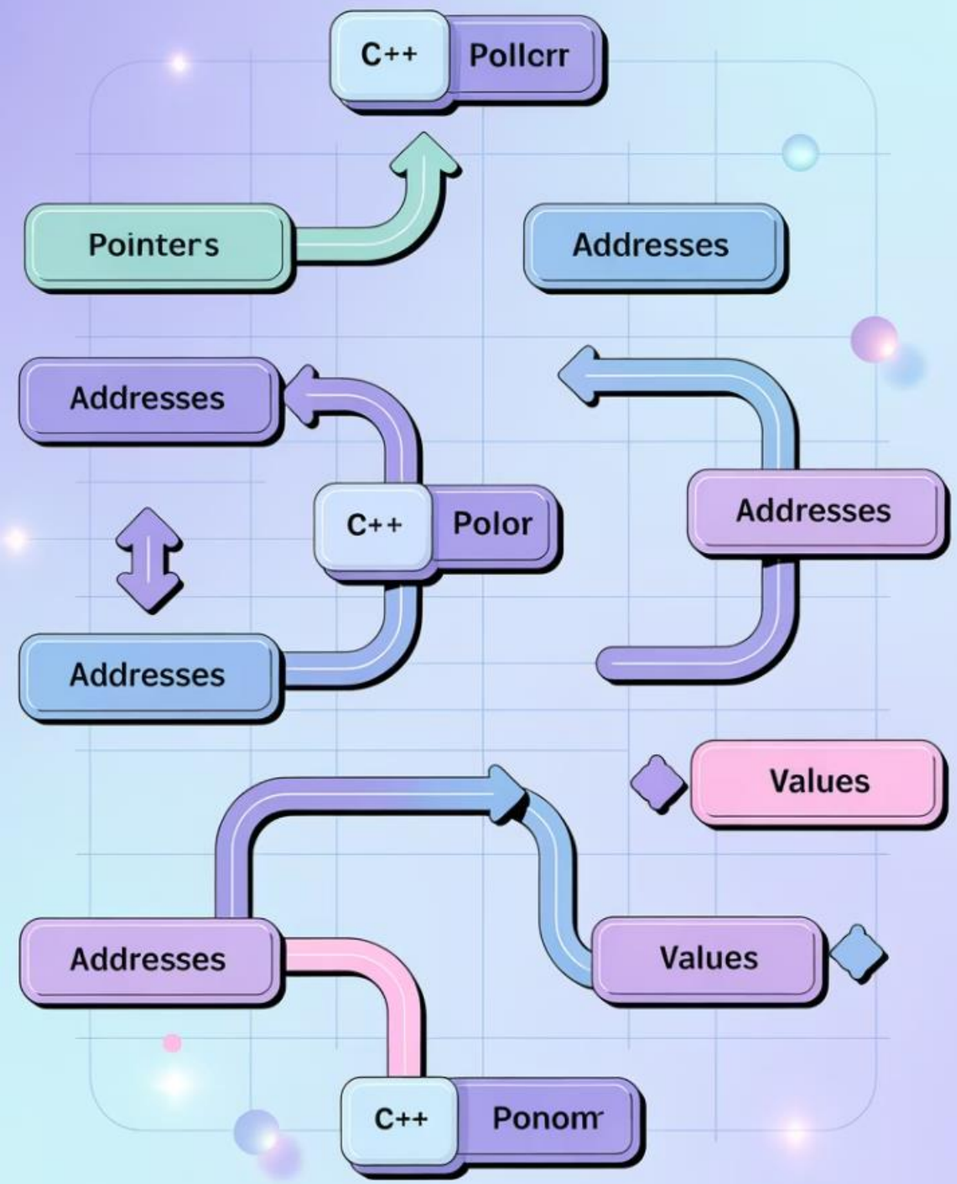


Lecture 10: Advanced Use of Pointers in C++

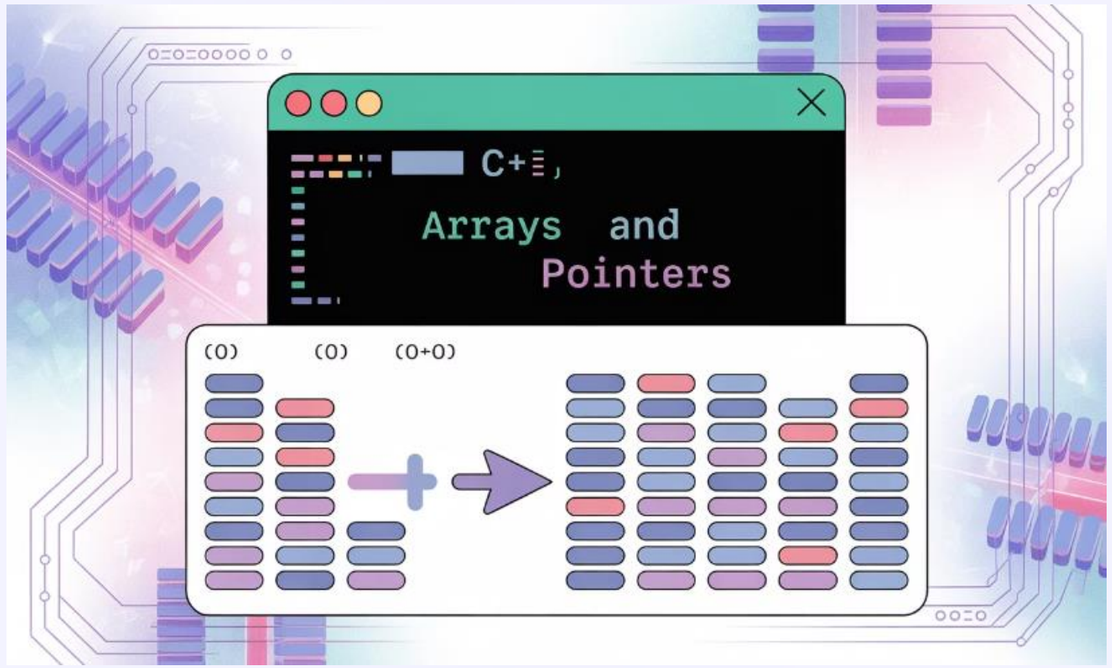
Lecture 10

Asst. Lect. Ali Al-khawaja

Part 1: Pointers and Arrays

Concept

In C++, the name of an array is actually a constant pointer to its first element.



Example: Accessing array elements with pointers

```
int arr[3] = {10, 20, 30};
int* p = arr;
cout << *p << endl; // 10
cout << *(p + 1) << endl; // 20
cout << *(p + 2) << endl; // 30
```

- arr[i] is equivalent to *(arr + i)
- p[i] is valid when p points to the start of an array

