**CPP ASSIGNMENT**

|  |  |
| --- | --- |
| **FULL NAME** | **Harshada Arvind Patil** |
| **ROLL NUMBER** | **258144** |
| **PRN** | **250841220061** |

**DAY 1 - LAB 1**

1. **write program to test Hello World.**

#include <iostream>

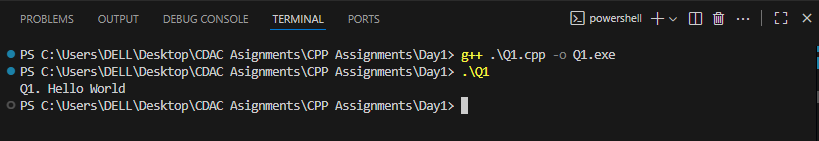
using namespace std;

int main() {

cout << "Q1. Hello World" << endl;

return 0;

}



1. **write program to test Hello World.**

#include <iostream>

using namespace std;

int main() {

    int a, b;

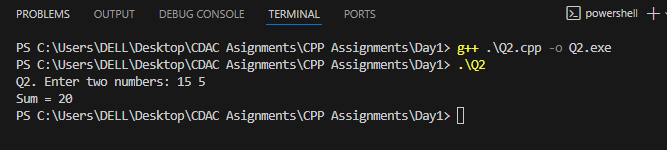
    cout << "Q2. Enter two numbers: ";

    cin >> a >> b;

    cout << "Sum = " << a + b << endl;

    return 0;

}



1. **Write a program to swap two numbers.**

#include <iostream>

using namespace std;

int main() {

    int x, y, a, b;

    cout << "Q3. Enter two numbers: ";

    cin >> x >> y;

    cout << "Before swap: x = " << x << " y = " << y << endl;

    int temp = x;

    x = y;

    y = temp;

    cout << "After swap: x = " << x << " y = " << y << endl;

    cout << "Enter two numbers (without 3rd variable): ";

    cin >> a >> b;

    cout << "Before swap: a = " << a << " b = " << b << endl;

    a = a + b;

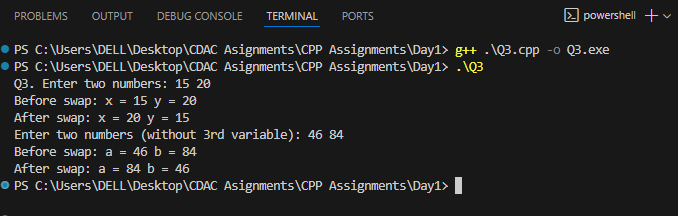
    b = a - b;

    a = a - b;

    cout << "After swap: a = " << a << " b = " << b << endl;

    return 0;

}

****

1. **Write a program to accept an integer and check if it is even or odd.**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Q4. Enter a number: ";

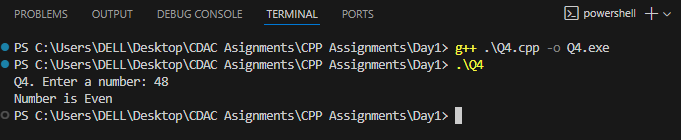
cin >> n;

if (n % 2 == 0) cout << "Number is Even" << endl;

else cout << "Number is Odd" << endl;

return 0;

}



1. **Write a program to accept a number and check if it is divisible by 5 and 7.**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Q5. Enter a number: ";

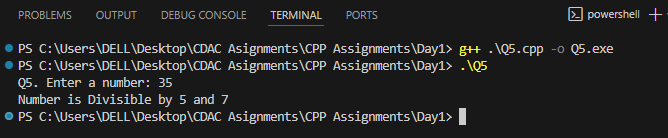
cin >> n;

if (n % 5 == 0 && n % 7 == 0) cout << "Divisible by 5 and 7" << endl;

else cout << "Not divisible by 5 and 7" << endl;

return 0;

}



1. **Write a program, which accepts annual basic salary of an employee and calculates and displays the**

**Income tax as per the following rules.**

**Basic: < 1, 50,000 Tax = 0**

**1, 50,000 to 3,00,000 Tax = 20%**

**> 3,00,000 Tax = 30%**

#include <iostream>

using namespace std;

int main() {

    double salary, tax=0;

    cout << "Q6. Enter annual salary: ";

    cin >> salary;

    if (salary < 150000) tax = 0;

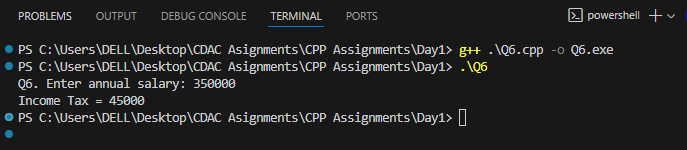
    else if (salary <= 300000) tax = 0.2 \* (salary - 150000);

    else tax = 0.3 \* (salary - 300000) + 0.2 \* 150000;

    cout << "Income Tax = " << tax << endl;

    return 0;

}



1. **Accept a lowercase character from the user and check whether the character is a vowel or consonant. (Hint: a, e, i, o, u are vowels)**

#include <iostream>

using namespace std;

int main() {

    char ch;

    cout << "Q7. Enter a lowercase character: ";

    cin >> ch;

    if (ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')

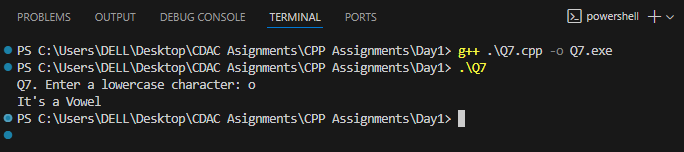
        cout << "Vowel" << endl;

    else

        cout << "Consonant" << endl;

    return 0;

}



1. **Write a program to input angles of a triangle and check whether triangle is valid or not.**

#include <iostream>

using namespace std;

int main() {

    int a, b, c;

    cout << "Q8. Enter three angles: ";

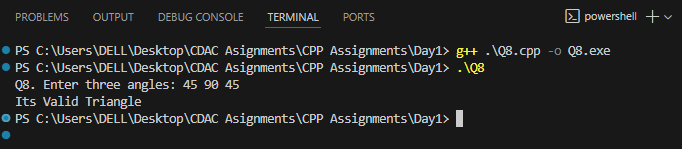
    cin >> a >> b >> c;

    if (a+b+c == 180) cout << "Valid Triangle" << endl;

    else cout << "Invalid Triangle" << endl;

    return 0;

}



1. **Write a program to find factorial of a given number. ( Ex: no5 fact=5\*4\*3\*2\*1=120 )**

#include <iostream>

using namespace std;

int main() {

    int n, fact=1;

    cout << "Q9. Enter number: ";

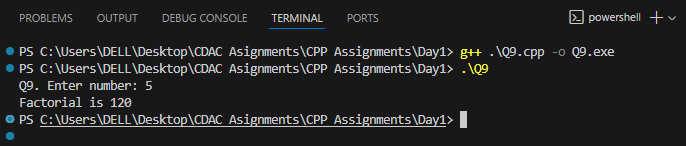
    cin >> n;

    for (int i=1; i<=n; i++) fact \*= i;

    cout << "Factorial is " << fact << endl;

    return 0;

}



1. **Write a program to find m to the power n. m=3 and n=4 so 3\*3\*3\*3**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    int m, n;

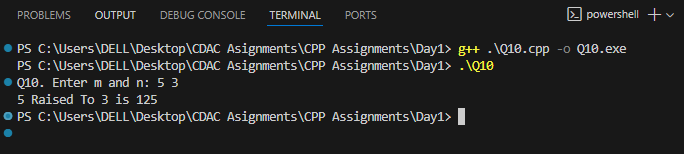
    cout << "Q10. Enter m and n: ";

    cin >> m >> n;

    cout << m << "^" << n << " = " << pow(m,n) << endl;

    return 0;

}



1. **Check if number is a prime number or not.**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    int n;

    cout << "Q11. Enter number: ";

    cin >> n;

    bool prime = true;

    if (n < 2) prime = false;

    for (int i=2; i<=sqrt(n); i++)

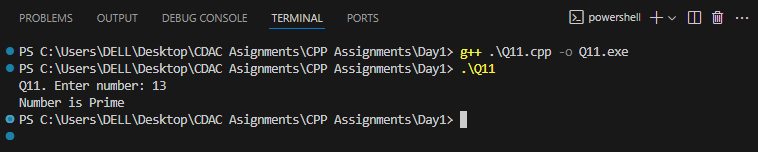
        if (n % i == 0) prime = false;

    if (prime) cout << "Number is Prime" << endl;

    else cout << "Number is Not Prime" << endl;

    return 0;

}



1. **Sum of series : 1+2+3+….+n**

#include <iostream>

using namespace std;

int main() {

    int n, sum=0;

    cout << "Q12. Enter n: ";

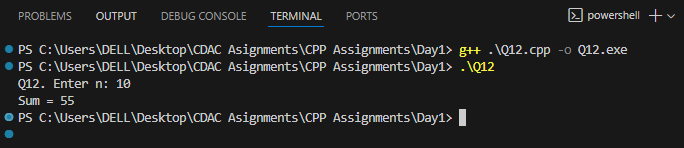
    cin >> n;

    for(int i=1;i<=n;i++) sum+=i;

    cout << "Sum = " << sum << endl;

    return 0;

}



1. **Check whether the number is palindrome or not?**

#include <iostream>

using namespace std;

int main() {

    int n, rev=0, orig;

    cout << "Q13. Enter number: ";

    cin >> n;

    orig = n;

    while(n>0){

        rev = rev\*10 + n%10;

        n /= 10;

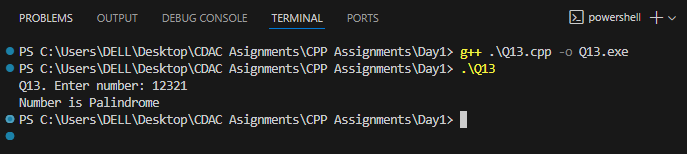
    }

    if(orig==rev) cout<<"Number is Palindrome"<<endl;

    else cout<<"Number is Not Palindrome"<<endl;

    return 0;

}



1. Write a program to find sum of all even and odd numbers between 1 to n.

#include <iostream>

using namespace std;

int main() {

    int n, even=0, odd=0;

    cout << "Q14. Enter n: ";

    cin >> n;

    for(int i=1;i<=n;i++){

        if(i%2==0) even+=i;

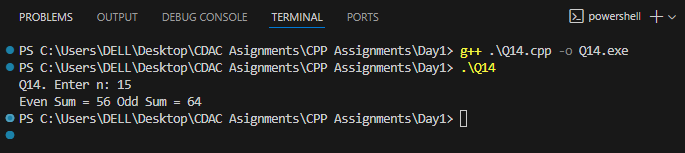
        else odd+=i;

    }

    cout<<"Even Sum = "<<even<<" Odd Sum = "<<odd<<endl;

    return 0;

}



1. **Write a program to enter a number and print its reverse.**

#include <iostream>

using namespace std;

int main() {

    int n, rev=0;

    cout << "Q15. Enter number: ";

    cin >> n;

    while(n>0){

        rev = rev\*10 + n%10;

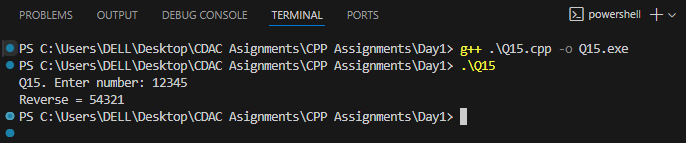
        n /= 10;

    }

    cout<<"Reverse = "<<rev<<endl;

    return 0;

}



1. **Write a program to print all Prime numbers between 1 to n.**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    int n;

    cout << "Q16. Enter n: ";

    cin >> n;

    for(int i=2;i<=n;i++){

        bool prime = true;

        for(int j=2;j<=sqrt(i);j++)

            if(i%j==0) prime=false;

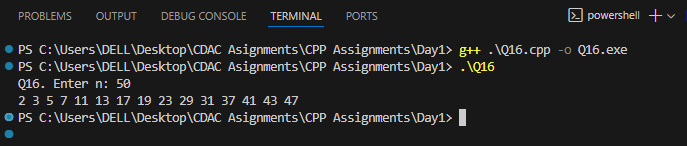
        if(prime) cout<<i<<" ";

    }

    cout<<endl;

    return 0;

