UNIVERSITI TUNKU ABDUL RAHMAN
LKC FES
UECS1104 Programming and Problem Solving
Assignment 202401

Lecturer in charge **:** Mr. Lau Kean Hong (laukh@utar.edu.my)

Group no : 15

Members :4

Name	ID	3E etc.	Email
Darren Kong Yan Ren	2207505	SE	xinachan@1utar.my
Lim Cammy	2206892	SE	cam2yl857.0@1utar.my
Liow Ke Han	2204872	SE	liow1144kehan@1utar.my
Wong Yu Chi	2207129	SE	yukiwong221@1utar.my

TABLE OF CONTENTS

1.0 INPUT AND OUTPUT DESIGN

The food ordering system is aimed at providing users with different operation functions based on the specific positions of different users. On the manager page, the manager will be allowed to edit the dish menu by adding or deleting a dish and its information, including dish code, dish name, and dish unit price that are stored in the system. Also, the system will produce reports for daily statistics or overall statistics, which include daily sales overall, daily order overall, daily sales per dish, quantity per dish, average preparation time per dish, average daily sales, and average daily orders for analysis purposes used by the manager. For the food operator, once the payment is made by the customer, the order will be sent to the kitchen, which will start the timer to count the preparation time for the dishes. The food operator page allows food operators to check and make a comment on the dish once it is ready. The preparation time per dish will be stored. On the customer page, customers will be able to place an order, and make a payment.

1.2 Main Menu

The main menu provides position options for user. User will be able to choose his or her position and proceed to a specific page or quit from the system. By inputting "A", "B", or "C", user will be brought to the manager page, the food operator page, or the customer page, respectively. On each page, the user will be able to perform different operations based on his or her roles. By inputting "Q", user will be able to quit from the food ordering system. If user enters an invalid input, the system will require the user to re-enter a valid input. If the input is valid, the page will be refreshed for a better user experience.

On the manager page, user will be able to input "A" to process into the edit menu, "B" to check the daily statistics and overall statistics, or "Q" to quit from the current page.

On the food operator page, On the food operator page, it will show completed and unfinished orders, and show all orders. Not only that, the preparation time of each order will also be displayed in hours, minutes, and seconds.

On the customer page, users will be able to proceed directly to order processing. Users can enter dish code, quantity and whether to continue ordering. Once the order is completed, a receipt will be generated to the customer so that he can clearly know the dish ordered and the price of the dish.

1.2.1 Edit Menu

Once the user enters the edit menu page, all dishes that are stored will be displayed in tabular format. After that, three options will be provided to the user. If the user enters “A” to add a dish to the dish menu, the system will require the user to key in the dish code, dish name, and dish unit price. If the dish code or dish name already exists or is in invalid format, an error message will display to tell the user the action is unsuccessful, and the adding process will be stopped. If the user enters “D” to delete a dish, the specified dish will be deleted from the dish list. If the dish code is in invalid format or it does not exist in the dish menu, an error message will be displayed to tell the user that the action was unsuccessful, and the deleting process will be stopped.

The updated dish menu will be stored, and an update success message will be displayed once the operations of adding or deleting are processed successfully. The user can enter “Q” to quit from the current page at any time. After the user is done adding, deleting, or entering invalid input, the page will be refreshed, and the user can enter “Y” to stay on the edit page to continue editing the dish menu, or “N” to quit the edit menu page.

1.2.2 Statistics Menu

Once the user enters the statistics menu part, the user will be required to choose from three options, which are ‘1’, ‘2’, and ‘3’. If user chooses ‘1’, the system will direct the user to the overall statistics of the restaurant, which includes all operating days’ daily sales overall, daily order overall, daily sales per dish, quantity per dish, and average preparation time per dish that will be displayed in table form, and average daily sales and average daily order which will be displayed separately, not included in the table. If user chooses ‘2’, the user will be required to input year, month, and day one-by-one. If the input is invalid, an error message will be prompted out and the user is required to re-enter the valid input. If the input is valid, the daily statistics for a particular date will be displayed. The data to be displayed includes daily sales overall, daily order overall, daily sales per dish, quantity per dish, and average preparation time per dish, but for the specified date only. User is allowed to enter ‘Q’ or ‘q’ to quit to the selection page. For choices ‘1’ and ‘2’, after the statistics is successfully displayed, the system will prompt out message to ask the user whether they want to repeat the operation again. If user enters ‘Y’ or ‘y’, the system will go to the selection page for the user to choose from the choices again. If user selects ‘3’, the system will return to the main menu.

1.3 Food Operator Page

Firstly, the food operators will view the incomplete orders and they are required to enter the order code. The order code that is entered by them will be validated. For example, the maximum length of the order code should be maximum 9 not including the delimiter, and all of them are randomly generated integers, for instance, 205631060. Next, the food operators will be taken to the page where all the dishes’ code, name, and quantity are displayed for that particular order chosen. Input will be prompted to food operators to select which dish should be marked as done and it is ready to serve to the customers. Once that order is marked as done, the order will be viewed under completed order sections. The time when the customers make their payments are recorded in the food ordering page, and once the food operator confirms to mark the order as done, the total preparation time is recorded by subtracting them.

1.4 Customers Page

Once the user enters the customers page, the user will be required to choose from options, which are <A> and <Q>. If the user selects option <A>, the user will enter the order page. On the order page, the user will be able to place an order. If the user selects option <Q>, the user will exit the selection page, and return to the main menu.

1.4.1 Order

Once the user enters the order page, all dishes that are stored will be displayed in the tabular format. After that, there are tips below the menu words: A for food, B for drinks, and C for cake. And each area is also planned with lines to make it look neater. Coming to the menu below, user can see that the leftmost is the dish code, the middle is the dish name, and the next is the price of the dish in Ringgit Malaysia (RM).

The system will ask the user to enter the dish code, the quantity of the dish, and whether to continue. If the dish code entered is incorrect, the page will be refreshed and asked to re-enter. If the quantity of the dish entered is less than 1 or greater than 99, the system will display invalid quantity amount. If the user selects “N”, then the receipt will be directly entered. If the user enters “Y”, the system will refresh the interface and ask the user to enter a new dish code, quantity, and whether to continue, until the user enters “N”.

On the receipt interface. User's order id will be displayed in the upper left corner. Next is the dish code, dish name, dish quantity, dish unit price and total price of each dish. At the bottom, the system will display the total price of all dishes ordered by the user. The user can enter "Y" to stay on the order page or enter "N" to exit the order page.

2.0 PROGRAM FILE AND DESCRIPTION OF DETAILS

No.	File Name	Description	Used In
1	dish_menu.txt	“dish_menu.txt” is used to store all dish items and its information, including dish code, dish name, and dish unit price. All changes including adding dish and deleting dish will be updated into this file. The edit or update of dish items in “dish_menu.txt” will be done by the manager in edit menu page.	edit_menu(), customer()
2	Statistics.txt	“statistics.txt” is used to store daily statistics. The text file consists of five columns, which are date, daily sales overall in Ringgit Malaysia, daily order overall, dish code, daily sales per dish in Ringgit Malaysia, quantity per dish, and average preparation time per dish. The daily statistics are stored based on the date and are separated by a line of separator “-----” between different dates section. The first row of data for each date consists of date, daily sales overall, and daily order overall, while the other row for each date will not have the date, daily sales overall, and daily order overall. All rows of data will have the records for daily sales per dish, quantity per dish, and average preparation time per dish.	statistics_menu()
3	order.txt	“order.txt” is used to store all order id and its information, including dish code, dish name, and quantity that order by the customer. All the customer order including dish code, dish name, and quantity that order by the customer the will be updated into this file. The order of customer in	customer()

		“dish_menu.txt” will be done by the customer in customers page.	
4	order_ini_time	It is used to store the order code with the time that the customers had already made their payment.	

3.0 DATA MEMORY, FORMULA AND DESCRIPTION

Variable	Data Type	Stored Value
Main Menu		
position	char	To store the input from user to identify position of user.
action	char	To store the input from user to identify operation or function choose by user.
Edit Menu		
read_dish_menu	ifstream	To handle input file stream for “dish_menu.txt” file.
store_dish_menu	ofstream	To handle output file stream for “dish_menu.txt” file.
Dishes[1000]	array of DISH_INFO	To store the information about dishes, including dish code, dish name, and dish unit price.
code	char	To store the dish code (a component of DISH_INFO structure).
name	char	To store the dish name (a component of DISH_INFO structure).
unit_price	double	To store the dish unit price (a component of DISH_INFO structure).
option	char	To store the input from user to identify operation choice by user in edit menu.
num_dishes	int	To store the integer value to represent the number of dishes read from “dish_menu.txt” file.
find_code	char	To store the input from user to identify code of the dish to be deleted.
Statistics Menu		
choice	char	To store the input from user to identify user’s opinion on display overall statistics, display statistics by date, or exit to the main menu.

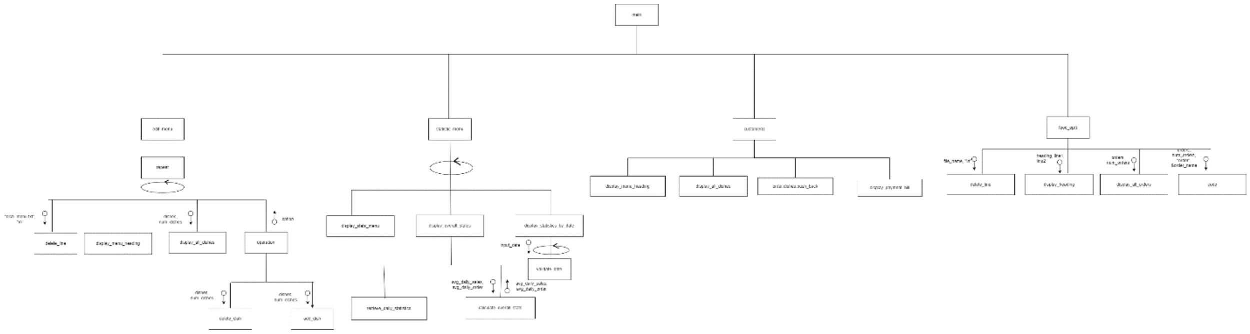
line[1000]	array with data type char	To store line from the text file, with each line can only store up to 1000 characters
In_stats_file	ifstream	To handle input file stream for “statistics.txt” file.
line	string	To store each line read from the file
in_date_section	bool	To indicate if the program read until the section for a specific date
total_sales	double	To store the total sales overall
total_orders	int	To store the total orders overall
num_days	int	To store the number of days with recorded statistics
temp(line)	istringstream	To extract data from the line
date	string	To store the date from text file
daily_sales	double	To store the daily sales from text file
daily_orders	int	To store the daily orders from text file
avg_daily_sales	double	To store the value of average daily sales
avg_daily_order	double	To store the value of average daily orders
year	string	To store the year inputted by user
month	string	To store the month inputted by user
day	string	To store the day inputted by user
input_date	string	To store the input date which consists of the combination of year, month, and day entered by user
header_displayed	bool	To indicate if the header has been displayed
Customers Page		
answer	char	To store the value that identify user want to continue or not.
quantity	int	To store the quantity of dishes that order by user
Dishes[1000]	array of DISH_INFO	To store the information about dishes, including dish code, dish name, and dish unit price.
Num_dishes	int	To store the integer value to represent the number of dishes read from “dish_menu.txt” file.

