**Completed At**

**1) Into to programming - 02**

**2) Arrays - 73**

**Into to programming -**

1) Basic Into to programming and Fundamentals - Flow Charts & Pseudo Codes ,Conditional & Looping Statement, Break & Continue, Functions, Pointers

2) digit sum -

3) Reverse Number -

4) Prime no. -

5) Nested Loops & Patterns -

5.2) Star Pattern

5.3) Inverted Star Pattern

5.4) half Pyramid Pattern

5.5) Character Pyramid Pattern

5.6) Hollow Rectangle Pattern

5.7) Inverted & Rotated Half Pyramid

5.8) Floyd's Triangle

5.9) Diamond Pattern

5.10) Butterfly Pattern

6) Number System -

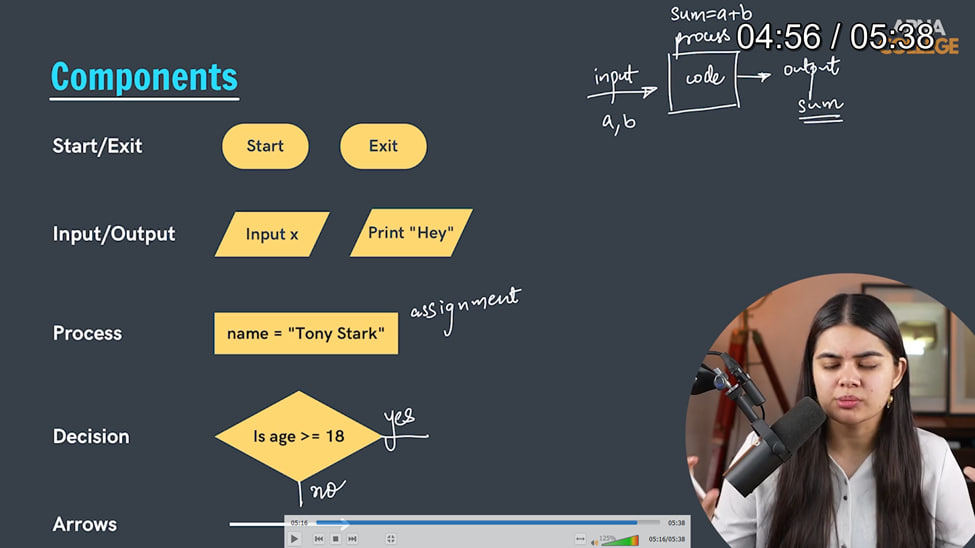
6.1) Binary to Decimal Conversion -

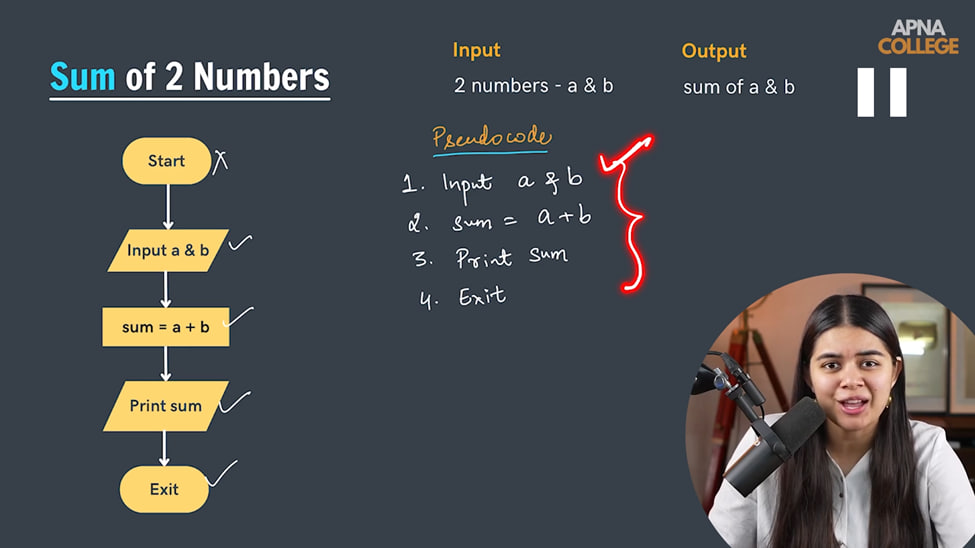
6.2) Decimal to Binary -

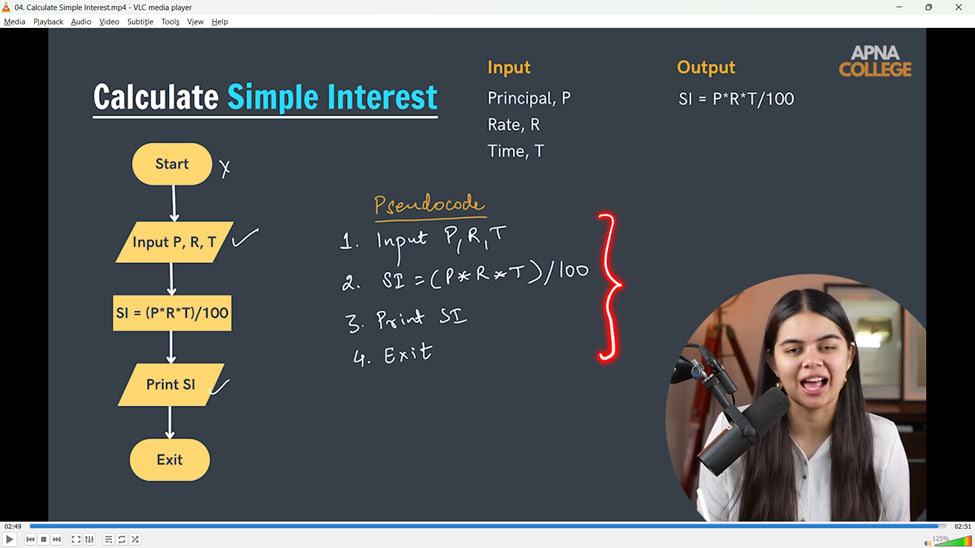
**1\_IntrotoProgramming –**

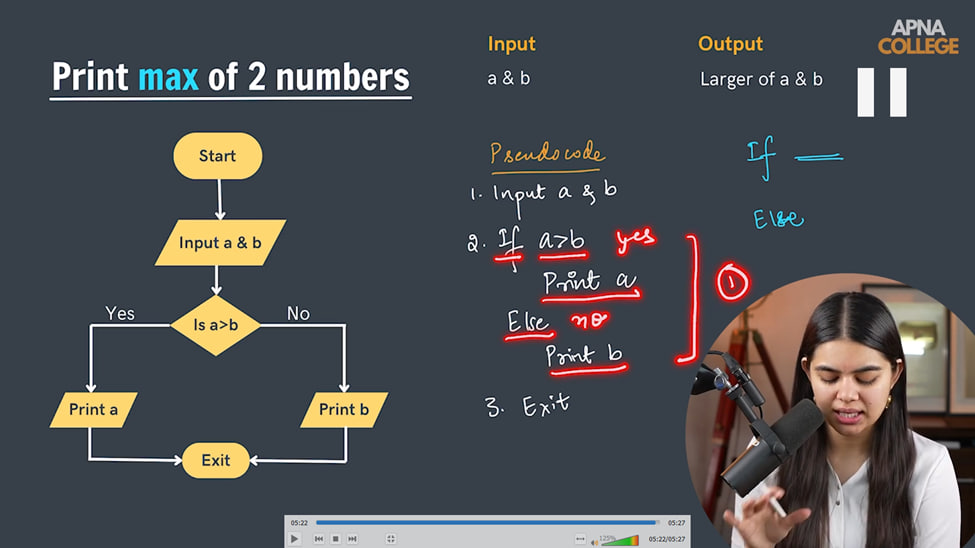
**1) Basic Into to programming and Fundamentals**

