

Let's solve it

19

Q Read n and display reverse in order ✓

i.e

0	12
1	25
2	7
3	19
4	21

→

21
19
7
25
12

int arr[5];

```
for (i=0; i<5; i++)
{
    scanf("%d", &arr[i]);
}
```

```
for (j=5; j>0; j--)
{
    printf("%d\n",
        arr[j-1]);
}
```

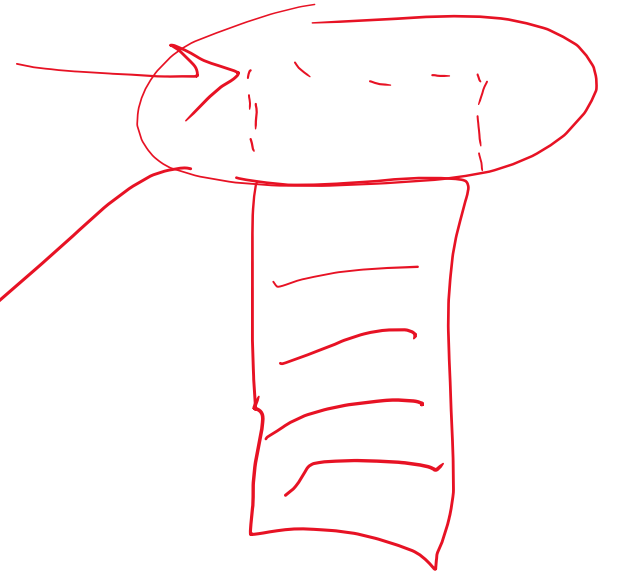
limitation of array in C

→ The use of index is not cross checked for validity at compile time.

arr[-1] ⇒

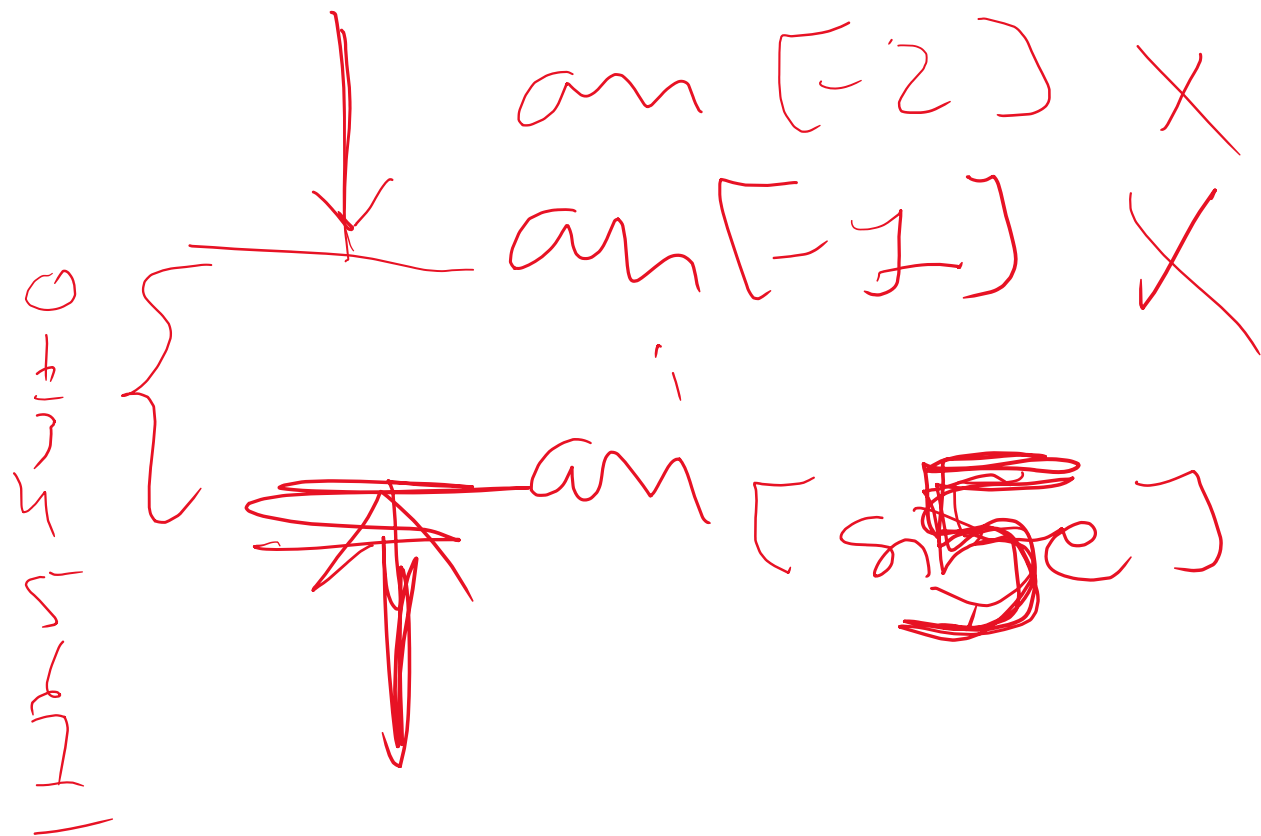
The program will

fail at runtime, if that memory belongs to someone else, i.e. segment.



