/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

typedef for structure

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

typedef struct Employee{

int Employee\_id;

int Department\_Number;

float loginTime;

float logoutTime;

} emp;

int main(){

emp emp1;

emp1.Employee\_id = 1234;

emp1.Department\_Number = 4;

return 0;

}

///typedef for pointer

#include <stdio.h>

typedef float\* fp;

int main()

{

float pi= 3.14;

fp pFlt = &pi;

printf("pi = %f \n",\*pFlt);

return 0;

}

////typedef for array

#include <stdio.h>

#define pi 3.14

typedef float arr[10];

int main()

{

arr a = {1,2,3,4,pi};

for(int i=0;i<10;i++){

printf("%.2f \n",a[i]);

}

return 0;

}

Problem 1: Inventory Management System

Description: Develop an inventory management system for an e-commerce platform.

Requirements:

Use a structure to define an item with fields: itemID, itemName, price, and quantity.

Use an array of structures to store the inventory.

Implement functions to add new items, update item details (call by reference), and display the entire inventory (call by value).

Use a loop to iterate through the inventory.

Use static to keep track of the total number of items added.

Output Expectations:

Display the updated inventory after each addition or update.

Show the total number of items.

#include <stdio.h>

#include <string.h>

#define MAX\_ITEMS 100 // Maximum number of items in the inventory

typedef struct {

int itemID;

char itemName[50];

float price;

int quantity;

} Item;

// Function prototypes

void addItem(Item inventory[], int \*totalItems);

void updateItem(Item \*inventory, int totalItems);

void displayInventory(Item inventory[], int totalItems);

int main() {

static int totalItems = 0;

Item inventory[MAX\_ITEMS];

int choice;

do {

printf("\nInventory Management System:\n");

printf("1. Add New Item\n");

printf("2. Update Item\n");

printf("3. Display Inventory\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addItem(inventory, &totalItems);

break;

case 2:

updateItem(inventory, totalItems);

break;

case 3:

displayInventory(inventory, totalItems);

break;

case 4:

printf("Exiting the program.\n");

break;

default:

printf("Invalid choice. Please try again.\n");

}

} while (choice != 4);

return 0;

}

// Function to add a new item to the inventory

void addItem(Item inventory[], int \*totalItems) {

if (\*totalItems >= MAX\_ITEMS) {

printf("Inventory is full. Cannot add more items.\n");

return;

}

printf("\nEnter details for the new item:\n");

printf("Item ID: ");

scanf("%d", &inventory[\*totalItems].itemID);

printf("Item Name: ");

scanf(" %[^\n]", inventory[\*totalItems].itemName);

printf("Price: ");

scanf("%f", &inventory[\*totalItems].price);

printf("Quantity: ");

scanf("%d", &inventory[\*totalItems].quantity);

(\*totalItems)++;

printf("Item added successfully! Total items: %d\n", \*totalItems);

displayInventory(inventory, \*totalItems);

}

// Function to update an existing item in the inventory

void updateItem(Item \*inventory, int totalItems) {

int id, found = 0;

printf("\nEnter the Item ID to update: ");

scanf("%d", &id);

for (int i = 0; i < totalItems; i++) {

if (inventory[i].itemID == id) {

found = 1;

printf("Updating details for Item ID: %d\n", id);

printf("New Item Name: ");

scanf(" %[^\n]", inventory[i].itemName);

printf("New Price: ");

scanf("%f", &inventory[i].price);

printf("New Quantity: ");

scanf("%d", &inventory[i].quantity);

printf("Item updated successfully!\n");

break;

}

}

if (!found) {

printf("Item with ID %d not found in the inventory.\n", id);

}

displayInventory(inventory, totalItems);

}

// Function to display the entire inventory

void displayInventory(Item inventory[], int totalItems) {

printf("\nCurrent Inventory:\n");

printf("--------------------------------------------------\n");

printf("ID\tName\t\tPrice\tQuantity\n");

printf("--------------------------------------------------\n");

for (int i = 0; i < totalItems; i++) {

printf("%d\t%s\t\t%.2f\t%d\n", inventory[i].itemID, inventory[i].itemName, inventory[i].price, inventory[i].quantity);

}

printf("--------------------------------------------------\n");

printf("Total Items: %d\n", totalItems);

}

Problem 2: Order Processing System

Description: Create an order processing system that calculates the total order cost and applies discounts.

Requirements:

Use a structure for Order containing fields for orderID, customerName, items (array), and totalCost.

Use const for the discount rate.

Implement functions for calculating the total cost (call by value) and applying the discount (call by reference).

Use a loop to process multiple orders.

Output Expectations:

Show the total cost before and after applying the discount for each order.

#include <stdio.h>

// Define the Order structure using typedef

typedef struct {

int orderID;

char customerName[50];

float items[10];

float totalCost;

} Order;

// Function to calculate the total cost

float calculateTotalCost(Order order) {

float total = 0;

for (int i = 0; i < 10; i++) {

total += order.items[i];

}

return total;

}

// Function to apply the discount

void applyDiscount(Order \*order, const float discountRate) {

order->totalCost \*= (1 - discountRate);

}

// Function to process and display orders

void processOrders(Order orders[], int count) {

const float discountRate = 0.10;

for (int i = 0; i < count; i++) {

// Calculate total cost

orders[i].totalCost = calculateTotalCost(orders[i]);

// Display total cost before discount

printf("Order ID: %d\n", orders[i].orderID);

printf("Customer Name: %s\n", orders[i].customerName);

printf("Total Cost Before Discount: %.2f\n", orders[i].totalCost);

// Apply discount

applyDiscount(&orders[i], discountRate);

// Display total cost after discount

printf("Total Cost After Discount: %.2f\n\n", orders[i].totalCost);

}

}

int main() {

int count;

printf("Enter the number of orders: ");

scanf("%d", &count);

Order orders[count];

for (int i = 0; i < count; i++) {

printf("Enter order ID: ");

scanf("%d", &orders[i].orderID);

printf("Enter customer name: ");

scanf("%49s", orders[i].customerName);

printf("Enter item prices (up to 10 items, enter 0 for unused items): ");

for (int j = 0; j < 10; j++) {

scanf("%f", &orders[i].items[j]);

}

}

processOrders(orders, count);

return 0;

}

Problem 3: Customer Feedback System

Description: Develop a feedback system that categorizes customer feedback based on ratings.

Requirements:

Use a structure to define Feedback with fields for customerID, feedbackText, and rating.

Use a switch case to categorize feedback (e.g., Excellent, Good, Average, Poor).

Store feedback in an array.

Implement functions to add feedback and display feedback summaries using loops.

Output Expectations:

Display categorized feedback summaries.

#include <stdio.h>

#include <string.h>

// Define the Feedback structure using typedef

typedef struct {

int customerID;

char feedbackText[200];

int rating;

} Feedback;

// Function to add feedback

void addFeedback(Feedback feedbacks[], int \*count) {

printf("Enter customer ID: ");

scanf("%d", &feedbacks[\*count].customerID);

printf("Enter feedback text: ");

scanf(" %[^\n]s", feedbacks[\*count].feedbackText); // Using %[^\n]s to read until newline

printf("Enter rating (1-Poor, 2-Average, 3-Good, 4-Excellent): ");

scanf("%d", &feedbacks[\*count].rating);

(\*count)++;

}

// Function to display categorized feedback summaries

void displayFeedbackSummaries(Feedback feedbacks[], int count) {

printf("\nFeedback Summaries:\n");

for (int i = 0; i < count; i++) {

printf("Customer ID: %d\n", feedbacks[i].customerID);

printf("Feedback: %s\n", feedbacks[i].feedbackText);

printf("Rating: ");

switch (feedbacks[i].rating) {

case 1:

printf("Poor\n");

break;

case 2:

printf("Average\n");

break;

case 3:

printf("Good\n");

break;

case 4:

printf("Excellent\n");

break;

default:

printf("Invalid rating\n");

}

printf("\n");

}

}

int main() {

int capacity = 10;

int count = 0;

Feedback feedbacks[capacity];

int choice;

while (1) {

printf("1. Add Feedback\n2. Display Feedback Summaries\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addFeedback(feedbacks, &count);

break;

case 2:

displayFeedbackSummaries(feedbacks, count);

break;

case 3:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

Problem 4: Payment Method Selection

Description: Write a program that handles multiple payment methods and calculates transaction charges.

Requirements:

Use a structure for Payment with fields for method, amount, and transactionCharge.

Use const for fixed transaction charges.

Use a switch case to determine the transaction charge based on the payment method.

Implement functions for processing payments and updating transaction details (call by reference).

Output Expectations:

Show the payment details including the method and transaction charge.

