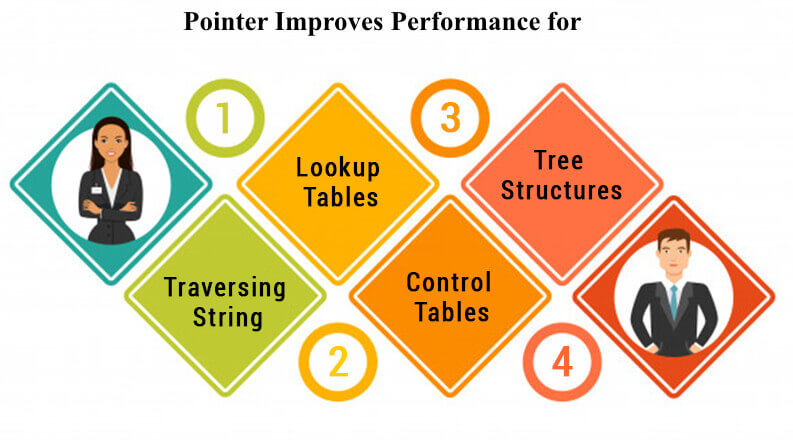
Pointer

Pointer is used to points the address of the value stored anywhere in the computer memory. To obtain the value stored at the location is known as dereferencing the pointer. Pointer improves the performance for repetitive process such as:

* Traversing String
* Lookup Tables
* Control Tables
* Tree Structures



Pointer Details

* **Pointer arithmetic:** There are four arithmetic operators that can be used in pointers: ++, --, +, -
* **Array of pointers:** You can define arrays to hold a number of pointers.
* **Pointer to pointer:** C allows you to have pointer on a pointer and so on.
* **Passing pointers to functions in C:** Passing an argument by reference or by address enable the passed argument to be changed in the calling function by the called function.
* **Return pointer from functions in C:** C allows a function to return a pointer to the local variable, static variable and dynamically allocated memory as well.



Program

Pointer

1. #include <stdio.h>
3. **int** main( )
4. {
5. **int** a = 5;
6. **int** \*b;
7. b = &a;
9. printf ("value of a = %d\n", a);
10. printf ("value of a = %d\n", \*(&a));
11. printf ("value of a = %d\n", \*b);
12. printf ("address of a = %u\n", &a);
13. printf ("address of a = %d\n", b);
14. printf ("address of b = %u\n", &b);
15. printf ("value of b = address of a = %u", b);
16. **return** 0;
17. }

Output

1. value of a = 5
2. value of a = 5
3. address of a = 3010494292
4. address of a = -1284473004
5. address of b = 3010494296
6. value of b = address of a = 3010494292

Program

Pointer to Pointer

1. #include <stdio.h>
3. **int** main( )
4. {
5. **int** a = 5;
6. **int** \*b;
7. **int** \*\*c;
8. b = &a;
9. c = &b;
10. printf ("value of a = %d\n", a);
11. printf ("value of a = %d\n", \*(&a));
12. printf ("value of a = %d\n", \*b);
13. printf ("value of a = %d\n", \*\*c);
14. printf ("value of b = address of a = %u\n", b);
15. printf ("value of c = address of b = %u\n", c);
16. printf ("address of a = %u\n", &a);
17. printf ("address of a = %u\n", b);
18. printf ("address of a = %u\n", \*c);
19. printf ("address of b = %u\n", &b);
20. printf ("address of b = %u\n", c);
21. printf ("address of c = %u\n", &c);
22. **return** 0;
23. }

Pointer to Pointer

1. value of a = 5
2. value of a = 5
3. value of a = 5
4. value of a = 5
5. value of b = address of a = 2831685116
6. value of c = address of b = 2831685120
7. address of a = 2831685116
8. address of a = 2831685116
9. address of a = 2831685116
10. address of b = 2831685120
11. address of b = 2831685120
12. address of c = 2831685128