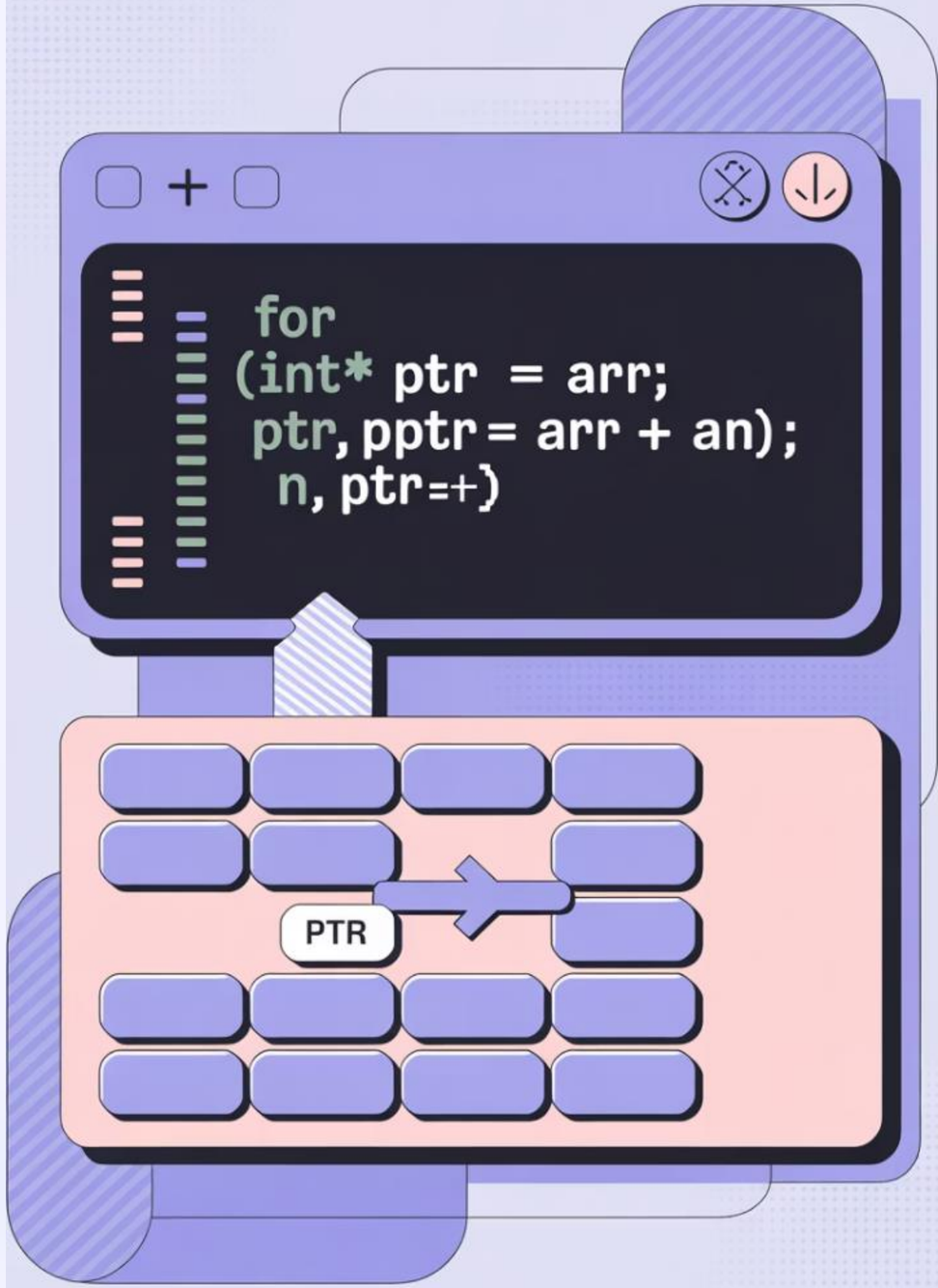
Example: Loop using pointer

Code Example

```
for (int i = 0; i < 3; i++) {
cout << *(p + i) << endl;
}
```

Notes

