**Báo cáo chương trình quản lý nhà hàng**

[I. Phân công công việc 1](#_Toc200793738)

[II. Các chức năng của hệ thống 1](#_Toc200793739)

[Quản lý khách hàng 1](#_Toc200793740)

[Quản lý bàn ăn 2](#_Toc200793741)

[Quản lý hàng đợi 4](#_Toc200793742)

[Quản lý món ăn và thông tin nhà hàng 5](#_Toc200793743)

[Giao diện người dùng 7](#_Toc200793744)

[Tiện ích và xử lý thời gian 11](#_Toc200793745)

[Chương trình chính 11](#_Toc200793746)

[III. Ưu điểm và Nhược điểm của chương trình..13](#_Toc200793747)

[Ưu điểm: 13](#_Toc200793748)

[Nhược điểm: 13](#_Toc200793749)

# I. Phân công công việc

- Nguyễn Thị Ánh Tuyết: Làm phần khách hàng (customer.c)

- Nghiêm Thị Thanh Hường: Làm phần món ăn (restaurant.h)

- Nguyễn Tuấn Anh: Làm phần bàn và hóa đơn (table.c, queue.c)

- Quyền Đức Anh: Làm phần bàn và hóa đơn (table.c, queue.c), merge các phần lại và thêm giao diện (main.c, ui.c)

# II. Các chức năng của hệ thống

## Quản lý khách hàng

**File: customer.c**

#include "restaurant.h"  
  
void addToList(List \*l, Customer c) {  
 if (l->count >= MAX) {  
 printf("Danh sach day\n");  
 return;  
 }  
 l->list[l->count++] = c;  
}  
  
void displayList(List \*l, const char \*title) {  
 if (l->count == 0) {  
 printf("Khong co KH trong %s\n", title);  
 return;  
 }  
 printf("%s:\n", title);  
 for (int i = 0; i < l->count; i++)  
 printf("%d. %s | SDT: %s | Thoi gian: %s\n", i + 1, l->list[i].name,  
 l->list[i].phone, l->list[i].time);  
}  
  
void addScheduledCustomer(List \*l) {  
 if (l->count >= MAX) {  
 printf("Danh sach day\n");  
 return;  
 }  
 Customer c;  
 printf("Ten KH: ");  
 getchar();  
 fgets(c.name, sizeof(c.name), stdin);  
 c.name[strcspn(c.name, "\n")] = '\0';  
 printf("SDT: ");  
 fgets(c.phone, sizeof(c.phone), stdin);  
 c.phone[strcspn(c.phone, "\n")] = '\0';  
 printf("Thoi gian hen (vd: 19:00): ");  
 fgets(c.time, sizeof(c.time), stdin);  
 c.time[strcspn(c.time, "\n")] = '\0';  
 l->list[l->count++] = c;  
 printf("Da them khach hen gio\n");  
}  
  
void moveToDiningList(Queue \*q, List \*diningList) {  
 if (isEmptyQueue(q)) {  
 printf("Khong co khach trong hang cho\n");  
 return;  
 }  
 Customer c = q->data[q->F];  
 getCurrentTime(c.time, sizeof(c.time));  
 addToList(diningList, c);  
 dequeue(q);  
 printf("Khach hang %s dang dung bua\n", c.name);  
}

## Quản lý bàn ăn

**File: table.c**

#include "restaurant.h"  
  
void init\_Table(Table \*\*table) { \*table = NULL; }  
  
void add\_Table(Table \*\*table, int tableNumber, int status) {  
 Table \*newTable = (Table \*)malloc(sizeof(Table));  
 newTable->tableNumber = tableNumber;  
 newTable->status = status;  
 strcpy(newTable->customerName, "");  
 strcpy(newTable->orderDetails[0], "Empty");  
 for (int i = 1; i < MAX\_ORDER; ++i)  
 strcpy(newTable->orderDetails[i], "");  
 newTable->next = \*table;  
 \*table = newTable;  
}  
  
