Problem Statement: Date Management System

Objective:

Develop a **Date Management System** using **Object-Oriented Programming (OOP) in C++**, incorporating concepts such as **classes and objects, constructors, this pointer, mutators, facilitators, and accessors**.

Problem Description:

A system is required to **store, modify, and display date values** efficiently. The system should provide functionalities for:

1. **Creating date objects** with:
    - A **default constructor** that initializes the date to a predefined value (e.g., 01-01-2000).
    - A **parameterized constructor** that allows the user to set the date during object creation.
2. **Updating and retrieving date values** using:
    - **Mutators (setYear)** to modify the year of a date object.
    - **Accessors (getYear)** to retrieve the year of a date object.
3. **Allowing user interaction** with:
    - **Accepting a date (accept)** as input from the user.
    - **Displaying the date (display)** in a readable format.

Constraints & Assumptions:

- The **day** should be between **1 and 31**, depending on the month.
- The **month** should be between **1 and 12**.
- The **year** should be a valid positive integer.
- The **mutators should validate inputs** before modifying date values.

Expected Outcome:

- A program that allows users to **create and manage date objects**.
- Implementation of **constructors**, **mutators**, **facilitators**, **accessors**, and the **this pointer**.
- Users should be able to **set, modify, and retrieve date values efficiently**.

Steps to solve

Steps to Solve the Date Management System in C++

Step 1: Define the Class and Data Members

- Create a Date class with the following **private** attributes:
    - day (integer)
    - month (integer)
    - year (integer)

Step 2: Implement Constructors

- **Default Constructor**: Initialize the date to a default value (e.g., 01-01-2000).
- **Parameterized Constructor**: Allow initialization with user-provided values.

Step 3: Implement Mutators (Setters)

- **setYear(int year)**: Update the year of the object.
- **accept()**: Allow the user to input day, month, and year values.

Step 4: Implement Accessors (Getters)

- **getYear()**: Return the year of the object.

Step 5: Implement Facilitators (Utility Methods)

- **display()**: Print the current date in DD-MM-YYYY format.

Step 6: Validate Inputs (Optional)

- Ensure **day** is between **1-31**, **month** between **1-12**, and **year** is a valid number.

Step 7: Create and Use Objects in main()

- Declare **objects** of Date class.
- Test **default constructor** and **parameterized constructor**.
- Modify the date using **setters** and retrieve values using **getters**.
- Call **accept()** to take user input and display the updated date.

Implementation Outline

1. **Define the Date class** in a header file (Date.h).
2. **Implement the methods** in a separate .cpp file (DateSrc.cpp).
3. **Write the main() function** in another .cpp file (client.cpp) to demonstrate the functionality.

1. Create a Date.h

```
//this project demonstrates classes and object,constructrs, this pointer
//mutators,facilitator,accessors, get and set

#pragma once
#include<iostream>
using namespace std;
class Date
{
private:
	int day, month, year;
public:
	Date(); //no-args constr
	Date(int day,int month,int year); //para constr
	void accept(); //mutator- changes the state of a current object
	void display(); //facilitator- display the current state of an object
	int getYear();  //accessor- reads the state of a current object
	void setYear(int year);  //mutator
};
```

2. Create DateSrc.cpp

```
#include"Date.h"

Date::Date()
{
	day = 1;
	month = 1;
	year = 2025;
}

Date::Date(int day, int month, int year)
{
	this->day = day;   //curr object's day=para day value
	this->month = month;
	this->year = year;
}
void Date::accept()
{
	cout << "\n Enter the date:";
	cin >> day >> month >> year;

}
void Date::display()
{
	cout << "\n the date is " << day << "/" << month << "/" << year;
}

int Date::getYear()
{
	return year;
}
void Date::setYear(int year)
{
	this->year = year;
}
```

Stepwise Explanation of DateSrc.cpp
This file implements the **Date Management System** by defining constructors, mutators, accessors, and facilitators for the Date class.

Step 1: Include the Header File (Date.h)
#include"Date.h"

- This line **includes the Date.h header file**, which contains the class definition.
- It allows us to use the Date class and its member functions.

Step 2: Implement the Default Constructor

```
Date::Date()
{
   day = 1;
   month = 1;
   year = 2025;
}
```

- This is a **default constructor** (i.e., it takes no arguments).
- It **initializes the day, month, and year attributes** to default values (1-1-2025).
- It ensures that an object can be created without passing any values.

Step 3: Implement the Parameterized Constructor

```
Date::Date(int day, int month, int year)
{
   this->day = day;   // Current object's day = parameter day
   this->month = month;
   this->year = year;
}
```

- This **constructor initializes a Date object with user-provided values**.
- The this-> keyword is used to refer to the **current object's attributes**, distinguishing them from the function parameters.
    - Example: this->day = day ensures that the object's day gets assigned the function argument day.

Step 4: Implement the accept() Method (Mutator)

```
void Date::accept()
{
   cout << "\n Enter the date:";
   cin >> day >> month >> year;
}
```

- This method **takes input from the user** and assigns values to day, month, and year.
- It is a **mutator method** because it **modifies the state of an object**.
- Example usage:

```
Date d1;
d1.accept(); // User enters: 12 10 2024
```

    - If the user enters 12 10 2024, day = 12, month = 10, and year = 2024.

Step 5: Implement the display() Method (Facilitator)

```
void Date::display()
{
   cout << "\n the date is " << day << "/" << month << "/" << year;
}
```

- This **facilitator method displays the date in DD/MM/YYYY format**.
- Example usage:

```
Date d1(12, 10, 2024);
d1.display(); // Output: the date is 12/10/2024
```

Step 6: Implement the getYear() Method (Accessor)

```
int Date::getYear()
{
   return year;
}
```

- This **accessor method returns the year value** of the object.
- Example usage:

```
Date d1(15, 8, 2023);
cout << "Year: " << d1.getYear(); // Output: Year: 2023
```

- It **does not modify** the object's state.

Step 7: Implement the setYear() Method (Mutator)

```
void Date::setYear(int year)
{
   this->year = year;
}
```

- This **mutator method updates the year attribute** of the object.
- Example usage:

```
Date d1;
d1.setYear(2030);
cout << d1.getYear(); // Output: 2030
```

- **this->year = year;** ensures that the function parameter is assigned to the object's attribute.

Final Overview:

- **Constructors:** Initialize object attributes.
- **Mutators (setYear, accept)**: Modify object data.
- **Accessors (getYear)**: Retrieve object data.
- **Facilitators (display)**: Display object data.