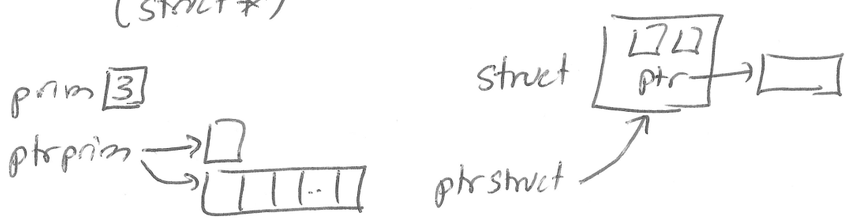
1/20/26 COP3502

Dynamic Memory Allocation Summary

(int, char, double) (int*) (struct)

type: primitive, prim pointers, user def type
used def type ptr
(struct*)

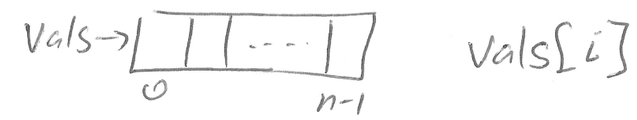


2 key idees: memory persists after the function its allocated ends, access to more memory (it's from heap not stack)

Use cases

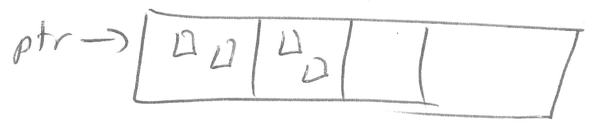
1) large array of prim

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



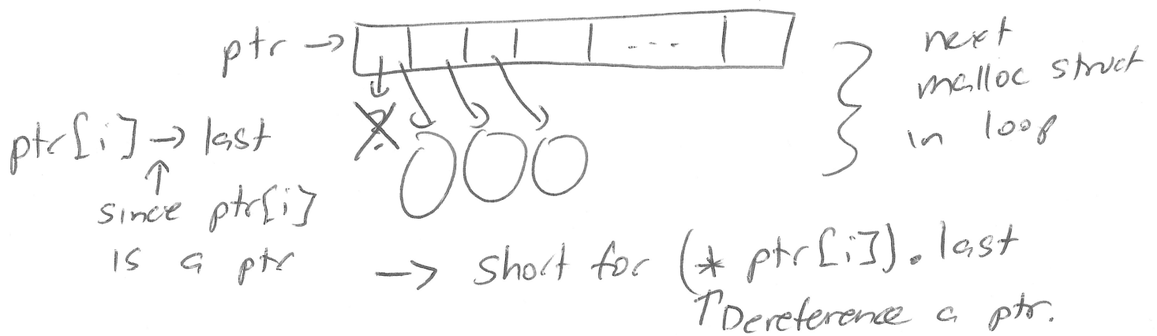
2) large array of struct

```
struct name* ptr = calloc(n, sizeof(struct name));
```



3) large array of ptr to struct

struct name ** ptr = calloc (n, sizeof (struct name *));



→ short for (* ptr[i]).last
↑ Dereference a ptr.

Strings

char first[20]; → read into first
Initially read into larger statically sized array.

int len = strlen(first);

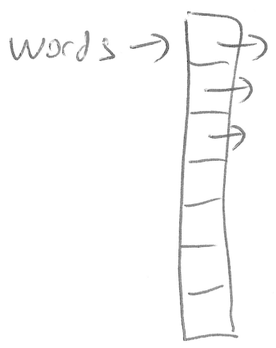
char* firstName = malloc (~~char~~ (len+1) * sizeof(char));

strcpy (firstName, first);

↑ Why? '\0' for null char

Array of Strings

char** words = calloc (numWords, sizeof(char*));




```
for (int i = 0; i < numWords; i++) {
    words[i] = calloc (20, sizeof(char));
```

OR use len after reading

Algorithm Analysis Part of Class

Goal: Helps us make predictions about the time + memory usage of algorithms we design.

Big Oh Notation

Summations

Recurrence Relations → after Exam 1 (not sure yet!)

Code Segment to analyze

```
for (int i = 0; i<n; i++)
    for (int j=0; j<n; j++)
        sum[i][j] += mat[i][j];
```

How long to execute?

- How many simple steps? → Defining this is difficult!
- Impossible due to architecture differences!

We won't care about constant factors, so $3n^2$ simples steps and $5n^2$ simple steps will be treated the same.

If I write $f(n) \in O(g(n))$

- $f(n)$: function in n
- $\in$: element of
- $O(g(n))$: Big Oh of $g(n)$ is a set of functions!

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, c is a constant

$10n \in O(n)$ → ~~Don~~

$n \in O(10n)$ Convention is to remove constants here

$5 \times 2^n \in O(2^n)$

$6n^3 - 4n^2 + 100n \in O(n^3)$

$=$

"If I say a function equals $O(n^2)$, I mean it's no bigger than a constant times n^2."

$3n \in O(n^2)$ is true!

Not a ~~tight bound.~~ → no smaller function would satisfy the claim

$3n \in O(n^n)$ is also true!

$f(n) \in \Omega(g(n))$ means that

$\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0$. Ω is lower bound

$f(n) \in \Theta(g(n))$ [$f(n) \in \Omega(g(n))$ ∧ $f(n) \in O(g(n))$

$\lim_{n \to \infty} \frac{f(n)}{g(n)} = C$, $C \in$ Const $C > 0$.

```
for (int i=0; i<n; i++)
    for (int j=0; j<n; j++)
        sum++;
```

$O(n^2)$

~~loop repeat~~ sum++; repeats n^2 ~~n~~ times

```
for (int i=0; i<strlen(s); i++)
    freq[s[i]-'a']++;
```

let n = strlen(s)

What's the run time?

Key: strlen is a function that has a loop in it, with run time $O(n)$.

→ repeats n times $O(n^2)$.

$O(n)$ ← `int n = strlen(s);`

`for (int i=0; i<n; i++)`

$O(n)$ `freq[s[i]-'a']++;`

$O(n^{2n})$

Simplification: if a function runs in $O(f(n))$ time, we'll let $T(n) = cf(n)$ be the run time of that function on input size n, for some constant c.

$O(n^2)$ alg $n = 5\times10^4$ run time 1.1 sec

What will run time be for $n = 10^5$

Let $T(n) = c\cdot n^2$ be ~~fun~~ run time of alg.

$$T(5\times10^4) = c\cdot(5\times10^4)^2 = 1.1 \text{ sec} \qquad \frac{(2x)^2}{x^2} = 4$$

$$c = \frac{1.1 \text{ sec}}{25\times10^8}$$

$$T(10^5) = c\cdot(10^5)^2 = \frac{1.1 \text{ sec}}{25\times10^8} \times 10^{10} = 4.4 \text{ sec}$$