|  |  |  |  |
| --- | --- | --- | --- |
| **Object Oriented Programming Lab 06** | | | |
| **Course Code:** | COMP-112L | **Class** | CS (B) |
| **Lab Engineer** | Laiba Khalid | **Semester** | 2nd |
| **Lab Title** | C++ Classes | **Section** | - |
| **Name** |  | **Reg no.** |  |
| **Content Covered** | C++ Class Implementation. | | |
| **Instructions:**  • Submit the file with your names following your registration numbers like AI001\_Name.  • Submit soft copy of the report before deadline. Marks will be deducted for late submissions. | | | |

**Constant and Non-Constant Functions & Data Members**

**Topic 1: Constant and Non-Constant Functions**

**Theory**

* A **constant function** does not modify any member variables of a class.
* Declared using the const keyword at the end of the function declaration.
* A **non-constant function** can modify member variables.

**Example**

|  |
| --- |
| #include <iostream>  using namespace std;  class Sample {  private:  int value;  public:  Sample(int v) : value(v) {}  void show() const { // Constant function  cout << "Value: " << value << endl;  }  void setValue(int v) { // Non-constant function  value = v;  }  };  int main() {  Sample obj(10);  obj.show();  obj.setValue(20);  obj.show();  return 0;  } |

**Practice Task:**

1. Write a C++ program to create a class Circle with a **constant function** to calculate the area and a **non-constant function** to modify the radius. Demonstrate calling both functions.

**Topic 2: Constant and Non-Constant Data Members**

**Theory**

* A **constant data member** must be initialized at declaration or in the constructor.
* A **non-constant data member** can be modified anytime.

**Example:**

|  |
| --- |
| #include <iostream>  using namespace std;  class Student {  private:  const int rollNo;  string name;  public:  Student(int r, string n) : rollNo(r), name(n) {}  void show() const {  cout << "Roll No: " << rollNo << ", Name: " << name << endl;  }  };  int main() {  Student s1(101, "Alice");  s1.show();  return 0;  } |

**Practice Task:**

1. Write a C++ program to create a class Rectangle with a **constant data member** for the width and a **non-constant data member** for the height. Use an initialization list to set the width.

**Real Life Case Scenario: Bank Account Management System**

In a banking system, an account's **account number** and **account holder's name** remain constant after creation to ensure data integrity. These are defined as **constant data members**. The system provides **constant member functions**, such as getAccountNumber() and getAccountHolderName(), to retrieve these values without modifying them. While the balance and transaction history can change, the **core identity of the account remains unchanged**, ensuring security and preventing accidental modifications.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Header definition:**   |  | | --- | | #include <iostream>  using namespace std; |   **Class definition:**   |  | | --- | | class BankAccount {  private:  const string accountNumber;  const string accountHolder;  double balance; |   **Constructor:**   |  | | --- | | BankAccount(string accNum, string holder, double initialBalance) : accountNumber(accNum), accountHolder(holder), balance(initialBalance) {} |   **Constant Member Function:**     |  | | --- | | string getAccountNumber() const  {  return accountNumber;  }  string getAccountHolderName() const  {  return accountHolder;  } |   **Transaction Functions:**   |  | | --- | | **//deposit function**  void deposit(double amount) {  balance += amount; // Adds the deposit amount to balance  }  **//withdraw function**  void withdraw(double amount) {  if (amount <= balance) balance -= amount; // Deducts amount if sufficient balance is available  else std::cout << "Insufficient funds!" << std::endl;  }  **//display function**  void displayBalance() const {  std::cout << "Balance: $" << balance << std::endl;  } |   **Main Function:**   |  | | --- | | int main() {  **// Creating a bank account with fixed account number and holder name**  BankAccount account("123456789", "John Doe", 1000.0);  **// Displaying account details**  cout << "Account Number: " << account.getAccountNumber() << endl;  cout << "Account Holder: " << account.getAccountHolderName() << endl;  **// Performing transactions**  account.deposit(500);  account.displayBalance(); // Expected balance: $1500  account.withdraw(300);  account.displayBalance(); // Expected balance: $1200  return 0;  } |   **Complete Code:**  #include <iostream>  using namespace std;  class BankAccount {  private:  const string accountNumber; // Constant data member (cannot be modified)  const string accountHolder; // Constant data member (cannot be modified)  double balance; // Variable data member (modifiable)  public:  // Constructor with initializer list to initialize constant data members  BankAccount(string accNum, string holder, double initialBalance)  : accountNumber(accNum), accountHolder(holder), balance(initialBalance) {}  // Constant member functions (do not modify class data)  string getAccountNumber() const { return accountNumber; }  string getAccountHolderName() const { return accountHolder; }  // Transaction functions  void deposit(double amount) {  balance += amount; // Adds the deposit amount to balance  }  void withdraw(double amount) {  if (amount <= balance)  balance -= amount; // Deducts amount if sufficient balance is available  else  cout << "Insufficient funds!" << endl;  }  void displayBalance() const {  cout << "Balance: $" << balance << endl;  }  };  int main() {  // Creating a bank account with fixed account number and holder name  BankAccount account("123456789", "John Doe", 1000.0);  // Displaying account details  cout << "Account Number: " << account.getAccountNumber() << endl;  cout << "Account Holder: " << account.getAccountHolderName() << endl;  // Performing transactions  account.deposit(500);  account.displayBalance(); // Expected balance: $1500  account.withdraw(300);  account.displayBalance(); // Expected balance: $1200  return 0;  } |