CodInClub

Powered By BridgeLabz

Understanding Pointers in C++

Topics Covered:

- What are pointers?
- Accessing the address of a variable
- Declaring and initializing pointers
- Accessing a variable through its pointer
- Practice coding problems from easy to hard

What is a Pointer?

Imagine that a variable is a box with something inside.
A pointer is not the box, but a sticky note that tells you where the box is kept.

In C++, a pointer is a variable that stores the memory address of another variable.

Example:

```
int age = 18;
int* ptr = &age;  // ptr stores the address of age
```

- `int* ptr` → This says: "Let's make a pointer that can store the address of an integer."
- `&age` → This means: "Give me the address of age."

Accessing the Address of a Variable

Every variable is stored somewhere in your computer's memory.
To find out where, use the `&` symbol (called the "address-of" operator).

Example:

```
int score = 95;
cout << "Address of score is: " << &score << endl;
```

`&score` gives us the memory location of the variable `score`.

Declaring and Initializing a Pointer

To declare a pointer:

```
<type>* pointer_name;
```

To initialize a pointer:

```
pointer_name = &variable_name;
```

Example:

```
int num = 5;
int* ptr = &num;
```

👉 *`ptr` is now pointing to `num`*

Accessing Value Using a Pointer (Dereferencing)

To get the value from a location a pointer is holding, use the `*` operator.

Example:

```
int marks = 90;
int* p = &marks;

cout << *p << endl;  // prints 90
```

Here:

- `p` contains the address of `marks`
- `*p` means: “Go to this address, and get the value stored there”

CodInClub

Powered By BridgeLabz

Why Use Pointers?

Pointers are super useful. They help when:

Use Case	Why They're Useful
🚛 Passing data to functions	You can pass big data efficiently
🗑️ Managing memory manually	You decide when to use/free memory
🧵 Building advanced structures	Needed for linked lists, trees
✨ Changing things directly	Can modify variables from functions

Code Examples

Example 1: Basic Pointer Usage

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

int main() {
    int number = 10;
    int* ptr = &number;

    cout << "Value of number: " << number << endl;
    cout << "Address of number: " << &number << endl;
    cout << "Pointer holds: " << ptr << endl;
    cout << "Value through pointer: " << *ptr << endl;

    return 0;
}
```

Example 2: Modify Value Using Pointer

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

void addOne(int* p) {
    (*p)++;
}

int main() {
    int value = 5;
    addOne(&value);
    cout << "After function call: " << value << endl;
```

```
    return 0;
}
```

Example 3: Swapping Two Numbers Using Pointers

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

void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x = 2, y = 3;
    swap(&x, &y);
    cout << "x: " << x << ", y: " << y << endl;
    return 0;
}
```

Practice Problems (4 Total)

Problem 1: (Easy)

Title: Show Address and Value
Task:

- Declare an `int` variable.
- Print its value and its address.

Sample Output:

```
Value: 50
```

```
Address: 0x61ff08
```

Problem 2: (Easy)

Title: Access Float Using Pointer
Task:

- Create a `float` variable and a pointer to it.
- Use the pointer to print its value.

Sample Output:

```
Value using pointer: 12.5
```

Problem 3: (Medium)

Title: Swap Two Numbers
Task:

- Create a function to swap two integers using pointers.
- Call the function and show the values before and after.

Sample Output:

```
Before: a = 5, b = 8
After: a = 8, b = 5
```

Problem 4: (Hard)

Title: Print Array Using Pointer
Task:

- Declare an array of integers.
- Use a pointer to print each element using pointer arithmetic (not indexing).

Sample Code Hint:

```
int arr[5] = {10, 20, 30, 40, 50};
int* p = arr;
```

```
for (int i = 0; i < 5; i++) {
    cout << *(p + i) << " ";
}
```

Expected Output:
`10 20 30 40 50`

Summary Table of Practice Problems

#	Problem Title	Level	What You Learn
1	Show Address and Value	Easy	Using `&` to get address
2	Access Float Via Pointer	Easy	Using `*` to get value from address
3	Swap Two Numbers	Medium	Modifying values via pointers in functions
4	Print Array with Pointer	Hard	Mastering pointer arithmetic

Final Tips

- Always initialize your pointers (or set them to `nullptr`).
- Never dereference a pointer until you're sure it's valid.
- Use `*` to get value, use `&` to get address.