túy, các file này được lưu dưới dạng nhị phân, chỉ bao gồm các số 0 và 1. Bạn cũng sẽ thấy các con số này nếu cố mở nó bằng 1 text editor kể trên.

Loại file này giúp lưu trữ được dữ liệu với kích thước lớn hơn, không thể đọc bằng các text editor thông thường và thông tin lưu trữ ở loại file được bảo mật hơn so với file văn bản.

3. Các thao tác làm việc với file:

a. Thao tác mở file:

Để đọc ghi file trong C cũng như trong mọi ngôn ngữ lập trình, việc đầu tiên bạn cần làm là mở file mà bạn muốn làm việc. Trong ngôn ngữ lập trình C, chúng ta có thể mở file bằng cách sử dụng hàm `fopen()` trong thư viện `stdio.h` như sau:

```
fptr = fopen("fileopen","mode");
```

Trong đó mode là một tham số chúng ta cần chỉ định.

Các tham số của “mode”:

Mode	Ý nghĩa
r	Mở file chỉ cho phép đọc
rb	Mở file chỉ cho phép đọc dưới dạng nhị phân.
w	Mở file chỉ cho phép ghi.
wb	Open for writing in binary mode.
a	Mở file ở chế độ ghi “append”. Tức là sẽ ghi vào cuối của nội dung đã có.
ab	Mở file ở chế độ ghi nhị phân “append”. Tức là sẽ ghi vào cuối của nội dung đã có.
r+	Mở file cho phép cả đọc và ghi.
rb+	Mở file cho phép cả đọc và ghi ở dạng nhị phân.
w+	Mở file cho phép cả đọc và ghi.
wb+	Mở file cho phép cả đọc và ghi ở dạng nhị phân.
a+	Mở file cho phép đọc và ghi “append”.
ab+	Mở file cho phép đọc và ghi “append” ở dạng nhị phân.

b. Thao tác đóng file

Khi làm việc với tập tin hoàn tất, kể cả là file nhị phân hay file văn bản cần đóng file sau khi đã làm việc xong.

Việc đóng file đang mở có thể được thực hiện bằng cách dùng hàm `fclose()`.

```
fclose(fptr);
```

c. Đọc/Ghi file văn bản trong C:

Để làm việc với file văn bản, chúng ta sẽ sử dụng `fprintf()` và `fscanf()`.

VD minh họa:

Ghi file sử dụng `fprintf()`

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main()
```

```
{
```

```
    int num;
```

```
    FILE *fptr;
```

```
    fptr = fopen("C:\\program.txt","w");
```

```
    if(fptr == NULL)
```

```
    {
```

```
        printf("Error!");
```

```
        exit(1);
```

```
    }
```

```
    printf("Enter num: ");
```

```
    scanf("%d",&num);
```

```
    fprintf(fptr,"%d",num);
```

```
    fclose(fptr);
```

```
    return 0; }
```

Chương trình nhận số num từ bàn phím và ghi vào file văn bản program.txt.

Sau khi bạn chạy chương trình này, bạn sẽ thấy file văn bản program.txt được tạo mới trong ổ C trên máy tính bạn. Khi mở file này lên, bạn sẽ thấy số mà bạn vừa nhập cho biến “num” kia.

Đọc file sử dụng fscanf()