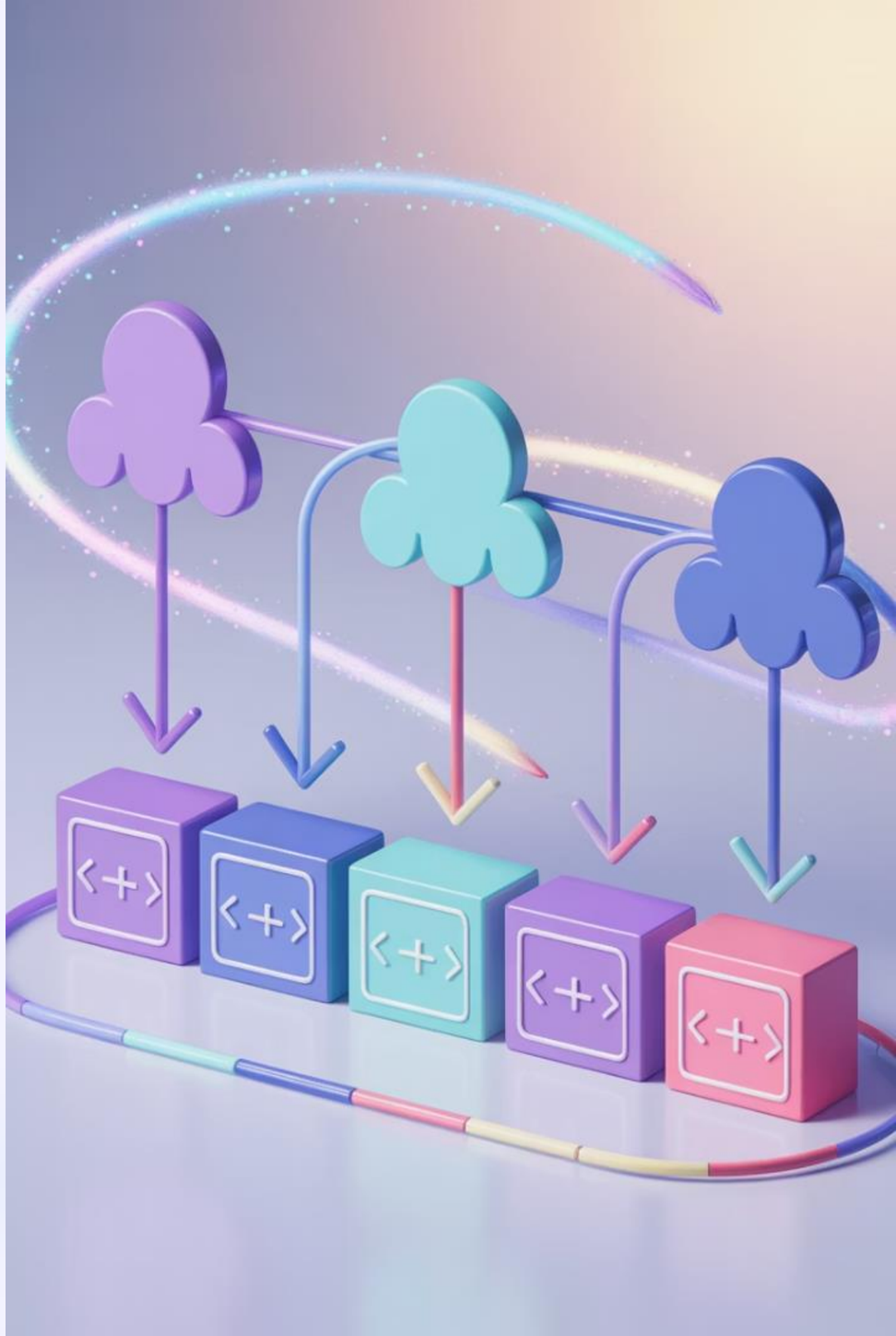
- arr itself is a constant pointer; it cannot be modified (e.g., arr++ is invalid).
- A separate pointer variable p can be incremented or reassigned.

