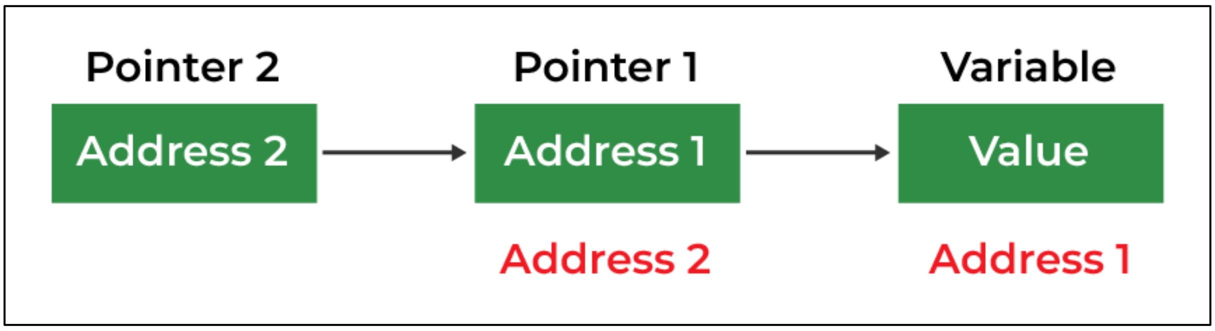
MyAnatomy

Pointers to Pointer

What is Pointer to Pointer (Double Pointer) in C++?

So, when we define a pointer to a pointer, the first pointer is used to store the address of the variables, and the second pointer stores the address of the first pointer. For this very reason, this is known as a Double Pointer or Pointer to Pointer.



The above diagram shows the memory representation of a Pointer to Pointer or a Double Pointer, we can easily understand that the address of the **variable (i.e Address 1)** is stored in **Pointer 1** and the address of **Pointer 1(i.e Address 2)** is stored in **Pointer 2.** This is known as Double Pointers or Pointer to Pointer.

Declaration of Pointer to Pointer:

Syntax- type **double_pointer = &pointer;

Examples

int val = 123;

int *single_ptr = &val; // storing address of val to pointer ptr.

int **double_ptr = &single_ptr; // pointer to a pointer declared which is pointing to an integer.

Note: if we change the value of highest pointer it will reflect in all the pointers.

Working of Pointers to Pointers

```
#include <iostream>
using namespace std;
int main(void)
{
   int val = 123;
   int *single_ptr = &val;
   int **double_ptr = &single_ptr;
   int ***triple_ptr = &double_ptr;
   int ****four_ptr = &triple_ptr;
   cout << val << endl;
   cout << single_ptr << endl;
   cout << double_ptr << endl;
   cout << triple_ptr << endl;
   cout << four_ptr << endl;
   ****four_ptr = 99;
   cout << val << endl;
}
```

Size of Pointers to Pointers

In the C++ programming language double pointer behave similarly to a normal pointer. So, the size of the variable of the double-pointer and the size of the normal pointer variable is always equal.

```
   cout << "sizeofval - " << sizeof(val) << endl;
   cout << "sizeof single_ptr - " << sizeof(single_ptr) << endl;
   cout << "sizeof double_ptr - " << sizeof(double_ptr) << endl;
   cout << "sizeof triple_ptr " << sizeof(triple_ptr) << endl;
   cout << "sizeof four_ptr " << sizeof(four_ptr) << endl;
```