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

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

DANH MỤC CHỮ VIẾT TẮT

STT	Mô tả	Ý nghĩa	Ghi chú

DANH MỤC HÌNH ẢNH

A. LÝ THUYẾT

1.1 Hàm:

1.1.1 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
```

}

1.1.2 Ví dụ minh họa:

- Ví dụ minh họa về hàm:

  + Code hàm:

```c
#include <stdio.h>
// sử dụng  hàm thư viện chuẩn math, để sài hàm sqrt()
#include <math.h>

// hàm do người lập trình tạo ra để thực hiện việc tổng hai chữ số
float sum(int a,int b);


// Hàm bắt buộc trong chương trình để chạy
int main(){
    int a,b;
    a=10;
    b=15;
    //sqrt là hàm của thư viên chuẩn math
    float canBacHai = sqrt(sum(a,b));
    printf("Tong cua chuong trinh la %.2f", sum(a,b));
    printf("Can bac 2 cua chuong trinh la %.2f", canBacHai);
}

float sum(int a,int b){
    return a+b;
}
```

  + Kết quả

```
Tong cua chuong trinh la 25.00
Can bac 2 cua chuong trinh la 5.00
```

1.2 Con trỏ:

1.2.1 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;

1.2.2 Ví dụ minh họa:

- Code con trỏ:

```c
int main(){
    int a,b;
    a=10;
    b=15;
    //*p truy cập giá trị tại địa chỉ mà p đang trỏ tới (tức là giá trị của a).
    int *p = &a;

    printf("%d\n", *p);
    printf("Tong cua chuong trinh la %.2d\n", *p +b);

}
```

- Kết quả:

```
10
Tong cua chuong trinh la 25
```

1.3 Con trỏ mảng:

1.3.1 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];

1.3.2 Ví dụ minh họa:

- Code con trỏ mảng:

```c
int main(){
    int a[2]; //mảng có 2 phần tử
    a[0]= 10;
    a[1]= 15;

    int (*p)[2] = &a;

    printf("%d\n", (*p)[0]);
    printf("%d\n", (*p)[1]);
    printf("Tong cua chuong trinh la %.2d\n", (*p)[0] +(*p)[1]);
}
```

- Kết quả:

```
10
15
Tong cua chuong trinh la 25
```

1.4 Mảng con trỏ:

1.4.1 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];

1.4.2 Ví dụ minh họa:

- Code mảng con trỏ:

```c
int main(){
    int a,b;
    a= 10;
    b= 15;

    int *p[2];
    p[0]=&a;
    p[1]=&b;

    printf("%d\n", *p[0]);
    printf("%d\n", *p[1]);
    printf("Tong cua chuong trinh la %.2d\n", *p[0] +*p[1]);
}
```

- Kết quả:

```
10
15
Tong cua chuong trinh la 25
```

1.5 Con trỏ hàm:

1.5.1 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.

1.5.2 Ví dụ minh họa:

- Code con trỏ hàm:

```c
int main(){
    int (*p[2])(int,int);
    p[0]= &sum;
    p[1]= &sub;
    int a,b;
    a= 24;
    b= 15;

    printf("Tong cua chuong trinh la %.2d\n", (*(p))(a,b));
    printf("Hieu cua chuong trinh la %.2d\n",(*(p+1))(a,b) );
}

int sum(int a, int b){
    return a+b;
}
int sub(int a,int b){
    return a-b;
}
```

- Kết quả:

```
Tong cua chuong trinh la 39
Hieu cua chuong trinh la 09
```

1.6 Cấp phát động:

1.6.1 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.

1.6.2 Ví dụ minh họa:

- Code cấp phát động:

```
    int n=5;
    int c,d;
    c=20;
    d=30;
    // cap phat dong
    int *q = (int*)malloc(n*sizeof(int));
    *(q+1)= c;
    *(q)=d;
    printf("Tong cua c va d chuong trinh la %.2d\n", *(q+1) + *(q));
    printf("Hieu cua c va d chuong trinh la %.2d\n", *(q+1) - *(q));
}
```

- Kết quả:

```
Tong cua c va d chuong trinh la 50
Hieu cua c va d chuong trinh la -10
```

1.7 Xử lý tệp:

1.7.1 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:

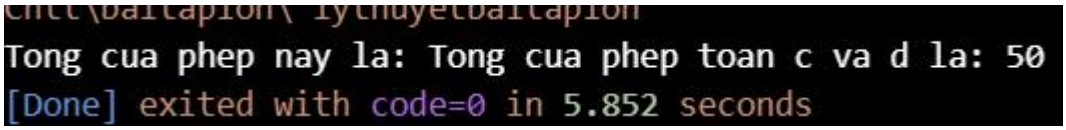
Chế độ	Công dụng
"a"	Mở file để ghi thêm (append). Dữ liệu mới được ghi vào cuối file.
"r+"	Mở file để đọc và ghi. File phải tồn tại.
"w+"	Mở file để đọc và ghi. Nếu file tồn tại sẽ xóa dữ liệu cũ, tạo mới nếu không có.
"a+"	Mở file để đọc và ghi thêm. Con trỏ file được đặt ở cuối file khi mở.
"rb"	Mở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

1.7.2 Ví dụ minh họa:

- Code xử lý tệp:

```c
int main()
{
    FILE *f;
    //Đừng khai báo lại FILE *f nhiều lần trong cùng một hàm
    f = fopen("tong.txt", "w");

    int n = 5;
    int c, d;
    c = 20;
    d = 30;
    // cap phat dong
    int *q = (int *)malloc(n * sizeof(int));
    *(q + 1) = c;
    *(q) = d;
    fprintf(f, "Tong cua phep toan c va d la: %d", *(q + 1) + *(q));
    fclose(f);

    char sum[100];
    f = fopen("tong.txt", "r");
    fgets(sum, sizeof(sum), f);
    printf("Tong cua phep nay la: %s", sum);
    fclose(f);
}
```

- Kết quả:



1.8 Kiểu cấu trúc:

1.8.1 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;

1.8.2 Ví dụ minh họa:

- Code kiểu cấu trúc:

```c
typedef struct
{
    int sum;
    int sub;
} Answer;

int main()
{
    int a, b,c,d;
    Answer answer1;
    Answer answer2;
    a=10;
    b=20;
    c=40;
    d=30;
    answer1.sub = a-b; answer1.sum = a+b;
    answer2.sub = c-d; answer2.sum = c+d;
    printf("Dap an bai toan thu nhat la a+b= %d, a-b=%d\n", answer1.sum,answer1.sub);
    printf("Dap an bai toan thu hai la c+b= %d, c-d=%d\n", answer2.sum,answer2.sub);
}
```

- Kết quả:

```
Dap an bai toan thu nhat la a+b= 30, a-b=-10
Dap an bai toan thu hai la c+b= 70, c-d=10
```

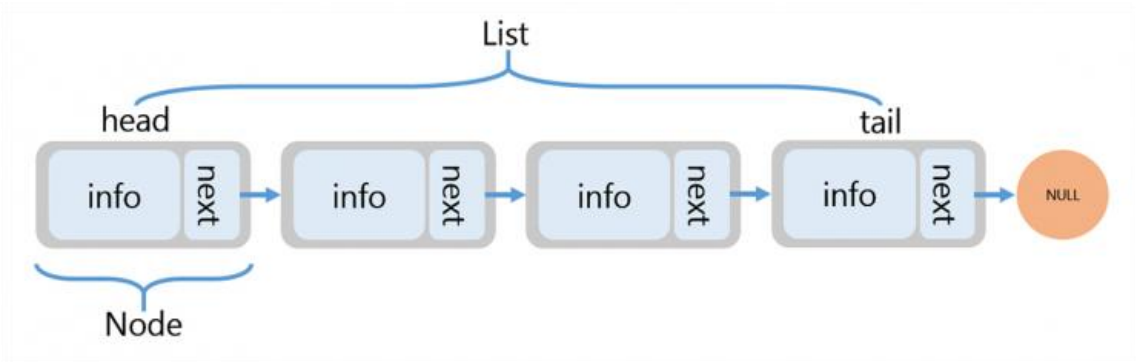
1.9 Danh sách liên kết:

1.9.1 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:



ghĩa 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;
```

