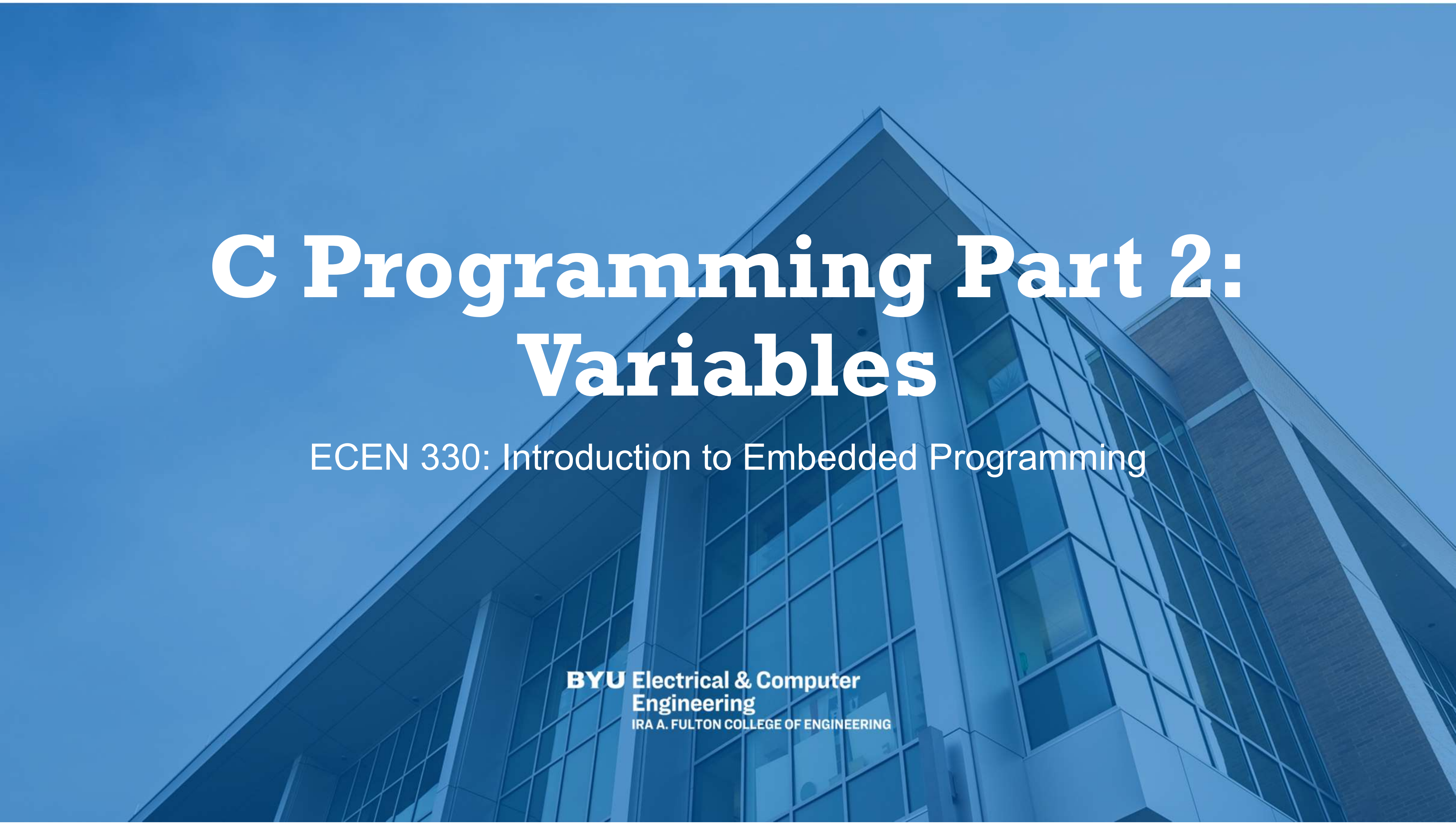


C Programming Part 2: Variables

ECEN 330: Introduction to Embedded Programming

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

What are variables?

Variables are **human-readable names for the computer's memory addresses** used by a running program.

Declarations

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

- Variables must be declared before use

- A declaration specifies a type and a list of one or more variables

```
int lower, upper, step;
char c, line[1000];
```

- Variables can be initialized

```
char esc = '\\';
int i = 0;
int limit = MAXLINE+1;
float eps = 1.0e-5;
```

Global variables:
    initialized to zero by default
    initializer must be a constant expression
    initialized once at beginning of program
Automatic (Local) variables:
    undefined by default
    initializer may be any expression
    initialized each time function entered

Variable Assignments

- Variable assignments return the value of the assignment:

```
anumber = anothernumber = yetanothernumber = 8;
```

- Sometimes you will see this used within IF statements:

```
if ( (x = getchar()) == ‘\n’ )
```

Variable Names

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

- Names are made up of letters and digits
- First character must be a letter
- Underscore "_" counts as a letter
  - Don't begin variable names with underscore since library routines often use such names
- Upper and lower case letters are distinct

