CUSTOMER FEEDBACK SYSTEM

RUBEN ROBY
COMPUTER SCIENCE DEPT.
ROLL NUMBER: 64

17/07/2024

INTRODUCTION

THE CUSTOMER FEEDBACK SYSTEM IS DESIGNED TO ALLOW CUSTOMERS TO PROVIDE FEEDBACK ON A SERVICE OR PRODUCT. IT PROVIDES FUNCTIONALITY TO COLLECT, STORE, AND DISPLAY CUSTOMER FEEDBACK, INCLUDING RATINGS AND COMMENTS.

"Developing a system for collecting and managing customer feedback, including surveys and rating."
is the problem statement.

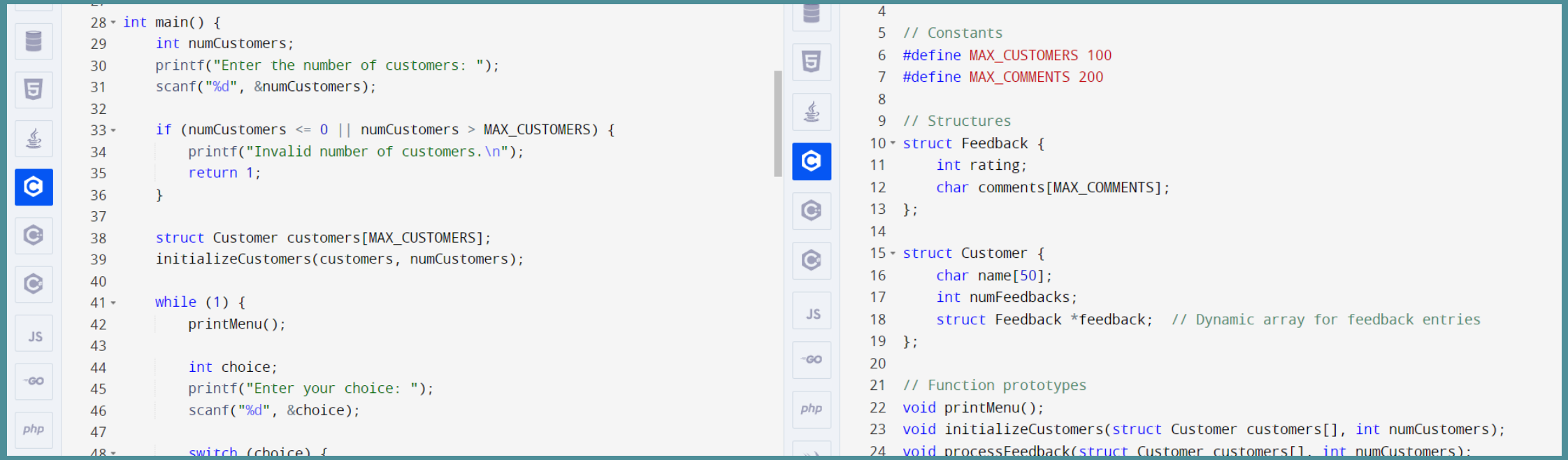
Our objective was to design and make a feedback system which will store the feedbacks but also allow to review it later by the customer.

SYSTEM REQUIREMENT

- **Processor (CPU):**
- Any modern processor should suffice, would handle the computational demands of this project well.
- **Memory (RAM):**
- The amount of memory is more than sufficient for processing typical data inputs and outputs in this type of application.
- **Operating System:**
- Windows, macOS, or Linux.