read_dish_menu	ifstream	To handle input file stream for “dish_menu.txt” file.
menuCode[5]	char	To store the dish code of the user enter
exist	bool	To indicate the Dish code is exist or not
chaos	int	To store a random value let the order ID create randomly
rand_no	int	To store a random value let the order ID create randomly
now	Time_t	To store the current time by a timestamp
order_file	ofstream	To handle output file stream for “order.txt” file.
total_price	double	To store the total price that order by user
dish_total_price	double	To store the each of the total price of each of the dish
Food Operator Page		

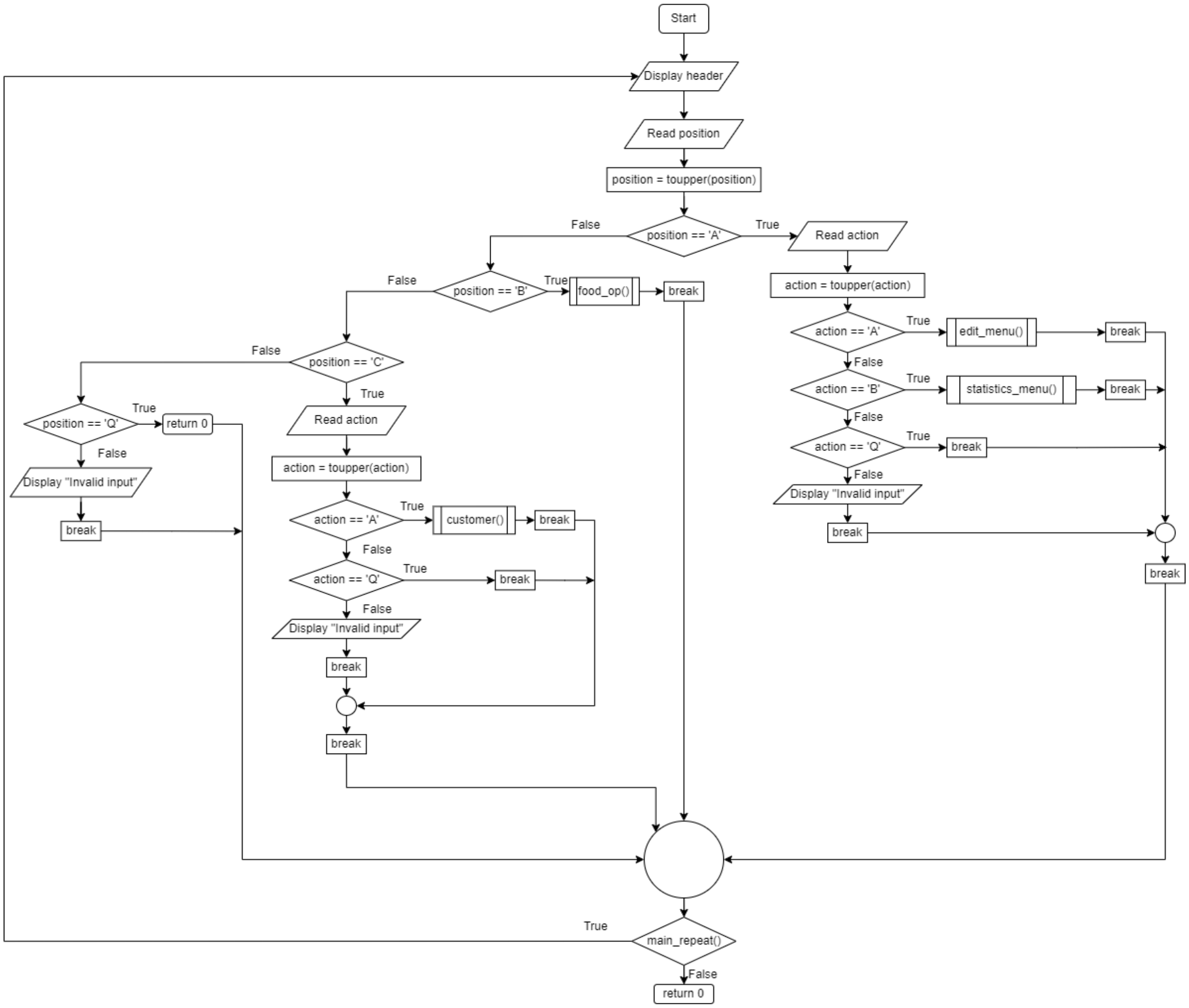
There are several formulae used in calculate_overall_stats (double* avg_daily_sales, double* avg_daily_order) function.

1) total_sales: variable that is used to store the total sales throughout the operating period of the restaurant. It is computed by adding up all the overall daily sales.
   The formula is: “total_sales += daily_sales”.
2) total_orders: variable that is used to store the total orders throughout the operating period of the restaurant. It is computed by adding up all the overall daily orders.
   The formula is: “total_orders += daily_orders”.
3) *avg_daily_sales: variable that is used to store the average value of daily sales. It is computed by dividing the total sales by number of operating days.
   The formula is: “*avg_daily_sales = total_sales / num_days”
4) *avg_daily_order: variable that is used to store the average value of daily order. It is computed by dividing the total orders by number of operating days.
   The formula is: “*avg_daily_order = total_orders / num_days”

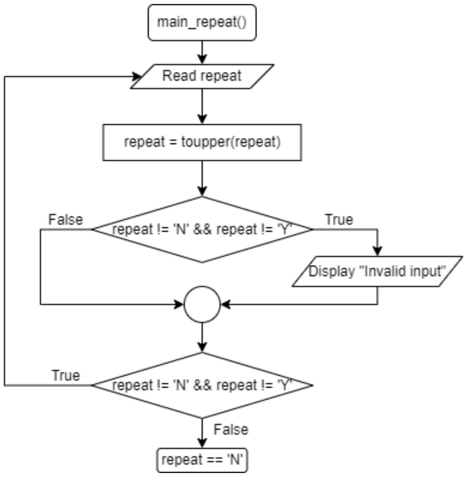
4.0 STRUCTURE CHART AND FLOW CHART



main()

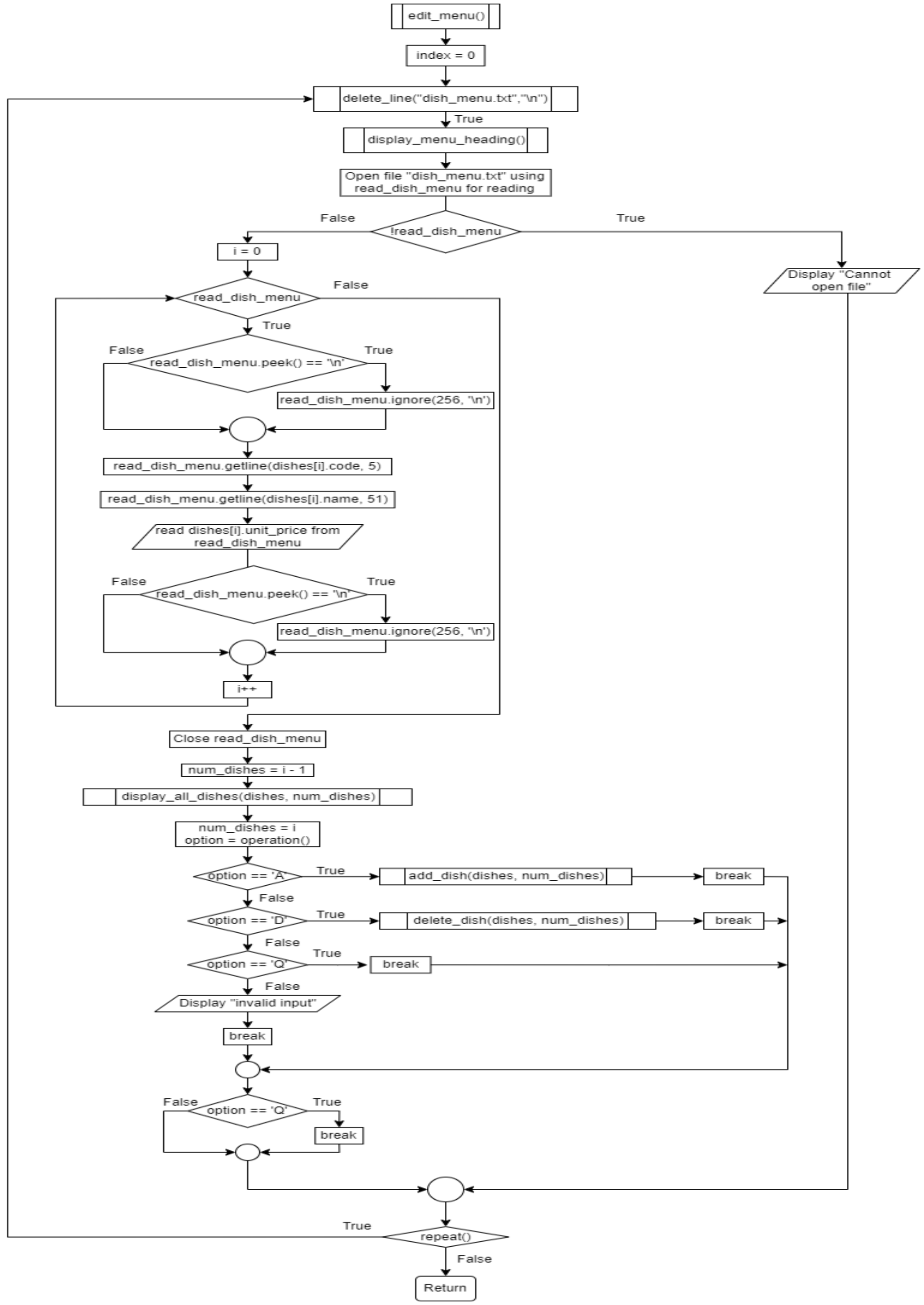


main_repeat()

