Statements

Last updated 6/16/23

These slides introduce statements in C

Statements

- Statement
  - Causes the processor to do something
  - 11 types of statements
    - Null
    - Expression
    - Return
    - Compound
    - Conditional
    - Labeled
    - Switch
    - Iterative
    - Break
    - Continue
    - Goto

Statements

- Null Statement
  - Causes nothing to happen

```
;

while(1){
  ;
}
```

Statements

- Expression Statement
  - An expression with a semi-colon added
  - Causes the processor to evaluate the expression
  - Causes the processor to complete any side effects
  - Processor discards the expression

  - Special note: the side effect of the assignment operator is to store a value into a variable

Statements

- Expression Statement - example

aa = 5;

; causes the expression to be evaluated → 5

side effect of the assignment (=) is aa holds the value 5

aa = bb = 5;

same precedence, operate R to L

bb = 5

value is 5, side effect is bb holds the value 5

aa = 5

value is 5 (value of BB), side effect is aa holds the value 5

note: this equals 5 (the value), not bb

Statements

- Expression Statement - example

```
ab = 5;
```

value is 5

side effect is ab takes the value 5

```
ab++;
```

value is 5

side effect is ab takes the value 6

the value (5) is then discarded (not assigned to anything)

Statements

- Return Statement
  - Terminates all functions (including main)

```
int main(void) {
…
  return 1;
}
```

Statements

- Compound Statement
  - Block of code containing zero or more statements
  - These statements are considered a single entity
  - Defined by {…}

```
int main(void) {
…                          // multiple statements
  return 1;
}
```

Statements

- Pre-processor commands vs statements

```
#define int_rate 0.25        // pre-processor command

#define int_rate 0.25;       // error

payment = int_rate * balance;
```

creates a compiler error at the “payment =” line

but you never see the expansion

payment = 0.25; * balance;

very difficult to catch