#include <stdio.h>

#include <string.h>

// Define the Payment structure using typedef

typedef struct {

char method[20];

float amount;

float transactionCharge;

} Payment;

// Function to process payment and update transaction details

void processPayment(Payment \*payment) {

const float CREDIT\_CARD\_CHARGE = 2.5;

const float DEBIT\_CARD\_CHARGE = 1.5;

const float PAYPAL\_CHARGE = 3.0;

// Determine the transaction charge based on the payment method

if (strcmp(payment->method, "CreditCard") == 0) {

payment->transactionCharge = payment->amount \* (CREDIT\_CARD\_CHARGE / 100);

} else if (strcmp(payment->method, "DebitCard") == 0) {

payment->transactionCharge = payment->amount \* (DEBIT\_CARD\_CHARGE / 100);

} else if (strcmp(payment->method, "PayPal") == 0) {

payment->transactionCharge = payment->amount \* (PAYPAL\_CHARGE / 100);

} else {

printf("Invalid payment method.\n");

payment->transactionCharge = 0;

}

}

// Function to display payment details

void displayPaymentDetails(Payment payment) {

printf("Payment Method: %s\n", payment.method);

printf("Payment Amount: %.2f\n", payment.amount);

printf("Transaction Charge: %.2f\n", payment.transactionCharge);

printf("\n");

}

int main() {

int count;

printf("Enter the number of payments: ");

scanf("%d", &count);

Payment payments[count];

for (int i = 0; i < count; i++) {

printf("Enter payment method (CreditCard, DebitCard, PayPal): ");

scanf("%19s", payments[i].method);

printf("Enter payment amount: ");

scanf("%f", &payments[i].amount);

// Process the payment to update the transaction charge

processPayment(&payments[i]);

displayPaymentDetails(payments[i]);

}

return 0;

}

Problem 5: Shopping Cart System

Description: Implement a shopping cart system that allows adding, removing, and viewing items.

Requirements:

Use a structure for CartItem with fields for itemID, itemName, price, and quantity.

Use an array to store the cart items.

Implement functions to add, remove (call by reference), and display items (call by value).

Use loops for iterating through cart items.

#include <stdio.h>

#include <string.h>

#define MAX\_CART\_SIZE 10 // Maximum number of items in the cart

// Define the CartItem structure using typedef

typedef struct {

int itemID;

char itemName[50];

float price;

int quantity;

} CartItem;

// Function prototypes

void addItem(CartItem cart[], int \*numItems);

void removeItem(CartItem cart[], int \*numItems);

void displayCart(CartItem cart[], int numItems);

int main() {

CartItem cart[MAX\_CART\_SIZE]; // Array to hold the cart items

int numItems = 0; // Track number of items in the cart

int choice;

while (1) {

printf("\nShopping Cart System\n");

printf("1. Add Item\n");

printf("2. Remove Item\n");

printf("3. Display Cart\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addItem(cart, &numItems);

break;

case 2:

removeItem(cart, &numItems);

break;

case 3:

displayCart(cart, numItems);

break;

case 4:

printf("Exiting program.\n");

return 0;

default:

printf("Invalid choice. Try again.\n");

}

}

return 0;

}

// Function to add an item to the cart

void addItem(CartItem cart[], int \*numItems) {

if (\*numItems < MAX\_CART\_SIZE) {

CartItem newItem;

printf("Enter item ID: ");

scanf("%d", &newItem.itemID);

// Read item name using scanf

printf("Enter item name: ");

// Use %[^\n] to capture the entire line of input for the item name

scanf(" %[^\n]", newItem.itemName);

printf("Enter price: ");

scanf("%f", &newItem.price);

printf("Enter quantity: ");

scanf("%d", &newItem.quantity);

cart[\*numItems] = newItem;

(\*numItems)++; // Increment number of items in the cart

printf("Item added successfully!\n");

} else {

printf("Cart is full, cannot add more items.\n");

}

}

// Function to remove an item from the cart

void removeItem(CartItem cart[], int \*numItems) {

if (\*numItems == 0) {

printf("Cart is empty, nothing to remove.\n");

return;

}

int itemID, found = 0;

printf("Enter item ID to remove: ");

scanf("%d", &itemID);

// Search for the item in the cart

for (int i = 0; i < \*numItems; i++) {

if (cart[i].itemID == itemID) {

// Shift remaining items left

for (int j = i; j < \*numItems - 1; j++) {

cart[j] = cart[j + 1];

}

(\*numItems)--; // Decrease number of items in the cart

found = 1;

printf("Item removed successfully!\n");

break;

}

}

if (!found) {

printf("Item not found in the cart.\n");

}

}

// Function to display the cart items

void displayCart(CartItem cart[], int numItems) {

if (numItems == 0) {

printf("Your cart is empty.\n");

return;

}

printf("\nItems in your cart:\n");

printf("-------------------------------------------------\n");

printf("| Item ID | Item Name | Price | Quantity |\n");

printf("-------------------------------------------------\n");

for (int i = 0; i < numItems; i++) {

printf("| %7d | %-22s | %.2f | %8d |\n", cart[i].itemID, cart[i].itemName, cart[i].price, cart[i].quantity);

}

printf("-------------------------------------------------\n");

}

Problem 6: Product Search System

Description: Create a system that allows searching for products by name or ID.

Requirements:

Use a structure for Product with fields for productID, productName, category, and price.

Store products in an array.

Use a loop to search for a product.

Implement functions for searching by name (call by value) and updating details (call by reference).

Output Expectations:

Display product details if found or a message indicating the product is not found

#include <stdio.h>

#include <string.h>

// Define the Product structure using typedef

typedef struct {

int productID;

char productName[50];

char category[30];

float price;

} Product;

// Function to search for a product by name

void searchByName(Product products[], int count, char name[]) {

int found = 0;

for (int i = 0; i < count; i++) {

if (strcmp(products[i].productName, name) == 0) {

printf("Product found:\n");

printf("ID: %d\n", products[i].productID);

printf("Name: %s\n", products[i].productName);

printf("Category: %s\n", products[i].category);

printf("Price: %.2f\n", products[i].price);

found = 1;

break;

}

}

if (!found) {

printf("Product with name \"%s\" not found.\n", name);

}

}

// Function to update product details

void updateProductDetails(Product \*product) {

printf("Enter new product name: ");

scanf("%49s", product->productName);

printf("Enter new category: ");

scanf("%29s", product->category);

printf("Enter new price: ");

scanf("%f", &product->price);

}

// Function to search for a product by ID

Product\* searchByID(Product products[], int count, int id) {

for (int i = 0; i < count; i++) {

if (products[i].productID == id) {

return &products[i];

}

}

return NULL;

}

int main() {

int capacity = 10;

int count = 0;

Product products[capacity];

int choice;

while (1) {

printf("1. Add Product\n2. Search Product by Name\n3. Search Product by ID and Update Details\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter product ID: ");

scanf("%d", &products[count].productID);

printf("Enter product name: ");

scanf("%49s", products[count].productName);

printf("Enter category: ");

scanf("%29s", products[count].category);

printf("Enter price: ");

scanf("%f", &products[count].price);

count++;

break;

case 2:

if (count == 0) {

printf("No products available.\n");

} else {

char name[50];

printf("Enter product name: ");

scanf("%49s", name);

searchByName(products, count, name);

}

break;

case 3:

if (count == 0) {

printf("No products available.\n");

} else {

int id;

printf("Enter product ID: ");

scanf("%d", &id);

Product \*product = searchByID(products, count, id);

if (product != NULL) {

updateProductDetails(product);

} else {

printf("Product with ID %d not found.\n", id);

}

}

break;

case 4:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

Problem 7: Sales Report Generator

Description: Develop a system that generates a sales report for different categories.

Requirements:

Use a structure for Sale with fields for saleID, productCategory, amount, and date.

Store sales in an array.

Use a loop and switch case to categorize and summarize sales.

Implement functions to add sales data and generate reports.

Output Expectations:

Display summarized sales data by category.

#include <stdio.h>

#include <string.h>

// Define the Sale structure using typedef

typedef struct {

int saleID;

char productCategory[30];

float amount;

char date[11]; // Format: YYYY-MM-DD

} Sale;

// Function to add sales data

void addSale(Sale sales[], int \*count) {

printf("Enter sale ID: ");

scanf("%d", &sales[\*count].saleID);

printf("Enter product category: ");

scanf("%29s", sales[\*count].productCategory);

printf("Enter sale amount: ");

scanf("%f", &sales[\*count].amount);

printf("Enter sale date (YYYY-MM-DD): ");

scanf("%10s", sales[\*count].date);

(\*count)++;

}

// Function to generate sales report

void generateSalesReport(Sale sales[], int count) {

float electronicsTotal = 0;

float clothingTotal = 0;

float groceriesTotal = 0;

for (int i = 0; i < count; i++) {

if (strcmp(sales[i].productCategory, "Electronics") == 0) {

electronicsTotal += sales[i].amount;

} else if (strcmp(sales[i].productCategory, "Clothing") == 0) {

clothingTotal += sales[i].amount;

} else if (strcmp(sales[i].productCategory, "Groceries") == 0) {

groceriesTotal += sales[i].amount;

}

}

printf("\nSales Report:\n");

printf("Electronics: %.2f\n", electronicsTotal);

printf("Clothing: %.2f\n", clothingTotal);

printf("Groceries: %.2f\n", groceriesTotal);

}

int main() {

int capacity = 100;

int count = 0;

Sale sales[capacity];

int choice;

while (1) {

printf("1. Add Sale\n2. Generate Sales Report\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addSale(sales, &count);

break;

case 2:

generateSalesReport(sales, count);

break;

case 3:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

Problem 8: Customer Loyalty Program

Description: Implement a loyalty program that rewards customers based on their total purchase amount.

Requirements:

Use a structure for Customer with fields for customerID, name, totalPurchases, and rewardPoints.