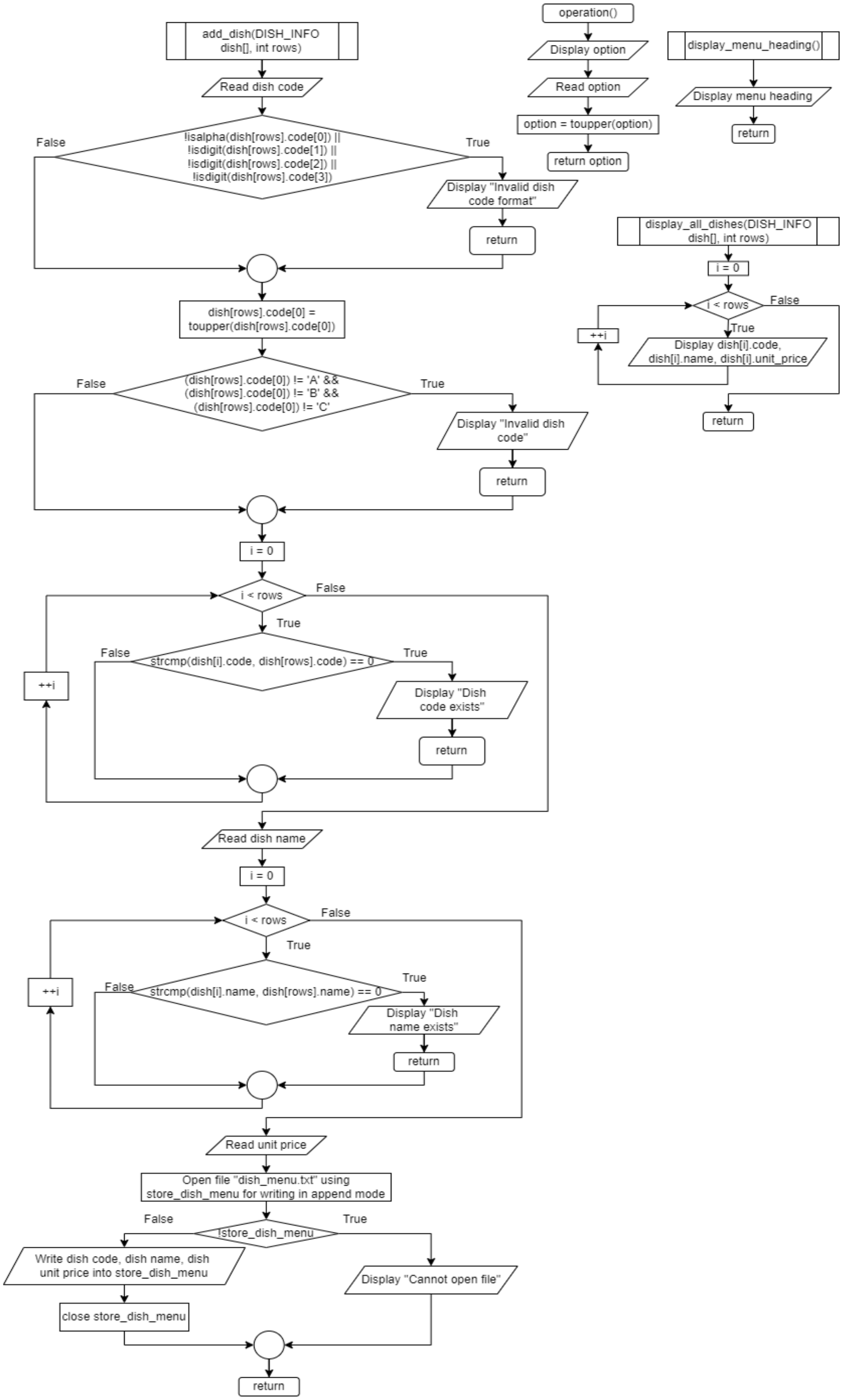


edit_menu()

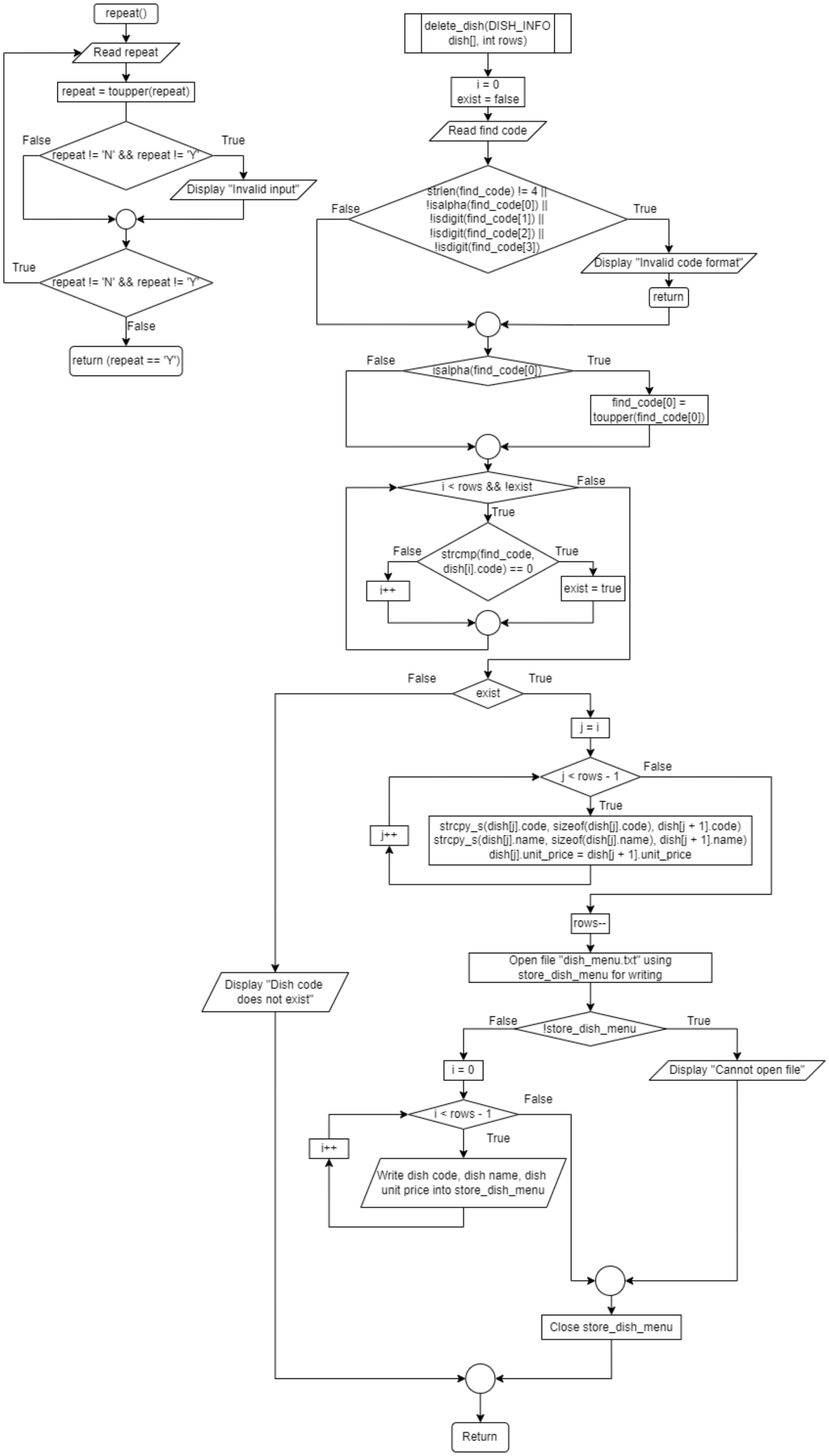
edit_menu()


edit_menu()
index = 0
delete_line("dish_menu.txt","\n")
True
display_menu_heading()
Open file "dish_menu.txt" using read_dish_menu for reading
False
!read_dish_menu
True
i = 0
Display "Cannot open file"
read_dish_menu
False
True
False
read_dish_menu.peek() == '\n'
True
read_dish_menu.ignore(256, '\n')
read_dish_menu.getline(dishes[i].code, 5)
read_dish_menu.getline(dishes[i].name, 51)
read dishes[i].unit_price from read_dish_menu
False
read_dish_menu.peek() == '\n'
True
read_dish_menu.ignore(256, '\n')
i++
Close read_dish_menu
num_dishes = i - 1
display_all_dishes(dishes, num_dishes)
num_dishes = i
option = operation()
option == 'A'
True
add_dish(dishes, num_dishes)
break
False
option == 'D'
True
delete_dish(dishes, num_dishes)
break
False
option == 'Q'
True
break
False
Display "invalid input"
break
False
option == 'Q'
True
break
True
repeat()
False
Return

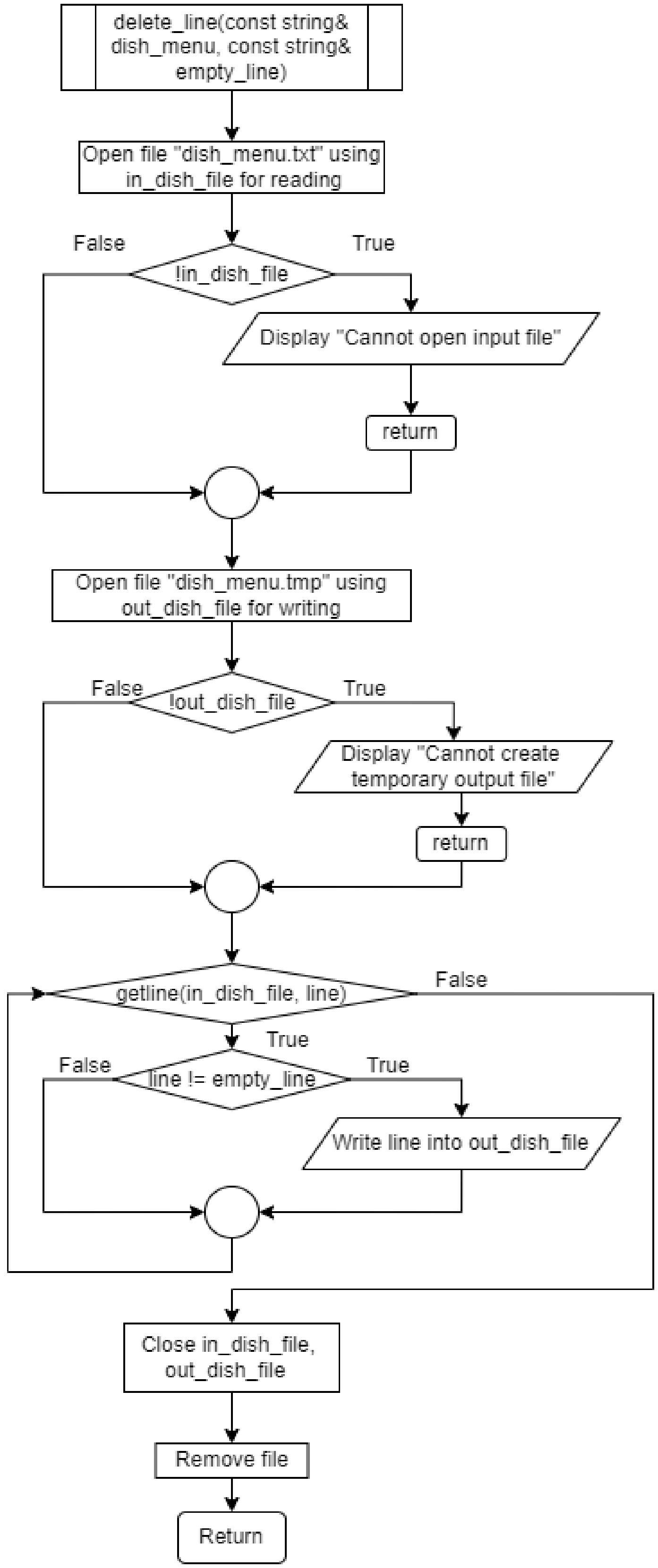
add_dish(DISH_INFO dish[], int rows), operation(), display_menu_heading(), display_all_dishes(DISH_INFO dish[], int rows)