```
#include <stdio.h>

#include <stdlib.h>

int main()
{
    int num;
    FILE *fptr;
    if ((fptr = fopen("C:\\program.txt", "r")) == NULL){
        printf("Error! opening file");
        // Program exits if the file pointer returns NULL.
        exit(1);
    }
    fscanf(fptr, "%d", &num);
    printf("Value of n=%d", num);
    fclose(fptr);
    return 0;
}
```

Chương trình này sẽ đọc giá trị số được lưu trong file program.txt mà chương trình vừa tạo ra và in lên màn hình.

II. Cấu trúc liên kết đơn

Danh sách được liên kết là một nhóm các đối tượng dữ liệu trong đó mỗi đối tượng có một con trỏ tới đối tượng trong nhóm. Mọi thứ bạn làm với danh sách được liên kết đều có thể được thực hiện trong bộ nhớ hoặc là một phần của tập tin đĩa.

Danh sách liên kết đơn là một tập hợp các Node được phân bố động, được sắp xếp theo cách sao cho mỗi Node chứa “một giá trị”(Data) và “một con trỏ”(Next). Con trỏ sẽ trỏ đến phần tử kế tiếp của danh sách liên kết đó. Nếu con trỏ mà trỏ tới NULL, nghĩa là đó là phần tử cuối cùng trong danh sách liên kết.



(2.1 Hình ảnh mô phỏng một danh sách liên kết đơn đầy đủ)

Cài đặt danh sách liên kết đơn:

a. Khai báo linked list

Để đơn giản hóa, data của chúng ta sẽ là số nguyên(int). Bạn cũng có thể sử dụng các kiểu nguyên thủy khác(float, char,...) hay kiểu dữ liệu struct tự tạo.

Khai báo trên sẽ được sử dụng cho mọi Node trong linked list. Trường data sẽ lưu giữ giá trị và next sẽ là con trỏ để trỏ đến vị trí kế tiếp của nó.

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

b. Tạo mới 1 Node

Tạo một kiểu dữ liệu của struct để code được gọn hơn:

```
typedef struct LinkedList *node;  
node CreateNode (int value){  
    node temp;  
    temp = (node)malloc(sizeof(struct LinkedList));  
    malloc()  
    temp->next = NULL;
```

```

temp->data = value;

return temp;
}

```

Mỗi một Node khi được khởi tạo, chúng ta cần cấp phát bộ nhớ cho nó, và mặc định cho con trỏ next trỏ tới NULL. Giá trị của Node sẽ được cung cấp khi thêm Node vào linked list. Trong đó:

- typedef được dùng để định nghĩa một kiểu dữ liệu trong C. VD: `typedef long long LL;`
- malloc là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng `new`
- sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc

c. Thêm Node vào danh sách liên kết:

Thêm vào đầu

Việc thêm vào đầu chính là việc cập nhật lại vị trí head. Ta gọi Node mới(temp), ta có:

Nếu head đang trỏ tới NULL, nghĩa là linked list đang trống, Node mới thêm vào sẽ làm head luôn

Ngược lại, ta phải thay thế vị trí head cũ bằng head mới. Việc này phải làm theo thứ tự như sau:

- Cho next của temp trỏ tới head hiện hành
- Đặt temp làm head mới

```

node AddHead(node head, int value){
node temp = CreateNode(value);

if(head == NULL){
head = temp;
}

else{
temp->next = head;
head = temp;
}
}

```

```
return head;
}
```

Thêm vào cuối

Chúng ta sẽ cần Node đầu tiên, và giá trị muốn thêm. Khi đó, ta sẽ:

- i. Tạo một Node mới với giá trị value
- ii. Nếu head = NULL, tức là danh sách liên kết đang trống. Khi đó Node mới(temp) sẽ là head luôn.
- iii. Ngược lại, ta sẽ duyệt tới Node cuối cùng(Node có next = NULL), và trở next của vị trí cuối tới Node mới(temp)

```
node AddTail(node head, int value){
    node temp,p;
    temp = CreateNode(value);
    if(head == NULL){
        head = temp;    }
    else{
        p = head;
        while(p->next != NULL){
            p = p->next;
        }
        p->next = temp
    }
    return head;
}
```

Thêm vào vị trí bất kì

Để làm được việc này, ta phải duyệt từ đầu để tìm tới vị trí của Node cần chèn, giả sử là Node Q, khi đó ta cần làm theo thứ tự sau:

- Cho next của Node mới trở tới Node mà Q đang trở tới
- Cho Node Q trở tới Node mới

Xóa Node khỏi danh sách liên kết

Xóa đầu