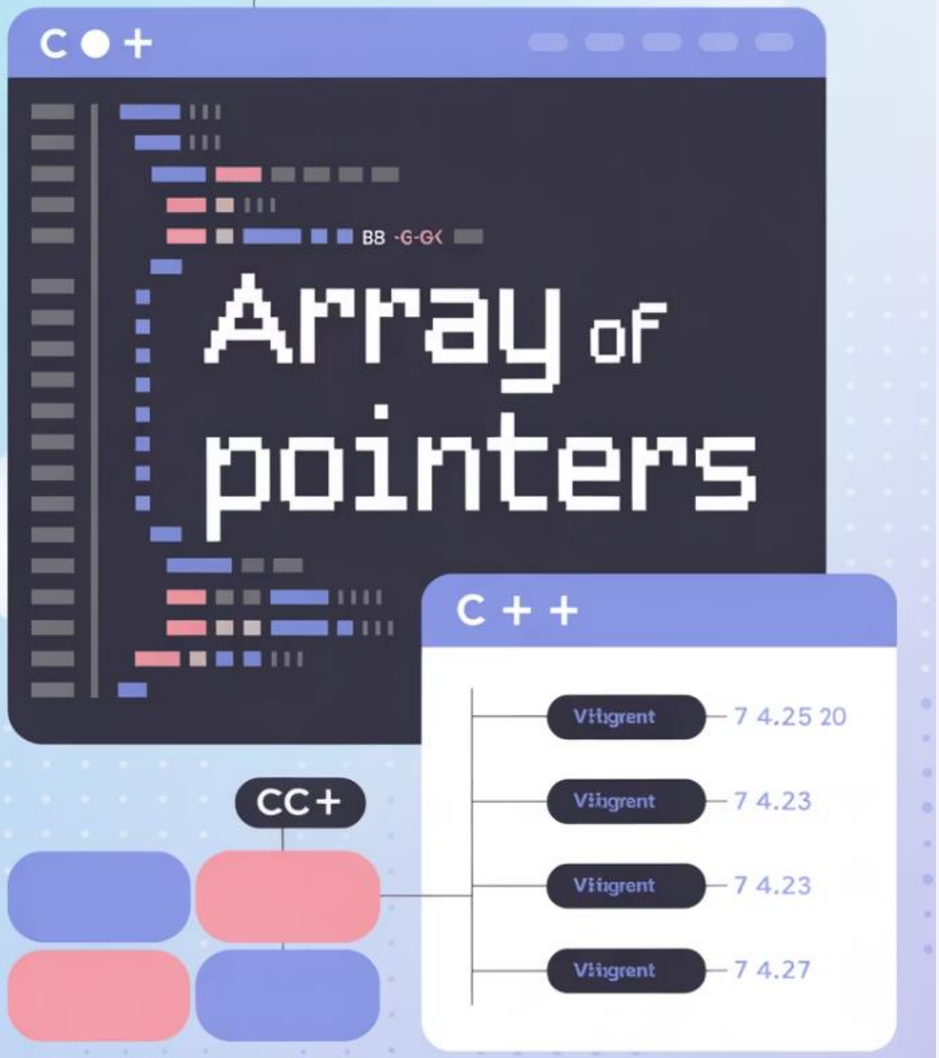


Part 2: Arrays of Pointers

Concept

An array of pointers is an array where each element holds the address of another variable or object.





Example: Array of pointers to integers

