9/29/20 Notes

Control structures and flow control

Boolean Algebra
- set mathematics on logical foundation
- Bool logic has true and false
- 3 basic operators
  - ∧ called and (& && in C)
  - ∨ disjunction → (|| in C)

And (conjunction)
- associative identities
- ~~DeMorgan's Law~~ distributable
- $A \wedge 1 = A$
- $A \wedge 0 = 0$
- $A \wedge A = A$

Or (inclusive / disjunction)
- $A \vee (B \vee C) = (A \vee B) \vee C$ - associative
- ~~DeMorgan's Law~~ distributive
- $A \vee 0 = A$
- $A \vee 1 = 1$
- $A \vee A = A$

Not (negation)
- DeMorgan:
  $\neg(A \wedge B) = \neg A \wedge \neg B$

Exclusive -or (XOR)
- $A \oplus B = (A \vee B) \wedge \neg(A \wedge B)$
- $A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$
- Algebraic properties
  - $A \oplus A = 0$
  - $A \oplus 0 = A$
  - $A \oplus 1 = \neg A$
  - $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

first thing or second thing but not both

⊕ (XOR)	T	F
T	F	T
F	T	F

Scanned with CamScanner

Logical Operators
- < less
- <= less than equal
- == equal
- > greater
- >= greater than equal

True and False in C
- In C, zero (0) is false
- All that is not false is true
- Logical exp. have type int
- You can have true and false if you:
  #include <stdbool.h>

if ()
- if executes next statement if Bool exp. is true
- even though { } are not required for a single statement, always use them
- Why?
  - it avoids errors when adding statements

if () { ... } else { ... }

Nested if ()

if () { ... } else if () { ... } else { ... }

Boolean Operators
- && = and
- || = or
- ~ = not

Scanned with CamScanner

Short circuit evaluation
- false && anything is false
- true || anything is true
- stop evaluating as soon we know the result
- suppose we evaluate exp:
  - we would be dividing by zero
- we would follow a null pointer

```
switch (exp) {
case 0:
    msg = "zero";
    break;
case 1:
    msg.
    break;
default:
    msg.
}
```

switch()
- allows you to select among a fixed set of alts.
- h

equivalent goto code

What's crazy w/ goto?
- it's not about how you got there
- you could goto middle of
  - an if statement
  - switch statement or
  - a loop
- It is not even clear what that would mean.

What is a loop?
- repeats a seq. of code,
- progs. spend vast majority of their execution time in loops

- we will focus on loops: while, for, and do-while
- you can also create loops w/ the goto but don't.

while()
- top-test loop.
  - test is evaluated before entering loop.
- executes statements as long as bool cond. remains true.
- ..

Equivalent goto code.
- you can implement it w/ goto statements
- not recommended!

for()
- also top test loop.
- three parts!
  - initialization
  - test, and
  - increment all together
- By convention they are related, but not required to be related.

Scanned with CamScanner

equivalent while()
- This is the ~~last~~ equivalent while statement
- the while statement is complete
  - which means you can implement any loop

do { } while
- bottom-test loop
- used when you want to perform the statement at least once
- continues to execute the enclosed statement as long as the Bool cond. remains true

Infinite loops
- execute forever
- the one you choose is a matter of style not of substance.
- How do you ever escape?
  - break;

break
- immediately exits enclosing loop

Factorial Example
- $n! = n * (n-1)!$

Assign 2: you don't need to do factorials if.

When can I use goto?
- one place: non-local error handling
- this is when an exceptional condition that you cannot handle occurs
- It is not pretty.

Caution:
- You may have times when you want to skip remainder of a loop
- but please use sparingly

Scanned with CamScanner