```
node DelHead(node head){
```

```

if(head == NULL){
printf("\nCha co gi de xoa het!");
}else{
head = head->next;
}
return head;
}

```

Xóa cuối

```

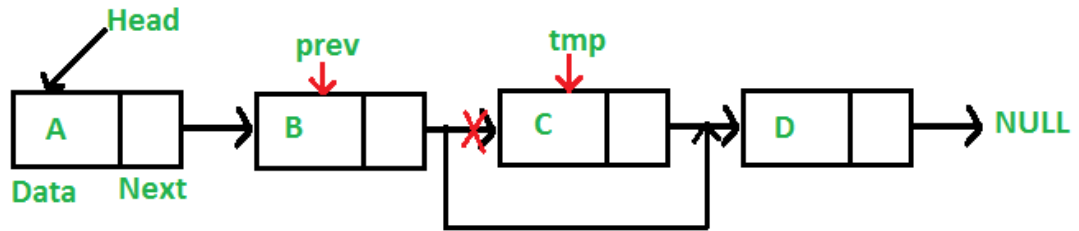
node DelTail(node head){
if (head == NULL || head->next == NULL){
return DelHead(head);
}
node p = head;
while(p->next->next != NULL){
p = p->next;
}
p->next = p->next->next; // Cho next bằng NULL
// Hoặc viết p->next = NULL cũng được
return head;
}

```

Thằng Node cuối – 1 là thằng có `p->next->next = NULL`. Bạn cho next của nó bằng NULL là xong.

Xóa ở vị trí bất kỳ

Việc xóa ở vị trí bất kỳ cũng khá giống xóa ở cuối kia. Đơn giản là chúng ta bỏ qua một phần tử, như ảnh sau:



(2.1c Hình ảnh minh họa)

Xóa node trong danh sách liên kết

Xóa node trong danh sách liên kết

Lưu ý: Chỉ số xóa bắt đầu từ 0 nhé các bạn. Việc tìm vị trí cần xóa chỉ duyệt tới Node gần cuối thôi(cuối – 1). Sau đây là code xóa Node ở vị trí bất kỳ

```

node DelAt(node head, int position){
if(position == 0 || head == NULL || head->next == NULL){
head = DelHead(head); // Nếu vị trí chèn là 0, tức là thêm vào đầu
}else{
// Bắt đầu tìm vị trí cần chèn. Ta sẽ dùng k để đếm cho vị trí
int k = 1;
node p = head;
while(p->next->next != NULL && k != position){
p = p->next;
++k;
}

if(k != position){
// Nếu duyệt hết danh sách lk rồi mà vẫn chưa đến vị trí cần chèn,
ta sẽ mặc định xóa cuối
// Nếu bạn không muốn xóa, hãy thông báo vị trí xóa không hợp lệ
head = DelTail(head);
// printf("Vi tri xoa vuot qua vi tri cuoi cung!\n");
}else{
p->next = p->next->next;
}
}
  
```

```

    }
    }
    return head;
}

```

d. Lấy giá trị ở vị trí bất kỳ

Chúng ta sẽ viết một hàm để truy xuất giá trị ở chỉ số bất kỳ nhé. Trong trường hợp chỉ số vượt quá chiều dài của linked list – 1, hàm này trả về vị trí cuối cùng. Do hạn chế là chúng ta không thể raise error khi chỉ số không hợp lệ. Tôi mặc định chỉ số bạn truyền vào phải là số nguyên không âm. Nếu bạn muốn kiểm tra chỉ số hợp lệ thì nên kiểm tra trước khi gọi hàm này.