add_dish(DISH_INFO dish[], int rows)
Read dish code
!isalpha(dish[rows].code[0]) || !isdigit(dish[rows].code[1]) || !isdigit(dish[rows].code[2]) || !isdigit(dish[rows].code[3])
False
True
Display "Invalid dish code format"
return
dish[rows].code[0] = toupper(dish[rows].code[0])
(dish[rows].code[0]) != 'A' && (dish[rows].code[0]) != 'B' && (dish[rows].code[0]) != 'C'
False
True
Display "Invalid dish code"
return
i = 0
i < rows
False
True
strcmp(dish[i].code, dish[rows].code) == 0
False
True
Display "Dish code exists"
return
++i
Read dish name
i = 0
i < rows
False
True
strcmp(dish[i].name, dish[rows].name) == 0
False
True
Display "Dish name exists"
return
++i
Read unit price
Open file "dish_menu.txt" using store_dish_menu for writing in append mode
!store_dish_menu
False
True
Write dish code, dish name, dish unit price into store_dish_menu
close store_dish_menu
Display "Cannot open file"
return
operation()
Display option
Read option
option = toupper(option)
return option
display_menu_heading()
Display menu heading
return
display_all_dishes(DISH_INFO dish[], int rows)
i = 0
i < rows
False
True
Display dish[i].code, dish[i].name, dish[i].unit_price
++i
return

repeat(), delete_dish(DISH_INFO dish[], int rows)

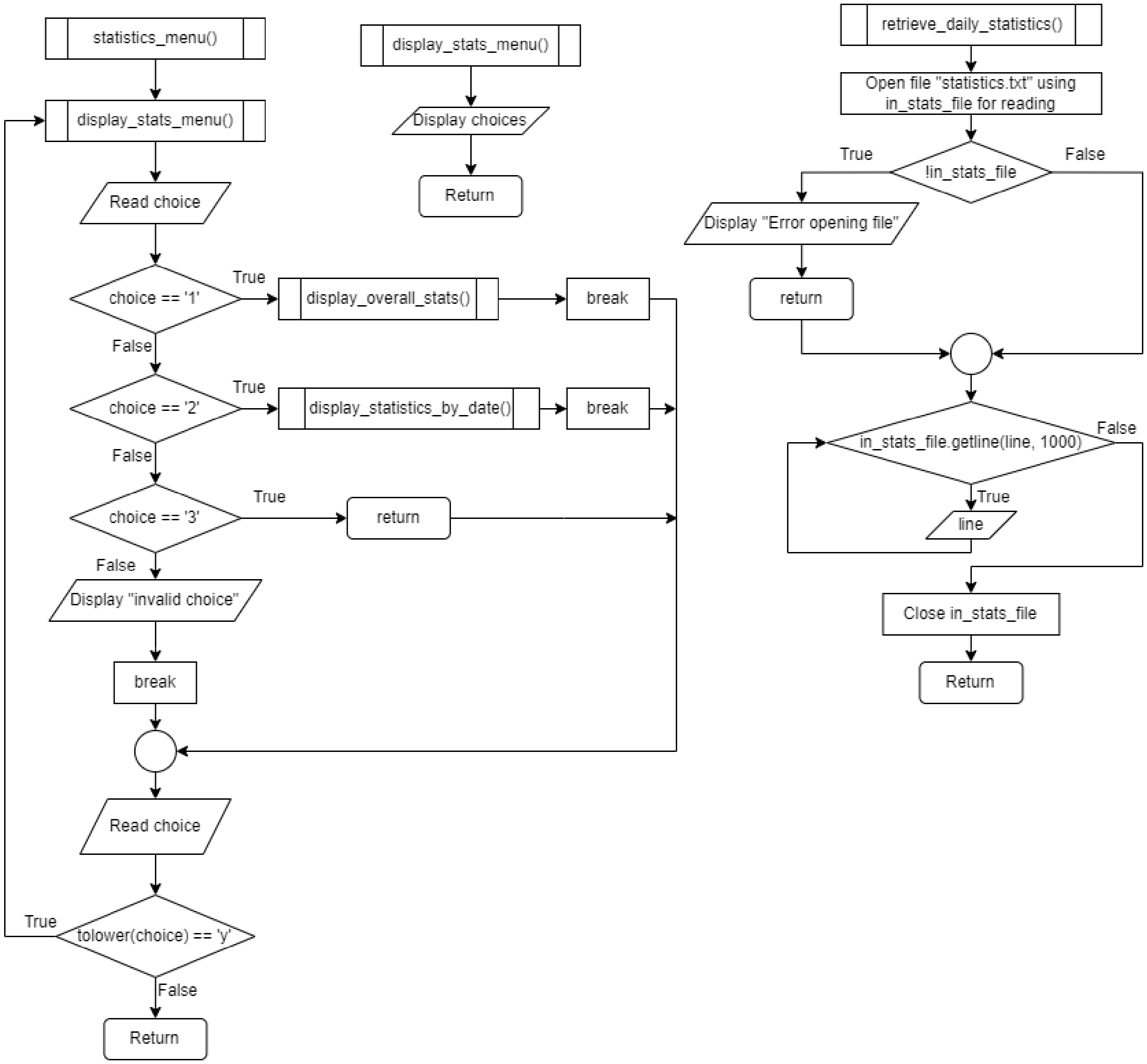

repeat()
Read repeat
repeat = toupper(repeat)
False
repeat != 'N' && repeat != 'Y'
True
Display "Invalid input"
True
repeat != 'N' && repeat != 'Y'
False
return (repeat == 'Y')
delete_dish(DISH_INFO dish[], int rows)
i = 0
exist = false
Read find code
strlen(find_code) != 4 || !isalpha(find_code[0]) || !isdigit(find_code[1]) || !isdigit(find_code[2]) || !isdigit(find_code[3])
False
True
Display "Invalid code format"
return
False
isalpha(find_code[0])
True
find_code[0] = toupper(find_code[0])
i < rows && !exist
False
True
False
strcmp(find_code, dish[i].code) == 0
True
i++
exist = true
False
exist
True
j = i
j < rows - 1
False
True
j++
strcpy_s(dish[j].code, sizeof(dish[j].code), dish[j + 1].code)
strcpy_s(dish[j].name, sizeof(dish[j].name), dish[j + 1].name)
dish[j].unit_price = dish[j + 1].unit_price
rows--
Open file "dish_menu.txt" using store_dish_menu for writing
Display "Dish code does not exist"
False
!store_dish_menu
True
i = 0
Display "Cannot open file"
False
i < rows - 1
True
i++
Write dish code, dish name, dish unit price into store_dish_menu
Close store_dish_menu
Return

delete_line(const string& dish_menu, const string& empty_line)

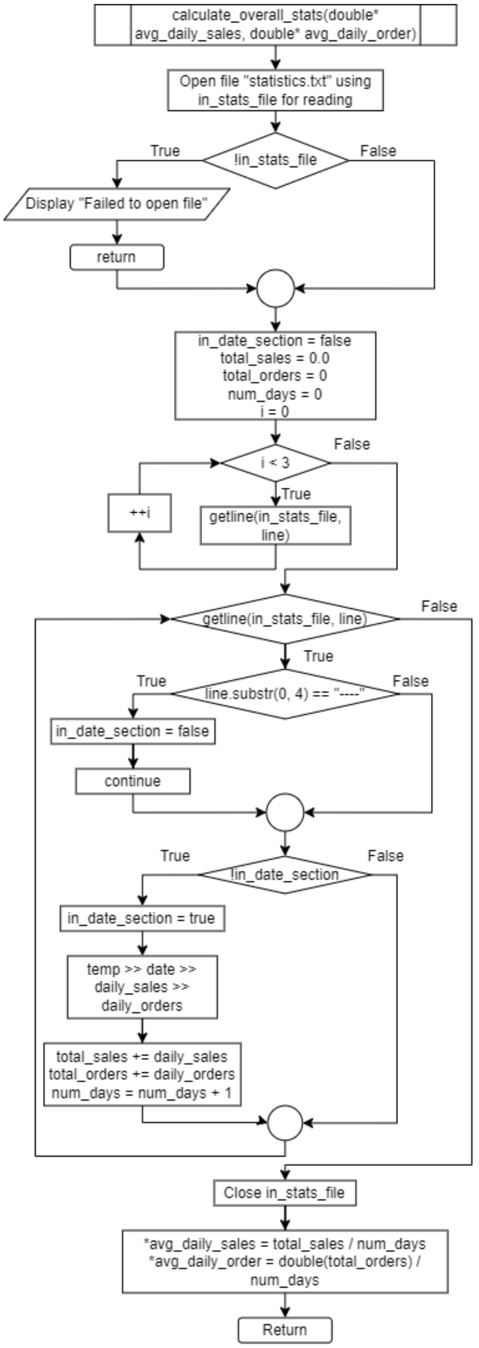

delete_line(const string& dish_menu, const string& empty_line)
Open file "dish_menu.txt" using in_dish_file for reading
False
True
!in_dish_file
Display "Cannot open input file"
return
Open file "dish_menu.tmp" using out_dish_file for writing
False
!out_dish_file
True
Display "Cannot create temporary output file"
return
getline(in_dish_file, line)
False
True
False
line != empty_line
True
Write line into out_dish_file
Close in_dish_file, out_dish_file
Remove file
Return