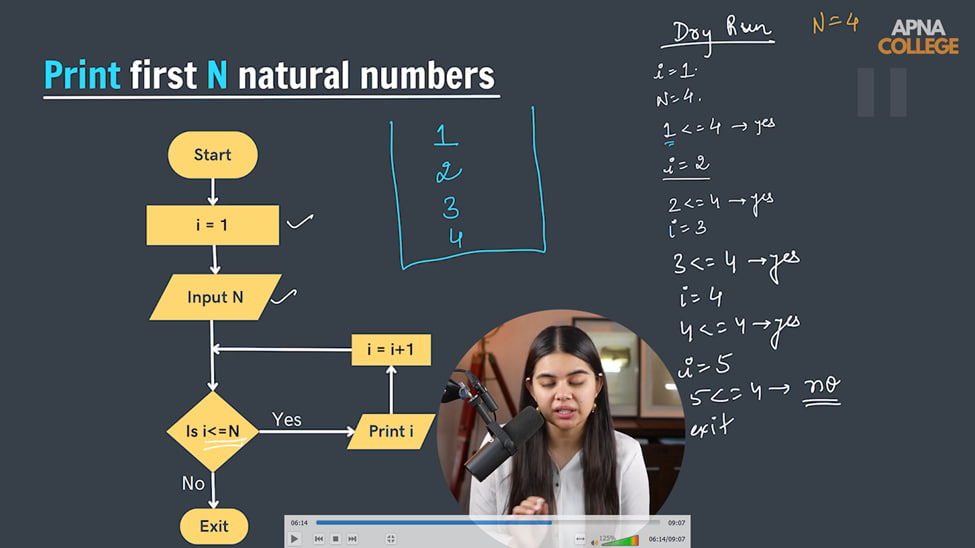
**Flow Charts & Pseudo Codes –**

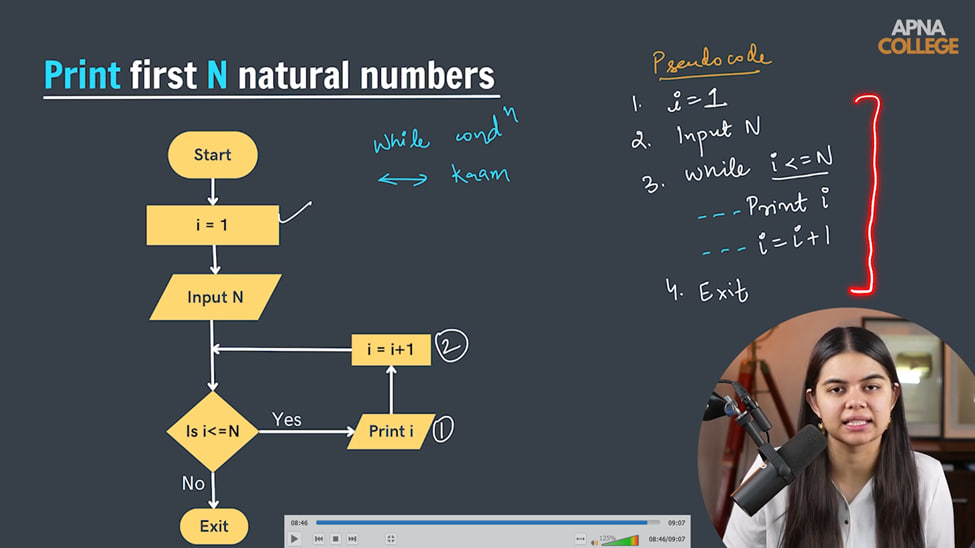
****

****

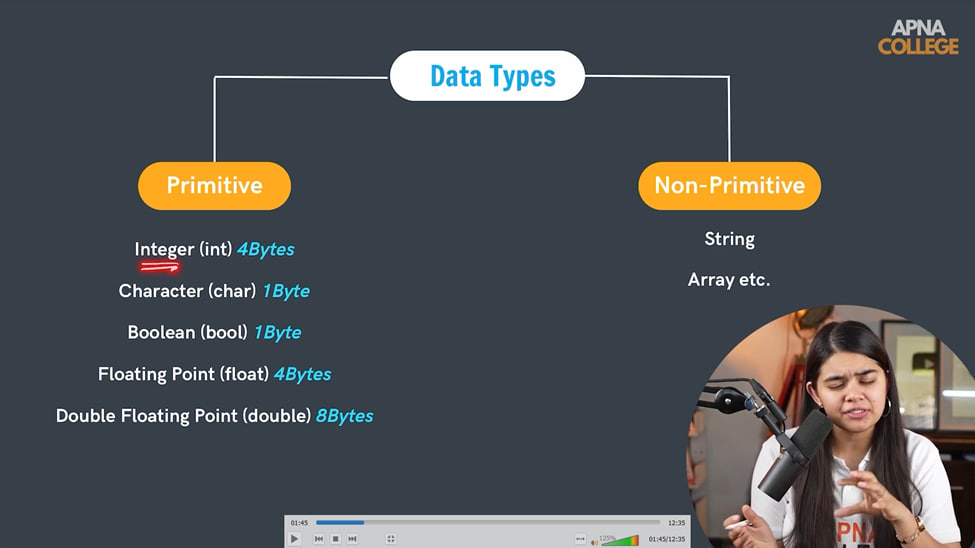
****

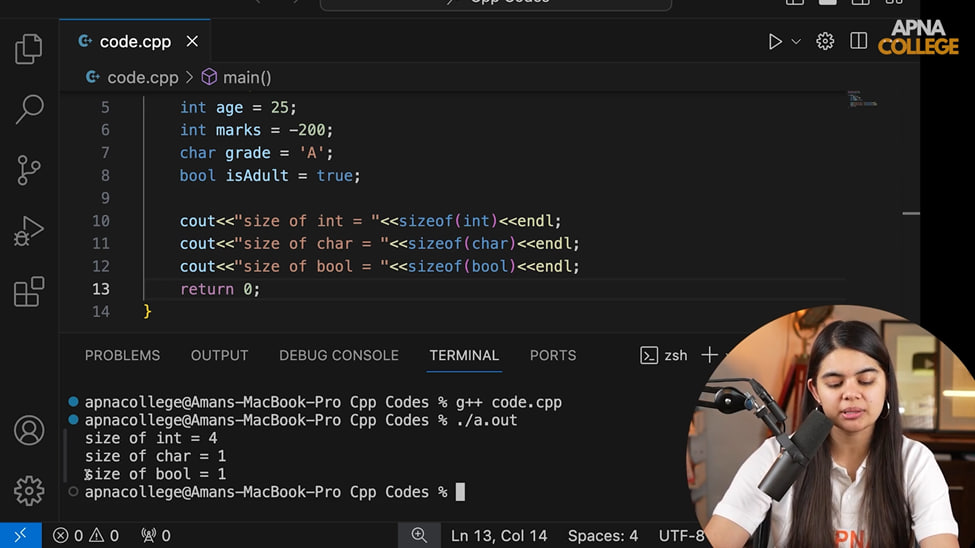
****

****

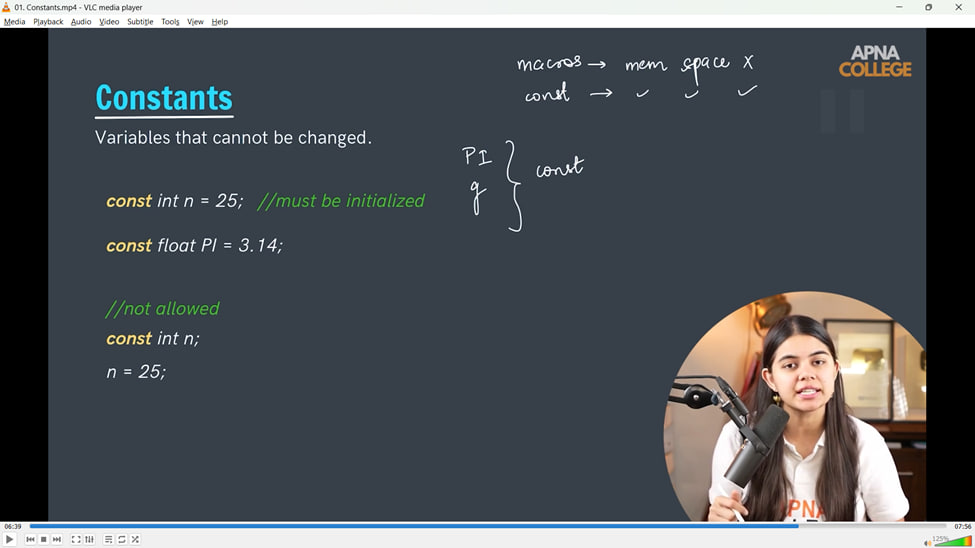
****

**Datatypes in cpp –**

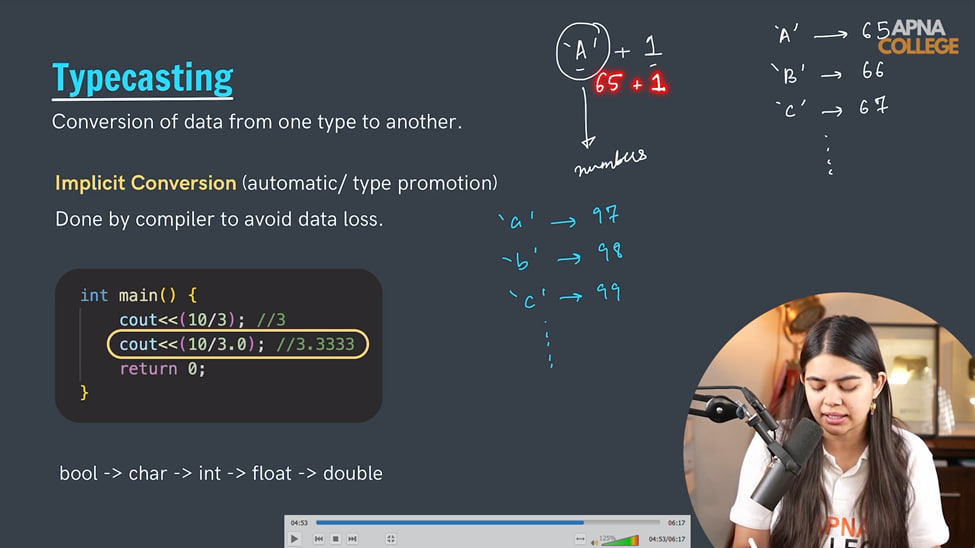
****

****

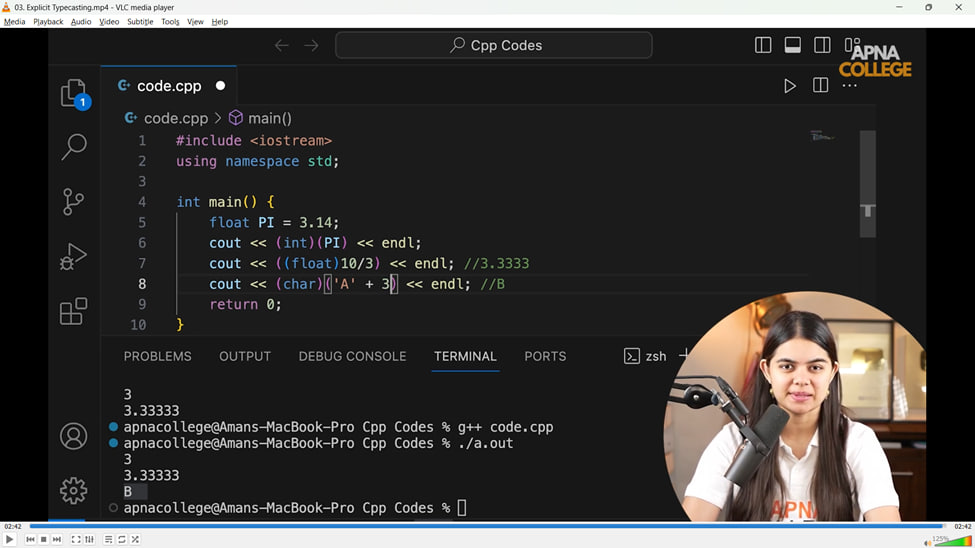
**Constants in cpp –**

****

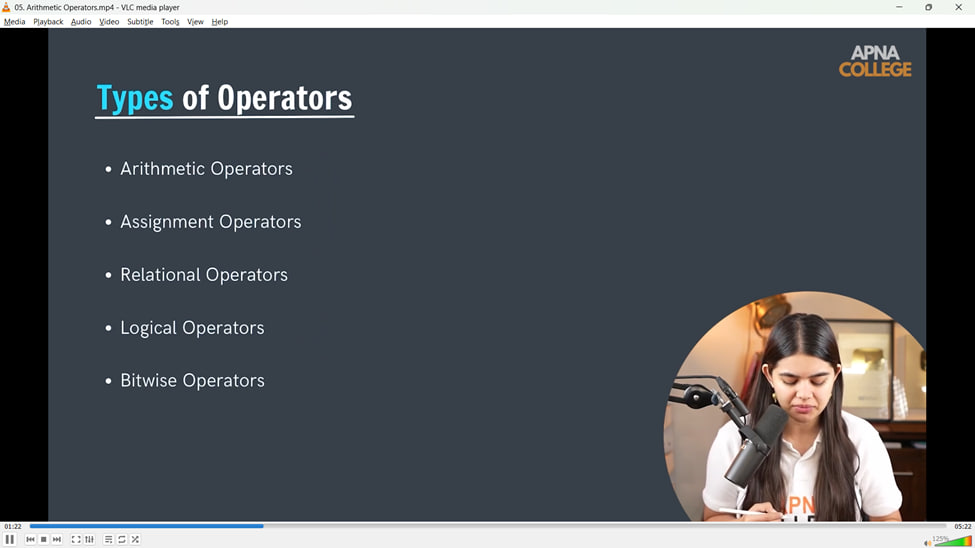
**Typecasting in cpp –**

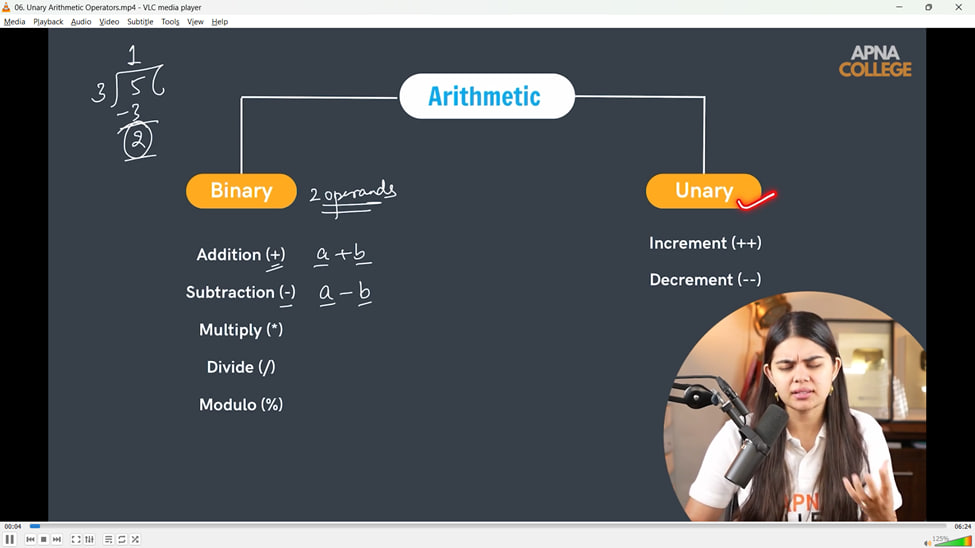
****

**Explicit Typecasting -**

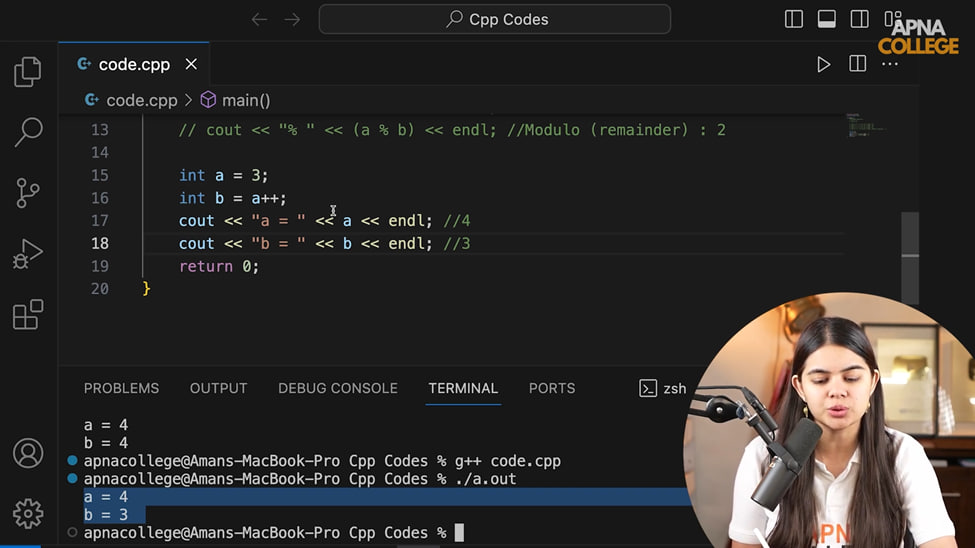
****

**Operatoes in cpp –**

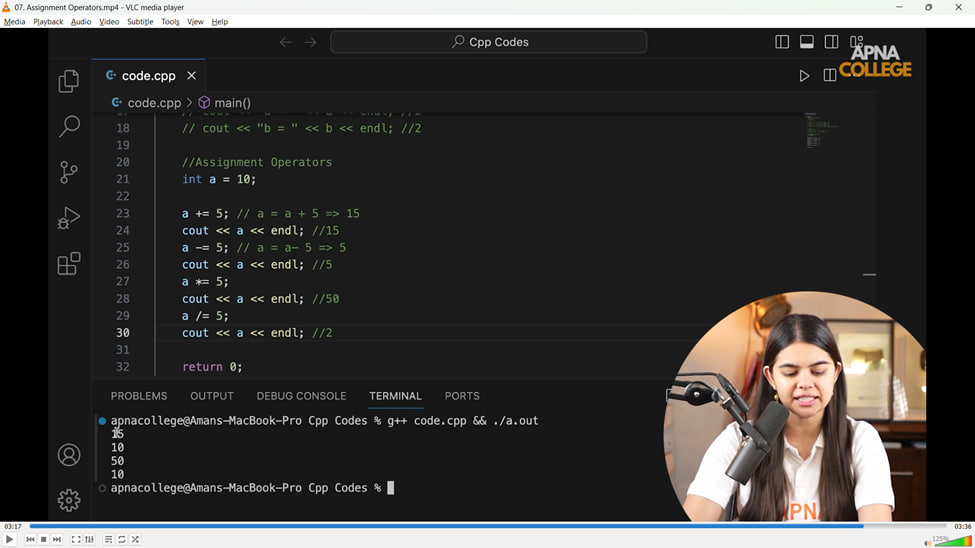
****

****

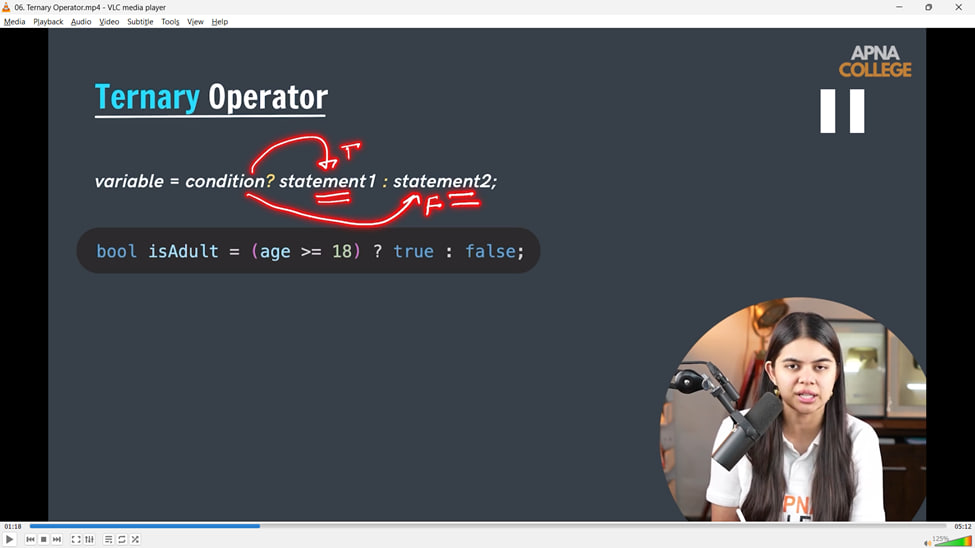
**Increment Decrement Operator -**

****

**Assignment Operator –**

****

**Ternary Operator –**

****

//**1) Basic Into to programming and Fundamentals**

int main()

{

    float PI = 3.14159265359;

    double PI2 = 3.14159265359;

    cout << "PI = " << PI << endl;

    cout << "PI 2 = " << PI2 << endl;

    // PI = 3.14159 - by defaalut the precision is 5. for increment it we can use setprecision function

    // PI 2 = 3.14159

    // setprecision function

    cout << setprecision(10) << "PI = " << PI << endl;

    cout << setprecision(10) << "PI2 = " << PI2 << endl;

    // PI = 3.141592741

    // PI2 = 3.141592654

    // Constants -

    const int n = 25; // must be initialized while defining

    const float py = 3.14;

    cout << "n is - " << n << endl;

    cout << "py is - " << py << endl;

    cout << "g - \n"

         << g; // g - 9.8

    // Typecasting order when different datatypes -

    // bool -> char -> int -> float -> double

    // Opertors -

    // ternary Op - simply single line code instead of if else

    bool isadult; // 1,0

    int age;

    cout << "What is the age " << endl;

    // cin >> age;

    isadult = age >= 18 ? true : false;

    cout << isadult << endl; // 1

    // largets of 2 mnumbers =-

    int a = 5;

    int b = 10;

    int largest = a >= b ? a : b;

    cout << largest << endl; // 10

    // Loops in cpp -

    // basoc square star pattern -

    for (int i = 1; i <= 4; i++)

    {

        cout << "\*\*\*\*" << endl;

    }

    /\*

    \*\*\*\*

    \*\*\*\*

    \*\*\*\*

    \*\*\*\*

    \*/

    for (int i = 5; i > 0; i--)

    {

        cout << i << " \n";

    }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**Reverse the number problem – **

**//2)  digit sum -**

    // int m;

    // cout << "Enter number value - " << endl;

    // cin >> m;

    // int sumdigit=0;

    // while (m > 0)

    // {

    //     int lastdigit = m % 10;

    //     sumdigit += lastdigit;

    //     m /= 10;

    // }

    // cout<<"So, the digit sum is - "<<sumdigit<<endl;

    // Odd digit sum -

    // int m;

    // cout << "Enter number value - " << endl;

    // cin >> m;

    // int sumdigit = 0;

    // while (m > 0)

    // {

    //     int lastdigit = m % 10;

    //     if (lastdigit % 2 != 0)

    //     {

    //         sumdigit += lastdigit;

    //     }

    //     m /= 10;

    // }

    // cout << "So, the digit sum is - " << sumdigit << endl;

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//3) Reverse Number -**

    // int m;

    // cout << "M's value - " << endl;

    // cin >> m;

    // int revesenumber;

    // while (m > 0)

    // {

    //     revesenumber = m % 10;

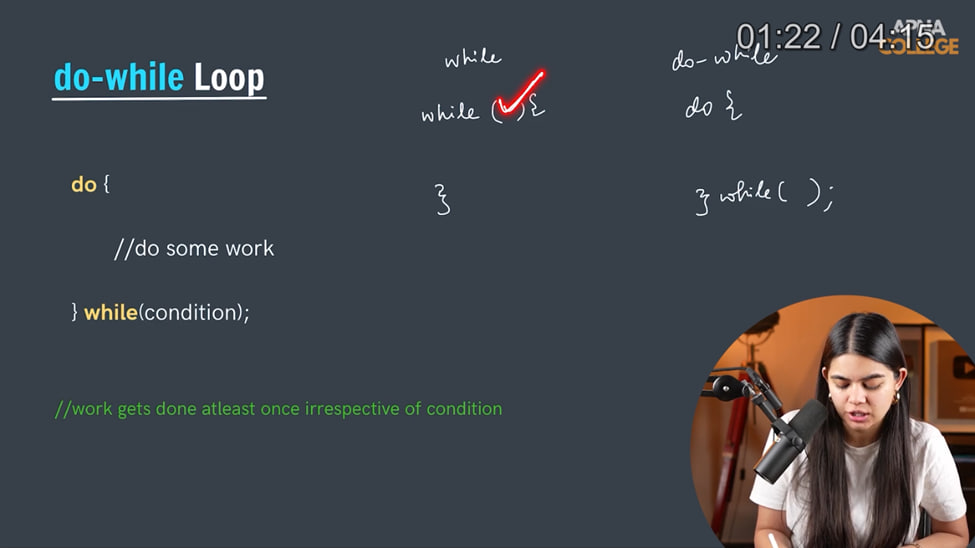
    //     cout<<revesenumber<<" ";

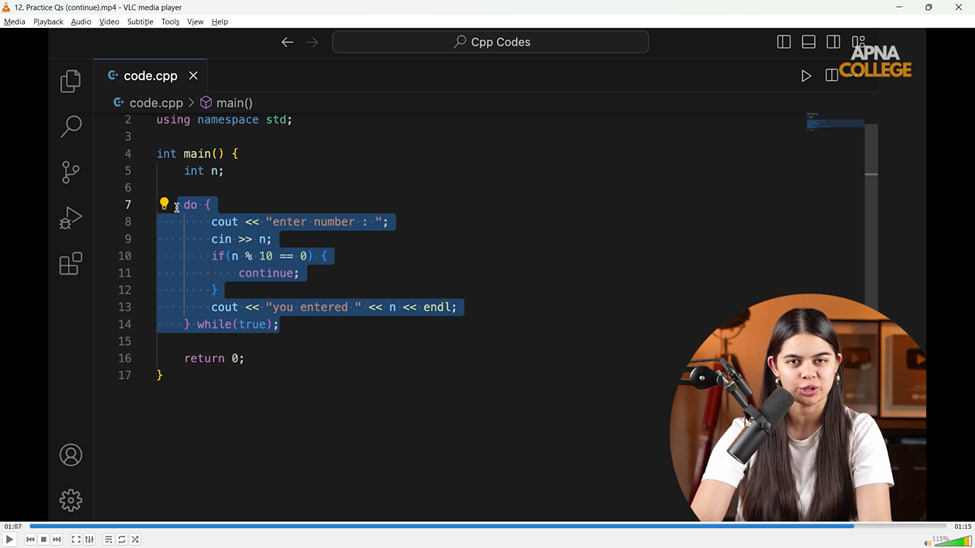
    //     m /= 10;

    // }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**Do-while loop –**

****

**Continue – **

**// Breaak Statem,ent -**

    // do- while loop basic syntax -

    // int val = 1;

    // do

    // {

    //     cout << "Learning do-while loop here " << endl;

    // } while (val > 5);

    // // Learning do-while loop here

    // while (val > 5)

    // {

    //     cout << "Learning using while loop here" << endl;

    // }

    // // Break statement in looping -

    // int i = 1;

    // while (i <= 10)

    // {

    //     if (i == 3)

    //     {

    //         break;

    //     }

    //     cout << i << endl;

    //     i++;

    // }

    /\*

    1

    2

    \*/

    // int n;

    // do {

    //     cout << "Enter Number - ";

    //     cin >> n;

    //     if (n % 10 == 0) {

    //         break;

    //     }

    //     cout << "You entered here - " << n << endl;

    // } while (true);

    // return 0;

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// Continue statement -**

    // for (int i = 1; i <= 10; i++)

    // {

    //     if (i == 3)

    //     {

    //         continue;

    //     }

    //     cout << i << endl;

    // }

    // /\*

    // 1

    // 2

    // 4

    // 5

    // 6

    // 7

    // 8

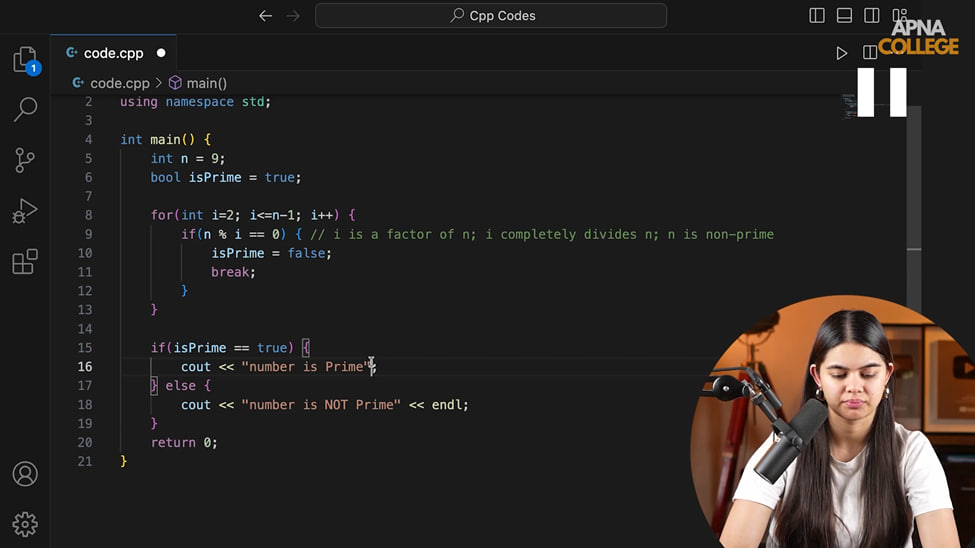
    // 9

    // 10

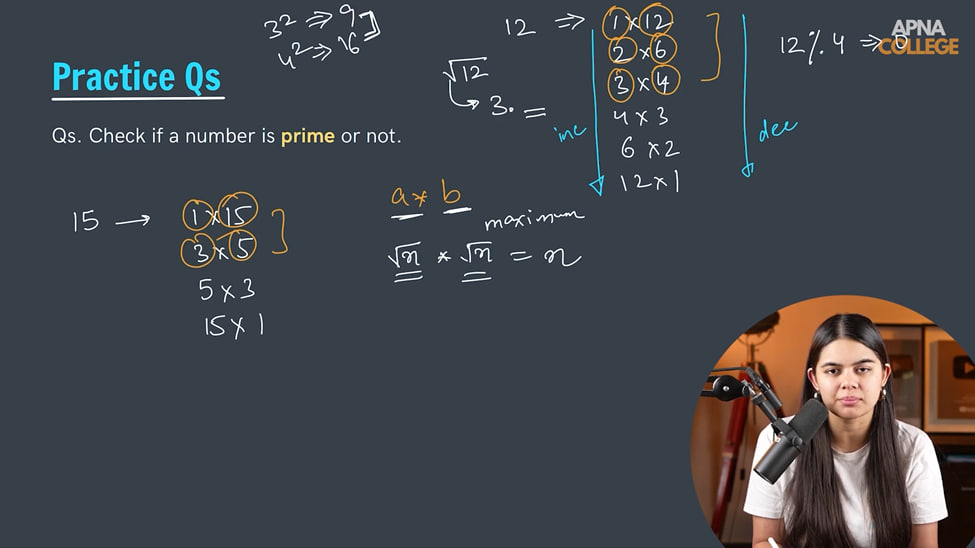
    // \*/

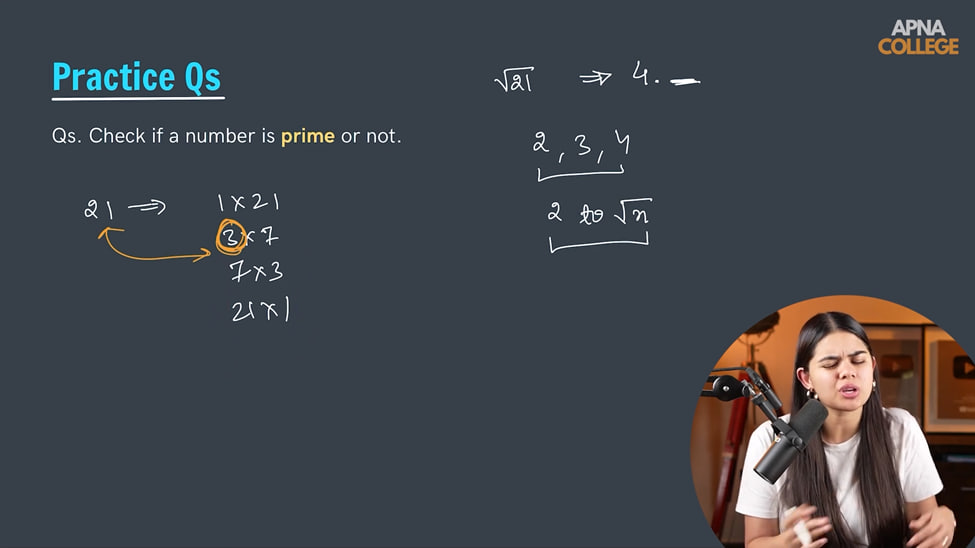
// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//4) Prime no. -**

****

**Prime Nuo. Squareroot logic –**

****

****

    // int num;

    // cout << "Enter the no. you wants to check for Prime or Not" << endl;

    // cin >> num;

    // if (num == 1)

    // {

    //     cout << "No. is Neither Prime Nor Compite" << endl;

    // }

    // for (int i = 2; i <= sqrt(num); i++)

    // {

    //     if (num % i == 0)

    //     {

    //         cout << "Not a Prime Number" << endl;

    //         return 0;

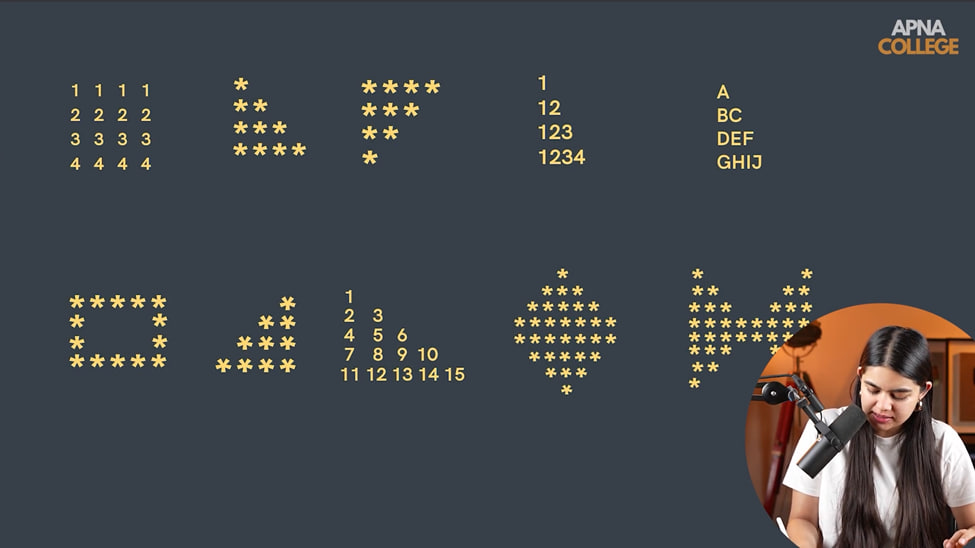
    //     }

    // }

    // cout << "prime number" << endl;

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**Patterns in cpp –**

****

**//5) Nested Loops  & Patterns –**

**// /\***

**// 1) Print the Nested Loop Pattern -**

**// 1 1 1 1**

**// 2 2 2 2**

**// 3 3 3 3**

**// 4 4 4 4**

**//  \*/**

    // int n;

    // cout << "n's value " << endl;

    // cin >> n;

    // for (int i = 1; i <= n; i++)

    // {

    //     for (int j = 1; j <= n; j++)

    //     {

    //         cout << i << " ";

    //     }

    //     cout << endl;

    // }

    // /\* O/p -

    // n's value

    // 6

    // 1 1 1 1 1 1

    // 2 2 2 2 2 2

    // 3 3 3 3 3 3

    // 4 4 4 4 4 4

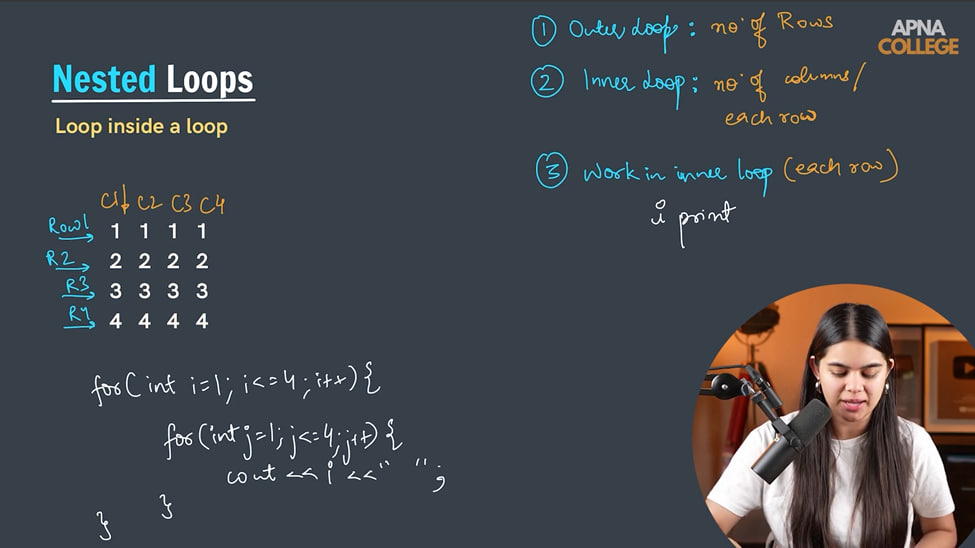
    // 5 5 5 5 5 5

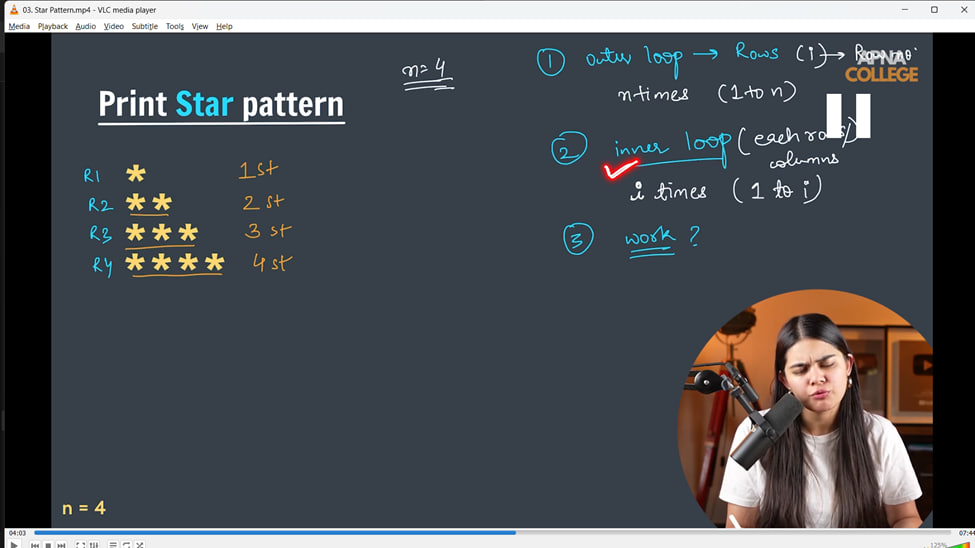
    // 6 6 6 6 6 6

    //  \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// Nested Loops -**

****

**5.2) Star Pattern**

**// /\*  2) Print the Star Pattern**

**// \***

**// \* \***

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

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

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

**//  \*/**

    //     int n;

    //     cout << "vakue of n - " << endl;

    //     cin >> n;

    //     for (int i = 1; i <= n; i++)

    //     {

    //         for (int j = 1; j <= i; j++)

    //         {

    //             cout << "\* ";

    //         }

    //         cout << endl;

    //     }

    // /\*

    // 5

    // \*

    // \* \*

    // \* \* \*

    // \* \* \* \*

    // \* \* \* \* \*

    //  \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// /\*  3) Print the Inverted Star Pattern**

**//  \* \* \* \* \***

**//  \* \* \* \***

**//  \* \* \***

**//  \* \***

**//  \***

**//  \*/**

    // int n;

    // cout << "vakue of n - " << endl;

    // cin >> n;

    // for (int i = 1; i <= n; i++)

    // {

    //     for (int j = 1; j <= n - i + 1; j++)

    //     {

    //         cout << "\* ";

    //     }

    //     cout << endl;

    // }

    // /\*

    // vakue of n -

    // 5

    // \* \* \* \* \*

    // \* \* \* \*

    // \* \* \*

    // \* \*

    // \*

    //  \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//  /\*  4) Print the half Pyramid  Pattern**

**// 1**

**// 1 2**

**// 1 2 3**

**// 1 2 3 4**

**// 1 2 3 4 5**

**//      \*/**

    //     int n;

    //     cout << "vakue of n - " << endl;

    //     cin >> n;

    //     for (int i = 1; i <= n; i++)

    //     {

    //         for (int j = 1; j <= i; j++)

    //         {

    //             cout << j << " ";

    //         }

    //         cout << endl;

    //     }

    // /\*

    // vakue of n -

    // 5

    // 1

    // 1 2

    // 1 2 3

    // 1 2 3 4

    // 1 2 3 4 5

    //  \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// /\***

**// 4.1) Print the  Pattern**

**// 1**

**// 2 2**

**// 3 3 3**

**// 4 4 4 4**

**// 5 5 5 5 5**

**// \*/**

    //     int n;