DEVELOPMENT

```
START

Define constants MAX_CUSTOMERS and MAX_COMMENTS

Define structure Feedback
    Integer rating
    String comments[MAX_COMMENTS]

Define structure Customer
    String name[50]
    Integer numFeedbacks
    Pointer to Feedback feedback

Function printMenu
    Print "1. Provide Feedback"
    Print "2. View Customer Feedback"
    Print "3. Exit"

Function initializeCustomers(customers, numCustomers)
    For i from 0 to numCustomers - 1
        Print "Enter customer i's name: "
        Read customers[i].name
        Set customers[i].numFeedbacks to 0
        Set customers[i].feedback to NULL

Function processFeedback(customers, numCustomers)
    Print "Enter your customer number (1-numCustomers): "
    Read customerId
    Decrement customerId by 1 (convert to zero-indexed)

    If customerId is invalid
        Print "Invalid customer number."
        Return

    Increment customers[customerId].numFeedbacks
    Allocate memory for new feedback entry and update customers[customerId].feedback

    Print "Rate us (1-5): "
    Read customers[customerId].feedback[last feedback index].rating
    Print "Enter your comments: "
    Read customers[customerId].feedback[last feedback index].comments

    Print "Thank you for your feedback!"

Function printCustomerFeedback(customers, numCustomers)
    For i from 0 to numCustomers - 1
        Print "Customer i: customers[i].name"
        If customers[i].numFeedbacks is 0
            Print "   No feedback given yet."
        Else
            For j from 0 to customers[i].numFeedbacks - 1
                Print "   - Rating: customers[i].feedback[j].rating"
                Print "     Comments: customers[i].feedback[j].comments"

Function freeMemory(customers, numCustomers)
    For i from 0 to numCustomers - 1
        Free memory allocated for customers[i].feedback

Main function
    Print "Enter the number of customers: "
    Read numCustomers

    If numCustomers is invalid
        Print "Invalid number of customers."
        Exit program

    Call initializeCustomers(customers, numCustomers)

    While true
        Call printMenu
        Print "Enter your choice: "
        Read choice

        If choice is 1
            Call processFeedback(customers, numCustomers)
        Else if choice is 2
            Call printCustomerFeedback(customers, numCustomers)
        Else if choice is 3
            Call freeMemory(customers, numCustomers)
            Print "Exiting program."
            Exit program
        Else
            Print "Invalid choice. Please enter again."

END
```



```c
int main() {
    int numCustomers;
    printf("Enter the number of customers: ");
    scanf("%d", &numCustomers);

    if (numCustomers <= 0 || numCustomers > MAX_CUSTOMERS) {
        printf("Invalid number of customers.\n");
        return 1;
    }

    struct Customer customers[MAX_CUSTOMERS];
    initializeCustomers(customers, numCustomers);

    while (1) {
        printMenu();

        int choice;
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
```

```c

// Constants
#define MAX_CUSTOMERS 100
#define MAX_COMMENTS 200

// Structures
struct Feedback {
    int rating;
    char comments[MAX_COMMENTS];
};

struct Customer {
    char name[50];
    int numFeedbacks;
    struct Feedback *feedback;  // Dynamic array for feedback entries
};

// Function prototypes
void printMenu();
void initializeCustomers(struct Customer customers[], int numCustomers);
void processFeedback(struct Customer customers[], int numCustomers);
```

PROJECT CODE

```c
            break;
        case 2:
            printCustomerFeedback(customers, numCustomers);
            break;
        case 3:
            freeMemory(customers, numCustomers);
            printf("Exiting program.\n");
            return 0;
        default:
            printf("Invalid choice. Please enter again.\n");
            break;
    }
  }

  return 0;
}

void printMenu() {
  printf("\n===== Customer Feedback System =====\n");
  printf("1. Provide Feedback\n");
  printf("2. View Customer Feedback\n");
  printf("3. Exit\n");
}

void initializeCustomers(struct Customer customers[], int numCustomers) {
```

```c
  // Initialize customers (for demonstration purposes)
  for (int i = 0; i < numCustomers; ++i) {
      printf("Enter customer %d's name: ", i + 1);
      scanf("%s", customers[i].name);
      customers[i].numFeedbacks = 0;
      customers[i].feedback = NULL;  // Initialize feedback pointer to NULL
  }
}

void processFeedback(struct Customer customers[], int numCustomers) {
  int customerId;
  printf("Enter your customer number (1-%d): ", numCustomers);
  scanf("%d", &customerId);
  customerId--;  // Convert to zero-indexed

  if (customerId < 0 || customerId >= numCustomers) {
      printf("Invalid customer number.\n");
      return;
  }

  // Increment number of feedbacks
  customers[customerId].numFeedbacks++;

  // Allocate memory for feedback entries
  struct Feedback *newFeedback = (struct Feedback *)realloc
```

```
        (customers[customerId].feedback, customers[customerId].numFeedbacks *
        sizeof(struct Feedback));
    if (newFeedback == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }

    customers[customerId].feedback = newFeedback;

    // Input new feedback
    printf("Hello %s! Please provide your feedback.\n", customers[customerId]
        .name);
    printf("Rate us (1-5): ");
    scanf("%d", &customers[customerId].feedback[customers[customerId]
        .numFeedbacks - 1].rating);

    printf("Enter your comments (max %d characters): ", MAX_COMMENTS - 1);
    getchar();  // Clear newline character from buffer
    fgets(customers[customerId].feedback[customers[customerId].numFeedbacks -
        1].comments, MAX_COMMENTS, stdin);

    printf("Thank you for your feedback!\n");
}

void printCustomerFeedback(struct Customer customers[], int numCustomers) {
```