Table \*search\_Table(Table \*table, int tableNumber) {  
 while (table != NULL) {  
 if (table->tableNumber == tableNumber)  
 return table;  
 table = table->next;  
 }  
 return NULL;  
}  
  
void book\_NewTable(Table \*\*table, int num) {  
 Table \*temp = search\_Table(\*table, num);  
 if (temp == NULL) {  
 printf("Khong tim thay ban\n");  
 return;  
 }  
 if (temp->status == 1) {  
 printf("Ban da duoc dat\n");  
 return;  
 }  
 temp->status = 1;  
 char name[MAX\_NAME\_LEN], order[MAX\_NAME\_LEN];  
 int i = 0;  
  
 printf("Nhap ten khach: ");  
 scanf("%s", name);  
 strcpy(temp->customerName, name);  
  
 printf("Nhap mon an (go X de ket thuc):\n");  
 while (1) {  
 scanf("%s", order);  
 if (strcmp(order, "X") == 0 || strcmp(order, "x") == 0)  
 break;  
 if (i < MAX\_ORDER) {  
 strcpy(temp->orderDetails[i], order);  
 i++;  
 } else {  
 printf("Danh sach mon day\n");  
 break;  
 }  
 }  
 for (int j = i; j < MAX\_ORDER; ++j)  
 strcpy(temp->orderDetails[j], "");  
}  
  
void show\_Table(Table \*table) {  
 if (table == NULL) {  
 printf("Khong co ban nao\n");  
 return;  
 }  
 while (table != NULL) {  
 printf("Ban so: %d - %s\n", table->tableNumber,  
 table->status == 0 ? "Trong" : "Da dat");  
 if (table->status == 1) {  
 printf(" Khach: %s\n Goi mon: ", table->customerName);  
 for (int i = 0; i < MAX\_ORDER && strlen(table->orderDetails[i]) > 0; ++i)  
 printf("%s ", table->orderDetails[i]);  
 printf("\n");  
 }  
 table = table->next;  
 }  
}  
  
void free\_TableList(Table \*\*table) {  
 while (\*table != NULL) {  
 Table \*temp = \*table;  
 \*table = (\*table)->next;  
 free(temp);  
 }  
}

## Quản lý hàng đợi

**File: queue.c**

#include "restaurant.h"  
  
void initQueue(Queue \*q) { q->F = q->R = -1; }  
int isEmptyQueue(Queue \*q) { return q->F == -1; }  
int isFullQueue(Queue \*q) { return q->R == MAX - 1; }  
  
void enqueue(Queue \*q) {  
 if (isFullQueue(q)) {  
 printf("Hang doi day\n");  
 return;  
 }  
 Customer c;  
 printf("Ten KH: ");  
 getchar();  
 fgets(c.name, sizeof(c.name), stdin);  
 c.name[strcspn(c.name, "\n")] = '\0';  
  
 printf("SDT: ");  
 fgets(c.phone, sizeof(c.phone), stdin);  
 c.phone[strcspn(c.phone, "\n")] = '\0';  
  
 getCurrentTime(c.time, sizeof(c.time));  
 if (isEmptyQueue(q))  
 q->F = q->R = 0;  
 else  
 q->R++;  
 q->data[q->R] = c;  
 printf("Da them vao hang doi\n");  
}  
  
void dequeue(Queue \*q) {  
 if (isEmptyQueue(q)) {  
 printf("Hang doi rong\n");  
 return;  
 }  
 printf("Da xoa: %s\n", q->data[q->F].name);  
 if (q->F == q->R)  
 q->F = q->R = -1;  
 else  
 q->F++;  
}  
  
void displayQueue(Queue \*q) {  
 if (isEmptyQueue(q)) {  
 printf("Khong co khach nao\n");  
 return;  
 }  
 printf("Danh sach hang doi:\n");  
 for (int i = q->F; i <= q->R; i++)  
 printf("%d. %s | SDT: %s | Thoi gian: %s\n", i - q->F + 1, q->data[i].name,  
 q->data[i].phone, q->data[i].time);  
}

## Quản lý món ăn và thông tin nhà hàng

**File: restaurant.h**

#ifndef RESTAURANT\_H  
#define RESTAURANT\_H  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <time.h>  
  
#define MAX\_ORDER 100  
#define MAX\_NAME\_LEN 100  
#define MAX 50  
  
// ==== Structs ====  
typedef struct Table {  
 int tableNumber;  
 int status;  
 char customerName[MAX\_NAME\_LEN];  
 char orderDetails[MAX\_ORDER][MAX\_NAME\_LEN];  
 struct Table \*next;  
} Table;  
  
typedef struct {  
 char name[50];  
 char phone[15];  
 char time[30];  
} Customer;  
  
typedef struct {  
 Customer data[MAX];  
 int F, R;  
} Queue;  
  
typedef struct {  
 Customer list[MAX];  
 int count;  
} List;  
  