    //     cout << "vakue of n - " << endl;

    //     cin >> n;

    //     for (int i = 1; i <= n; i++)

    //     {

    //         for (int j = 1; j <= i; j++)

    //         {

    //             cout << i << " ";

    //         }

    //         cout << endl;

    //     }

    // /\*

    // vakue of n -

    // 5

    // 1

    // 2 2

    // 3 3 3

    // 4 4 4 4

    // 5 5 5 5 5

    //  \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// /\***

**//     5) Print the Character Pyramid Pattern**

**// A**

**// B C**

**// D E F**

**// G H I J**

**// K L M N O**

**//     \*/**

    //     int n;

    //     cout << "vakue of n - " << endl;

    //     cin >> n;

    //     char ch = 'A';

    //     for (int i = 1; i <= n; i++)

    //     {

    //         for (int j = 1; j <= i; j++)

    //         {

    //             cout << ch << " ";

    //             ch++;

    //         }

    //         cout << endl;

    //     }

    //     /\*

    //     vakue of n -

    //     5

    //     A

    //     B C

    //     D E F

    //     G H I J

    //     K L M N O

    //      \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// /\***

**// 5.1) Print the Character Pyramid Pattern for same Char in the line**

**// A**

**// B B**

**// C C C**

**// D D D D**

**// E E E E E**

**//     \*/**

    // int n;

    // cout << "vakue of n - " << endl;

    // cin >> n;

    // char ch = 'A';

    // for (int i = 1; i <= n; i++)

    // {

    //     for (int j = 1; j <= i; j++)

    //     {

    //         cout << ch << " ";

    //     }

    //     ch++;

    //     cout << endl;

    // }

    // /\*

    // vakue of n -

    // 5

    // A

    // B B

    // C C C

    // D D D D

    // E E E E E

    //  \*/

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

    /\*

**6) Print the Hollow Rectabgle Patternn**

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

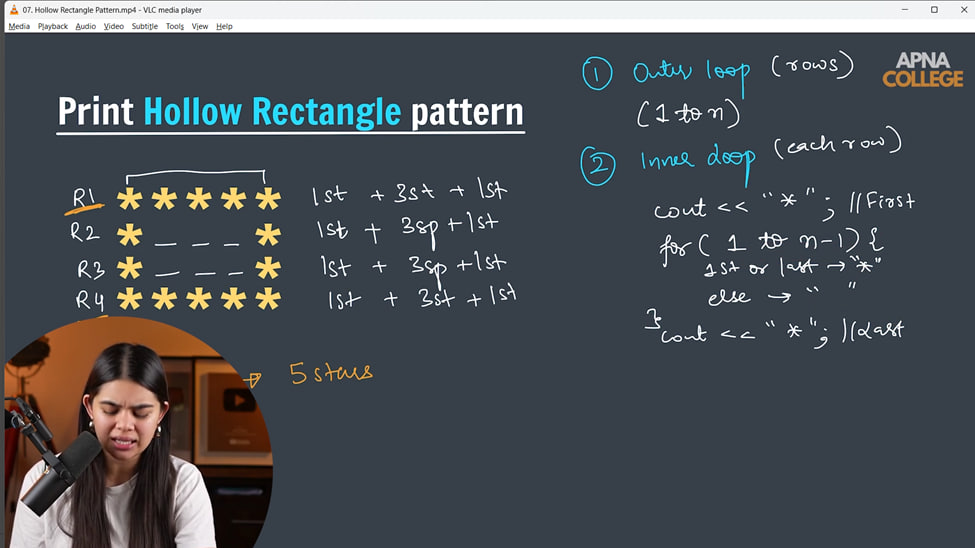
**\*       \***

**\*       \***

**\*       \***

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

**\*/**

****

// int n;

    // cout << "vakue of n - " << endl;

    // cin >> n;

    // for (int i = 1; i <= n; i++)

    // {

    //     cout<<"\*";//First Row

    //     for (int j = 1; j <= n - 1; j++)//sapce printing

    //     {

    //         if (i == 1 || i == n)

    //         {

    //             cout << "\*";

    //         }

    //         else

    //         {

    //             cout << " ";

    //         }

    //     }

    //     cout<<"\*"<<endl;//Last Row

    // }

    // Lgic 2

    // for (int i = 1; i <= n; i++)

    // {

    //     for (int j = 1; j <= n; j++)

    //     {

    //         if (i == 1 || i == n || j == 1 || j == n)

    //         {

    //             cout << "\* ";

    //         }

    //         else

    //         {

    //             cout<<"  ";

    //         }

    //     }

    //     cout << endl;

    // }

    /\*

    vakue of n -

    5

    \* \* \* \* \*

    \*       \*

    \*       \*

    \*       \*

    \* \* \* \* \*

     \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

// /\*

**// 7) Print Inverted & Rotated Half Pyramid**

**//         \***

**//       \* \***

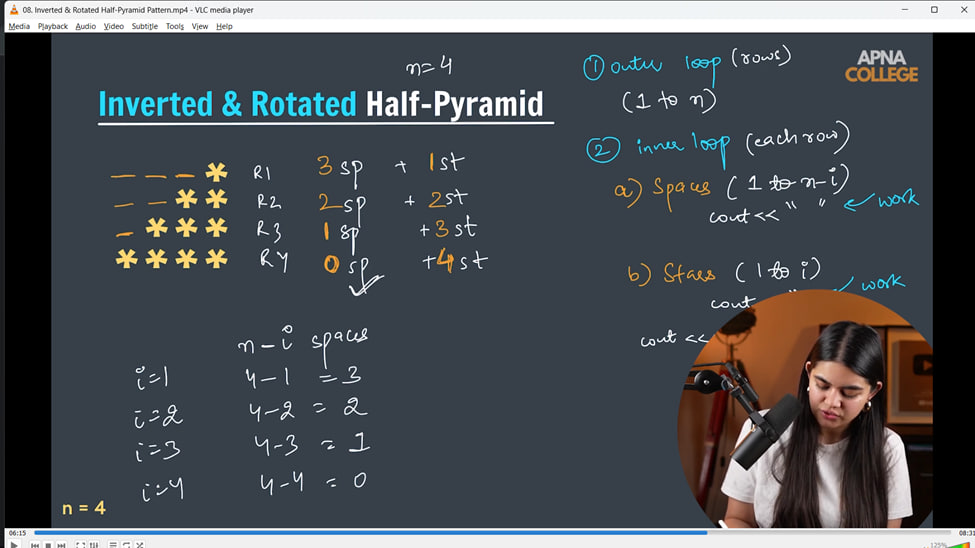
**//     \* \* \***

**//   \* \* \* \***

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

**// \*/**

****

****

//     int n;

    //     cout << "vakue of n - " << endl;

    //     cin >> n;

    //     for (int i = 1; i <= n; i++)

    //     {

    //         // Printoing SPce

    //         for (int j = 1; j <= n - i; j++)

    //         {

    //             cout << "  ";

    //         }

    //         // Printing Str

    //         for (int k = 1; k <= i; k++)

    //         {

    //             cout << "\* ";

    //         }

    //         cout << endl;

    //     }

    // /\*

    // vakue of n -

    // 5

    //         \*

    //       \* \*

    //     \* \* \*

    //   \* \* \* \*

    // \* \* \* \* \*

    //  \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// /\***

**// 8) Print Floyd's Triangle**

**// 1**

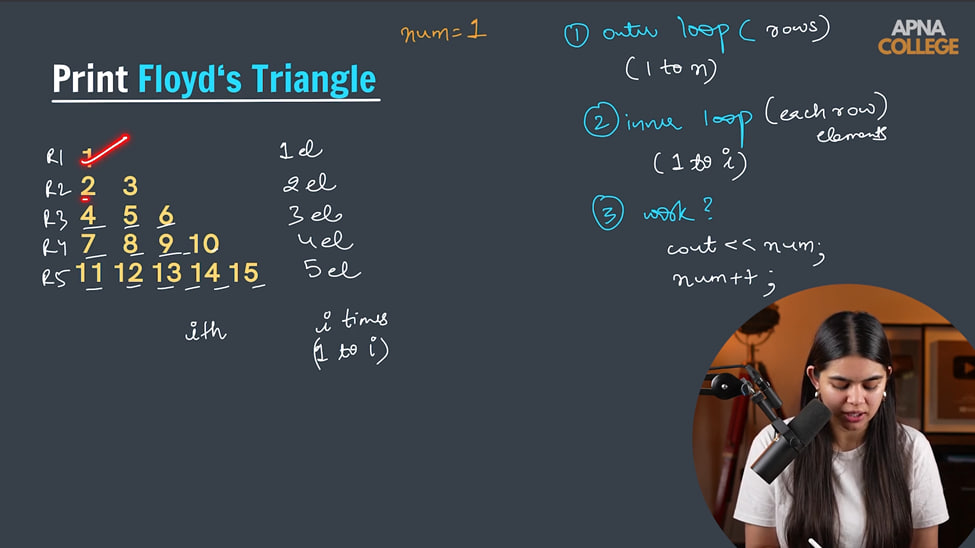
**// 2 3**

**// 4 5 6**

**// 7 8 9 10**

**// 11 12 13 14 15**

**// \*/**

****

// int n;

    // cout << "vakue of n - " << endl;

    // cin >> n;

    // int num = 1;

    // for (int i = 1; i <= n; i++)

    // {

    //     for (int j = 1; j <= i; j++)

    //     {

    //         cout << num << " ";

    //         num++;

    //     }

    //     cout << endl;

    // }

    // /\*

    // vakue of n -

    // 5

    // 1

    // 2 3

    // 4 5 6

    // 7 8 9 10

    // 11 12 13 14 15

    //  \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//     /\***

**//     9) Print the Diamond Pattern**

**//         \***

**//       \* \* \***

**//     \* \* \* \* \***

**//   \* \* \* \* \* \* \***

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

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

**//   \* \* \* \* \* \* \***

**//     \* \* \* \* \***

**//       \* \* \***

**//         \***

**//     \*/**

//     int n;

    //     cout << "vakue of n - " << endl;

    //     cin >> n;

    //     // 1st Pyramid

    //     for (int i = 1; i <= n; i++)

    //     {

    //         // Printing SPace

    //         for (int j = 1; j <= n - i; j++)

    //         {

    //             cout << "  ";

    //         }

    //         // Printing Star

    //         for (int k = 1; k <= 2 \* i - 1; k++)

    //         {

    //             cout << "\* ";

    //         }

    //         cout << endl;

    //     }

    //     // 2nd Pyramid

    //     for (int i = n; i >= 1; i--)

    //     {

    //         // Printing SPace

    //         for (int k = 1; k <= n - i; k++)

    //         {

    //             cout << "  ";

    //         }

    //         // Printing Star

    //         for (int j = 1; j <= 2 \* i - 1; j++)

    //         {

    //             cout << "\* ";

    //         }

    //         cout << endl;

    //     }

    // /\*

    // vakue of n -

    // 5

    //         \*

    //       \* \* \*

    //     \* \* \* \* \*

    //   \* \* \* \* \* \* \*

    // \* \* \* \* \* \* \* \* \*

    // \* \* \* \* \* \* \* \* \*

    //   \* \* \* \* \* \* \*

    //     \* \* \* \* \*

    //       \* \* \*

    //         \*

    //  \*/

    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**/\***

**10) Print the Butterfly Pattern**

**\*                 \***

**\* \*             \* \***

**\* \* \*         \* \* \***

**\* \* \* \*     \* \* \* \***

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

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

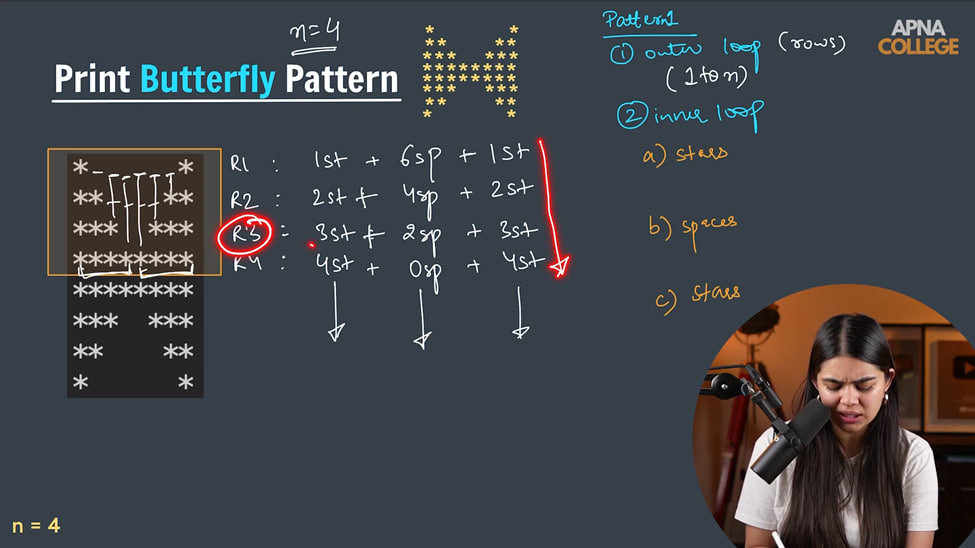
**\* \* \* \*     \* \* \* \***

**\* \* \*         \* \* \***

**\* \*             \* \***

**\*                 \***

**\*/**

****

int n;

    cout << "vakue of n - " << endl;

    cin >> n;

    // 1st Butterfly

    // Printing Star

    for (int i = 1; i <= n; i++)

    {

        for (int j = 1; j <= i; j++)

        {

            cout << "\* ";

        }

        // Printing SPace

        for (int j = 1; j <= 2 \* (n - i); j++)

        {

            cout << "  ";

        }

        // Prinmting Start for last

        for (int k = 1; k <= i; k++)

        {

            cout << "\* ";

        }

        cout << endl;

    }

    // 2nd Half Butterfly

    for (int i = n; i >= 1; i--)

    {

        for (int j = 1; j <= i; j++)

        {

            cout << "\* ";

        }

        // Printing SPace

        for (int j = 1; j <= 2 \* (n - i); j++)

        {

            cout << "  ";

        }

        // Prinmting Start for last

        for (int k = 1; k <= i; k++)

        {

            cout << "\* ";

        }

        cout << endl;

    }

    /\*

    vakue of n -

    5

    \*                 \*

    \* \*             \* \*

    \* \* \*         \* \* \*

    \* \* \* \*     \* \* \* \*

    \* \* \* \* \* \* \* \* \* \*

    \* \* \* \* \* \* \* \* \* \*

    \* \* \* \*     \* \* \* \*

    \* \* \*         \* \* \*

    \* \*             \* \*

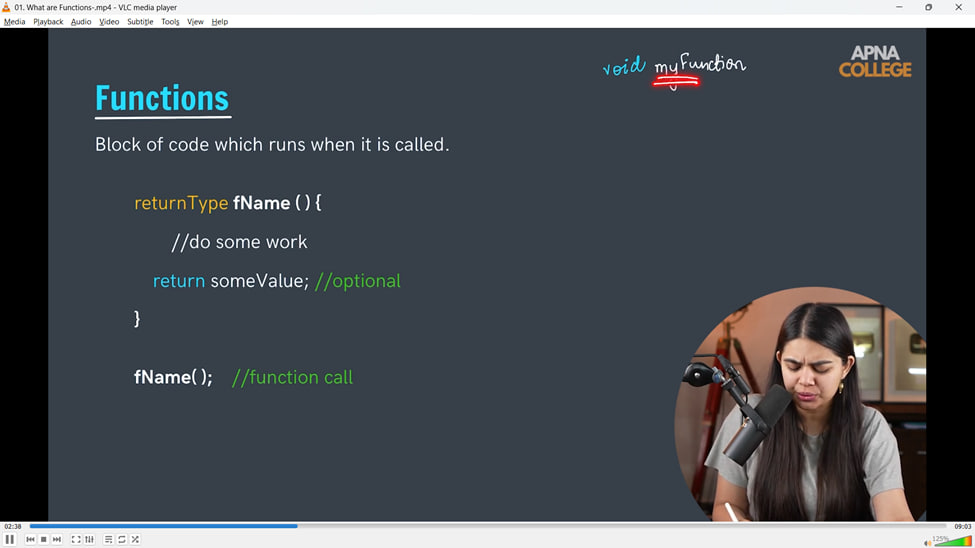
    \*                 \*

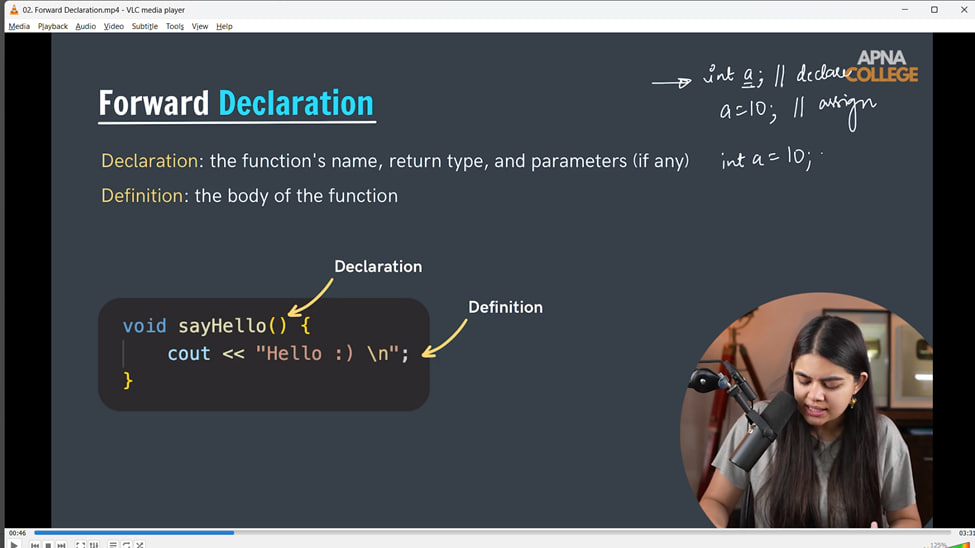
     \*/

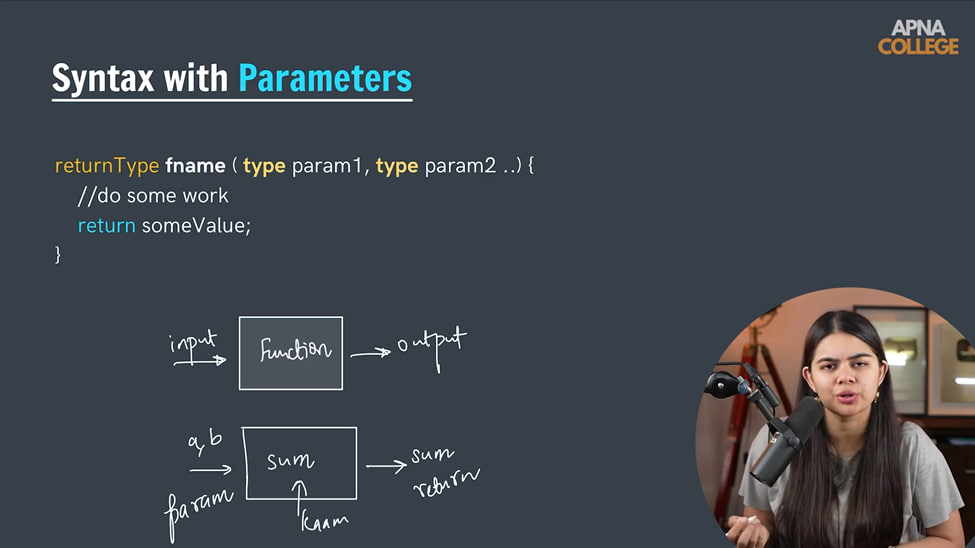
    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

}

**Functions in cpp –**

****

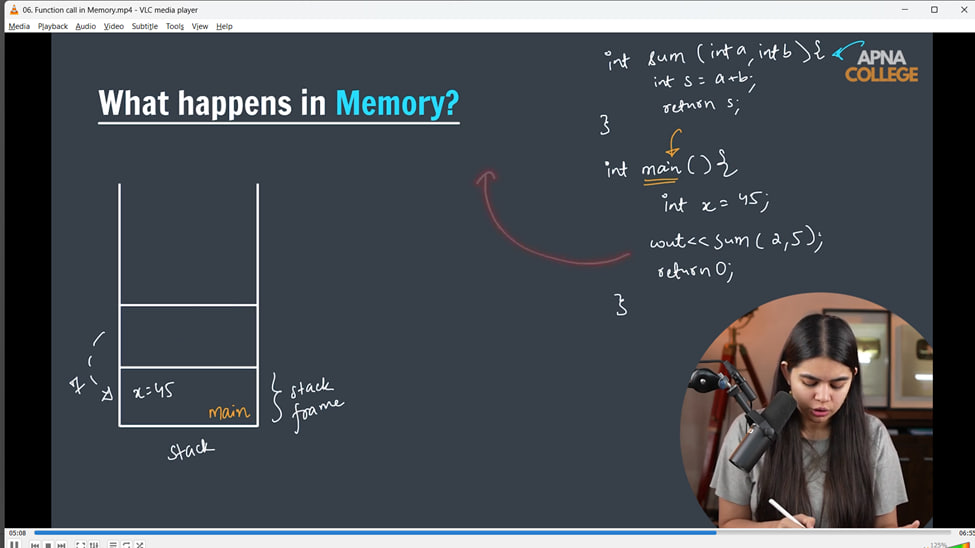
****

****

**Parameters – Function declare krne ke baad usme jo variables pass kiya jaate h is called Paramaeter.**

**Arguments – Function caaling me jo value pass hoti he use Arguments.**

**Forwaard Declaration - When a function os declared before main function and defininf after the mauin fucntion. It's fdorward Declaration. Pehele hi declare kr diya and ab define kr rheh**

****

// void sayHello() // Function Declaration

// {               // inside the function its Funciton Definition - How is the function defined to do work

//     cout << "Hello, Shubham you'll be get placed in Microsoft in August 2025" << endl;

//     cout << "All the very best." << endl;

// }

// void assistance()

// {

//     sayHello();

//     cout << "Work done boss" << endl;

// }

// int sum(int a, int b){ // a & b are Parameters here.

//     int sum = a + b;

//     return sum;

// }

// int mul(int a, int b){

//     int mul = a\*b;

//     return mul;

// }

// int sub(int a, int b=1){ //passing a value in 2nd parameter as by default 1, if argumeent didn't pass in function then system  will take this 1 as 2nd arguement

//     int sub = a-b;

// }

// int main()

// {

//     sayHello();

//     sayHello(); // Function Calling

//     sayHello();

//     assistance();

//     /\*

//     Hello, Shubham you'll be get placed in Microsoft in August 2025

//     All the very best.

//     Hello, Shubham you'll be get placed in Microsoft in August 2025

//     All the very best.

//     Hello, Shubham you'll be get placed in Microsoft in August 2025

//     All the very best.

//     Hello, Shubham you'll be get placed in Microsoft in August 2025

//     All the very best.

//     Work done boss

//      \*/

//     //  Forwaard Declaration - When a function os declared before main function and defininf after the mauin fucntion. It's fdorward Declaration. Pehele hi declare kr diya and ab define kr rheh

//     cout<<sum(2,4)<<endl; // 2,4 are arguements which are need to be pass in the parameters

//     cout<<mul(2,4)<<endl;// 8

//     cout<<sub(2)<<endl;//1 - when no 2nd argumenet passed but 2nd parameter defined then by default defined 2nd parameter 1 will becoming the 2nd arguement here.

