Tutorial 50 - Revisiting Pointers: new and delete Keywords in C++

1. Pointers in C++

- **Definition**: A pointer is a variable used to store the address of another variable.
- **Syntax**: Use `*` to declare a pointer.

Code Example 1: Basic Pointer Usage

```
int a = 4;
int* ptr = &a;
cout << "The value of a is " << *(ptr) << endl;

```

- **Explanation**:
  a. `a` is an integer variable with value `4`.
  b. `ptr` is a pointer storing the address of `a`.
  c. `*(ptr)` dereferences the pointer to access the value stored at that address (`4`).

---

2. `new` Keyword

- **Purpose**: Dynamically allocates memory during runtime.

Code Example 2: Using `new` with Single Variable

```
float *p = new float(40.78);
cout << "The value at address p is " << *(p) << endl;

```

- **Explanation**:
  a. Dynamically allocates memory for a float with value `40.78`.
  b. `p` points to the allocated memory, and `*(p)` retrieves the value.

Code Example 3: Using `new` with Arrays

```
int *arr = new int[3];
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
cout << "The value of arr[0] is " << arr[0] << endl;
cout << "The value of arr[1] is " << arr[1] << endl;
cout << "The value of arr[2] is " << arr[2] << endl;

```

- **Explanation**:
  a. Dynamically allocates memory for an integer array of size 3.
  b. Initializes the array with `10`, `20`, and `30`.
  c. Prints the values using `arr[0]`, `arr[1]`, and `arr[2]`.

---

3. `delete` Keyword

- **Purpose**: Frees dynamically allocated memory to avoid memory leaks.

Code Example 4: Using `delete` with Arrays

```
int *arr = new int[3];
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
delete[] arr; // Frees allocated memory
cout << "The value of arr[0] is " << arr[0] << endl;
cout << "The value of arr[1] is " << arr[1] << endl;
cout << "The value of arr[2] is " << arr[2] << endl;

```

- **Explanation**:
  a. `delete[] arr` deallocates memory used by the array.
  b. Accessing `arr[0]`, `arr[1]`, and `arr[2]` after deletion gives **garbage values**.

---

4. Key Points to Remember

1. **Pointers**:
   - Store the address of variables.
   - Use `*` to declare and dereference.
2. `new` **Keyword**:
   - Dynamically allocates memory during runtime.
   - Example: `int *p = new int(10);`
3. `delete` **Keyword**:
   - Frees dynamically allocated memory.
   - Use `delete ptr;` for single variables and `delete[] ptr;` for arrays.
4. **Garbage Values**:
   - Accessing memory after using `delete` results in garbage values.

Notebook Short Notes

1. **Pointers**: Variables storing addresses; use `*` to declare and dereference.
2. `new`:
   - Allocates memory dynamically.
   - Single variable: `int *p = new int(5);`
   - Array: `int *arr = new int[3];`
3. `delete`:
   - Frees allocated memory.
   - Single variable: `delete ptr;`
   - Array: `delete[] arr;`
4. **Access after Deletion**: Results in **garbage values**.