Use const for the reward rate.

Implement functions to calculate and update reward points (call by reference).

Use a loop to process multiple customers.

Output Expectations:

Display customer details including reward points after updating.

#include <stdio.h>

#include <string.h>

// Define the Customer structure using typedef

typedef struct {

int customerID;

char name[50];

float totalPurchases;

int rewardPoints;

} Customer;

// Function to calculate and update reward points

void updateRewardPoints(Customer \*customer, const float rewardRate) {

customer->rewardPoints = customer->totalPurchases \* rewardRate;

}

// Function to display customer details

void displayCustomerDetails(Customer customer) {

printf("Customer ID: %d\n", customer.customerID);

printf("Name: %s\n", customer.name);

printf("Total Purchases: %.2f\n", customer.totalPurchases);

printf("Reward Points: %d\n", customer.rewardPoints);

printf("\n");

}

int main() {

const float rewardRate = 0.05; // 5% reward rate

int capacity = 100;

int count = 0;

Customer customers[capacity];

int choice;

while (1) {

printf("1. Add Customer\n2. Update and Display Reward Points\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter customer ID: ");

scanf("%d", &customers[count].customerID);

printf("Enter name: ");

scanf("%49s", customers[count].name);

printf("Enter total purchases: ");

scanf("%f", &customers[count].totalPurchases);

customers[count].rewardPoints = 0;

count++;

break;

case 2:

for (int i = 0; i < count; i++) {

updateRewardPoints(&customers[i], rewardRate);

displayCustomerDetails(customers[i]);

}

break;

case 3:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

Problem 9: Warehouse Management System

Description: Create a warehouse management system to track stock levels of different products.

Requirements:

Use a structure for WarehouseItem with fields for itemID, itemName, currentStock, and reorderLevel.

Use an array to store warehouse items.

Implement functions to update stock levels (call by reference) and check reorder status (call by value).

Use a loop for updating stock.

Output Expectations:

Display the stock levels and reorder status for each item.

#include <stdio.h>

#include <string.h>

// Define the WarehouseItem structure using typedef

typedef struct {

int itemID;

char itemName[50];

int currentStock;

int reorderLevel;

} WarehouseItem;

// Function to update stock levels

void updateStock(WarehouseItem \*item, int stockChange) {

item->currentStock += stockChange;

}

// Function to check reorder status

void checkReorderStatus(WarehouseItem item) {

printf("Item ID: %d\n", item.itemID);

printf("Item Name: %s\n", item.itemName);

printf("Current Stock: %d\n", item.currentStock);

printf("Reorder Level: %d\n", item.reorderLevel);

if (item.currentStock <= item.reorderLevel) {

printf("Status: Reorder Needed\n\n");

} else {

printf("Status: Stock Sufficient\n\n");

}

}

int main() {

int capacity = 10;

int count = 0;

WarehouseItem items[capacity];

int choice;

while (1) {

printf("1. Add Warehouse Item\n2. Update Stock Levels\n3. Check Reorder Status\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

if (count < capacity) {

printf("Enter item ID: ");

scanf("%d", &items[count].itemID);

printf("Enter item name: ");

scanf("%49s", items[count].itemName);

printf("Enter current stock: ");

scanf("%d", &items[count].currentStock);

printf("Enter reorder level: ");

scanf("%d", &items[count].reorderLevel);

count++;

} else {

printf("Warehouse is full. Cannot add more items.\n");

}

break;

case 2:

if (count == 0) {

printf("No items in the warehouse.\n");

} else {

int itemID, stockChange;

printf("Enter item ID to update stock: ");

scanf("%d", &itemID);

printf("Enter stock change (positive for adding stock, negative for removing stock): ");

scanf("%d", &stockChange);

int found = 0;

for (int i = 0; i < count; i++) {

if (items[i].itemID == itemID) {

updateStock(&items[i], stockChange);

found = 1;

break;

}

}

if (!found) {

printf("Item with ID %d not found.\n", itemID);

}

}

break;

case 3:

if (count == 0) {

printf("No items in the warehouse.\n");

} else {

for (int i = 0; i < count; i++) {

checkReorderStatus(items[i]);

}

}

break;

case 4:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

Problem 10: Discount Management System

Description: Design a system that manages discounts for different product categories.

Requirements:

Use a structure for Discount with fields for category, discountPercentage, and validTill.

Use const for predefined categories.

Use a switch case to apply discounts based on the category.

Implement functions to update and display discounts (call by reference).

Output Expectations:

Show the updated discount details for each category.

#include <stdio.h>

#include <string.h>

// Define the Discount structure using typedef

typedef struct {

char category[30];

float discountPercentage;

char validTill[11]; // Format: YYYY-MM-DD

} Discount;

// Function to update discount

void updateDiscount(Discount \*discount, float newPercentage, const char \*newValidTill) {

discount->discountPercentage = newPercentage;

strcpy(discount->validTill, newValidTill);

}

// Function to display discount details

void displayDiscount(Discount discount) {

printf("Category: %s\n", discount.category);

printf("Discount Percentage: %.2f%%\n", discount.discountPercentage);

printf("Valid Till: %s\n\n", discount.validTill);

}

// Function to apply discounts based on the category

void applyDiscounts(Discount discounts[], int count) {

for (int i = 0; i < count; i++) {

if (strcmp(discounts[i].category, "Electronics") == 0) {

updateDiscount(&discounts[i], 10.0, "2025-12-31");

} else if (strcmp(discounts[i].category, "Clothing") == 0) {

updateDiscount(&discounts[i], 15.0, "2025-11-30");

} else if (strcmp(discounts[i].category, "Groceries") == 0) {

updateDiscount(&discounts[i], 5.0, "2025-10-31");

} else {

printf("No predefined discount for category: %s\n", discounts[i].category);

}

displayDiscount(discounts[i]);

}

}

int main() {

const int capacity = 10;

int count = 0;

Discount discounts[capacity];

int choice;

while (1) {

printf("1. Add Discount\n2. Apply and Display Discounts\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

if (count < capacity) {

printf("Enter category: ");

scanf("%29s", discounts[count].category);

printf("Enter discount percentage: ");

scanf("%f", &discounts[count].discountPercentage);

printf("Enter valid till (YYYY-MM-DD): ");

scanf("%10s", discounts[count].validTill);

count++;

} else {

printf("Cannot add more discounts. Capacity reached.\n");

}

break;

case 2:

applyDiscounts(discounts, count);

break;

case 3:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

UNION------------------------------------

#include <stdio.h>

union num{

int a;

char b;

};

union add{

int c[10];

char d;

};

int main (){

union num x1;

union add x2;

printf("size of x1 = %ld\n",sizeof(x1));

printf("size of x2 = %ld\n",sizeof(x2));

printf("address of x1.a = %p\n",&x1.a);

printf("address of x1.b = %p\n",&x1.b);

return 0;

}

Typedef and union

#include<stdio.h>

typedef union{

int a;

char b;

}n;

int main(){

n x1;

n \*ptr = &x1;

ptr->a = 10;

ptr->b = 'a';

printf("001 x1.a = %d \n", ptr->a);

printf("002 x1.b = %c \n", ptr->b);

return 0;

}

Union inside Structure

#include <stdio.h>

struct Student{

char section;

int RollNo;

union{

float mathMarks;

float chemMarks;

float TotalMarks;

}marks;

};

int main(){

struct Student s1;

s1.section = 'A';

s1.RollNo = 1;

s1.marks.mathMarks = 45.5;

s1.marks.chemMarks = 54.5;

s1.marks.TotalMarks = s1.marks.mathMarks + s1.marks.chemMarks;

printf("Student 1:\nSection = %c \nRoll No.: = %d \n",s1.section,s1.RollNo);

printf("Math Marks: = %.2f\nChem Marks: = %.2f\nTotal Marks: = %.2f\n",s1.marks.mathMarks,s1.marks.chemMarks,s1.marks.TotalMarks);

return 0;

}

**Problem 1: Union for Mixed Data**

**Description**: Create a union that can store an integer, a float, or a character. Write a program that assigns values to each member and displays them

#include <stdio.h>

typedef union{

int rollno;

float totalmarks;

char division;

}student;

int main(){

student s1;

s1.rollno = 1;

printf("Rollno = %d\n",s1.rollno);

s1.totalmarks = 20.5;

printf("total marks = %.2f\n",s1.totalmarks);

s1.division = 'A';

printf("division = %c",s1.division);

return 0;

}

**Problem 2: Student Data with Union**

**Description**: Define a union to store either a student’s roll number (integer) or name (string). Write a program to input and display student details using the union.

**Problem 3: Union for Measurement Units**

**Description**: Create a union that can store a distance in either kilometers (float) or miles (float). Write a program to convert and display the distance in both units.