}



1. **Write a program to check entered number is Armstrong number or not.**

#include <iostream>

using namespace std;

int main() {

    int n, orig, sum=0;

    cout << "Q17. Enter number: ";

    cin >> n;

    orig = n;

    while(n>0){

        int d = n % 10;

        sum += d\*d\*d;

        n /= 10;

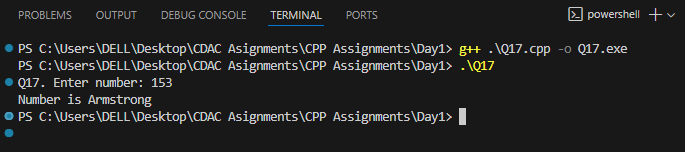
    }

    if(sum==orig) cout<<"Number is Armstrong"<<endl;

    else cout<<"Number is Not Armstrong"<<endl;

    return 0;

}



1. **Write a program to find greatest of three numbers using nested if-else.**

#include <iostream>

using namespace std;

int main() {

    int a,b,c;

    cout<<"Q18. Enter three numbers: ";

    cin>>a>>b>>c;

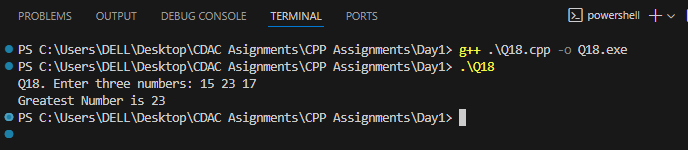
    if(a>=b && a>=c) cout<<"Greatest Number is "<<a<<endl;

    else if(b>=a && b>=c) cout<<"Greatest Number is "<<b<<endl;

    else cout<<"Greatest Number is "<<c<<endl;

    return 0;

}



1. **Create menu driven program for Pizza Shop. And display total amount.**

#include <iostream>

using namespace std;

int main() {

    int choice, qty, total=0;

    do {

        cout<<"Q19. Menu: 1.Pizza(100) 2.Burger(50) 3.Exit"<<endl;

        cin>>choice;

        if(choice==1){ cout<<"Qty: ";cin>>qty; total+=100\*qty; }

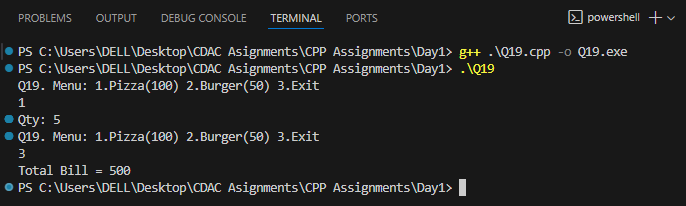
        else if(choice==2){ cout<<"Qty: ";cin>>qty; total+=50\*qty; }

    } while(choice!=3);

    cout<<"Total Bill = "<<total<<endl;

    return 0;

}



1. **Accept a single digit from the user and display it in words. For example, if digit entered is 9, display Nine.**

#include <iostream>

#include <string>

using namespace std;

int main() {

    int d;

    cout<<"Q20. Enter digit: ";

    cin>>d;

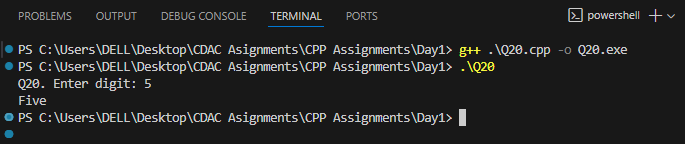
    string words[]={"Zero","One","Two","Three","Four","Five","Six","Seven","Eight","Nine"};

    if(d>=0 && d<=9) cout<<words[d]<<endl;

    else cout<<"Its a Not a digit"<<endl;

    return 0;

}



1. **Write a program, which accepts two integers and an operator as a character (+ - \* / ), performs the corresponding operation and displays the result.**

#include <iostream>

using namespace std;

int main() {

    int a,b;

    char op;

    cout<<"Q21. Enter a b and operator(+ - \* /): ";

    cin>>a>>b>>op;

    switch(op){

        case '+': cout<<"Result="<<a+b; break;

        case '-': cout<<"Result="<<a-b; break;

        case '\*': cout<<"Result="<<a\*b; break;

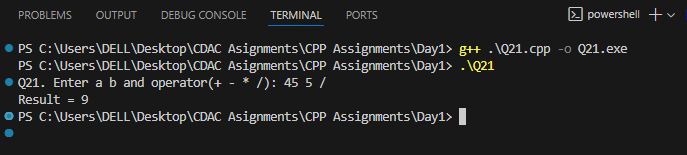
        case '/': if(b!=0) cout<<"Result="<<a/b; else cout<<"Div by 0"; break;

        default: cout<<"Invalid operator";

    }

    return 0;

}



**DAY 1 - LAB 2**

1. **Write a program that accepts numbers continuously as long as the number is positive and prints the sum of the given numbers.**

#include <iostream>

using namespace std;

int main() {

    int n, sum = 0;

    cout << "Enter positive numbers (negative to stop):" << endl;

    while (true) {

        cin >> n;

        if (n < 0) break;

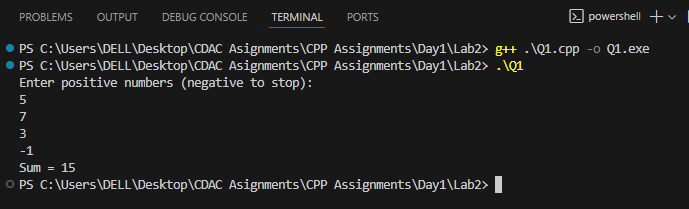
        sum += n;

    }

    cout << "Sum = " << sum << endl;

    return 0;

}



1. **Write a program to accept two integers x and n and compute x raised to n.**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    int x, n;

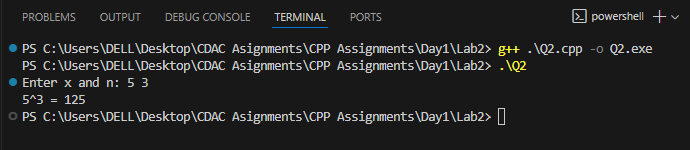
    cout << "Enter x and n: ";

    cin >> x >> n;

    cout << x << "^" << n << " = " << pow(x, n) << endl;

    return 0;

}



1. **Write a program to accept a character, an integer n and display the next n characters.**

#include <iostream>

using namespace std;

int main() {

    char ch;

    int n;

    cout << "Enter a character and an integer n: ";

    cin >> ch >> n;

    cout << "Next " << n << " characters: ";

    for (int i = 1; i <= n; i++) {

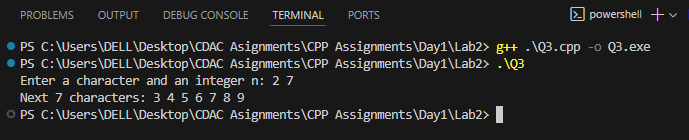
        cout << char(ch + i) << " ";

    }

    cout << endl;

    return 0;

}



1. **Write a program to calculate factorial of a number. For e.g. factorial of 5 = 5! = 5 \*4\*3\*2\*1 = 120**

#include <iostream>

using namespace std;

int main() {

    int n, fact = 1;

    cout << "Enter a number: ";

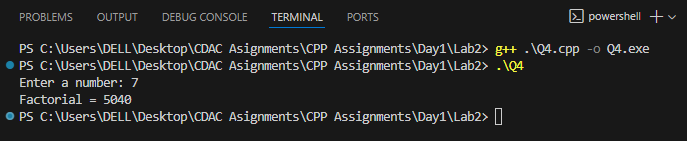
    cin >> n;

    for (int i = 1; i <= n; i++) fact \*= i;

    cout << "Factorial = " << fact << endl;

    return 0;

}



1. **Write a program to calculate factors of a given number.**

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter a number: ";

    cin >> n;

    cout << "Factors: ";

    for (int i = 1; i <= n; i++) {

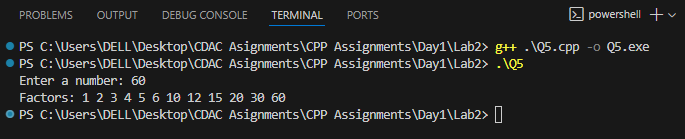
        if (n % i == 0) cout << i << " ";

    }

    cout << endl;

    return 0;

}



1. **Accept two numbers and calculate GCD of them.**

#include <iostream>

using namespace std;

int main() {

    int a, b;

    cout << "Enter two numbers: ";

    cin >> a >> b;

    while (b != 0) {

        int temp = b;

        b = a % b;

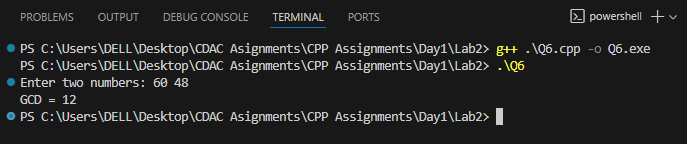
        a = temp;

    }

    cout << "GCD = " << a << endl;

    return 0;

}

****

1. **Write a menu driven program to do following operations :**

**a) Compute area of circle b) Compute area of rectangle**

**c) Compute area of triangle d) Exit**

**Display menu, ask choice to the user, depending on choice accept the parameters and perform the**

**operation. Continue this process until user selects exit option.**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    int choice;

    do {

        cout << "\nMenu:\n";

        cout << "1. Area of Circle\n";

        cout << "2. Area of Rectangle\n";

        cout << "3. Area of Triangle\n";

        cout << "4. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        if (choice == 1) {

            double r;

            cout << "Enter radius: ";

            cin >> r;

            cout << "Area of Circle = " << 3.14159 \* r \* r << endl;

        }

        else if (choice == 2) {

            double l, w;

            cout << "Enter length and width: ";

            cin >> l >> w;

            cout << "Area of Rectangle = " << l \* w << endl;

        }

        else if (choice == 3) {

            double b, h;

            cout << "Enter base and height: ";

            cin >> b >> h;

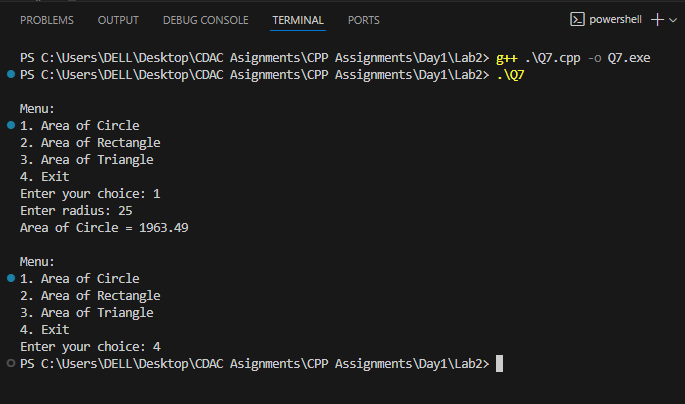
            cout << "Area of Triangle = " << 0.5 \* b \* h << endl;

        }

    } while (choice != 4);

    return 0;

}



1. **Write a program to print all prime numbers between 1 to n**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    int n;

    cout << "Enter n: ";

    cin >> n;

    cout << "Prime numbers up to " << n << ": ";

    for (int i = 2; i <= n; i++) {

        bool prime = true;

        for (int j = 2; j <= sqrt(i); j++) {

            if (i % j == 0) {

                prime = false;

                break;

            }

        }

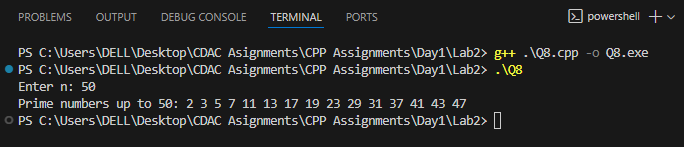
        if (prime) cout << i << " ";

    }

    cout << endl;

    return 0;

}



**DAY 1 - LAB 3**

1. **Write a program to create an array of integers and perform following operations on that array like finding the sum, average, maximum and minimum number in that array. Accept the numbers of the array from user.**

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n], sum = 0, max, min;

    cout << "Enter " << n << " elements: ";

    for(int i=0; i<n; i++) {

        cin >> arr[i];

        sum += arr[i];

    }

    max = min = arr[0];

    for(int i=1; i<n; i++) {

        if(arr[i] > max) max = arr[i];

        if(arr[i] < min) min = arr[i];