// ==== Function Declarations ====  
void getCurrentTime(char \*buffer, int size);  
  
// Table  
void init\_Table(Table \*\*table);  
void add\_Table(Table \*\*table, int tableNumber, int status);  
Table \*search\_Table(Table \*table, int tableNumber);  
void book\_NewTable(Table \*\*table, int num);  
void show\_Table(Table \*table);  
void free\_TableList(Table \*\*table);  
  
// Queue  
void initQueue(Queue \*q);  
int isEmptyQueue(Queue \*q);  
int isFullQueue(Queue \*q);  
void enqueue(Queue \*q);  
void dequeue(Queue \*q);  
void displayQueue(Queue \*q);  
  
// Customer list  
void addToList(List \*l, Customer c);  
void displayList(List \*l, const char \*title);  
void addScheduledCustomer(List \*l);  
void moveToDiningList(Queue \*q, List \*diningList);  
  
#endif // RESTAURANT\_H

## Giao diện người dùng

**File: ui.c**

#include "ui.h"  
#include "restaurant.h"  
#include "utils.h"  
#include <conio.h> // Neu dung Windows  
#include <stdio.h>  
#include <string.h>  
  
int menuGiaoDien(const char \*options[], int optionCount) {  
 int selection = 0;  
 char key;  
  
 do {  
 clearScreen();  
 printf("+----------------------------------------------------------+\n");  
 printf("|~~~~~~~~~~~~~~~~~~ NHA HANG 9YOK MINIONS ~~~~~~~~~~~~~~~~~|\n");  
 printf("+----------------------------------------------------------+\n\n");  
  
 for (int i = 0; i < optionCount; ++i) {  
 if (i == selection)  
 printf(">> %s <<\n", options[i]);  
 else  
 printf(" %s\n", options[i]);  
 }  
  
 key = \_getch();  
  
 if (key == 72) // ↑  
 selection = (selection - 1 + optionCount) % optionCount;  
 else if (key == 80) // ↓  
 selection = (selection + 1) % optionCount;  
 else if (key == 13) // Enter  
 break;  
  
 } while (1);  
  
 return selection;  
}  
  
void xoaMonAn(struct Table \*tableList) {  
 int tableNum;  
 printf("Nhap so ban can sua: ");  
 scanf("%d", &tableNum);  
 Table \*t = search\_Table(tableList, tableNum);  
 if (!t || t->status == 0) {  
 printf("Khong tim thay ban da dat.\n");  
 return;  
 }  
 printf("Danh sach mon hien tai:\n");  
 for (int i = 0; i < MAX\_ORDER && strlen(t->orderDetails[i]) > 0; i++) {  
 printf("%d. %s\n", i + 1, t->orderDetails[i]);  
 }  
 int idx;  
 printf("Nhap so thu tu mon muon xoa: ");  
 scanf("%d", &idx);  
 idx--;  
 if (idx < 0 || idx >= MAX\_ORDER || strlen(t->orderDetails[idx]) == 0) {  
 printf("Mon khong hop le.\n");  
 return;  
 }  
 for (int i = idx; i < MAX\_ORDER - 1; i++) {  
 strcpy(t->orderDetails[i], t->orderDetails[i + 1]);  
 if (strlen(t->orderDetails[i + 1]) == 0)  
 break;  
 }  
 strcpy(t->orderDetails[MAX\_ORDER - 1], "");  
 printf("Da xoa mon an.\n");  
}  
  
void inHoaDon(Table \*tableList) {  
 int tableNum;  
 printf("Nhap so ban can in hoa don: ");  
 scanf("%d", &tableNum);  
 Table \*t = search\_Table(tableList, tableNum);  
 if (!t || t->status == 0) {  
 printf("Khong tim thay ban da dat.\n");  
 return;  
 }  
 printf("\n=== HOA DON ===\n");  
 printf("Ban so: %d\n", t->tableNumber);  
 printf("Khach hang: %s\n", t->customerName);  
 printf("Danh sach mon:\n");  
 for (int i = 0; i < MAX\_ORDER && strlen(t->orderDetails[i]) > 0; i++) {  
 printf("- %s\n", t->orderDetails[i]);  
 }  
 printf("===============\n\n");  
}  
  
