TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO

MÔN: KĨ THUẬT LẬP TRÌNH

ĐỀ TÀI: BÀI TẬP LỚN

Giảng viên: ThS. TRẦN PHONG NHÃ

Sinh viên thực hiện: LÊ HOÀNG NHÂN

ĐỖ MINH TÍN

Lớp: CQ.65.CNTT

Khoá: 65

Tp. Hồ Chí 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: KĨ THUẬT LẬP TRÌNH

ĐỀ TÀI: BÀI TẬP LỚN

Giảng viên: ThS. TRẦN PHONG NHÃ

Sinh viên thực hiện: LÊ HOÀNG NHÂN

ĐỖ MINH TÍN

Lớp: CQ.65.CNTT

Khoá: 65

Tp. Hồ Chí Minh, tháng 5 năm 2025

LỜI CẢM ƠN

Lời nói đầu tiên, chúng 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.

Chúng em xin chân thành cảm ơn quý thầy cô đã giúp đỡ và tạo điều kiện để em hoàn thành bài báo cáo với đề tài "**Bài tập lớn Kỹ thuật lập trình**". Đặc biệt chúng em xin cảm ơn thầy Trần Phong Nhã đã nhiệt tình giúp đỡ, hướng dẫn cho chúng em kiến thức, định hướng và kỹ năng để có thể hoàn thành bài báo cáo 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 chúng em rất mong nhận được sự đóng góp ý kiến của Quý thầy cô bộ môn để đề tài của chúng em có thể hoàn thiên hơn.

Lời sau cùng, chúng 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 và nhiều thành công trong công việc. Chúng em xin chân thành cảm ơn!

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp. Hồ Chí Minh, ngày … tháng … năm …

Giáo viên hướng dẫn

ThS. Trần Phong Nhã

MỤC LỤC

Link Github: https://github.com/Aurora1701/baocaoktlt

CHƯƠNG 1. LÍ THUYẾT

1.1) Hàm (Function)

- Hàm là một khối mã có chức năng thực hiện một nhiệm vụ cụ thể có những ưu điểm như: tính tái sử dụng nhiều lần, dễ sử dụng và sửa lỗi và làm cho chương trình có tính tổ chức cao hơn.
- Cấu trúc của một hàm thường có dạng như sau:

```
kiểu_trả_về tên_hàm(danh_sách_tham_số)
{
  các câu lệnh  // phần thân hàm:
    return giá_trị; // nếu kiểu trả về khác void
}
```

- Nguyên tắc sử dụng hàm là hàm phải được khai báo hoặc định nghĩa trước khi sử dụng và tên hàm không được trùng với từ khóa.
- Hàm main luôn được chương trình đọc đầu tiên.
- Muốn sử dụng hàm thì phải có lời gọi hàm (Function call).

Ví dụ:

```
#include <stdio.h>
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int main() {
    int a = 1;
    int b = 2;
    swap(&a, &b);
    printf("%d %d\n", a, b);
    return 0;
}
```

Output:

```
2 1
```

- Có thể khai báo hàm trước main rồi định nghĩa hàm sau, hàm được khai báo trước được gọi là hàm nguyên mẫu (Prototype).

Ví dụ:

```
#include <stdio.h>
void swap(int *a, int *b);
int main() {
    int a = 1;
    int b = 2;
    swap(&a, &b);
    printf("%d %d\n", a, b);
    return 0;
}
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

Output:

```
2 1
```

- Hàm đệ quy (Recursive Function) là hàm tự gọi lại chính nó trong phần thân của nó, nhằm lặp lại quá trình tính toán theo một quy tắc nhất định cho đến khi đạt điều kiện dừng.

Ví dụ:

```
int fibonacci(int n) {
    if (n == 0 || n == 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

1.2) Con trỏ (Pointer)

- Con trỏ là biến dùng để lưu địa chỉ biến khác.
- Để sử dụng con trỏ phải biết cách dùng hai toán tử cơ bản là toán tử lấy địa chỉ (&) và toán tử lấy giá trị (*).
- Con trỏ dù lấy địa chỉ biến khác nhưng bản thân nó vẫn là một biến nên vẫn có địa chỉ riêng.