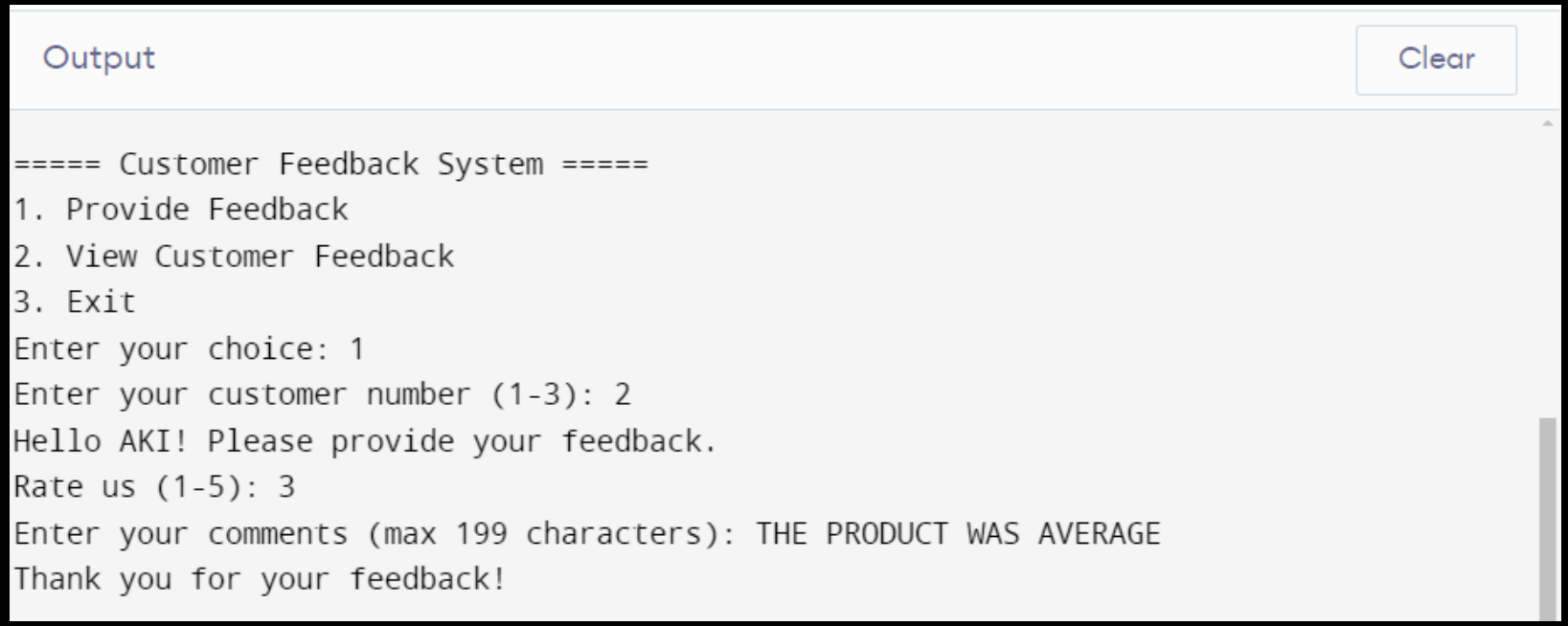
```
    for (int i = 0; i < numCustomers; ++i) {
        printf("Customer %d: %s\n", i + 1, customers[i].name);

        if (customers[i].numFeedbacks == 0) {
            printf("   No feedback given yet.\n");
        } else {
            printf("   Feedback:\n");
            for (int j = 0; j < customers[i].numFeedbacks; ++j) {
                printf("   - Rating: %d\n", customers[i].feedback[j].rating);
                printf("     Comments: %s\n", customers[i].feedback[j]
                    .comments);
            }
        }
    }
}

void freeMemory(struct Customer customers[], int numCustomers) {
    for (int i = 0; i < numCustomers; ++i) {
        free(customers[i].feedback);  // Free allocated feedback array
    }
}