```

int Get(node head, int index){
    int k = 0;
    node p = head;
    while(p->next != NULL && k != index){
        ++k;
        p = p->next;
    }
    return p->data;
}

```

Lý do dùng `p->next != NULL` là vì chúng ta chỉ muốn đi qua các phần tử có value.

e. Tìm kiếm trong danh sách liên kết

Hàm tìm kiếm này sẽ trả về chỉ số của Node đầu tiên có giá trị bằng với giá trị cần tìm. Nếu không tìm thấy, chúng ta trả về -1.

```

int Search(node head, int value){
    int position = 0;
    for(node p = head; p != NULL; p = p->next){
        if(p->data == value){

```

```

return position;
}
++position;
}
return -1;
}

```

Chúng ta có thể sử dụng hàm này để xóa tất cả các Node trong danh sách liên kết có giá trị chỉ định như sau:

```

node DelByVal(node head, int value){
int position = Search(head, value);
while(position != -1){
DelAt(head, position);
position = Search(head, value);
}
return head;
}

```

f. Duyệt danh sách liên kết

Việc duyệt danh sách liên kết cực đơn giản. Khởi tạo từ Node head, bạn cứ thế đi theo con trỏ next cho tới trước khi Node đó NULL.

```

void Traverser(node head){
printf("\n");
for(node p = head; p != NULL; p = p->next){
printf("%5d", p->data);
}
}

```

}

III. Các thuật toán sắp xếp :

- ✓ Cách sắp xếp được thực hiện trong C:
- ✓ Sắp xếp có thể được thực hiện theo nhiều cách khác nhau dựa trên thuật toán sắp xếp . Trong ngôn ngữ lập trình C, chúng tôi có một số cách tiếp cận để sắp xếp danh sách. Các trạng thái sắp xếp thuật ngữ sắp xếp dữ liệu theo cách cụ thể thường theo thứ tự tăng dần. Mặc dù cách sắp xếp dữ liệu là khác nhau trong tất cả các thuật toán sắp xếp , kết quả của tất cả chúng là như nhau.
- ✓ Thông thường, trong việc sắp xếp, chương trình tìm kiếm số lượng tối thiểu và chuyển số đó sang đầu danh sách và lặp lại các tìm kiếm tương tự. Một lần nữa khi gặp phải một số lượng nhỏ khác, nó được chuyển sang không gian tiếp theo trong danh sách ngay sau chỉ mục đầu tiên và quá trình này tiếp tục lặp lại cho đến khi có được danh sách sắp xếp. Đây là cách sắp xếp được thực hiện trong ngôn ngữ lập trình C.
- ✓ Trong tất cả các cách tiếp cận để sắp xếp danh sách, mảng đóng vai trò rất quan trọng trong ngôn ngữ lập trình C. Trong mọi thuật toán, mảng đã được sử dụng để lưu trữ danh sách các phần tử phải được sắp xếp. Ví dụ, trong sắp xếp bong bóng, các phần tử được lưu trữ trong một mảng và các giá trị trong mảng đã được xử lý để chuyển đổi chúng thành một danh sách dữ liệu được sắp xếp.
- ✓ Trong sắp xếp lựa chọn, cùng một mảng đã được coi là hai mảng trong đó mảng đầu tiên được coi là bỏ trống để cho biết các giá trị được sắp xếp trong khi mảng thứ hai giữ danh sách chưa sắp xếp. Để phục vụ mục đích sắp xếp mảng được sử dụng rất thường xuyên thay vì giữ các giá trị trong các biến riêng lẻ. Trong số tất cả các thuật toán, sắp xếp nhanh hoạt động rất nhanh và do đó được đặt tên là sắp xếp nhanh. Nó mất ít thời gian hơn nhiều so với các thuật toán sắp xếp khác.

1. Sắp xếp đổi chỗ:

Xuất phát từ đầu dãy, lần lượt so sánh phần tử đầu dãy với các phần tử còn lại, nếu thấy lớn hơn thì đổi chỗ cho nhau, mục đích là để sau khi quét một lượt, phần tử bé nhất sẽ về đầu dãy.

VD:

