C Syntax

What the Compiler Expects

What's in a C program

Pre-Processing

- Any line that starts with "#" is a "pre-processor directive"
- Pre-processor consumes that entire line
  - Possibly replacing it with other C code
  - For example "#include <stdio.h>" replaces the #include line with the contents of the stdio.h file.

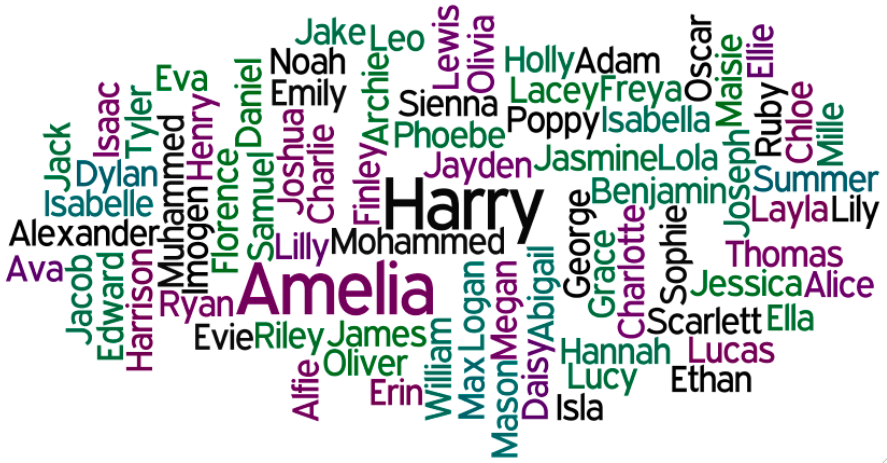
White Space

- Needed to separate tokens
  - e.g. "then break" are two contiguous keywords, but "thenbreak" is a single identifier
- Otherwise, ignored
  - "then break" is the same as "then break" is the same as "then break"
- Enables programmer to choose formatting preferences

Keywords

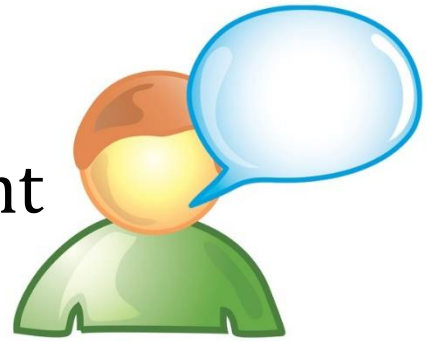
asm auto break case char const continue default do double else enum extern float for fortran goto if int long register return short signed sizeof static struct switch typedef union unsigned void volatile while

Identifiers



- Function, parameter, and variable names
- Must start with a letter or an underscore
  - Underscores usually avoided
- May not contain white space
- After the first letter, can be any number, letter, or underscore
- Identifiers are case sensitive
  - "polyArea" is different from "PolyArea"
- Choose names that are descriptive, and easy to type
  - "Be4aTgh9_fr37200aBy" is probably not a good choice

Comments



- Anything starting with "/*" up to the next "*/" is a comment
  - /* */ comments do NOT nest
  - /* This is not a /* comment */ within a comment */

```
                                    ^
```

syntax error…. within not declared

  - /* comments may span lines
    and continue on to the next line */
- // causes a comment to the end of the line (white space matters)
- Use comments to help reader understand what the code does!
  - No need to comment the obvious: a=a+3; // add three to a
- Comments ignored by compiler

Block Comments

```
/* ----------------------------------------

I like this style of comment because I can add or remove
lines from the comment without special comment reformat
intervention.

Besides, it looks clean.

----------------------------------------*/
```

C Statements

- A statement is a list of tokens that ends with a semi-colon
  - a=a+3;
  - int c=7;
  - int cent_to_far(int c);
- A C statement may span more than one line in the file
  - int MyLongFunctionNamedFunctionThatTakesLotsOfArguments(
    int arg1, int arg2, int arg3, int arg4,
    int arg5, int arg6, int arg7, int arg8);
- There may be more than one statement on one line in the file
  - int a=5; a=a+6; int b; b=cent_to_far(a);

C Blocks

- A block of statements is a list of statements, surrounded by { and }
  - { - Left curly brace
  - } – Right curly brace
- A block can be used anywhere a statement can be used
- Blocks of statements can be nested
  - { statement1; { statement2; statement3; } statement4; }

Function Definition

- Function Signature,
- Followed by Embodiment Block of statements

C File

- Some stuff up front…
  - #include pre-processor directives
  - Function Declarations
  - Global Variable Declarations (more in the future)

•Function Definitions

Statements in Functions

- Variable Declarations (Lecture 06)
- Assignment/"Expression" Statements (Lecture 07/08)
- Control Flow Statements (Lecture 09/10/11)

Resources

- Programming in C, Chapter 2
- WikiPedia: C Syntax (https://en.wikipedia.org/wiki/C_syntax)