Ví dụ:

```
#include <stdio.h>
int main() {
   int a = 2;
   int *p;
   p = &a;
   printf("Gia tri cua bien a: %d\n", a);
   printf("Dia chi cua bien a: %p\n", &a);

   printf("Gia tri cua con tro p: %p\n", p);
   printf("Dia chi cua con tro p: %p\n", &p);

   return 0;
}
```

Output:

```
Gia tri cua bien a: 2
Dia chi cua bien a: 000000000062FE4C
Gia tri cua con tro p: 000000000062FE4C
Dia chi cua con tro p: 000000000062FE40
```

- Để thay đổi giá trị của một biến trong hàm main bằng hàm khác ta dùng phương pháp truyền tham trị hay còn hiểu là truyền địa chỉ của biến vào hàm.

Ví dụ:

```
#include<stdio.h>
void thaydoi(int *x) {
   *x = 100;
}
int main() {
   int a = 10;
   thaydoi(&a);
   printf("%d", a);
   return 0;
}
```

Output:

```
100
```

1.3) Con trỏ mảng

- Con trỏ mảng là con trỏ trỏ đến toàn bộ mảng.

- Tên của mảng là con trỏ trỏ đến phần tử đầu tiên của mảng và chỉ số của các phần tử trong mảng chính là địa chỉ của phần tử đó trong mảng.

Ví dụ:

```
#include<stdio.h>
int main(){
int array[3] = {1, 2, 3};
for(int i = 0; i < 3; i++){
        printf("%d co dia chi: %p\n", array[i], &array[i]);
        printf("%d co dia chi: %p\n\n", *(array + i), (array + i));
}
return 0;
}
```

Output:

```
1 co dia chi: 000000000062FE40
1 co dia chi: 000000000062FE40

2 co dia chi: 000000000062FE44
2 co dia chi: 000000000062FE44

3 co dia chi: 000000000062FE48
3 co dia chi: 000000000062FE48
```

- Chú ý: Bản thân mảng chứa địa chỉ của phần tử đầu tiên của mảng, mảng có thể được thao tác GIỐNG như 1 con trỏ (nhưng mảng và con trỏ KHÔNG hoàn toàn như nhau).

1.4) Mảng con trỏ

- Mảng con trỏ là mảng mà trong đó mỗi phần tử là một con trỏ hay nói cách khác là mảng dùng để chứa địa chỉ.

Ví dụ:

```
#include <stdio.h>
int main() {
   int a = 5, b = 10, c = 15;
   int *array[3];
   array[0] = &a;
   array[1] = &b;
   array[2] = &c;
   for (int i = 0; i < 3; i++) {
      printf("Con tro array[%d] luu dia chi %p va tro toi bien co gia tri %d\n", i, array[i],
*array[i]);
   }
   return 0;
}
```

Output:

```
Con tro array[0] luu dia chi 000000000062FE48 va tro toi bien co gia tri 5
Con tro array[1] luu dia chi 000000000062FE44 va tro toi bien co gia tri 10
Con tro array[2] luu dia chi 000000000062FE40 va tro toi bien co gia tri 15
```

Lưu ý:

```
int *a[3];     // mảng gồm 3 con trỏ
int (*a)[3];   // con trỏ trỏ đến mảng 3 phần tử
```

1.5) Con trỏ hàm

- Con trỏ hàm là con trỏ trỏ đến một hàm trong bộ nhớ ,nói cách khác, con trỏ hàm dùng để lưu địa chỉ của một hàm để có thể gọi hàm đó thông qua con trỏ.

Ví dụ:

```
#include <stdio.h>
int add(int a, int b) {
   return a + b;
}
int main() {
   int (*ptr)(int, int);
   ptr = add;
   int a = 5, b = 10;
   printf("Tong: %d\n", ptr(a, b));
return 0; }
```