#include <stdio.h>

typedef union {

float kilometers;

float miles;

} Distance;

// Function to convert kilometers to miles

float kilometersToMiles(float km) {

return km \* 0.621371;

}

// Function to convert miles to kilometers

float milesToKilometers(float mi) {

return mi / 0.621371;

}

int main() {

Distance distance;

int choice;

float value;

printf("Enter 1 to input distance in kilometers or 2 to input distance in miles: ");

scanf("%d", &choice);

if (choice == 1)

{

printf("Enter distance in kilometers: ");

scanf("%f", &value);

distance.kilometers = value;

printf("Distance: %.2f kilometers\n", distance.kilometers);

printf("Distance: %.2f miles\n", kilometersToMiles(distance.kilometers));

} else if (choice == 2)

{

printf("Enter distance in miles: ");

scanf("%f", &value);

distance.miles = value;

printf("Distance: %.2f miles\n", distance.miles);

printf("Distance: %.2f kilometers\n", milesToKilometers(distance.miles));

} else {

printf("Invalid choice.\n");

}

return 0;

}

**Problem 4: Union for Shape Dimensions**

**Description**: Define a union to store dimensions of different shapes: a radius (float) for a circle, length and width (float) for a rectangle. Write a program to calculate and display the area based on the selected shape.

#include <stdio.h>

// Define the union to store dimensions of different shapes

typedef union {

float radius;

struct {

float length;

float width;

} rectangle;

} ShapeDimensions;

void calculateCircleArea(ShapeDimensions dimensions) {

float area = 3.14159f \* dimensions.radius \* dimensions.radius;

printf("Area of the circle: %.2f\n", area);

}

// Function to calculate and display the area of a rectangle

void calculateRectangleArea(ShapeDimensions dimensions) {

float area = dimensions.rectangle.length \* dimensions.rectangle.width;

printf("Area of the rectangle: %.2f\n", area);

}

int main() {

ShapeDimensions dimensions;

int choice;

printf("Enter 1 for circle or 2 for rectangle: ");

scanf("%d", &choice);

if (choice == 1) {

printf("Enter radius of the circle: ");

scanf("%f", &dimensions.radius);

calculateCircleArea(dimensions);

} else if (choice == 2) {

printf("Enter length of the rectangle: ");

scanf("%f", &dimensions.rectangle.length);

printf("Enter width of the rectangle: ");

scanf("%f", &dimensions.rectangle.width);

calculateRectangleArea(dimensions);

} else {

printf("Invalid choice.\n");

}

return 0;

}

**Problem 5: Union for Employee Data**

**Description**: Create a union to store either an employee’s ID (integer) or salary (float). Write a program to input and display either ID or salary based on user choice.

#include <stdio.h>

typedef union {

int employeeID;

float salary;

} EmployeeData;

int main() {

EmployeeData employee;

int choice;

printf("Enter 1 to input employee ID or 2 to input employee salary: ");

scanf("%d", &choice);

if (choice == 1)

{

printf("Enter employee ID: ");

scanf("%d", &employee.employeeID);

printf("Employee ID: %d\n", employee.employeeID);

} else if (choice == 2)

{

printf("Enter employee salary: ");

scanf("%f", &employee.salary);

printf("Employee salary: %.2f\n", employee.salary);

} else

{

printf("Invalid choice.\n");

}

return 0;

}

**Problem 6: Union for Sensor Data**

**Description**: Define a union to store sensor data, either temperature (float) or pressure (float). Write a program to simulate sensor readings and display the data.

#include <stdio.h>

typedef union {

float temperature;

float pressure;

} SensorData;

int main() {

SensorData sensor;

int choice;

printf("Enter 1 to simulate temperature reading or 2 to simulate pressure reading: ");

scanf("%d", &choice);

if (choice == 1) {

sensor.temperature = 36.5;

printf("Temperature reading: %.2f °C\n", sensor.temperature);

} else if (choice == 2) {

sensor.pressure = 1000.25;

printf("Pressure reading: %.2f\n", sensor.pressure);

} else {

printf("Invalid choice.\n");

}

return 0;

}

**Problem 7: Union for Bank Account Information**

**Description**: Create a union to store either a bank account number (integer) or balance (float). Write a program to input and display either the account number or balance based on user input.

#include<stdio.h>

typedef union {

int acntno;

float balance;

}Bankacnt;

int main(){

Bankacnt account;

int choice;

printf("enter 1 for acount number and 2 for input balance");

scanf("%d",&choice);

if (choice == 1){

printf("entr account number :");

scanf("%d",&account.acntno);

printf(" account number : %d",account.acntno);

}

else if(choice==2){

printf("enter the balance");

scanf("%f",&account.balance);

printf(" balance : %f",account.balance);

}

return 0;

}

**Problem 8: Union for Vehicle Information**

**Description**: Define a union to store either the vehicle’s registration number (integer) or fuel capacity (float). Write a program to input and display either the registration number or fuel capacity.

#include<stdio.h>

typedef union {

int regno;

float fuelcap;

}vehicleinfo;

int main(){

vehicleinfo vehicle;

int choice;

printf("Enter 1 for registration number or 2 for fuel capacity");

scanf("%d",&choice);

if(choice == 1){

printf("enter the registration number :");

scanf("%d",&vehicle.regno);

printf("registration number is : %d",vehicle.regno);

}

else if(choice==2){

printf("enter the fuel capacity");

scanf("%f",&vehicle.fuelcap);

printf("fuel capacity is : %.2f",vehicle.fuelcap);

}

else {

printf("ivalid choice");

}

return 0;

}

**Problem 9: Union for Exam Results**

**Description**: Create a union to store either a student’s marks (integer) or grade (char). Write a program to input marks or grade and display the corresponding value.

#include <stdio.h>

typedef union {

int marks;

char grade;

} ExamResult;

int main() {

ExamResult result;

int choice;

printf("Enter 1 to input marks or 2 to input grade: ");

scanf("%d", &choice);

if (choice == 1) {

printf("Enter student's marks: ");

scanf("%d", &result.marks);

printf("Student's marks: %d\n", result.marks);

} else if (choice == 2) {

printf("Enter student's grade: ");

scanf(" %c", &result.grade);

printf("Student's grade: %c\n", result.grade);

} else {

printf("Invalid choice.\n");

}

return 0;

}

**Problem 10: Union for Currency Conversion**

**Description**: Define a union to store currency values in either USD (float) or EUR (float). Write a program to input a value in one currency and display the equivalent in the other.

#include <stdio.h>

typedef union {

float usd;

float eur;

} Currency;

const float USD\_TO\_EUR = 0.85;

const float EUR\_TO\_USD = 1.18;

int main() {

Currency currency;

int choice;

float value;

printf("Enter 1 to input value in USD or 2 to input value in EUR: ");

scanf("%d", &choice);

if (choice == 1) {

printf("Enter value in USD: ");

scanf("%f", &value);

currency.usd = value;

printf("Value in USD: %.2f\n", currency.usd);

printf("Equivalent value in EUR: %.2f\n", currency.usd \* USD\_TO\_EUR);

} else if (choice == 2) {

printf("Enter value in EUR: ");

scanf("%f", &value);

currency.eur = value;

printf("Value in EUR: %.2f\n", currency.eur);

printf("Equivalent value in USD: %.2f\n", currency.eur \* EUR\_TO\_USD);

} else {

printf("Invalid choice.\n");

}

return 0;

}

**Problem 1: Aircraft Fleet Management**

**Description**: Develop a system to manage a fleet of aircraft, tracking their specifications and operational status.

**Requirements**:

* Define a struct for Aircraft with fields: aircraftID, model, capacity, and status.
* Use an array of Aircraft structures.
* Implement functions to add new aircraft (call by reference), update status, and display fleet details (call by value).
* Use static to track the total number of aircraft.
* Utilize a switch case to manage different operational statuses.
* Employ loops to iterate through the fleet.

**Output Expectations**:

* Display updated fleet information after each operation.

#include <stdio.h>

#include <string.h>

// Define the Aircraft struct

typedef struct {

int aircraftID;

char model[50];

int capacity;

char status[20];

} Aircraft;

// Function to add new aircraft

void addAircraft(Aircraft fleet[], int \*count) {

printf("Enter aircraft ID: ");

scanf("%d", &fleet[\*count].aircraftID);

printf("Enter model: ");

scanf("%49s", fleet[\*count].model);

printf("Enter capacity: ");

scanf("%d", &fleet[\*count].capacity);

printf("Enter status (Operational/Maintenance/Out of Service): ");

scanf("%s", fleet[\*count].status);

(\*count)++;

printf("Aircraft added successfully!\n");

}

// Function to update status

void updateStatus(Aircraft fleet[], int count) {

int id;

printf("Enter aircraft ID to update status: ");

scanf("%d", &id);

int found = 0;

for (int i = 0; i < count; i++) {

if (fleet[i].aircraftID == id) {

printf("Enter new status (Operational/Maintenance/Out of Service): ");

scanf("%s", fleet[i].status);

found = 1;

printf("Status updated successfully!\n");

break;

}

}

if (!found) {

printf("Aircraft with ID %d not found.\n", id);

}

}

// Function to display fleet details (call by value)

void displayFleet(Aircraft fleet[], int count) {

printf("\nCurrent Fleet:\n");

printf("ID\tModel\t\tCapacity\tStatus\n");

for (int i = 0; i < count; i++) {

printf("%d\t%s\t\t%d\t\t%s\n", fleet[i].aircraftID, fleet[i].model, fleet[i].capacity, fleet[i].status);

}

printf("\n");

}

int main() {

int capacity = 100;

int count = 0;

static int totalAircraft = 0; // Static variable to track total number of aircraft

Aircraft fleet[capacity];

int choice;

while (1) {

printf("1. Add New Aircraft\n2. Update Aircraft Status\n3. Display Fleet\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addAircraft(fleet, &count);

totalAircraft++;

printf("Total number of aircraft: %d\n", totalAircraft);

break;

case 2:

updateStatus(fleet, count);

break;

case 3:

displayFleet(fleet, count);

break;

case 4:

printf("Total number of aircraft: %d\n", totalAircraft);

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

**Problem 2: Satellite Data Processing**

**Description**: Create a system to process and analyze satellite data.

**Requirements**:

* Define a union for SatelliteData to store either image data (array) or telemetry data (nested structure).
* Use struct to define Telemetry with fields: temperature, velocity, and altitude.
* Implement functions to process image and telemetry data (call by reference).
* Use const for fixed telemetry limits.
* Employ loops to iterate through data points.