"segmentation fault"

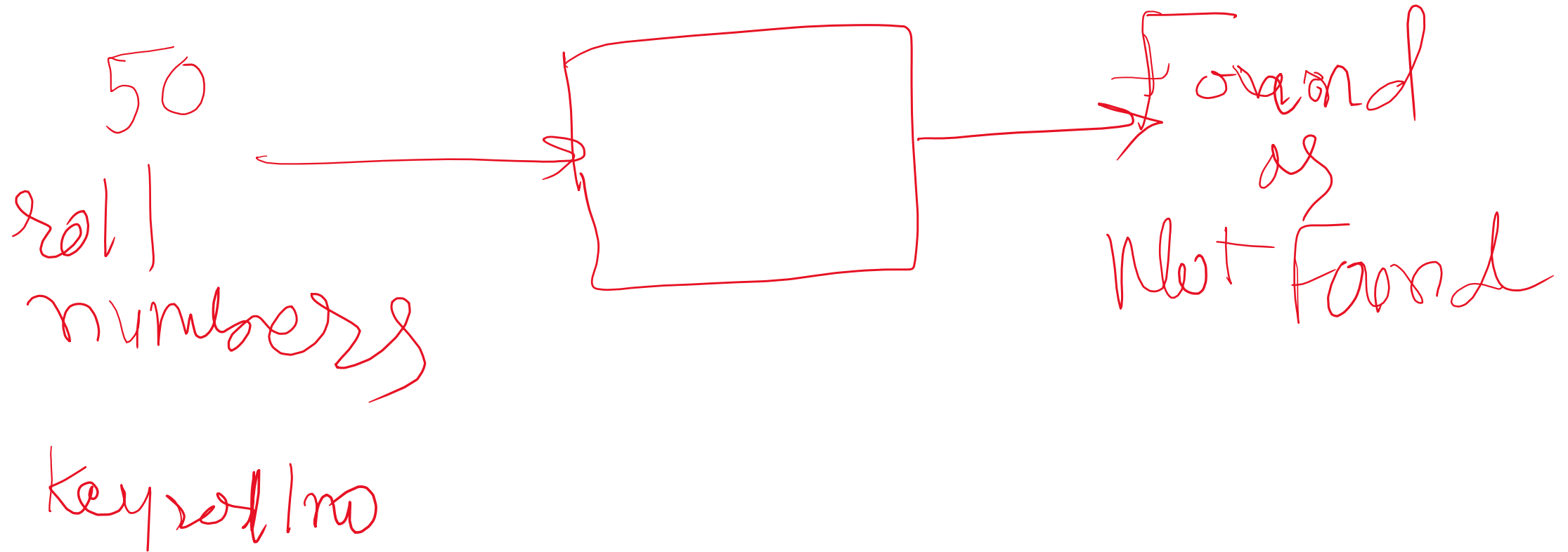
There is a high chance that
array index is out of bound
(boundary)



Size of 5
0 to 4

It is developer's responsibility
to make sure that array index
is in valid range for that array.

Ex: 2 linear search



```
int main()
```

```
{ int cno, sno; scanf("%d", &sno);
```

```
for (i = 0; i < 50; i++)
```

```
{ scanf("%d", &cno);
```

```
if (cno == sno)
```

```
{ printf("Found");
```

```
} exit(0); // return 0;
```

```
} printf("Not Found");
```

Reading
input

```
int ans[50], sno;
```

```
for (i = 0; i < 50; i++)  
{  
    scanf("%d", &ans[i]);  
}
```



```
scanf("%d", &sno);
```

Searching

```
for (i = 0; i < 50; i++)  
{  
    if (sno == ans[i])  
    {  
        printf("Found");  
    }  
}  
exit(0);
```


printf("Not found");

~~int arr[10] = {0};~~

Print array with  without 

Initialize array

int arr[] = {1, 2, 3, 4, 5}; ✓

int arr[5] = {1, 2, 3, 4, 5}; ✓

int arr[3] = {1, 2, 3, 4, 5}; ?

int arr[7] = {1, 2, 3, 4, 5};

Ex: Find frequencies of elements.

2
3
9
3
2
5
7
9
11
10
8
9

2	—
3	—
9	—
5	—
7	—
11	—
10	—
8	—

Q12 Duplicates Vs unique

→ Given list of elements (numbers) show only unique (Remove duplicates)

~~Remove~~
in another array

Remove from the
same array
source itself.