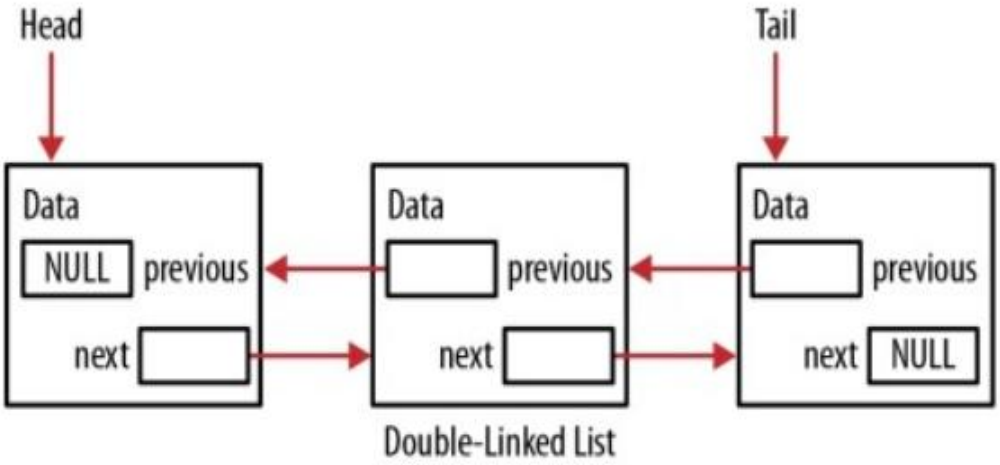
- Thêm vào danh sách (cuối):

```
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:



a 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;
```

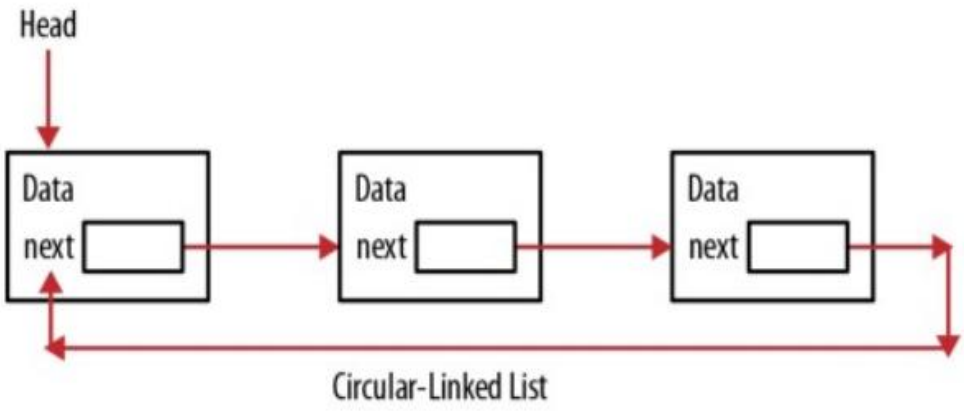
- Thêm vào danh sách (cuối):

```
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:



hĩa 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.

1.9.2 Ví dụ minh họa:

- Code về danh sách liên kết:

```c
typedef struct Node{
    int data;
    struct Node *next;
} Node;
int main()
{
    // khai bao head va tail
    Node *head = NULL;
    Node *tail = NULL;
    for(int i=0; i<=10;i++){
   // tạo node mới
    Node *newNode = (Node*)malloc(sizeof(Node));
    newNode->data= i;
    newNode->next = NULL;
    //gan node vào cuoi danh sách
    if(head==NULL){
        head = newNode;
        tail = newNode;
    } else {
        tail->next= newNode;
        tail=newNode;
    }
    }
    printf("Day so la: ");
    for(Node *temp = head; temp != NULL; temp = temp ->next ){
        printf(" %d ", temp->data);
    }
}
```

- Kết quả:

```
Day so la:  0  1  2  3  4  5  6  7  8  9  10
```

B. ỨNG DỤNG

1.1 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:
        - Nhập sản phẩm vào kho.
        - Tìm kiếm sản phẩm.
        - Sắp xếp sản phẩm(tên, giá,..).
        - Kho chứa.

1.2 Ứng dụng danh sách liên kết:

- Dùng để lưu danh sách các sản phẩm nhập vào.
- Code:

```c
typedef struct Item
{
    char name[100];
    int barcode;
    char category[100];
    int price;
    int stockQuantity;
    char description[200];
    char supplier[100];
    Expiry expiryData;
    char unit[20];
} Item;
typedef struct Node
{
    Item item;
    struct Node *next;
} Node;
```

```c
void addFistList(List *ds, Item item)
{
    Node *newNode = createNode(item);
    if (newNode == NULL)
    {
        printf("Loi vi newNode== NULL");
        return;
    }
    if (ds->head == NULL)
    {
        ds->head = newNode;
        ds->tail = newNode;
    }
    else
    {
        newNode->next = ds->head;
        ds->head = newNode;
    }
}
```

1.3 Ứng dụng file:

- Dùng để lưu trữ hàng hóa, nhập hàng và mặt khẩu nhân viên.
- Code:

```
FILE *f;
f = fopen("items.txt", "r");
char itemInfo[512];

while (fgets(itemInfo, sizeof(itemInfo), f))
{

    if (itemInfo[strlen(itemInfo) - 1] == '\n')
    {
        itemInfo[strlen(itemInfo) - 1] = '\0';
    }
    if (i != 1)
    {

        addItem(&listItem, itemInfo);
    }
    i++;
}

fclose(f);
menu(&listItem);
```

1.4 Ứng dụng con trỏ hàm:

- Dùng để lựa chọn cho các mục sắp xếp, rút ngắn đoạn code
- Code:

```
typedef char *(*ChooseChar)(Item *item);
typedef int (*ChooseNumber)(Item *item);
int choosePrice(Item *item);
int chooseStock(Item *item);
int chooseDate(Item *item);
char *chooseName(Item *item);
char *chooseCategory(Item *item);
char *chooseSupplier(Item *item);
char *chooseUnit(Item *item);
```

```
void arrange(List *ds, int choose, int inc)
{
    switch (choose)
    {
    case 1:
        // theo ten

        sortItem(ds, chooseName, inc, NULL, 0);

        break;
    case 2:
        // theo the loai

        sortItem(ds, chooseCategory, inc, NULL, 0);

        break;
    case 3:
        // theo gia tien

        sortItem(ds, NULL, inc, choosePrice, 0);

        break;
    case 4:
        // theo so luong

        sortItem(ds, NULL, inc, chooseStock, 0);