Data Types and Sizes

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

• `char`	a single byte, capable of holding one character
• `int`	an integer, typically the natural size of machine

Qualifiers that can be applied to these basic types

• `short`	at least 16 bits, not longer than `int`
• `long`	at least 32 bits, not shorter than `int`

```
short int sh;
long int counter;
```

The word `int` can be omitted in such declarations

When you need to know the exact size, use <stdint.h>:

- `int8_t`
- `uint8_t`
- `int16_t`
- `uint16_t`
- `int32_t`
- `uint32_t`
- `int64_t`
- `uint64_t`

Floating Point

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

- `float` single-precision floating point (32 bits)
- `double` double-precision floating point (64 bits)

- How are the bits used? (Similar to representing scientific notation)
  - 1 sign bit
  - Several exponent bits
  - Several significand/mantissa bits

- Floating-point numbers are inexact, but have high range

Literals

- Integer

  1234 (int)
  1234L (long)
  1234U (unsigned)
  1234UL (unsigned long)

  037 (int in octal)
  0x1F (int in hex)
  0x1FUL (unsigned long in hex)

- Floating-point

  123.4 (double)
  1.234e2 (double)
  1e-2 (double)

  123.4F (float)
  1.234e2F (float)
  1e-2F (float)

Suffixes can be upper or lower case
(same for x and e)

Character Constants

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

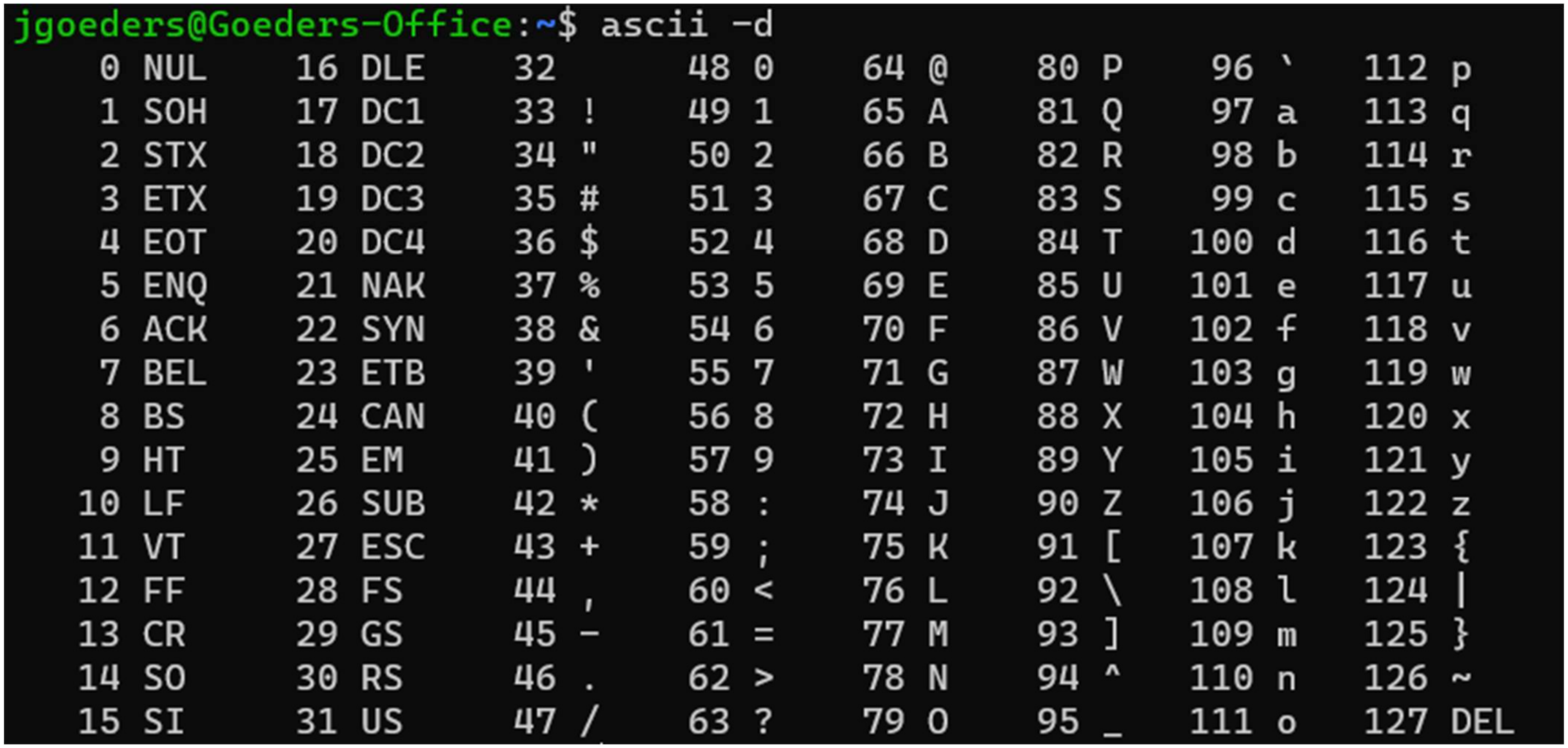
- Written as one character within single quotes
  `'x'`
- Type is an **integer**
- Value is ASCII encoding
  `'0'` has the value 48