    }

    cout << "Sum = " << sum << endl;

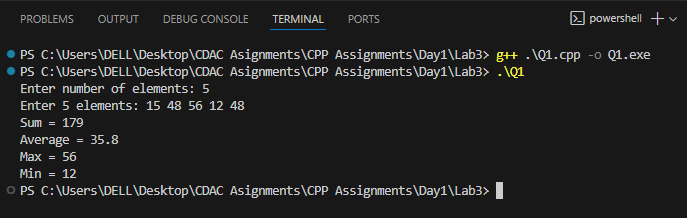
    cout << "Average = " << (double)sum/n << endl;

    cout << "Max = " << max << endl;

    cout << "Min = " << min << endl;

    return 0;

}



1. **Write a program to Accept a number and display its sum of digits. Ex= 568 -> 5+6+8**

#include <iostream>

using namespace std;

int main() {

    int n, sum = 0;

    cout << "Enter a number: ";

    cin >> n;

    int temp = n;

    while(n > 0) {

        sum += n % 10;

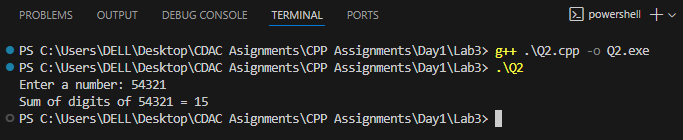
        n /= 10;

    }

    cout << "Sum of digits of " << temp << " = " << sum << endl;

    return 0;

}



1. **Write a program to find sum of all even and odd numbers between 1 to n.**

#include <iostream>

using namespace std;

int main() {

    int n, evenSum=0, oddSum=0;

    cout << "Enter n: ";

    cin >> n;

    for(int i=1; i<=n; i++) {

        if(i % 2 == 0) evenSum += i;

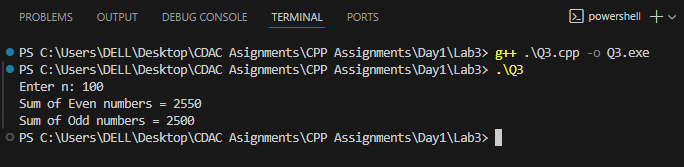
        else oddSum += i;

    }

    cout << "Sum of Even numbers = " << evenSum << endl;

    cout << "Sum of Odd numbers = " << oddSum << endl;

}



1. **Write a program to print all Prime numbers between 1 to n.**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    int n;

    cout << "Enter n: ";

    cin >> n;

    cout << "Prime numbers up to " << n << ": ";

    for(int i=2; i<=n; i++) {

        bool prime = true;

        for(int j=2; j<=sqrt(i); j++) {

            if(i % j == 0) {

                prime = false;

                break;

            }

        }

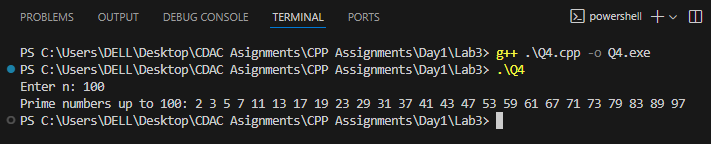
        if(prime) cout << i << " ";

    }

    cout << endl;

    return 0;

}



1. **Write a program to accept array from user. Accept number from user and search number is present in array or not.**

#include <iostream>

using namespace std;

int main() {

    int n, key;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter " << n << " elements: ";

    for(int i=0; i<n; i++) cin >> arr[i];

    cout << "Enter number to search: ";

    cin >> key;

    bool found = false;

    for(int i=0; i<n; i++) {

        if(arr[i] == key) {

            found = true;

            break;

        }

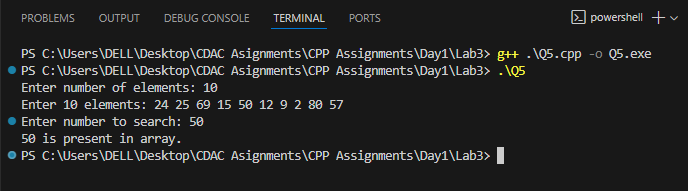
    }

    if(found) cout << key << " is present in array." << endl;

    else cout << key << " is not present in array." << endl;

    return 0;

}



1. **Write a program to print following pattern.**

**\***

**\* \***

**\* \* \***

**\* \* \* \***

**\* \* \* \* \***

#include <iostream>

using namespace std;

int main()

{

    int n;

    cout << "Enter Number of Rows = ";

    cin >> n;

    cout << endl;

    for (int i = 1; i <= n; i++)

    {

        for (int j = 1; j <= i; j++)

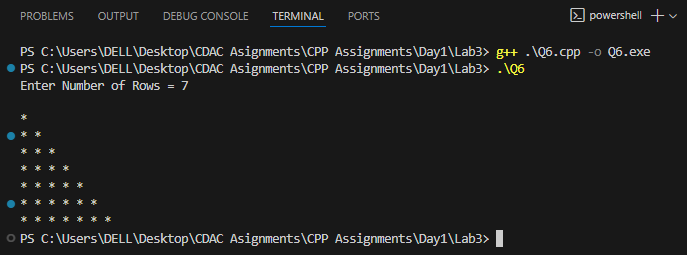
        { cout << "\* "; }

        cout << endl;

    }

    return 0;

}



**DAY 2 - LAB 1**

1. **Array Input & Output, Take `n` elements from the user and display them.**

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter " << n << " elements: ";

    for(int i=0;i<n;i++)

        cin >> arr[i];

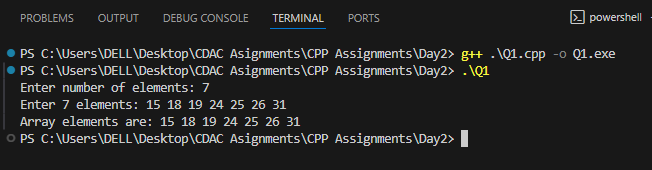
    cout << "Array elements are: ";

    for(int i=0;i<n;i++)

        cout << arr[i] << " ";

    return 0;

}



1. **Sum & Average of Array, Input marks of `n` students, find total & average.**

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter " << n << " elements: ";

    for(int i=0;i<n;i++)

        cin >> arr[i];

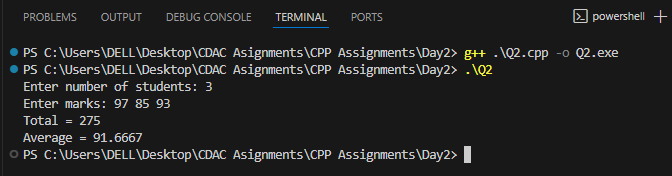
    cout << "Array elements are: ";

    for(int i=0;i<n;i++)

        cout << arr[i] << " ";

    return 0;

}



1. **Find Maximum and Minimum, Input array and print the largest & smallest element.**

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter elements: ";

    for(int i=0;i<n;i++) cin >> arr[i];

    int min = arr[0], max = arr[0];

    for(int i=1;i<n;i++) {

        if(arr[i] < min) min = arr[i];

        if(arr[i] > max) max = arr[i];

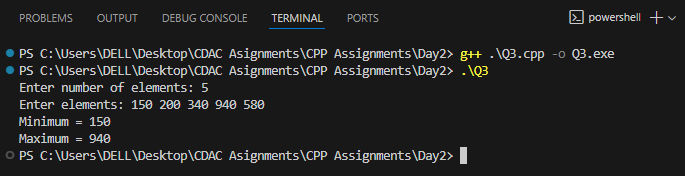
    }

    cout << "Minimum = " << min << endl;

    cout << "Maximum = " << max << endl;

    return 0;

}



1. **Search an Element (Linear Search), Input an array and search if a number exists**

#include <iostream>

using namespace std;

int main() {

    int n, key;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter elements: ";

    for(int i=0;i<n;i++) cin >> arr[i];

    cout << "Enter element to search: ";

    cin >> key;

    bool found = false;

    for(int i=0;i<n;i++) {

        if(arr[i] == key) {

            cout << key << " found at index " << i << endl;

            found = true;

            break;

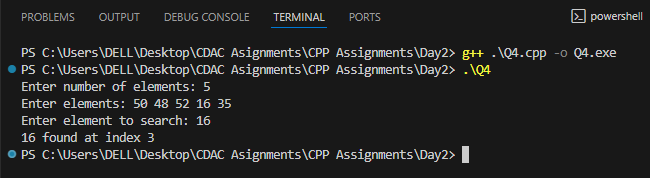
        }

    }

    if(!found) cout << key << " not found." << endl;

    return 0;

}



1. **Reverse Array, Reverse an array and display the result.**

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter elements: ";

    for(int i=0;i<n;i++) cin >> arr[i];

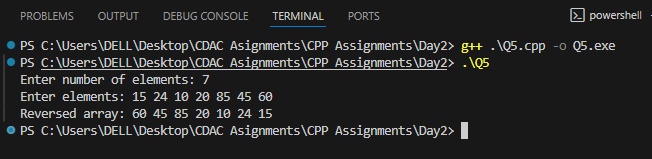
    cout << "Reversed array: ";

    for(int i=n-1;i>=0;i--)

        cout << arr[i] << " ";

    return 0;

}



1. **Count Even And Odd Numbers, Count how many even and odd numbers are in the array**

#include <iostream>

using namespace std;

int main() {

    int n, even=0, odd=0;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter elements: ";

    for(int i=0;i<n;i++) {

        cin >> arr[i];

        if(arr[i] % 2 == 0) even++;

        else odd++;

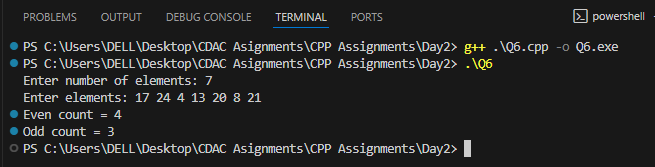
    }

    cout << "Even count = " << even << endl;

    cout << "Odd count = " << odd << endl;

    return 0;

}



1. **Sort Array (Ascending & Descending), Implement a simple bubble sort**

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter elements: ";

    for(int i=0;i<n;i++) cin >> arr[i];

    for(int i=0;i<n-1;i++) {

        for(int j=0;j<n-i-1;j++) {

            if(arr[j] > arr[j+1]) {

                int temp = arr[j];

                arr[j] = arr[j+1];

                arr[j+1] = temp;

            }

        }

    }

    cout << "Sorted ascending: ";

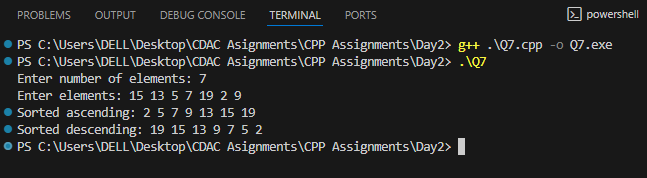
    for(int i=0;i<n;i++) cout << arr[i] << " ";

    cout << "\nSorted descending: ";

    for(int i=n-1;i>=0;i--) cout << arr[i] << " ";

    return 0;

}



1. **Second Largest Element, Find the second largest number without sorting.**

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter elements: ";

    for(int i=0;i<n;i++) cin >> arr[i];

    int max1 = arr[0], max2 = -1e9;

    for(int i=1;i<n;i++) {

        if(arr[i] > max1) {

            max2 = max1;

            max1 = arr[i];

        } else if(arr[i] > max2 && arr[i] < max1) {

            max2 = arr[i];

        }

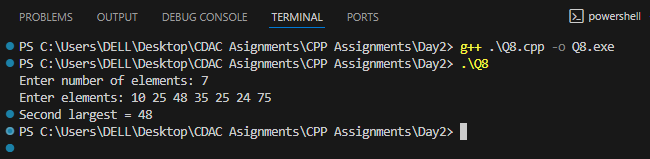
    }

    if(max2 == -1e9) cout << "No second largest element.\n";

    else cout << "Second largest = " << max2 << endl;

    return 0;

}



1. Frequency of Each Element, Count how many times each element occurs.

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n], freq[n];

    cout << "Enter elements: ";

    for(int i=0;i<n;i++) {

        cin >> arr[i];

        freq[i] = -1;

    }

    for(int i=0;i<n;i++) {

        if(freq[i] != 0) {

            int count = 1;

            for(int j=i+1;j<n;j++) {

                if(arr[i] == arr[j]) {

                    count++;

                    freq[j] = 0;

