Understanding the Relationship Between Arrays and Pointers in C++

Objectives

- Understand the basics of arrays and pointers.
- Learn how arrays and pointers are related.
- Manipulate arrays using pointers.
- Pass arrays to functions and understand the implications.

Introduction to Arrays and Pointers

- Arrays:
  - Collection of elements of the same type.
  - Stored in contiguous memory locations.
  - Example: int arr[5] = {1, 2, 3, 4, 5};

- Pointers:
  - Variables that store memory addresses.
  - Example: int* ptr = &arr[0];

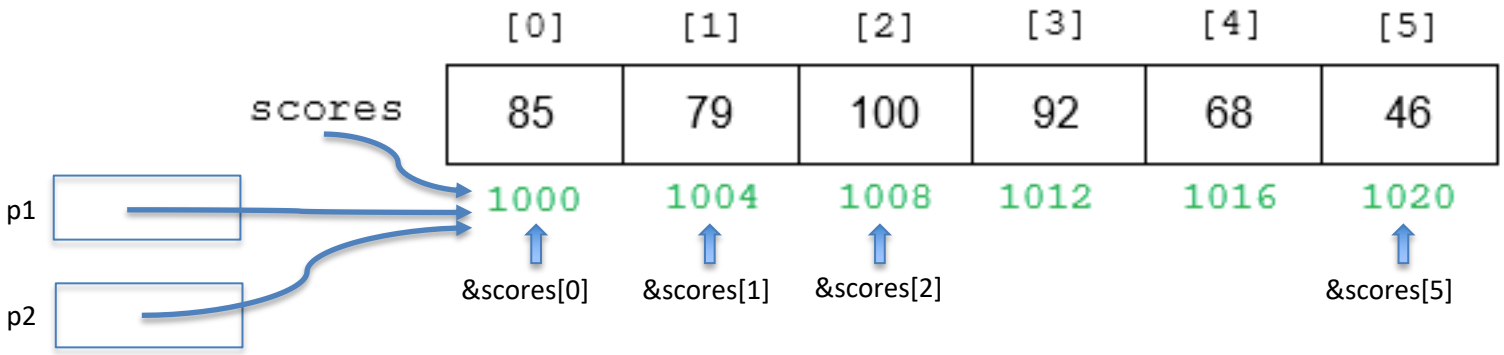
How are Arrays stored in memory

```
int scores[6] = {85, 79, 100, 92, 68, 46};

int * p1 = scores;

int * p2 = &scores[0];

cout << *p1 ; // will output 85
```



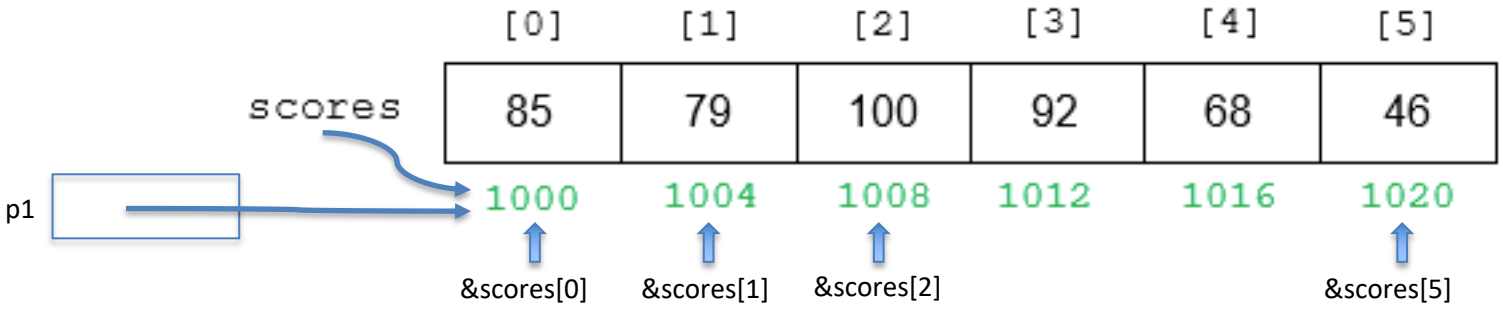
Relationship Between Arrays and Pointers

- Array name (arr) is a pointer to the first element (&arr[0]).
- Pointers can be used to access and modify array elements.

- Example:
- int arr[5] = {1, 2, 3, 4, 5};
- int* ptr = arr; // ptr points to the first element of arr
- std::cout << *ptr; // Output: 1

Pointer Arithmetic

- Pointers support arithmetic operations.
- (p1 + 1) points to the next element in the array.
  - cout << p1; // output: ??
  - cout << *(p1 + 1); // Output: 79. p1+3 = p1 + (3* sizeof(int))
  - cout << (p1+4); // Output ??
  - cout << *(p1 + 4); // Output:?? *(p1+4) = p1[4]

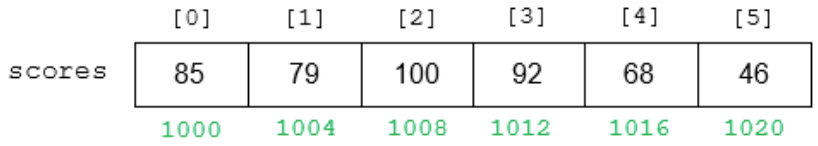


Pointer Arithmetic

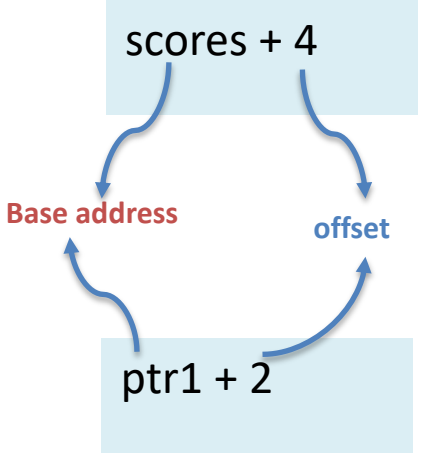
```
int* ptr1 = &scores[2]
ptr1 += 3;

int* ptr2 = &scores[4];
ptr2 -= 2;

int distance = ptr1 - ptr2;
```