//     cout<<sub(5,2)<<endl;//3 - here when, on passing both the argumenets, system will not consider the byu defaukt defined the 2nd parameter = 1 or any value

//     return 0;

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

// int Prod(int a, int b = 2)

// {

//     int theMultiplicaiton = a \* b;

//     return theMultiplicaiton;

// }

// bool isprime(int n)

// {

//     if (n == 1)

//     {

//         cout << "Hey, it's exceptional dude neither prime nor composite buddy" << endl;

//     }

//     for (int i = 2; i <= sqrt(n); i++)

//     {

//         if (n % 2 == 0)

//         {

//             return false;

//         }

//         else

//         {

//             return true;

//         }

//     }

//     return isprime;

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

// int factorial(int c)

// {

//     int fact = 1;

//     for (int i = 1; i <= c; i++)

//     {

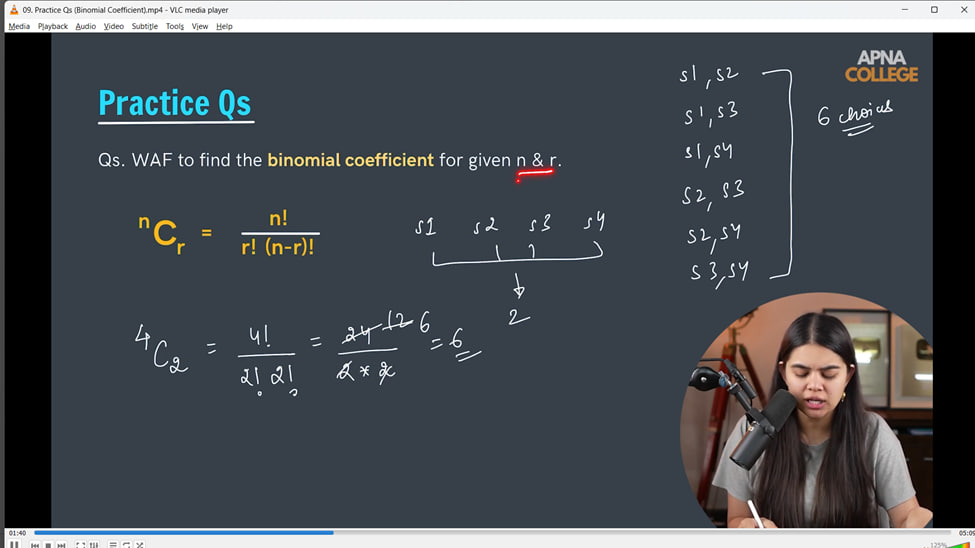
//         fact \*= i;

//     }

//     cout << "Factorial of no. - " << c << " is - " << fact << endl;

//     return fact;

// }



// int BinomialCoefficient(int n, int r)

//     {

//        int val1 = factorial(n);

//        int val2 = factorial(r);

//        int val3 = factorial(n-r);

//        int result = val1 / (val2 \* val3);

//        cout<<"BinomialCoeffiecent result is - "<<result<<endl;

//     }

// int main()

// {

//     cout << Prod(5) << endl;      // 10 - by using default defined parametr 2 with 1 argumenet

//     cout << Prod(5, 10) << endl;  // 50 - by using both  parameters with both arguements

//     cout << isprime(23) << endl;  // 1

//     cout << isprime(2) << endl;   // 1 - Prime Numbe

//     cout << isprime(8) << endl;   // 0 - Not a prime number

//     isprime(1);                   // Hey, it's exceptional dude neither prime nor composite buddy

//     cout << factorial(5) << endl; // 120

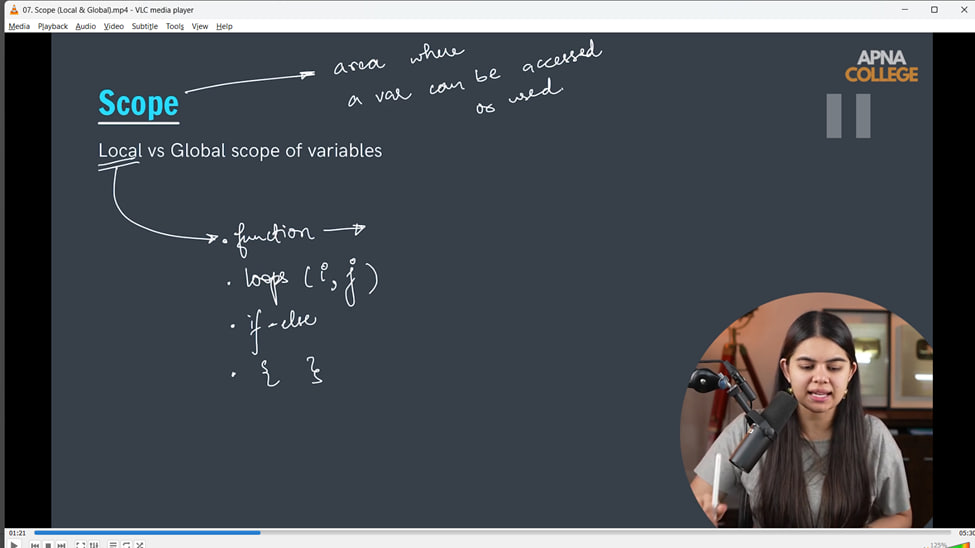
//     factorial(10);                // Factorial of no. - 10 is - 3628800

//     BinomialCoefficient(4,2);

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// // Scoping in Funciton -**



// /\*

// Locally Defined- If we define a variable only in the declaring funciton and want to use in the main funciton then - Not Pssible as inly locally declard

// Locally Defined- If we define the variable in any loop or in any if else statement then, outside of that we can't access the varfiable as defined withi that local scope.

// Globally Defined- So, for avoiding such issues, we can define particular variable at the top just after the header files. Above the declaratory function as weell as main function

//  \*/

// int num = 25;

// int sum(int a, int b)

// {

//     int addition = a + b;

//     cout << addition << endl;

//     cout << "No. which is declared globally is - " << num << endl;

// }

// int main()

// {

//     sum(15, 10);

//     cout << "No. which is declared globally is - " << num << endl;

// /\*

// 25

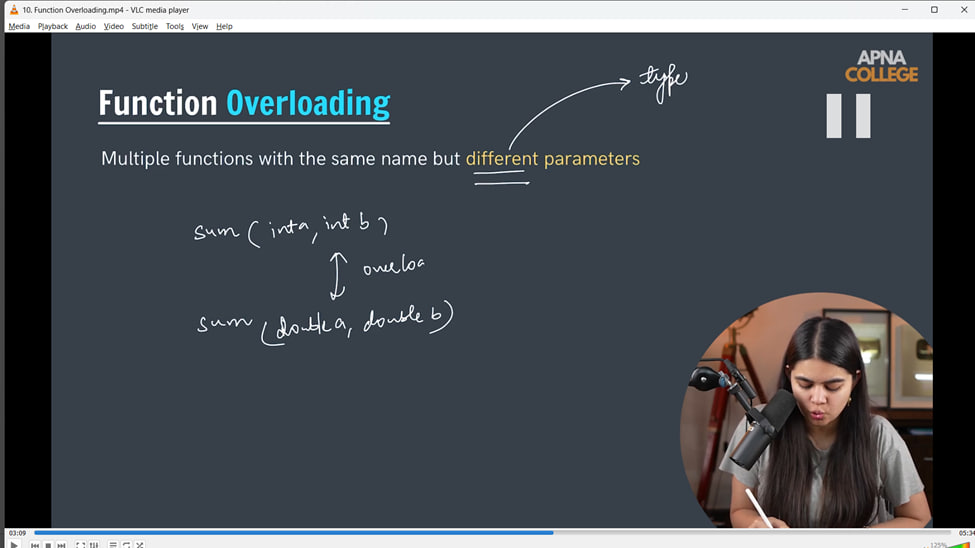
// No. which is declared globally is - 25

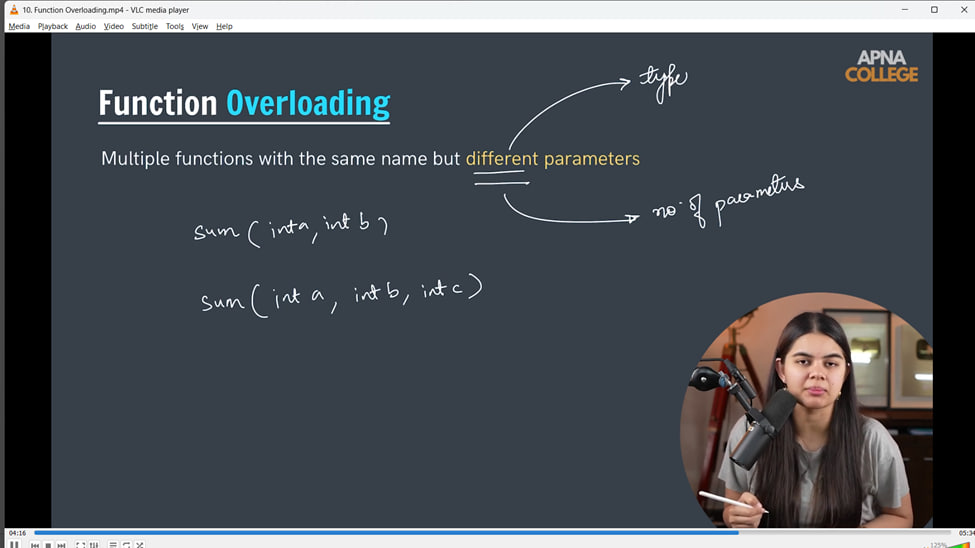
// No. which is declared globally is - 25

//  \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_





// int sum(int a, int b)

// {

//     cout<<(a+b)<<endl;

//     return a+b;

// }

// int sum(double a, double b)

// {

//     cout<<(a+b)<<endl;

//     return a+b;

// }

// int sum(int a, int b, int c)

// {

//     cout<<(a+b+c)<<endl;

//     return a+b+c;

// }

// int main()

// {

//     sum(45,5);//50

//     sum(12.5,2.5);//14 -

//     // it is 15 actually but due to considering as a double value compiler giving 14, so for overcoming trhis issueds we can use the same funciton with doffernt data gtype - it's called as fduncitn  ovberloading

//     sum(12.5,2.5);//15 - compiler won't confuse here ki kisme value pass kare, vo datatype ke according kr lega, its overloading

//     // AND THE SAME GOES FOR multiple paramters wuith same funciton name

//     sum(5,4,3);//12

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

// QUn - Print all prime numbers from 2 to range n

// bool isPrime(int n)

// {

//     if (n == 1)

//     {

//         cout << "No prime nor Composite" << endl;

//     }

//     for (int i = 2; i\*i <=n; i++)

//     {

//         if (n % i == 0)

//         {

//             return false;

//         }

//     }

//     return true;

// }

// void AllPrimes(int n)

// {

//     for (int i = 2; i <= n; i++)

//     {

//         if (isPrime(i))

//         {

//             cout << i << "  is prime number" << endl;

//         }

//     }

//     cout<<endl;

// }

// int main()

// {

//     AllPrimes(13);

//     return 0;

// /\*

// 2  is prime number

// 3  is prime number

// 5  is prime number

// 7  is prime number

// 11  is prime number

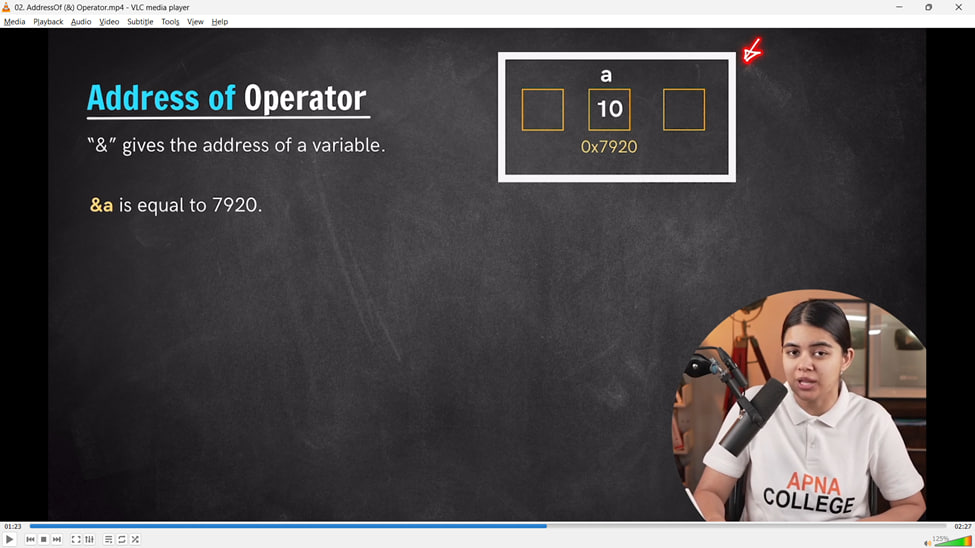
// 13  is prime number

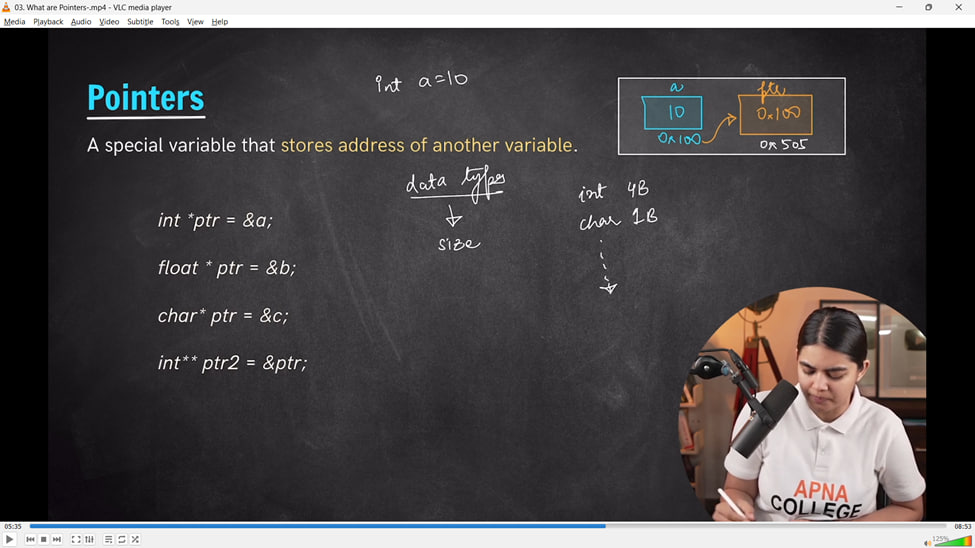
//  \*/

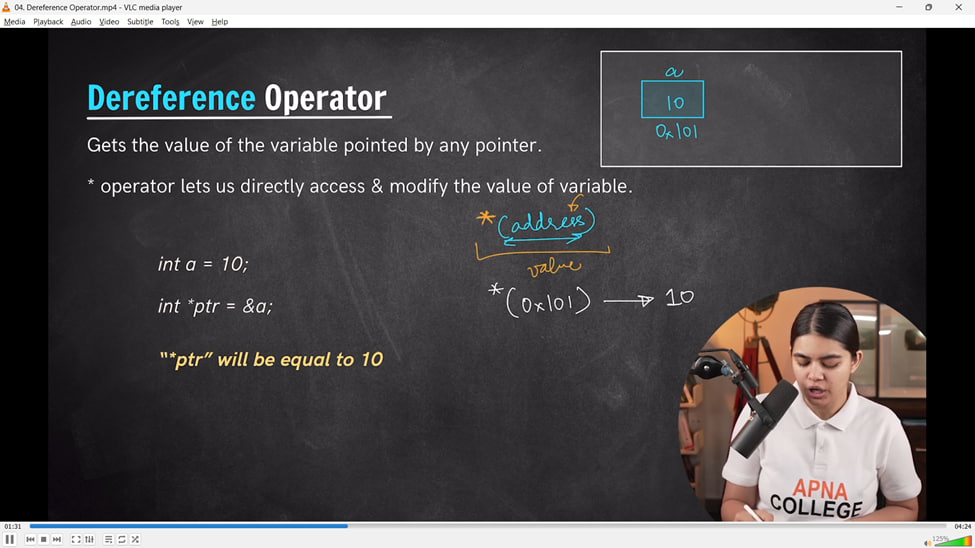
// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// Pointers in cpp -**







// int main()

// {

//     int c = 10;

//     int \*ptr = &c;                     // int ptr tyep can never store the value, it always stores the address itself

//     cout << &c << "==" << ptr << endl; // 0x61fed4==0x61fed4

//     cout << sizeof(ptr) << endl;//4 - size of int pointer

//     float p = 3.1416;

//     float \*ptr2 = &p;

//     cout << &p << "==" << ptr2 << endl;//0x61ff00==0x61ff00

//     cout << sizeof(ptr2) << endl;//4 - size of float pointer

//     // Pointer of Pointer Approach -

//     int \*\*pptr = &ptr;

//     cout<<&ptr<<"=="<<pptr<<endl;//0x61ff00==0x61ff00

//     // Dereference Operator -

//     int a = 20;

//     int \*ptr3 = &a;

//     cout<<&a<<endl;//0x61fef8

//     cout<<\*(&a)<<endl;//20

//     cout<<\*ptr3<<endl;//20

//     // Null Pointer

//     // if pointer is not pointing the address of any variable then this will give Garbage Value

//     int \*ptr4;

//     cout<<ptr4<<"\n";//0x76e18c1d - Automatically gives Garbage Value

//     int \*ptr5=NULL;

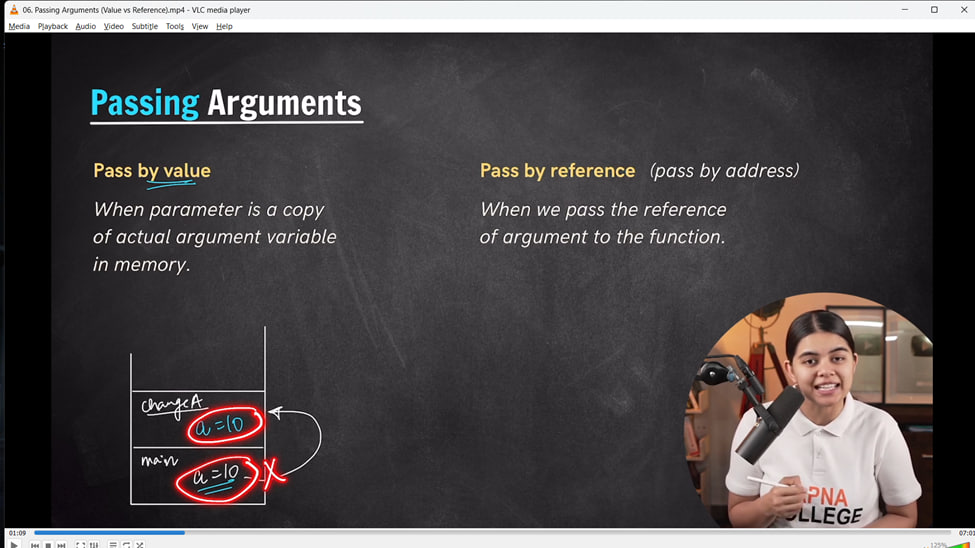
//     cout<<ptr5<<endl;//0x74e18c1d

//     cout<<\*ptr5<<endl;//0

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**Passing Arguments**

****

**Pass by Value –**

// Passing Arguements in funciton -

// Pass by value - when parameter is a copy of actual arguement variable in memory

// void changeA(int a)

// {

//     a=30;

//     cout<<a<<endl;

// }

// int main()

// {

//     int a = 10;

//     changeA(a);//30 - Funciton call kiya he to function declaratin me jaa kr function me variable ki assigned value ko call krega

//     cout<<a<<endl;//10 - vapas se main fun me aa gya h to iss scope me jo variable defined h usko value ko print krega

// }

// Pass by Reference - using pointer

// void changeB(int \*ptr)

// {

//     \*ptr = 20;

//     cout << \*ptr << endl;

// }

// int main()

// {

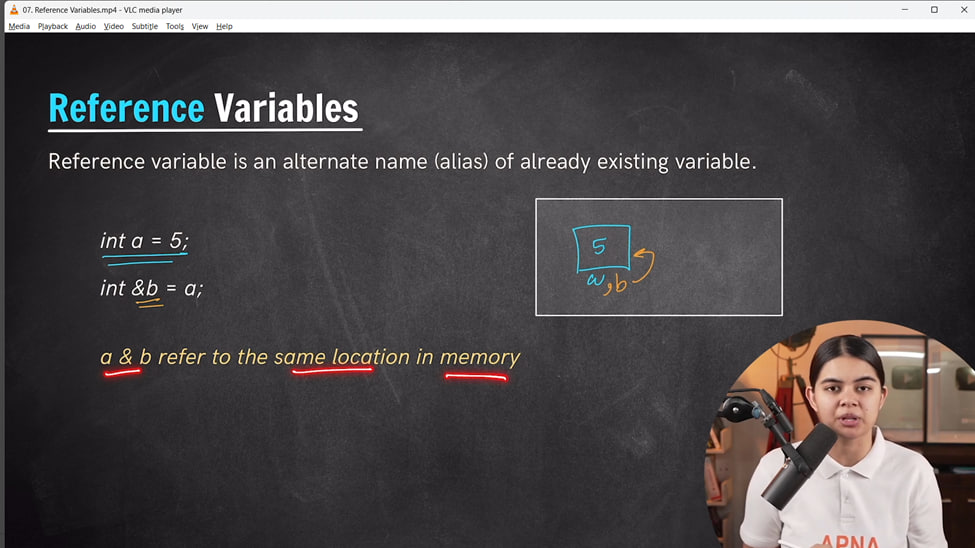
//     int a = 10;

//     changeB(&a);//20 - call by address he. Ptr me &a h and function declaration me us pointer se uski valuie assign hui he. to calling me funciton ki under variable ki assigned value print hogi

//     cout << a << endl;//20  - return to main function - to uske ander var ki jo value defined he voprint hogi

// }

**// Pass by Reference - using Reference Variables**

****

// void changeC(int \*ptr)

// {

//     \*ptr = 20;

//     cout<<\*ptr<<endl;

// }

// void changeD(int &c)

// {

//     c=20;

//     cout<<c<<endl;

// }

// int main()

// {

//     int a = 10;

//     int &b = a;

//     b=25;

//     cout<<b<<endl;//25

//     cout<<a<<endl;//25

//     int c =10;

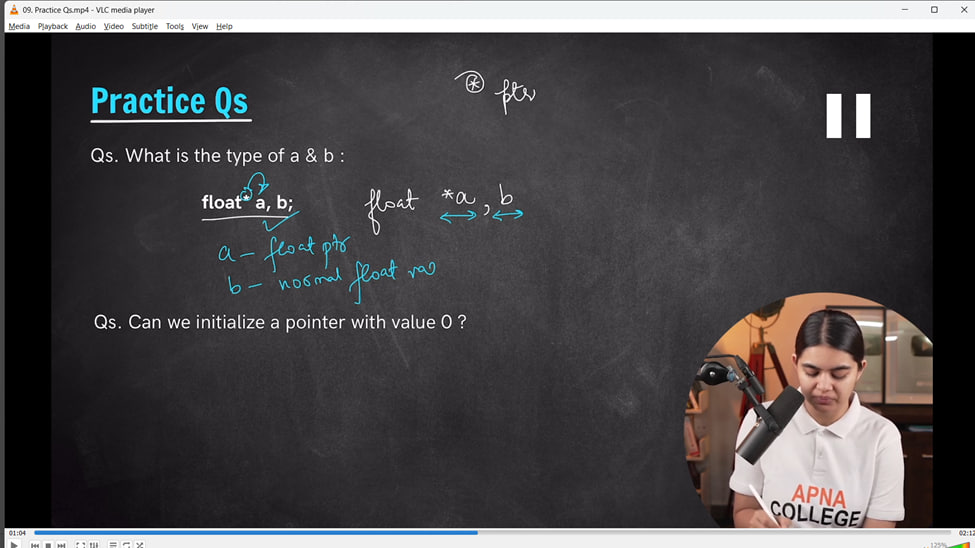
//     changeD(c);//20

//     cout<<c<<endl;//20 - yaha pr pointer ke saath reference diya h fiun declaration me to main value me bhi change aayag

//     // This is call by reference using reference variable. memory will back to main function but as we're used reference var, so there will also change in main fun too.

