

⇒ **Function Pointers**

```
int x = 10;
```

```
int *xptr = &x;
```

```
cout << *xptr; // 100
```

x (int)
10
4B

(int*) 8B
100
xptr

gptr

```
int (*aptr)(int, int) = &add or add;
```

```
void greet() {  
  cout << "namaste";  
}
```

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

```
bool ascending(int a, int b) {  
  return a > b;  
}
```

&greet
or
greet

&add
or
add

&ascending
or
ascending

```
void (*gptr)() = &greet; or greet
```

```
bool (*asptr)(int, int) = &ascending; or ascending
```

(*gptr) () ; // namaste

(*aptr) (2, 3) ;
5

(*asptr) (2, 3)
false

$$a_{i+1} = \frac{1}{2} (a_i + \frac{1}{a_i})$$

```
void bubbleSort(int* arr, int n) {
    for(int i=1; i<n; i++) {
        for(int j=0; j<n-i; j++) {
            if(arr[j] > arr[j+1]) {
                swap(arr[j], arr[j+1]);
            }
        }
    }
}
```

↑ing

Ling

```

bool ascending(int a, int b) {
    return a > b;
}

void bubbleSort(int* arr, int n) {
    for(int i=1; i<n; i++) {
        for(int j=0; j<n-i; j++) {
            if(ascending(arr[j], arr[j+1])) {
                swap(arr[j], arr[j+1]);
            }
        }
    }
}

```

```
bool descending(int a, int b) {
    return a < b;
}

void bubbleSort(int* arr, int n) {
    for(int i=1; i<n; i++) {
        for(int j=0; j<n-i; j++) {
            if(descending(arr[j], arr[j+1])) {
                swap(arr[j], arr[j+1]);
            }
        }
    }
}
```

bool (*f)(int, int) {
 bubbleSort(arr, n, {ascending
 or
 ascending})
 bubbleSort(arr, n, {descending
 or
 descending})
}

```
bool ascending(int a, int b) {
    return a > b;
}

bool descending(int a, int b) {
    return a < b;
}

void bubbleSort(int* arr, int n, bool (*f)(int, int)) {
    for(int i=1; i<n; i++) {
        for(int j=0; j<n-i; j++) {
            if(f(arr[j], arr[j+1])) // (*f)(arr[j], arr[j+1])
                swap(arr[j], arr[j+1]);
        }
    }
}
```

bubblesort(arr, n, ascending)

```
bubblesort(arr, n, ascending)
```

`bubblesort(arr, n, descending)` & descending