                }

            }

            freq[i] = count;

        }

    }

    cout << "Frequency of elements:\n";

    for(int i=0;i<n;i++) {

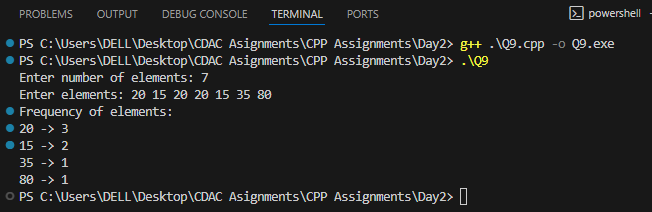
        if(freq[i] > 0)

            cout << arr[i] << " -> " << freq[i] << endl;

    }

    return 0;

}



1. **Merge Two Array, Take two arrays and merge them into a third array.**

#include <iostream>

using namespace std;

int main()

{

    int n1, n2;

    cout << "Enter size of first array: ";

    cin >> n1;

    int arr1[n1];

    cout << "Enter elements of first array: ";

    for (int i = 0; i < n1; i++)

        cin >> arr1[i];

    cout << "Enter size of second array: ";

    cin >> n2;

    int arr2[n2];

    cout << "Enter elements of second array: ";

    for (int i = 0; i < n2; i++)

        cin >> arr2[i];

    int merged[n1 + n2];

    for (int i = 0; i < n1; i++)

        merged[i] = arr1[i];

    for (int i = 0; i < n2; i++)

        merged[n1 + i] = arr2[i];

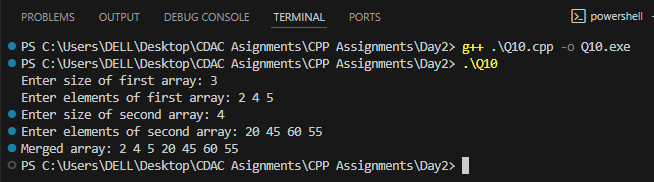
    cout << "Merged array: ";

    for (int i = 0; i < n1 + n2; i++)

        cout << merged[i] << " ";

    return 0;

}



**DAY 2 - LAB 2**

1. **Write a program using inline functions to calculate:**

**Area of a square (side × side)**

**Area of a rectangle (length × breadth)**

**Area of a circle (π × r × r, use π = 3.14159)**

#include <iostream>

using namespace std;

inline float areaSquare(float side) {

    return side \* side;

}

inline float areaRectangle(float length, float breadth) {

    return length \* breadth;

}

inline float areaCircle(float r) {

    return 3.14159f \* r \* r;

}

int main() {

    float side, length, breadth, radius;

    cout << "Enter side of square: "; cin >> side;

    cout << "Area of square = " << areaSquare(side) << endl;

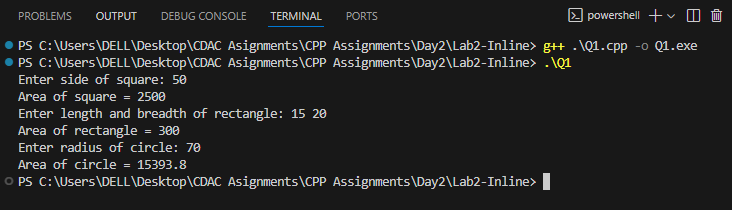
    cout << "Enter length and breadth of rectangle: "; cin >> len >> bread;

    cout << "Area of rectangle = " << areaRectangle(len, bread) << endl;

    cout << "Enter radius of circle: "; cin >> radius;

    cout << "Area of circle = " << areaCircle(radius) << endl;

    return 0; }



**DAY 3**

1. **Write a program Write a program to create student class with data members rollno, marks1,mark2,mark3. Accept data (acceptInfo()) and display using display member function. Also display total,percentage and grade.**

#include <iostream>

using namespace std;

class Student {

    int rollno;

    int marks1, marks2, marks3;

public:

    void acceptInfo() {

        cout << "Enter Roll No: ";

        cin >> rollno;

        cout << "Enter Marks in 3 Subjects: ";

        cin >> marks1 >> marks2 >> marks3;

    }

    void display() {

        int total = marks1 + marks2 + marks3;

        float percentage = total / 3.0f;

        char grade;

        if (percentage >= 75) grade = 'A';

        else if (percentage >= 60) grade = 'B';

        else if (percentage >= 50) grade = 'C';

        else grade = 'F';

        cout << "\nRoll No: " << rollno;

        cout << "\nMarks: " << marks1 << ", " << marks2 << ", " << marks3;

        cout << "\nTotal: " << total;

        cout << "\nPercentage: " << percentage << "%";

        cout << "\nGrade: " << grade << endl;

    }

};

int main() {

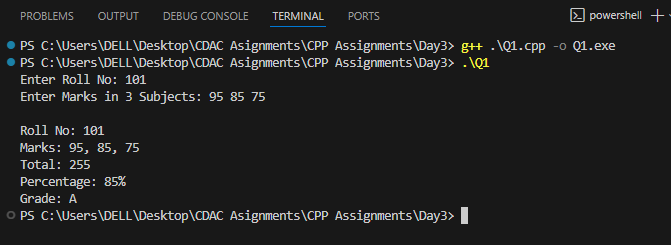
    Student s;

    s.acceptInfo();

    s.display();

    return 0;

}



1. **Create a class Person with data members as name, age, city. Write getters and setters for all the data members. Also add the display function. Create Default and Parameterized constructors. Create the object of this class in main method and invoke all the methods in that class.**

#include <iostream>

using namespace std;

class Person

{

    string name, city;

    int age;

public:

    Person()

    {

        name = "Unknown";

        age = 0;

        city = "NA";

    }

    Person(string n, int a, string c)

    {

        name = n;

        age = a;

        city = c;

    }

    void setName(string n) { name = n; }

    void setAge(int a) { age = a; }

    void setCity(string c) { city = c; }

    string getName() { return name; }

    int getAge() { return age; }

    string getCity() { return city; }

    void display()

    {

        cout << "\nName: " << name << "\nAge: " << age << "\nCity: " << city << endl;

    }

};

int main()

{

    Person p1;

    Person p2("Tejas", 22, "Pune");

    p1.display();

    p2.display();

    p1.setName("Ravi");

    p1.setAge(25);

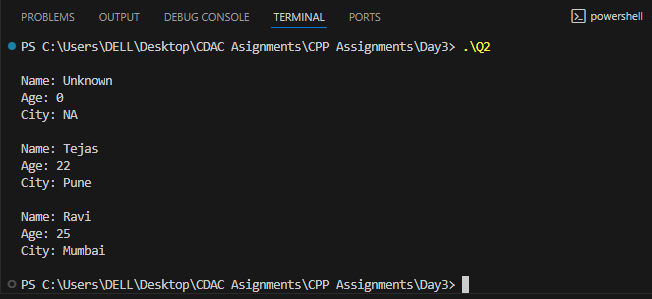
    p1.setCity("Mumbai");

    p1.display();

    cout << "" <<endl;

    return 0;

}



1. **Create a class Date with data members as dd, mm, yy. Write getters and setters for all the data members. Also add the display function. Create Default and Parameterized constructors. Create the object of this class in main method and invoke all the methods in that class.**

#include <iostream>

using namespace std;

class Date {

    int dd, mm, yy;

public:

    Date() { dd = mm = yy = 0; }

    Date(int d, int m, int y) { dd = d; mm = m; yy = y; }

    void setDay(int d) { dd = d; }

    void setMonth(int m) { mm = m; }

    void setYear(int y) { yy = y; }

    int getDay() { return dd; }

    int getMonth() { return mm; }

    int getYear() { return yy; }

    void display() {

        cout << "Date: " << dd << "/" << mm << "/" << yy << endl;

    }

};

int main() {

    Date d1;

    Date d2(23, 8, 2025);

    d1.display();

    d2.display();

    d1.setDay(1);

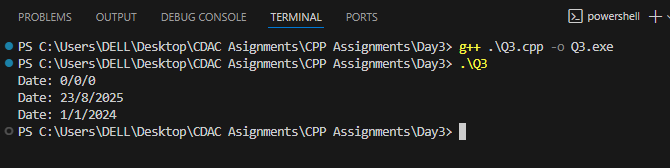
    d1.setMonth(1);

    d1.setYear(2024);

    d1.display();

    return 0;

}



1. **Create a class Book with data members as bname,id,author,price. Write getters and setters for all the data members. Also add the display function. Create Default and Parameterized constructors. Create the object of this class in main method and invoke all the methods in that class**

#include <iostream>

using namespace std;

class Book {

    string bname, author;

    int id;

    float price;

public:

    Book() { bname = "NA"; id = 0; author = "NA"; price = 0; }

    Book(string bn, int i, string a, float p) {

        bname = bn; id = i; author = a; price = p;

    }

    void setBname(string bn) { bname = bn; }

    void setId(int i) { id = i; }

    void setAuthor(string a) { author = a; }

    void setPrice(float p) { price = p; }

    string getBname() { return bname; }

    int getId() { return id; }

    string getAuthor() { return author; }

    float getPrice() { return price; }

    void display() {

        cout << "\nBook Name: " << bname << "\nID: " << id

             << "\nAuthor: " << author << "\nPrice: " << price << endl;

    }

};

int main() {

    Book b1;

    Book b2("C++ Programming", 101, "Bjarne Stroustrup", 499.50);

    b1.display();

    b2.display();

    b1.setBname("Java");

    b1.setId(102);

    b1.setAuthor("James Gosling");

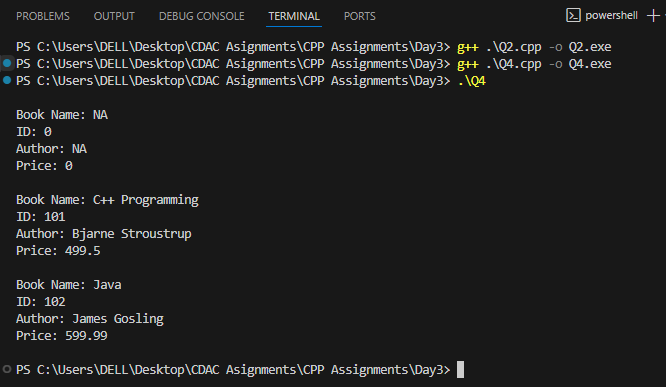
    b1.setPrice(599.99);

    b1.display();

    cout << endl;

    return 0;

}

****

1. **Create a class Point with data members as x,y. Create Default and Parameterized constructors. Write getters and setters for all the data members. Also add the display function. Create the object of this class in main method and invoke all the methods in that class.**

#include <iostream>

using namespace std;

class Point {

    int x, y;

public:

    Point() { x = y = 0; }

    Point(int a, int b) { x = a; y = b; }

    void setX(int a) { x = a; }

    void setY(int b) { y = b; }

    int getX() { return x; }

    int getY() { return y; }

    void display() {

        cout << "Point: (" << x << ", " << y << ")" << endl;

    }

};

int main() {

    Point p1;

    Point p2(10, 20);

    p1.display();

    p2.display();

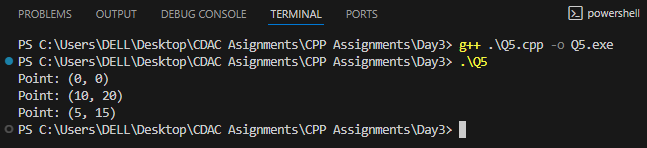
    p1.setX(5);

    p1.setY(15);

    p1.display();

    return 0;

}



1. **Create a class ComplexNumber with data members real, imaginary. Create Default and Parameterized constructors. Write getters and setters for all the data members. Also add the display function. Create the object of this class in main method and invoke all the methods in that class.**

#include <iostream>

using namespace std;

class ComplexNumber {

    int real, imag;

public:

    ComplexNumber() { real = imag = 0; }

    ComplexNumber(int r, int i) { real = r; imag = i; }

    void setReal(int r) { real = r; }

    void setImag(int i) { imag = i; }

    int getReal() { return real; }

    int getImag() { return imag; }

  void display() {

        cout << "Complex Number: " << real;

        if (imag >= 0) cout << "+" << imag << "i" << endl;

        else cout << imag << "i" << endl;

    }

};

int main() {

    ComplexNumber c1;

    ComplexNumber c2(5, -3);

    c1.display();

    c2.display();

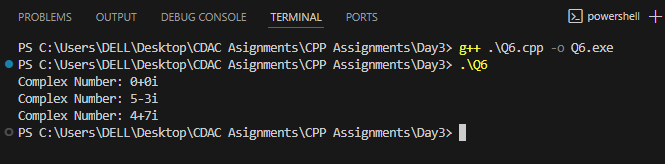
    c1.setReal(4);

    c1.setImag(7);

    c1.display();

    return 0;

}



**DAY 4**

1. **Create Date class with members day,month ,year. Write no argument and parameterised constructor .Create two object s and initialize them using no argument and parameterised constructor respectively.Print date using display function**.