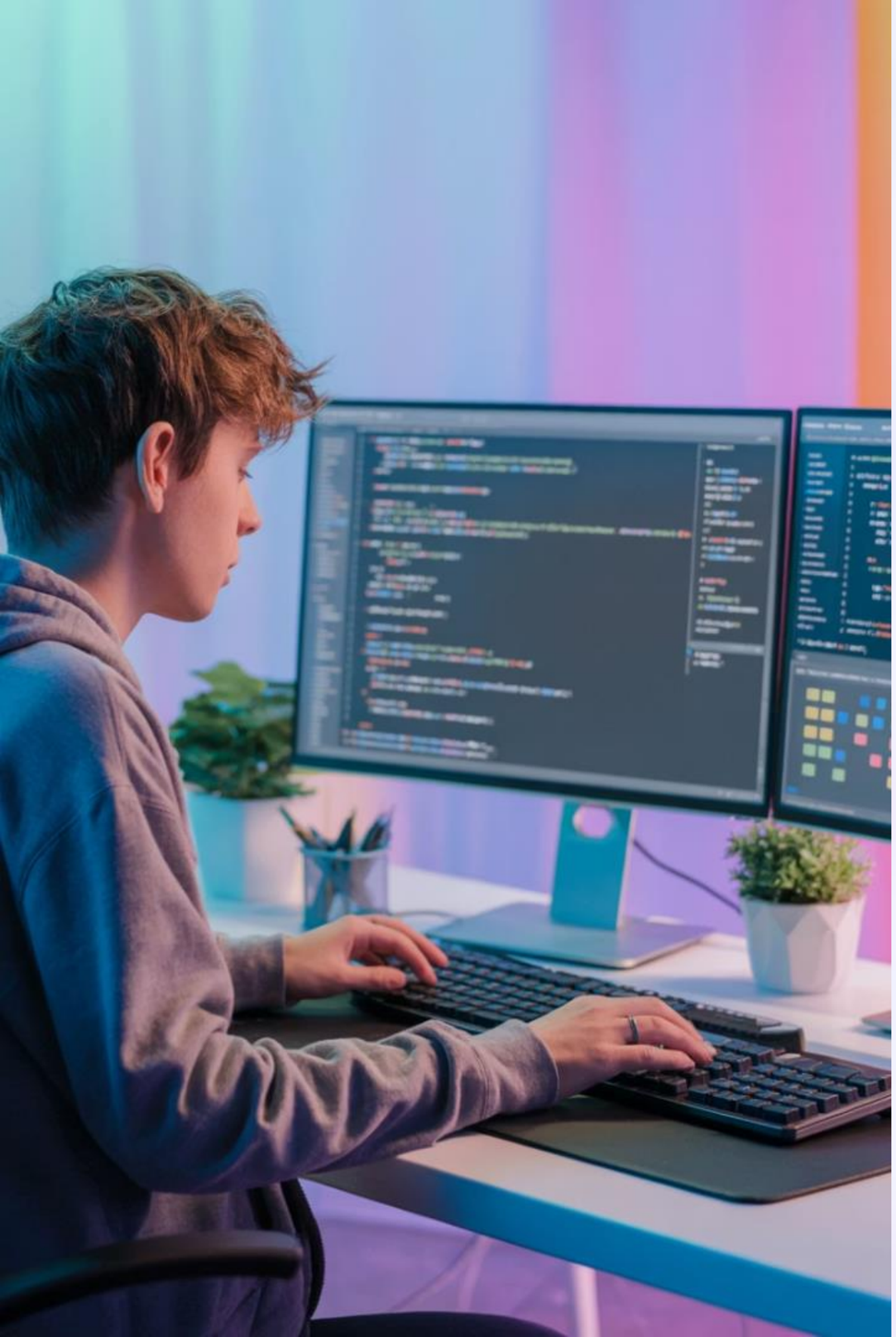
1 Array of pointers to integers

```
int a = 5, b = 10, c = 15;
int* ptrs[3] = {&a, &b, &c};
for (int i = 0; i < 3; i++) {
cout << *ptrs[i] << endl;
}
```

