EEE60104 PROGRAMMING TECHNIQUES ASSIGNMENT

NAME: SURRYARAJ A/L POOBALAN

STUDENT ID: 0353127

PROGRAMME: ROBOTIC DESIGN AND DEVELOPMENT

Contents

[Description of Program 1](#_Toc151378051)

[Flowchart 2](#_Toc151378052)

[Program Codes 3](#_Toc151378053)

# Description of Program

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

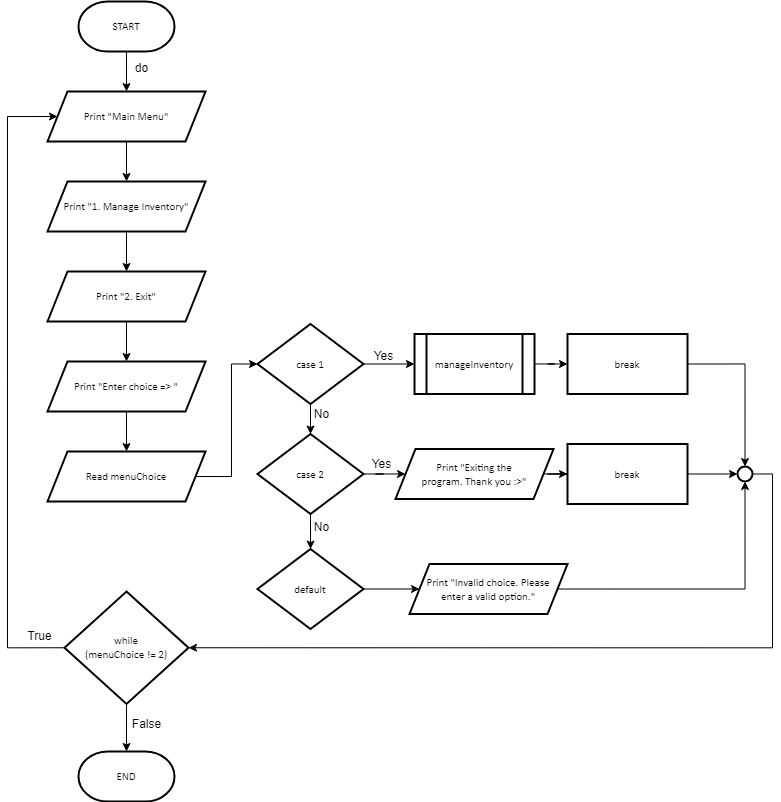
A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

# Flowchart



# Program Codes

A screenshot of a computer program

Description automatically generated

Two header inclusions are used in this program code. First, ‘#include stdio.h’ on line 1. This line includes standard input-output library, which allows this program to use the ‘printf’ and ‘scanf’ functions. Second, ‘#include string.h’ on line 2. This allows for string manipulation functions, for example, string comparison (‘strcmp’) used in this program.

‘#define MAX\_ITEMS 50’ is a macro definition that creates a macro ‘MAX\_ITEMS’ with a set value of 50, the maximum number of items that can be stored in the inventory.

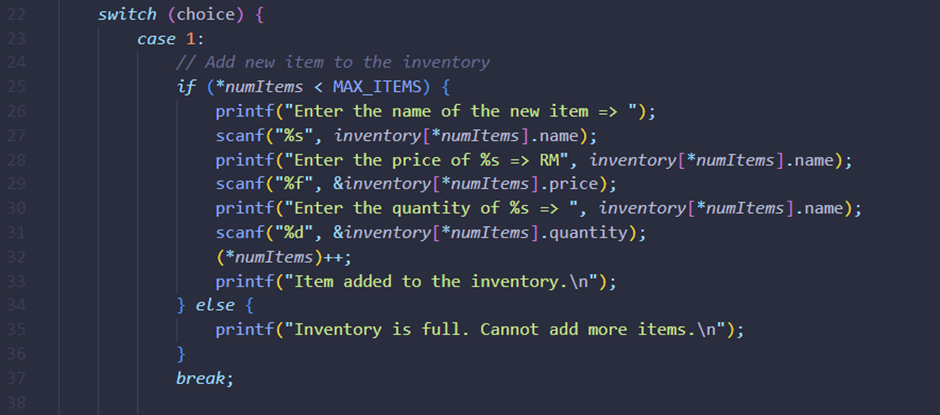
‘struct InventoryItem’ is a structure definition that provides a structure named ‘InventoryItem’ to represent an item in the inventory. It comprises of the essential details of an item in the inventory. This includes the item name, stored in a character array of maximum length of 50 characters (‘char name[50];’), the price of the item defined by float to allow floating numbers or decimals (‘float price;’), and the quantity of item in stock defined by int to accept whole number values only (‘int quantity;’).

