Pointers Concepts

**Basic Pointer Usage**

Concepts: Address-of operator &, dereferencing \*, pointer declaration.

**Explanation:**

* int x = 10; declares an integer variable x with value 10.
* int\* ptr = &x; creates a pointer ptr that stores the address of x.
* \*ptr dereferences the pointer, accessing the value stored at that address — which is 10.

#include <iostream>

using namespace std;

int main() {

int x = 10;

int\* ptr = &x; // ptr stores the address of x

cout << "Value of x: " << x << endl;

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

cout << "Value stored in ptr: " << ptr << endl;

cout << "Value pointed to by ptr: " << \*ptr << endl;

return 0;

}

**Changing Value Using Pointer**

*Concepts*: Pointer manipulation, indirect value change.

### Explanation:

* p points to a.
* \*p = 20; changes the value at the address p points to — which is a.
* So a becomes 20.

#include <iostream>

using namespace std;

int main() {

int a = 5;

int\* p = &a;

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

\*p = 20; // Changing value of a through pointer

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

return 0;

}

Pointer to Pointer

Concepts: Pointer to pointer (int\*\*), multi-level dereferencing.

### Explanation:

#include <iostream>

using namespace std;

int main() {

int num = 100;

int\* p = &num;

int\*\* pp = &p;

cout << "Value of num: " << num << endl;

cout << "Value via \*p: " << \*p << endl;

cout << "Value via \*\*pp: " << \*\*pp << endl;

return 0;

}

Function with Pointer Parameter

Concepts: Passing pointer to function, modifying original variable.

#include <iostream>

using namespace std;

void updateValue(int\* ptr) {

\*ptr = 50; // Modify value at the address

}

int main() {

int x = 10;

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

updateValue(&x); // Pass address of x

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

return 0;

}

Array and Pointer Relationship

*Concepts*: Pointer arithmetic, array traversal using pointers.

#include <iostream>

using namespace std;

int main() {

int arr[3] = {10, 20, 30};

int\* p = arr; // Points to first element

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

cout << "Element " << i << ": " << \*(p + i) << endl;

}

return 0;

}