Types of pointers in C++

There are different types of pointers in C++, and they are as follows:

- Null Pointers
- Double Pointers
- Void Pointers
- Wild Pointers
- Dangling Pointer

1. Null Pointers:
A NULL pointer is a pointer that is pointing to nothing. If we don't have the address to be assigned to a pointer, we can use NULL.

Advantages of Null pointer are:

- We can initialize a pointer variable when that pointer variable is not assigned any actual memory address.
- We can pass a null pointer to a function argument when we are unwilling to pass any actual memory address.

Example:

```
int *p; //Contains garbage value
int *p = NULL; //NULL is constant with vaue 0
int *q = 0; // Same as above
```

Here, we have created a pointer variable that contains garbage values. To dereference the pointer, we have initialized it to NULL to avoid unexpected behavior.

Note: An uninitialized pointer variable contains garbage; this will lead to unexpected results or segmentation faults. Hence, we should never leave a pointer uninitialized and instead.

Example:

```
#include <iostream>
using namespace std;
int main() {
    // Null Pointer
    int * ptr = NULL;
    cout << "The value of ptr is " << ptr;
    return 0;
}
Output:
The value of ptr is 0
```

2. Double Pointers:
We can create a pointer to a pointer that in turn may point to data or another pointer. The first pointer is used to store the address of the variable. And the second pointer is used to store the address of the first pointer. That is why they are also known as double pointers.

Example:

```
int a = 10;
int *p = &a;
int **q = &p;
```

Here q is a pointer to a pointer, i.e., a double-pointer, as indicated by **.

Example:

```
#include<iostream>
using namespace std;
int main() {
    int a = 10;
    int * p = & a; //pointer
    int ** q = & p; //pointer-to-pointer
    /* Next three statements will print same value i.e. address o
    cout << & a << endl;
    cout << p << endl;
    cout << * q << endl;
    /* Next two statements will print same value i.e. address of
    cout << & p << endl;
    cout << q << endl;
    /* Next three statements will print same value i.e. value of
    cout << a << endl;
    cout << * p << endl;
    cout << ** q << endl;
    return 0;
}
Output:
```