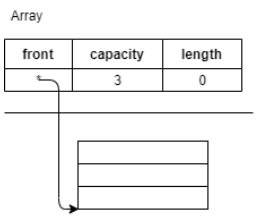
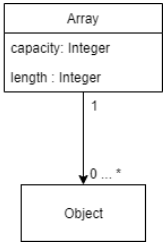
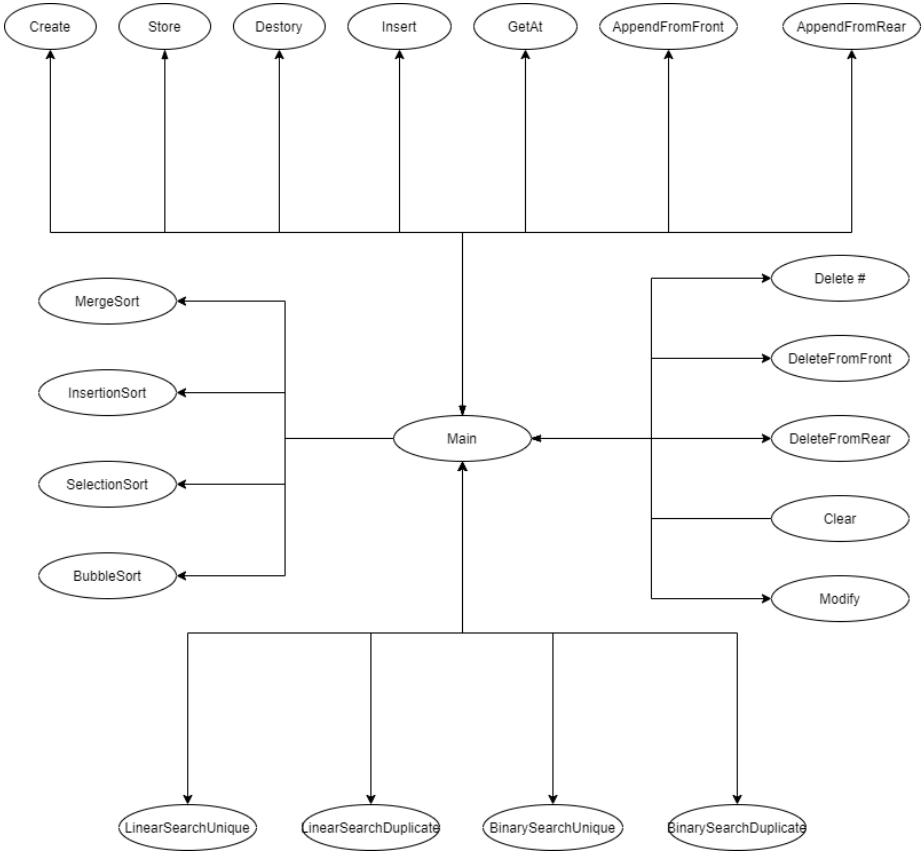
Array

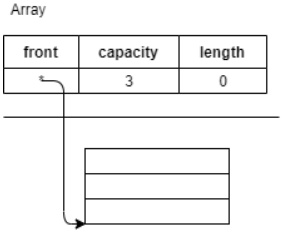
by D4R6

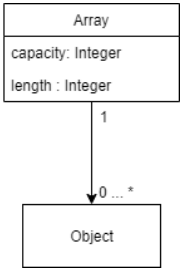
Array Design











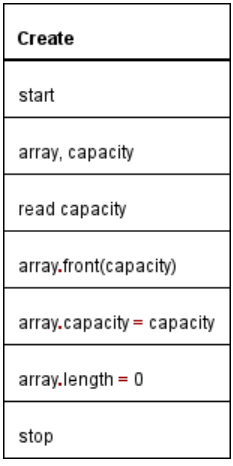

```c
typedef signed long int Long;

typedef struct _array {
    void(*front);
    Long capacity;
    Long length;
}Array;
```

처리 과정

1.할당량을 입력 받는다.

2.할당량 만큼 메모리를 할당 한다.

3.끝낸다.




```
void Create(Array* array, Long capacity, size_t size)
{
    array->front = calloc(capacity, size);
    array->capacity = capacity;
    array->length = 0;
}
```

처리 과정

1. array 를 할당 해제한다.

2.끝낸다.


```
void Destory(Array* array)
{
        if (array->front != NULL) {
        free(array->front);
        array->front = NULL;
    }
}
```

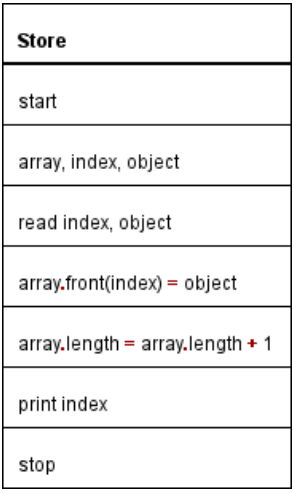
처리 과정

1.위치와 개체를 입력 받는다.

2.받은 개체를 위치에 적는다.

3.위치를 출력한다.

4.끝낸다.




```
void Store(Array* array, Long index, void* object, size_t size)
{
    // sizeof(void) : unknown size.
    memcpy((char*)(array->front) + (index * size), object, size);
    array->length++;
    return index;
}
```

처리 과정

1. 위치와 개체를 입력 받는다.

2. 새로운 배열을 할당 한다.

3. 위치보다 작은 동안 반복한다.
- 기존 배열 요소를 옮겨 적는다.

4. 사용량 만큼 반복한다.
- 기존 배열 요소를 옮겨 적는다.

5. 기존 배열을 지운다.

6. 새로운 배열의 위치 개체를 적는다.

7. 위치를 출력한다.

8. 끝낸다.

```
Insert
start
array, index, object, temps(), i = 1, j = 1
read index, object
temps(array.capacity + 1)
while( i < index )
    temps(i) = array.front(j)
    j = j + 1
    i = i + 1
i = i + 1
while(j ≤ array.length)
    temps(i) = array.front(j)
    i = i + 1
    j = j + 1
array.front = null
array.front = temps
array.capacity = array.capacity + 1
array.front(index) = object
array.length = array.length + 1
print index
stop
```

```
void Insert(Array* array, Long index, void* object, size_t size)
{
    void(*temp);
    Long i = 0;
    Long j = 0;
    Long capacity = array->capacity + 1;

    temp = calloc(capacity, size);

    while (i < index) {
        memcpy((char*)temp + (i * size),
        ((char*)array->front) + (i * size),
        size);
        i++;
        j++;
    }

    i++;

    while (j < array->length) {
        memcpy((char*)temp + (i * size),
        ((char*)array->front) + (i * size),
        size);
        i++;
        j++;
    }

    if (array->front != NULL) {
        array->front = temp;
        array->capacity++;
        memcpy(((char*)array->front) + (index * size),
        object,
        size);

        array->length++;
        return index;
    }
}
```

처리 과정

1. 개체를 입력 받는다.

2. 새로운 배열을 할당 한다.

3. length 만큼 반복한다.

- 새로운 배열 두 번째 칸부터 기존 배열 내용을 기록한다.

4. 새로운 배열 첫 칸에 개체를 기록한다.

5. 기존 배열을 새로운 배열로 대체한다.

6. 위치를 출력한다.

7. 끝낸다.