Each element in ptrs is a pointer to an int.

2 Array of pointers to strings

```
const char* fruits[] = {"Apple", "Banana", "Cherry"};
for (int i = 0; i < 3; i++) {
cout << fruits[i] << endl;
}
```



When to Use

Variable-length data

When managing variable-length data (like strings).

Dynamic memory

When dynamically assigning memory to array elements.

Function parameters

For passing arrays of strings to functions.

Part 3: Pointers to Pointers

Concept

A pointer to a pointer is a variable that stores the address of another pointer.

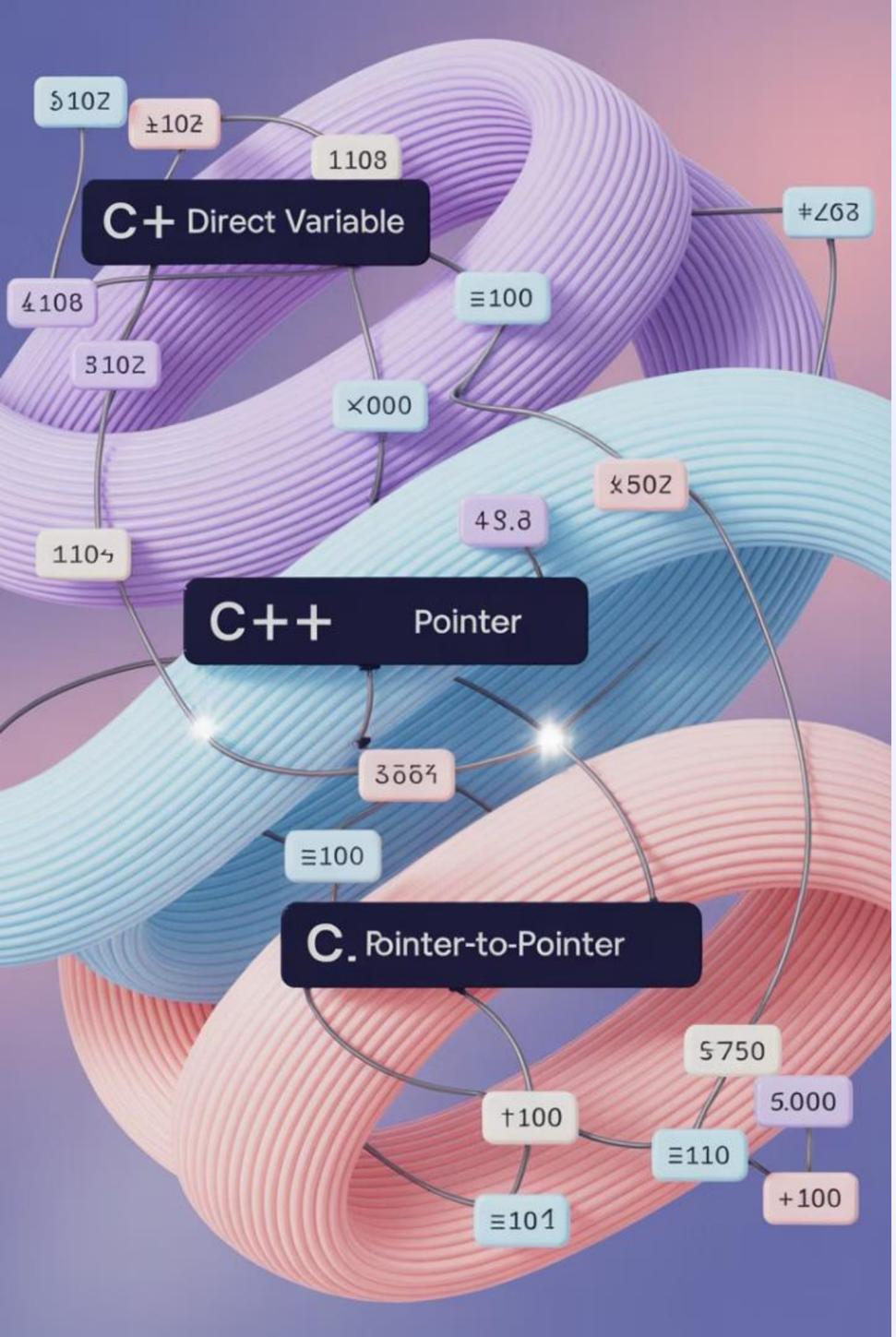
Syntax

Using double asterisk notation

</>

Example

```
int x = 50;
int* p = &x;
int** pp = &p;
```



Accessing Data

1

Direct Access

```
cout << x << endl; // 50
```

2

Via Pointer

```
cout << *p << endl; // 50
```

3

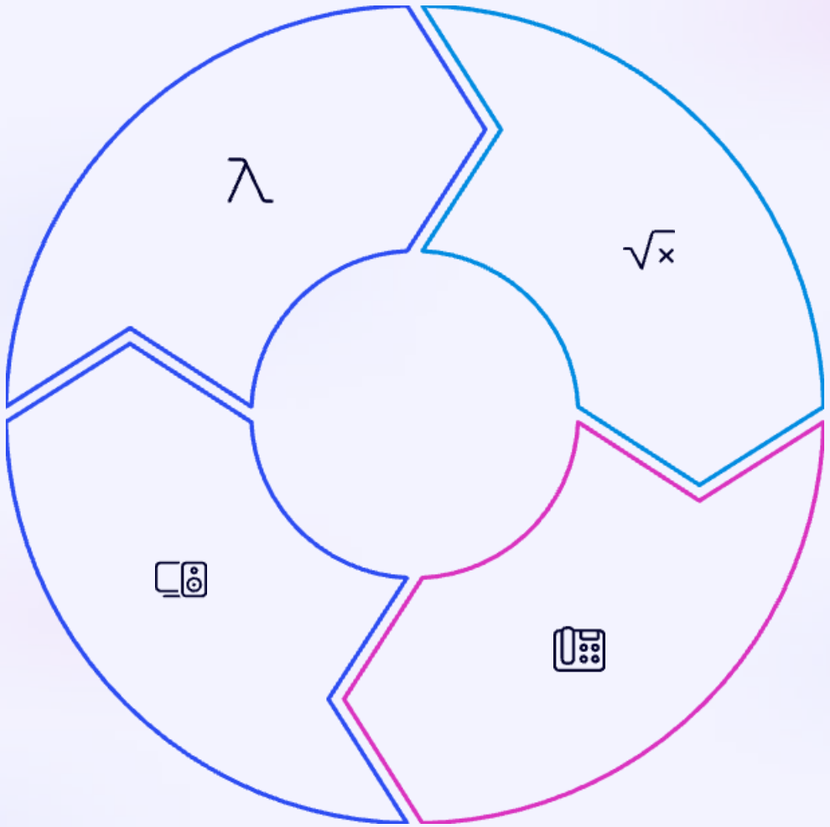
Via Pointer to Pointer

```
cout << **pp << endl; // 50
```

