* ***Purpose :*** **Classwork.**

* ***Date :*** **26/12/2025**

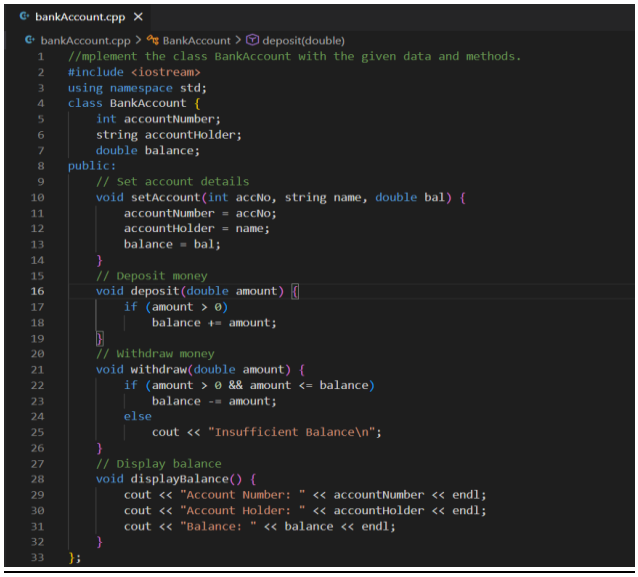
* ***Author :*** **Vikas Srivastava**

* ***ID :*** **55984**

* ***Batch ID :*** **25SUB4505**

---

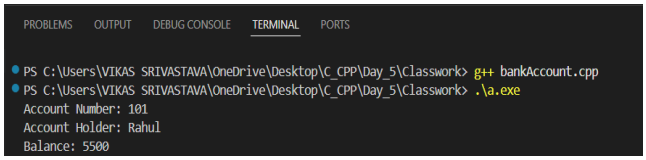
1. ***Code :*** Write a program to implement a BankAccount class using object-oriented programming concepts in C++, where data members store account details and member functions perform operations such as deposit, withdrawal, and balance display.



```cpp
//mplement the class BankAccount with the given data and methods.
#include <iostream>
using namespace std;
class BankAccount {
    int accountNumber;
    string accountHolder;
    double balance;
public:
    // Set account details
    void setAccount(int accNo, string name, double bal) {
        accountNumber = accNo;
        accountHolder = name;
        balance = bal;
    }
    // Deposit money
    void deposit(double amount) {
        if (amount > 0)
            balance += amount;
    }
    // Withdraw money
    void withdraw(double amount) {
        if (amount > 0 && amount <= balance)
            balance -= amount;
        else
            cout << "Insufficient Balance\n";
    }
    // Display balance
    void displayBalance() {
        cout << "Account Number: " << accountNumber << endl;
        cout << "Account Holder: " << accountHolder << endl;
        cout << "Balance: " << balance << endl;
    }
};
```

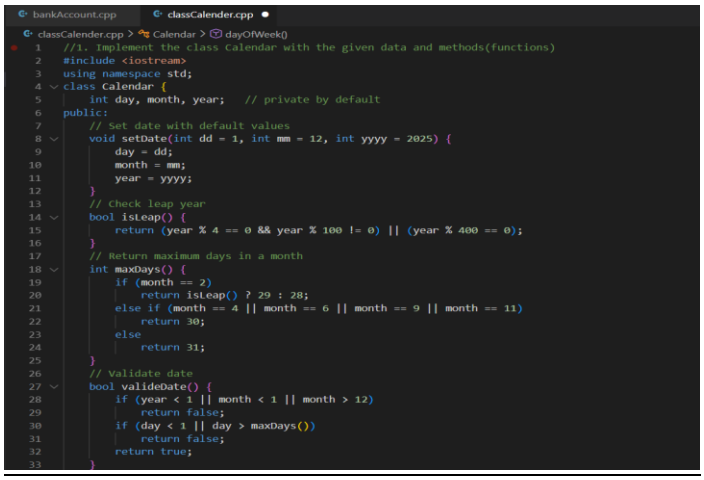
```cpp
int main() {
  BankAccount acc;
    acc.setAccount(101, "Rahul", 5000);
    acc.deposit(2000);
    acc.withdraw(1500);
    acc.displayBalance();

    return 0;
```

Output :

