To design and implement a C program that presents a user-friendly menu, enabling the selection and execution of various Data Structures and Algorithms (DSA) operations.

Certainly, let's craft a C program to showcase a menu-driven interface that incorporates some core DSA concepts for demonstration.

Aim:

- To design and implement a C program that presents a user-friendly menu, enabling the selection and execution of various Data Structures and Algorithms (DSA) operations.

Theory:

- A menu-driven program enhances user interaction by offering a list of choices, facilitating intuitive navigation and operation selection.
- DSA provides the foundational tools for organizing and manipulating data efficiently within the program. We'll exemplify this using a Stack as our chosen data structure.

Procedure:

1. **Header Inclusion**: Include the necessary header files:
   - `stdio.h` for input/output operations
   - `stdlib.h` for general utility functions (e.g., `exit`)
2. **Stack Implementation**: Define a structure or utilize an array to represent the Stack data structure along with essential operations:
   - `push`: Adds an element to the top of the stack
   - `pop`: Removes and returns the top element from the stack
   - `display`: Prints the current contents of the stack
3. **Menu Creation**: Design the menu using a `do-while` loop:
   - Present the user with numbered options (e.g., 1. Push, 2. Pop, 3. Display, 4. Exit)
   - Prompt the user to input their choice.
   - Use a `switch` statement to execute the corresponding operation based on the user's input.
4. **User Interaction**: Within each case of the `switch` statement:
   - For `push`: Get an element from the user and add it to the stack.
   - For `pop`: Remove the top element if the stack is not empty.
   - For `display`: Print the stack contents.
   - For `exit`: Terminate the program.

- Handle invalid choices gracefully with an error message.

Code:

C

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE 100

int stack[MAX_SIZE];
int top = -1;

void push(int item) {
    if (top >= MAX_SIZE - 1) {
        printf("Error:    Stack Overflow\n");
    } else {
        stack[++top] = item;
        printf("%d pushed to stack\n", item);
    }
}

int pop() {
    if (top < 0) {
        printf("Error: Stack Underflow\n");
        return -1; // Or handle the error in a suitable way
    } else {
        int item = stack[top--];
        printf("%d popped from stack\n", item);
        return item;
    }
}

void display() {
    if (top < 0) {
        printf("Stack is empty\n");
    } else {
        printf("Stack elements are:\n");
        for (int i = top; i >= 0; i--) {
            printf("%d\n", stack[i]);
        }
    }
}
```

```
int main() {
    int choice, item;

    do {
        printf("\nMenu:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch    (choice) {
            case 1:
                printf("Enter item to push: ");
                scanf("%d", &item);
                push(item);
                break;
            case    2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice\n");
        }

    } while (choice != 4);

    return    0;
}
```

Learning Outcomes:

- **Menu-Driven Interface**: Understanding how to create user-friendly interfaces for interacting with DSA operations.
- **Stack Operations**: Practical implementation of stack functionalities like `push`, `pop`, and `display`.

- **Error Handling**: Learning how to handle potential errors such as stack overflow and underflow gracefully.
- **Modular Design**: Appreciation for organizing code into functions for better maintainability and reusability.

Feel free to ask if you'd like to explore other DSA concepts or enhance this menu-driven program further!