Output:

```
Tong: 15
```

1.6) Cấp phát động

- Cấp phát động (Dynamic memory allocation) là một kỹ thuật giúp bạn có thể xin cấp phát một vùng nhớ phù hợp với nhu cầu của bài toán trong lúc thực thi thay vì phải khai báo cố định.
- Cấp phát động thường được sử dụng để cấp phát mảng động hoặc sử dụng trong các cấu trúc dữ liệu.
- Có nhiều cấu trúc dữ liệu quan trọng dựa trên kỹ thuật này như : Danh sách liên kết, cây nhị phân hay các cấu trúc dữ liệu dạng mảng động.
- Khi bạn sử dụng cấp phát động thì vùng nhớ cấp phát sẽ là vùng nhớ heap.
- Trong ngôn ngữ lập trình C cung cấp cho bạn 4 hàm trong thư viện <stdlib> để bạn có thể thao tác với việc cấp phát động vùng nhớ và giải phóng vùng nhớ sau khi sử dụng, bao gồm : malloc(), calloc(), free(), realloc().
- Trong đó malloc() là cấp phát động vùng nhớ dùng để cấp phát bộ nhớ theo kích thước byte mong muốn.

Ví dụ:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int *a;
   int n;

   printf("Nhap so phan tu: ");
   scanf("%d", &n);

   a = (int*)malloc(n * sizeof(int));

   if(a == NULL) {
      printf("Khong du bo nho");
      return 1;
   }

   for(int i = 0; i < n; i++) {
      printf("a[%d] = ", i);
      scanf("%d", &a[i]);
   }

   printf("Mang vua nhap:\n");
   for(int i = 0; i < n; i++) {
      printf("%d ", a[i]);
   }
   free(a);
   return 0;
}
```

Output:

```
Nhap so phan tu: 4
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
Mang vua nhap:
1 2 3 4
```

- Chú ý:

- Luôn kiểm tra kết quả trả về của realloc(), calloc() và malloc() vì khi cấp phát thất bại kết quả trả về sẽ là NULL, việc sử dụng con trỏ NULL sẽ gây lỗi chương trình.
- Nếu không giải phóng vùng nhớ bằng free() thì vùng nhớ sẽ không được trả lại gây rò rỉ vùng nhớ (memory leak).

1.7) Xử lí tệp

- Ta có thể làm việc với tệp tin trong C thông qua các hàm trong thư viện <stdlib.h>. Để làm việc với tệp tin thì đầu tiên ta phải khai báo con trỏ kiểu FILE.

Ví dụ: khai báo con trỏ FILE có tên là p.

```
FILE *p;
```

- Dưới đây là các thao tác cơ bản với tệp tin:
  - fopen(): Mở tệp tin, nếu chưa có thì sẽ tự động được tạo
  - fscanf(), fgets(), fgetc(): Đọc từ tệp
  - fprintf(), fputs(), fputc(): Ghi vào tệp
  - fclose(): Đóng file
  - feof(*FILE): Kiểm tra tín hiệu EOF (end of file)
- Với việc mở tệp bằng fopen() ta có thể chọn chế độ làm việc với tệp với cú pháp:

```
FILE *(tên con trỏ FILE) = fopen(“ten_tep.txt”, “chế độ”):
```

- các chế độ cơ bản khi sử dụng fopen()
  - “r”: Mở tệp để đọc
  - “w”: Mở để ghi hoặc xóa nội dung
  - “a”: Thêm vào cuối tệp
  - “rb”: Mở tệp nhị phân
  - “wb”: Mở để ghi hoặc xóa nội dung nhị phân

Ví dụ: Chương trình sao chép một chuỗi nhập từ bàn phím vào file

```c
#include <stdio.h>
#include <stdlib.h>
void ghiChuoiVaoFile(char *tenFile, char *chuoi) {
    FILE *f = fopen(tenFile, "w");
    if (f == NULL) {
        printf("Khong the mo file de ghi!\n");
        exit(1);
    }
    fputs(chuoi, f);
    fclose(f);
}

void docChuoiTuFile(char *tenFile, char *chuoi, int kichThuoc) {
    FILE *f = fopen(tenFile, "r");
    if (f == NULL) {
        printf("Khong the mo file de doc!\n");
        exit(1);
    }
    fgets(chuoi, kichThuoc, f);
    fclose(f);
}

int main() {
    char s[100];
    char doc[100];
    char tenFile[] = "chuoi.txt";

    printf("Nhap mot chuoi: ");
    fgets(s, sizeof(s), stdin);

    ghiChuoiVaoFile(tenFile, s);
    docChuoiTuFile(tenFile, doc, sizeof(doc));

    printf("Chuoi doc tu file: %s", doc);

    return 0;
}
```

Output:

```
Nhap mot chuoi: xin chao
Chuoi doc tu file: xin chao
```

- Lúc này thư mục "chuoi.txt" sẽ được tạo trong cùng một thư mục chứa chương trình này trong trường hợp chưa có. Nội dung file "chuoi.txt" như sau:

```
xin chao
```

1.8) Kiểu cấu trúc

- Cấu trúc (struct) là một kiểu dữ liệu do người lập trình tự định nghĩa.
- Nó cho phép nhóm nhiều biến khác nhau (có thể khác kiểu dữ liệu) vào cùng một kiểu mới.
- Cấu trúc thường được dùng để biểu diễn một đối tượng cụ thể trong thực tế, ví dụ như: một sinh viên gồm mã số, họ tên, điểm số,...
- Mỗi biến trong cấu trúc được gọi là thành phần (member) của cấu trúc.
- Để khai báo một biến thuộc kiểu cấu trúc, dùng cú pháp :

```
struct TênCấuTrúc tên_biến;
```

Ví dụ về cấu trúc :

```
struct SinhVien {
   int maSV;
   float diem;
};
```

Để truy cập vào các trường dữ liệu của cấu trúc bạn dùng toán tử '.' (dot operator), ví dụ : sv.maSV.

Dùng "typedef" để tạo biệt danh (alias) cho struct, việc này giúp người viết không cần phải khai báo lại struct mỗi lần gọi.

Ví dụ:

```
typedef struct Sinhvien
 SinhVien;
```

Ví dụ về cấu trúc sinh viên:

```
#include <stdio.h>
#include <string.h>
struct SinhVien {
char hoten[30];
   int maSV;
   float diem;
};

typedef struct SinhVien SinhVien;

int main() {
   SinhVien sv;
    strcpy(sv.hoten, "Le Hoang Nhan");
   sv.maSV = 101;
   sv.diem = 8.5;
   printf("Ho ten sinh vien: %s\n", sv.hoten);
   printf("Ma so sinh vien: %d\n", sv.maSV);
   printf("Diem cua sinh vien: %.2f\n", sv.diem);
   return 0;
}
```

Output:

```
Ho ten sinh vien: Le Hoang Nhan
Ma so sinh vien: 101
Diem cua sinh vien: 8.50
```

1.9) Danh sách liên kết

- Danh sách liên kết là một cấu trúc dữ liệu được sử dụng để lưu trữ các phần tử tương tự như mảng nhưng vẫn có những điểm khác biệt. Những phần tử trong DSLK gọi là Node gồm có 2 phần là:
  - Dữ liệu (data): Chứa giá trị cần lưu
  - Con trỏ (next): Dùng để trỏ đến Node tiếp theo
- Ví dụ khai báo cấu trúc tự trỏ:

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

- Tính chất:
    - DSLK có thể mở rộng hay thu hẹp một cách linh hoạt
    - Các phần tử trong DSLK được gọi là Node được cấp phát động
    - Phần tử cuối cùng trong danh sách liên kết trỏ tới NULL
    - Đây là cấu trúc dữ liệu cấp phát động nên khi còn bộ nhớ thì sẽ còn thêm được phần tử trong danh sách liên kết
    - Dù không lãng phí bộ nhớ nhưng vẫn cần bộ nhớ để lưu con trỏ

Ví dụ minh họa về vị trí con trỏ trỏ tới trong DSLK:

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
int main() {
    struct Node *head = NULL;
    struct Node *n1 = NULL;
    struct Node *n2 = NULL;
    n1 = (struct Node*)malloc(sizeof(struct Node));
    n2 = (struct Node*)malloc(sizeof(struct Node));
    n1->data = 10;
    n1->next = n2;
    n2->data = 20;
    n2->next = NULL;
    head = n1;
    struct Node *temp = head;
    while (temp != NULL) {
        printf("Gia tri cua node: %d\n", temp->data);
        printf("Dia chi node hien tai: %p\n", temp);
        printf("Dia chi node tiep theo: %p\n\n", temp->next);
        temp = temp->next;
    }
    free(n1);
    free(n2);
    return 0;
}
```

