**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**MÔN: KỸ THUẬT LẬP TRÌNH**

**ĐỀ TÀI: BÀI TẬP LỚN**

Giảng viên: KS. TRẦN PHONG NHÃ

Sinh viên thực hiện: CAO TRUNG TÍNH

Lớp: CQ.65.CNTT

Khóa: 65

Tp. Hồ Chí Minh, tháng 11 năm 2024

Tp. Hồ Chi Minh, tháng 5 năm 2025

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**MÔN: KỸ THUẬT LẬP TRÌNH**

**ĐỀ TÀI: BÀI TẬP LỚN**

Giảng viên: KS. TRẦN PHONG NHÃ

Sinh viên thực hiện: CAO TRUNG TÍNH

Lớp: CQ.65.CNTT

Khóa: 65

Tp. Hồ Chi Minh, tháng 5 năm 2025

**LỜI CẢM ƠN**

Lời nói đầu tiên, em xin gửi tới Quý Thầy Cô Bộ môn Công nghệ Thông tin Trường Đại học Giao thông vận tải phân hiệu tại thành phố Hồ Chí Minh lời chúc sức khỏe và lòng biết ơn sâu sắc.

Em xin chân thành cảm ơn quý thầy cô đã giúp đỡ tạo điều kiện để em hoàn thành báo cáo với đề tài **“Bài tập lớn”**. Đặc biệt em xin cảm ơn thầy Trần Phong Nhã đã nhiệt tình giúp đỡ, hướng dẫn cho em kiến thức, định hướng và kỹ năng để có thể hoàn thành bài báo cáo này này.

Tuy đã cố gắng trong quá trình nghiên cứu tìm hiểu tuy nhiên do kiến thức còn hạn chế nên vẫn còn tồn tại nhiều thiếu sót. Vì vậy em rất mong nhận được sự đóng góp ý kiến của Quý thầy cô bộ môn để đề tài của em có thể hoàn thiện hơn.

Lời sau cùng, em xin gửi lời chúc tới Quý Thầy Cô Bộ môn Công nghệ thông tin và hơn hết là thầy Trần Phong Nhã có thật nhiều sức khỏe, có nhiều thành công trong công việc. Em xin chân thành cảm ơn!

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

TP. HCM , ngày … tháng… năm 2025

|  |
| --- |
| **Giáo viên hướng dẫn**  **Trần Phong Nhã** |

**MỤC LỤC**

[LỜI CẢM ƠN i](#_Toc15197)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN ii](#_Toc15673)

[MỤC LỤC iii](#_Toc10065)

[DANH MỤC CHỮ VIẾT TẮT v](#_Toc19634)

[DANH MỤC HÌNH ẢNH vi](#_Toc22915)

[A. LÝ THUYẾT 7](#_Toc1124)

[1.1 Hàm: 7](#_Toc30519)

[1.1.1 Định nghĩa: 7](#_Toc19978)

[1.1.2 Ví dụ minh họa: 8](#_Toc24221)

[1.2 Con trỏ: 8](#_Toc17647)

[1.2.1 Định nghĩa: 8](#_Toc1095)

[1.2.2 Ví dụ minh họa: 9](#_Toc9)

[1.3 Con trỏ mảng: 9](#_Toc27986)

[1.3.1 Định nghĩa: 9](#_Toc6649)

[- Kích thước mảng được cố định khi khai báo con trỏ. 9](#_Toc13298)

[1.4 Mảng con trỏ: 10](#_Toc19947)

[1.4.1 Định nghĩa: 10](#_Toc12607)

[1.4.2 Ví dụ minh họa: 10](#_Toc30902)

[1.5 Con trỏ hàm: 11](#_Toc32366)

[1.5.1 Định nghĩa: 11](#_Toc1954)

[1.5.2 Ví dụ minh họa: 11](#_Toc20705)

[1.6 Cấp phát động: 12](#_Toc27329)

[1.6.1 Định nghĩa 12](#_Toc4022)

[1.6.2 Ví dụ minh họa: 13](#_Toc3784)

[1.7 Xử lý tệp: 13](#_Toc14005)

[1.7.1 Định nghĩa: 13](#_Toc17076)

[1.7.2 Ví dụ minh họa: 14](#_Toc31186)

[1.8 Kiểu cấu trúc: 15](#_Toc20439)

[1.8.1 Định nghĩa: 15](#_Toc30109)

[1.8.2 Ví dụ minh họa: 17](#_Toc32503)

[1.9 Danh sách liên kết: 17](#_Toc23640)

[1.9.1 Định nghĩa: 17](#_Toc23452)

[1.9.2 Ví dụ minh họa: 22](#_Toc19590)

[B. ỨNG DỤNG 24](#_Toc20181)

[1.1 Giới thiệu chung: 24](#_Toc9523)

[1.2 Ứng dụng danh sách liên kết: 24](#_Toc13545)

[1.3 Ứng dụng file: 26](#_Toc3400)