// }

****

****

// Qun -

// int main()

// {

//     int x=5, y=10;

//     int \*ptr1 = &x, \*ptr2 = &y;

//     ptr2=ptr1;

//     cout<<ptr2<<endl;//0x61ff04

//     cout<<ptr1<<endl;//0x61ff04

//     cout<<&x<<endl;//0x61ff04

//     // What os the typr of the following variables -

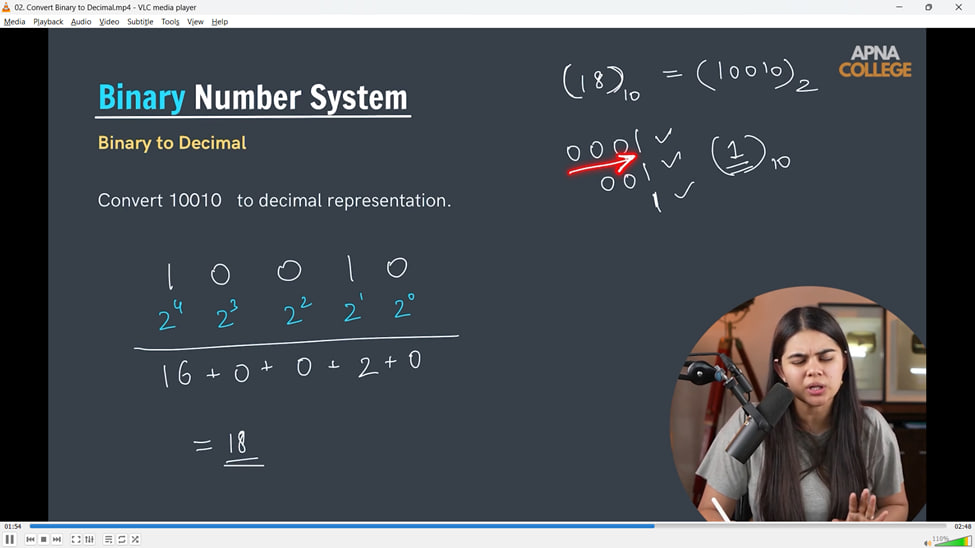
//     float\*a,b;

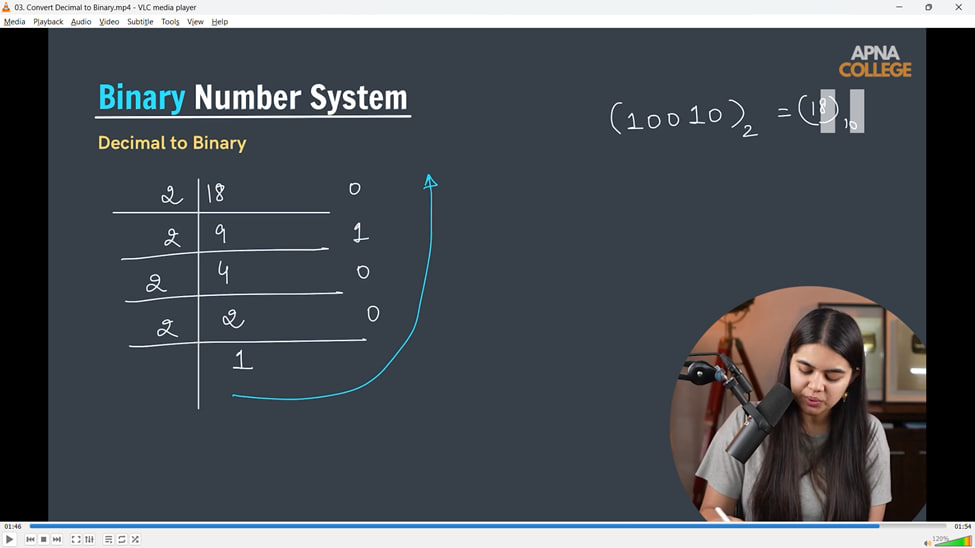
//     // a - float pointer, b - normal float variable

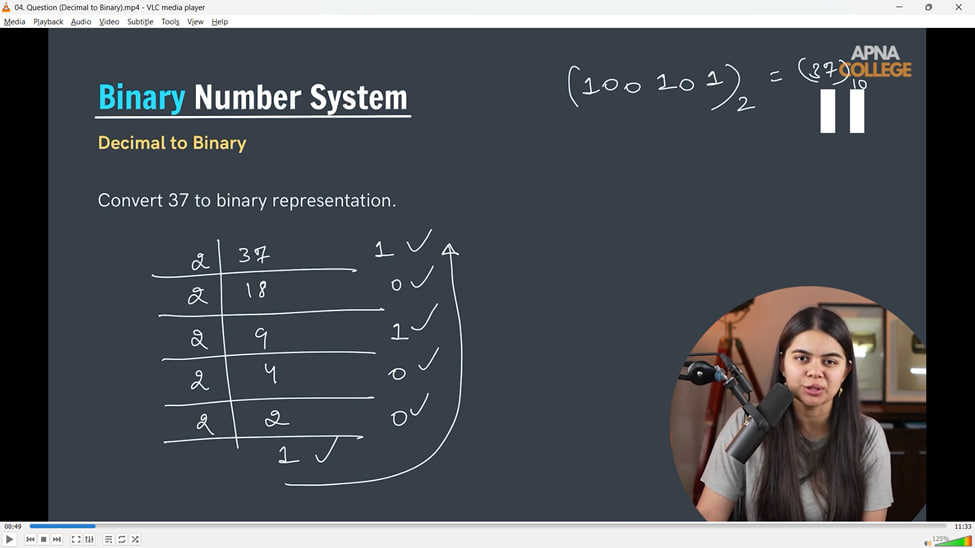
// }

**Number System in Programming–**

**Binary to Decimal –**

****

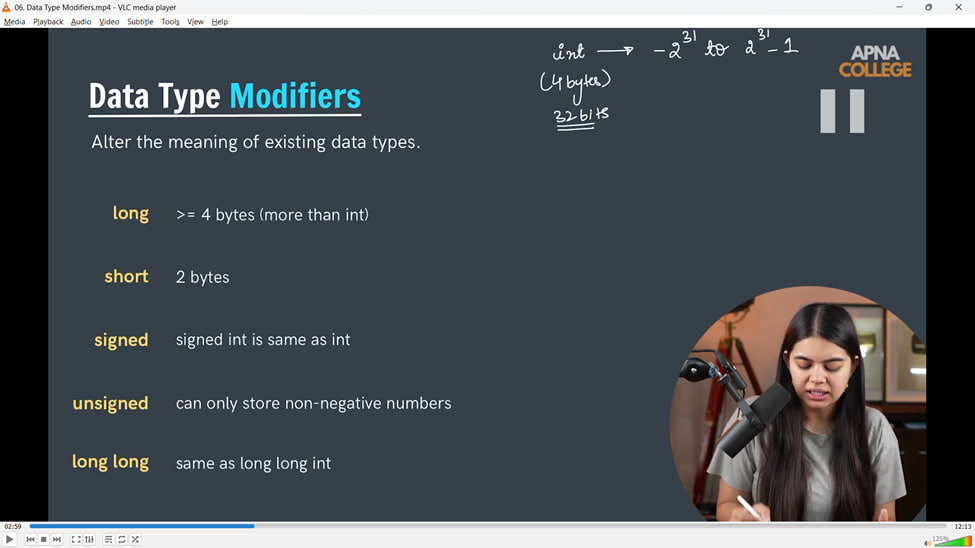
****

****

****

****

**Data Type Modifiers & Range of Int –**



**Binary to Decimal**



// void binToDec(int binNum)

// {

//     int n = binNum;

//     int decNum = 0;

//     int pow = 1; // 2^0 2^1 2^2 2^3..

//     while (n > 0)

//     {

//         int lastDig = n % 10;

//         decNum += lastDig \* pow;

//         pow = pow \* 2;

//         n = n / 10;

//     }

//     cout << decNum << endl;

// }

// int main()

// {

//     // Size of for different int data types -

//     // cout<<sizeof(int)<<endl;//4

//     // cout<<sizeof(long int)<<endl;//4

//     // cout<<sizeof(short int)<<endl;//2

//     // cout<<sizeof(long long int)<<endl;//8

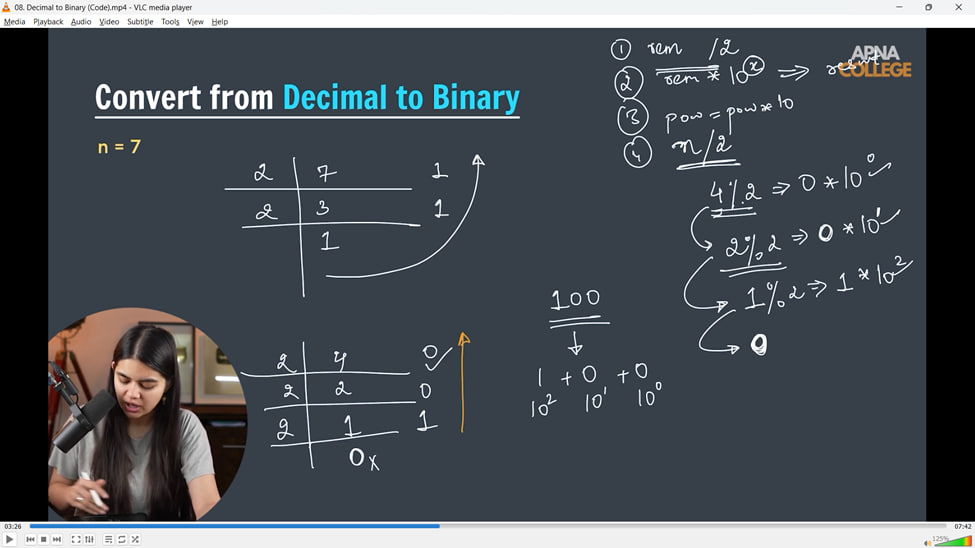
//     binToDec(101);//5

//     binToDec(1011);//11

//     binToDec(1111);//15

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**Decimal to Bianry – **

void DecToBin(int DecNum)

{

    int n = DecNum;

    int pow = 1; // 10^0, 10^1, 10^2...

    int BinNum = 0;

    while (n > 0)

    {

        int rem = n % 2;

        BinNum += rem \* pow;

        n = n / 2;

        pow = pow \* 10;

    }

    cout<<BinNum<<endl;

}

int main()

{

    DecToBin(15);//1111

    DecToBin(4);//100

}

/\*

Do the comparision b/w botht the methods -

in Bin to Dec - Modulus by 10 as well as devide by 10 and pow of 2, while

in Dec to Bin - Modulues/Remainder by 2 as well as divide by 2 and pow of 10

 \*/

**// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**Arrays-**

1)Initiation Methods of an array and Output & input of an array -

2) Largest Value/Max Element in the array & 2.1) Min Element -

3) Dereferences of Pointer concept in Array -

4) Linear Search in array -

5) Reverse an array -

5.1) using extra space

5.2 - W/o using Extra Space1 - 2 POINTERS APPROACH

6) Binary Search - It always for the Sorted Array

7) Array Pointer -

7.1. Pointer Arithmetic Approaches -

7.2 - Addition & Subtraction of Constants -

7.3- Addition & Subtraction of Pointers

7.4 - Comparison Operators

8) Subarrays -

9) Max Subarray Sum -

9.1 - Clearly Brute Force Approach -

9.2 - Slightly Optimized approach - for decreasing the time complexity

9.3 - Using Most Optimized Approach - Kadane's Algorithms. For very less TIme Complexity

10) buy & sell stock problem-

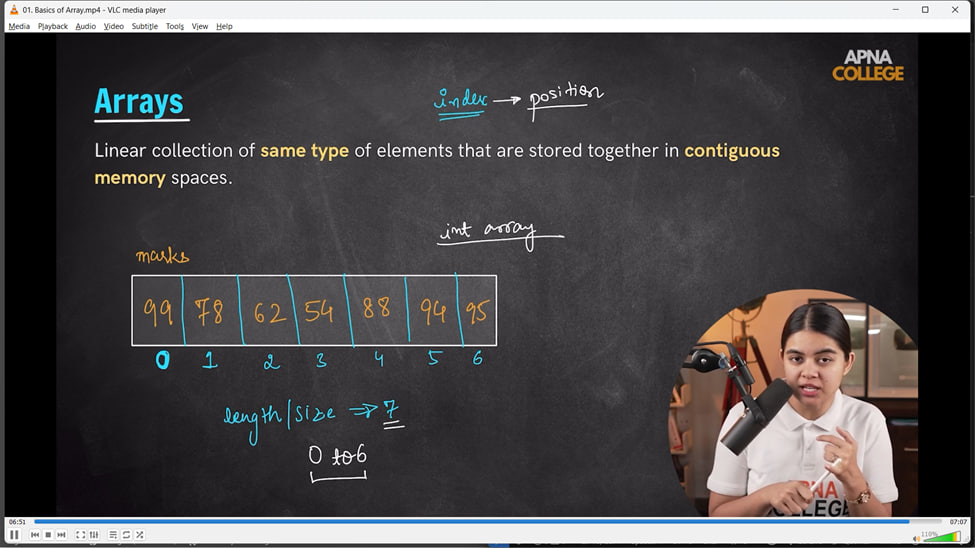
11) Trapping Rainwater Problem -

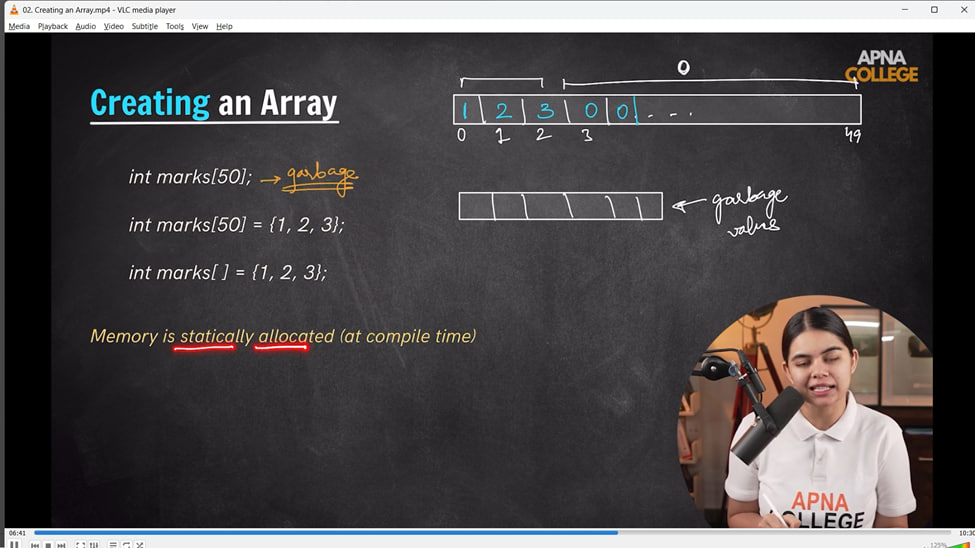
**2) Array –**

**Arrays in cpp -**Linear Collection of Same Type of ELements that are stored together in

Contiguos Memory Spaces.

**//1)Initliazetion Methods of an array -**

****

****

// int marks[50];             // 0 to 49

// cout << marks[0] << endl;  // 4199136 - whn we don;t innitialize the array with any value it takes any garbage value in the memory

// cout << marks[1] << endl;  // 17566816

// cout << marks[2] << endl;  // 17566816

// cout << marks[50] << endl; // 4201056

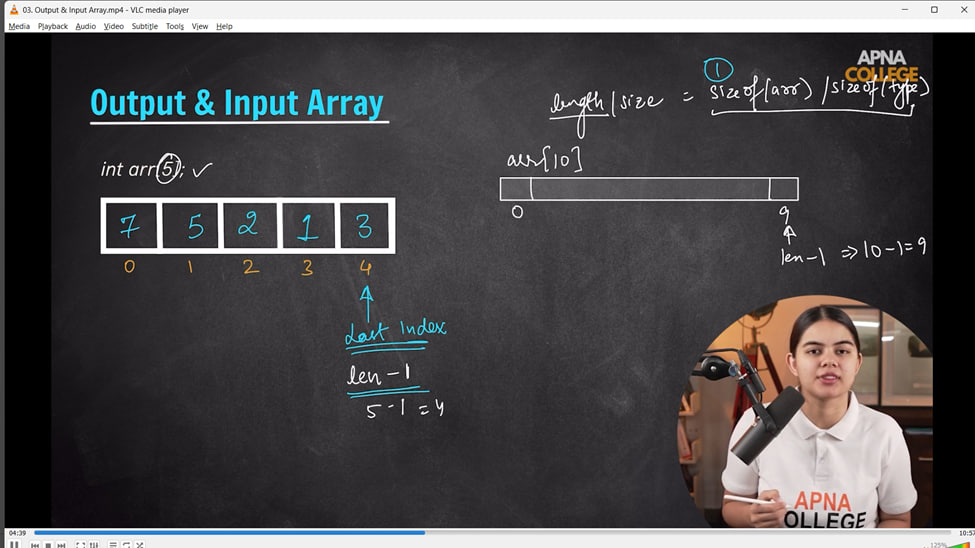
// int percentage[5] = {0, 1, 2, 3, 4};

// cout << percentage << endl;

// cout << sizeof(percentage) << endl;                                          // 20 - int size = 4 bytes , so 5 int = 4\*5 = 20 bytes

// cout << "lenght of array is - " << sizeof(percentage) / sizeof(int) << endl; // lenght of array is - 5

**// 1.1)// Ouput & input of an array -**

****

// 1.1)// Ouput & input of an array -

// int arr[5] = {0, 1, 2, 3, 4};

// for (int i = 0; i < 5; i++)

// {

//     cout << i << " "; // 0 1 2 3 4

// }

// cout << endl;

// int len = sizeof(arr) / sizeof(int);

// for (int idx = 0; idx <= len - 1; idx++)

// {

//     cout << arr[idx] << " "; // 0 1 2 3

// }

// cout << endl;

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 2) Largets Value/Max Element in the array -**

// int n;

// cout << "n's value - " << endl;

// cin >> n;

// cout << "ENte array values - " << endl;

// int arr2[n];

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

// {

//     cin >> arr2[i];

// }

// cout << "Entered array's value is - " << endl;

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

// {

//     cout << arr2[i] << " ";

// }

// cout << endl;

// int maxEle = arr2[0];

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

// {

//     if (arr2[i] > maxEle)

//     {

//         maxEle = arr2[i];

//         cout << "Current assigned max value is - " << arr2[i] << endl;

//     }

// }

// cout << "So, Finally max value is - " << maxEle << endl;

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 2.1) Min Element  -**

//     int n;

//     cout << "n's value - " << endl;

//     cin >> n;

//     cout << "ENte array values - " << endl;

//     int arr2[n];

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr2[i];

//     }

//     cout << "Entered array's value is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr2[i] << " ";

//     }

//     cout << endl;

//     // For Min Element

//     int minEle = arr2[0];

//     for (int i = 0; i < n; i++)

//     {

//         if (arr2[i] < minEle)

//         {

//             minEle = arr2[i];

//             cout << "Current assigned min value is - " << arr2[i] << endl;

//         }

//     }

//     cout << "So, Finally min value is - " << minEle << endl;

// /\*

// n's value -

// 6

// ENte array values -

// 89 45 23 12 56 78

// Entered array's value is -

// 89 45 23 12 56 78

// Current assigned min value is - 45

// Current assigned min value is - 23

// Current assigned min value is - 12

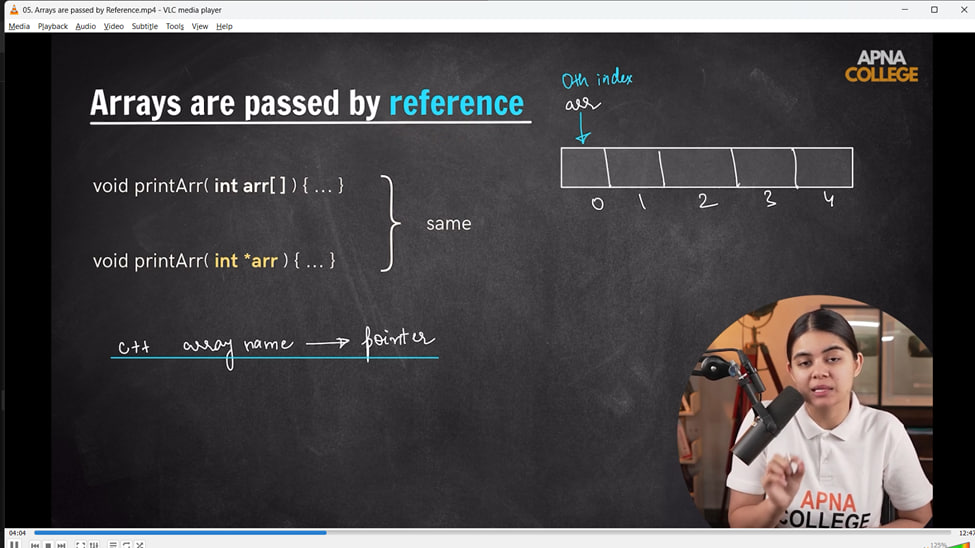
// So, Finally min value is - 12

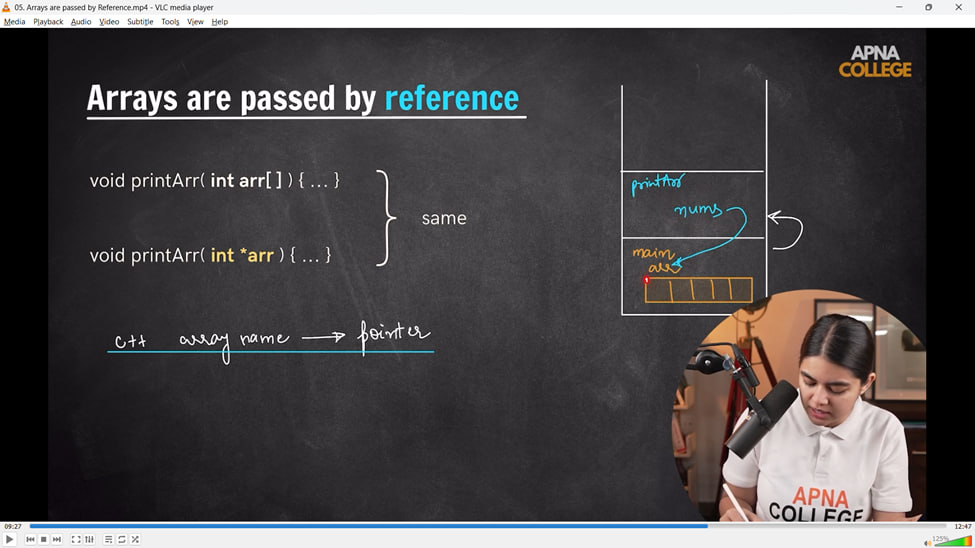
//  \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 3)  Dereferences of Pointer concept in Array -**

****

****

// Array's are by default passed by reference. The value change in function affects in main fun also

// void func(int arr[])

// {

//     arr[0] = 1000;

//     cout << arr[0] << endl;

// }

// void func2(int \*ptr)

// {

//     ptr[0] = 1000;

//     cout << ptr[0] << endl;

// }

// void PrintArr(int arr2[])

// {

//     int n = sizeof(arr2)/sizeof(int);

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr2[i] << " ";

//     }

//     cout << endl;

// }

// void PrintArr2(int arr2[], int n)

// {

//     // int n = sizeof(arr2)/sizeof(int);

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr2[i] << " ";

//     }

//     cout << endl;

// }

// int main()

// {

//     int a = 5;

//     int \*ptr = &a;

//     cout << ptr << endl;  // 0x61ff08

//     cout << \*ptr << endl; // 5 - Derefenrece operation - using pointer getting variables value

//     int arr[] = {2, 4, 6, 8, 10};

//     cout << \*arr << endl;       // 2 - can also got by usng cout<<arr[0]. Here using Aestric as its dererfenrece of array

//     cout << \*(arr + 1) << endl; // 4

//     cout << \*(arr + 2) << endl; // 6

//     func(arr);              // 1000 - jb fun decln me array pass kiya jaata he to yh by default  CALL BY REFERENCE use krta h

//     cout << arr[0] << endl; // 1000 - Passing array name is eq. to passing the pointer

//     func2(ptr);             // 1000

//     // so clearly, pointer and array are quivalent. in case of array

//     int arr2[] = {5, 10, 15, 20};

//     PrintArr(arr2);// w/o paasing the array size too in paramter we canb't print thte array. Because it consider as thee pointer size not the array size

// /\*

// 1\_IntroToArray.cpp: In function 'void PrintArr(int\*)':

// 1\_IntroToArray.cpp:150:24: warning: 'sizeof' on array function parameter 'arr2' will return size of 'int\*' [-Wsizeof-array-argument]

//      int n = sizeof(arr2)/sizeof(int);

//                         ^

// 1\_IntroToArray.cpp:147:24: note: declared here

//  void PrintArr(int arr2[])