```

OUTPUT

Output Clear

```
/tmp/OHPOYLlU1D.o
Enter the number of customers: 3
Enter customer 1's name: RYE
Enter customer 2's name: AKI
Enter customer 3's name: ADI

===== Customer Feedback System =====
1. Provide Feedback
2. View Customer Feedback
3. Exit
Enter your choice: 1
Enter your customer number (1-3): 1
Hello RYE! Please provide your feedback.
Rate us (1-5): 4
Enter your comments (max 199 characters): THE PRODUCT IS SATISFACTORY
Thank you for your feedback!
```

OUTPUT


Output
Clear
===== Customer Feedback System =====
1. Provide Feedback
2. View Customer Feedback
3. Exit
Enter your choice: 1
Enter your customer number (1-3): 2
Hello AKI! Please provide your feedback.
Rate us (1-5): 3
Enter your comments (max 199 characters): THE PRODUCT WAS AVERAGE
Thank you for your feedback!

OUTPUT

Output Clear

```
===== Customer Feedback System =====
1. Provide Feedback
2. View Customer Feedback
3. Exit
Enter your choice: 1
Enter your customer number (1-3): 3
Hello ADI! Please provide your feedback.
Rate us (1-5): 1
Enter your comments (max 199 characters): THE PRODUCT WAS NOT GOOD BUT THE SERVICE
    WAS GOOD
Thank you for your feedback!
```

```
===== Customer Feedback System =====
1. Provide Feedback
2. View Customer Feedback
3. Exit
Enter your choice: 2
Customer 1: RYE
   Feedback:
   - Rating: 4
     Comments: THE PRODUCT IS SATISFACTORY

Customer 2: AKI
   Feedback:
   - Rating: 3
     Comments: THE PRODUCT WAS AVERAGE

Customer 3: ADI
   Feedback:
   - Rating: 1
     Comments: THE PRODUCT WAS NOT GOOD BUT THE SERVICE WAS GOOD
```

OUTPUT

```
===== Customer Feedback System =====
1. Provide Feedback
2. View Customer Feedback
3. Exit
Enter your choice: 3
Exiting program.


=== Code Execution Successful ===
```

OUTPUT

CONCLUSION

This program is a customer feedback system in C. It allows customers to provide feedback, view feedback, and handles dynamic memory allocation for feedback entries. Here's a brief explanation of each function and the main logic:

Main Program Flow:

- Initialize customers based on user input.
- Continuously display a menu and handle user choices:
  - Provide Feedback
  - View Feedback
  - Exit and free memory

FUTURE ENCHANCEMENTS

1.GRAPHICAL USER INTERFACE: THE USER INTERFACE CAN BE UPDATED ABD TO THE LIKINGS OF THE CUSTOMER.

2. DATABASE INCREMENT: THE INCREASE IN DATABASE CAN HELP IN THE STORAGE OF MANY MORE FEEDBACKS.

3.ADVANCED SEARCHING AND STORAGE: WITH THE INCREASE OF USER FRIENDLY DESINGS IT WILL ALSO HELP GREATLY IF THE USER SIDE OF REVIEWING IS ALSO DESIGNED TO THEIR LIKINGS

4.DETECTION OF SPAM: IN A AGE OF SPAMS AND RANDOM MESSSAGES IT WILL BE GREAT IF THE SYSTEM HAD A IN BUILT PROGRAM TO DETECT SPAMS

5. ERROR HANDELLING: AUTO CORRECTION AND ERROR DETECTION WITH T5HE PROMISE OF IMMEDIATE REVISION OF THE ERROR WOULD ALSO BE GREATEFULL.

THANK YOU

RUBEN ROBY

CSE B, 64

7/17/2024