[1.4 Ứng dụng con trỏ hàm: 26](#_Toc28117)

[1.5 Ứng dụng cấp phát động: 28](#_Toc20131)

[1.6 Giao diện: 28](#_Toc9867)

[TÀI LIỆU THAM KHẢO 30](#_Toc27341)

**DANH MỤC CHỮ VIẾT TẮT**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mô tả** | **Ý nghĩa** | **Ghi chú** |
|  |  |  |  |

**DANH MỤC HÌNH ẢNH**

[+ Code hàm: 8](#_Toc23060)

[- Code con trỏ: 9](#_Toc32263)

[- Code con trỏ mảng: 9](#_Toc10054)

[- Code mảng con trỏ: 10](#_Toc14836)

[- Code con trỏ hàm: 11](#_Toc11678)

[- Code cấp phát động: 13](#_Toc2737)

[- Code xử lý tệp: 15](#_Toc22964)

[- Code kiểu cấu trúc: 17](#_Toc20133)

[+ Danh sách liên kết đơn: 18](#_Toc19419)

[+ Danh sách liên kết đôi: 19](#_Toc31310)

[+ Danh sách liên kết vòng: 21](#_Toc3618)

[- Code về danh sách liên kết: 22](#_Toc201)

**A. LÝ THUYẾT**

## Hàm:

### Định nghĩa:

* + - * Hàm( function) là một các khối lệnh có nhiêm vụ thực hiện một chức năng nào đó.
      * Mỗi chương trình trong C có ít nhất một hàm là hàm main(). Tất cả hầu hết các chương trình bình thường đều định nghĩa thêm các hàm vì thế bạn có thể chia đoạn code của bạn thành các hàm riêng biệt.
      * Hàm được chia làm 2 loại:
        + Standard library functions
        + User-defined functions
      * Khái quát về từng loại:
        + Standard library functions( hàm thư viện chuẩn) là những hàm có sẵn trong lập trình C

Cấu trúc

#include <tên\_thư\_viện.h>

// Bắt buộc để dùng hàm chuẩn

kiểu\_trả\_về tên\_hàm(tham\_số);

* + - * + User-defined functions( hàm do người dùng định nghĩa) là hàm do lập trình viên tự tạo ra để thực hiện các công việc cụ thể

Cấu trúc:

kiểu\_trả\_về tên\_hàm(danh\_sách\_tham\_số) {

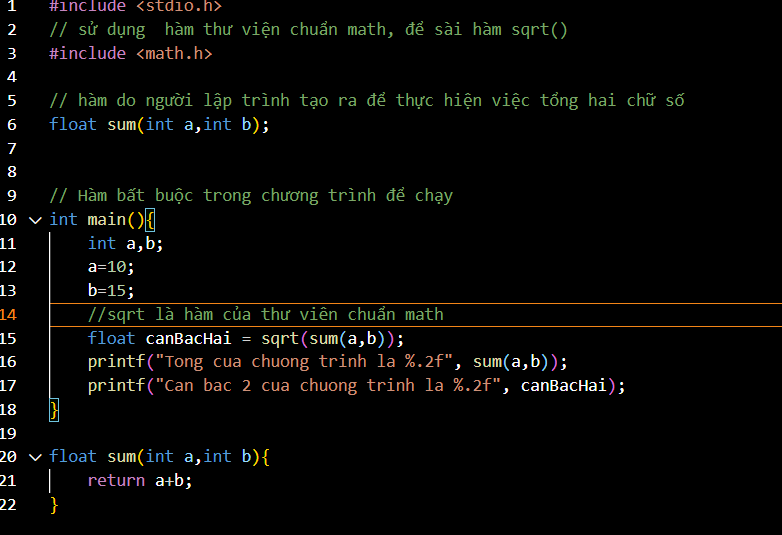
// phần thân hàm - chứa các lệnh thực hiện

return giá\_trị; // nếu có kiểu trả về khác void

}

## Ví dụ minh họa:

* + - * Ví dụ minh họa về hàm:
        + Code hàm:



* + - * + Kết quả



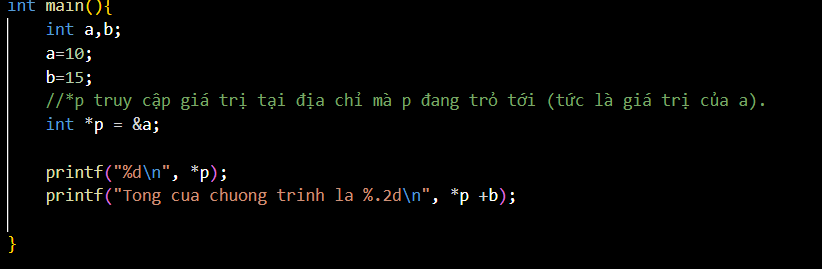
## Con trỏ:

### Định nghĩa:

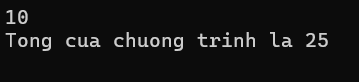
* + - * Con trỏ hay biến con trỏ( Pointer) là một biến dùng để lưu trữ địa chỉ bộ nhớ của một biến khác. Vì thế nên cho phép các chương trình có quyền thao tác trực tiếp vào bộ nhớ. Kí hiệu là (\*).
      * Cú pháp khai báo: Kiểu\_trả\_về \*tên\_biến;
        + Ví dụ như: int \*p;
      * Cú pháp lấy địa chỉ của một biến: &(tên\_biến);
        + Ví dụ như &a;

### Ví dụ minh họa:

* + - * Code con trỏ:



* + - * Kết quả:



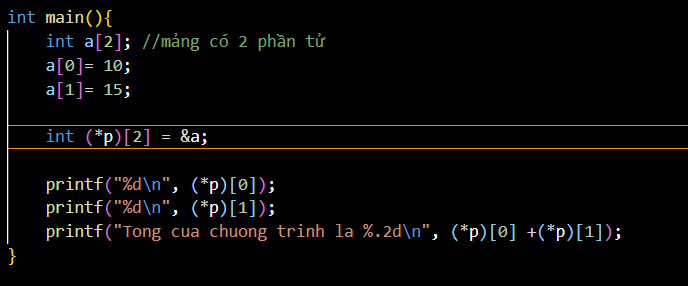
## Con trỏ mảng:

### Định nghĩa:

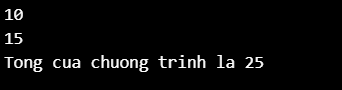
* + - * Con trỏ mảng là một con trỏ trỏ tới toàn bộ một mảng 1 chiều có kích thước cố định. Điều này khác với con trỏ phần tử( int \*p) vốn chỉ trỏ tới phần tử đầu tiên.

### Kích thước mảng được cố định khi khai báo con trỏ.

* + - * Dùng để trỏ tới mảng 1 chiều, truy cập các phần tử mảng.
      * Cú pháp khai báo: kiểu\_dữ\_liệu (\*con\_trỏ)[kích\_thước\_mảng];
    1. Ví dụ minh họa:
       - Code con trỏ mảng:



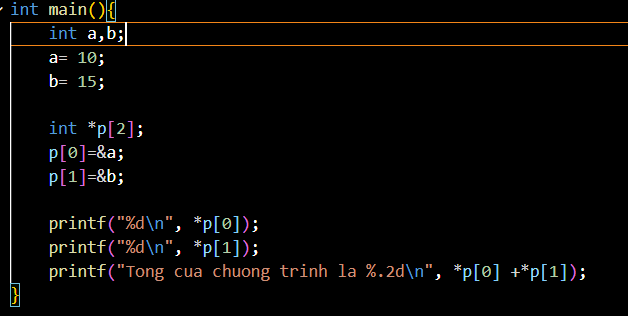
* + - * Kết quả:



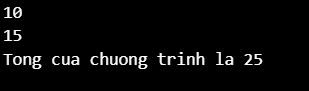
## Mảng con trỏ:

### Định nghĩa:

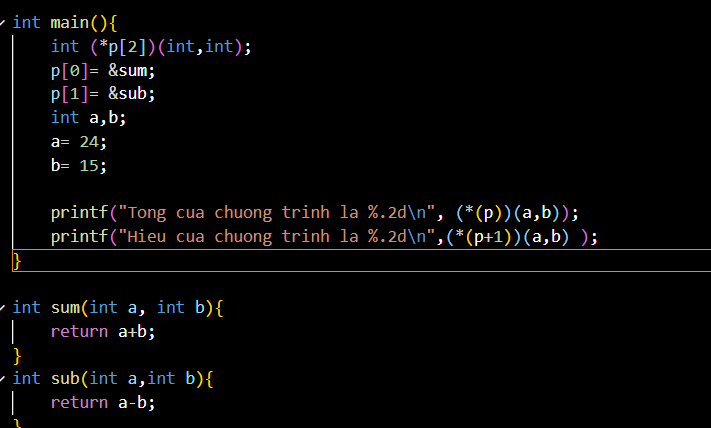
* + - * Mảng con trỏ là một mảng chứa các phần tử là con trỏ.
      * Mỗi phần tử trong mảng này là một con trỏ trỏ tới một địa chỉ( thường là địa chỉ biến hoặc vùng nhớ).
      * Mảng con trỏ dùng để quản lý nhiều biến cùng kiểu hay quản lý cấp phát động nhiều vùng nhớ.
      * Cú pháp khai báo: kiểu\_dữ\_liệu \*tên\_mảng[số\_phần\_tử];
      * Lưu ý: khác với con trỏ mảng thì cú pháp khai báo ở đây có sự khác biết là
        + Con trỏ mảng: int (\*p)[2];
        + Mảng con trỏ: int \*p[2];
    1. Ví dụ minh họa:
       - Code mảng con trỏ:



* + - * Kết quả:



* 1. Con trỏ hàm:
     1. Định nghĩa:
        + Con trỏ hàm là một con trỏ dùng để lưu địa chỉ của một hàm. Thông qua con trỏ này, bạn có thể gọi hàm giống như gọi trực tiếp.Giúp tăng tính linh hoạt.
        + Dùng trong xử lý sự kiện, lặp mảng, hoặc thiết kế plugin, callback, hook...
        + Cú pháp khai báo:kiểu\_trả\_về (\*tên\_con\_trỏ)( danh\_sách\_tham\_số);
          - Danh sách tham số là kiểu và số lượng tham số của hàm mà con trỏ sẽ trỏ tới.
          - Giá trị trả về của con trỏ hàm phải trùng với kiểu giá trị trả về của hàm mà nó trỏ tới.
     2. Ví dụ minh họa:
        + Code con trỏ hàm:



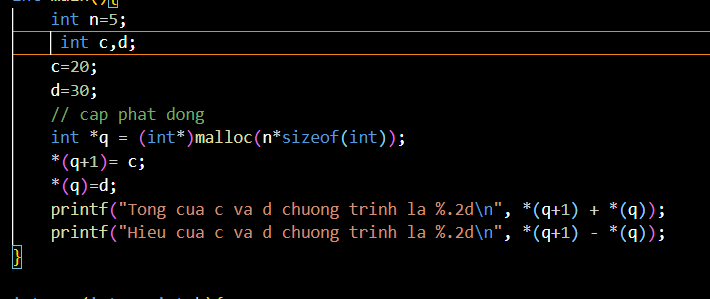
* + - * Kết quả:



* 1. Cấp phát động:
     1. Định nghĩa
        + Cấp phát động là cấp phát bộ nhớ khi chương trình đang chạy ( runtime) thay vì khi biên dịch.
        + Dùng khi không biết trước kích thước mảng, chuỗi, cấu trúc...Khi muốn tiết kiệm bộ nhớ bằng cách chỉ cấp phát khi cần và giải phóng sau khi dùng xong.
        + Cú pháp khai báo
          - Các hàm cấp phát động:

|  |  |  |
| --- | --- | --- |
| Hàm | Mục đích | Cú pháp |
| malloc | Cấp phát bộ nhớ chưa khởi tạo | <kiểu\_dữ\_liệu>\* con\_tro = (kiểu\_dữ\_liệu\*) malloc(số\_bytes); |
| calloc | Cấp phát bộ nhớ và khởi tạo về 0 | <kiểu\_dữ\_liệu>\* con\_tro = (kiểu\_dữ\_liệu\*) calloc(số\_phần\_tử, kích\_thước\_mỗi\_phần\_tử); |
| realloc | Thay đổi kích thước vùng nhớ đã cấp phát | con\_tro = (kiểu\_dữ\_liệu\*) realloc(con\_tro\_cũ, số\_bytes\_mới); |
| free | Giải phóng vùng nhớ đã cấp phát | free(con\_tro); |

* + - * + Các hàm cấp phát động thuộc thư viện stdlib.h.
    1. Ví dụ minh họa:
       - Code cấp phát động:



* + - * Kết quả:



* 1. Xử lý tệp:
     1. Định nghĩa:
        + Xử lý tệp (file handling) trong C là thao tác đọc, ghi, mở, đóng file trên đĩa. Đây là cách bạn làm việc với dữ liệu lưu ngoài bộ nhớ, rất quan trọng để lưu trữ lâu dài.
        + Cú pháp khai báo:
          - Các bước cơ bản:

FILE \*con\_tro;

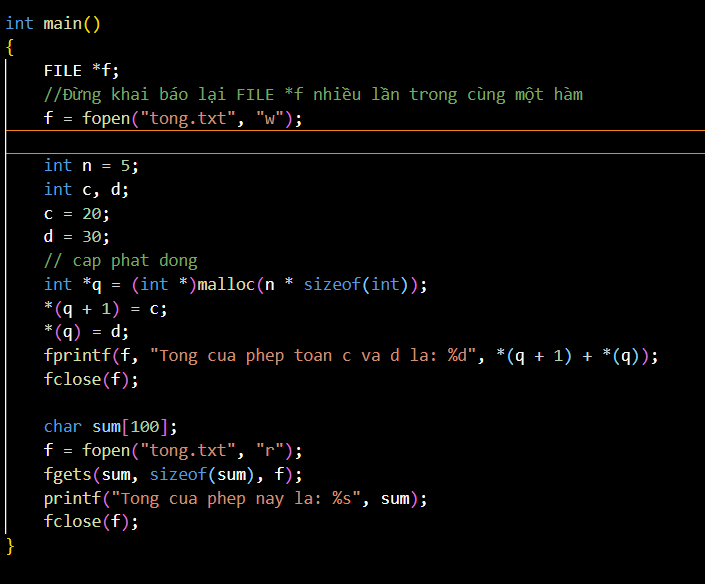
Mở tệp (mở file) với hàm fopen().

