Pointers: (8)

operations: (*) (int)

1. Addition } Jump
2. Sub
3. Assignment
4. Relational

Pointers and Functions.

1) pass pointers to function
   - → value
   - → reference.
2) return from a function.

```
int main()
{
    int x=10;
    int *p= &x;
    cout<<2;
    fun(2);
    fun(x);
    fun2(x);
    fun2(34);
    return 0;
}
```

```
by value
void fun(int x)
{
    x++;
    cout<<x;
}
```

```
by reference
void fun2(int &x)
{
    x++;
    cout<<x;
}
```

```
int main()
{
    int x=10;
    int *p= &x;
    fun(p);
    cout<<x<<*p;
    fun(&x);
    fun(NULL);
}
```

```
void fun(int *p)
{
    cout<<p;
    cout<<*p;
    *p=100;
    p++;
}
```

```
void fun2(int *&p)
{
    cout<<p;
    cout<<*p;
    *p=100;
    p++;
}
```

pointer by value or by reference

Data by reference ✗

```
fun2(p);
cout<<x<<*p;
```

```
fun2(&x);
```

Error

```
int *p;
p=&x;
p++;
cout<<p;
cout<<*p;
```

use of const with pointers:

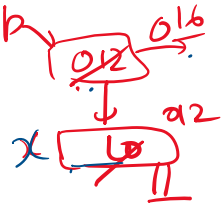
int x = 100;
const int y = 50; (Named constants)

Const

1) Non-constant pointer to Non-constant data.

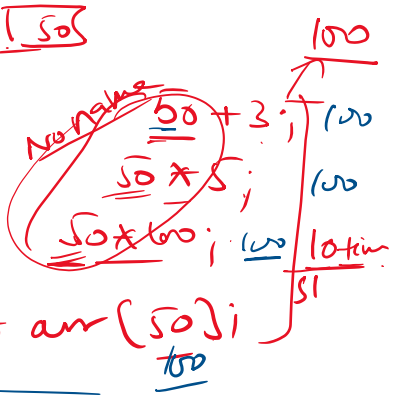
Address / data int/char/double

int x = 10;
int *p = &x;
p++;
(*p)++;



x++;
y++;
y = 500;

const int y = 50;
{ y+3; y*5; y*100; }
int arr[y];



int arr[50];
100

main
{ int x = 50; fun(x); x++; }

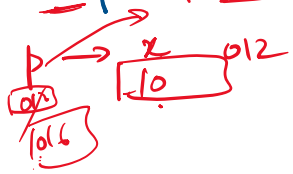
void fun(const int &x)
{ cout << x; x++; } read write

fun(int x, const int y);
read + write
read

2) Non-constant pointer to constant data.

const int x = 10;
const int *p = &x;
p++;
(*p)++;
p = NULL;
p = &y;

*p = 100;



3) Constant pointer to non-constant data

int x = 10;
int * const p = &x;
p++; (*p)++;
int y = 100;
p = &y; X
p = NULL; X
*p = 100;
*p = y;

Reader function.

4) constant pointer to constant data.

const int x = 10;
const int * const p = &x;
p++; X
p = &y; X
(*p)++; X
*p = 100; X
*p = y; X

int *p = &y;
read ✓ write ✓

int y = 100; read ✓ write ✓
const int *p = &y; ✓
Data read ✓ write ✗
g(100)

const int x = 10; ✗ write
const int *p = &x;

void fun(const int *p)
{ *p = 500; X
cout << *p; ✓
}

fun(&y);
y++; y = 50;