**UNIVERSITY INSTITUTE OF COMPUTING**

**PROJECT REPORT**

**ON RECEIPT**

Program Name: BCA

Subject Name/Code: Data Structures (24CAT-152)

Submitted by: Submitted to:

**Name: Selusiwe Nkosi Name: Mehak Bhatia**

**UID: 24BCA10008 Designation: Assistant Professor**

**Section: BCA – 4A**

**Group: 4**

Table of Contents

[Abstract 3](#_Toc195120889)

[Introduction 3](#_Toc195120890)

[Technique 3](#_Toc195120891)

[System Configuration 4](#_Toc195120892)

[Summary 4](#_Toc195120893)

[Input 5](#_Toc195120894)

[Process 5](#_Toc195120895)

[Diagram 6](#_Toc195120896)

[Mind-Blowing Feature: 7](#_Toc195120897)

[Conclusion 7](#_Toc195120899)

[The code 8](#_Toc195120900)

[Result 10](#_Toc195120901)

# Abstract

This project presents a C-based interactive menu system for a South African-themed restaurant. The program allows users to select options for each of nine courses in a meal, calculates the total price, and provides a seamless user experience. The system demonstrates the application of arrays, constants, and conditional logic in C programming.

# Introduction

Dining experiences are enhanced by providing customers with tailored options. This project leverages Data structures to create an interactive meal selection system for a nine-course South African-themed dining experience. Users can select their desired options for each course, ensuring customization while maintaining an easy-to-use interface.

# Technique

The implementation uses arrays and constants to define meal options and prices. A loop iterates through each course, allowing the user to make a selection. Conditional logic ensures input validation and default choices for invalid inputs. The program dynamically calculates the total cost of the selected items.

The system employs several programming techniques to achieve functionality and maintainability:

1. **Data Representation:**

* Arrays are used to store the course names, meal options, and their respective prices. This enables efficient indexing and retrieval of data during execution.
* Constants are employed to define fixed values, such as the number of courses, ensuring consistency throughout the program.

1. **Input and Validation:**

* User input is captured using scanf for each course, with validation to check whether the input falls within the acceptable range (1-3). If the user provides invalid input, the program defaults to a pre-defined option.

1. **Iterative Processing:**

* A for loop iterates through each course, displaying the available options and prompting the user for a selection. This avoids code repetition and ensures scalability if additional courses or options are added in the future.

1. **Dynamic Calculation:**

* The program maintains a cumulative total price, updated dynamically based on the user’s selections. This is achieved by indexing into the prices array using the user’s choice.

1. **Output Formatting:**

* User-friendly messages are displayed throughout the program to guide the selection process and present the final summary.

1. **Scalability:**

* By using modular arrays and loops, the program is easily extendable. Additional courses or meal options can be added with minimal changes to the core logic.

# System Configuration

1. **Operating System**: Any OS supporting GCC compiler (e.g., Linux, Windows, macOS).
2. **Compiler**: GCC 9.3.0 or higher.
3. **Language**: C Programming Language.
4. **Hardware Requirements**: Minimum 1 GHz CPU, 512 MB RAM, and 50 MB disk space.

# Summary

The program comprises the following components:

1. **Meal Definitions**: Arrays store the course names, meal options, and prices.
2. **Interactive Menu**: A loop prompts the user for selections and validates inputs.
3. **Price Calculation**: Total price is calculated by summing selected options' prices.
4. **Output**: Displays the chosen items and the total price.

# Input

User choices for each course, entered as integers ranging from 1 to 3.

# Process

1. **Display Menu:**

The system sequentially displays the name of each course along with its corresponding meal options and prices. This provides users with a clear view of their choices.

1. **Capture and Validate Input:**

For each course, the user enters their preferred option. Input validation checks the range of the value entered. Invalid inputs are replaced with a default option, and an appropriate message is displayed.