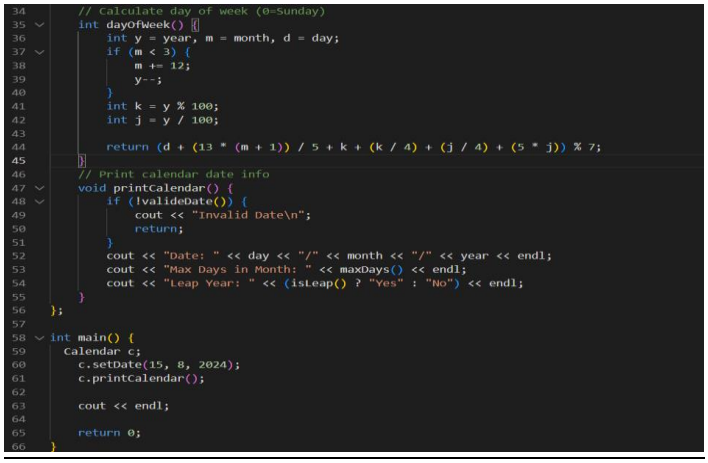


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ bankAccount.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Account Number: 101
Account Holder: Rahul
Balance: 5500
```

2. ***Code*** : Write a program to implement a Calendar class in C++ that stores date information and provides member functions to set the date (with default values), validate the date, check for leap year, determine the maximum days in a month, and display calendar-related details using object-oriented principles.

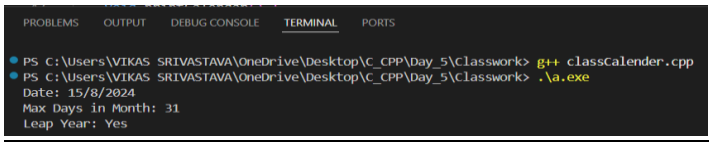


```cpp
//1. Implement the class Calendar with the given data and methods(functions)
#include <iostream>
using namespace std;
class Calendar {
    int day, month, year;   // private by default
public:
    // Set date with default values
    void setDate(int dd = 1, int mm = 12, int yyyy = 2025) {
        day = dd;
        month = mm;
        year = yyyy;
    }
    // Check leap year
    bool isLeap() {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }
    // Return maximum days in a month
    int maxDays() {
        if (month == 2)
            return isLeap() ? 29 : 28;
        else if (month == 4 || month == 6 || month == 9 || month == 11)
            return 30;
        else
            return 31;
    }
    // Validate date
    bool valideDate() {
        if (year < 1 || month < 1 || month > 12)
            return false;
        if (day < 1 || day > maxDays())
            return false;
        return true;
    }
```



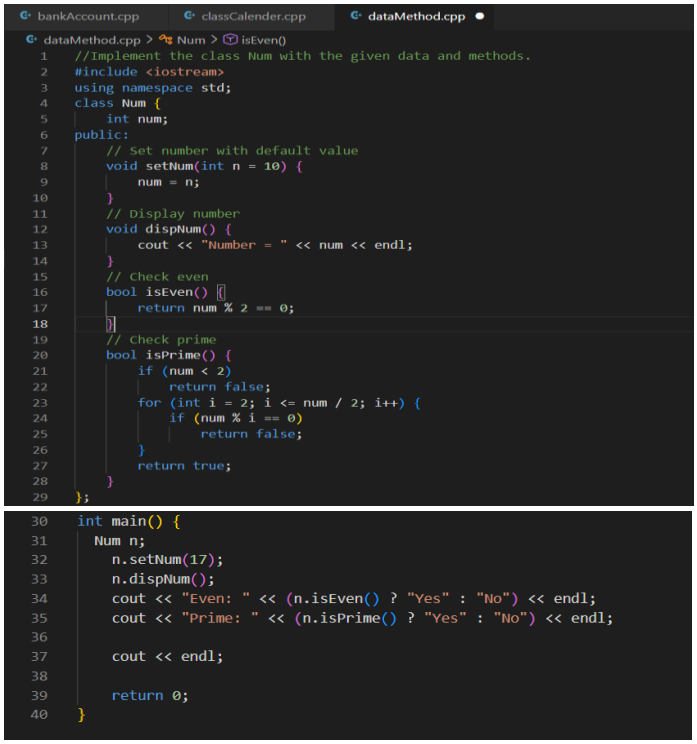
```cpp
    // Calculate day of week (0=Sunday)
    int dayOfWeek() {
        int y = year, m = month, d = day;
        if (m < 3) {
            m += 12;
            y--;
        }
        int k = y % 100;
        int j = y / 100;