- Escape sequences

`\0`	null character
`\n`	newline
`\r`	carriage return
`\t`	horizontal tab

ASCII Values

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING



```
jgoeders@Goeders-Office:~$ ascii -d
    0 NUL    16 DLE    32      48 0    64 @    80 P     96 `   112 p
    1 SOH    17 DC1    33 !    49 1    65 A    81 Q     97 a   113 q
    2 STX    18 DC2    34 "    50 2    66 B    82 R     98 b   114 r
    3 ETX    19 DC3    35 #    51 3    67 C    83 S     99 c   115 s
    4 EOT    20 DC4    36 $    52 4    68 D    84 T    100 d   116 t
    5 ENQ    21 NAK    37 %    53 5    69 E    85 U    101 e   117 u
    6 ACK    22 SYN    38 &    54 6    70 F    86 V    102 f   118 v
    7 BEL    23 ETB    39 '    55 7    71 G    87 W    103 g   119 w
    8 BS     24 CAN    40 (    56 8    72 H    88 X    104 h   120 x
    9 HT     25 EM     41 )    57 9    73 I    89 Y    105 i   121 y
   10 LF     26 SUB    42 *    58 :    74 J    90 Z    106 j   122 z
   11 VT     27 ESC    43 +    59 ;    75 K    91 [    107 k   123 {
   12 FF     28 FS     44 ,    60 <    76 L    92 \    108 l   124 |
   13 CR     29 GS     45 -    61 =    77 M    93 ]    109 m   125 }
   14 SO     30 RS     46 .    62 >    78 N    94 ^    110 n   126 ~
   15 SI     31 US     47 /    63 ?    79 O    95 _    111 o   127 DEL
```

String Literals

- Sequence of zero or more characters in double quotes

```
"I am a string" /* quotes are not part of string */
```

- A string is an array of characters with a `'\0'` at the end

`‘h’`	`‘e’`	`‘l’`	`‘l’`	`‘o’`	`‘\0’`

- Be careful to distinguish between `'x' and "x"`

- `strlen(s)` returns the length of the string `s` excluding `'\0'`

Enumeration Constants

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

- An enumeration is a list of constant integer values

```
enum boolean { NO, YES }; /* NO = 0, YES = 1 */
```

- The first name in an `enum` has value 0, the next 1, and so on

- Values can be specified

```
enum months { JAN = 1, FEB, MAR, APR, MAY, JUN,
              JUL, AUG, SEP, OCT, NOV, DEC };
```

- Unspecified values continue the progression from the last specified value

```
enum states { S_IDLE, S_BUSY = 4, S_DONE };
           /*      0,        4          5 */
```

Constants using const & #define

The `const` qualifier specifies that a variable's value will not be changed (can not write to it)

```
const double e = 2.71828182845905;
const char msg[] = "warning: ";
int strlen(const char[]);
```

#define A B

- Before compilation, any “A” text is replaced with “B”

Tradeoffs

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

const variable

- Has type, scope
- May or may not occupy memory space (compiler dependent)
- Not a true constant – can't be used as case in switch or to size array

```
const int size = 10;
int size2 = size;
```

(the size2 global variable won't compile)

```
error: initializer element is not constant
```

#define value

- Relies on simple text substitution by preprocessor (before compilation)
- Can result in subtle bugs:

```
#define OFFSET 5
#define SIZE OFFSET+3
…
char buf[SIZE * 2];
```

```
Expands to
5 + 3 * 2 = 11
```

The `static` keyword

1. Static Local Variable
   - Variable maintains value across function invocations

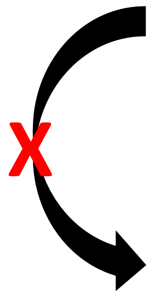
```
void foo() {
  static int x = 0;
  x++;
  printf("%d\n", x);
}
```

..and the other completely different meaning…

2. Scope limited to its own .c file
   - Global variables, or
   - Functions

drawingSM.c

```
static cnt = 0;
static void foo() {
   …
}
```



controlSM.c

```
static cnt = 3;
static int foo() {
   …
}
```

Extern

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

`extern` keyword indicates that a function or variable is defined in a different file.

Functions:

- `extern void foo();`
  - Indicates that foo is in another file.
  - This is optional. <u>So don't do it </u>– it just litters your code

Variables:

- `extern int x;`
  - Indicates x is a global variable in another file, and <u>ensures space is not allocated in this file</u>.
  - Without extern, you can still link to variables in other files (even unintentionally)
  - Best approach:
    - If you mean to link to a variable in another file, use extern
    - If you only want the global to be accessed in the current file, use static.