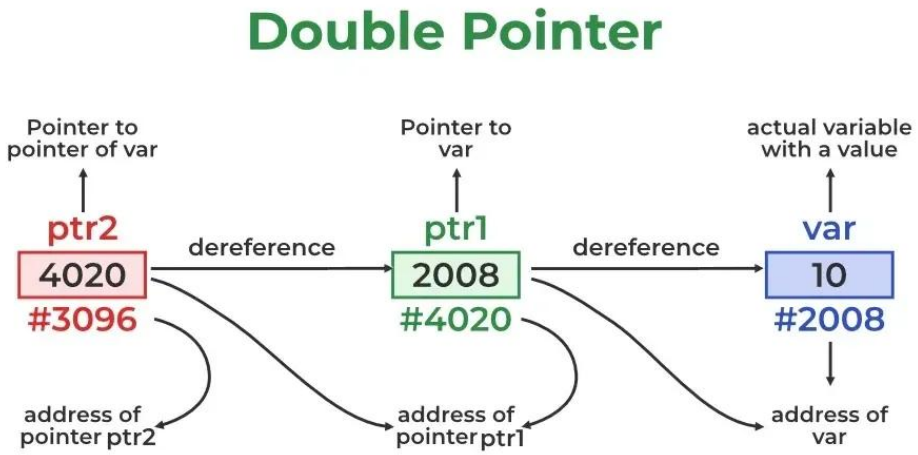
double pointers:

are those pointers which stores the address of 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 a **pointer to pointer**.

```c
#include <stdio.h>

int main() {

    // A variable
    int var = 10;

    // Pointer to int
    int *ptr1 = &var;

    // Pointer to pointer (double pointer)
    int **ptr2 = &ptr1;

    printf("var: %d\n", var);
    printf("*ptr1: %d\n", *ptr1);
    printf("**ptr2: %d", **ptr2);

    return 0;
}
```

Output

```
var: 10
*ptr1: 10
**ptr2: 10
```



Pointers and Arrays:

- An **array name** acts like a **constant pointer** to the first element of the array.
- You can use pointer arithmetic to traverse an array.
- Array indexing (arr[i]) is equivalent to pointer dereferencing (*(arr + i)).

```c
int arr[5] = {10, 20, 30, 40, 50};
int *ptr = arr; // ptr points to the first element of arr

printf("%d\n", arr[2]);  // 30 (using array indexing)
printf("%d\n", *(ptr + 2)); // 30 (using pointer arithmetic)
```

Pointers and Strings:

- A **pointer** can be used to traverse and manipulate strings.
- Since strings are arrays, pointer arithmetic applies similarly.

```c
char str[] = "Pointer";
char *ptr = str;

while (*ptr != '\0') {
    printf("%c ", *ptr);
    ptr++; // Move to the next character
}
// Output: P o i n t e r
```

A pointer to a function : is similar to a pointer to a variable. However, instead of pointing to a variable, it points to the address of a function. This allows the function to be called indirectly, which is useful in situations like callback functions or event-driven programming.

```c
#include <stdio.h>

int add(int a, int b) { return a + b; }

int main() {

    // Declaring function pointer
    int (*fptr)(int, int);

    // Assigning address of add() functoin to fptr
    fptr = &add;

    // Calling function using function pointer
    int res = fptr(3, 4);

    printf("%d", res);
    return 0;
}
```

Output

```
7
```