        break;
```

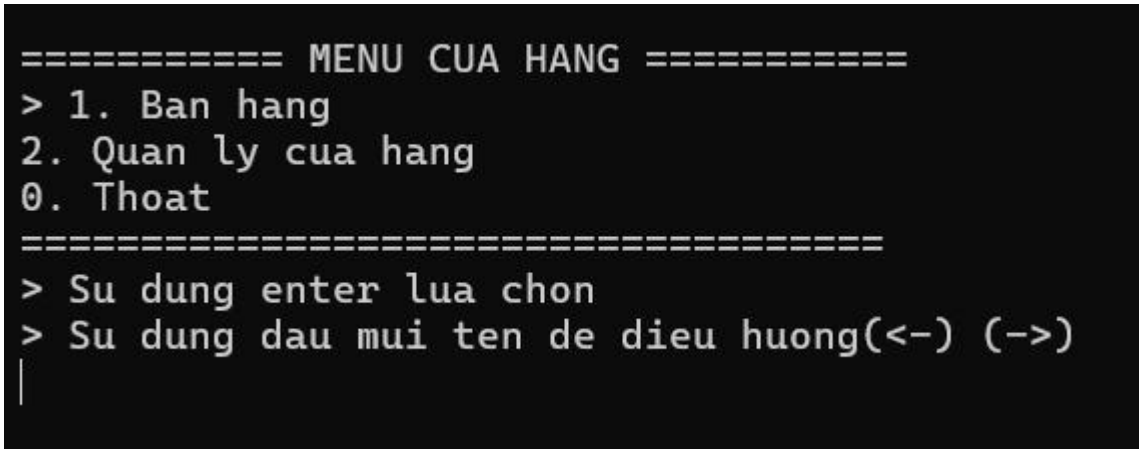
1.5 Ứng dụng cấp phát động:

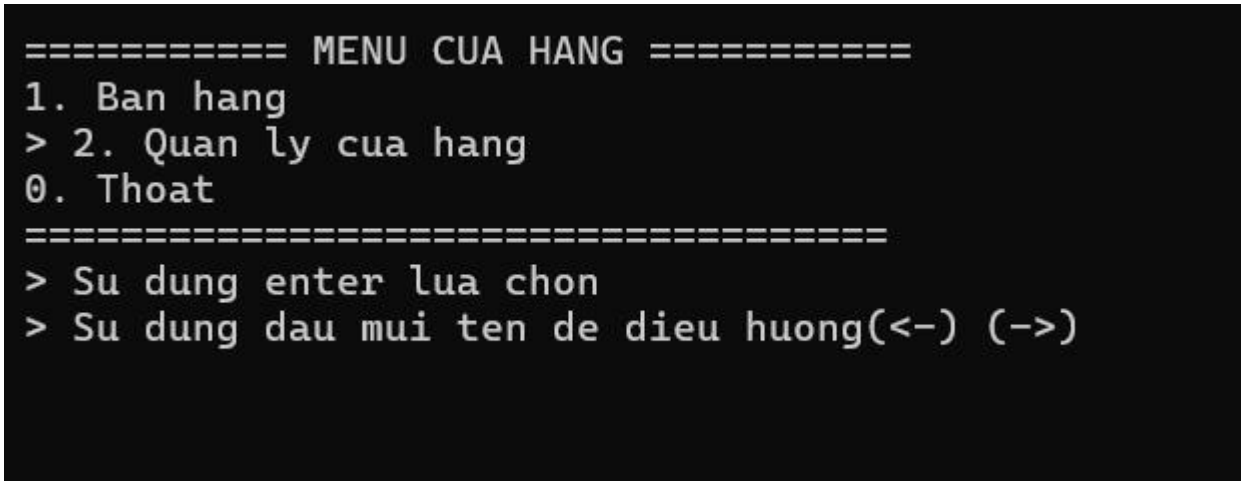
- Cấp phát động bộ nhớ cho danh sách tài khoản từ file, nếu tài khoản trong file vượt quá giới hạn thì sẽ cấp thêm vùng nhớ.
- Code:

```
Account *account = malloc(quantityA * sizeof(Account));
char copy[512];
fgets(copy, sizeof(copy), f);
while (fgets(copy, sizeof(copy), f))
{
    char *temp = strtok(copy, "|");
    if (temp == NULL)
        continue;

    temp = deleteWhiteSpace(temp);
    strcpy((account + i)->employeeCode, temp);

    temp = strtok(NULL, "|");
    if (temp == NULL)
        continue;
    temp = deleteWhiteSpace(temp);

    strcpy((account + i)->password, temp);

    i++;
    if (i >= quantityA)
    {
        quantityA *= 2;
        account = realloc(account, quantityA * sizeof(Account));
    }
}
fclose(f);
```

1.6 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:


========== MENU CUA HANG ==========
> 1. Ban hang
2. Quan ly cua hang
0. Thoat
====================================
> Su dung enter lua chon
> Su dung dau mui ten de dieu huong(<-) (->)


========== MENU CUA HANG ==========
1. Ban hang
> 2. Quan ly cua hang
0. Thoat
====================================
> Su dung enter lua chon
> Su dung dau mui ten de dieu huong(<-) (->)

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].