Output:

```
Gia tri cua node: 10
Dia chi node hien tai: 0000000000A513F0
Dia chi node tiep theo: 0000000000A51410

Gia tri cua node: 20
Dia chi node hien tai: 0000000000A51410
Dia chi node tiep theo: 0000000000000000
```

- Từ ví dụ trên ta rút ra được nhận xét:
  - Mỗi Node đều chứa địa chỉ Node kế tiếp của nó
  - DSLK là việc tạo những Node rỗng -> đưa dữ liệu vào -> liên kết chúng với nhau bằng con trỏ next

CHƯƠNG 2. ỨNG DỤNG

A. Đề bài: Xây dựng ứng dụng cho bài toán cụ thể với đầy đủ tính năng cần thiết (VD: bài quản lý SV, Đoàn viên, Số Phức…)

***Yêu cầu:**

1. Ứng dụng con trỏ, cấp phát động để lấy ví dụ minh họa về cấu trúc mảng
2. Ứng dụng danh sách liên kết lấy tối thiểu 1 ví dụ
3. Giao diện/menu chương trình: sẽ được cộng điểm

B. Bài làm

2.1) Quản lí món ăn

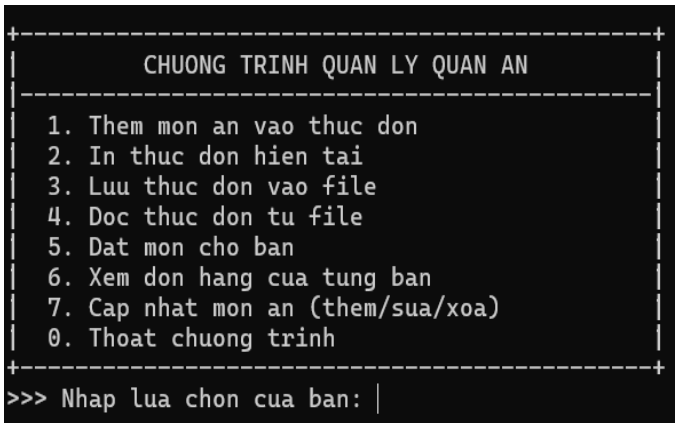
2.1.1) Mục tiêu

- Chương trình hỗ trợ quản lý thực đơn và đặt món cho từng bàn trong quán ăn nhỏ. Người dùng có thể thêm, sửa, xóa món ăn, lưu/đọc file và quản lý đơn hàng cho 5 bàn.

2.1.2) Chức năng

- Nhập/xuất thực đơn
- Đặt món theo bàn
- Xem đơn hàng và tổng tiền
- Cập nhật món ăn (thêm, sửa, xóa)
- Lưu/đọc thực đơn từ file

2.1.3) Giao diện



2.2) Tạo giao diện đăng nhập

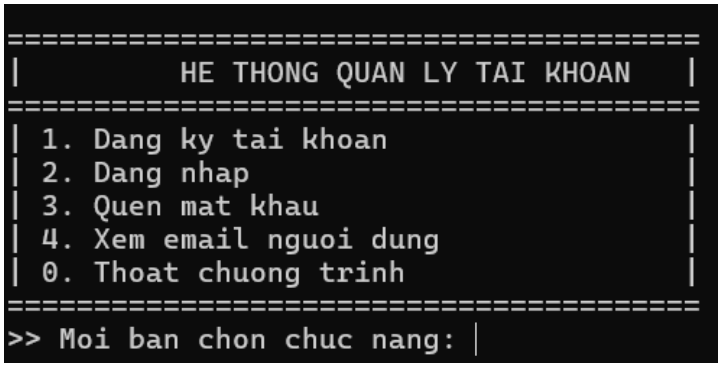
2.2.1) Mục tiêu

- Xây dựng hệ thống đăng ký, đăng nhập và khôi phục tài khoản đơn giản sử dụng file văn bản để lưu trữ. Người dùng có thể tạo tài khoản, đăng nhập hệ thống và khôi phục mật khẩu thông qua email đã đăng ký

