CS 131

Building Blocks: Expressions and Types

About Expressions

- Your programs are a sequence of <u>expressions</u>
- <u>Expressions</u> have four important features:
  - VALUE : “What it *means*”
  - SIDE EFFECTS : “What it *does*”
  - TYPE: “What values can *can it be*"
  - SEMANTICS: “What *kind* of expression is it?
- As you encounter new expressions and work with old ones, practice asking “What does this *mean*? What does this *do*? What *can it be*? What *kind* of expression is it?”

Types, or
What an expression can be

- **`int`** (integer) a whole number like 4,-2,0,96
- **`float`** (floating point number) a number with decimal places like 3.14, 6.78, -3.0, and 0.0
- **`bool`** (boolean) [needs `#include <stdbool.h>`] either `true` or `false`
- **`void`** (the "no-values" type)
- **`char`** (character) one(1) symbol that can be printed on the screen. Letters ('a', 'R') digits ('2', '0') marks ('!', '$') spaces (' '), tabs ('\t'), newlines ('\n') and others.
- **`char*`** (string) a sequence of `char`s, such as "Hi" (2 chars), "" (0 chars), "4" (1 char), "Hello" (5 chars)
- Others we'll see later

Kinds of Expressions

http://en.wikipedia.org/wiki/Operators_in_C_and_C++

- Literals (hardcoded)
- Function calls
- Arithmetic (one or two numbers in, one number out)
- Boolean (one or two booleans in, one boolean out)
- Comparison (two comparable numbers in, one bool out)
- Assignment (mostly for side-effect, to store data)
- Variable (retrieving data that was stored)

Kinds of Expressions
Literals

- integer literals
  3, 1, 9, 0, 159487
- character literals
  ‘3’, ‘1’, ‘9’, ‘a’, ‘z’, ‘\n’
- string literals
  “Howdy”, “”, “z”
- boolean literals
  true, false

Kinds of Expressions
Function Calls

- Argument-less functions
  makeABeep()
- Functions with arguments
  printf(“Hi”) (1 argument)
  printf(“Hello %s World”, “cruel”) (2 args)
  printf(“His truck is a %d by %d\n”, 4,4) (3 args)

Kinds of Expressions Arithmetic

- Binary arithmetic operators + - * / %
  3+4
  7/7
  5+printf(“ex”)
  (3+4)/(5+printf(“ex”))
  4*3+2/1%5 AKA (4*3)+((2/1)%5)

- Unary arithmetic operators + -
  -4
  +3

Kinds of Expressions Arithmetic

- Binary arithmetic operators + - * / %
  + “Plus”, “Addition”
  - “Minus”, “Subtraction”
  * “Times”, “Multiplication”
  / “Divided by”, “Integer division”
  % “Mod”, “Integer remainder”
- Unary arithmetic operators + -
  - “Negative”, “Additive negation”
  + “Positive”

Kinds of Expressions Comparison/Relational

- Binary comparison operators < <= == >= > !=
  3<4
  5==4
  5==printf(“Hello”)
  3>=(2*2)

Kinds of Expressions Comparison/Relational

- Binary comparison operators < <= == >= > !=
  < “is less than?”
  <= “is less than or equal to?”
  == “is equal to?”
  >= “is greater than or equal to?”
  > “is greater than?”
  != “is not equal to?”
- Don’t forget to say the “is” and the “?”

Kinds of Expressions Logical/Boolean

- Binary boolean operators && ||
  true && true
  false || (3==3)

- Unary boolean operator !
  !true
  !(3>4)

Kinds of Expressions Logical/Boolean

- Binary boolean operators && ||
  && “and”, “both ... and ... are true”
  || “or”, “either ... or ... (or both) are true”

- Unary boolean operator !
  ! “not”, “Logical Negation”

Kinds of Expressions Assignment

- There's one main assignment operator: =

  age=5
  initial= 'w'
  distance = 3.4
  title = "Poohbah, Grand"
  gotMilk = false
  lettersPrinted = printf("Something")

- Others (+=, *=, %=, etc) are shortcuts.

Kinds of Expressions Assignment

- There's one main assignment operator: =
  = "gets", "variable assignment"
  Do **NOT** read this as "Equals". If *I* do, call me on it!

- Others (+=, *=, %=, ++, --, etc) are shortcuts.
  += "increment by"
  *= "multiply by"
  ++ "increment (by 1)
  -- "decrement (by 1)

Kinds of Expressions Variable

- To use the value in a variable, write its name where you'd use an expression of the same type:

  age+**lettersPrinted**

  printf("Herbert %c Jones, %s.\n",**initial**, **title**)

  if(**gotMilk**) printf("Moo\n");

  if(**distance**<26.2) printf("Not so far.\n");

Exercise: Fill in the table

- Imagine the <u>expression example table</u> had blanks in it... which blanks could you fill in? Which ones don’t make sense?