**Output Expectations**:

* Display processed image or telemetry data based on user input.

#include <stdio.h>

#include <string.h>

// Define the Telemetry struct

typedef struct {

float temperature;

float velocity;

float altitude;

} Telemetry;

// Define the SatelliteData union

typedef union {

unsigned char imageData[256];

Telemetry telemetry;

} SatelliteData;

// Function to process image data

void processImageData(SatelliteData \*data) {

printf("Processing image data...\n");

// Example: Display image data as hex values

for (int i = 0; i < 256; i++) {

printf("%02X ", data->imageData[i]);

if ((i + 1) % 16 == 0) {

printf("\n");

}

}

}

// Function to process telemetry data

void processTelemetryData(SatelliteData \*data) {

const float TEMP\_LIMIT = 100.0f; // Example temperature limit

const float VELOCITY\_LIMIT = 30000.0f; // Example velocity limit

const float ALTITUDE\_LIMIT = 500000.0f; // Example altitude limit

printf("Processing telemetry data...\n");

printf("Temperature: %.2f °C\n", data->telemetry.temperature);

printf("Velocity: %.2f m/s\n", data->telemetry.velocity);

printf("Altitude: %.2f m\n", data->telemetry.altitude);

if (data->telemetry.temperature > TEMP\_LIMIT) {

printf("Warning: Temperature exceeds limit!\n");

}

if (data->telemetry.velocity > VELOCITY\_LIMIT) {

printf("Warning: Velocity exceeds limit!\n");

}

if (data->telemetry.altitude > ALTITUDE\_LIMIT) {

printf("Warning: Altitude exceeds limit!\n");

}

}

int main() {

SatelliteData data;

int choice;

// Ask the user to choose the type of satellite data to process

printf("Enter 1 to process image data or 2 to process telemetry data: ");

scanf("%d", &choice);

if (choice == 1) {

// Simulate image data

for (int i = 0; i < 256; i++) {

data.imageData[i] = i;

}

processImageData(&data);

} else if (choice == 2) {

// Input and process telemetry data

printf("Enter temperature (°C): ");

scanf("%f", &data.telemetry.temperature);

printf("Enter velocity (m/s): ");

scanf("%f", &data.telemetry.velocity);

printf("Enter altitude (m): ");

scanf("%f", &data.telemetry.altitude);

processTelemetryData(&data);

} else {

printf("Invalid choice.\n");

}

return 0;

}

**Problem 3: Mission Control System**

**Description**: Develop a mission control system to manage spacecraft missions.

**Requirements**:

* Define a struct for Mission with fields: missionID, name, duration, and a nested union for payload (either crew details or cargo).
* Implement functions to add missions (call by reference), update mission details, and display mission summaries (call by value).
* Use static to count total missions.
* Use loops and switch case for managing different mission types.

**Output Expectations**:

* Provide detailed mission summaries including payload information.

#include <stdio.h>

#include <string.h>

// Define the CrewDetails struct

typedef struct {

char crewNames[3][50];

int crewCount;

} CrewDetails;

// Define the CargoDetails struct

typedef struct {

float cargoWeight;

char cargoType[50];

} CargoDetails;

// Define the Payload union

typedef union {

CrewDetails crew;

CargoDetails cargo;

} Payload;

// Define the Mission struct

typedef struct {

int missionID;

char name[50];

int duration; // Duration in days

char missionType[10];

Payload payload;

} Mission;

// Function to add a new mission

void addMission(Mission missions[], int \*count) {

printf("Enter mission ID: ");

scanf("%d", &missions[\*count].missionID);

printf("Enter mission name: ");

scanf("%s", missions[\*count].name);

printf("Enter mission duration (days): ");

scanf("%d", &missions[\*count].duration);

printf("Enter mission type (Crew/Cargo): ");

scanf("%s", missions[\*count].missionType);

if (strcmp(missions[\*count].missionType, "Crew") == 0) {

printf("Enter number of crew members: ");

scanf("%d", &missions[\*count].payload.crew.crewCount);

for (int i = 0; i < missions[\*count].payload.crew.crewCount; i++) {

printf("Enter name of crew member %d: ", i + 1);

scanf("%s", missions[\*count].payload.crew.crewNames[i]);

}

} else if (strcmp(missions[\*count].missionType, "Cargo") == 0) {

printf("Enter cargo weight (kg): ");

scanf("%f", &missions[\*count].payload.cargo.cargoWeight);

printf("Enter cargo type: ");

scanf("%s", missions[\*count].payload.cargo.cargoType);

} else {

printf("Invalid mission type.\n");

return;

}

(\*count)++;

printf("Mission added successfully!\n");

//displayMissionSummaries(missions, \*count);

}

// Function to update mission details

void updateMissionDetails(Mission missions[], int count) {

int id;

printf("Enter mission ID to update details: ");

scanf("%d", &id);

int found = 0;

for (int i = 0; i < count; i++) {

if (missions[i].missionID == id) {

printf("Enter new mission name: ");

scanf("%s", missions[i].name);

printf("Enter new mission duration (days): ");

scanf("%d", &missions[i].duration);

printf("Enter new mission type (Crew/Cargo): ");

scanf("%s", missions[i].missionType);

if (strcmp(missions[i].missionType, "Crew") == 0) {

printf("Enter number of crew members: ");

scanf("%d", &missions[i].payload.crew.crewCount);

for (int j = 0; j < missions[i].payload.crew.crewCount; j++) {

printf("Enter name of crew member %d: ", j + 1);

scanf("%s", missions[i].payload.crew.crewNames[j]);

}

} else if (strcmp(missions[i].missionType, "Cargo") == 0) {

printf("Enter cargo weight (kg): ");

scanf("%f", &missions[i].payload.cargo.cargoWeight);

printf("Enter cargo type: ");

scanf("%s", missions[i].payload.cargo.cargoType);

} else {

printf("Invalid mission type.\n");

return;

}

found = 1;

printf("Mission details updated successfully!\n");

break;

}

}

if (!found) {

printf("Mission with ID %d not found.\n", id);

}

//displayMissionSummaries(missions, count);

}

// Function to display mission summaries

void displayMissionSummaries(Mission missions[], int count) {

printf("\nMission Summaries:\n");

for (int i = 0; i < count; i++) {

printf("Mission ID: %d\n", missions[i].missionID);

printf("Name: %s\n", missions[i].name);

printf("Duration: %d days\n", missions[i].duration);

printf("Type: %s\n", missions[i].missionType);

if (strcmp(missions[i].missionType, "Crew") == 0) {

printf("Crew Count: %d\n", missions[i].payload.crew.crewCount);

for (int j = 0; j < missions[i].payload.crew.crewCount; j++) {

printf("Crew Member %d: %s\n", j + 1, missions[i].payload.crew.crewNames[j]);

}

} else if (strcmp(missions[i].missionType, "Cargo") == 0) {

printf("Cargo Weight: %.2f kg\n", missions[i].payload.cargo.cargoWeight);

printf("Cargo Type: %s\n", missions[i].payload.cargo.cargoType);

}

printf("\n");

}

}

int main() {

Mission missions[100]; // Array to store missions

int count = 0;

static int totalMissions = 0; // Static variable to count total missions

int choice;

while (1) {

printf("1. Add Mission\n2. Update Mission Details\n3. Display Mission Summaries\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addMission(missions, &count);

totalMissions++;

printf("Total number of missions: %d\n", totalMissions);

break;

case 2:

updateMissionDetails(missions, count);

break;

case 3:

displayMissionSummaries(missions, count);

break;

case 4:

printf("Total number of missions: %d\n", totalMissions);

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

**Problem 4: Aircraft Maintenance Tracker**

**Description**: Create a tracker for aircraft maintenance schedules and logs.

**Requirements**:

* Use a struct for MaintenanceLog with fields: logID, aircraftID, date, and a nested union for maintenance type (routine or emergency).
* Implement functions to add maintenance logs (call by reference) and display logs (call by value).
* Use const for maintenance frequency.
* Employ loops to iterate through maintenance logs.

**Output Expectations**:

* Display maintenance logs categorized by type.

