CprE 288 – Introduction to Embedded Systems (C: History, Variables, Arrays, and Strings) Instructors: Dr. Zhao Zhang (Sections A, B, C, D, E) Dr. Phillip Jones (Sections F, G, J)

• Announcements • C History • Intro to C • Variables • Arrays & Strings http://class.ecc.lastate.edu/cpre288

Announcements

- Labs start this week
 - Lab safety training
 - Find your partners, diversity is encouraged
 - Be careful with the Blue Box, don't turn off the power
- Homework 1 is due on Thursday turn in a typed paper copy in class.

http://class.ece.iastate.edu/cpre288

HISTORY OF C

History of C

C was developed in parallel with UNIX

- Martin Richards wrote BCPL in mid 1960s
- Ken Thompson wrote B in 1970
- Dennis Ritchie designed most creative parts of C in 1972
- C is used to re-write UNIX in 1973
- Dennis Ritchie and Brian Kernighan wrote "The C Programming Language" in 1978
- C was standardized during 1983-1988 by ANSI

http://class.ece.iastate.edu/core288

History of C

C and its predecessors were designed as **system programming languages**

- BCPL needs to be compiled on a DEC PDP-7 machine with 8K 18-bit words
- $-\,$ B was used to write utility programs on a DEC PDP-11 with 24KB memory running UNIX
- C was used to re-write that UNIX on the same machine

It has to be simple!

http://class.ece.iastate.edu/cpre288



Compare C and Java/C++ C is a procedural language No classes or objects "Function" is the building block C philosophy As simple as possible Uses a minimum set of language constructs

void main() { while (1); // do forever... } • Most embedded programs run forever

```
#include <stdio.h>

void main()
{
    printf("hello, world\n");
}

To build and run on a Linux/unix machine:
$ gcc -o helloworld helloworld.c
$ ./helloworld
hello, world
```

```
; A semicolon marks the end of an expression; a C statement is an expression ended with a semicolon

{} Braces mark a code block

// or /* ... */ Comments
```

```
Expression and Statement

Which of the follow are valid C statements?

a = a + b;

a;

a + b;

10 + 20;

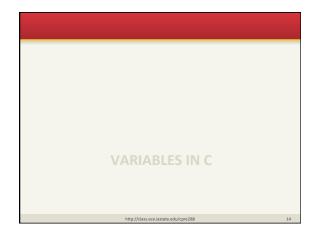
a = (b = c);

;
```

Expression and Statement

Which of the following code segments works as intended?

```
// sum up all elements in an array
for (i = 0, sum = 0; i < N; i++);
  sum += X[i];
// if flag is set, print a message
if (flag = 1)
  print ("flag has been set");
// enter an idle loop
While (1)
```



Variables

- · Variables are the primary mechanism for storing data to be processed by your program
- · Naming rules are similar to Java
- Examples:
 - area, graph, distance, file1, file2, height, wheel_right
- The underscore is the only punctuation mark allowed
- · Must start with a letter or underscore, no digit
- · Case sensitive
 - MyVariable is different from myvariable

Variables

- Must not be a reserved keyword (next slide)
- Good practice: use descriptive variable names
 - Good names: height, input_file, area
 - Bad names: h, if, a
- Exception: names of iterators in loops
 - Common names for iterators: i, j, k, x, y, z
- Rule of thumb: Always code as though the person maintaining your code knows where you sleep... and has anger management issues.

Reserved Words: Primitive Data Types

- char short
- break
- auto
- case
- const

- int
- continue
- extern

- long
- default

- double
- do
- register

- Float
- signed

- else
- static unsigned volatile

sizeof

- enum

- for
- goto
- struct union typedef
- if return
- switch
- while void

Variables

• Like Java, a variable must be declared by specifying the variable's **name** and the **type** of information that it will

> variable name data type int total; int count, temp, result;

Multiple variables can be created in one declaration

Variables

- A variable can be given an initial value in the declaration
- If no initial value is given, do not assume the default value is 0

```
int sum = 0;
int base = 32, max = 149;
int k, i;
for (i = 0; i < 10; i++) {
    k = k + 1;
}</pre>
```

http://class.ece.iastate.edu/cpre288

Name	Number of Bytes sizeof()	Range	
char	1	-128 to 127	
signed char	1	-128 to 127	
unsigned char	1	0 to 255	
short	2	-32,768 to 32,767	
unsigned short	2	0 to 65,535	
int	Varies by platform	Varies by platform	
int (on ATmega 128)	2	-32,768 to 32,767	
(pointer)	Varies by platform	Varies by platform	
(pointer on ATmega 128)	2	Address Space	
, ,	: char, short, int, long, for primitive types is signo		


```
variables: Size
unsigned char my_number = 255;
unsinged char my_number_too_big = 257;

• my_number in:
    Binary: 0b1111 1111
    Decimal: 255
• my_number_too_big in:
    Binary: 0b0000 0001
    Decimal: 1
```

```
void main()
{
  int num_apples, num_oranges = 0;
  int num_fruits = 0;

  num_apples = 5;
  num_oranges = 4;
  num_fruits = num_apples + num_oranges;
}
```



