

Day-15 of 101days Coding challenge.

-----POINTER-----

- * dereferencing , (or used to access the value)
- & referencing , (or using the address of the variable)

⇒ Pointer----

```
#include<iostream>

using namespace std;

int main()
{
    int value = 20;
    int *ptr; // here ptr data type should be as value type (int)
    ptr = &value; // assigning address of value to the ptr
    cout<<ptr<<endl; // it will print the address of the value
    cout<<&ptr<<endl; // it will print the address of the ptr
    cout<<*ptr<<endl; // it will print the value stored into the ptr

    *ptr = 30;
    cout<<*ptr<<endl; // it will return the updated value 30

    // applying increment and decrement to the pointer
    ptr++; // address will increase as per data type
    ++ptr; // address will increase;

    cout<<ptr<<endl;
    // note we can't increase the value of the pointer as shown below
    // it will generate the random value , because it initializes once (solution we can traverse the array)
    *ptr++;
    ++*ptr;
    cout<<*ptr<<endl;
}
```

⇒ Output-----

```
0x70fe1c
0x70fe10
20
30
0x70fe24
4199401

-----
Process exited after 0.08807 seconds with return value 678938
Press any key to continue . . .
```

⇒ Pointer To array

```
#include<iostream>
#include<stdlib.h>
using namespace std;

int main()
{
    int arr[] = {10,20,30,40,50,60,70,80}; // array declaration

    int *ptr = arr; //it will point the first elements of the array (point to the base address)

    // using for loop to increment the pointer for traversing the elements of the array
    for(int i = 0; i<8; i++)
    {
        cout<<i<<":index value="<<*ptr<<endl;
        ptr++; // incrementing the address of the ptr so that pointer can access the value
    }
    return 0;
}
```

⇒ Output

```
0:index value=10
1:index value=20
2:index value=30
3:index value=40
4:index value=50
5:index value=60
6:index value=70
7:index value=80

-----
Process exited after 0.1138 seconds with return value 0
Press any key to continue . . .
```

⇒ Pointer To pointer

```
#include<iostream>

using namespace std;

int main()
{
    int v = 40;
    int *ptr;
    ptr = &v;
    int **ptr1 = &ptr; // here pointer to pointer is used to access the single pointer

    cout<<*ptr<<endl; // will give the value of the pointer(40)
    cout<<ptr<<endl; // adress
    cout<<**ptr1<<endl; // value
    return 0;
}
```

⇒ Output

```
40
0x70fe14
40

-----
Process exited after 0.09166 seconds with return value 0
Press any key to continue . . .
```

-----Now Some Online Found Resources-----

⇒ Pointer

Pointers are variables that store the address of other variables

```
int main(){  
    int a = 10;  
    int *aptr;  
    aptr = &a;  
  
    cout<<&a<<endl; // 2000  
    cout<<aptr<<endl; // 2000  
  
    cout<<*aptr<<endl; // 10  
  
    return 0;  
}
```

Memory

The diagram illustrates memory allocation for the provided C++ code. It features a vertical stack of light blue rectangular blocks representing memory cells. On the left side of the stack, the addresses 2000 and 4000 are marked. The block at address 2000 is labeled 'a = 10' in red text. The block at address 4000 is labeled 'aptr = 2000' in red text. The rest of the memory stack is empty.

⇒ Pointer To Pointer

Pointer to Pointer

```
int main(){
```

```
    int a = 10;
```

```
    int *p;
```

```
    p = &a;
```

```
    cout<<*p<<endl; // 10
```

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

```
    cout<<*q<<endl; // 2000
```

```
    cout<<**q<<endl; // 10
```

```
    return 0;
```

```
}
```

Memory