        return (d + (13 * (m + 1)) / 5 + k + (k / 4) + (j / 4) + (5 * j)) % 7;
    }
    // Print calendar date info
    void printCalendar() {
        if (!valideDate()) {
            cout << "Invalid Date\n";
            return;
        }
        cout << "Date: " << day << "/" << month << "/" << year << endl;
        cout << "Max Days in Month: " << maxDays() << endl;
        cout << "Leap Year: " << (isLeap() ? "Yes" : "No") << endl;
    }
};

int main() {
  Calendar c;
    c.setDate(15, 8, 2024);
    c.printCalendar();

    cout << endl;

    return 0;
}
```

Output :



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ classCalender.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Date: 15/8/2024
Max Days in Month: 31
Leap Year: Yes
```

3. ***Code*** : Write a program to implement a Num class in C++ that stores a number and provides member functions to set and display the number, check whether it is even, and check whether it is a prime number, demonstrating basic object-oriented programming concepts.



```cpp
//Implement the class Num with the given data and methods.
#include <iostream>
using namespace std;
class Num {
    int num;
public:
    // Set number with default value
    void setNum(int n = 10) {
        num = n;
    }
    // Display number
    void dispNum() {
        cout << "Number = " << num << endl;
    }
    // Check even
    bool isEven() {
        return num % 2 == 0;
    }
    // Check prime
    bool isPrime() {
        if (num < 2)
            return false;
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0)
                return false;
        }
        return true;
    }
};
int main() {
  Num n;
    n.setNum(17);
    n.dispNum();
    cout << "Even: " << (n.isEven() ? "Yes" : "No") << endl;
    cout << "Prime: " << (n.isPrime() ? "Yes" : "No") << endl;

    cout << endl;

    return 0;
}
```

Output :

```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ dataMethod.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Number = 17
Even: No
Prime: Yes
```

4. ***Code :*** Write a program to demonstrate how a C-style string can be stored inside a class using a character array, initialized through a constructor, and processed using standard string functions like strcpy() and strlen() to display the string and its length.

```cpp
//program where char name[] is replaced with char *name here the string or name is a pointer to character array.

#include <iostream>
  #include <cstring>
  using namespace std;

  class Test{
      char name[30]; //c style string //char *name;
      int len;
  public:
      Test(const char arg[]="Some string") {
          strcpy(name, arg);
          len = strlen(arg);
      }
      void disp(){
          cout<<"Len: "<<len<<"\tname: "<<name<<endl;
      }
      ~Test(){       }
  };

  int main(){
      Test obj;
      obj.disp();
  }
```

Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ char.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Len: 11 name: Some string
```

5. ***Code*** : Write a program to define an Employee class in C++ with private data members and public member functions to accept employee details (ID, name, salary) and display them, demonstrating encapsulation and basic object-oriented programming concepts.

classOne.cpp

```cpp
#include <iostream>
using namespace std;

class Employee{
  private:
    int id;
    char name[25];
    double sal;
  public:
    void getDetails(){
      cout<<"Enter ID, name and Sal: ";
      cin>>id>>name>>sal;
    }
    void dispDetails(){
      cout<<"Employee Details ID: "<<id<<"\tName: "<<name<<"\tSal: "<<sal<<endl;
    }
};//blue print of an obj
int main(){
  Employee varOne;
  varOne.getDetails();
  varOne.dispDetails();
}
```

Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ classOne.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Enter ID, name and Sal: 56743 Vikas 50000
Employee Details ID: 56743      Name: Vikas     Sal: 50000
```

6. ***Code :*** Write a program to demonstrate different types of constructors in C++, including the default constructor, single-parameterized constructor, and copy constructor, showing how objects are created and initialized in different ways.

classOne.cpp | char.cpp | ctorFour.cpp

ctorFour.cpp

```cpp
#include <iostream>
using namespace std;

class Test{
  public:
    Test(){}
    Test(int x){}
    Test(const Test &){}
};

int main(){
  Test objTwo;
  Test obj=100; //single parameterized ctor
  Test objOne = obj; //copy ctor
  cout<<"In main()"<<endl;
}
```

Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ ctorFour.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
In main()
```

7. ***Code :*** Write a program to demonstrate the automatic invocation of a constructor and destructor in C++, where the constructor is called when an object is created and the destructor is called when the object goes out of scope at the end of the program.

