Malnad College of Engineering

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

Hassan, Karnataka, India– 573202

Course Title: Data Structures

Course Code: 23AI304

Submitted by:

Rishitha Raghavan HV	4MC23CI045
Priyadarshini M	4MC23CI039
Saniya Banu	4MC23CI048
Keerthi	4MC23CI064

Submitted to:

Dr. Balaji Prabhu BV

Associate Professor and HOD

Dept. of CSE (AI and ML)

MCE, Hassan

Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning

CONTENT:

4. Input And Output

5. Final Analysis

1. Data Structures

1.1 Array Of Structures

A simple, contiguous block of memory storing multiple elements of the same type (Business in this case).

1.2 Fixed Size

The size of the array is defined by MAX (which is 10 in your code), meaning it can hold up to 10 Business structures.

1.3 Indexing

Access to the elements is done via an index, which is an integer value starting from 0 up to MAX-1

2. Steps followed in the program

2.1 Define the Structure and Constants

Business structure is defined with three members: name, category, and phone .A constant MAX is defined to set the maximum number of businesses the program can store.

2.2 Declare Global Variables businesses[MAX]:

An array of Business structures to hold the list of businesses.

Business Count: An integer variable to keep track of the number of businesses added to the list.

2.3 Define add Business Function

This function takes name, category, and phone as arguments.

It checks if business Count is less than MAX.

If true, it copies the provided details into the next available index in the businesses array.

Increments the business Count.

2.4 Define lookup By Category Function

This function takes a category string as an argument.

It iterates through the businesses array.

If the category matches, it prints the corresponding name and phone.

2.5 Define look up By Name Function

`This function takes a name string as an argument.

It iterates through the businesses array.

If the name matches, it prints the name, category, and phone of the business and returns.

If no match is found, it prints "Business not found."

3. Pseudo Code

```
DEFINE MAX as 10
STRUCT Business
   STRING name
   STRING category
   STRING phone
END STRUCT
DECLARE ARRAY businesses[MAX] OF Business
DECLARE INTEGER businessCount = 0
FUNCTION addBusiness(name, category, phone)
   IF businessCount < MAX THEN
      SET businesses[businessCount].name TO name
      SET businesses[businessCount].category TO category
      SET businesses[businessCount].phone TO phone
```

```
        INCREMENT businessCount
    ELSE
        PRINT "Cannot add more businesses."
    END IF
END FUNCTION

FUNCTION lookupByCategory(category)
    FOR i FROM 0 TO businessCount - 1 DO
        IF businesses[i].category == category THEN
            PRINT "Name: " + businesses[i].name + ", Phone: " + businesses[i].phone
        END IF
    END FOR
END FUNCTION
FUNCTION lookupByName(name)
    FOR i FROM 0 TO businessCount - 1 DO
        IF businesses[i].name == name THEN
            PRINT "Found: Name: " + businesses[i].name +
                ", Category: " + businesses[i].category +
                ", Phone: " + businesses[i].phone
            RETURN
        END IF
    END FOR
    PRINT "Business not found."
END FUNCTION

FUNCTION main()
```

```
    CALL addBusiness("TechCorp", "Technology", "1234567890")
    CALL addBusiness("Foodies", "Restaurant", "9876543210")

    PRINT "Businesses in category 'Technology':"
    CALL lookupByCategory("Technology")

    PRINT "Lookup by name 'Foodies':"
    CALL lookupByName("Foodies")
END FUNCTION
CALL main()
```

4. Inputs and Outputs for the Program

Inputs:

4.1 Adding Businesses

The program internally adds businesses using the addBusiness function. In this case:

AddBusiness("TechCorp", "Technology", "1234567890")

AddBusiness("Foodies", "Restaurant", "9876543210")

4.2 Searching by Category

The category to search for businesses: "Technology"

4.3 Searching by Name

The name to search for: "Foodies”

Outputs:

4.4 After Searching by Category

For the category "Technology", the output will list businesses in that category.

Output:

Businesses in category 'Technology':

Name: TechCorp, Phone: 1234567890

4.5 After Searching by Name:

For the name "Foodies", the output will display the details of the matching business.

Output:

Lookup by name 'Foodies':

Found: Name: Foodies, Category: Restaurant, Phone: 9876543210

4.6. If no match is found

If you search for a name or category that doesn't exist, it will print an appropriate message.

Example Output (if searching for a non-existent category "Retail"):

Businesses in category 'Retail':

Example Output (if searching for a non-existent name "NonExistent"):

Lookup by name 'NonExistent':

Business not found.

~These outputs are printed based on the search results for the specified category or name.

5. Final Analysis of the Program

This program provides basic functionality for managing a list of businesses and searching them by category or name. Here's a comprehensive analysis:

5.1 Simple Data Management:

The program uses a straightforward array-based approach to store business details.

Each business is represented by a structured format, making data handling clear and organized.

5.2 Basic Functionality:

The addBusiness function n allows adding new businesses until the maximum limit is reached.

The lookupByCategory and lookupByName functions provide basic search capabilities.

5.3 Ease of Use:

The program is simple to use with clearly defined functions for adding and searching businesses. Outputs are straightforward and easy to understand, displaying relevant information or appropriate messages when searches are unsuccessful.

6. Limitations

6.1 Fixed Size Array

The array businesses has a fixed size of MAX (10), which limits the number of businesses that can be stored .If more businesses need to be added, the program will need to either increase MAX or adopt dynamic memory allocation.

6.2 Linear Search

Both lookup By Category and look up ByName use linear search, which can be inefficient for a large number of businesses .As the list grows, search operations may become slower.

6.3 No Error Handling for Input Strings

There is no validation for input strings (e.g., checking if they are too long for the allocated fields or if they are null).

6.4 Limited Functionality

The program only supports adding and searching businesses.

There is no functionality for updating or deleting a business, nor for listing all businesses.

7. Potential Enhancements

7.1 Dynamic Memory Allocation:

Using dynamic memory (e.g., malloc in C) can allow the list of businesses to grow as needed, rather than being limited by a fixed size.

7.2 Enhanced Search Techniques:

Implementing more efficient search algorithms (e.g., binary search) or using data structures like hash tables could improve the search performance.

7.3 Additional Features:

Add functions to update or delete businesses. Provide a function to list all businesses, optionally sorted by name or category.

7.4 Input Validation:

Include input validation to ensure that names, categories, and phone numbers fit within their allocated space and meet expected formats.

Conclusion

This program is a good starting point for managing a small list of businesses. It demonstrates basic array operations, string handling, and conditional logic in C. However, to make it more robust and scalable, enhancements like dynamic memory allocation, more efficient searching, and additional features could be implemented.