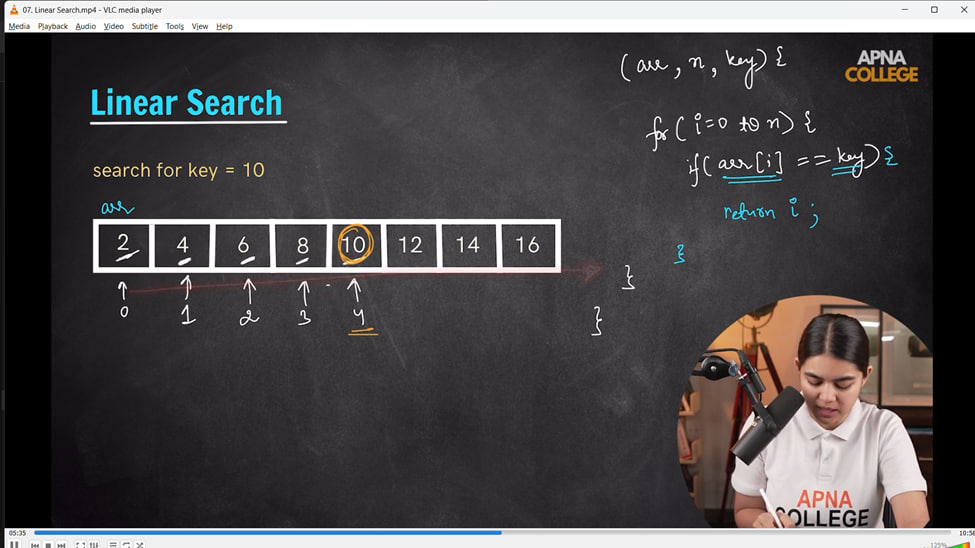
// \*/

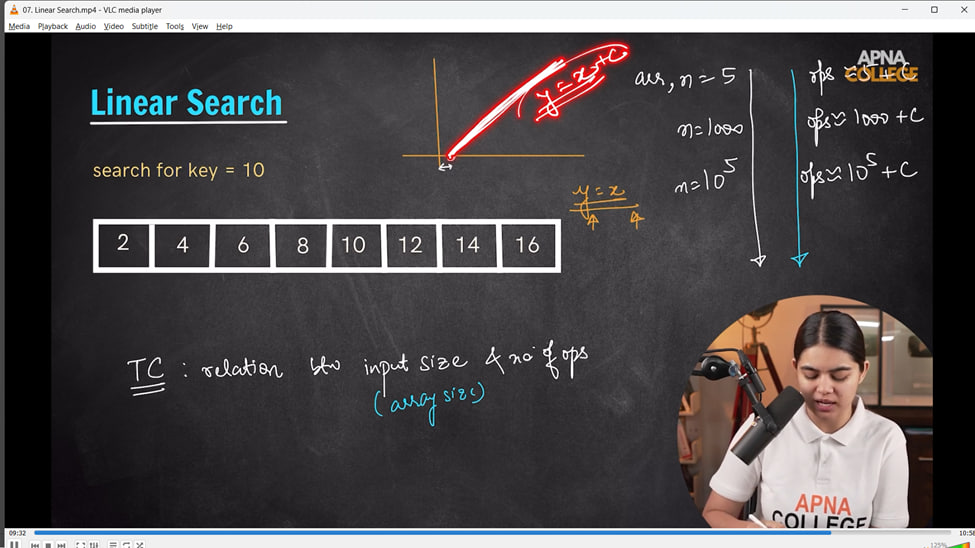
//     PrintArr2(arr2,4);//5 10 15 20 - so make sure array apne aap me ek pointer h which by defualt use sthe Call by reference propeerty. and funciton me array pass krte tym, hmesha size bhi as a paramter pass kre

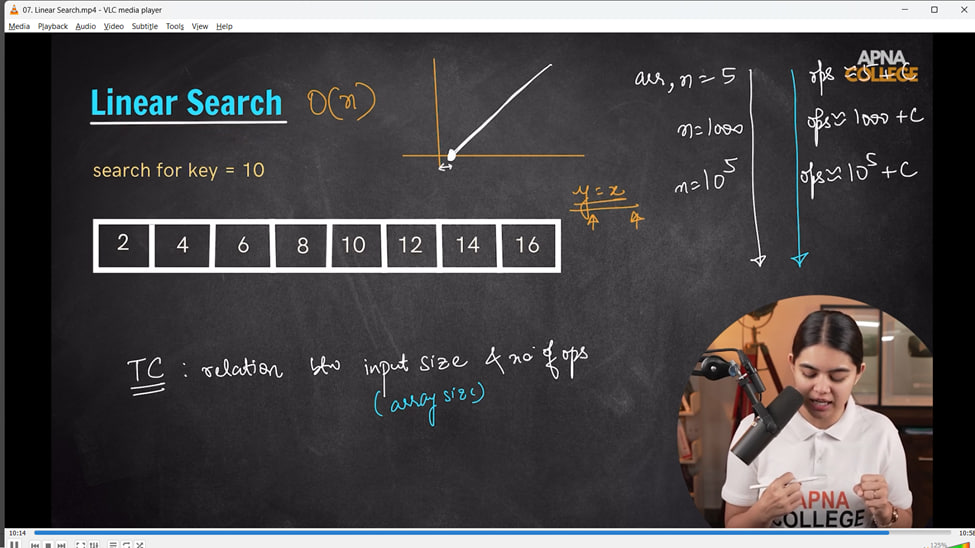
// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//4) Linear Search in array -**

****

****

****

// my method -

// int linearSeach()

// {

//     int n;

//     int key;

//     cout << "No of elements in Array - " << endl;

//     cin >> n;

//     int arr[n];

//     cout << "What are the array's eleemnts" << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the enetred aray is -" << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     cout << "Now mention the element you want to search for - " << endl;

//     cin >> key;

//     for (int i = 0; i < n; i++)

//     {

//         if (arr[i] == key)

//         {

//             cout << "So, key is - " << key << "  and found at " << i << endl;

//         }

//     }

// }

// int main()

// {

//     linearSeach();

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

// using proper method -

// int linearSearch(int arr[], int key)

// {

//     for (int i = 0; i < 8; i++)

//     {

//         if (arr[i] == key)

//         {

//             return i;

//         }

//     }

//     return -1;

// }

// int main()

// {

//     int arr[8] = {8, 16, 24, 32, 40, 48, 56, 64};

//     cout<<linearSearch(arr,15)<<endl;//-1

//     cout<<linearSearch(arr, 40)<<endl;//4

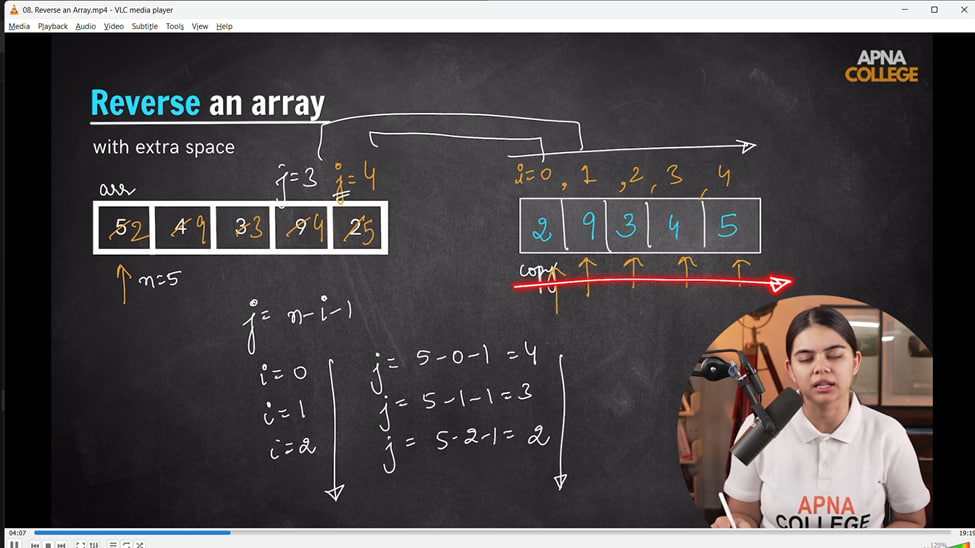
//     return 0;

// // T.C -is simply depends on no. of itersations which is depend on no. of elements - O(n)

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//5) Reverse an array**

****

**// 5.1) using extra space**

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int arr[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     int copyArr[n]; // an empty new array for getting all previous arra's value

//     for (int i = 0; i < n; i++)

//     {

//         int j = n - i - 1;

//         copyArr[i] = arr[j];

//     }

//     // now elements are stored in copyarr in reverse ordedr, so storing in the origina  again

//     for (int i = 0; i < n; i++)

//     {

//         arr[i] = copyArr[i];

//     }

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     return 0;

// /\*

// value of array size

// 5

// Write down the array elemenets -

// 56 89 12 23 78

// So, the inserted array is -

// 56 89 12 23 78

// 78 23 12 89 56

//  \*/

// }

//  Same approach using functions -

// void reverseArray(int arr[], int n)

// {

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int arr[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     int copyArr[n]; // an empty new array for getting all previous arra's value

//     for (int i = 0; i < n; i++)

//     {

//         int j = n - i - 1;

//         copyArr[i] = arr[j];

//     }

//     // now elements are stored in copyarr in reverse ordedr, so storing in the origina  again

//     for (int i = 0; i < n; i++)

//     {

//         arr[i] = copyArr[i];

//     }

//     reverseArray(arr, n);

// /\*

// 📌 - T. C - is same - O(n) - depends on iterations whichd epend on no. of elements

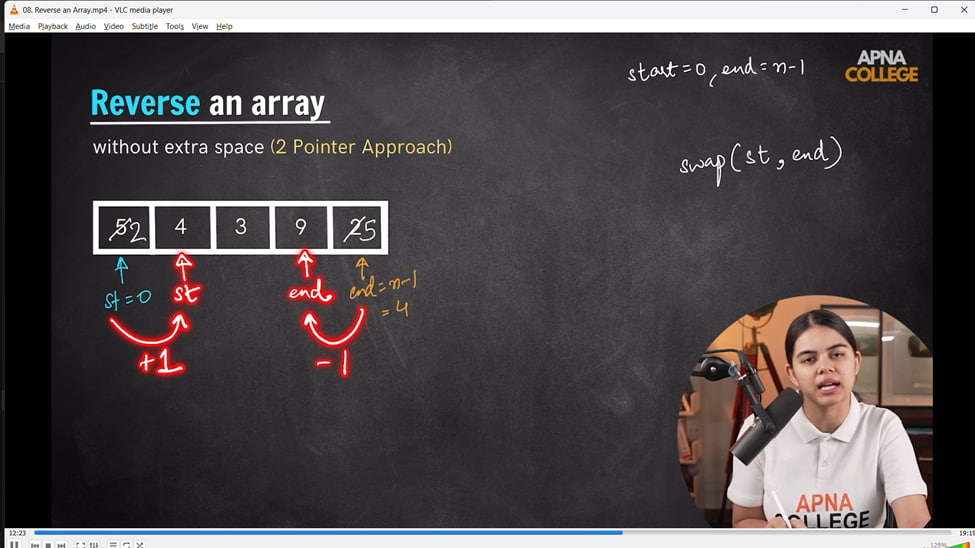
// S.C - O(n) for Original array, but due to creation another array which is extra space also O(n) - so S.C - O(n\*n) = O(n^2)

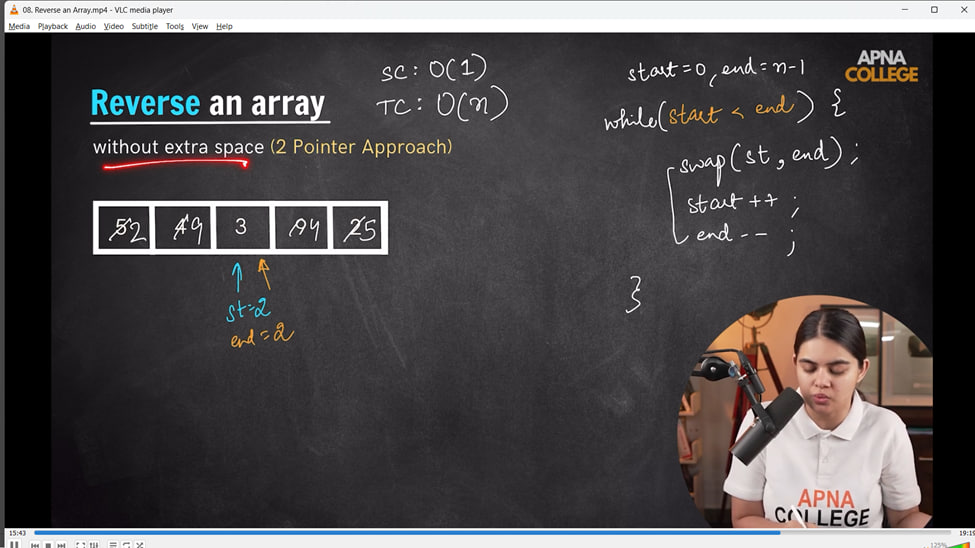
//  \*/

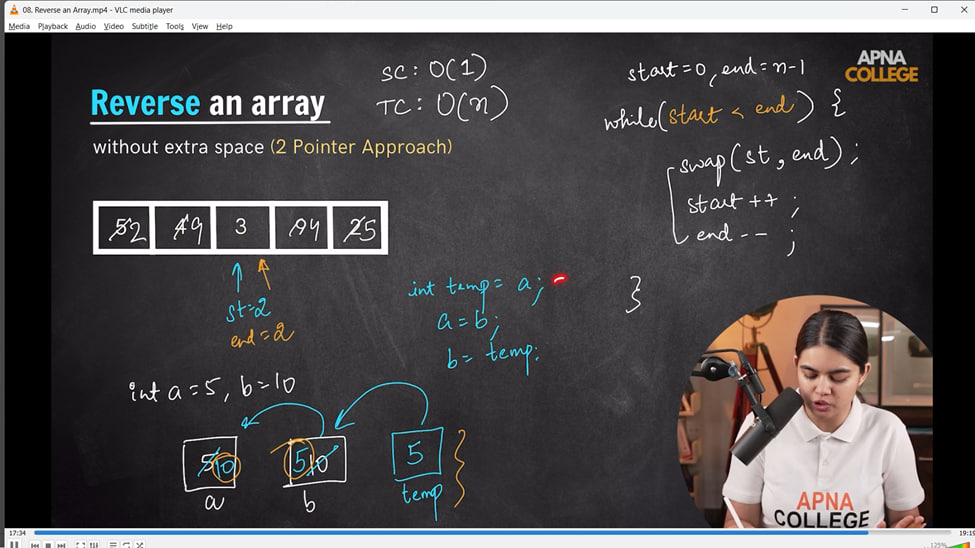
// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 5.2 - W/o using Extra Space1 - 2 POINTERS APPROACH**

****

****

****

// void printArray(int arr[], int n)

// {

//     cout << "Hence, the reversed array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int arr[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     // usingn 2 pointer approch -

//     int start = 0, end = n - 1;

//     while (start < end)

//     {

//         swap(arr[start], arr[end]);

//         /\* or by usign swap method

//         int temp = arr[start];

//         arr[start] = arr[end];

//         arr[end] = temp;

//         \*/

//         start++;

//         end--;

//     }

//     printArray(arr, n);

//     return 0;

//     /\*

//     value of array size

//     5

//     Write down the array elemenets -

//     89 56 45 23 12

//     So, the inserted array is -

//     89 56 45 23 12

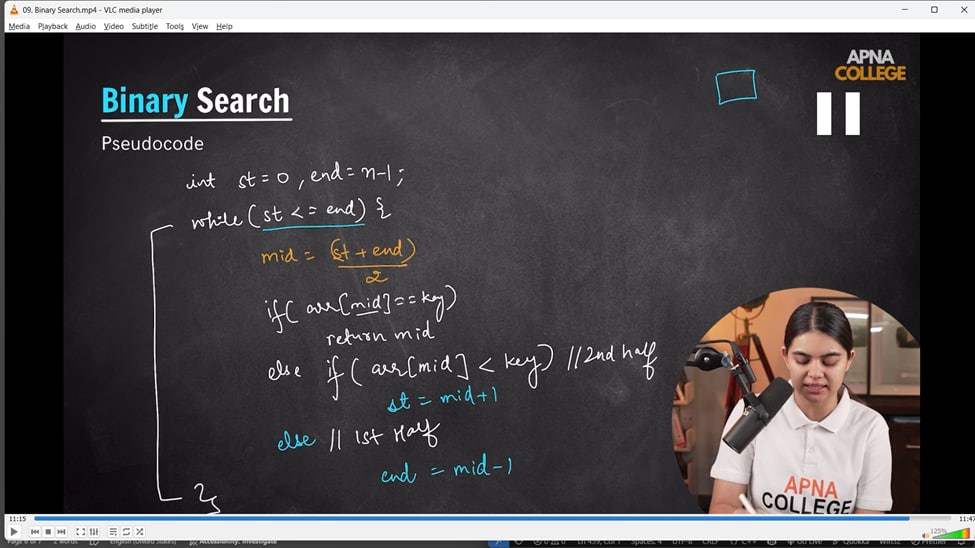
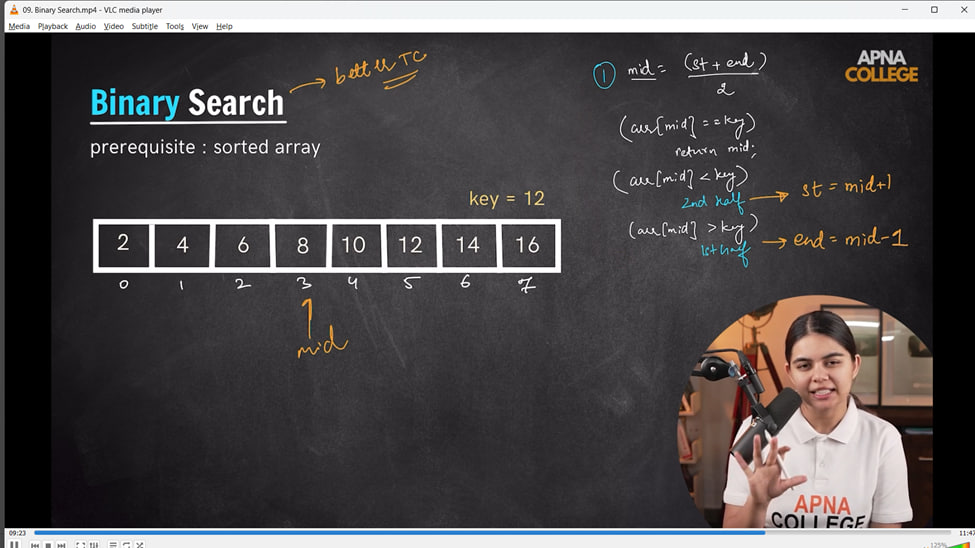
//     Hence, the reversed array is -

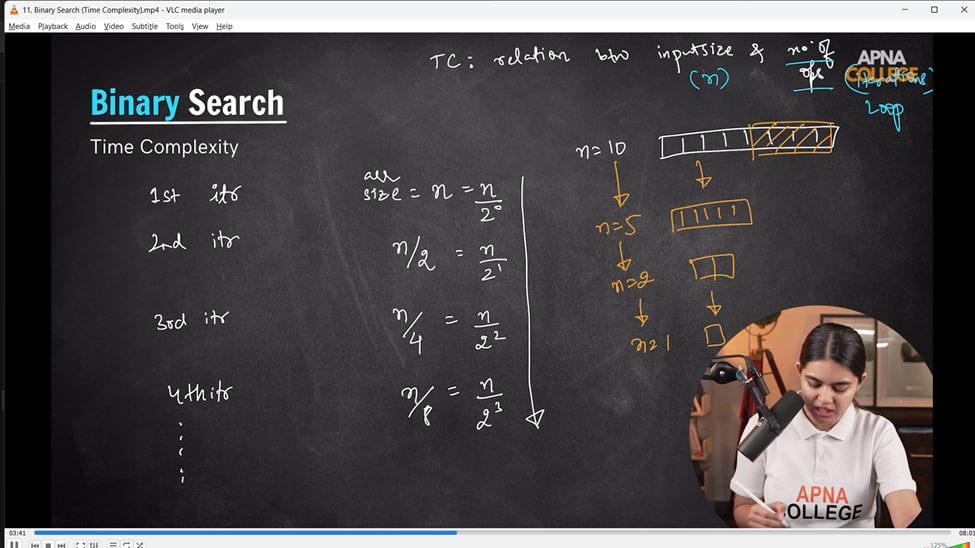
//     12 23 45 56 89

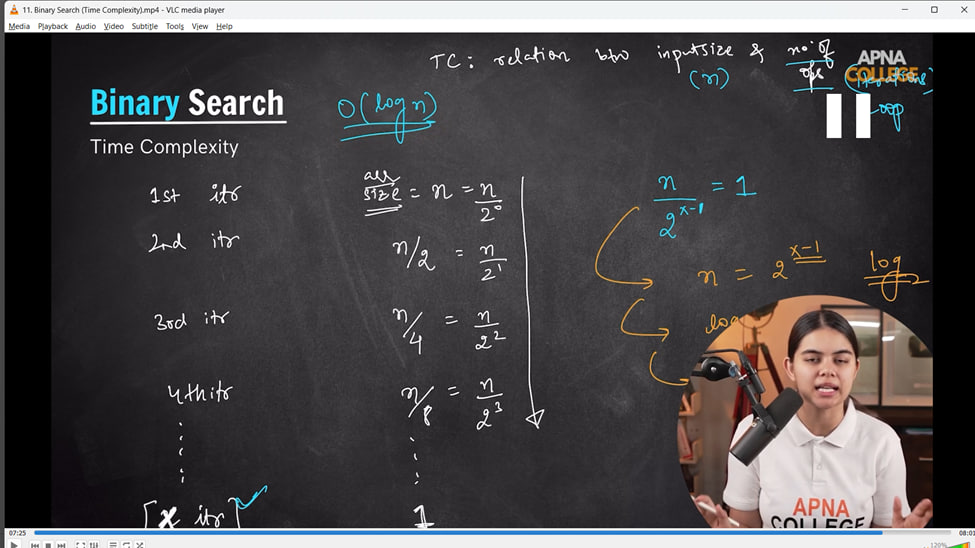
//     \*/

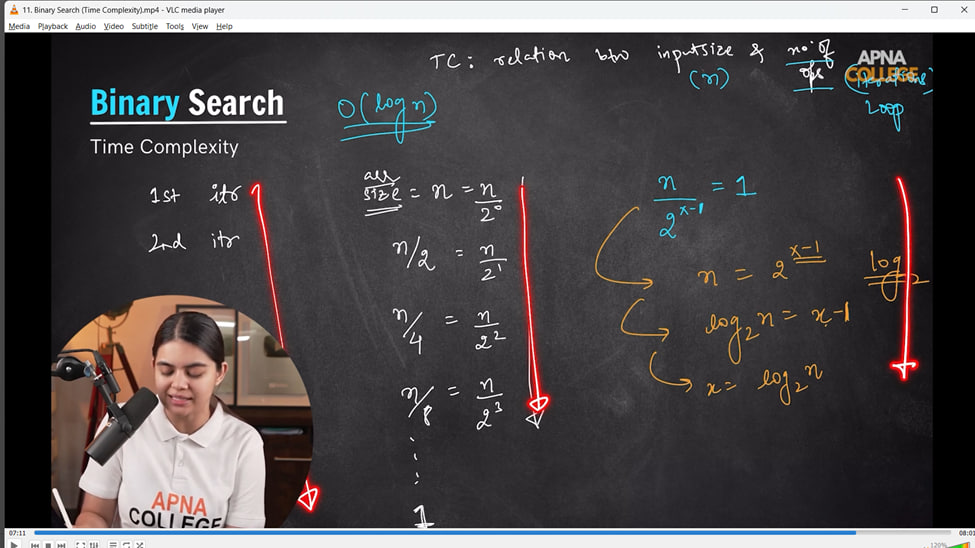
// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//6) Binary Search - It alywas for the Sorted Array**

****

****

****

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int arr[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     int key;

//     cout << "Now, enter the key you want to search for - " << endl;

//     cin >> key;

//     // For Binary Search Approach -

//     int start = 0, end = n - 1;

//     while (start <= end)

//     {

//         int mid = (start + end) / 2;

//         if (arr[mid] == key)

//         {

//             cout<<"Element is here and got at - "<<mid<<endl;

//             return 0;

//         }

//         else if (arr[mid] < key)

//         {

//             start = mid + 1;

//         }

//         else if (arr[mid] > key)

//         {

//             end = mid - 1;

//         }

//         cout << "Element not found" << endl;

//     }

//     return 0;

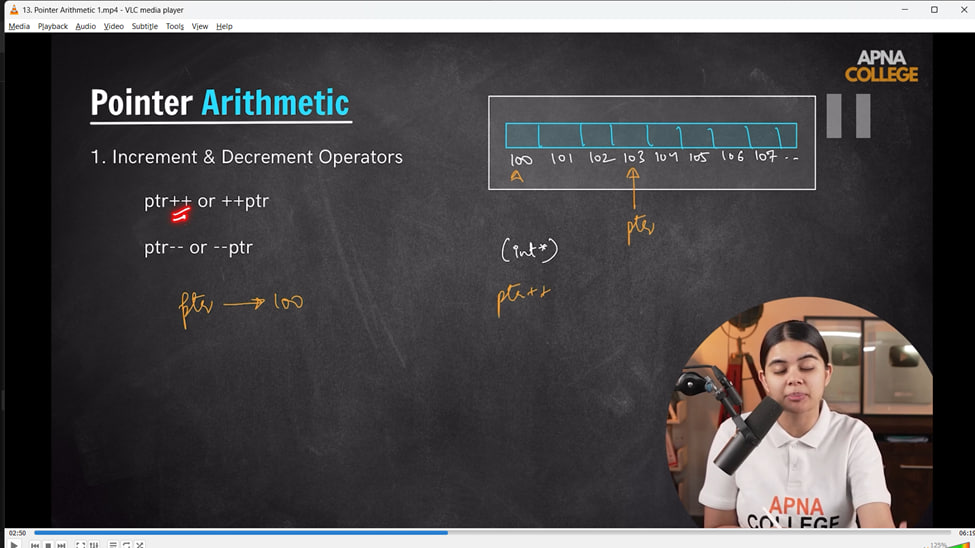
// T.c -O(log n)

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//7) ArrayPointer - Array is always treat as constant pointer, array pointer can;t be modifed as in another variable poinyer**

**//7.1. Pointer Arithmatic Approaches -**

****

// int main()

// {

// int x = 10;

// int \*ptr = &x;

// int y=25;

// ptr = &y;

// cout<<\*ptr<<"\n";//25 - in variable it is possible to update the pointer with another value and variable

// return 0;

// // int arr[5];

// // cout<<arr<<endl;

// // int y = 25;

// // arr = &y; // showing error as  - Expresion must be a modifiabble value

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

// //1. using increment decrement operator. In array using pointer on incrementing and decrementing it does the operations by the datatype size.