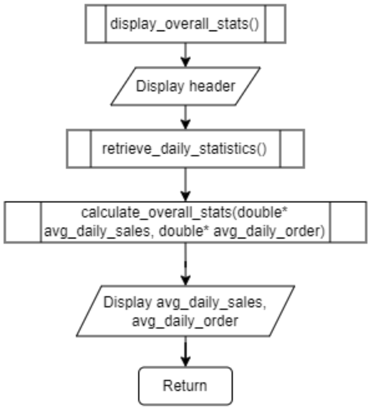
statistics_menu()

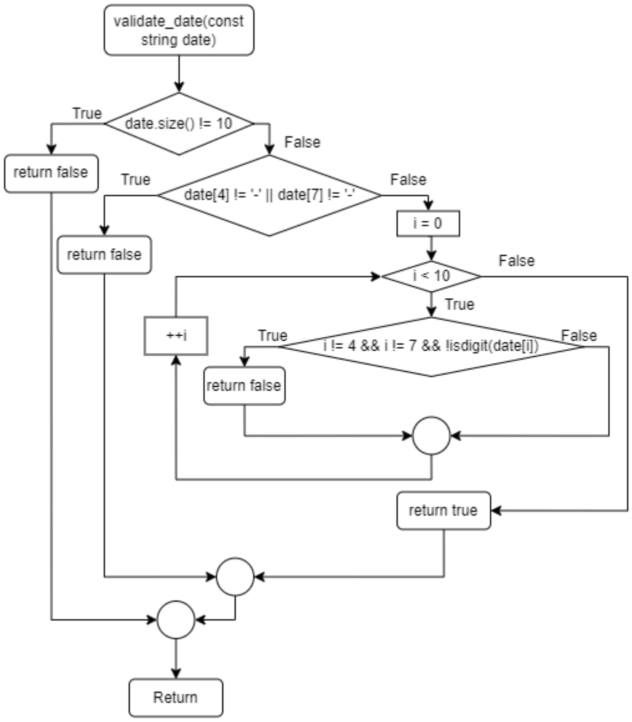
statistics_menu(), display_stats_menu(), retrieve_daily_statistics()



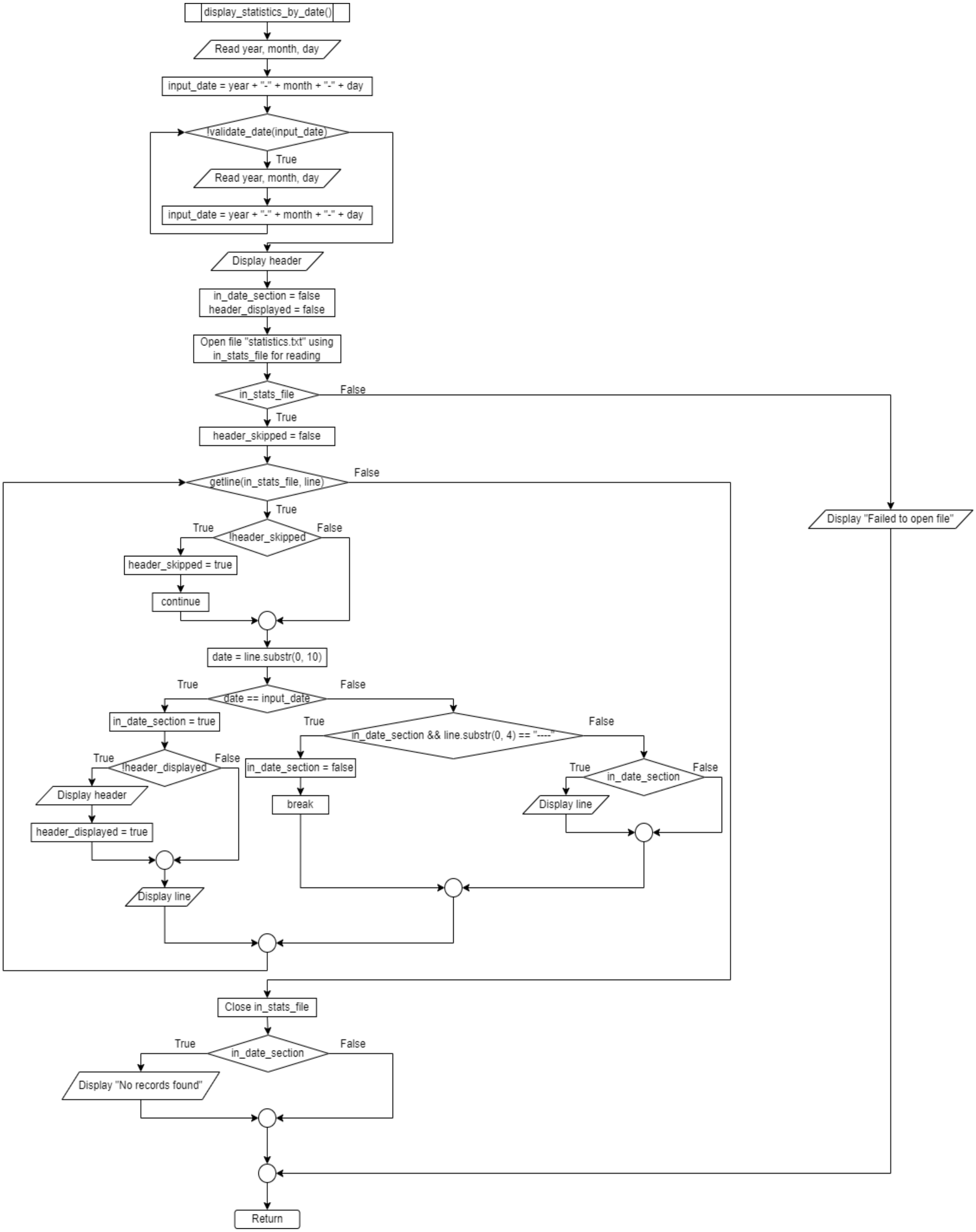
calculate_overall_stats(double*avg_daily_sales,double*avg_daily_order), display_overall_stats(), validate_date(const string date)


calculate_overall_stats(double* avg_daily_sales, double* avg_daily_order)
Open file "statistics.txt" using in_stats_file for reading
!in_stats_file
True
False
Display "Failed to open file"
return
in_date_section = false
total_sales = 0.0
total_orders = 0
num_days = 0
i = 0
i < 3
False
True
++i
getline(in_stats_file, line)
getline(in_stats_file, line)
False
True
line.substr(0, 4) == "----"
True
False
in_date_section = false
continue
!in_date_section
True
False
in_date_section = true
temp >> date >> daily_sales >> daily_orders
total_sales += daily_sales
total_orders += daily_orders
num_days = num_days + 1
Close in_stats_file
*avg_daily_sales = total_sales / num_days
*avg_daily_order = double(total_orders) / num_days
Return


display_overall_stats()
Display header
retrieve_daily_statistics()
calculate_overall_stats(double* avg_daily_sales, double* avg_daily_order)
Display avg_daily_sales, avg_daily_order
Return

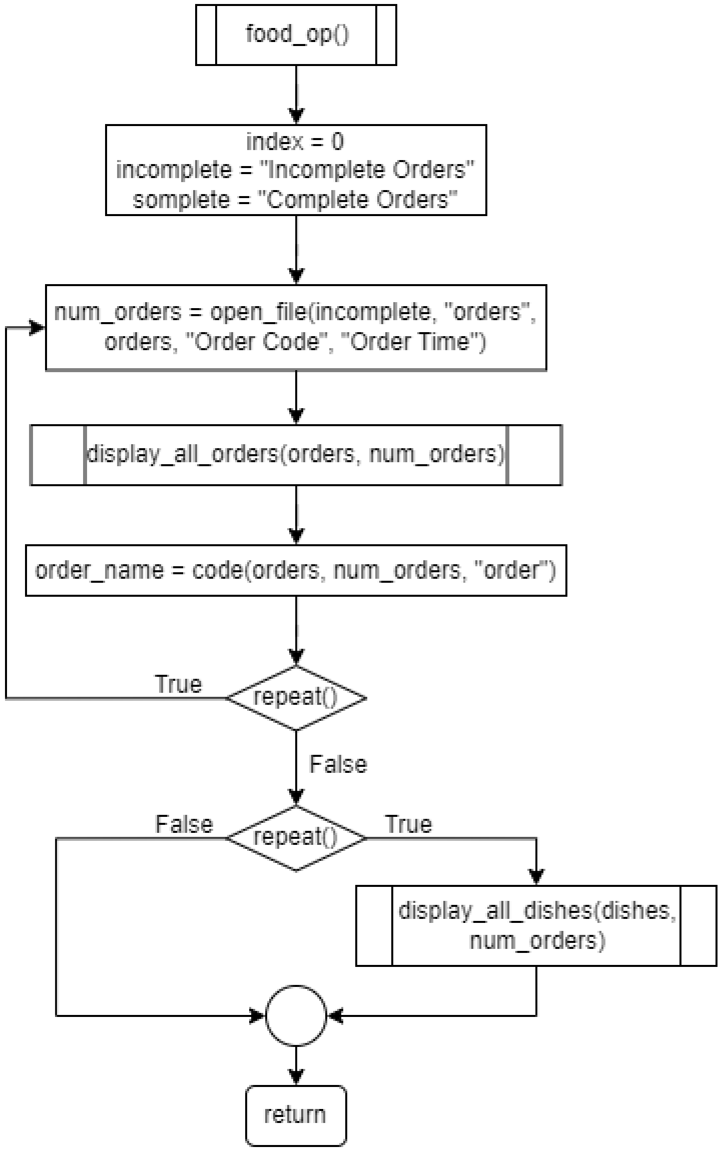


display_statistics_by_date()

