COP3502 8/29/23

1. Array of struct
2. Array of ptr to struct
3. dynamic list struct
4. Insurance example

Announcements

- ✓ 0. LAs
- ✓ 1. No ~~class~~ LECTURE R
- ✓ 2. Quiz Next Tues
- ✓ 3. Extra Video Tonight
- ✓ 4. P1 posted tomorrow

---

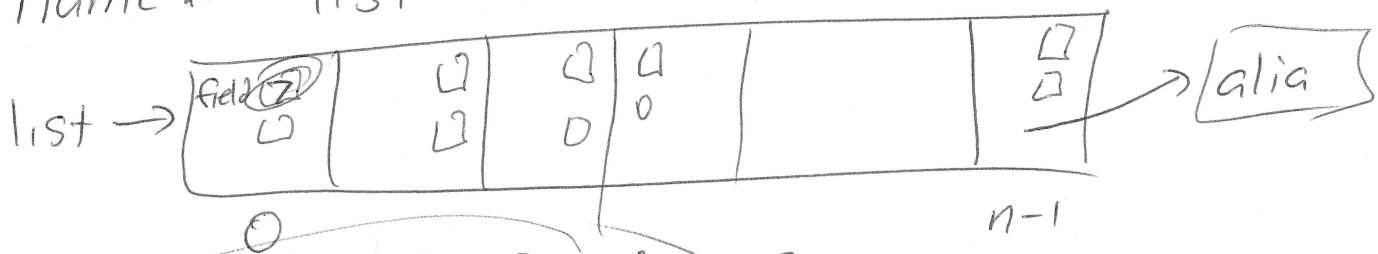
typedef struct name {

    field

} name;

# of slot

```
name* list = malloc(n * sizeof(name));
```




```
list[0].field = 7;
```

how to access member of a struct

```
free(list);
```

Pro

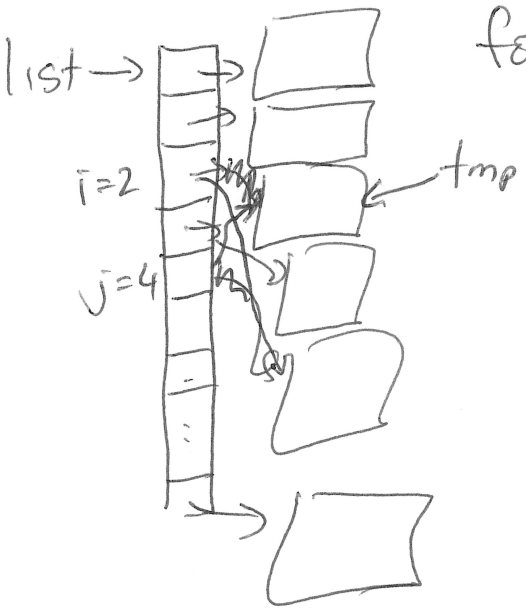
easy

Cons

things like swapping elements are a bit more time consuming

Array of ptr to struct

```
name **  list = calloc(n, sizeof(name*));
```




```
for (i=0; i<n; i++) {
    list[i] = malloc(sizeof(name));
```

Swap

```
name * tmp = list[i];
list[i] =  list[j];
list[j] =  tmp
```

more common → `list[i] -> field = 7;`

`(*list[i]).field = 7;`

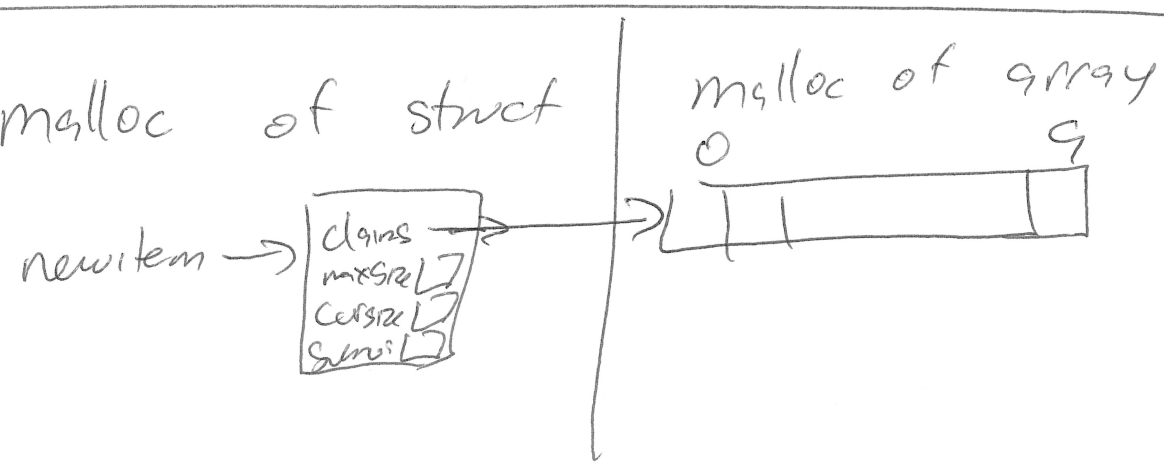
} Diff way access

Dynamic List Structure

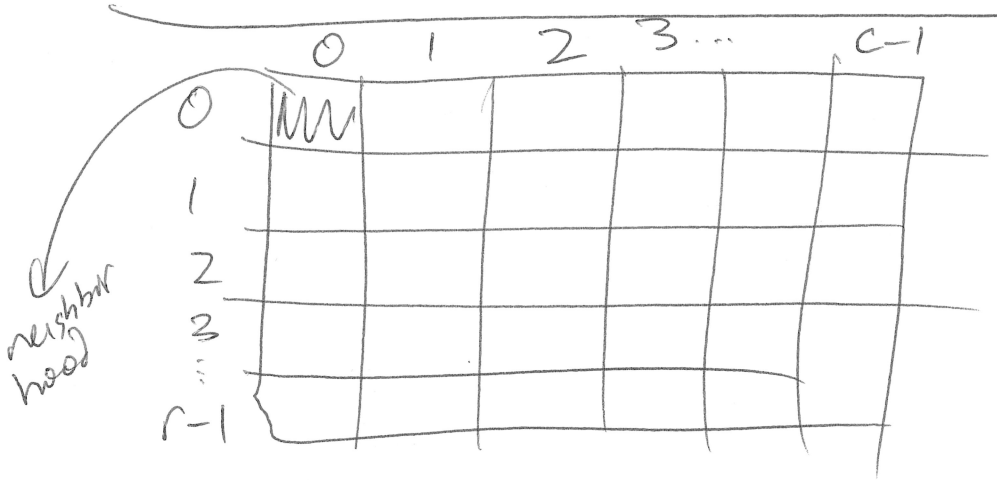
```
typedef struct claimlist {
    int*  claims;
    int   maxsize; // actual size array claims
    int   cursize; // # values currently stored
    long long sumcost;
} claimlist;

claimlist* makeNewClaimList(); // starts @ maxsize 10

void addClaim(claimlist* ptr, int value);
    // add value to end of array
    // update cursize
    // updt sumcost
    // if array full realloc first...
```



Insurance Claims Problem




```
claimlist*** orlando = calloc(r, sizeof(claimlist**));
for (int i=0; i<r; i++) {
    orlando[i] = calloc(c, sizeof(claimlist*));
    for (int j=0; j<c; j++) {
        orlando[i][j] = malloc(sizeof(claimlist));
```

---

r c

$n

1 r_{claim} c_{claim} m_{claim}

2 r_{claim} c_{claim}

Input

num operations

add claim

// print num_{sum}