COP 3502 1/13/26

---

static allocated memory comes from stack (amt known at compile time)

```
int x[100]; // 100 ints → 800 bytes
```

LIMITATIONS:

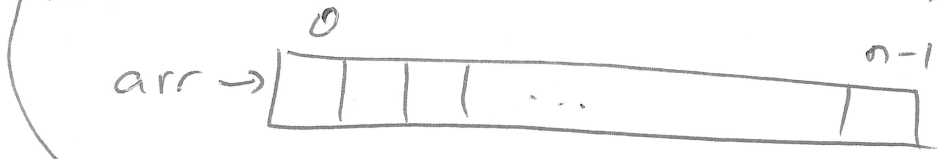
1) what to do if memory requirement is unknown at compile time?

2) stack is small!

## C functions to allow dyn mem alloc

1) malloc
2) calloc
3) realloc

```
int* arr = malloc(n * sizeof(int));
```




effectively gives me a pointer to an array of size n (of int)

Memory comes from heap -

1) larger
2) memory persists after the function within which it's created finishes

~~2D array~~

~~int** arr~~

## Calloc

size of each elem

```
int* arr = calloc( n, sizeof(int) );
```

# elements for array

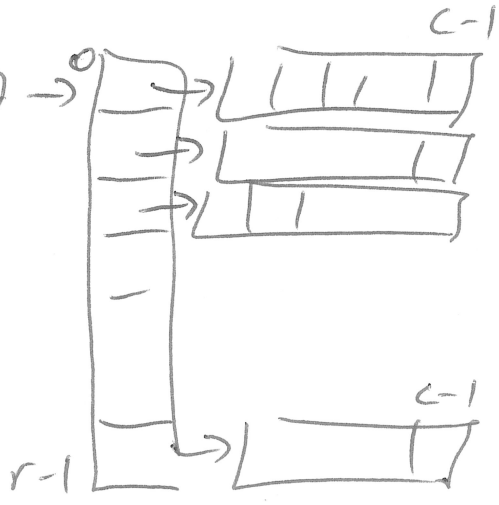
Calloc sets each element to 0.

malloc does NOT!

## 2D array

array of arrays

```
int** grid = calloc( r, sizeof(int*) );
for (int i=0; i<r; i++)
    grid[i] = calloc(c, sizeof(int));
    :
for (int i=0; i<r; i++)
    free(grid[i]);
free(grid);
```



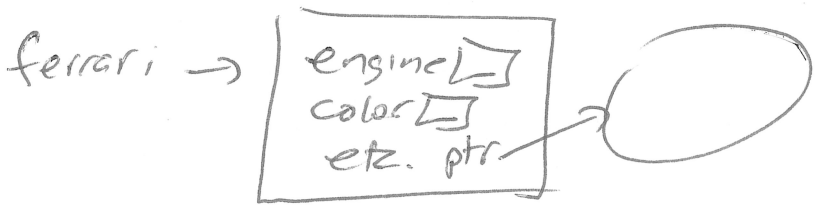
What about structs?

```
Struct car* makeCar( ... ) {

}
```

main

```
Struct car* ferrari = makeCar( ... );
```




(*ferrari).engine  SAME AS

(*) dereferencing  (.) access in struct

ferrari → engine

If the expression to left of the dot is a struct, use a dot.

If this expression is a ptr to struct, use →.

# Options

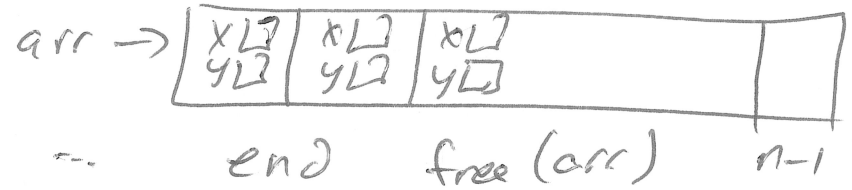
Array of struct

Array of ptr to struct

```
typedef struct pt {
    int x;
    int y;
} pt;
```

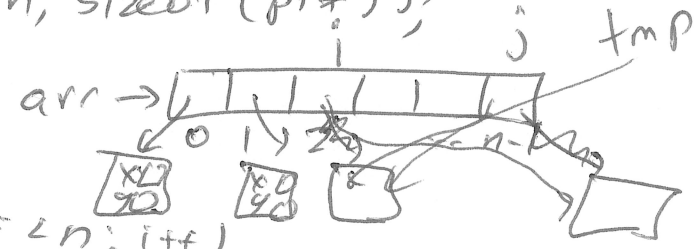
```
pt * arr = calloc(n, sizeof(pt));
```