```cpp
#include <iostream>
using namespace std;

class Test{
  public:
    Test() {
      cout<<"Test() ---> Constructor"<<endl;
    }
    ~Test(){
      cout<<"~Test() ----> Destructor"<<endl;
    }
};

int main(){
  Test obj;
  cout<<"In main()"<<endl;
}
```

Output :

```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ ctorOne.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Test() ---> Constructor
In main()
~Test() ----> Destructor
```

8. ***Code :*** Write a program to demonstrate the use of default and parameterized constructors in an Employee class to initialize employee details and display them, highlighting constructor overloading and object initialization in C++.

```cpp
using namespace std;
class Employee{
  private:
    int id;
    char name[25];
    double sal;
  public:
    Employee(){
      cout<<"Enter ID, name and Sal: ";
      cin>>id>>name>>sal;
    }
    Employee(int n, const char na[], double s){
      id = n;
      strcpy(name, na);
      sal = s;
    }
    void dispDetails(){
      cout<<"Employee Details ID: "<<id<<"\tName: "<<name<<"\tSal: "<<sal<<endl;
    }
};//blue print of an object
int main(){
  Employee varOne(10001, "Vikas Srivastava", 12000.00);
  varOne.dispDetails();
}
```

Output :

```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ ctorTwo.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Employee Details ID: 10001      Name: Vikas Srivastava  Sal: 12000
```

9. ***Code :*** Write a program to demonstrate call by reference in C++ using pointers, where a function modifies the value of a variable defined in the calling function by accessing its memory address.

ctorTwo.cpp | *funCallAddr.cpp*

funCallAddr.cpp > ...

```cpp
#include <iostream>
using namespace std;

void fun(int *);

int main(){
  int var = 10;//variable of main()
  cout<<"Before Var: "<<var<<endl;
  fun(&var);
  cout<<"After Var: "<<var<<endl;
}

void fun(int *arg) //formal arguments
{
  *arg += 100;  //*arg = *arg + 100
}
```

Output :

```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ funCallAddr.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Before Var: 10
After Var: 110
```

10. ***Code :*** Write a program to demonstrate call by reference in C++ using reference variables, where a function directly modifies the original variable without using pointers.

ctorTwo.cpp | *funCallRef.cpp*

funCallRef.cpp > ...

```cpp
#include <iostream>
using namespace std;

void fun(int &);

int main(){
  int var = 10;//variable of main()
  cout<<"Before Var: "<<var<<endl;
  fun(var);
  cout<<"After Var: "<<var<<endl;
}

void fun(int &arg) //formal arguments
{
  arg+=100;
}
```

Output :

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ funCallRef.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Before Var: 10
After Var: 110
```

11. Code : Write a program to demonstrate function overloading in C++, where multiple functions with the same name but different parameter lists are defined and the appropriate function is selected at compile time based on the arguments passed.

ctorTwo.cpp | funOne.cpp

funOne.cpp > fun(double)

```cpp
#include <iostream>
using namespace std;
void fun();
void fun(int);
void fun(double);
void fun(int,int);
void fun(double, double);
int main(){
  fun();
  fun(10);
  fun(123.345);
  fun(100,200);
  fun(234.345, 453.234);
}
void fun(){
  cout<<"void fun()"<<endl;
}
void fun(int){
  cout<<"void fun(int)"<<endl;
}
void fun(double){
  cout<<"void fun(double)"<<endl;
}
void fun(int,int){
  cout<<"void fun(int, int)"<<endl;
}
void fun(double, double){
  cout<<"void fun(double, double)"<<endl;
}
```

Output :

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ funOne.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
void fun()
void fun(int)
void fun(double)
void fun(int, int)
void fun(double, double)
```

12. Code : Write a program to demonstrate function overloading in C++ and show how explicit type casting affects function selection, where casting a floating-point value to int forces the compiler to call the integer version of the overloaded function.

ctorTwo.cpp | funOne.cpp | *funOverTwo.cpp*

funOverTwo.cpp > ...

```cpp
#include <iostream>
using namespace std;

void fun(int);
void fun(float);


int main(){
  fun((int)10.234);
}

void fun(int arg){
  cout<<"Int type: arg  "<<arg<<endl;
}
void fun(float arg){
  cout<<"float type: arg  "<<arg<<endl;
}

```

Output :

```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> g++ funOverTwo.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_5\Classwork> .\a.exe
Int type: arg  10
```