Đọc hoặc ghi dữ liệu với fscanf(), fprintf(), fread(), fwrite() hoặc các hàm tương tự.

Đóng tệp với fclose().

* + - * + Các chế độ mở file thường dùng:

|  |  |
| --- | --- |
| Chế độ | Công dụng |
| "a" | Mở file để ghi thêm (append). Dữ liệu mới được ghi vào cuối file. |
| "r+" | Mở file để đọc và ghi. File phải tồn tại. |
| "w+" | Mở file để đọc và ghi. Nếu file tồn tại sẽ xóa dữ liệu cũ, tạo mới nếu không có. |
| "a+" | Mở file để đọc và ghi thêm. Con trỏ file được đặt ở cuối file khi mở. |
| "rb" | Mở file để đọc nhị phân. File phải tồn tại. |
| "wb" | Mở file để ghi nhị phân. Xóa dữ liệu cũ hoặc tạo mới. |
| "ab" | Mở file để ghi thêm nhị phân. Thêm dữ liệu vào cuối file. |
| "rb+" | Mở file để đọc/ghi nhị phân. File phải tồn tại. |
| "wb+" | Mở file để đọc/ghi nhị phân. Xóa dữ liệu cũ hoặc tạo mới. |
| "ab+" | Mở file để đọc/ghi thêm nhị phân. Thêm dữ liệu vào cuối file. |

* + 1. Ví dụ minh họa:
       - Code xử lý tệp: 
       - Kết quả:







* 1. Kiểu cấu trúc:
     1. Định nghĩa:
        + Kiểu cấu trúc( Struct) là một kiểu dữ liệu do người dùng định nghĩa, cho phép nhóm các biến có thể thuộc nhiều kiểu dữ liệu khác nhau thành một đơn vị duy nhất.
        + Struct giúp tổ chức dữ liệu phức tạp dễ quản lý hơn
        + Xây dựng, mô hình hóa, các cấu trúc dữ liệu phức tạp hay theo nhóm.
        + Cú pháp khai báo:
          - Cấu pháp khai báo có tên:

struct TenCauTruc {

kieu\_du\_lieu ten\_thanh\_vien1;

kieu\_du\_lieu ten\_thanh\_vien2;

...

};

Khi sử dụng:

struct TenCauTruc bien1;

* + - * + Cấu pháp khai alias( sử dụng từ khóa typedef):

struct TenStruct {

// Các thành viên

kieu\_du\_lieu1 ten\_thanh\_vien1;

kieu\_du\_lieu2 ten\_thanh\_vien2;

};

typedef struct TenStruct AliasTen;

Hoặc

typedef struct {

kieu\_du\_lieu1 ten\_thanh\_vien1;

kieu\_du\_lieu2 ten\_thanh\_vien2;

} AliasTen;

Khi sử dụng:

Alias bien1;

* + - * + Truy cập vào gia trị:

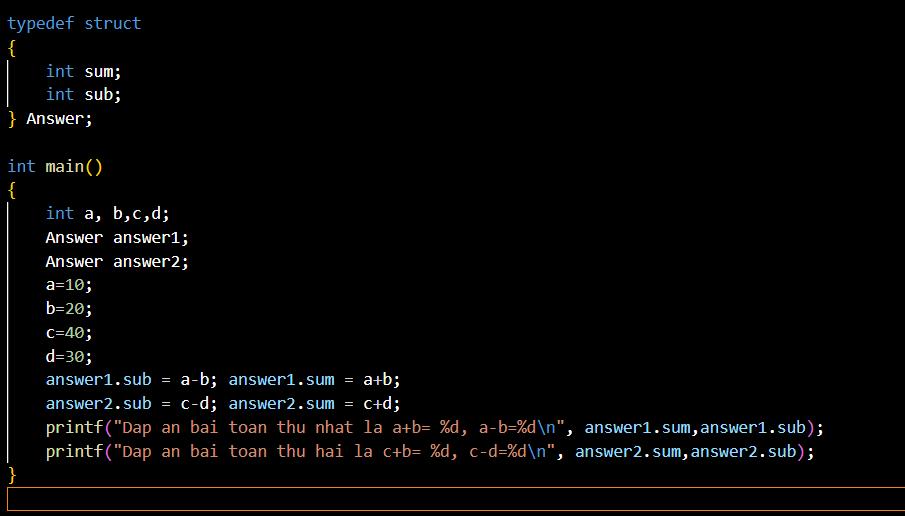
Trực tiếp:

bien1.ten\_thanh\_vien1;

Con trỏ:

bien1->ten\_thanh\_vien1;

* + 1. Ví dụ minh họa:
       - Code kiểu cấu trúc:



* + - * Kết quả:

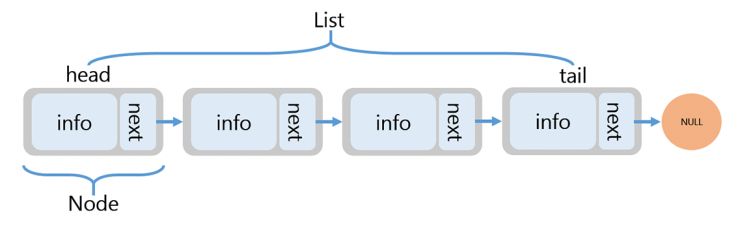


* 1. Danh sách liên kết:
     1. Định nghĩa:
        + Danh sách liên kết là một cấu trúc dữ liệu tuyến tính, gồm nhiều node (nút) liên kết với nhau bằng con trỏ.
        + Danh sách liên kết giúp quản lý và thao tác dữ liệu động hiệu quả, dễ dàng thêm hoặc xóa phần tử bất kỳ mà không cần cấp phát lại toàn bộ bộ nhớ.
        + Phân loại:

|  |  |
| --- | --- |
| Loại danh sách | Đặc điểm chính |
| Danh sách đơn (Singly Linked List) | Mỗi node trỏ đến node kế tiếp. Chỉ đi một chiều. |
| Danh sách đôi (Doubly Linked List) | Mỗi node có 2 con trỏ: trỏ trước và trỏ sau. Đi hai chiều. |
| Danh sách vòng (Circular Linked List) | Node cuối trỏ lại node đầu. Có thể là đơn hoặc đôi. |

* + - * Cú pháp khai báo:
        + Danh sách liên kết đơn:

Nguyên lý hoạt động:



Định nghĩa Node:

struct Node {

int data;

struct Node\* next;

};

Khai báo con trỏ head và tail:

struct Node\* head = NULL;

struct Node\* tail = NULL;

//có thể bỏ tail, vì thêm vào để tiện quản lý danh sách hơn

Cấp phát bộ nhớ cho Node mới:

struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));

newNode->data = <data>;

newNode->next = NULL;

Thêm vào danh sách ( cuối):

if (head == NULL) {

head = newNode;

tail = newNode;

} else {

tail->next = newNode;

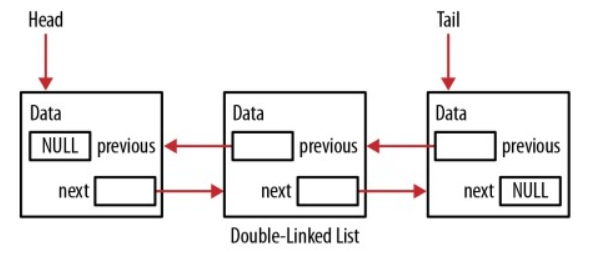
tail = newNode;

}

-> Dựa trên nguyên lý cơ bản: head là con trỏ vào đầu danh sách, tail là con trỏ vào cuối danh sách.

* + - * + Danh sách liên kết đôi:

Nguyên lý hoạt động:

Định nghĩa Node:

struct Node {

int data;

struct Node\* prev;

struct Node\* next;

};

Khai báo con trỏ head và tail:

struct Node\* head = NULL;

struct Node\* tail = NULL;

//có thể bỏ tail, vì thêm vào để tiện quản lý danh sách hơn

Cấp phát bộ nhớ cho Node mới:

struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));

newNode->data = <data>;

newNode->prev = NULL;

newNode->next = NULL;

Thêm vào danh sách ( cuối):

if (head == NULL) {

head = newNode;

tail = newNode;

} else {

tail->next = newNode;

newNode->prev = tail;

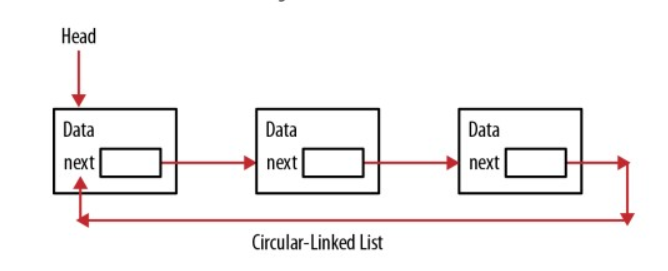
tail = newNode;

}

-> Dựa trên nguyên lý cơ bản: head là con trỏ vào đầu danh sách, tail là con trỏ vào cuối danh sách.