//     int a = 10;

//     int \*ptr = &a;

//     cout<<ptr<<endl;//0x61ff08

//     ptr++;

//     cout<<ptr<<endl;//0x61ff0c

//     ptr--;

//     cout<<ptr<<endl;// 0x61ff08

// /\*

// 0x61ff08 - address of var pointer a

// 0x61ff0c - on incrementing by ++ it'll increase by 4 bytes as int size is 4 bytes, so the new address is indcreased.

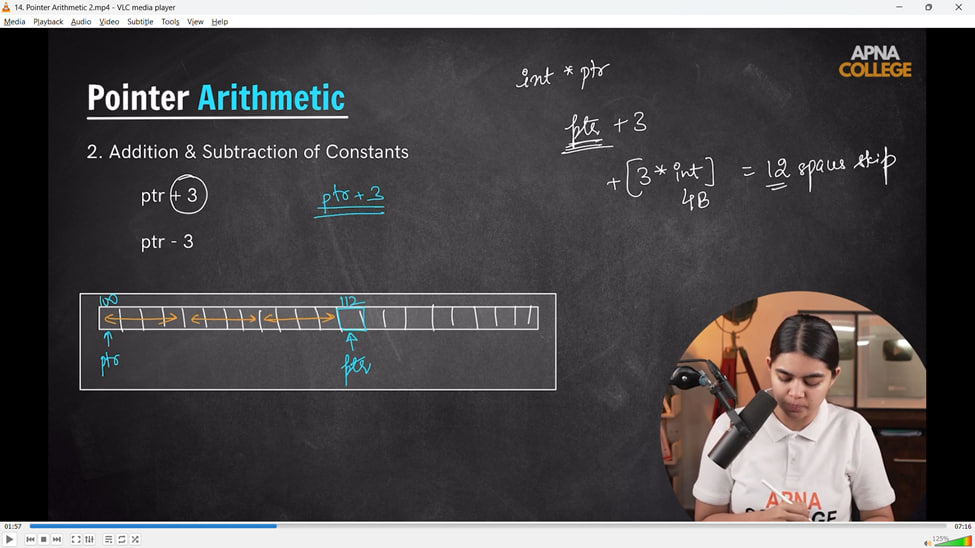
// In hexadecimal initially it was 08, but now it's 0c - which is 12

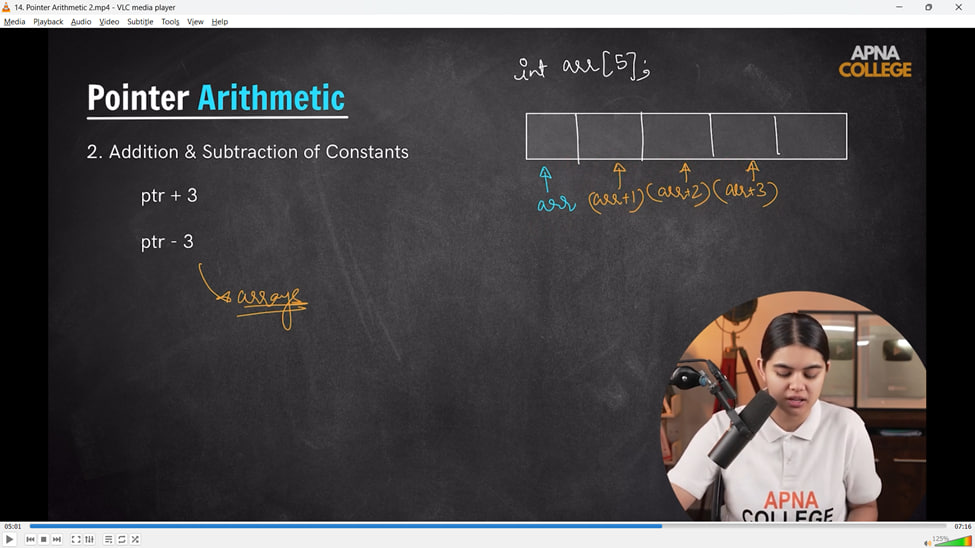
// 0x61ff08 - on doing the decrement operation, address again decreased by 4 bytes.

// \*/

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// // 7.2 - Addition & Substraction of Constants -**

****

****

// // 2. Addition & Substraction of Constants -

//     int a1 = 5;

//     int \*ptr1 = &a1;

//     cout<<ptr1<<endl;//0x61ff00 - here address is 0

//     cout<<(ptr1+3)<<endl;//0x61ff0c - here increased by3\*4bytes = 12 bytes which is c.

// // Pointer me addition ya sustr. krne pr int ke size ke terms me hota he. Yaha add 3 \* 4 byte as size of int = 12 bytes

//     cout<<(ptr1-3)<<endl;

// }

// interms of array by Arithmatic operation using Dereferenc eoperator -

// void printArr(int \*ptr, int n)

// {

//     for (int i = 0; i < 7; i++)

//     {

//         cout << \*(ptr + i) << endl;

//     }

// }

// int main()

// {

//     int arr[] = {2, 4, 5, 6, 10, 12, 17};

//     int n = sizeof(arr) / sizeof(int);

//     printArr(arr, n);

// }

// /\*

// 2

// 4

// 5

// 6

// 10

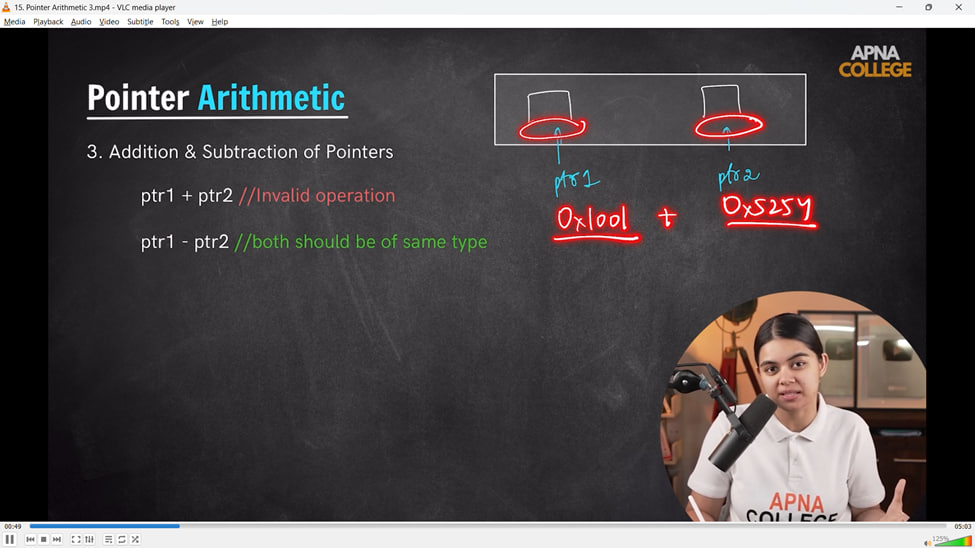
// 12

// 17

//  so here pritning all arrray values usig dereference operator \*/

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 7.3- Addition & Substraction of Pointers**

****

int main()

{

    int a = 5;

int \*ptr1 = &a;

int \*ptr2 = ptr1 + 3;

cout<<ptr1<<endl;//0x61ff04

cout<<ptr2<<endl;//0x61ff10

cout<<ptr2 - ptr1 <<endl;//3

// interms of array -

int arr[20] = {1,2,3,4,5,6};

int \*ptr3 = arr; // base address of array

int \*ptr4 = ptr3+3;// base add + 3bytes

cout<<\*ptr3<<endl;//1

cout<<\*ptr4<<endl;//4

**// 7.4 - Comparision Operators**

cout<<(ptr2>ptr1)<<endl;//1 - Yes : True

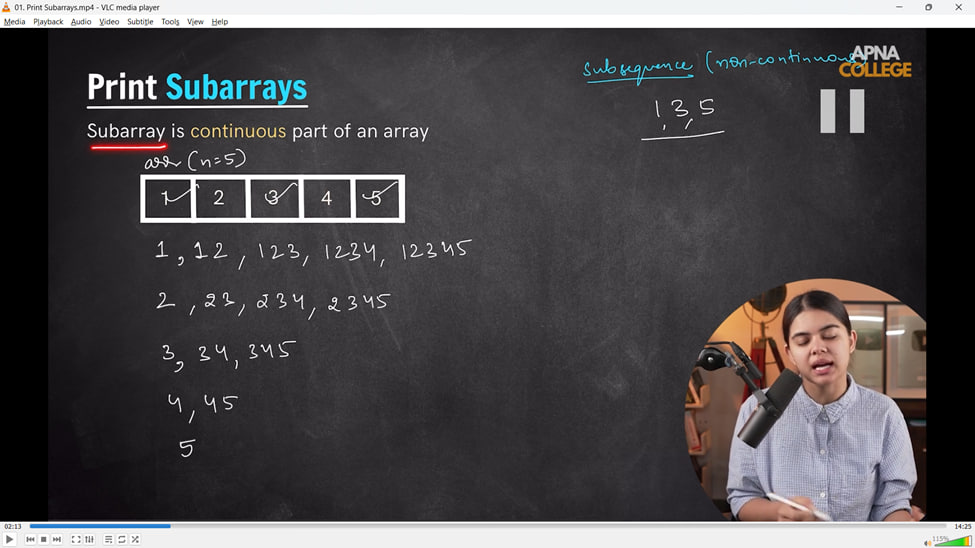
cout<<(ptr4<ptr3)<<endl;// 0 - NO : False

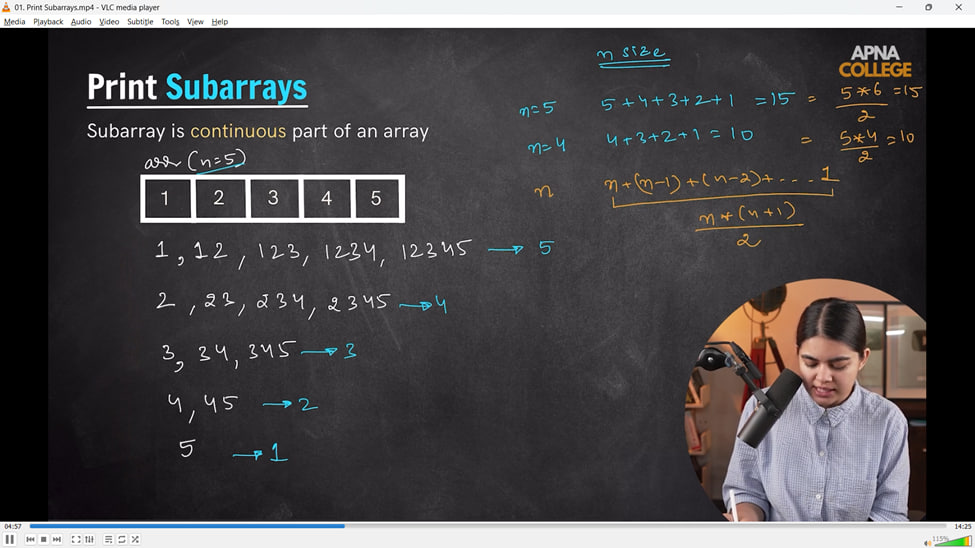
cout<<(ptr3 == arr)<<endl;//1 - YES ptr3 and arr are same pointing

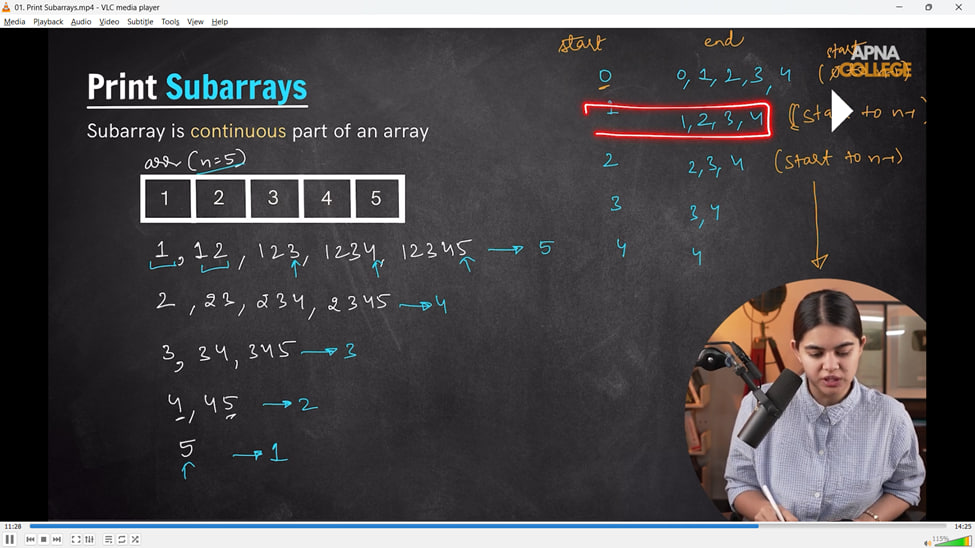
}

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//8) Subarrays -**







// void printSubArray(int Prices[], int n)

// {

//     cout << "So, pairwise subarrays are - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         for (int j = i; j < n; j++)

//         {

//             cout << "(" << Prices[i] << "," << Prices[j] << ")";

//         }

//         cout << endl;

//     }

// }

// void SubArray2(int Prices[], int n)

// {

//     cout << "and the subarrays are - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         for (int j = i; j < n; j++)

//         {

//             cout << Prices[j] << " ";

//         }

//         cout << endl;

//     }

// }

// // OR

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int Prices[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> Prices[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << Prices[i] << " ";

//     }

//     cout << endl;

//     printSubArray(Prices, n);

//     SubArray2(Prices, n);

//     /\*

//     value of array size

// 5

// Write down the array elemenets -

// 12 23 56 45 89

// So, the inserted array is -

// 12 23 56 45 89

// So, pairwise subarrays are -

// (12,12)(12,23)(12,56)(12,45)(12,89)

// (23,23)(23,56)(23,45)(23,89)

// (56,56)(56,45)(56,89)

// (45,45)(45,89)

// (89,89)

// and the subarrays are -

// 12 23 56 45 89

// 23 56 45 89

// 56 45 89

// 45 89

// 89

// T.C -  O(n^3)

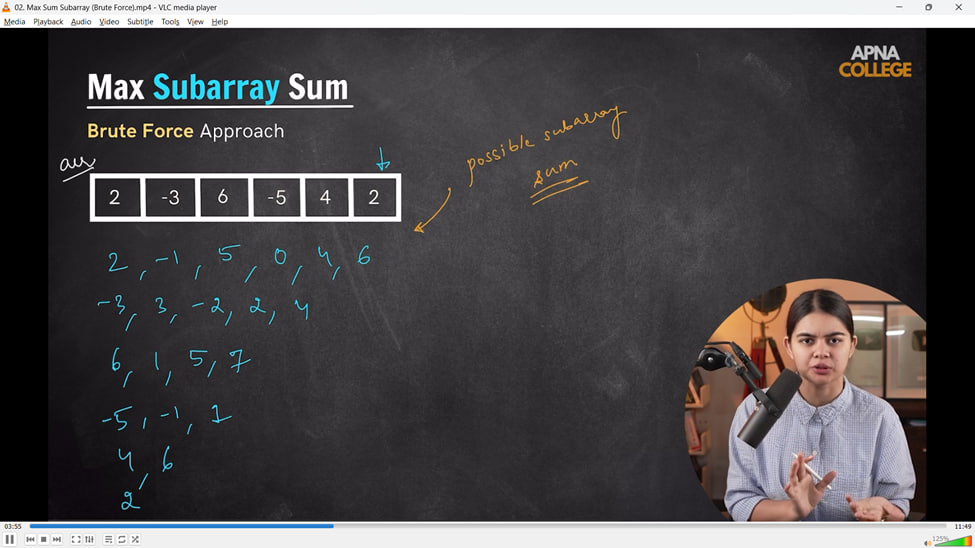
// \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 9) Max Subarray Sum -**

**// 9.1 - Clearly Brute Force Approach -**



// void maxSubArraySum(int Prices[], int n)

// {

//     int maxSum = INT\_MIN;

//     cout << "Noe for maximum subarray sum - " << endl;

//     // in form of subarray the sum

//     for (int i = 0; i < n; i++)

//     {

//         for (int j = i; j < n; j++)

//         {

//             int currSum = 0;

//             for (int k = i; k <= j; k++)

//             {

//                 currSum += Prices[k]; // for prnitng the current sum  of every subarray possible from every element

//             }

//             cout << currSum << ",";

//             maxSum = max(maxSum, currSum);

//         }

//         cout << endl;

//     }

//     cout << "Maximum Subarray sum is - " << maxSum << endl;

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int Prices[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> Prices[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << Prices[i] << " ";

//     }

//     cout << endl;

//     maxSubArraySum(Prices, n);

//     /\*

//     value of array size

//     6

//     Write down the array elemenets -

//     2 -3 6 -5 4 2

//     So, the inserted array is -

//     2 -3 6 -5 4 2

//     Noe for maximum subarray sum -

//     2,-1,5,0,4,6,

//     -3,3,-2,2,4,

//     6,1,5,7,

//     -5,-1,1,

//     4,6,

//     2,

//     Maximum Subarray sum is - 7

//     T.C - O(n^3)

//      \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 9.2 - SLightly Optimizede approch - for decreasing the time complexity**

// void maxSubArraySum2(int Prices[], int n)

// {

//     int maxSUm = INT\_MIN;

//     for (int i = 0; i < n; i++)

//     {

//         int currSum = 0;

//         for (int j = i; j < n; j++)

//         {

//             currSum += Prices[j];

//             maxSUm = max(maxSUm, currSum);

//         }

//         cout << endl;

//     }

//     cout << "Maximum Subarray sum is - " << maxSUm << endl;

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int Prices[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> Prices[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << Prices[i] << " ";

//     }

//     cout << endl;

//     // maxSubArraySum(Prices, n);

//     maxSubArraySum2(Prices, n);

//     /\*

//     value of array size

//     6

//     Write down the array elemenets -

//     2 -3 6 -5 4 2

//     So, the inserted array is -

//     2 -3 6 -5 4 2

//     Maximum Subarray sum is - 7

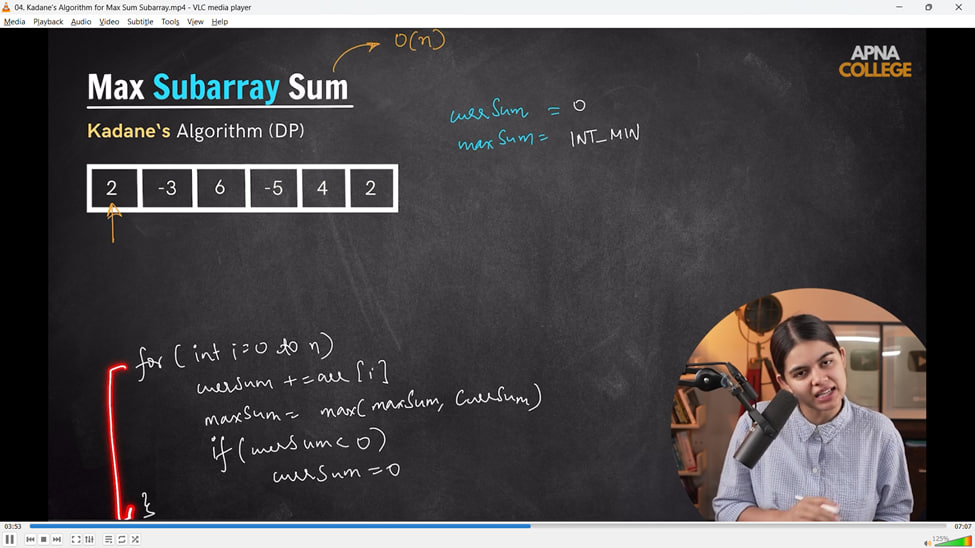
//         T.C - O(n^2)

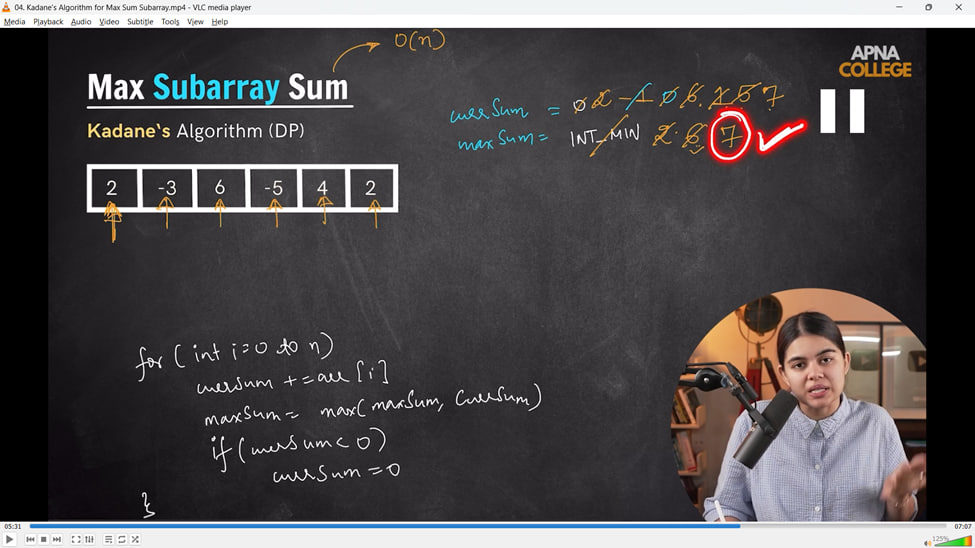
//      \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 9.3 - Using Most Optimized Approach - Kadane's Algorithms. For very less TIme Complexity**

****



// void maxSubArraySum3(int Prices[], int n)

// {

//     int maxSUm = INT\_MIN;

//     int currSum = 0;

//     for (int i = 0; i < n; i++)

//     {

//         currSum += Prices[i];

//         maxSUm = max(maxSUm, currSum);

//         if (currSum < 0)

//         {

//             currSum = 0;

//         }

//     }

//     cout << "Maximum Subarray sum is - " << maxSUm << endl;

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int Prices[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> Prices[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << Prices[i] << " ";

//     }

//     cout << endl;

//     // maxSubArraySum(Prices, n);

//     maxSubArraySum3(Prices, n);

//     /\*

//     value of array size

//     6

//     Write down the array elemenets -

//     2 -3 6 -5 4 2

//     So, the inserted array is -

//     2 -3 6 -5 4 2

//     Maximum Subarray sum is - 7

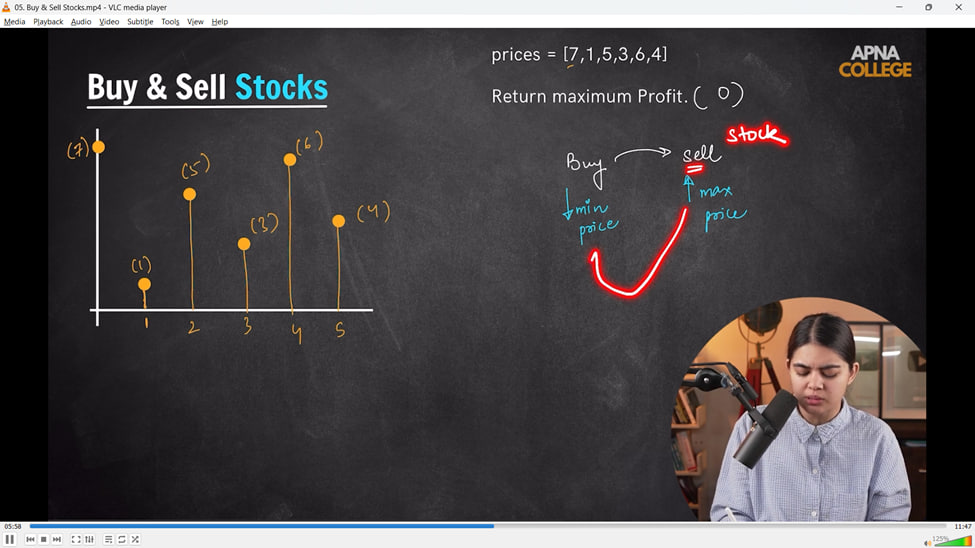
//     T.C - O(n)