#include <iostream>

using namespace std;

class Date {

    int day, month, year;

public:

    Date() { day = 1; month = 1; year = 2000; }

    Date(int d, int m, int y) { day = d; month = m; year = y; }

    void display() {

        cout << day << "/" << month << "/" << year << endl;

    }

};

int main() {

    Date d1;

    Date d2(23, 8, 2025);

    cout << "Date using No-arg Constructor: ";

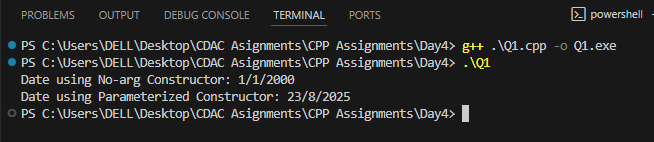
    d1.display();

    cout << "Date using Parameterized Constructor: ";

    d2.display();

    return 0;

}

****

1. **Create Employee class with members id(int),name(string),dob(Date).Use above created Date class. Write default and parameterised constructor in Employee Class.Write accept() function to accept information and display() to display emp information.**

#include <iostream>

using namespace std;

class Date

{

    int day, month, year;

public:

    Date()

    {

        day = 1;

        month = 1;

        year = 2000;

    }

    Date(int d, int m, int y)

    {

        day = d;

        month = m;

        year = y;

    }

    void accept()

    {

        cout << "Enter day month year: ";

        cin >> day >> month >> year;

    }

    void display()

    {

        cout << day << "/" << month << "/" << year;

    }

};

class Employee : public Date

{

    int id;

    string name;

    Date dob;

public:

    Employee()

    {

        id = 0;

        name = "NA";

    }

    Employee(int i, string n, Date d) : Date(d)

    {

        this->id = i;

        this->name = n;

    }

    void accept()

    {

        cout << "Enter Employee ID: ";

        cin >> id;

        cout << "Enter Employee Name: ";

        cin >> name;

        cout << "Enter Date of Birth -> ";

        dob.accept();

    }

    void display()

    {

        cout << "\nEmployee ID: " << id

             << "\nName: " << name

             << "\nDate of Birth: ";

        dob.display();

        cout << endl;

    }

};

int main()

{

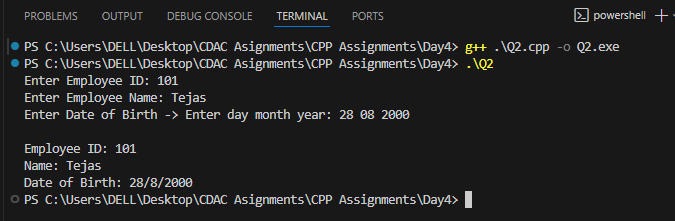
    Employee e1;

    e1.accept();

    e1.display();

    return 0;

}



1. **Consider that payroll software needs to be developed for computerization of**

**operations of an ABC organization. The organization has employees.**

**3.1. Construct a class Employee with following members using private access**

**specifies:**

**Employee Id integer**

**Employee Name string**

**Basic Salary double**

**HRA double**

**Medical double=1000**

**PF double**

**PT double**

**Net Salary double**

**Gross Salary double**

**Please use following expressions for calculations://Note:Don't accept HRA,PF PT from user**

**\* HRA = 50% of Basic Salary**

**\* PF = 12% of Basic Salary**

**\* PT = Rs. 200**

**3.2. Write methods to display the details of an employee and calculate the gross**

**and net salary.**

**\* Goss Salary = Basic Salary + HRA + Medical**

**\* Net Salary = Gross Salary – (PT + PF)**

**Create Object of employee class and assign values and display Details.**

#include <iostream>

using namespace std;

class Employee

{

    int empId;

    string empName;

    double basicSalary;

    double hra, medical, pf, pt;

    double grossSalary, netSalary;

public:

    Employee(int id, string name, double basic)

    {

        empId = id;

        empName = name;

        basicSalary = basic;

        medical = 1000;

        hra = 0.5 \* basicSalary;

        pf = 0.12 \* basicSalary;

        pt = 200;

        grossSalary = basicSalary + hra + medical;

        netSalary = grossSalary - (pf + pt);

    }

    void display()

    {

        cout << "\nEmployee ID = " << empId

             << "\nName = " << empName

             << "\n \t Gross Salary = " << grossSalary

             << "\n \t Net Salary = " << netSalary

             << endl;

    }

};

int main()

{

    int id;

    string name;

    double basic;

    cout << endl << "Enter Employee ID =  ";

    cin >> id;

    cout << "Enter Employee Name =  ";

    cin >> name;

    cout << "Enter Basic Salary =  ";

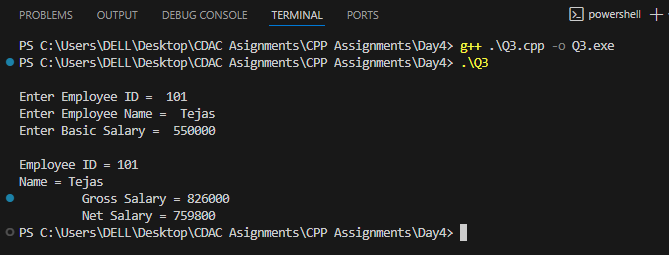
    cin >> basic;

    Employee e1(id, name, basic);

    e1.display();

    return 0;

}



4. 1 Solve this.

Fresh business scenario to apply inheritance , polymorphism to emp based organization scenario.

Create Emp based organization structure --- Emp , Mgr , Worker

1.1 Emp state--- id(int), name, deptId , basicSalary(double)

Accept all of above in constructor arguments.

Methods ---

1.2. compute net salary ---ret 0

(eg : public double computeNetSalary(){return 0;})

1.2 Mgr state ---id,name,basic,deptId , perfBonus

Add suitable constructor

Methods ----

1. compute net salary (formula: basic+perfBonus) -- override computeNetSalary

1.3 Worker state --id,name,basic,deptId,hoursWorked,hourlyRate

Methods :

1. compute net salary (formula: = basic+(hoursWorked\*hourlyRate) --override computeNetSalary

2. get hrlyRate of the worker -- add a new method to return hourly rate of a worker.(getter)

Create suitable array to store organization details.

Provide following options

1. Hire Manager I/P : all manager details

2. Hire Worker I/P : all worker details

3. Display information of all employees net salary (by invoking computeNetSal),

4. Exit

#include <iostream>

#include <string>

using namespace std;

class Emp {

protected:

    int empId;

    string empName;

    int deptId;

    double basicSalary;

public:

    Emp() : empId(0), empName(""), deptId(0), basicSalary(0) {}

    Emp(int id, string name, int dept, double basic)

        : empId(id), empName(name), deptId(dept), basicSalary(basic) {}

    virtual double computeNetSalary() { return 0; }

    virtual void display() {

        cout << "ID: " << empId

             << ", Name: " << empName

             << ", DeptID: " << deptId

             << ", Net Salary: " << computeNetSalary() << endl;

    }

};

class Manager : public Emp {

    double performanceBonus;

public:

    Manager() : Emp(), performanceBonus(0) {}

    Manager(int id, string name, int dept, double basic, double bonus)

        : Emp(id, name, dept, basic), performanceBonus(bonus) {}

    double computeNetSalary() override {

        return basicSalary + performanceBonus;

    }

};

class Worker : public Emp {

    int hoursWorked;

    double hourlyRate;

public:

    Worker() : Emp(), hoursWorked(0), hourlyRate(0) {}

    Worker(int id, string name, int dept, double basic, int hours, double rate)

        : Emp(id, name, dept, basic), hoursWorked(hours), hourlyRate(rate) {}

    double computeNetSalary() override {

        return basicSalary + (hoursWorked \* hourlyRate);

    }

};

int main() {

    Manager managers[50];

    Worker workers[50];

    int managerCount = 0, workerCount = 0;

    int menuChoice;

    do {

        cout << "\n1. Hire Manager\n2. Hire Worker\n3. Display All Employees\n4. Exit\nEnter choice: ";

        cin >> menuChoice;

        switch(menuChoice) {

            case 1: {

                if(managerCount >= 50) { cout << "Manager limit reached!\n"; break; }

                int id, dept;

                double basic, bonus;

                string name;

                cout << "Enter Manager ID: ";

                cin >> id;

                cin.ignore();

                cout << "Enter Manager Name: ";

                getline(cin, name);

                cout << "Enter DeptID, Basic Salary, Performance Bonus: ";

                cin >> dept >> basic >> bonus;

                managers[managerCount++] = Manager(id, name, dept, basic, bonus);

                break;

            }

            case 2: {

                if(workerCount >= 50) { cout << "Worker limit reached!\n"; break; }

                int id, dept, hours;

                double basic, rate;

                string name;

                cout << "Enter Worker ID: ";

                cin >> id;

                cin.ignore();

                cout << "Enter Worker Name: ";

                getline(cin, name);

                cout << "Enter DeptID, Basic Salary, Hours Worked, Hourly Rate: ";

                cin >> dept >> basic >> hours >> rate;

                workers[workerCount++] = Worker(id, name, dept, basic, hours, rate);

                break;

            }

            case 3: {

                if(managerCount == 0)

                    cout << "\nNo Managers to display.\n";

                else {

                    cout << "\nManagers:\n";

                    for (int i = 0; i < managerCount; i++)

                        managers[i].display();

                }

                if(workerCount == 0)

                    cout << "\nNo Workers to display.\n";

                else {

                    cout << "\nWorkers:\n";

                    for (int i = 0; i < workerCount; i++)

                        workers[i].display();

                }

                break;

            }

            case 4:

                cout << "Exiting program...\n";

                break;

            default:

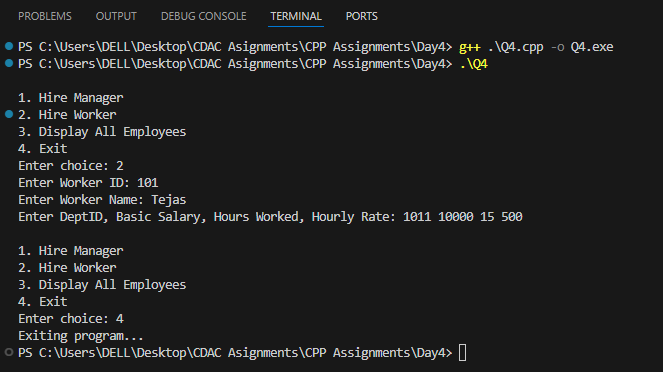
                cout << "Invalid choice! Please try again.\n";

        }

    } while(menuChoice != 4);

    return 0;

}



5. Create cpp application for bank account handling.

**5.1.** Create a class BankAccount -- acct no(int),customer name(string),balance(double) Add constr. (2 constrs : first to accept all details )

**5.2** Add Business logic methods Methods public void withdraw(double amt) and public void deposit(double amt)

**5.3**: Create object of account class and test withdraw and deposit methods.

#include <iostream>

#include <string>

using namespace std;

class BankAccount {

private:

    int acctNo;

    string custName;

    double balance;

public:

    BankAccount(int acc, string name, double bal) {

        acctNo = acc;

        custName = name;

        balance = bal;

    }

    BankAccount() {

        acctNo = 0;

        custName = "";

        balance = 0.0;

    }

    void deposit(double amt) {

        if (amt > 0) {

            balance += amt;

            cout << "Deposited: " << amt << endl;

        } else {

            cout << "Invalid deposit amount!" << endl;

        }

    }

    void withdraw(double amt) {

        if (amt > 0 && amt <= balance) {

            balance -= amt;

            cout << "Withdrawn: " << amt << endl;

