Programming in C

Declare variables in advance

Collection of functions

Assignment, if, while, for

ret value type

```
int factorial(int n){
  int val;
   if (n == 0) {
        return(1);
   }else{
      val = n * factorial(n-1);
      return(n *
              factorial(
      return(val);   n-1));
   }
}
```

```
int factorial2(int n){
    int val,i;
    val=1;
    for(i=1; i<=n; i=i+1){
        val = val*i;
    }
    return(val);
}
```

Swap?

```
for(i=1; i<=n; i=i+1,
               val=val*i);
```

Will it work always?

In some situations, C promises a left to right order

```
if (c1 && c2) {..}
```

c1 evaluated before c2

"short cut" : c1 is false ⇒ c2 is not evaluated

Array with n elements

```
if (i<n && a[i] <= max) {..}
```

C & ~~Python~~ Java & .. allow

$x{+}{+}$

as an abbreviation for $x = x+1$

`i += 2;`
for
`i = i+2;`
also `x /= 3` etc.

`a[i++] = 7;` `a[++i] = 7;`

Python `i++` ✓ `a[i++]`

How arguments are passed to functions?

Python Mutable vs Immutable

$\downarrow$ Mutable: Updates are reflected in the original value

$\downarrow$ Immutable: Updating parameter in fn leaves original value undisturbed

More standard way of describing parameter passing

Value is copied — Call By Value

Get a "reference" to the original value — Call By Reference

Python: Immutable values : Call By Value
Mutable : Call By Reference

In C, all parameters are passed by <u>Value</u>

But this is too weak, in general

Sort in place?

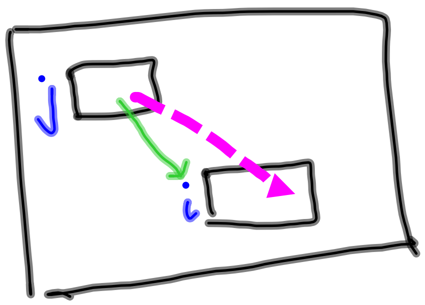
Swap two integers?

Solution: Have variables whose values are "references" — i.e. memory addresses, or pointers int i;

&i refers to the address of i

if j is a variable that holds an address

*j refers to the location pointed to



j = &i

*j is same as i

How to declare that j is a reference to int?

```
int *j;
```

What j refers to is an int

How do we swap two ints?

```
int x,y;
  :
swap(x,y); X  swap(&x,&y);
```

```
void swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}   return;
```

Functions can return pointers:

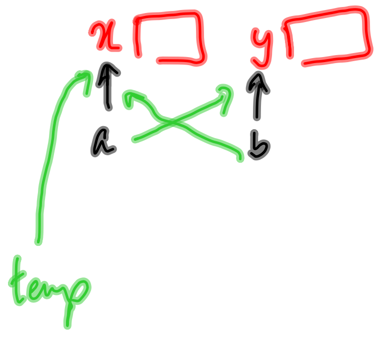
```
int *f( ... ) { .. }
```

Still call by value:

```
void swap2(int *a, int *b){
    int *temp;
    temp = a;
    a = b;
    b = temp
}
```

No effect!

swap(&x, &y)



Input & output

x = input() returns a string in Python

In C: input fn takes variables to be read as arguments — needs call by reference to update values

Some initial mantras:

```
#include <stdio.h>
```

Input	scanf
Output	printf

f stands for formatted

```
printf("The value of _ is _", x, y);
```

describe how the value is printed
"format specifier"

Format specifiers: %d "digit" ⇒ integer

%f float

```
printf("The factorial of %d is %d", n, m);
```

print value of n as an integer

```
scanf ("%d %d", &m, &n);
```

Reed into m & n as integers