Sequence of a specific variable type stored in memory Zero-indexed (starts at zero rather than one) Define an array as Type VariableName [ArraySize]; Example: int my_array[100] Last element is found at N-1 location Curly brackets can be used to initialize the array

Arrays in C

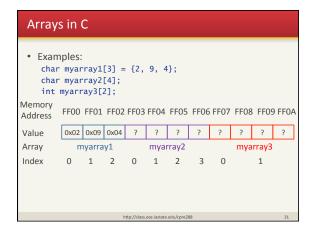
```
Arrays in C

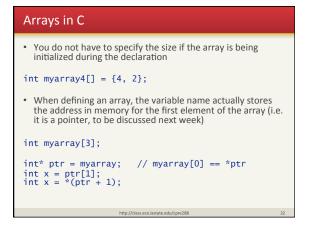
• Examples:

// allocates and initializes 3 chars's char myarray1[3] = {2, 9, 4};

// allocates memory for 5 char's char myarray2[5];

// allocates memory for 2 ints's int myarray3[2];
```





```
Page 24

Pa
```

```
Arrays in C
 • Examples:
   char myarray1[3] = \{2, 9, 4\};
    char myarray2[4];
    int myarray3[2];
Memory
       FF00 FF01 FF02 FF03 FF04 FF05 FF06 FF07 FF08 FF09 FF0A
Address
        0x02 0x09 0x04 ? ? ?
                                         ? ?
Value
Array
          myarray1
                        myarray2
                                         myarray3
        0 1 2 0 1 2 3
                                      0 1 2 3
Index
```

```
Arrays in C
 • Examples:
   char myarray1[3] = \{2, 9, 4\};
   char myarray2[4];
   int myarray3[2];
   myarray1[0] // First element of myarray1
Memory
       FF00 FF01 FF02 FF03 FF04 FF05 FF06 FF07 FF08 FF09 FF0A
Address
Value
       0x02 0x09 0x04 ? ? ?
                                         ? ? ?
Array
          myarray1
                        myarray2
                                         myarray3
        0 1 2 0 1 2
Index
                                  3 0 1 2
                                                 3
```

```
Arrays in C
 • Examples:
   char myarray1[3] = \{2, 9, 4\};
   char myarray2[4];
   int myarray3[2];
   myarray1[2] // Last element of myarray1
Memory
       FF00 FF01 FF02 FF03 FF04 FF05 FF06 FF07 FF08 FF09 FF0A
Address
        0x02 0x09 0x04
                         ? ?
Value
                                      ?
                                         ? ? ?
Arrav
          myarray1
                        myarray2
                                         myarray3
                    0 1 2
Index
        0 1 2
                                  3 0 1 2
                                                  3
```

```
Arrays in C
 • Examples:
    char myarray1[3] = \{2, 9, 4\};
    char myarray2[4];
    int myarray3[2];
   myarray1[3] // Passed end of myarray1!!!
// Overwrote myarray2!!
Memory
        FF00 FF01 FF02 FF03 FF04 FF05 FF06 FF07 FF08 FF09 FF0A
Address
        0x02 0x09 0x04
                            ?
                                ?
                                              ?
Value
Array
           myarray1
                           myarray2
                                              myarray3
Index
                       0
                                          0
         0 1 2
                           1
                               2
                                     3
                                              1 2 3
```

```
Array Copy Example

int TestArray1[20]; // An array of 20 integers int TestArray2[20]; // An array of 20 integers

TestArray1 = TestArray2; // This does not "copy"

for (int i = 0; i < 20; i++) {

TestArray1[i] = TestArray2[i]; // This copies
}
```

```
Arrays in C

• Looping through an array

int myarray[5] = {1, 2, 3, 4, 5};
int x;

for(int i=0; i < 5; i++) {
    x = myarray[i];
    // do something with x
}</pre>
```

```
STRINGS IN C

http://dass.ece.iastate.edu/cpre288 40
```

There are no Strings in C like in Java (there are no classes) Strings are represented as char arrays char is a primitive data type stores 8 bits of data, not necessarily a character can be used to store small numbers

Character Strings in C

A string of characters can be represented as a string literal by putting double quotes around the text: Examples:

```
"This is a string literal."
"123 Main Street"
"X"
```

```
The end of a string (char array) is signified by a null byte

Null bytes have a value of 0
String literals have an automatic null byte included

str1, str2, and str3 below each consume 4 bytes of memory and are equivalent in value:

char* str1 = "123"; // pointer, discuss next week
char str2[] = "123";
char str3[4] = {'1', '2', '3', 0};
```

Character Strings in C

Character Strings in C Do not use statements like: if (str2 == str3) to test equality – str1, str2, and str3 are all pointers (the address of the first char in each array is different) – Use a function like strcmp to test if char arrays are equivalent char str1[] = "123"; char str2[] = "123"; if (strcmp(str1, str2) == 0) { // str1 matches str2 }

http://class.ece.iastate.edu/cpre288

```
    Each character is encoded in 8 bits using ASCII:
    The following statements are equivalent:
    char str[] = "hi";
    char str[3] = { 'h', 'i', '\0' };
    char str[3] = { 104, 105, 0 };
```