1. **Dynamic Price Calculation:**

As users make their selections, the system dynamically updates the total price by adding the cost of the chosen item from the price array. This calculation occurs in real-time during the loop iteration for each course.

1. **Compile Selection Summary:**

After all inputs are captured, the system compiles a detailed summary of the user’s choices, including the names and prices of selected items.

1. **Present Final Output:**

The program concludes by displaying a formatted summary table containing the chosen meals and their total cost, ensuring clarity and user satisfaction.

# Diagram

|  |  |
| --- | --- |
| **South African Themed Restaurant** | |
| **Course** | **Selected Options** |
| **Aperitif** | **Wine** |
| **Soup** | **Tomato soup** |
| **Starter** | **Salad** |
| **Fish Course** | **Grilled salmon** |
| **Main Dish** | **Steak** |
| **Salad** | **Caesar salad** |
| **Cheese Course** | **Brie** |
| **Dessert** | **Chocolate cake** |
| **Digestive** | **Brandy** |
| **Total Price: $\_\_.\_\_** | |

# Mind-Blowing Feature

## **Intelligent Pairing Suggestion**

Incorporate an additional feature where the system intelligently recommends options based on the user’s previous choices. For example, if the user selects "Steak" as the main dish, the program could suggest "Red Wine" for the aperitif and "Brandy" for the digestive to enhance the culinary experience.

# Conclusion

This project showcases the practicality and flexibility of C programming for creating interactive applications. The program’s structure ensures maintainability and scalability, making it adaptable to various use cases beyond restaurant menus. Future enhancements could include graphical interfaces, additional input validations, expanded menu options, and intelligent pairing recommendations for an enriched user experience.

# The code

#include <stdio.h>

#define NUM\_COURSES 9

// Define meals and prices

const char \*courses[NUM\_COURSES] = {

"Apéritif", "Soup", "Starter (Entrée)", "Fish Course",

"Main Dish", "Salad", "Cheese Course", "Dessert", "Digestif"

};

const char \*meal\_options[NUM\_COURSES][3] = {

{"Wine - $5", "Whiskey - $7", "Juice - $3"},

{"Tomato Soup - $4", "Mushroom Soup - $5", "Chicken Soup - $6"},

{"Salad - $5", "Pâté - $6", "Spring Rolls - $4"},

{"Grilled Salmon - $10", "Fish Curry - $9", "Prawns - $12"},

{"Steak - $15", "Chicken Roast - $12", "Veg Lasagna - $10"},

{"Greek Salad - $5", "Caesar Salad - $6", "Fruit Salad - $4"},

{"Cheddar - $6", "Brie - $7", "Blue Cheese - $8"},

{"Chocolate Cake - $7", "Ice Cream - $5", "Fruit Tart - $6"},

{"Brandy - $8", "Liqueur - $7", "Herbal Tea - $4"}

};

int prices[NUM\_COURSES][3] = {

{5, 7, 3},

{4, 5, 6},

{5, 6, 4},

{10, 9, 12},

{15, 12, 10},

{5, 6, 4},

{6, 7, 8},

{7, 5, 6},

{8, 7, 4}

};

int main() {

int choice;

int total\_price = 0;

printf("Welcome to the South African Themed Restaurant!\n");

printf("Plan your meal by selecting an option for each course:\n");

for (int i = 0; i < NUM\_COURSES; i++) {

printf("\n%s:\n", courses[i]);

for (int j = 0; j < 3; j++) {

printf("%d) %s\n", j + 1, meal\_options[i][j]);

}

printf("Enter your choice (1-3): ");

scanf("%d", &choice);

if (choice < 1 || choice > 3) {

printf("Invalid choice. Defaulting to option 1.\n");

choice = 1;

}

printf("You chose: %s\n", meal\_options[i][choice - 1]);

total\_price += prices[i][choice - 1];

}

printf("\nYour meal is ready! Total price: $%d\n", total\_price);

return 0;

}

# Result