now arr[0] is of type pt




## OR

```
pt** arr = calloc(n, sizeof(pt*));
```



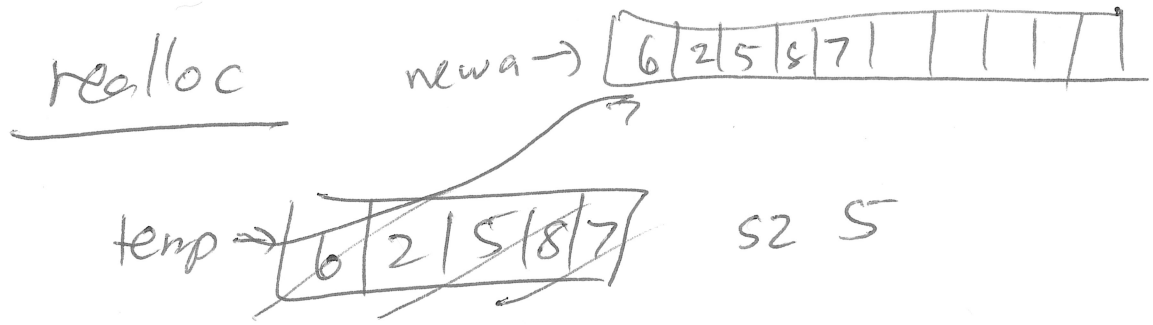

```
for (int i=0; i<n; i++)
    arr[i] = malloc(sizeof(pt));
```

## SWAP

```
pt* tmp = arr[i];
arr[i] = arr[j];
arr[j] = tmp;
```

When you read code, pt** is ambiguous. Could be 2D array of pt, or 1D array of ptr to pt.

# realloc




How do I allocate more space?

## Slow way

```
int* newarray = calloc(2*n, sizeof(int));
for (int i = 0; i<n; i++)
    newarray[i] = temp[i];
free(temp);
temp = newarray
```

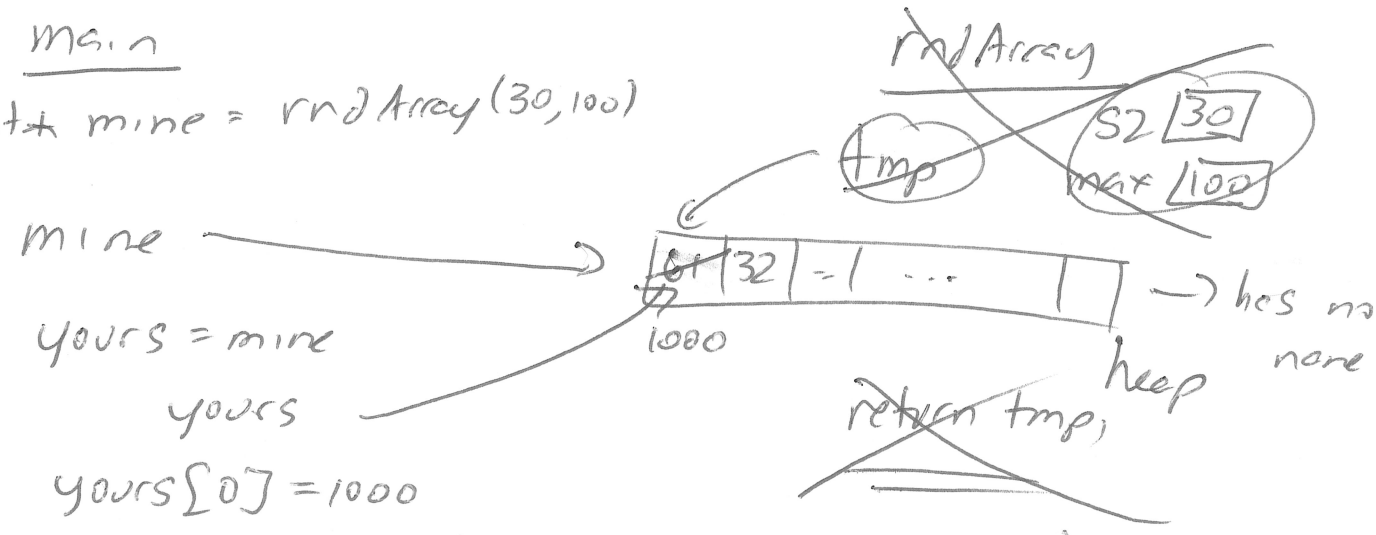
```
temp = realloc( temp, 2*n+ sizeof(int));
```

- temp: ptr you want to pt to the new mem
- temp: ptr want realloc
- 2*n+ sizeof(int): new size

# Picture rndarray

main

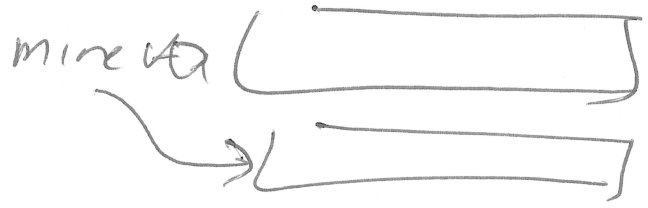
```
int* mine = rndArray(30,100)
```




```
int* yours = mine
yours[0] = 1000
printf("%d %d\n", mine[0], yours[0]);
free(mine)
```


If I try free(yours), it ~~crashes~~!
CRASHES!

# Memory leak




```
mine = malloc( ... )
```

each an every malloc has an equal and opposite free.