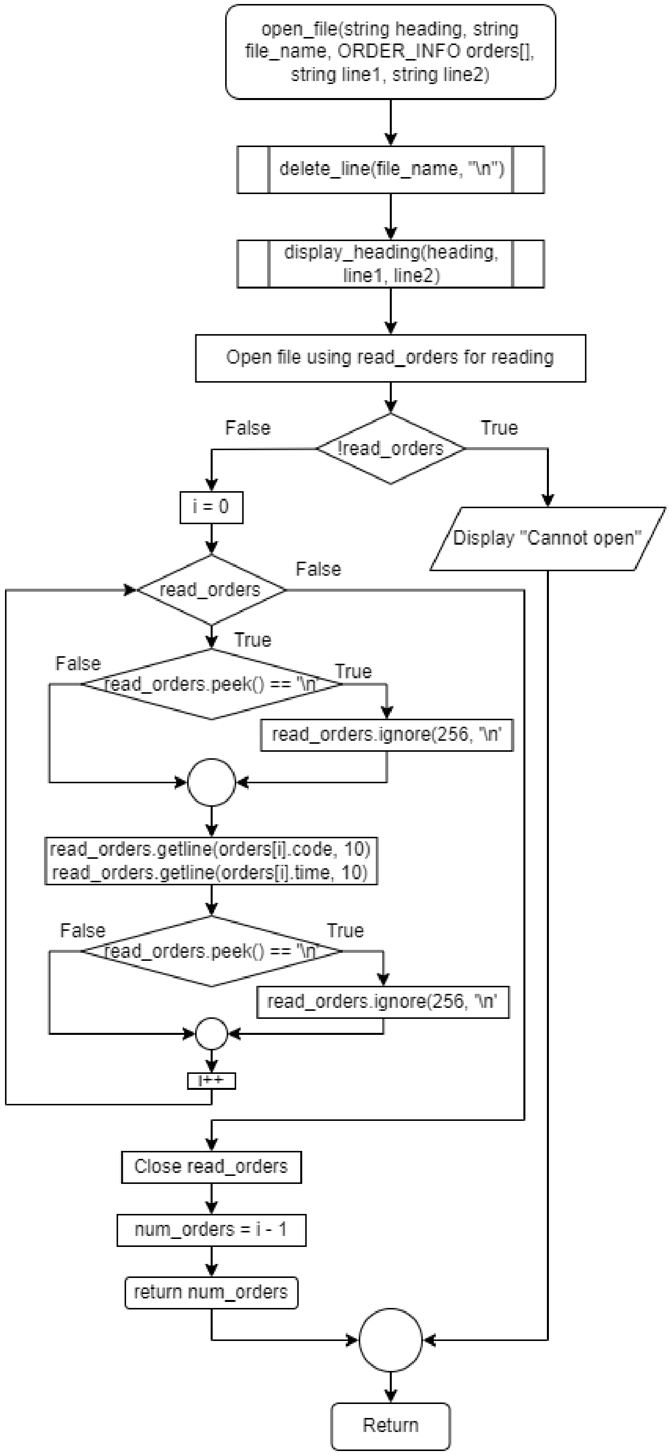


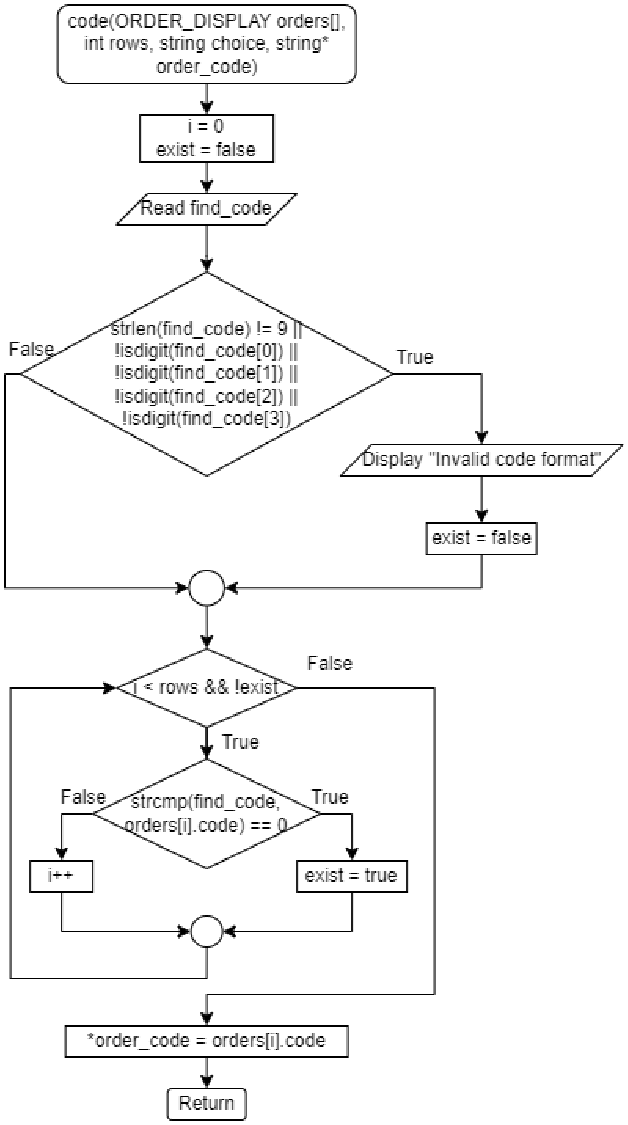
food_op()

food_op()

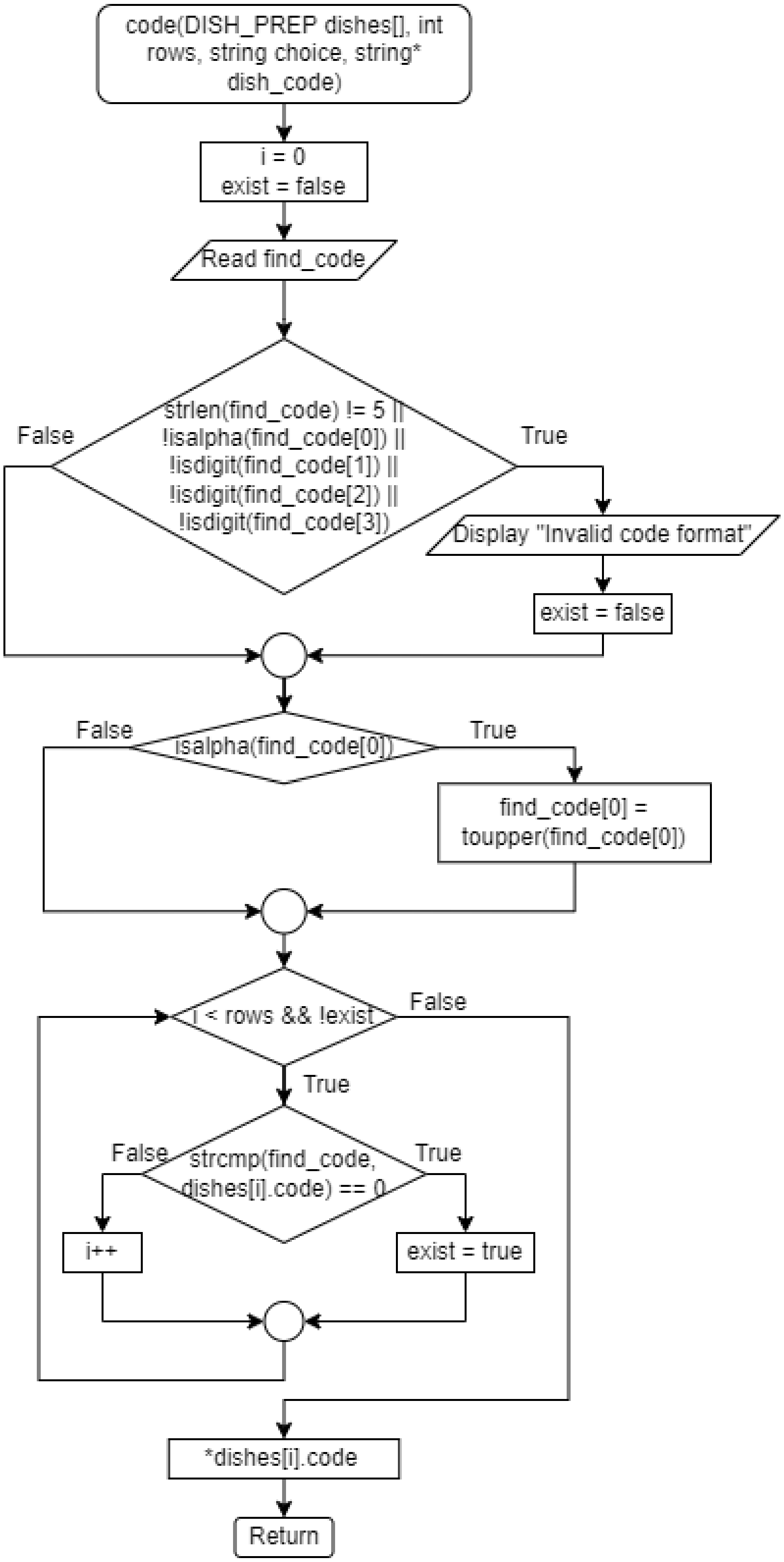

food_op()
index = 0
incomplete = "Incomplete Orders"
somplete = "Complete Orders"
num_orders = open_file(incomplete, "orders", orders, "Order Code", "Order Time")
display_all_orders(orders, num_orders)
order_name = code(orders, num_orders, "order")
True
repeat()
False
False
repeat()
True
display_all_dishes(dishes, num_orders)
return

open_file(string heading, string file_name, ORDER_INFO orders[], string line1, string line2), code(ORDER_DISPLAY orders[], int rows, string choice, string* order_code)

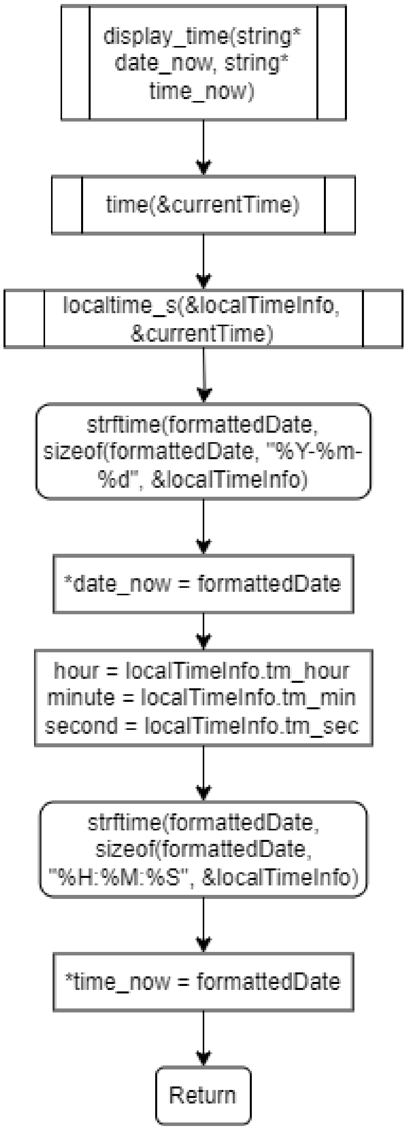

open_file(string heading, string file_name, ORDER_INFO orders[], string line1, string line2)
delete_line(file_name, "\n")
display_heading(heading, line1, line2)
Open file using read_orders for reading
!read_orders
False
True
Display "Cannot open"
i = 0
read_orders
False
True
read_orders.peek() == '\n'
False
True
read_orders.ignore(256, '\n'
read_orders.getline(orders[i].code, 10)
read_orders.getline(orders[i].time, 10)
read_orders.peek() == '\n'
False
True
read_orders.ignore(256, '\n'
i++
Close read_orders
num_orders = i - 1
return num_orders
Return