#include <stdio.h>

#include <string.h>

typedef struct {

char description[100];

int hoursSpent;

} RoutineMaintenance;

typedef struct {

char description[100];

int hoursSpent;

char severity[20];

} EmergencyMaintenance;

typedef union {

RoutineMaintenance routine;

EmergencyMaintenance emergency;

} MaintenanceType;

typedef struct {

int logID;

int aircraftID;

char date[11];

char maintenanceCategory[10];

MaintenanceType maintenance;

} MaintenanceLog;

// Function to add a maintenance log

void addMaintenanceLog(MaintenanceLog logs[], int \*count) {

printf("Enter log ID: ");

scanf("%d", &logs[\*count].logID);

printf("Enter aircraft ID: ");

scanf("%d", &logs[\*count].aircraftID);

printf("Enter date (YYYY-MM-DD): ");

scanf("%s", logs[\*count].date);

printf("Enter maintenance category (Routine/Emergency): ");

scanf("%s", logs[\*count].maintenanceCategory);

if (strcmp(logs[\*count].maintenanceCategory, "Routine") == 0) {

printf("Enter routine maintenance description: ");

scanf("%s", logs[\*count].maintenance.routine.description);

printf("Enter hours spent on routine maintenance: ");

scanf("%d", &logs[\*count].maintenance.routine.hoursSpent);

} else if (strcmp(logs[\*count].maintenanceCategory, "Emergency") == 0) {

printf("Enter emergency maintenance description: ");

scanf("%s", logs[\*count].maintenance.emergency.description);

printf("Enter hours spent on emergency maintenance: ");

scanf("%d", &logs[\*count].maintenance.emergency.hoursSpent);

printf("Enter severity of emergency maintenance: ");

scanf("%s", logs[\*count].maintenance.emergency.severity);

} else {

printf("Invalid maintenance category.\n");

return;

}

(\*count)++;

printf("Maintenance log added successfully!\n");

}

// Function to display maintenance logs

void displayMaintenanceLogs(MaintenanceLog logs[], int count) {

printf("\nMaintenance Logs:\n");

for (int i = 0; i < count; i++) {

printf("Log ID: %d\n", logs[i].logID);

printf("Aircraft ID: %d\n", logs[i].aircraftID);

printf("Date: %s\n", logs[i].date);

printf("Category: %s\n", logs[i].maintenanceCategory);

if (strcmp(logs[i].maintenanceCategory, "Routine") == 0) {

printf("Description: %s\n", logs[i].maintenance.routine.description);

printf("Hours Spent: %d\n", logs[i].maintenance.routine.hoursSpent);

} else if (strcmp(logs[i].maintenanceCategory, "Emergency") == 0) {

printf("Description: %s\n", logs[i].maintenance.emergency.description);

printf("Hours Spent: %d\n", logs[i].maintenance.emergency.hoursSpent);

printf("Severity: %s\n", logs[i].maintenance.emergency.severity);

}

printf("\n");

}

}

int main() {

MaintenanceLog logs[100];

int count = 0;

const int maintenanceFrequency = 100;

int choice;

while (1) {

printf("1. Add Maintenance Log\n2. Display Maintenance Logs\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addMaintenanceLog(logs, &count);

break;

case 2:

displayMaintenanceLogs(logs, count);

break;

case 3:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

**Problem 5: Spacecraft Navigation System**

**Description**: Develop a navigation system for spacecraft to track their position and velocity.

**Requirements**:

* Define a struct for NavigationData with fields: position, velocity, and a nested union for navigation mode (manual or automatic).
* Implement functions to update navigation data (call by reference) and display the current status (call by value).
* Use static to count navigation updates.
* Use loops and switch case for managing navigation modes.

**Output Expectations**:

* Show updated position and velocity with navigation mode details.

#include <stdio.h>

#include <string.h>

// Define the ManualMode struct

typedef struct {

char pilotName[50];

char controlStatus[50];

} ManualMode;

// Define the AutomaticMode struct

typedef struct {

char systemName[50];

int errorCode;

} AutomaticMode;

// Define the NavigationMode union

typedef union {

ManualMode manual;

AutomaticMode automatic;

} NavigationMode;

// Define the NavigationData

typedef struct {

float position[3]; // Position in 3D space (x, y, z)

float velocity[3]; // Velocity in 3D space (vx, vy, vz)

char mode[10];

NavigationMode navigation;

} NavigationData;

// Function to update navigation data

void updateNavigationData(NavigationData \*data) {

static int updateCount = 0;

updateCount++;

printf("Enter position (x y z): ");

scanf("%f %f %f", &data->position[0], &data->position[1], &data->position[2]);

printf("Enter velocity (vx vy vz): ");

scanf("%f %f %f", &data->velocity[0], &data->velocity[1], &data->velocity[2]);

printf("Enter navigation mode (Manual/Automatic): ");

scanf("%s", data->mode);

if (strcmp(data->mode, "Manual") == 0) {

printf("Enter pilot name: ");

scanf("%s", data->navigation.manual.pilotName);

printf("Enter control status: ");

scanf("%s", data->navigation.manual.controlStatus);

} else if (strcmp(data->mode, "Automatic") == 0) {

printf("Enter system name: ");

scanf("%s", data->navigation.automatic.systemName);

printf("Enter error code: ");

scanf("%d", &data->navigation.automatic.errorCode);

} else {

printf("Invalid navigation mode.\n");

return;

}

printf("Navigation data updated successfully! Total updates: %d\n", updateCount);

}

// Function to display navigation status

void displayNavigationStatus(NavigationData data) {

printf("\nCurrent Navigation Status:\n");

printf("Position: (%.2f, %.2f, %.2f)\n", data.position[0], data.position[1], data.position[2]);

printf("Velocity: (%.2f, %.2f, %.2f)\n", data.velocity[0], data.velocity[1], data.velocity[2]);

printf("Mode: %s\n", data.mode);

if (strcmp(data.mode, "Manual") == 0) {

printf("Pilot Name: %s\n", data.navigation.manual.pilotName);

printf("Control Status: %s\n", data.navigation.manual.controlStatus);

} else if (strcmp(data.mode, "Automatic") == 0) {

printf("System Name: %s\n", data.navigation.automatic.systemName);

printf("Error Code: %d\n", data.navigation.automatic.errorCode);

}

printf("\n");

}

int main() {

NavigationData navData;

int choice;

while (1) {

printf("1. Update Navigation Data\n2. Display Navigation Status\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

updateNavigationData(&navData);

break;

case 2:

displayNavigationStatus(navData);

break;

case 3:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

**Problem 6: Flight Simulation Control**

**Description**: Create a control system for flight simulations with different aircraft models.

**Requirements**:

* Define a struct for Simulation with fields: simulationID, aircraftModel, duration, and a nested union for control settings (manual or automated).
* Implement functions to start simulations (call by reference), update settings, and display simulation results (call by value).
* Use const for fixed simulation parameters.
* Utilize loops to run multiple simulations and a switch case for selecting control settings.

**Output Expectations**:

* Display simulation results with control settings.

#include <stdio.h>

#include <string.h>

// Define the ManualSettings struct

typedef struct {

char pilotName[50];

char controlMode[50];

} ManualSettings;

// Define the AutomatedSettings struct

typedef struct {

char systemName[50];

int autoLevel; // Automation level (e.g., 1-5)

} AutomatedSettings;

// Define the ControlSettings union

typedef union {

ManualSettings manual;

AutomatedSettings automated;

} ControlSettings;

// Define the Simulation struct

typedef struct {

int simulationID;

char aircraftModel[50];

int duration; // Duration in minutes

char controlType[10]; // "Manual" or "Automated"

ControlSettings settings;

} Simulation;

// Function to start a simulation

void startSimulation(Simulation \*sim) {

printf("Enter simulation ID: ");

scanf("%d", &sim->simulationID);

printf("Enter aircraft model: ");

scanf("%s", sim->aircraftModel);

printf("Enter duration (minutes): ");

scanf("%d", &sim->duration);

printf("Enter control type (Manual/Automated): ");

scanf("%s", sim->controlType);

if (strcmp(sim->controlType, "Manual") == 0) {

printf("Enter pilot name: ");

scanf("%s", sim->settings.manual.pilotName);

printf("Enter control mode: ");

scanf("%s", sim->settings.manual.controlMode);

} else if (strcmp(sim->controlType, "Automated") == 0) {

printf("Enter system name: ");

scanf("%s", sim->settings.automated.systemName);

printf("Enter automation level (1-5): ");

scanf("%d", &sim->settings.automated.autoLevel);

} else {

printf("Invalid control type.\n");

return;

}

printf("Simulation started successfully!\n");

}

// Function to update settings

void updateSettings(Simulation \*sim) {

printf("Update settings for simulation ID %d:\n", sim->simulationID);

printf("Enter new duration (minutes): ");

scanf("%d", &sim->duration);

printf("Enter new control type (Manual/Automated): ");

scanf("%s", sim->controlType);

if (strcmp(sim->controlType, "Manual") == 0) {

printf("Enter new pilot name: ");

scanf("%s", sim->settings.manual.pilotName);

printf("Enter new control mode: ");

scanf("%s", sim->settings.manual.controlMode);

} else if (strcmp(sim->controlType, "Automated") == 0) {

printf("Enter new system name: ");

scanf("%s", sim->settings.automated.systemName);

printf("Enter new automation level (1-5): ");

scanf("%d", &sim->settings.automated.autoLevel);

} else {

printf("Invalid control type.\n");

return;

}

printf("Settings updated successfully!\n");