        } else if (amt > balance) {

            cout << "Insufficient balance!" << endl;

        } else {

            cout << "Invalid withdraw amount!" << endl;

        }

    }

    void display() {

        cout << "Account No: " << acctNo << endl;

        cout << "Customer Name: " << custName << endl;

        cout << "Balance: " << balance << endl;

    }

};

int main() {

    BankAccount account1(101, "Tejas", 5000.0);

    cout << "Initial Account Details:" << endl;

    account1.display();

    cout << "--------------------------" << endl;

    // Test deposit

    account1.deposit(2000.0);

    account1.display();

    cout << "--------------------------" << endl;

    // Test withdraw

    account1.withdraw(1500.0);

    account1.display();

    cout << "--------------------------" << endl;

    // Test withdraw exceeding balance

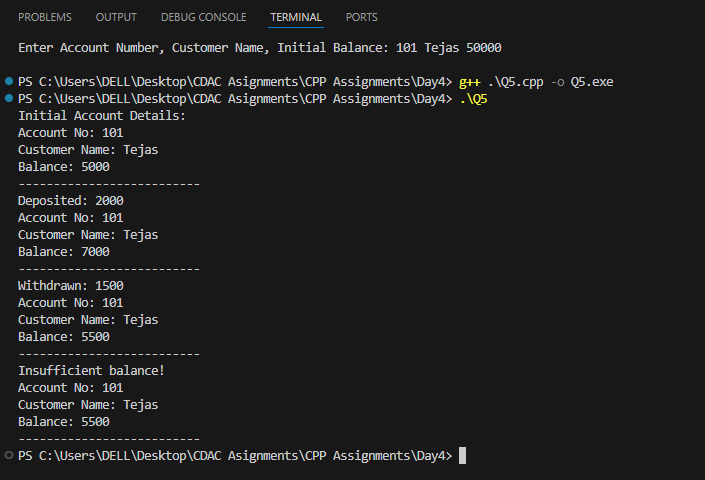
    account1.withdraw(7000.0);

    account1.display();

    cout << "--------------------------" << endl;

    return 0;

}



**DAY 5**

#include<bits/stdc++.h>

using namespace std;

class Bank{

private:

long long accountNo;

string CustomerName;

double balance;

public:

Bank(string CustomerName){

srand(time(0));

accountNo = 1000000000LL + rand() % 9000000000LL;

this->CustomerName= CustomerName;

balance =0;

}

void Withdraw(double amount){

if(balance < amount){

cout<<"Insufficient Balance"<<endl;

return;

}

balance-=amount;

cout<<"Please wait till money, then remove your card"<<endl;

}

void deposit(double amount){

balance+= amount;

}

double displayBalance(){

return balance;

}

int getAccountNum(){

return accountNo;

}

};

int main(){

int choice;

Bank c1("chandu");

do{

cout<<"1.Deposit\n2.Withdrawl\n3.Check balance0.exit"<<endl;

cout<<"enter choice"<<endl;

cin>>choice;

switch (choice)

{

case 1:

int amountD;

cout<<"enter amount to deposit"<<endl;

cin>>amountD;

c1.deposit(amountD);

break;

case 2:

int amountW;

cout<<"enter amount to withdraw"<<endl;

cin>> amountW;

c1.Withdraw(amountW);

break;

case 3:

cout<<"Your account number is: "<<c1.getAccountNum()<<endl;

cout<<"Your account balance is: "<<c1.displayBalance()<<endl;

break;

case 0:exit(0);

break;

default:

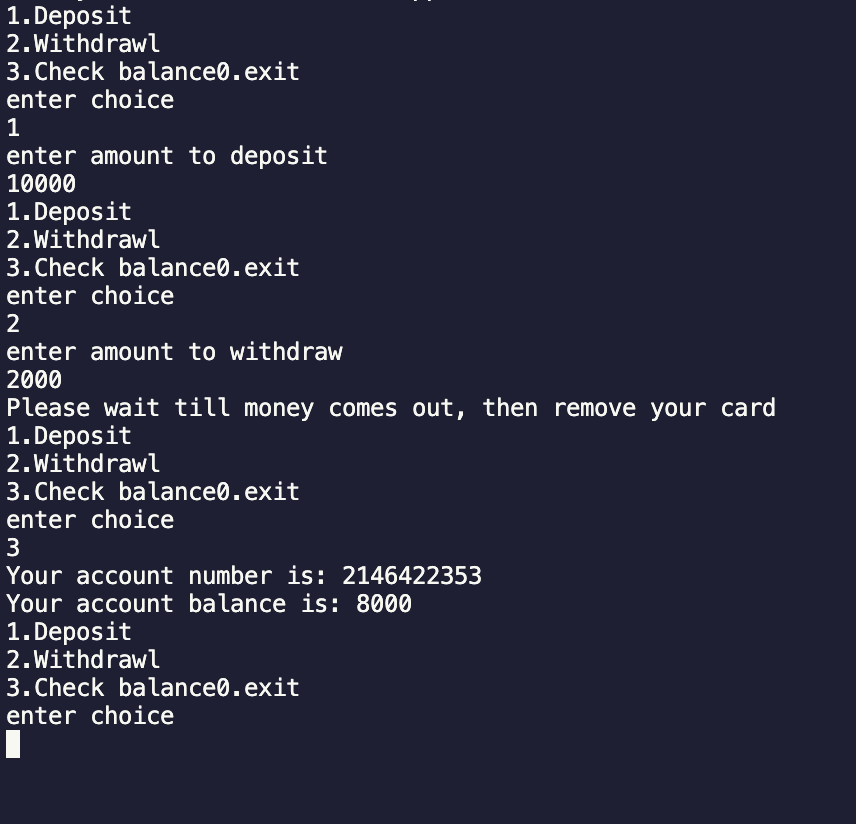
cout<<"Invalid choice"<<endl;

break;

}}while(choice!=0);

return 0;

}



#include<bits/stdc++.h>

using namespace std;

class Employee{

private:

int empId,deptId;

string name;

protected: double basicSalary;

public:

Employee(){

empId = 101;

deptId = 1;

name = "Default";

basicSalary = 50000;

}

Employee(int empId,int deptId,string name,double basicSalary){

this->name = name;

this->basicSalary = basicSalary;

this->deptId = deptId;

this->empId = empId;

}

double computeNetSal(){

return 0;

}

void display(){

cout<<"ID "<<empId<<endl;

cout<<"DeptId "<<deptId<<endl;

cout<<"name "<<name<<endl;

cout<<"basicSalary "<<basicSalary<<endl;

}

};

class Manager:public Employee{

private:

int perfBonus;

public:

Manager(){

perfBonus = 200;

}

Manager(int empId,int deptId,string name,double basicSalary,int perfBonus):Employee(empId,deptId,name,basicSalary){

this->perfBonus = perfBonus;

}

double computeNetSal(){

return basicSalary + perfBonus;

}

void display(){

Employee::display();

cout<<"perfBonus: "<<perfBonus<<endl;

}

};

class Worker:public Employee{

private:

double hourlyRate;

int hoursWorked;

public:

Worker(){

hourlyRate = 50.00;

hoursWorked = 8;

}

Worker(int empId,int deptId,string name,double basicSalary,int hoursWorked,double hourlyRate):Employee(empId,deptId,name,basicSalary){

this->hourlyRate = hourlyRate;

this->hoursWorked = hoursWorked;

}

double computeNetSal(){

return basicSalary + (hourlyRate\*hoursWorked);

}

double getHourlyRate(){

return hourlyRate;

}

void display(){

Employee::display();

cout<<"hoursWorked: "<<hoursWorked<<endl;

cout<<"hourlyRate: "<<hourlyRate<<endl;

}

};

int main(){

int choice;

vector<Manager> m;

vector<Worker> w;

do{

cout<<"1.Hire manager\n2.Hire worker\n3.display salary\n0.Exit"<<endl;

cout<<"Enter choice"<<endl;

cin>>choice;

switch (choice)

{

case 0:exit(0);

case 1:

int n;

cout<<"enter number of managers to enter"<<endl;

cin >>n;

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

int tempId,tempDeptId,tempPerfBonus;

double tempBasicSal;

string tempName;

cout<<"enter details"<<endl;

cin>>tempId>>tempDeptId>>tempName>>tempBasicSal>>tempPerfBonus;

Manager m1(tempId,tempDeptId,tempName,tempBasicSal,tempPerfBonus);

m.push\_back(m1);

}

break;

case 2:

int num;

cout<<"enter number of worker to enter"<<endl;

cin >>num;

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

int tempId,tempDeptId,tempBasicSal,tempHoursWorked;

double tempHourlyRate;;

string tempName;

cout<<"enter details"<<endl;

cin>>tempId>>tempDeptId>>tempName>>tempHoursWorked>>tempHourlyRate;

Worker w1(tempId,tempDeptId,tempName,tempBasicSal,tempHoursWorked,tempHourlyRate);

w.push\_back(w1);

}

case 3:

cout<<"-------------------MANAGERS--------------------"<<endl;

for(int i=0;i<m.size();i++){

m[i].display();

}

cout<<"-------------------WORKERS--------------------"<<endl;

for(int i=0;i<m.size();i++){

w[i].display();

}

break;

default:

cout<<"wrong choice, enter again"<<endl;

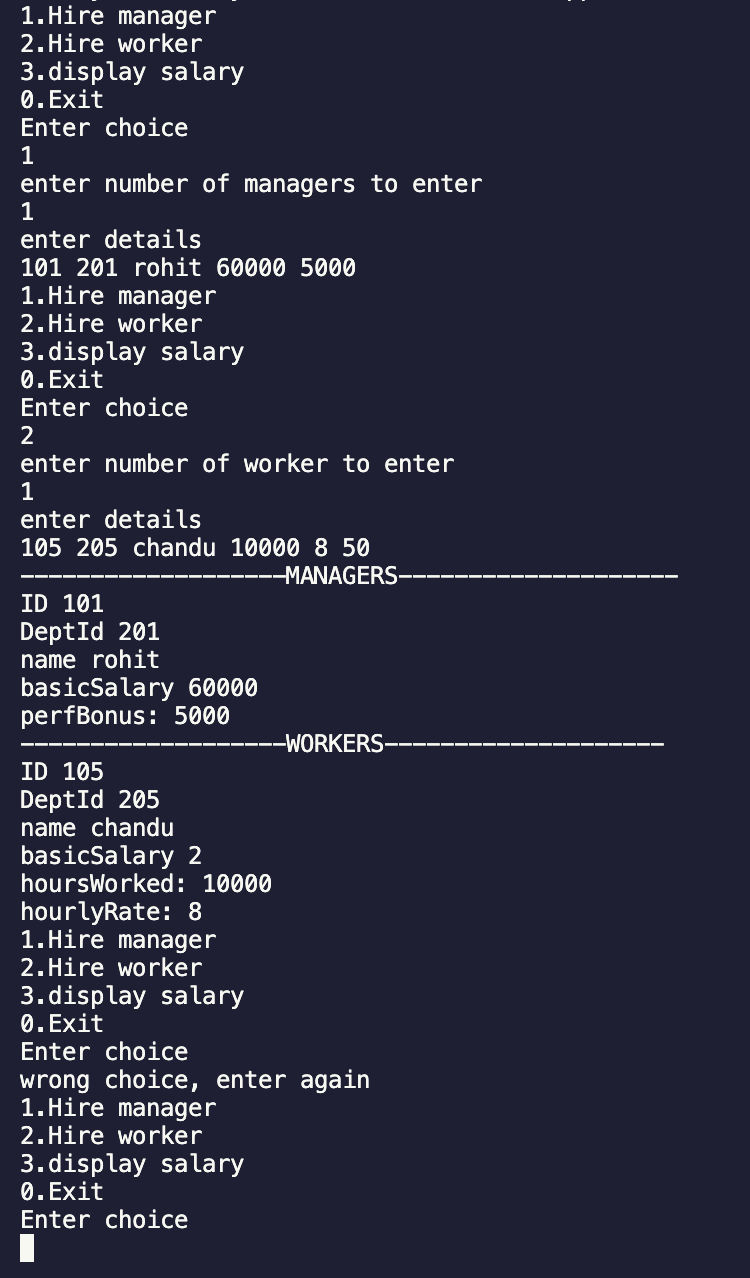
break;

}

}while(choice!=0);

return 0;

}



**DAY 6**

1. Basic Template Assignments. Function Template – Swap. Write a function template swapValues() that swaps two variables of any type. Test with int, double, and string.