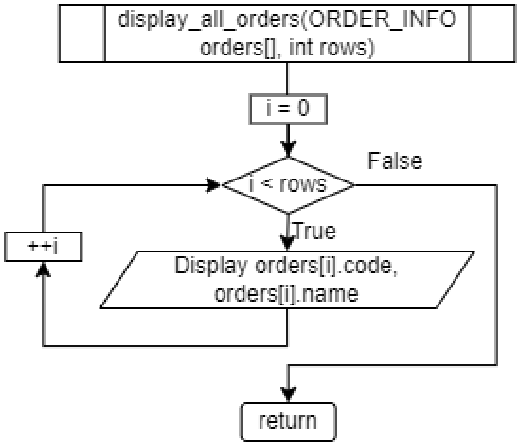

code(ORDER_DISPLAY orders[], int rows, string choice, string* order_code)
i = 0
exist = false
Read find_code
strlen(find_code) != 9 ||
!isdigit(find_code[0]) ||
!isdigit(find_code[1]) ||
!isdigit(find_code[2]) ||
!isdigit(find_code[3])
False
True
Display "Invalid code format"
exist = false
i < rows && !exist
False
True
strcmp(find_code, orders[i].code) == 0
False
True
i++
exist = true
*order_code = orders[i].code
Return

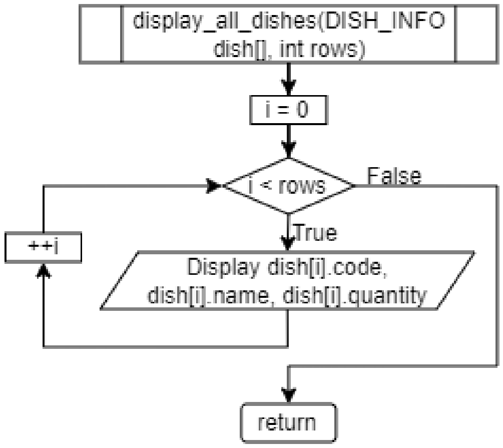
code(DISH_PREP dishes[], int rows, string choice, string* dish_code)

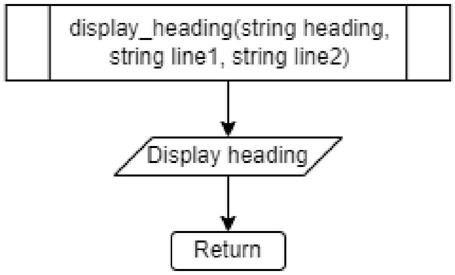

code(DISH_PREP dishes[], int rows, string choice, string* dish_code)
i = 0
exist = false
Read find_code
strlen(find_code) != 5 || !isalpha(find_code[0]) || !isdigit(find_code[1]) || !isdigit(find_code[2]) || !isdigit(find_code[3])
False
True
Display "Invalid code format"
exist = false
isalpha(find_code[0])
False
True
find_code[0] = toupper(find_code[0])
i < rows && !exist
False
True
strcmp(find_code, dishes[i].code) == 0
False
True
i++
exist = true
*dishes[i].code
Return

display_time(string* date_now, string* time_now), display_all_orders(ORDER_INFO orders[], int rows), display_all_dishes(DISH_INFO dish[], int rows), display_heading(string heading, string line1, string line2)

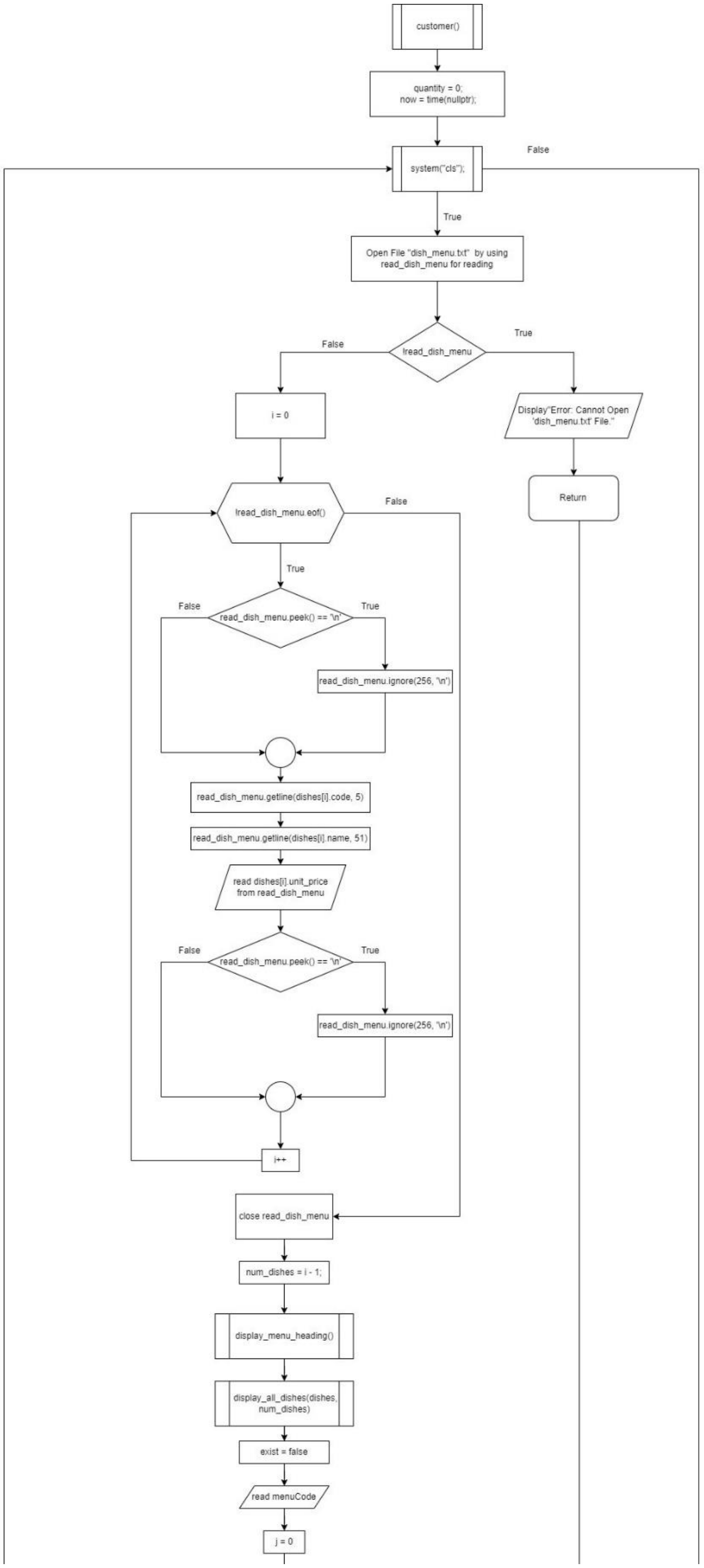

display_time(string* date_now, string* time_now)
time(&currentTime)
localtime_s(&localTimeInfo, &currentTime)
strftime(formattedDate, sizeof(formattedDate, "%Y-%m-%d", &localTimeInfo)
*date_now = formattedDate
hour = localTimeInfo.tm_hour
minute = localTimeInfo.tm_min
second = localTimeInfo.tm_sec
strftime(formattedDate, sizeof(formattedDate, "%H:%M:%S", &localTimeInfo)
*time_now = formattedDate
Return