}

// Function to display simulation results

void displaySimulationResults(Simulation sim) {

printf("\nSimulation Results:\n");

printf("Simulation ID: %d\n", sim.simulationID);

printf("Aircraft Model: %s\n", sim.aircraftModel);

printf("Duration: %d minutes\n", sim.duration);

printf("Control Type: %s\n", sim.controlType);

if (strcmp(sim.controlType, "Manual") == 0) {

printf("Pilot Name: %s\n", sim.settings.manual.pilotName);

printf("Control Mode: %s\n", sim.settings.manual.controlMode);

} else if (strcmp(sim.controlType, "Automated") == 0) {

printf("System Name: %s\n", sim.settings.automated.systemName);

printf("Automation Level: %d\n", sim.settings.automated.autoLevel);

}

printf("\n");

}

int main() {

const int maxSimulations = 100; // Example fixed simulation parameter

Simulation simulations[maxSimulations];

int count = 0;

int choice;

while (1) {

printf("1. Start Simulation\n2. Update Simulation Settings\n3. Display Simulation Results\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

if (count < maxSimulations) {

startSimulation(&simulations[count]);

count++;

} else {

printf("Maximum number of simulations reached.\n");

}

break;

case 2:

if (count > 0) {

int simID;

printf("Enter simulation ID to update: ");

scanf("%d", &simID);

int found = 0;

for (int i = 0; i < count; i++) {

if (simulations[i].simulationID == simID) {

updateSettings(&simulations[i]);

found = 1;

break;

}

}

if (!found) {

printf("Simulation ID %d not found.\n", simID);

}

} else {

printf("No simulations to update.\n");

}

break;

case 3:

for (int i = 0; i < count; i++) {

displaySimulationResults(simulations[i]);

}

break;

case 4:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

**Problem 7: Aerospace Component Testing**

**Description**: Develop a system for testing different aerospace components.

**Requirements**:

* Use a struct for ComponentTest with fields: testID, componentName, and a nested union for test data (physical or software).
* Implement functions to record test results (call by reference) and display summaries (call by value).
* Use static to count total tests conducted.
* Employ loops and switch case for managing different test types.

**Output Expectations**:

* Display test results categorized by component type.

#include <stdio.h>

#include <string.h>

// Define the PhysicalTest struct

typedef struct {

float loadCapacity;

float temperatureTolerance;

} PhysicalTest;

// Define the SoftwareTest struct

typedef struct {

int passedCases;

int totalCases;

} SoftwareTest;

// Define the TestData union

typedef union {

PhysicalTest physical;

SoftwareTest software;

} TestData;

// Define the ComponentTest struct

typedef struct {

int testID;

char componentName[50];

char testType[10];

TestData data;

} ComponentTest;

// Function to record test result

void recordTestResult(ComponentTest \*test) {

printf("Enter test ID: ");

scanf("%d", &test->testID);

printf("Enter component name: ");

scanf("%s", test->componentName);

printf("Enter test type (Physical/Software): ");

scanf("%s", test->testType);

if (strcmp(test->testType, "Physical") == 0) {

printf("Enter load capacity (kg): ");

scanf("%f", &test->data.physical.loadCapacity);

printf("Enter temperature tolerance (°C): ");

scanf("%f", &test->data.physical.temperatureTolerance);

} else if (strcmp(test->testType, "Software") == 0) {

printf("Enter number of passed cases: ");

scanf("%d", &test->data.software.passedCases);

printf("Enter total number of cases: ");

scanf("%d", &test->data.software.totalCases);

} else {

printf("Invalid test type.\n");

return;

}

static int totalTests = 0;

totalTests++;

printf("Test recorded successfully! Total tests conducted: %d\n", totalTests);

//displayTestSummary(\*test);

}

// Function to display test summaries

void displayTestSummary(ComponentTest test) {

printf("\nTest Summary:\n");

printf("Test ID: %d\n", test.testID);

printf("Component Name: %s\n", test.componentName);

printf("Test Type: %s\n", test.testType);

if (strcmp(test.testType, "Physical") == 0) {

printf("Load Capacity: %.2f kg\n", test.data.physical.loadCapacity);

printf("Temperature Tolerance: %.2f °C\n", test.data.physical.temperatureTolerance);

} else if (strcmp(test.testType, "Software") == 0) {

printf("Passed Cases: %d\n", test.data.software.passedCases);

printf("Total Cases: %d\n", test.data.software.totalCases);

printf("Pass Rate: %.2f%%\n", (float)test.data.software.passedCases / test.data.software.totalCases \* 100);

}

printf("\n");

}

int main() {

const int maxTests = 100;

ComponentTest tests[maxTests];

int count = 0;

int choice;

while (1) {

printf("1. Record Test Result\n2. Display Test Summaries\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

if (count < maxTests) {

recordTestResult(&tests[count]);

count++;

} else {

printf("Maximum number of tests recorded.\n");

}

break;

case 2:

for (int i = 0; i < count; i++) {

displayTestSummary(tests[i]);

}

break;

case 3:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

/\*

Problem 8: Space Station Crew Management

Description: Create a system to manage crew members aboard a space station.

Requirements:

Define a struct for CrewMember with fields: crewID, name, role, and a nested union for role-specific details (engineer or scientist).

Implement functions to add crew members (call by reference), update details, and display crew lists (call by value).

Use const for fixed role limits.

Use loops to iterate through the crew list and a switch case for role management.

Output Expectations:

Show updated crew information including role-specific details.

\*/

#include <stdio.h>

#include <string.h>

// Define the EngineerDetails struct

typedef struct {

char specialization[50];

int yearsOfExperience;

} EngineerDetails;

// Define the ScientistDetails struct

typedef struct {

char researchField[50];

int publishedPapers;

} ScientistDetails;

// Define the RoleDetails union

typedef union {

EngineerDetails engineer;

ScientistDetails scientist;

} RoleDetails;

// Define the CrewMember struct

typedef struct {

int crewID;

char name[50];

char role[10];

RoleDetails details;

} CrewMember;

// Function to add a crew member

void addCrewMember(CrewMember crew[], int \*count) {

printf("Enter crew ID: ");

scanf("%d", &crew[\*count].crewID);

printf("Enter name: ");

scanf("%s", crew[\*count].name);

printf("Enter role (Engineer/Scientist): ");

scanf("%s", crew[\*count].role);

if (strcmp(crew[\*count].role, "Engineer") == 0) {

printf("Enter specialization: ");

scanf("%s", crew[\*count].details.engineer.specialization);

printf("Enter years of experience: ");

scanf("%d", &crew[\*count].details.engineer.yearsOfExperience);

} else if (strcmp(crew[\*count].role, "Scientist") == 0) {

printf("Enter research field: ");

scanf("%s", crew[\*count].details.scientist.researchField);

printf("Enter number of published papers: ");

scanf("%d", &crew[\*count].details.scientist.publishedPapers);

} else {

printf("Invalid role.\n");

return;

}

(\*count)++;

printf("Crew member added successfully!\n");

//displayCrewList(crew, \*count);

}

// Function to update crew member detail

void updateCrewMemberDetails(CrewMember crew[], int count) {

int id;

printf("Enter crew ID to update details: ");

scanf("%d", &id);

int found = 0;

for (int i = 0; i < count; i++) {

if (crew[i].crewID == id) {

printf("Enter new name: ");

scanf("%s", crew[i].name);

printf("Enter new role (Engineer/Scientist): ");

scanf("%s", crew[i].role);

if (strcmp(crew[i].role, "Engineer") == 0) {

printf("Enter new specialization: ");

scanf("%s", crew[i].details.engineer.specialization);

printf("Enter new years of experience: ");

scanf("%d", &crew[i].details.engineer.yearsOfExperience);

} else if (strcmp(crew[i].role, "Scientist") == 0) {

printf("Enter new research field: ");

scanf("%s", crew[i].details.scientist.researchField);

printf("Enter new number of published papers: ");

scanf("%d", &crew[i].details.scientist.publishedPapers);

} else {

printf("Invalid role.\n");

return;

}

found = 1;

printf("Crew member details updated successfully!\n");

break;

}

}

if (!found) {

printf("Crew member with ID %d not found.\n", id);

}

// displayCrewList(crew, count);

}

// Function to display crew list

void displayCrewList(CrewMember crew[], int count) {

printf("\nCrew List:\n");

for (int i = 0; i < count; i++) {

printf("Crew ID: %d\n", crew[i].crewID);

printf("Name: %s\n", crew[i].name);

printf("Role: %s\n", crew[i].role);

if (strcmp(crew[i].role, "Engineer") == 0) {

printf("Specialization: %s\n", crew[i].details.engineer.specialization);

printf("Years of Experience: %d\n", crew[i].details.engineer.yearsOfExperience);

} else if (strcmp(crew[i].role, "Scientist") == 0) {

printf("Research Field: %s\n", crew[i].details.scientist.researchField);

printf("Published Papers: %d\n", crew[i].details.scientist.publishedPapers);

}

printf("\n");

}

}

int main() {

const int maxCrewMembers = 100;

CrewMember crew[maxCrewMembers];

int count = 0;

int choice;

while (1) {

printf("1. Add Crew Member\n2. Update Crew Member Details\n3. Display Crew List\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

if (count < maxCrewMembers) {

addCrewMember(crew, &count);

} else {

printf("Maximum number of crew members reached.\n");

}

break;

case 2:

if (count > 0) {

updateCrewMemberDetails(crew, count);

} else {

printf("No crew members to update.\n");

}

break;

case 3:

displayCrewList(crew, count);

break;

case 4:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

**Problem 9: Aerospace Research Data Analysis**

**Description**: Develop a system to analyze research data from aerospace experiments.

**Requirements**:

* Use a struct for ResearchData with fields: experimentID, description, and a nested union for data type (numerical or qualitative).
* Implement functions to analyze data (call by reference) and generate reports (call by value).
* Use static to track the number of analyses conducted.
* Employ loops and switch case for managing different data types.