void runUI() {  
 Queue q;  
 List scheduledList = {.count = 0};  
 List diningList = {.count = 0};  
 Table \*tableList;  
  
 initQueue(&q);  
 init\_Table(&tableList);  
  
 const char \*menu[] = {"1. Them khach vao hang doi",  
 "2. Them khach hen gio",  
 "3. Hien thi hang doi",  
 "4. Hien thi danh sach hen",  
 "5. Chuyen khach sang dang dung bua",  
 "6. Hien thi danh sach dang dung bua",  
 "7. Dat ban",  
 "8. Hien thi cac ban",  
 "9. Xoa mon an",  
 "10. In hoa don",  
 "11. Thoat"};  
 int menuCount = sizeof(menu) / sizeof(menu[0]);  
  
 while (1) {  
 int choice = menuGiaoDien(menu, menuCount);  
 clearScreen();  
  
 switch (choice) {  
 case 0:  
 enqueue(&q);  
 break;  
 case 1:  
 addScheduledCustomer(&scheduledList);  
 break;  
 case 2:  
 displayQueue(&q);  
 break;  
 case 3:  
 displayList(&scheduledList, "Danh sach khach hen");  
 break;  
 case 4:  
 moveToDiningList(&q, &diningList);  
 break;  
 case 5:  
 displayList(&diningList, "Dang dung bua");  
 break;  
 case 6: {  
 int tableNum;  
 printf("Nhap so ban can dat: ");  
 scanf("%d", &tableNum);  
 book\_NewTable(&tableList, tableNum);  
 break;  
 }  
 case 7:  
 show\_Table(tableList);  
 break;  
 case 8:  
 xoaMonAn(tableList);  
 break;  
 case 9:  
 inHoaDon(tableList);  
 break;  
 case 10:  
 free\_TableList(&tableList);  
 return;  
 }  
  
 printf("\nNhan phim bat ky de tiep tuc...");  
 getchar();  
 getchar();  
 }  
}

**File: ui.h**

#ifndef UI\_H  
#define UI\_H  
  
#include "restaurant.h"  
  
void runUI();  
  
int menuGiaoDien(const char \*options[], int optionCount);  
  
void xoaMonAn(Table \*tableList);  
  
void inHoaDon(Table \*tableList);  
  
#endif

## Tiện ích và xử lý thời gian

**File: utils.c**

#include <stdio.h>  
#include <stdlib.h>  
  
#ifdef \_WIN32  
#include <windows.h>  
#else  
#include <unistd.h>  
#endif  
  
void clearScreen() {  
#ifdef \_WIN32  
 system("cls");  
#else  
 system("clear");  
#endif  
}

**File: utils.h**

#ifndef UTILS\_H  
#define UTILS\_H  
  
void clearScreen();  
  
#endif // UTILS\_H

**File: timeutils.c**

#include "restaurant.h"  
  
void getCurrentTime(char \*buffer, int size) {  
 time\_t now = time(NULL);  
 struct tm \*t = localtime(&now);  
 strftime(buffer, size, "%d/%m/%Y %H:%M:%S", t);  
}

## Chương trình chính

**File: main.c**

#include "ui.h"  
  
int main() {  
 runUI();  
 return 0;  
}

# III. Ưu điểm và Nhược điểm của chương trình

## Ưu điểm:

- Giao diện console đơn giản, dễ thao tác.  
- Có chức năng quản lý khách chờ, khách hẹn và khách đang ăn.  
- Cho phép đặt bàn, gọi món, xóa món và in hóa đơn.  
- Dễ dàng mở rộng hoặc chỉnh sửa các chức năng khác.

## Nhược điểm:

- Chưa có lưu trữ dữ liệu vĩnh viễn (mất khi tắt chương trình).  
- Chỉ có giao diện console, chưa thân thiện với người dùng phổ thông.  
- Chưa có tính năng quản lý nhân viên và báo cáo doanh thu.  
- Việc xử lý dữ liệu vẫn thủ công, khó quản lý khi quy mô lớn.