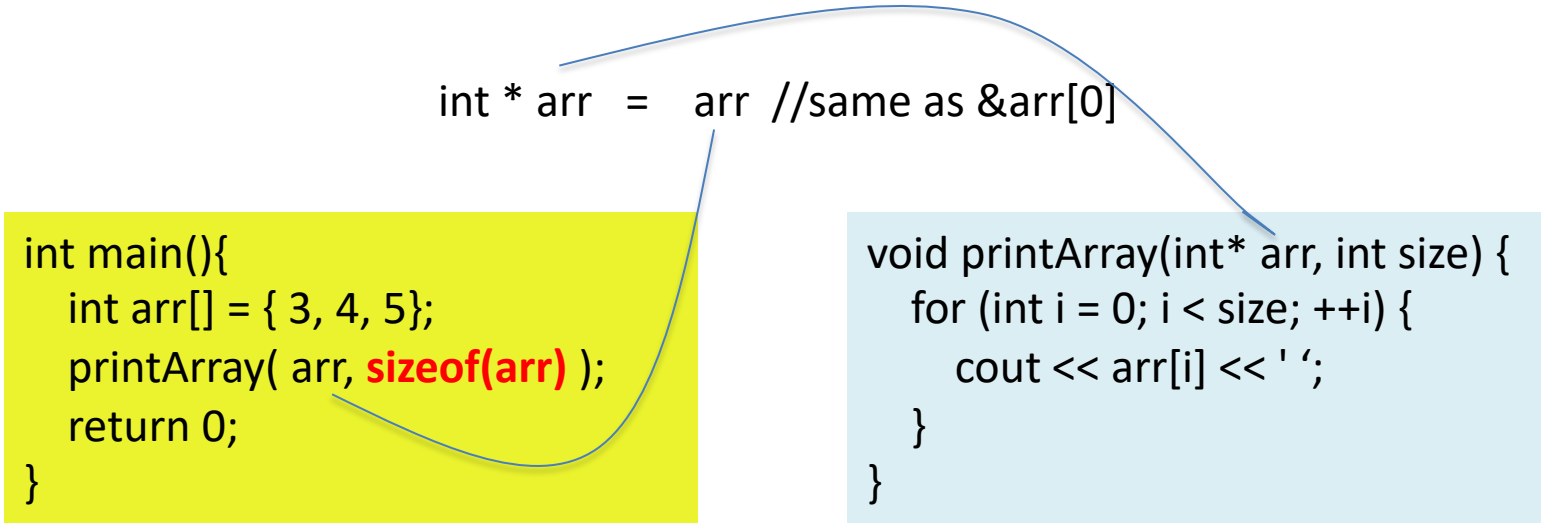
*scores + 1 != *(score +1)



Expression	Value	Data Type	Expression	Value	Data Type
scores[0]	85	int	*(scores+0)	85	int
scores[1]	79	int	*(scores+1)	79	int
scores[2]	100	int	*(scores+2)	100	int
&scores[0]	1000	int*	(scores+0)	1000	int*
&scores[1]	1004	int*	(scores+1)	1004	int*
&scores[2]	1008	int*	(scores+2)	1008	int*

Passing Arrays to Functions

- Arrays are passed to functions as pointers.
- Function receives a pointer to the first element of the array.



Hands-on Exercise

- Task:
  - Declare an array.
  - Use a pointer to access and modify elements.
  - Pass the array to a function that modifies its elements.

```
void doubleArray(int* arr, int size) {
        // implement: double each element of array
}
```

Implement the same with reference

Code Example (continued)

```
int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    int* ptr = arr;

    std::cout << 'Original array: ';
    for (int i = 0; i < 5; ++i) {
        std::cout << arr[i] << ' ';
    }
    std::cout << std::endl;

    modifyArray(ptr, 5);

    std::cout << 'Modified array: ';
    for (int i = 0; i < 5; ++i) {
        std::cout << arr[i] << ' ';
    }
    std::cout << std::endl;

    return 0;
}
```

Review and Q&A

- - Recap key points:
- - Arrays and pointers relationship.
- - Pointer arithmetic.
- - Passing arrays to functions.
- - Q&A session.

Homework Assignment

- Task:
- - Implement a function that takes an array of integers and its size, and returns the sum of its elements using pointers.
- - Write a program that initializes a 2D array and uses a pointer to access and print its elements in row-major order.

Homework Example

- Example:
- int sumArray(int* arr, int size) {
- int sum = 0;
- for (int i = 0; i < size; ++i) {
- sum += arr[i];
- }
- return sum;
- }

- int main() {
- int arr[5] = {1, 2, 3, 4, 5};
- std::cout << 'Sum: ' << sumArray(arr, 5) << std::endl;
- return 0;
- }

References

- - "Programming: Principles and Practice Using C++" by Bjarne Stroustrup
- - "C++ Primer" by Stanley B. Lippman
- - NIU CS241 Notes on Arrays and Pointers