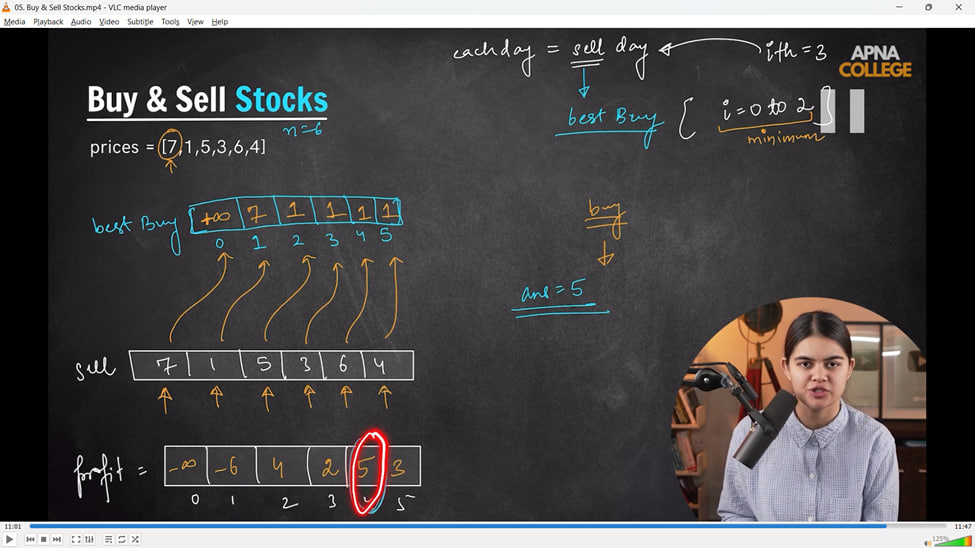
//      \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//10) bUY & sELL sTOCK pROBLEM-**





// void maxprofit(int Prices[], int n)

// {

//     int bestBuy[100000]; // as per the given max limits

//     bestBuy[0] = INT\_MAX;

//     for (int i = 0; i < n; i++)

//     {

//         bestBuy[i] = min(bestBuy[i - 1], Prices[i - 1]);

//     }

//     int maxProfit = 0;

//     for (int i = 0; i < n; i++)

//     {

//         int currProfit = Prices[i] - bestBuy[i];

//         maxProfit = max(maxProfit, currProfit);

//     }

//     cout << "MaxProfit is - " << maxProfit << endl;

// }

// int main()

// {

//     int n;

//     cout << "what is the vale of n - " << endl;

//     cin >> n;

//     int Prices[n];

//     cout << "Mentin the array elementa" << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> Prices[i];

//     }

//     cout << "So, the entered elements of array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << Prices[i] << " ";

//     }

//     cout << endl;

//     maxprofit(Prices, n);

//     /\*

//     what is the vale of n -

//     5

//     Mentin the array elementa

//     9 5 15 8 2

//     So, the entered elements of array is -

//     9 5 15 8 2

//     MaxProfit is - 15

//     what is the vale of n -

//     6

//     Mentin the array elementa

//     7 1 5 3 6 4

//     So, the entered elements of array is -

//     7 1 5 3 6 4

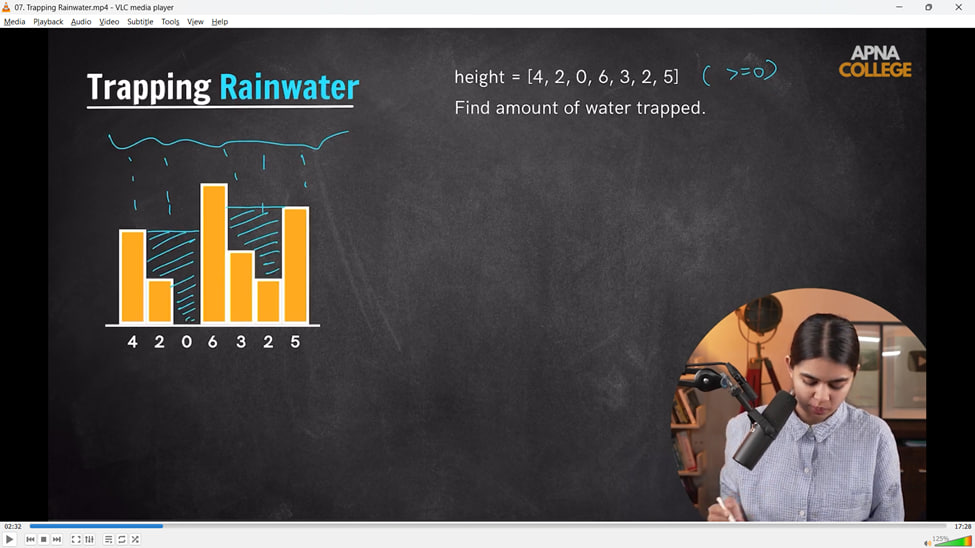
//     MaxProfit is - 7

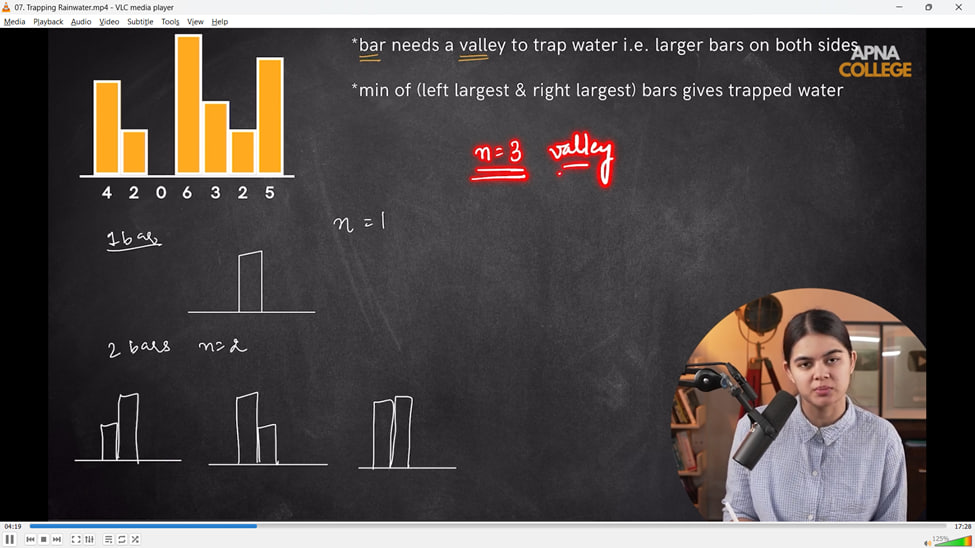
//      \*/

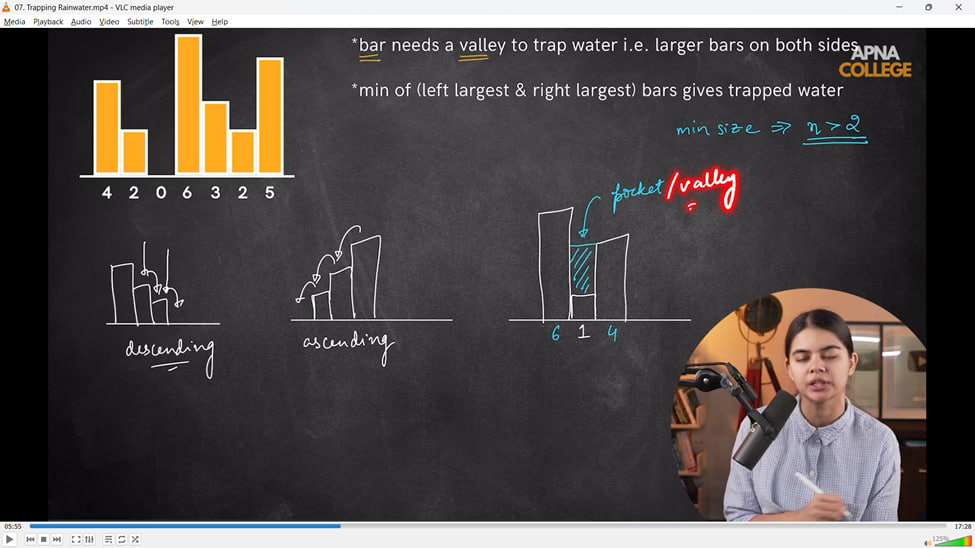
// }

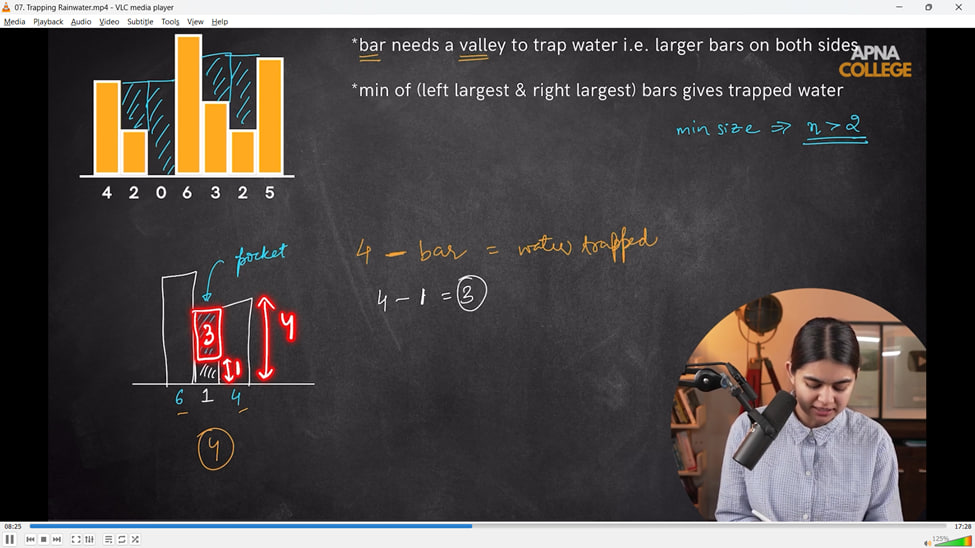
// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

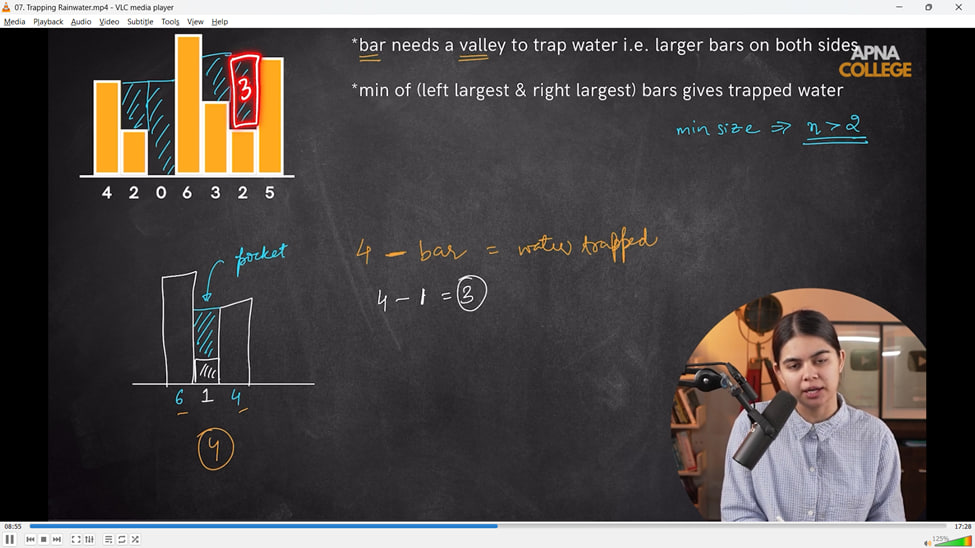
**//11) Trapping Rainwater Problem -**

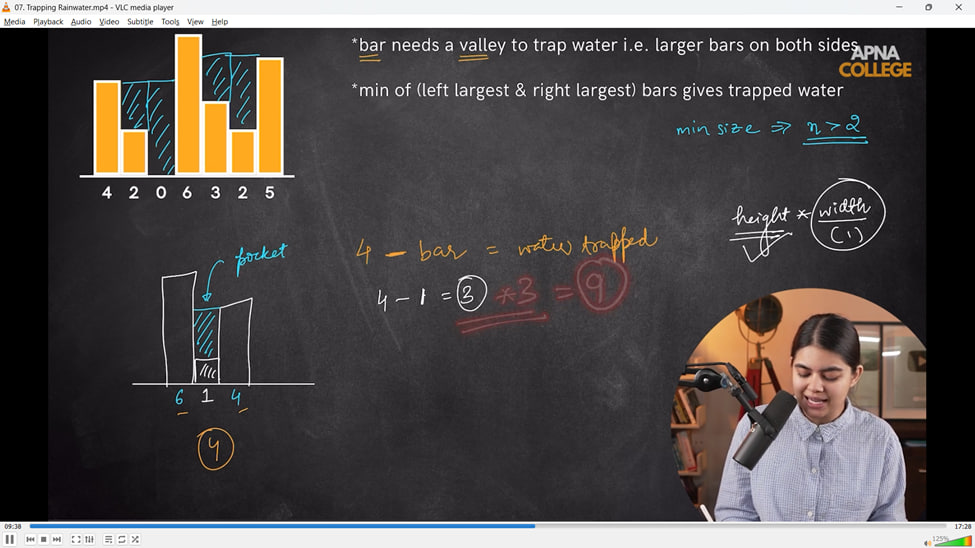


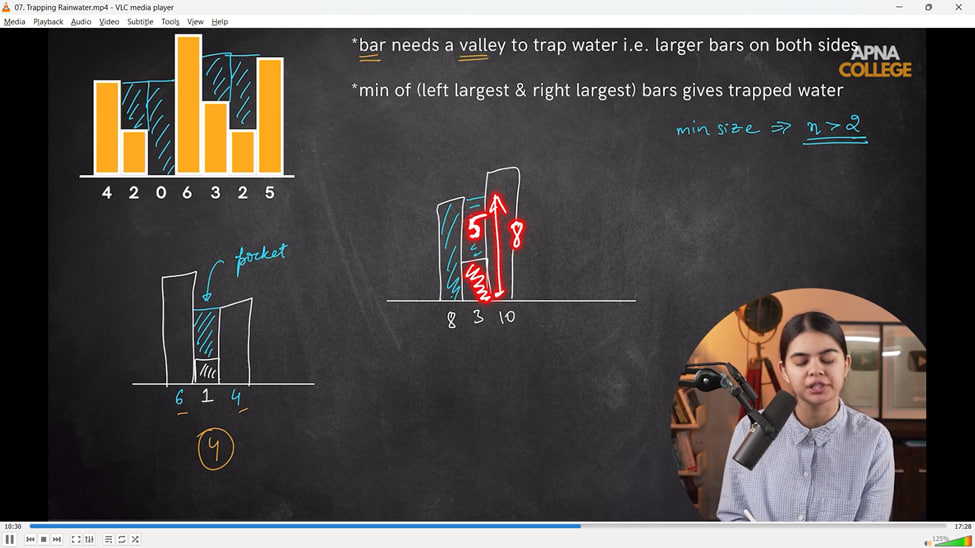


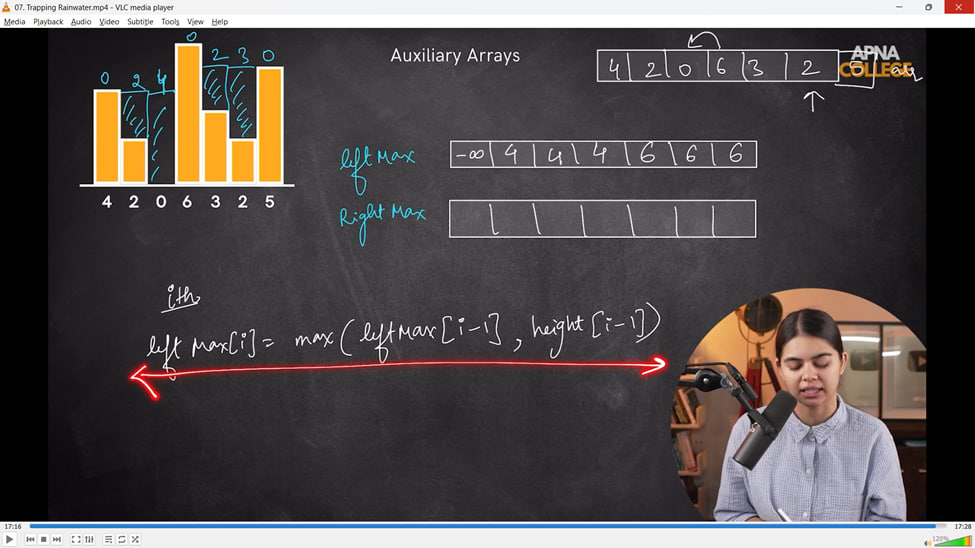




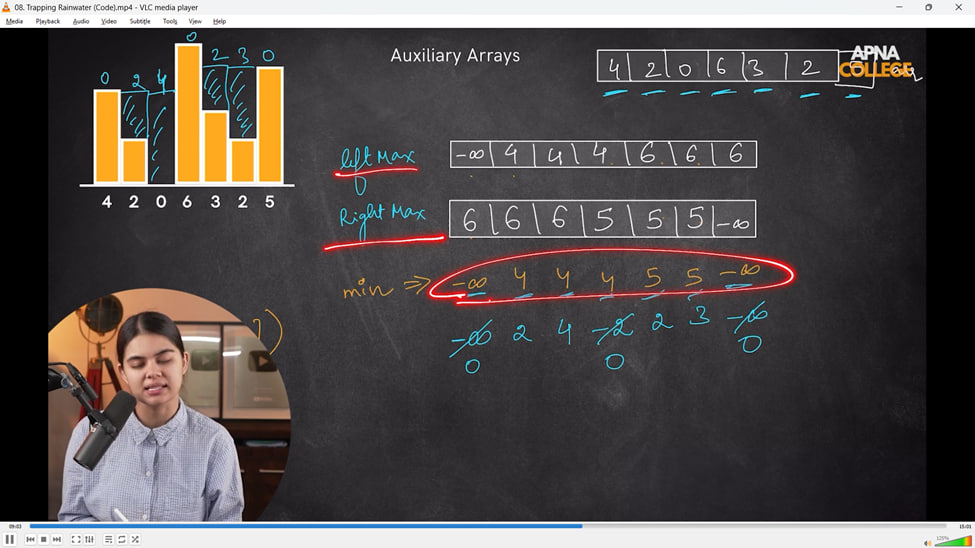


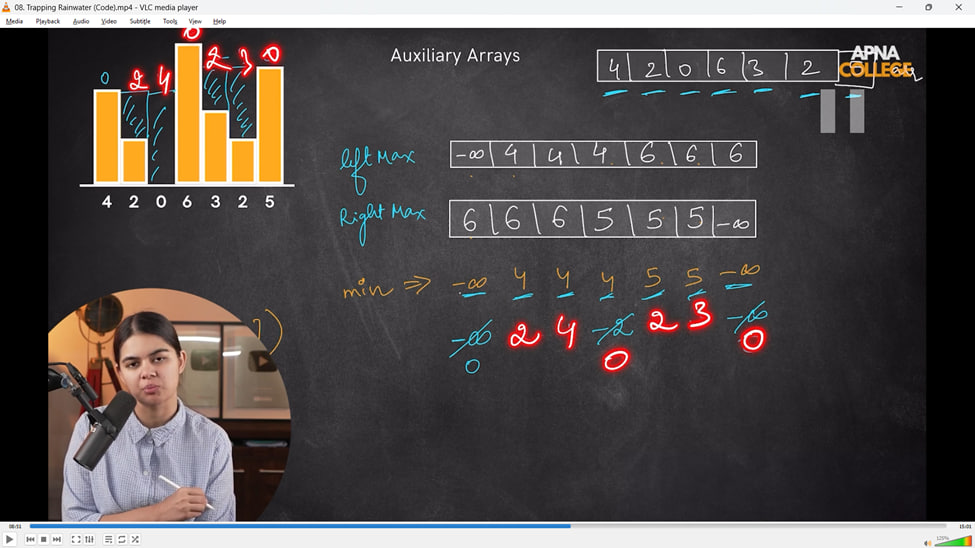


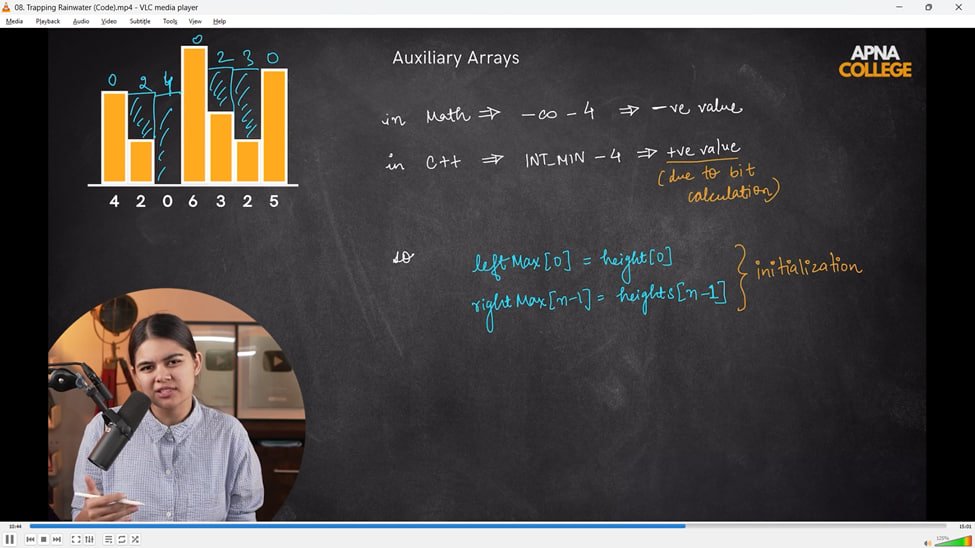












int Trap(int heights[], int n)

{

    int leftMax[20000], rightMax[20000];

    leftMax[0] = heights[0];

    rightMax[n - 1] = heights[n - 1];

    // For getting the left max side for every bars

    cout << "So, the Left Valley max values for each bar is - " << endl;

    cout << leftMax[0] << ",";

    for (int i = 1; i < n; i++)

    {

        leftMax[i] = max(leftMax[i - 1], heights[i - 1]);

        cout << leftMax[i] << ",";

    }

    // Similarly For getting the right max side for every bars

    cout << "again the right valley max values for each bar is - " << endl;

    for (int i = n - 2; i >= 0; i--)

    {

        rightMax[i] = max(rightMax[i + 1], heights[i + 1]);

        cout << rightMax[i] << ",";

    }

    cout << rightMax[n - 1] << ",";

    cout << endl;

    int WaterTrappedd = 0;

    for (int i = 0; i < n; i++)

    {

        int currWater = min(leftMax[i], rightMax[i]) - heights[i];

        if (currWater > 0)

        {

            WaterTrappedd += currWater;

        }

    }

    cout << "So, the Water Trapp is - " << WaterTrappedd << endl;

    return WaterTrappedd;

}

int main()

{

    int n;

    cout << "value of array size" << endl;

    cin >> n;

    int heights[n];

    cout << "Write down the array elemenets - " << endl;

    for (int i = 0; i < n; i++)

    {

        cin >> heights[i];

    }

    cout << "So, the inserted array is - " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << heights[i] << " ";

    }

    cout << endl;

    Trap(heights, n);

    /\*

    value of array size

    7

    Write down the array elemenets -

    4  2 0 6 3 2 5

    So, the inserted array is -

    4 2 0 6 3 2 5

    So, the Left Valley max values for each bar is -

    4,4,4,4,6,6,6,again the right valley max values for each bar is -

    5,5,5,6,6,6,5,

    So, the Water Trapp is - 11

    value of array size

    5

    Write down the array elemenets -

    5 4 3 2 1

    So, the inserted array is -

    5 4 3 2 1

    So, the Left Valley max values for each bar is -

    5,5,5,5,5,again the right valley max values for each bar is -

    1,2,3,4,1,

    So, the Water Trapp is - 0

    T.C - O(n)

    \*/

}

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_