```
int a = 10;
int *ptr = &a; //normal pointer

const int a = 10;
const int *ptr = &a; //pointer to a constant variable
meaning we cannot modify value of a using ptr or any other way

int a = 10;
int *const ptr = &a; //constant pointer to a variable
meaning the address of the pointer can't be changed but the value at
the address can be changed

const int a = 10;
const int *const ptr = &a; //constant pointer to a constant variable
meaning both the address and the value at the address can't be
changed
```

1. Normal Pointer

cpp

Copy code

int a = 10;

int *ptr = &a;

- ptr is a normal pointer to an integer.
- **Behavior:**
  - You can change both the value of a through ptr (e.g., *ptr = 20).
  - You can make ptr point to another variable (e.g., ptr = &b).

---

2. Pointer to a Constant Variable

cpp

Copy code

const int a = 10;

const int *ptr = &a;

- ptr is a pointer to a **constant integer**.
- **Behavior:**
  - The value of a cannot be modified through ptr (e.g., *ptr = 20 is **not allowed**).
  - ptr can point to another constant integer (e.g., ptr = &b, where b is also const int).

---

3. Constant Pointer to a Variable

cpp

Copy code

int a = 10;

int *const ptr = &a;

- ptr is a **constant pointer** to an integer.
- **Behavior:**
  - The address stored in ptr cannot be changed (e.g., ptr = &b is **not allowed**).
  - The value of a can still be modified through ptr (e.g., *ptr = 20 is allowed).

---

4. Constant Pointer to a Constant Variable

cpp

Copy code

const int a = 10;

const int *const ptr = &a;

- ptr is a **constant pointer** to a **constant integer**.
- **Behavior:**
  - The address stored in ptr cannot be changed (e.g., ptr = &b is **not allowed**).
  - The value of a cannot be modified through ptr or any other way (e.g., *ptr = 20 is **not allowed**).

Key Differences and Use Cases

Declaration	**Address Change (ptr = &b)**	**Value Change (*ptr = value)**	**Example Use Case**
int *ptr	✅ Allowed	✅ Allowed	Standard pointers for dynamic operations.
const int *ptr	✅ Allowed	❌ Not Allowed	Useful for read-only access to variables.
int *const ptr	❌ Not Allowed	✅ Allowed	When you want to lock the pointer to a specific variable but allow value changes.
const int *const ptr	❌ Not Allowed	❌ Not Allowed	When both the pointer address and the value need to remain unchanged.