Advanced C++ Pointer Concepts - Hinglish Notes

1. Fundamentals of Pointers (Pointer kya hota hai?)

Pointer ek aisa variable hota hai jo kisi doosre variable ke address ko store karta hai.

```
int x = 10;

int *ptr = &x;  // ptr stores address of x
```

```
* => Dereference operator (value nikaalne ke liye)

& => Address-of operator (address lene ke liye)
```

```
cout << "Address of x: " << &x;

cout << "Value at ptr: " << *ptr;
```

2. New and Delete Operators

Dynamic memory allocation ke liye use hote hain.

```
int *p = new int;

*p = 50;

delete p;
```

Array ke liye:

```
int *arr = new int[5];

delete[] arr;
```

3. Pointer Declaration & Initialization

```
int *ptr1;

int x = 5;

int *ptr2 = &x;
```

Wild Pointer:

```
int *ptr;
```

```cpp
*ptr = 10; // Undefined behavior
```

4. Operations on Pointers

```cpp
int a = 10;

int *p = &a;


p++;  // address next int location pe chala gaya


Valid operations:

- Pointer arithmetic: ++, --, +, -

- Comparison: ==, !=, <, >

- Dereferencing: *p
```

5. Passing Pointers to Functions

```cpp
void update(int *x) {

    *x = *x + 10;

}


int main() {

    int a = 5;

    update(&a);

    cout << a; // 15

}
```

6. Passing an Entire Array to Function Using Pointer

```cpp
void display(int *arr, int size) {

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

        cout << arr[i] << " ";

    }

}


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

```cpp
display(arr, 5);
```

7. Pointers and Two-Dimensional Arrays

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

int *p = &arr[0][0];


for(int i = 0; i < 2; i++) {

    for(int j = 0; j < 3; j++) {

        cout << *(p + i*3 + j) << " ";

    }

}
```

8. Array of Pointers

```cpp
const char *names[] = {"Ram", "Shyam", "Geeta"};


for(int i = 0; i < 3; i++) {

    cout << names[i] << endl;

}


int a = 10, b = 20;

int *arr[2] = {&a, &b};
```

9. Passing Functions to Other Functions

```cpp
int add(int a, int b) {

    return a + b;

}


int compute(int x, int y, int (*func)(int, int)) {

    return func(x, y);

}


int main() {
```

```cpp
    cout << compute(5, 3, add); // Output: 8
}
```

10. Pointers to Structures

```cpp
struct Distance {
    int feet;
    float inch;
};


Distance d;
Distance *ptr = &d;
ptr->feet = 5;
ptr->inch = 6.2;


cout << "Feet: " << ptr->feet << ", Inch: " << ptr->inch;
```

11. This Pointer

```cpp
class MyClass {
    int data;
public:
    MyClass(int data) {
        this->data = data;
    }
    void show() {
        cout << this->data;
    }
};


MyClass& set(int x) {
    this->data = x;
    return *this;
}
```