2.2.2) Chức năng

- Đăng ký tài khoản:
  - Yêu cầu tên đăng nhập ≥ 5 ký tự.
  - Mật khẩu ≥ 8 ký tự, chứa cả chữ hoa, chữ thường và số.
  - Ghi thông tin vào file taikhoan.txt.
- Đăng nhập:
  - Đọc file taikhoan.txt, so sánh tên đăng nhập và mật khẩu.
  - Mỗi tài khoản được nhập sai tối đa 3 lần.
  - Quên mật khẩu:
- Yêu cầu nhập đúng tên đăng nhập.
- Gửi thông báo mật khẩu về email đã lưu
- Xem email người dùng và thông báo mật khẩu nếu đã yêu cầu khôi phục.

2.2.3) Giao diện



2.3) Quản lí sinh viên

2.3.1) Mục tiêu

- Xây dựng hệ thống quản lý danh sách sinh viên sử dụng danh sách liên kết đơn. Hệ thống hỗ trợ nhập, tìm kiếm, xoá, thống kê, lọc và xuất dữ liệu.

2.3.2) Chức năng

- Nhập sinh viên liên tục:
  - Cho phép người dùng nhập sinh viên đến khi nhấn ESC.
  - Dữ liệu bao gồm MSSV, họ tên, điểm.
- Hiển thị danh sách sinh viên:
  - In dạng bảng gồm MSSV, họ tên, điểm.
- Tìm sinh viên theo MSSV.
- Xoá sinh viên theo MSSV.
- Tìm sinh viên có điểm cao nhất.
- Tìm sinh viên trong khoảng điểm nhập từ bàn phím.
- Tách danh sách theo điều kiện đạt (điểm ≥ 5) và không đạt:
  - Ghi 2 danh sách ra file dat.txt và khongdat.txt.

2.3.3) Giao diện

CHƯƠNG 3. KẾT QUẢ VÀ KIẾN NGHỊ

3.1) Kết quả đạt được

- Thông qua việc xây dựng các chương trình trên, sinh viên đã vận dụng hiệu quả các kiến thức nền tảng của ngôn ngữ C, bao gồm: khai báo và sử dụng struct, thao tác với mảng, con trỏ, xử lý tệp văn bản, cũng như thiết kế giao diện menu thân thiện trên môi trường dòng lệnh.
- Chương trình không chỉ đáp ứng đầy đủ các yêu cầu cơ bản của một hệ thống quản lý quán ăn nhỏ mà còn cho thấy khả năng mở rộng linh hoạt, tổ chức dữ liệu rõ ràng và dễ bảo trì.

3.2) Kiến nghị

- Trong các lần nâng cấp sau, chương trình có thể bổ sung một số tính năng để sử dụng thuận tiện hơn như :

- Cho phép tìm kiếm món ăn theo tên hoặc giá.
- Thêm chức năng tính tổng doanh thu của quán theo từng ngày hoặc từng bàn.
- Lưu dữ liệu bằng file nhị phân để đảm bảo độ an toàn và tiết kiệm bộ nhớ hơn.
- Bổ sung mục in hóa đơn cho từng bàn sau khi đặt món.
- Cải thiện giao diện menu để dễ thao tác hơn, nhất là khi danh sách món nhiều.

TÀI LIỆU THAM KHẢO

[1]. https://blog.28tech.com.vn/c **”Ngôn Ngữ Lập Trình C”**

[Truy cập 5/2025].

[2].https://drive.google.com/file/d/1X_ULt4u_qEYl2HPbQwTVKagNPCQatw2C/view?hl=enhttps://vnce.vn/7-cong-cu-quan-ly-chat-luong **“Learn C The Hard Way”**

[Truy cập 5/2025].

[3].https://drive.google.com/file/d/1eUuBh1heBiapbwNybyhhWmsEU-by1aAt/view?hl=en **“The C Programming Language 2nd Ed – Prentice Hall Brian.W.Kernighan and Dennis.M.Ritchie”**

[Truy cập 5/2025].

[4].https://drive.google.com/file/d/1edGs_p9oJYEE1FnUl-Zc8WSjaJsLvlGx/view?hl=en **“Giáo Trình”**

[Truy cập 5/2025].