* + - * + Danh sách liên kết vòng:

Nguyên lý hoạt động:



Định nghĩa Node:

struct Node {

int data;

struct Node\* next;

};

Khai báo con trỏ head và tail:

struct Node\* head = NULL;

Cấp phát bộ nhớ cho Node mới:

struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));

newNode->data = <data>;

newNode->next = NULL;

Thêm vào danh sách(cuối):

if (head == NULL) {

head = newNode;

} else {

struct Node\* temp = head;

while (temp->next != head) {

temp = temp->next;

}

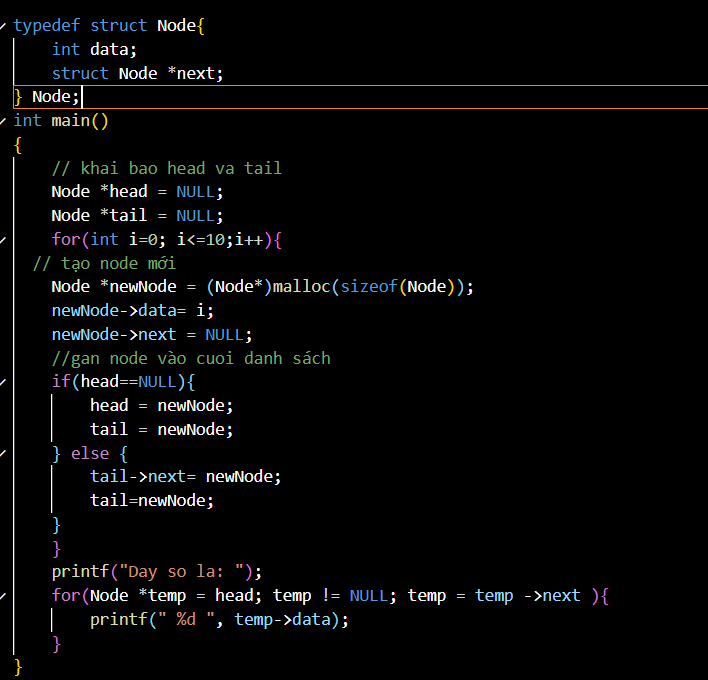
temp->next = newNode;

newNode->next = head;

}

-> Dựa trên nguyên lý cơ bản: head là con trỏ vào đầu danh sách và node cuối sẽ nói vào node đầu nên tạo thành danh sách liên kết vòng, thực tế thêm cuối và đầu giống nhau, khác ở chỗ thay đổi vị trí head thôi.

* + 1. Ví dụ minh họa:
       - Code về danh sách liên kết:



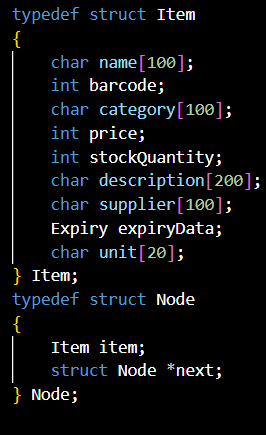
* + - * Kết quả:

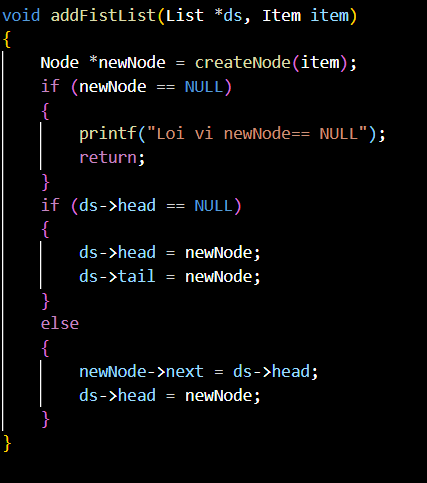


1. **ỨNG DỤNG**

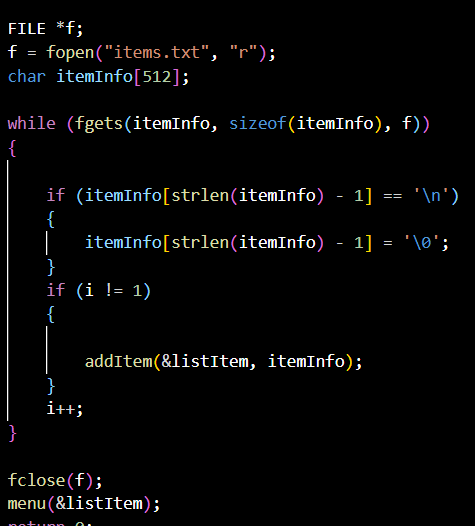
## Giới thiệu chung:

* + - Ứng dụng dựa trên máy bán hàng trong của hàng tiện lợi.
    - Các chức năng chính:
      * Chúc năng bán hàng:
        + Nhập mã thanh toán.
        + Tìm kiếm sản phẩm.
        + Lịch sử bill.
      * Chức năng quản lý của hàng:
        + Nhâp sản phẩm vào kho.
        + Tìm kiếm sản phẩm.
        + Sắp xếp sản phẩm( tên, giá,.. ).
        + Kho chứa.
  1. Ứng dụng danh sách liên kết:
     + Dùng để lưu danh sách các sản phẩm nhập vào.
     + Code:

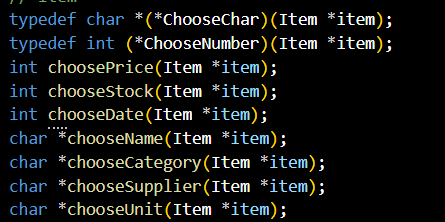


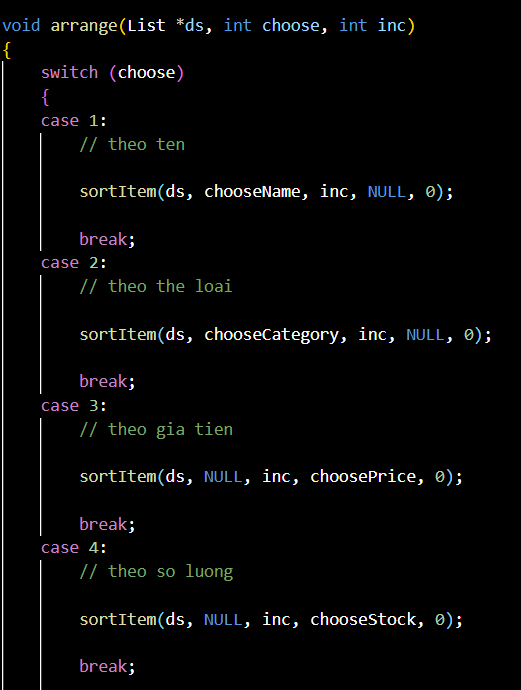


* 1. Ứng dụng file:
     + Dùng để lưu trữ hàng hóa, nhập hàng và mặt khẩu nhân viên.
     + Code:

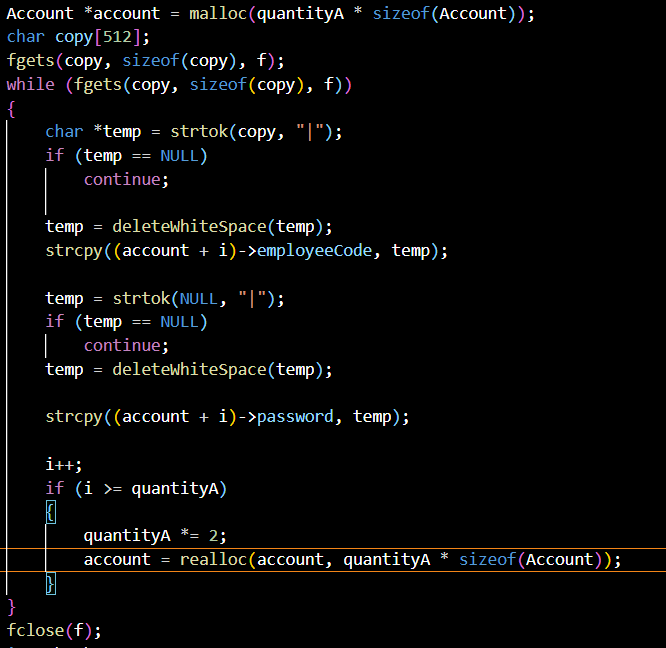


* 1. Ứng dụng con trỏ hàm:
     + Dùng để lựa chọn cho các mục sắp xếp, rút ngắn đoạn code
     + Code:

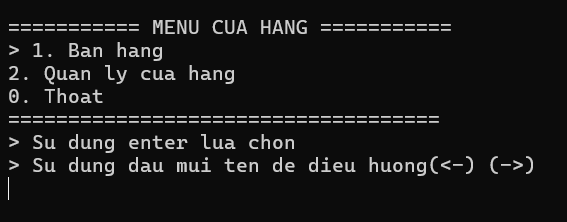


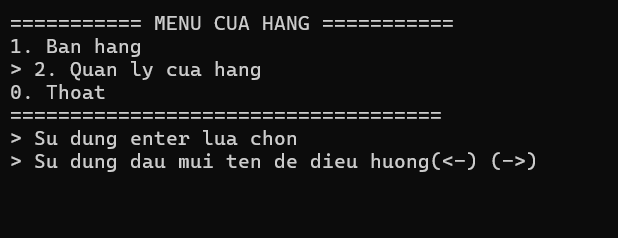


* 1. Ứng dụng cấp phát động:
     + Cấp phát động bộ nhớ cho danh sách tài khoản từ file, nếu tài khoản trong file vượt quá giới hạn thì sẽ cấp thêm vùng nhớ.
     + Code:



* 1. Giao diện:
     + Tuy giao diện không được đẹp nhưng có bổ sung thêm tính năng menu có điều hướng bằng phím mũi tên trái/phải và Enter trong giao diện console:





# TÀI LIỆU THAM KHẢO

[1]. https://www.geeksforgeeks.org/getch-function-in-c-with-examples/ ,“Thiết kế giao diện chuyển hướng bằng phím mũi tên” .

[Truy cập 15 5 2025].

[2].https://howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban/con-tro-ham-trong-c-function-pointers-3916, “Tham khảo con trỏ hàm”.

[Truy cập 17 5 2025].