A screenshot of a computer program

Description automatically generated

The program has one function named ‘manageInventory’ that takes two arguments. Firstly, ‘inventory[]’, an array representing the inventory of the structure ‘InventoryItem’. Secondly, ‘\*numItems’, a pointer to an integer representing the number of items in the inventory.

The ‘manageInventory()’ function carries out different inventory management tasks according to the user's input. This includes, adding new items, updating item quantities, displaying inventory, and exiting to the main menu page.



After the user has entered an input for ‘manageInventory()’, the switch statement checks the value of ‘choice’ to execute the specific functionality.

Case 1, when 'choice' is 1, the code is called to add a new item to the inventory. It begins by determining if the number of items ('\*numItems') is fewer than the maximum capacity ('MAX\_ITEMS'). If there is space, the user is prompted to enter the new item's name, price, and quantity. The corresponding values in the 'inventory[]' array at the index '\*numItems' are then updated, and '\*numItems' is incremented by one to represent the addition of the new item. Finally, after the item is successfully added to the inventory, a success message is displayed.

If the inventory is full, a notification will appear stating that no more items can be added.

A screen shot of a computer program

Description automatically generated

Case 2, when ‘choice’ is 2, executes code to update the quantity of the existing item. The item names will be saved in the character array 'itemName' for updating. The user is prompted to input the item's name they intend to update. The program iterates over the 'inventory[]' array to locate the provided item name by comparing the input 'itemName' with the names in the inventory using 'strcmp()'. If a match is found, the user is prompted to enter the item's updated quantity, which updates the 'quantity' field of the matched item in the 'inventory[]'. Ends with a quantity update confirmation message.

If the loop completes without locating the item, a message stating that the item was not located in the inventory is displayed.

A screen shot of a computer program

Description automatically generated

Case 3, when ‘choice’ is 3, invokes code to display the current inventory. The program prints the title “Current Inventory:” followed by a header for displaying the current inventory details, which include name, price, and quantity. The ‘inventory[]’ array is then iterated through to print the details of each item. The name, price, and quantity of each item is formatted and displayed in a structured manner.

The format specifiers are used for displaying the header to its item value accordingly. ‘%-20s’ indicates a left-aligned string with a width of 20 characters, ‘%-15s’ a width of 15 characters, and ‘%-10s’ a width of 10 characters accordingly.

‘printf’ is used within the loop to format and print the information of each item into a table format. ‘inventory[i].name’, ‘inventory[i].price’, and ‘inventory[i].quantity’ allow for specific information about each item in the inventory[] array to be obtained.

A screen shot of a computer program

Description automatically generated

Case 4, when ‘choice’ is 4, exits the function, returning to the main menu. A message indicating the exit from the inventory management functionality is displayed.

The default case is executed when the user’s input does not match any of the defined cases in the switch statement. It serves as a way to handle invalid inputs entered by the user and prompts the user to enter a valid option by displaying a message to the user that their input is invalid.

A screenshot of a computer program

Description automatically generated

The main function sets critical variables required for the program to run. It includes the inventory[] array, designed to store InventoryItem structures to hold inventory data. The 'numItems' variable keeps track of the current quantity of items within the inventory, while 'menuChoice' captures the user's menu selection.

A do-while loop is used in the menu display and user input area to constantly present the main menu to users. This menu prompts users to select between managing the inventory or exiting the program by inputting a numerical choice.

In terms of menu options, selecting '1' triggers a call to the 'manageInventory()' function, which handles various inventory management tasks. Choosing '2' prompts the program to display an exit message and terminates the program. The default case guides users encountering invalid inputs, requesting them to input a valid choice.

The loop's termination condition is determined by the user's option, with the menu being visible until the user chooses to stop the program by selecting '2'. When the program ends, the return statement indicates the successful execution of the main() function and the end of the program's execution.