```
void interchangeSort (int a[], int n) {  
    for (int i = 0; i < n - 1; i++)  
        for (j = i + 1; j < n; j++)  
            if (a[i] > a[j]) swap(a, i, j);  
}
```

```
void swap (int a[], int i, int j) {  
    int tmp = a[i];  
    a[i] = a[j];  
    a[j] = tmp;  
}
```

2. Sắp xếp chọn (selection sort):

Tìm phần tử nhỏ nhất trước, rồi mới đổi chỗ nó cho phần tử đầu tiên, xong lặp lại với phần còn lại.

VD:

```
#include<stdio.h>  
  
#include<conio.h>  
  
void main()  
{  
    int total_count,counter1,counter2,minimum,temp_value;  
    int a[20];  
    printf("\n Enter the Number of Elements: ");  
    scanf("%d",&total_count);  
    printf("\n Enter %d Elements: ",total_count);
```

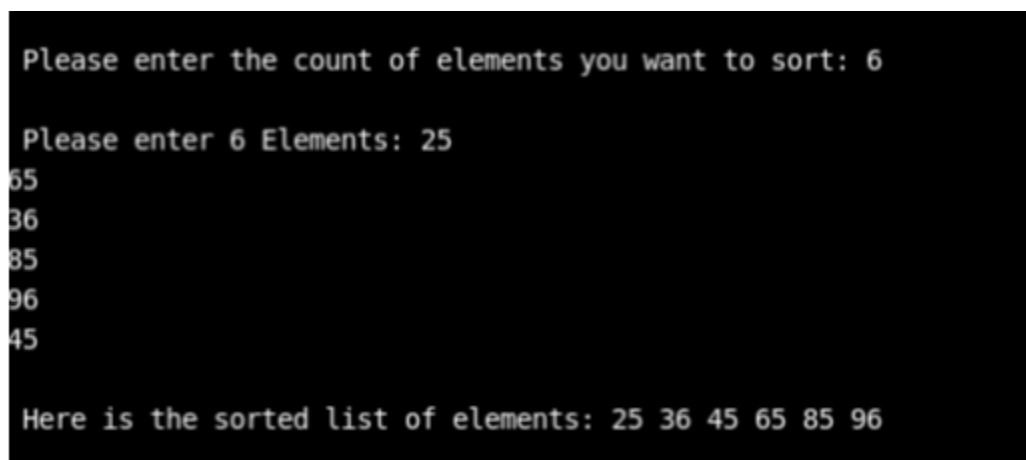
```

for(counter1=0;counter1<total_count;counter1++)
{
scanf("%d",&a[counter1]);
}
for(counter1=0;counter1<total_count-1;counter1++)
{
minimum=counter1;
for(counter2=counter1+1;counter2<total_count;counter2++)
{
if(a[minimum]>a[counter2])
minimum=counter2;
}
if(minimum!=counter1)
{
temp_value=a[counter1];
a[counter1]=a[minimum];
a[minimum]=temp_value;
}
}
printf("\n The Sorted array in ascending order: ");
for(counter1=0;counter1<total_count;counter1++)
{
printf("%d ",a[counter1]);
}

```

```
getch();  
}
```

Đầu ra:



```
Please enter the count of elements you want to sort: 6  
Please enter 6 Elements: 25  
65  
36  
85  
96  
45  
  
Here is the sorted list of elements: 25 36 45 65 85 96
```

3. Sắp xếp nổi bọt (bubble sort):

Lần lượt đổi chỗ các cặp đôi phần tử cạnh nhau nếu chúng ngược thứ tự, mục đích cũng là sau một lượt, phần tử bé nhất sẽ về đầu dãy.

VD:

```
#include <stdio.h>  
  
int main()  
{  
  
int total_count, counter, counter1, swap_var;
```

```

int array[20];

printf("How many number you want to input?\n");

scanf("%d", &total_count);

printf("Please enter %d integers that has to be sorted\n", total_count);

for (counter = 0; counter < total_count; counter++)

scanf("%d", &array[counter]);

for (counter = 0 ; counter < total_count - 1; counter++)
{
for (counter1 = 0 ; counter1 < total_count - counter - 1; counter1++)
{
if (array[counter1] > array[counter1+1]) /* For decreasing order use < */
{
swap_var      = array[counter1];
array[counter1] = array[counter1+1];
array[counter1+1] = swap_var;
}
}
}

printf("Below is the list of elements sorted in ascending order:\n");

for (counter = 0; counter < total_count; counter++)

printf("%d\n", array[counter]);

return 0;

}

```


Đầu ra:

```
How many number you want to input?
7
Please enter 7 integers that has to be sorted
5
3
66
14
1
2
645
Below is the list of elements sorted in ascending order:
1
2
3
5
14
66
645
```

4. Sắp xếp hỗn hợp (cocktail shaker sort):

Lấy hình ảnh xóc bình cocktail, biến j có thể chạy 2 chiều, chiều xuôi thì đưa phần tử lớn nhất về cuối dãy, sau đó đảo lại để đưa phần tử bé nhất về đầu dãy.

VD:

```
void shakerSort (int a[], int n) {
    int left = 0, right = n - 1, k = left;
    while (left < right) {
        for (int i = left; i < right; i++)
```

```

        if (a[i] > a[i + 1]) { swap(a, i, i + 1); k = i; }
    right = k;
    for (int i = right; i > left; i--)
        if (a[i - 1] > a[i]) { swap(a, i - 1, i); k = i; }
    left = k;
}
}

```

Đây là thuật toán cải tiến của sắp xếp nổi bọt nên nó cũng nhận diện được mảng đã sắp xếp. Thuật toán này có lợi thế hơn nổi bọt khi mảng có dài các phần tử đã được sắp xếp.

5. Sắp xếp chèn (insertion sort):

Chúng ta coi phần bên trái của biến chạy i là phần đã được sắp xếp theo đúng thứ tự (sẽ chuyển dần các phần tử từ mớ hỗn độn ở phần bên phải sang). Tìm vị trí thích hợp trong nửa trái đó để chèn phần tử $a[i]$ vào cho đúng thứ tự, sau đó dịch chuyển dần các phần tử bên phải kể từ vị trí vừa tìm ra để nhường lấy một chỗ trống, rồi mới chèn x (mang giá trị của $a[i]$) vào.

VD:

Đầu ra:

```

Please enter the total count of the elements that you want to sort:
6
Please input the elements that has to be sorted:
35
24
68
95
62
3

Output generated after using insertion sort
3 24 35 62 68 95

```

6. Chèn nhị phân (binary insertion sort):

Là cải tiến một chút của insertion sort, thay vì tìm vị trí để chèn một cách tuần tự, ta áp dụng ý tưởng như thuật toán tìm kiếm nhị phân, cứ chia đôi rồi chia đôi.

VD:

```
void binaryInsertionSort (int a[], int n) {  
    int left, right, mid, x;  
    for (int i = 1; i < n; i++) {  
        x = a[i]; left = 0; right = i - 1;  
        while (left <= right) {  
            mid = (left + right) / 2;  
            if (x < a[mid]) right = mid - 1;  
            else left = mid + 1;  
        }  
        for (int j = i - 1; j >= left; j--) a[j + 1] = a[j];  
        a[left] = x;  
    }  
}
```

7. Sắp xếp nhanh (quick sort):

Chọn phần tử x ở giữa mảng. Tìm ở bên trái của x phần tử nào lớn hơn x, ở bên phải có phần tử nào nhỏ hơn x, rồi đổi chỗ 2 phần tử đó. Quét hết lượt thì lặp lại thuật toán với từng nửa. Đến khi nào độ dài các đoạn để quét là 1 thì dừng.

VD:

```
#include <stdio.h>  
void quicksort_method (int [], int, int);  
int main()  
{
```

```

int element_list[50],count, counter;
printf("Please enter the total count of the elements that you want to sort: ");
scanf("%d", &count);
printf("Please input the elements that has to be sorted:\n");
for (counter = 0; counter < count; counter++)
{
scanf("%d", &element_list[counter]);
}
quicksort_method(element_list, 0, count - 1);
printf("Output generated after using quick sort\n");
for (counter = 0; counter < count; counter++)
{
printf("%d ", element_list[counter]);
}
printf("\n");
return 0;
}
void quicksort_method(int element_list[], int low, int high)
{
int pivot, value1, value2, temp;
if (low < high)
{
pivot = low;
value1 = low;
value2 = high;
while (value1 < value2)
{
while (element_list[value1] <= element_list[pivot] && value1 <= high)
{
value1++;
}
while (element_list[value2] > element_list[pivot] && value2 >= low)
{
value2--;
}
if (value1 < value2)
{

```

```

temp = element_list[value1];
element_list[value1] = element_list[value2];
element_list[value2] = temp;
}
}
temp = element_list[value2];
element_list[value2] = element_list[pivot];
element_list[pivot] = temp;
quicksort_method(element_list, low, value2 - 1);
quicksort_method(element_list, value2 + 1, high);
}
}

```

Đầu ra:

```

Please enter the total count of the elements that you want to sort: 6
Please input the elements that has to be sorted:
56
35
24
86
98
2
Output generated after using quick sort
2 24 35 56 86 98

```

8. Sắp xếp trộn (merge sort):

Chia đôi dãy thành 2 dãy con bằng hàm mergeSort(). Giả sử rằng mỗi dãy con đã được sắp xếp theo thứ tự, thì giờ chỉ cần trộn hai dãy con với nhau thành một dãy duy nhất bằng hàm merge(). Nếu dãy con chưa được sắp xếp thì lại áp dụng hàm mergeSort() lên dãy con đó, cứ chia đôi như vậy cho đến khi ko cần phải chia nữa.

VD:

```
#include<stdio.h>
```

```
void algo_merge_sort(int val[],int counter1,int counter2);
```

```

void perfrom_merge(int val[],int counter1,int counter2,int counter22,int
counter21);

int main()
{
int val[100],chk,counter1;

printf("Please enter the total count of the elements that you want to sort: \n");
scanf("%d",&chk);

printf("Please input the elements that has to be sorted:\n");
for(counter1=0;counter1<chk;counter1++)
scanf("%d",&val[counter1]);

algo_merge_sort(val,0,chk-1);

printf("\n Output generated after using quick sort \n");
for(counter1=0;counter1<chk;counter1++)
printf("%d ",val[counter1]);

return 0;
}

void algo_merge_sort(int val[],int counter1,int counter2)
{
int mid;

if(counter1<counter2)
{
mid=(counter1+counter2)/2;
algo_merge_sort(val,counter1,mid);
algo_merge_sort(val,mid+1,counter2);
}
}

```

```

perfrom_merge(val,counter1,mid,mid+1,counter2);
}
}

void perfrom_merge(int val[],int counter11,int counter12,int counter22,int
counter21)
{
int temp_val[50];
int c1,c2,c3;
c1=counter11;
c2=counter22;
c3=0;
while(c1<=counter12 && c2<=counter21)
{
if(val[c1]<val[c2])
temp_val[c3++]=val[c1++];
else
temp_val[c3++]=val[c2++];
}
while(c1<=counter12)
temp_val[c3++]=val[c1++];
while(c2<=counter21)
temp_val[c3++]=val[c2++];
for(c1=counter11,c2=0;c1<=counter21;c1++,c2++)
val[c1]=temp_val[c2];

```

```
}
```

Đầu ra:

```
Please enter the total count of the elements that you want to sort:
6
Please input the elements that has to be sorted:
32
36
65
98
45
8

Output generated after using quick sort
8 32 36 45 65 98
```

9. Sắp xếp vun đống (heap sort):

Sở dĩ gọi là vun đống vì mô hình thuật toán có dạng hình như quả núi.

Để dễ gọi hơn, mình gọi là cây phả hệ, mỗi một nốt là bố, mỗi bố có 2 nốt con (ở đây chọn mô hình phân hoạch nhị phân cho dễ), tương tự với các đời con sau.

Bước 1: Hiệu chỉnh thành dạng vun đống.

Bước 2: So sánh bố với con. Giả sử bài toán là sắp xếp mảng tăng dần, vậy ta hoán đổi vị trí sao cho vị trí lớn hơn lên làm bố. Bắt đầu từ thế hệ cháu chắt chút chút trước, sau đó đẩy dần lên thế hệ ông cụ kị, đến cuối cùng ta sẽ được vị trí to nhất lên chức cụ tổ. Khi cụ tổ đã ở đúng vị trí của nó trên bàn thờ rồi, ta chỉ việc lặp lại thuật toán với đám con cháu còn lại, cũng hiệu chỉnh thành đống rồi so sánh bố con. Chương trình kết thúc khi đám con cháu hoá ra chỉ còn 1 đứa.

VD:

```
void shift (int a[], int l, int r) {
```

```
    int x, i, j;
```

```
    i = l; j = 2 * i + 1;
```

```
    x = a[i];
```

```
    while (j <= r) {
```



```

    if( j < r && a[j] < a[j + 1]) j++;
    if (a[j] <= x) return;
    else {
        a[i] = a[j];
        a[j] = x;
        i = j; j = 2 * i + 1;
        x = a[i];
    }
}
}

```

```

void createHeap (int a[], int n) {
    int l = n / 2 - 1;
    while (l >= 0) {
        shift(a, l, n - 1);
        l--;
    }
}

```

```

void heapSort (int a[], int n) {
    createHeap(a, n);
    int r = n - 1;
    while (r > 0) {
        swap(a, 0, r);
        r--;
        if (r > 0) shift(a, 0, r);
    }
}

```

```
}  
}
```

IV. Các thuật toán tìm kiếm :

1. Tìm kiếm tuyến tính:

Là kiểm tra tuần tự từng phần tử của mảng, đến khi nào giống thì thôi.

VD:

```
int linearSearch (int a[], int n, int x) {  
    for (int i = 0; i < n; i++) {  
        if (a[i] == x) return i;  
        else return -1;  
    }  
}
```

2. Tìm kiếm nhị phân:

Điều kiện của thuật toán này là mảng đã được sắp xếp tăng dần. So sánh x với giá trị của phần tử nằm ở giữa mảng ($mid = (left + right) / 2$). Nếu x nhỏ hơn $a[mid]$ thì nó chỉ có thể nằm ở nửa bên trái, ngược lại x lớn hơn $a[mid]$ thì x nằm ở nửa bên phải. Xác định x nằm ở nửa nào thì ta lặp lại thuật toán với nửa đó. Như vậy số lần kiểm tra giảm đi nhiều do không phải mất công kiểm tra những phần tử thuộc nửa còn lại.

VD:

```
int binarySearch (int a[], int n, int x) {  
    int left = 0, right = n - 1, mid;  
    do {  
        mid = (left + right) / 2;  
        if (a[mid] == x) return mid;  
        else if (a[mid] < x) left = mid + 1;
```

```
    else right = mid - 1;  
} while (left <= right);  
return -1;}
```

PHẦN BA: TÀI LIỆU THAM KHẢO

<https://www.educba.com/sorting-in-c/>

<https://nguyenvanhieu.vn/>

<https://www.programiz.com/c-programming/c-file-input-output>