#include <iostream>

using namespace std;

template <typename T>

void swapValues(T &a, T &b) {

    T temp = a;

    a = b;

    b = temp;

}

int main() {

    int x=10, y=20;

    swapValues(x,y);

    cout<<"Swapped Int: "<<x<<" "<<y<<endl;

    double p=1.1, q=2.2;

    swapValues(p,q);

    cout<<"Swapped Double: "<<p<<" "<<q<<endl;

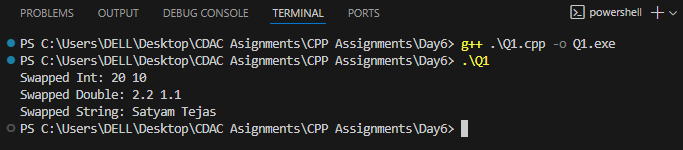
    string s1="Tejas", s2="Satyam";

    swapValues(s1,s2);

    cout<<"Swapped String: "<<s1<<" "<<s2<<endl;

    return 0;

}



2. Function Template – Maximum. Write a function template findMax() that returns the maximum of two values. Test with int, float, and char.

#include <iostream>

using namespace std;

template <typename T>

T findMax(T a, T b) {

    return (a > b) ? a : b;

}

int main() {

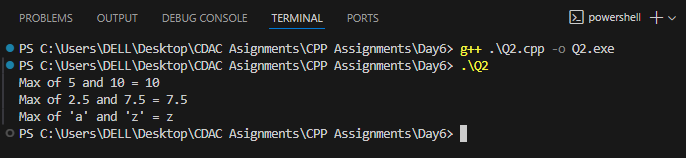
    cout<<"Max of 5 and 10 = "<<findMax(5,10)<<endl;

    cout<<"Max of 2.5 and 7.5 = "<<findMax(2.5f,7.5f)<<endl;

    cout<<"Max of 'a' and 'z' = "<<findMax('a','z')<<endl;

    return 0;

}



3. Class Template – Box. Implement a class template Box<T> that stores one value of any type and provides getValue() and setValue() methods.

#include <iostream>

using namespace std;

template <typename T>

class Box {

    T value;

public:

    void setValue(T v) { value = v; }

    T getValue() { return value; }

};

int main() {

    Box<int> intBox;

    intBox.setValue(100);

    cout<<"Int Box: "<<intBox.getValue()<<endl;

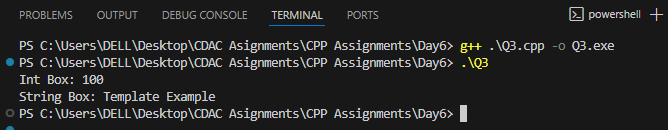
    Box<string> strBox;

    strBox.setValue("Template Example");

    cout<<"String Box: "<<strBox.getValue()<<endl;

    return 0;

}



4. Class Template – Calculator. Create a class template Calculator<T> with functions:

add(), subtract(), multiply(), divide(). Test with int and double.

#include <iostream>

using namespace std;

template <typename T>

class Calculator {

public:

    T add(T a, T b) { return a+b; }

    T subtract(T a, T b) { return a-b; }

    T multiply(T a, T b) { return a\*b; }

    T divide(T a, T b) { return (b!=0)?a/b:0; }

};

int main() {

    Calculator<int> intCalc;

    cout<<"Int Add: "<<intCalc.add(5,3)<<endl;

    Calculator<double> doubleCalc1;

    cout<<"Double Divide: "<<doubleCalc1.divide(7.5,2.5)<<endl;

     Calculator<double> doubleCalc2;

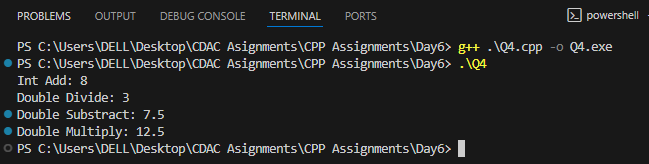
    cout<<"Double Substract: "<<doubleCalc2.subtract(10,2.5)<<”\n”;

    Calculator<double> doubleCalc3;

    cout<<"Double Multiply: "<<doubleCalc3.multiply(5,2.5)<<endl;

    return 0;

}



5. Function Template – Array Sum. Write a function template sumArray() that accepts an array of any type and returns the sum of its elements.

#include <iostream>

using namespace std;

template <typename T>

T sumArray(T arr[], int size)

{

    T sum = 0;

    for (int i = 0; i < size; i++)

    {

        sum += arr[i];

    }

    return sum;

}

int main()

{

    int arr1[] = {1, 2, 3, 4, 5};

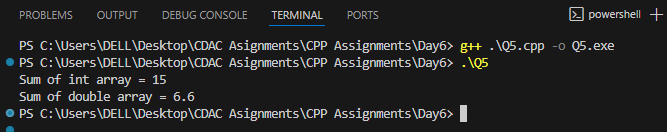
    cout << "Sum of int array = " << sumArray(arr1, 5) << endl;

    double arr2[] = {1.1, 2.2, 3.3};

    cout << "Sum of double array = " << sumArray(arr2, 3) << endl;

    return 0;

}



6. Class Template – Stack. Implement a class template Stack<T> with functions:

push(), pop(), peek(), isEmpty(). Test with int and string.

#include <iostream>

#include <vector>

using namespace std;

template <typename T>

class Stack

{

    vector<T> data;

public:

    void push(T val) { data.push\_back(val); }

    void pop()

    {

        if (!data.empty())

            data.pop\_back();

    }

    T peek() { return data.back(); }

    bool isEmpty() { return data.empty(); }

};

int main()

{

    Stack<int> intStack;

    intStack.push(10);

    intStack.push(20);

    cout << "Top of int stack: " << intStack.peek() << endl;

    intStack.pop();

    cout << "Top after pop: " << intStack.peek() << endl;

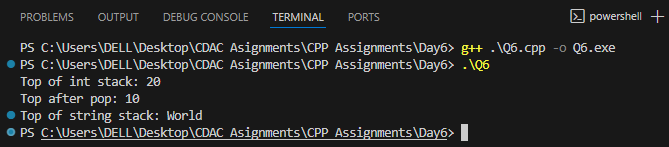
    Stack<string> strStack;

    strStack.push("Hello");

    strStack.push("World");

    cout << "Top of string stack: " << strStack.peek() << endl;

    return 0; }



7. Class Template – Queue. Implement a class template Queue<T> with functions:

enqueue(), dequeue(), front(), isEmpty().

#include <iostream>

#include <queue>

using namespace std;

template <typename T>

class Queue

{

    queue<T> q;

public:

    void enqueue(T val) { q.push(val); }

    void dequeue()

    {

        if (!q.empty())

            q.pop();

    }

    T front() { return q.front(); }

    bool isEmpty() { return q.empty(); }

};

int main()

{

    Queue<int> intQueue;

    intQueue.enqueue(1);

    intQueue.enqueue(2);

    intQueue.enqueue(3);

    cout << "Front of int queue: " << intQueue.front() << endl;

    intQueue.dequeue();

    cout << "Front after dequeue: " << intQueue.front() << endl;

    Queue<string> strQueue;

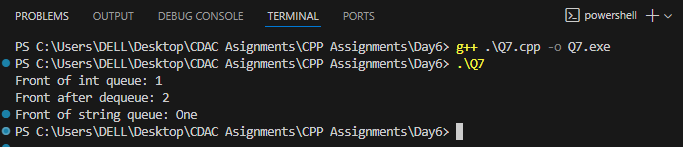
    strQueue.enqueue("One");

    strQueue.enqueue("Two");

    cout << "Front of string queue: " << strQueue.front() << endl;

    return 0;

}



**DAY 7**

1. Create an application for storing int values in vector.

Create menu drivin app for following menu;

1:add 2:show all 3:search 4:sort 5:reverse 6:clear

#include <iostream>

#include <stdlib.h>

#include <vector>

#include <algorithm>

using namespace std;

void add\_element(vector<int> &myVector)

{

    int num;

    cout << "Enter Number to Add = ";

    cin >> num;

    myVector.push\_back(num);

    cout << "Element Added!" << endl;

}

void show\_all(vector<int> &myVector)

{

    if (myVector.size() <= 0)

    {

        cout << "Vector is Empty!";

    }

    else

    {

        cout << "All Elements Are = ";

        for (int v : myVector)

        {

            cout << v << " ";

        }

    }

}

void search(vector<int> &myVector)

{

    if (myVector.size() <= 0)

    {

        cout << "Vector is Empty!";

        return;

    }

    int num, check = 0;

    cout << " Element to Search = ";

    cin >> num;

    for (int i = 0; i < myVector.size(); i++)

    {

        if (myVector[i] == num)

        {

            cout << "Element " << myVector[i] << " Found at Index - " << i << endl;

            check = 1;

        }

    }

    if (check != 1)

    {

        cout << "Element is NOT Found!" << endl;

    }

}

void sort\_elements(vector<int> &myVector)

{

    sort(myVector.begin(), myVector.end());

    cout << "Elements Are Sorted!";

}

void sort\_elements\_reverese(vector<int> &myVector)

{

    reverse(myVector.begin(), myVector.end());

    cout << "Elements Are Reversed!";

}

void clear\_vector(vector<int> &myVector)

{

    myVector.clear();

    cout << "Elements Are Cleared!";

}

int main()

{

    vector<int> myVector;

    int choice, num;

    do

    {

        cout << "0:Exit, 1:Add, 2:Show All, 3:Search, 4:Sort, 5:Reverse, 6:Clear/Empty" << endl;

        cout << "Enter Choice = ";

        cin >> choice;

        switch (choice)

        {

        case 0:

            exit(0);

        case 1:

            add\_element(myVector);

            break;

        case 2:

            show\_all(myVector);

            break;

        case 3:

            search(myVector);

        case 4:

            sort\_elements(myVector);

            break;

        case 5:

            sort\_elements\_reverese(myVector);

            break;

        case 6:

            clear\_vector(myVector);

            break;

        default:

            cout << "Invalid Choice!" << endl;

            break;

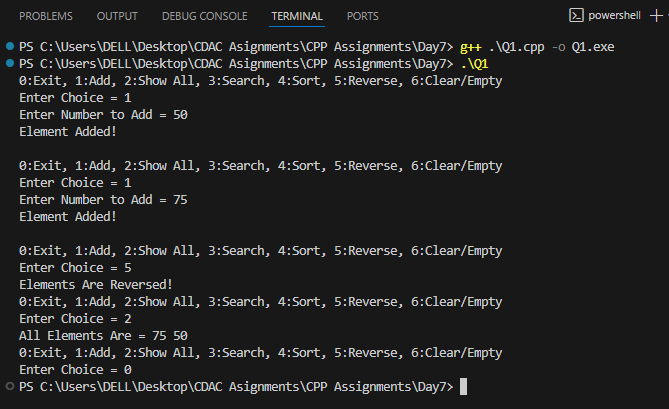
        }

        cout << endl;

    } while (choice != 0);

    return 0;

}



1. Create an application for storing user information in vector.

(Hint:User class with data member userid,name,email,pwd)

Create Menu Driven app

1:add user

2:display all users

3:search user

4:change pwd

5:delete all

#include <iostream>

#include <stdlib.h>

#include <vector>

#include <algorithm>

using namespace std;

class User

{

private:

    int userid;

    string name, email, pwd;

public:

    User() {}

    User(int uid, string uname, string uemail, string upwd) : userid(uid), name(uname), email(uemail), pwd(upwd) {}

    void display()

    {

        cout << "User Id = " << userid << ", User Name = " << name << ", User Email = " << email << " & User Password = " << pwd << endl;

    }

    int getId()

    {

        return userid;

    }

    string getName()

    {

        return name;

    }

    void setPwd(string pwd)

    {

        this->pwd = pwd;

    }

};

int main()

{

    vector<User> myVector;

    int choice;

    int id;

    string name, email, pwd;

    User u1;

    do

    {

        cout << "0:Exit, 1:Add, 2:Show All, 3:Search, 4.Change Pwd, 5:Delete" << endl;

        cout << "Enter Choice = ";

        cin >> choice;

        switch (choice)