**Output Expectations**:

* Provide detailed reports of analyzed data.

#include <stdio.h>

#include <string.h>

// Define the NumericalData struct

typedef struct {

float values[100];

int count;

} NumericalData;

// Define the QualitativeData struct

typedef struct {

char observations[200];

} QualitativeData;

// Define the DataType union

typedef union {

NumericalData numerical;

QualitativeData qualitative;

} DataType;

// Define the ResearchData struct

typedef struct {

int experimentID;

char description[100];

char dataType[12];

DataType data;

} ResearchData;

// Function to analyze data

void analyzeData(ResearchData \*data) {

static int analysisCount = 0;

analysisCount++;

printf("Analyzing data for experiment ID: %d\n", data->experimentID);

printf("Description: %s\n", data->description);

printf("Data Type: %s\n", data->dataType);

if (strcmp(data->dataType, "Numerical") == 0) {

float sum = 0;

for (int i = 0; i < data->data.numerical.count; i++) {

sum += data->data.numerical.values[i];

}

float average = sum / data->data.numerical.count;

printf("Number of values: %d\n", data->data.numerical.count);

printf("Sum of values: %.2f\n", sum);

printf("Average of values: %.2f\n", average);

} else if (strcmp(data->dataType, "Qualitative") == 0) {

printf("Observations: %s\n", data->data.qualitative.observations);

} else {

printf("Invalid data type.\n");

return;

}

printf("Analysis completed. Total analyses conducted: %d\n", analysisCount);

// generateReport(\*data);

}

// Function to generate a report

void generateReport(ResearchData data) {

printf("\nGenerated Report:\n");

printf("Experiment ID: %d\n", data.experimentID);

printf("Description: %s\n", data.description);

printf("Data Type: %s\n", data.dataType);

if (strcmp(data.dataType, "Numerical") == 0) {

float sum = 0;

for (int i = 0; i < data.data.numerical.count; i++) {

sum += data.data.numerical.values[i];

}

float average = sum / data.data.numerical.count;

printf("Number of values: %d\n", data.data.numerical.count);

printf("Sum of values: %.2f\n", sum);

printf("Average of values: %.2f\n", average);

} else if (strcmp(data.dataType, "Qualitative") == 0) {

printf("Observations: %s\n", data.data.qualitative.observations);

}

printf("\n");

}

int main() {

const int maxExperiments = 100;

ResearchData experiments[maxExperiments];

int count = 0;

int choice;

while (1) {

printf("1. Add Experiment Data\n2. Analyze Experiment Data\n3. Display Reports\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

if (count < maxExperiments) {

printf("Enter experiment ID: ");

scanf("%d", &experiments[count].experimentID);

printf("Enter description: ");

scanf("%s", experiments[count].description);

printf("Enter data type (Numerical/Qualitative): ");

scanf("%s", experiments[count].dataType);

if (strcmp(experiments[count].dataType, "Numerical") == 0) {

printf("Enter number of values: ");

scanf("%d", &experiments[count].data.numerical.count);

for (int i = 0; i < experiments[count].data.numerical.count; i++) {

printf("Enter value %d: ", i + 1);

scanf("%f", &experiments[count].data.numerical.values[i]);

}

} else if (strcmp(experiments[count].dataType, "Qualitative") == 0) {

printf("Enter observations: ");

scanf("%s", experiments[count].data.qualitative.observations);

} else {

printf("Invalid data type.\n");

continue;

}

count++;

printf("Experiment data added successfully!\n");

} else {

printf("Maximum number of experiments reached.\n");

}

break;

case 2:

if (count > 0) {

int expID;

printf("Enter experiment ID to analyze: ");

scanf("%d", &expID);

int found = 0;

for (int i = 0; i < count; i++) {

if (experiments[i].experimentID == expID) {

analyzeData(&experiments[i]);

found = 1;

break;

}

}

if (!found) {

printf("Experiment ID %d not found.\n", expID);

}

} else {

printf("No experiments to analyze.\n");

}

break;

case 3:

for (int i = 0; i < count; i++) {

generateReport(experiments[i]);

}

break;

case 4:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

**Problem 10: Rocket Launch Scheduler**

**Description**: Create a scheduler for managing rocket launches.

**Requirements**:

* Define a struct for Launch with fields: launchID, rocketName, date, and a nested union for launch status (scheduled or completed).
* Implement functions to schedule launches (call by reference), update statuses, and display launch schedules (call by value).
* Use const for fixed launch parameters.
* Use loops to iterate through launch schedules and a switch case for managing status updates.

**Output Expectations**:

* Display detailed launch schedules and statuses.

#include <stdio.h>

#include <string.h>

// Define the ScheduledStatus struct

typedef struct {

char plannedDate[11]; // Format: YYYY-MM-DD

} ScheduledStatus;

// Define the CompletedStatus struct

typedef struct {

char actualDate[11]; // Format: YYYY-MM-DD

char missionOutcome[20];

} CompletedStatus;

// Define the LaunchStatus union

typedef union {

ScheduledStatus scheduled;

CompletedStatus completed;

} LaunchStatus;

// Define the Launch struct

typedef struct {

int launchID;

char rocketName[50];

char statusType[10];

LaunchStatus status;

} Launch;

// Function to schedule a launch

void scheduleLaunch(Launch \*launch) {

printf("Enter launch ID: ");

scanf("%d", &launch->launchID);

printf("Enter rocket name: ");

scanf("%s", launch->rocketName);

printf("Enter status type (Scheduled/Completed): ");

scanf("%s", launch->statusType);

if (strcmp(launch->statusType, "Scheduled") == 0) {

printf("Enter planned date (YYYY-MM-DD): ");

scanf("%s", launch->status.scheduled.plannedDate);

} else if (strcmp(launch->statusType, "Completed") == 0) {

printf("Enter actual date (YYYY-MM-DD): ");

scanf("%s", launch->status.completed.actualDate);

printf("Enter mission outcome (Success/Failure): ");

scanf("%s", launch->status.completed.missionOutcome);

} else {

printf("Invalid status type.\n");

return;

}

printf("Launch scheduled successfully!\n");

// displayLaunchSchedule(\*launch);

}

// Function to update launch status

void updateLaunchStatus(Launch \*launch) {

printf("Enter new status type (Scheduled/Completed): ");

scanf("%s", launch->statusType);

if (strcmp(launch->statusType, "Scheduled") == 0) {

printf("Enter new planned date (YYYY-MM-DD): ");

scanf("%s", launch->status.scheduled.plannedDate);

} else if (strcmp(launch->statusType, "Completed") == 0) {

printf("Enter new actual date (YYYY-MM-DD): ");

scanf("%s", launch->status.completed.actualDate);

printf("Enter new mission outcome (Success/Failure): ");

scanf("%s", launch->status.completed.missionOutcome);

} else {

printf("Invalid status type.\n");

return;

}

printf("Launch status updated successfully!\n");

//displayLaunchSchedule(\*launch);

}

// Function to display launch schedule

void displayLaunchSchedule(Launch launch) {

printf("\nLaunch Schedule:\n");

printf("Launch ID: %d\n", launch.launchID);

printf("Rocket Name: %s\n", launch.rocketName);

printf("Status Type: %s\n", launch.statusType);

if (strcmp(launch.statusType, "Scheduled") == 0) {

printf("Planned Date: %s\n", launch.status.scheduled.plannedDate);

} else if (strcmp(launch.statusType, "Completed") == 0) {

printf("Actual Date: %s\n", launch.status.completed.actualDate);

printf("Mission Outcome: %s\n", launch.status.completed.missionOutcome);

}

printf("\n");

}

int main() {

const int maxLaunches = 100; // Example fixed launch parameter

Launch launches[maxLaunches];

int count = 0;

int choice;

while (1) {

printf("1. Schedule Launch\n2. Update Launch Status\n3. Display Launch Schedules\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

if (count < maxLaunches) {

scheduleLaunch(&launches[count]);

count++;

} else {

printf("Maximum number of launches scheduled.\n");

}

break;

case 2:

if (count > 0) {

int launchID;

printf("Enter launch ID to update: ");

scanf("%d", &launchID);

int found = 0;

for (int i = 0; i < count; i++) {

if (launches[i].launchID == launchID) {

updateLaunchStatus(&launches[i]);

found = 1;

break;

}

}

if (!found) {

printf("Launch ID %d not found.\n", launchID);

}

} else {

printf("No launches to update.\n");

}

break;

case 3:

for (int i = 0; i < count; i++) {

displayLaunchSchedule(launches[i]);

}

break;

case 4:

return 0;

default:

printf("Invalid choice.\n");

}

}

}