Example: Modifying variable through pointer to pointer

Define Function

```
void change(int** pp) {
**pp = 100;
}
```



Setup Variables

```
int x = 10;
int* p = &x;
```

Check Result

```
cout << x << endl; // Output: 100
```

Call Function

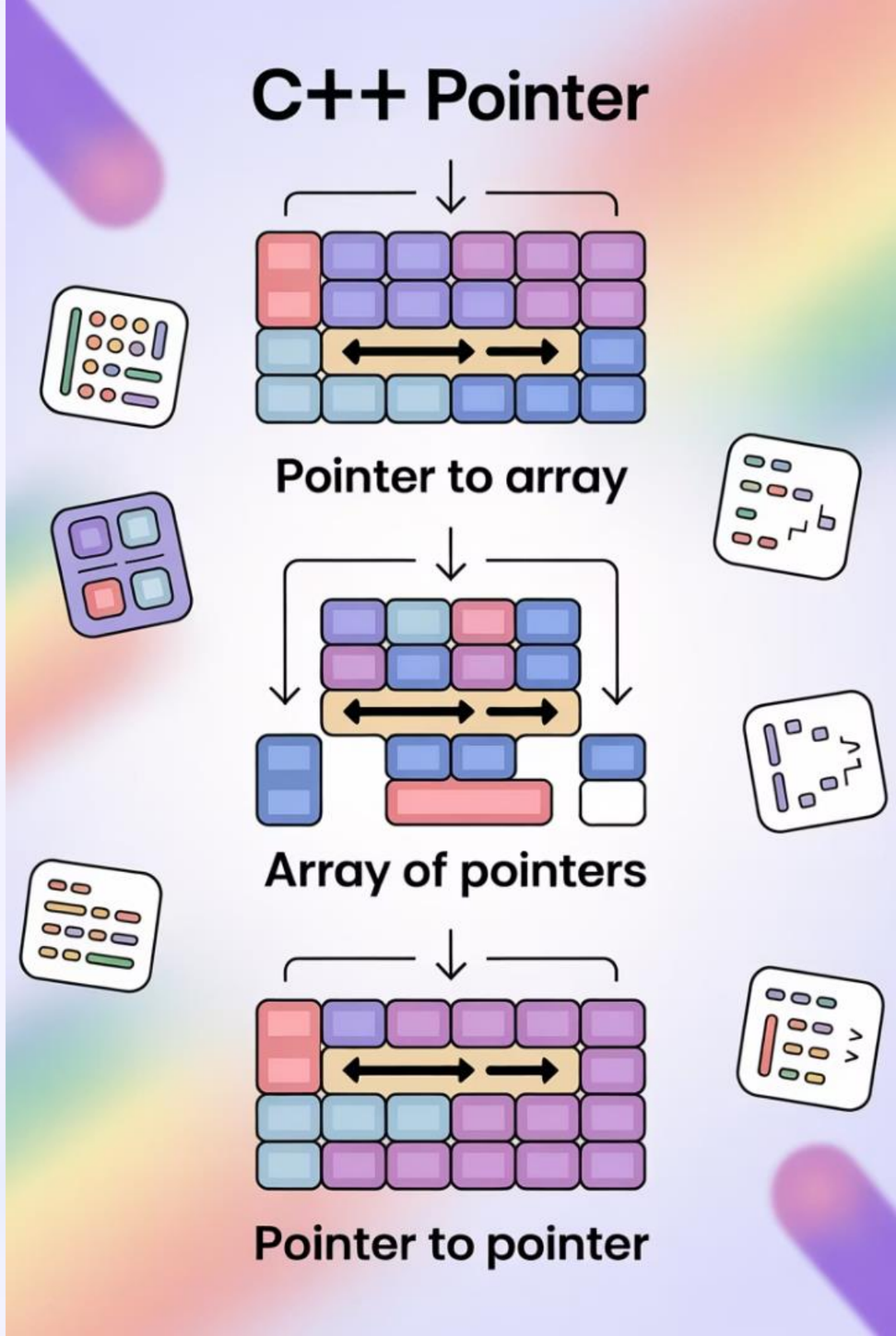
```
change(&p);
```

Common Use Cases

- Dynamic allocation of multi-dimensional arrays.
- Modifying a pointer inside a function.
- Advanced data structures (e.g., linked lists, trees).

Comparison Table

Concept	Description	Example Code
Pointer to Array	Array name acts as pointer	int* p = arr;
Array of Pointers	Array that stores addresses	int* ptrs[3];
Pointer to Pointer	Pointer that stores address of another ptr	int** pp = &p;



Homework

Assignment 1

Write a C++ program that:

- Declares an array of 3 strings using a pointer array
- Prints each string using pointer notation

Assignment 2

Dynamically allocate a 2D matrix using int** and populate it with values using loops.