        {

        case 0:

            exit(0);

        case 1:

            cout << "Enter User Id, Name, Email, Pwd = ";

            cin >> id >> name >> email >> pwd;

            u1 = User(id, name, email, pwd);

            myVector.push\_back(u1);

            break;

        case 2:

            if (myVector.empty())

            {

                cout << "No Users to Display!" << endl;

            }

            else

            {

                cout << "All Users are: " << endl;

                for (User u2 : myVector)

                {

                    u2.display();

                }

                cout << endl;

                break;

            }

        case 3:

        {

            int flag = 0;

            cout << "Enter User ID To Search = ";

            cin >> id;

            for (int i = 0; i < myVector.size(); i++)

            {

                if (id == myVector[i].getId())

                {

                    myVector[i].display();

                    flag = 1;

                }

            }

            if (flag != 1)

            {

                cout << "User Not Found!";

            }

            cout << endl;

            break;

        }

        case 4:

        {

            int flag = 0;

            cout << "Enter User ID & Password to change password = ";

            cin >> id >> pwd;

            for (int i = 0; i < myVector.size(); i++)

            {

                if (id == myVector[i].getId())

                {

                    myVector[i].setPwd(pwd);

                    cout << "Password Is Updated For User " + myVector[i].getName();

                    flag = 1;

                }

            }

            if (flag != 1)

            {

                cout << "User Not Found, Password NOT Changed!";

            }

            cout << endl;

            break;

        }

        case 5:

        {

            myVector.clear();

            cout << "All Users Are Removed!" << endl;

            break;

        }

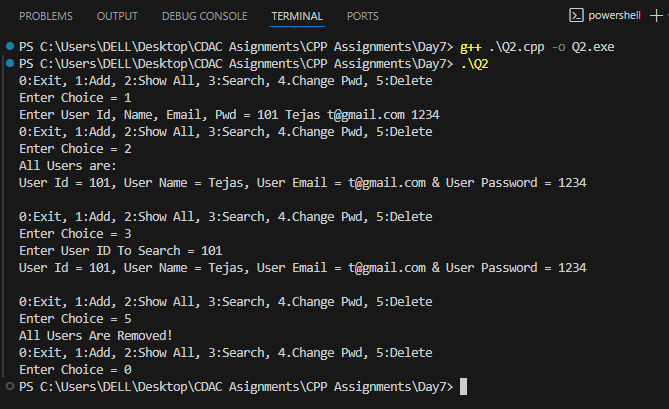
        default:

            break;

        }

    } while (choice != 0);

}



1. Create an application using set . Accept name of city from user & store in set.

Create Menu drivien app

1:add city

2:display all city

3: serach city

#include <iostream>

#include <stdlib.h>

#include <set>

using namespace std;

int main()

{

    set<string> mySet;

    string city;

    int choice;

    do

    {

        cout << "Enter choice, 1.Add-City 2.Display-City 3.Search-City = ";

        cin >> choice;

        switch (choice)

        {

        case 0:

            exit(0);

        case 1:

            cout << "Enter City Name = ";

            cin >> city;

            mySet.insert(city);

            cout << "City is Added!" << endl;

            break;

        case 2:

            cout << "Cities = ";

            for (string c : mySet)

            {

                cout << c << ", ";

            }

            cout << endl;

            break;

        case 3:

            cout << "Search City = ";

            cin >> city;

            for (string c : mySet)

            {

                if (c == city)

                    cout << "City " << c << " Found in SET!";

            }

            cout << endl;

            break;

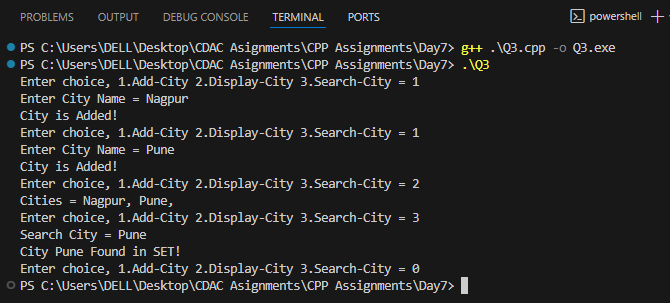
        default:

            break;

        }

    } while (choice != 0);

}



4. Create an application using map for storing key and value

key:int and value:Account type

Create Account class with actid ,name,balance

Create Menu driven app

1:Add Account

2:Display all

3:Search account by actid;

4:Remove all

#include <iostream>

#include <map>

#include <algorithm>

#include <stdlib.h>

using namespace std;

class Account

{

private:

    int accountId;

    string name;

    double balance;

public:

    Account() {}

    Account(int id, string nm, double b) : accountId(id), name(nm), balance(b) {}

    void display()

    {

        cout << "Account Id = " << accountId << ", Name = " << name << " & Balance = " << balance << endl;

    }

};

int main()

{

    int accountId, choice;

    string name;

    double balance;

    map<int, Account> myAcc;

    do

    {

        cout << "1.Add Account, 2.Display All, 3.Search Account by ID, 4.Remove All, 0.Exit" << endl;

        cout << "Enter Choice = ";

        cin >> choice;

        switch (choice)

        {

        case 0:

            exit(0);

        case 1:

        {

            cout << "Enter Account Id = ";

            cin >> accountId;

            cout << "Enter Name = ";

            cin >> name;

            cout << "Enter Balance = ";

            cin >> balance;

            Account acc(accountId, name, balance);

            myAcc.insert({accountId, acc});

            cout << "Account Added!"<<endl;

            break;

        }

        case 2:

        {

            cout << "All Accounts = "<<endl;

            for (auto ac : myAcc)

            {

                ac.second.display();

            }

            cout << "Account Added!"<<endl;

            break;

        }

        case 3:

        {

            cout << "Enter Account Id to Search = ";

            cin >> accountId;

            for (auto ac : myAcc)

            {

                if (ac.first == accountId)

                {

                    cout << "Account Found with Id = " << ac.first << " & Details = \n";

                    ac.second.display();

                    break;

                }

            }

            break;

        }

        case 4:

        {

            myAcc.clear();

            cout << "All Accounts Are Removed!" << endl;

            break;

        }

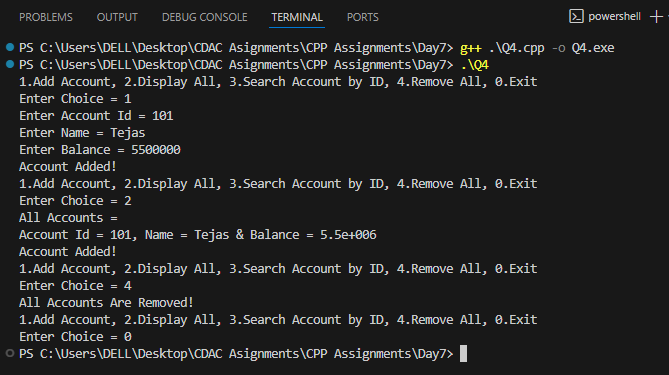
        default:

            break;

        }

    } while (choice != 0);

}



7. Create Student class in namespace CDAC namespace .

Create another Student class in IACSD namespace.

Try to access both student classes using namespace

#include<iostream>

using namespace std;

namespace IACSD{

    class Student{

        private:

            int id;

            string name, address;

        public:

            Student(){}

            Student(int i, string nm, string ad):id(i), name(nm), address(ad){}

            void display(){

                cout<<"Student ID = "<<id<<", Name = "<<name<<" & Address = "<<address<<endl;

            }

    };

}

namespace CDAC{

    class Student{

        private:

            int id;

            string name, address;

Public:

            Student(){}

            Student(int i, string nm, string ad):id(i), name(nm), address(ad){}

            void display(){

                cout<<"Student ID = "<<id<<", Name = "<<name<<" & Address = "<<address<<endl;

            }

    };

}

int main(){

    IACSD::Student s1(101, "Tejas", "Pune");

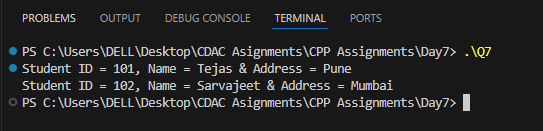
    s1.display();

    CDAC::Student s2(102, "Sarvajeet", "Mumbai");

    s2.display();

    return 0;

}



5. Create an File IO application for basic operation

1:Write file:accept data from user and store in file

2:Read file:display line by line

3:copy data from one file into another file

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void writefile(string &data) {

    string filename = "myfile.txt";

    ofstream outfile(filename, ios::app);

    cout << "Enter data you want to add: " << endl;

    getline(cin, data);

    outfile << data << endl;

    outfile.close();

}

void readfile(string f) {

    string filename = f;

    ifstream inputfile(filename);

    string line;

    if (inputfile.fail()) {

        cout << "File not found" << endl;

    } else {

        while (getline(inputfile, line)) {

            cout << line << endl;

        }

    }

    inputfile.close();

}

void copyData() {

    string filename = "myfile.txt";

    string filename1 = "myfile1.txt";

    ifstream inputfile(filename);

    string line;

    if (inputfile.fail()) {

        cout << "File not found" << endl;

    } else {

        while (getline(inputfile, line)) {

            ofstream outfile(filename1, ios::app);

            outfile << line << endl;

            outfile.close();

        }

    }

}

int main() {

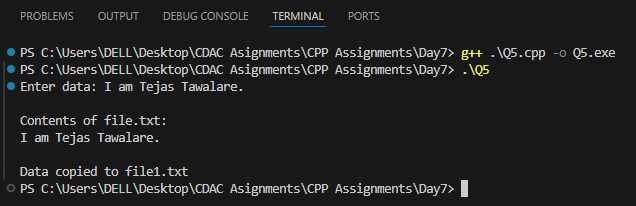
    string data;

    copyData();

    readfile("myfile1.txt");

    return 0;

}



6. Create CRUD Shop Application Using File

Write class Product with data member prdid,name,qty,price;

Menus:

1:Add Prd

2:Display Prds

3:Search Prd

4:Update/Modify prd

5:delete prd

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Product {

public:

int prdid, qty;

string name;

double price;

void input() {

cout << "Enter ID Name Qty Price: ";

cin >> prdid >> name >> qty >> price;

}

void display() {

cout << prdid << " " << name << " " << qty << " " << price << endl;

}

};

void addProduct() {

Product p; p.input();

ofstream out("shop.txt", ios::app);

out << p.prdid << " " << p.name << " " << p.qty << " " << p.price << endl;

out.close();

}

void displayProducts() {

ifstream in("shop.txt");

Product p;

while (in >> p.prdid >> p.name >> p.qty >> p.price) p.display();

in.close();

}

void searchProduct(int id) {

ifstream in("shop.txt");

Product p; bool found = false;

while (in >> p.prdid >> p.name >> p.qty >> p.price) {

if (p.prdid == id) { p.display(); found = true; break; }

}

if (!found) cout << "Product not found\n";

in.close();

}

void updateProduct(int id) {

ifstream in("shop.txt");

ofstream out("temp.txt");

Product p; bool found = false;

while (in >> p.prdid >> p.name >> p.qty >> p.price) {

if (p.prdid == id) {

cout << "Enter new details:\n";

p.input();

found = true;

}

out << p.prdid << " " << p.name << " " << p.qty << " " << p.price << endl;

}

in.close(); out.close();

remove("shop.txt"); rename("temp.txt", "shop.txt");

if (!found) cout << "Product not found\n";

}

void deleteProduct(int id) {

ifstream in("shop.txt");

ofstream out("temp.txt");

Product p; bool found = false;

while (in >> p.prdid >> p.name >> p.qty >> p.price) {

if (p.prdid == id) { found = true; continue; }

out << p.prdid << " " << p.name << " " << p.qty << " " << p.price << endl;

}

in.close(); out.close();

remove("shop.txt"); rename("temp.txt", "shop.txt");

if (!found) cout << "Product not found\n";

}

int main() {

int ch, id;

do {

cout << "\n1.Add 2.Display 3.Search 4.Update 5.Delete 0.Exit\nChoice: ";

cin >> ch;

switch (ch) {

case 1: addProduct(); break;

case 2: displayProducts(); break;

case 3: cout << "Enter ID: "; cin >> id; searchProduct(id); break;

case 4: cout << "Enter ID: "; cin >> id; updateProduct(id); break;

case 5: cout << "Enter ID: "; cin >> id; deleteProduct(id); break;

}

} while (ch != 0);

return 0;

}