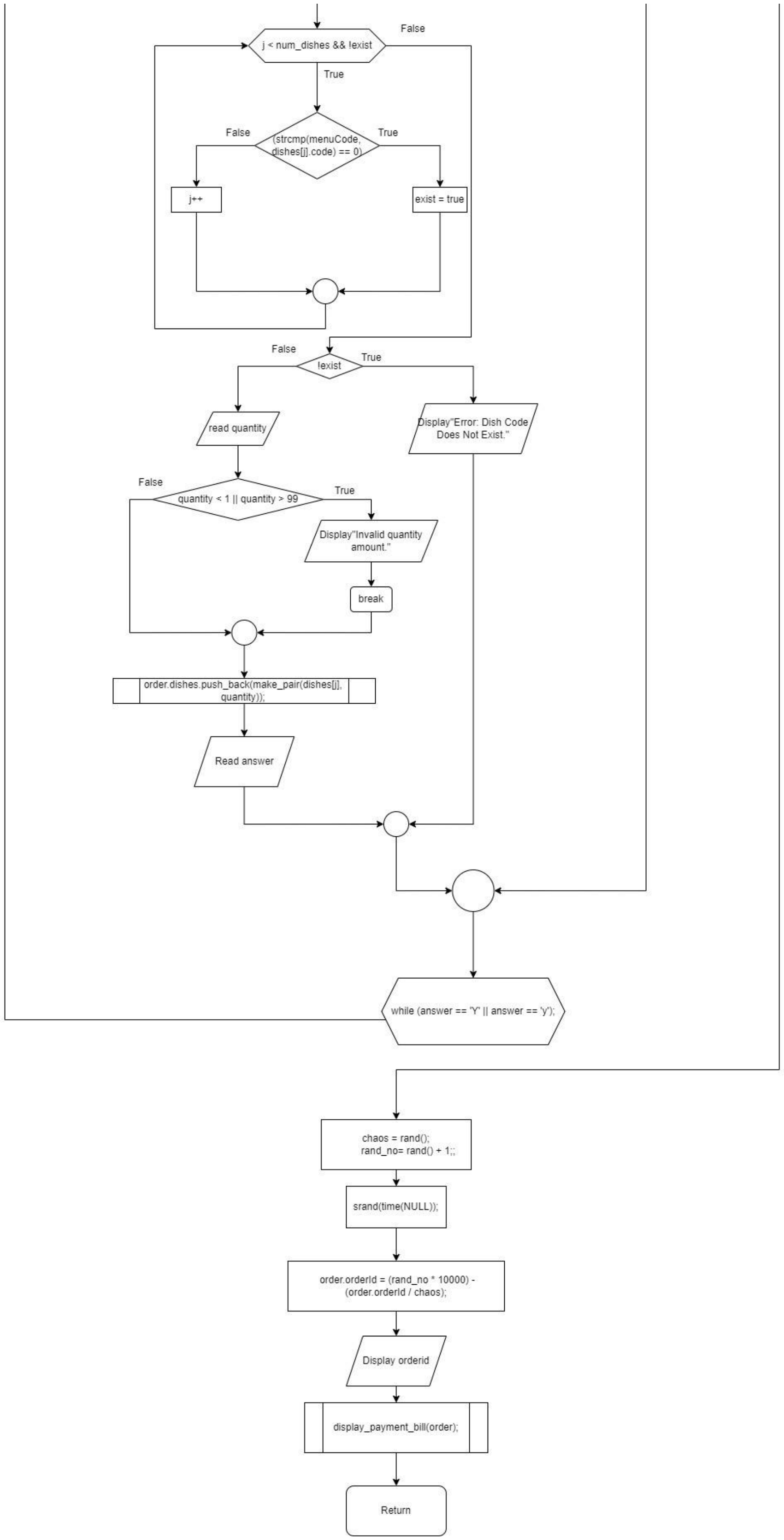

display_all_orders(ORDER_INFO orders[], int rows)
i = 0
i < rows
False
True
++i
Display orders[i].code, orders[i].name
return


display_all_dishes(DISH_INFO dish[], int rows)
i = 0
i < rows
False
True
++i
Display dish[i].code, dish[i].name, dish[i].quantity
return

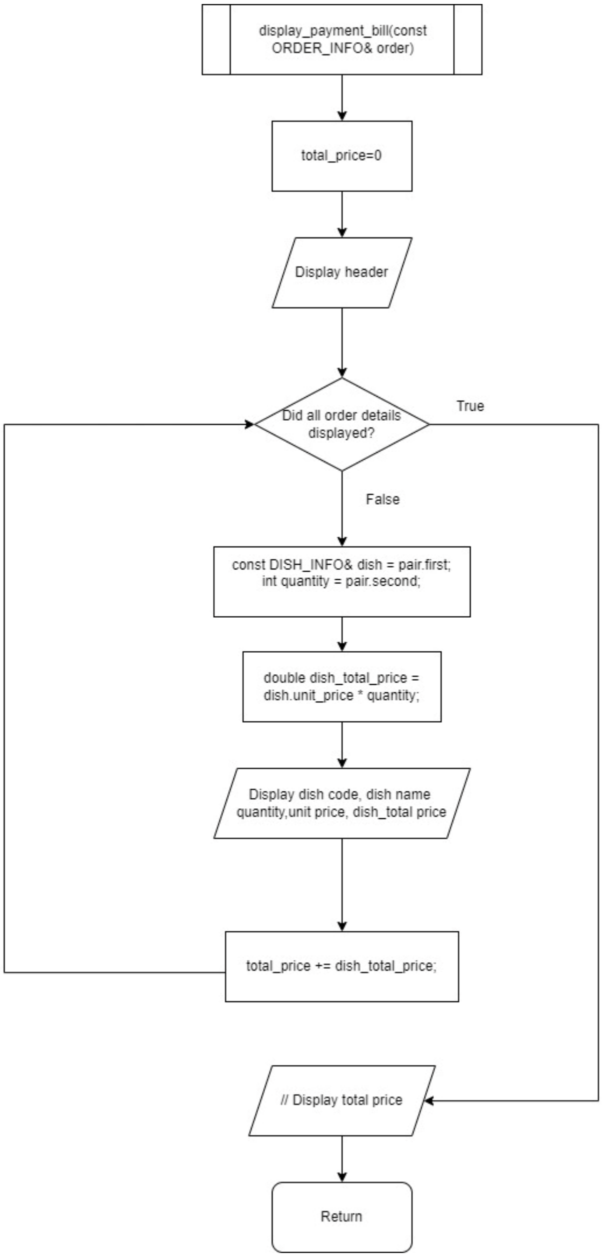


customer()

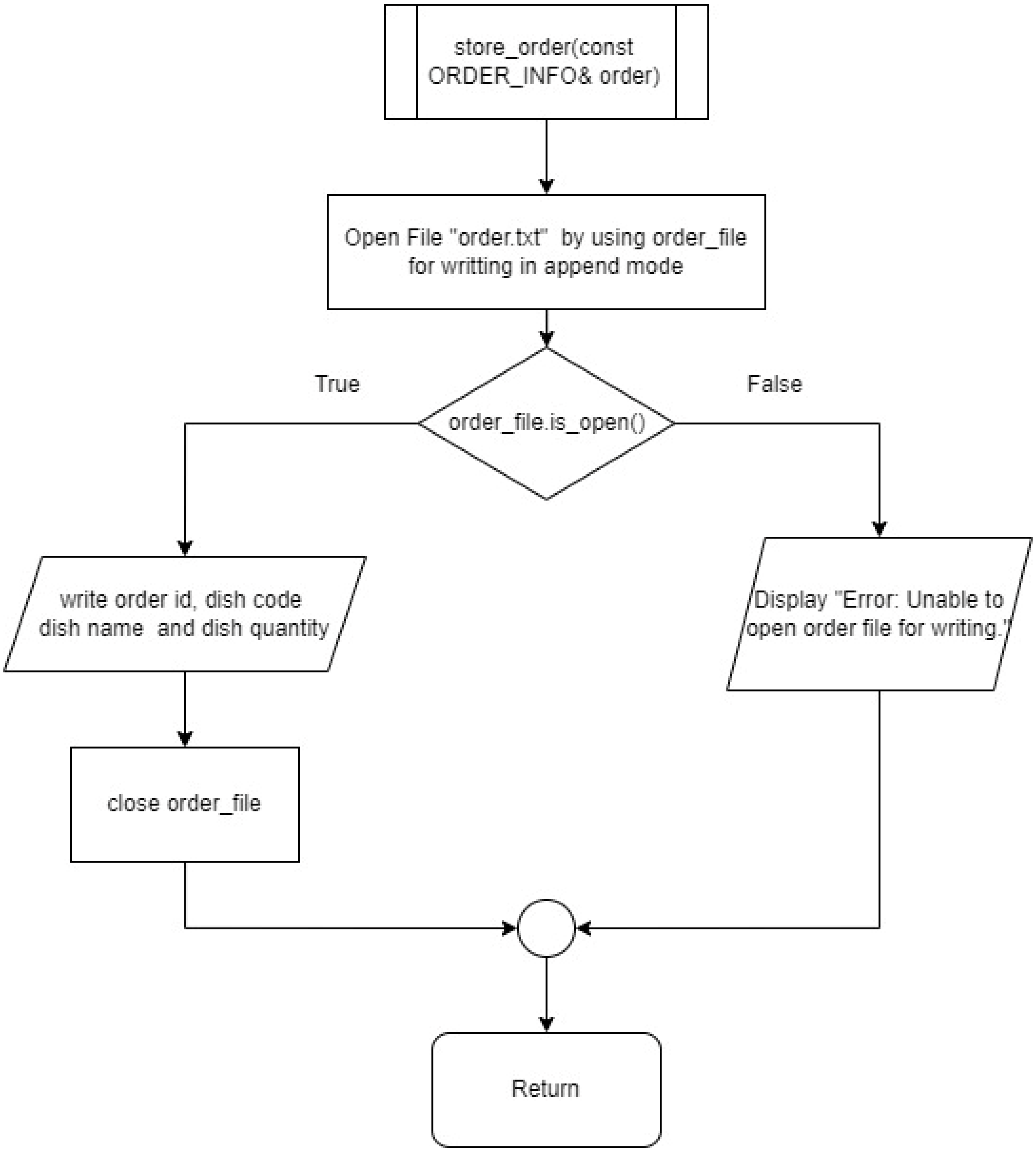

customer()
quantity = 0;
now = time(nullptr);
system("cls");
False
True
Open File "dish_menu.txt" by using read_dish_menu for reading
!read_dish_menu
False
True
i = 0
Display"Error: Cannot Open 'dish_menu.txt' File."
Return
!read_dish_menu.eof()
False
True
read_dish_menu.peek() == '\n'
False
True
read_dish_menu.ignore(256, '\n')
read_dish_menu.getline(dishes[i].code, 5)
read_dish_menu.getline(dishes[i].name, 51)
read dishes[i].unit_price from read_dish_menu
read_dish_menu.peek() == '\n'
False
True
read_dish_menu.ignore(256, '\n')
i++
close read_dish_menu
num_dishes = i - 1;
display_menu_heading()
display_all_dishes(dishes, num_dishes)
exist = false
read menuCode
j = 0


j < num_dishes && !exist
False
True
False
(strcmp(menuCode, dishes[j].code) == 0)
True
j++
exist = true
False
!exist
True
read quantity
Display"Error: Dish Code Does Not Exist."
False
quantity < 1 || quantity > 99
True
Display"Invalid quantity amount."
break
order.dishes.push_back(make_pair(dishes[j], quantity));
Read answer
while (answer == 'Y' || answer == 'y');
chaos = rand();
rand_no= rand() + 1;;
srand(time(NULL));
order.orderId = (rand_no * 10000) - (order.orderId / chaos);
Display orderid
display_payment_bill(order);
Return

display_payment_bill(const ORDER_INFO& order);


display_payment_bill(const ORDER_INFO& order)
total_price=0
Display header
Did all order details displayed?
True
False
const DISH_INFO& dish = pair.first; int quantity = pair.second;
double dish_total_price = dish.unit_price * quantity;
Display dish code, dish name quantity,unit price, dish_total price
total_price += dish_total_price;
// Display total price
Return

store_order(const ORDER_INFO& order);


store_order(const ORDER_INFO& order)
Open File "order.txt" by using order_file for writting in append mode
order_file.is_open()
True
False
write order id, dish code dish name and dish quantity
Display "Error: Unable to open order file for writing."
close order_file
Return

5.0 TABLE OF CONTRIBUTIONS

Student Name	**Contribution**
Darren Kong Yan Ren (2207505)	- Customer Page - Combine each page into Main Menu
Lim Cammy (2206892)	- Food Operator Page
Liow Ke Han (2204872)	- Statistics Page (Manager Page) and flowchart
Wong Yu Chi (2207129)	- Main Menu Page - Edit Menu (Manager Page)