Escape Sequences What if we wanted to print the quote character? The following line would confuse the compiler because it would interpret the second quote as the end of the string: char* str = "I said "Hello" to you."; An escape sequence is a series of characters that represents a special character An escape sequence begins with a backslash character (\) char* str = "I said \"Hello\" to you.";

Escape Sequences										
Binary	Oct	Dec	Hex	Abbr	Carrot	Escape	Description			
000 0000	0	0	0	NUL	^@	\0	Null character			
000 0111	7	7	7	BEL	^G	\a	Bell			
000 1000	10	8	8	BS	^H	\b	Backspace			
000 1001	11	9	9	HT	^	\t	Horizontal Tab			
000 1010	12	10	0A	LF	^J	\n	Line feed			
000 1011	13	11	OB	VT	^K	\v	Vertical Tab			
000 1100	14	12	0C	FF	^L	\f	Form feed			
000 1101	15	13	0D	CR	^M	\r	Carriage return			
001 1011	33	27	1B	ESC	^[\e	Escape			
010 0111	47	39	27	1		\'	Single Quote			
010 0010	42	34	22	11		\"	Double Quote			
101 1100	134	92	5C	\		\\	Backslash			
					state.edu/cpre28			48		

Multiline String Literals

- The compiler will concatenate string literals that are only separated by white space.
- The following are equivalent expressions:

 If you need to concatenate string variables, use a function from the standard library like strcat by including <string.h> or sprintf by including <stdio.h>

http://class.ece.iastate.edu/cpre288

Formatting Strings

- printf, sprintf, fprintf = standard library functions for printing data into char arrays
- Must include stdio.h in order to use these function #include <stdio.h>
- These functions have an argument called a formatter string that accepts % escaped variables
- Review the documentation on functionality of $\mathit{sprint} f$
 - Google "sprintf", first result is:
 - http://www.cplusplus.com/reference/clibrary/cstdio/sprintf/
- TAs will review basic string manipulation functions in Lab

http://class.ece.iastate.edu/cpre288

LAB 1 OVERVIEW

Lab 1: Introduction to the Platform

Purpose: Introduction to the AVR Studio 5 and VORTEX Platform

- AVR Studio 5: The integrated development environment (IDE) for Atmel AVR platforms
- VORTEX: An integrated hardware platform of iRobot Create and Cerebot II microcontroller board

AVR Studio 5

An IDE from Atmel for AVR platforms

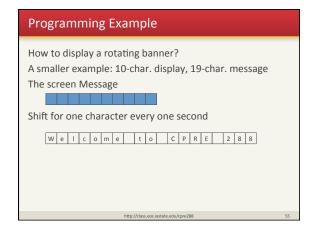
- Source code editing
- Compiling building
- Download binary to boards
- Debug
- Simulation

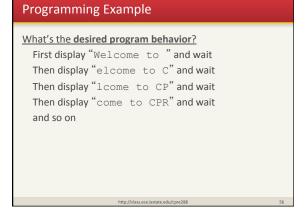
Lab 1

Lab 1: Introduction to the AVR Studio 5

- Part 1 "Hello, world"
 - Build, download, and execute
- Part 2 Simulated Environment
- Part 3 Rotating Banner
 - The message has 34 characters and the LCD can only show 20 characters per line at a time

http://class.ece.iastate.edu/cpre288





Programming Example

Give a general but precise description

First show characters 0-9 and wait Then show characters 1-10 and wait Then show characters 2-11 and wait Then show characters 3-12 and wait and so on

http://class.ece.iastate.edu/cpre288

Programming Example

Describe program's behavior

set starting position at 0
loop forever
clear the screen
display 10 chars from the starting pos.
shift the starting pos. to the next position
wait for one second
end loop

http://class.ece.iastate.edu/cpre288

Programming Example

Some details to take care

"display 10 chars from the starting pos."

"shift the starting pos. to the next position"

http://class.ece.iastate.edu/cpre288

Lab 1 Programming Exercise

Part 3. Rotating Banner

Show "Microcontrollers are loads of fun!" in a rotating style

- The message has 34 characters and the LCD line has 20
- Shift in first 20 characters one by one, with 0.3 second delay
- Start to rotate and continue till the last character is shown, with 0.5 second delay
- Continue rotating until the screen becomes clear, with 0.5 second delay
- Repeat this procedure

http://class.ece.iastate.edu/cpre288

Lab 1 Programming Exercise

First, have a function to print the banner for one time

void print_banner(char *msg, int start, int end);

This makes the rest of programming easier

http://class.ece.iastate.edu/cpre288

Lab 1 Programming Exercise

Idea 1: A forever loop of three phases

Phase 1: Shift in the first 20 characters

Phase 2: Rotate until the last character is displayed

Phase 3: Rotate until the last character is shifted out

http://class.ece.iastate.edu/cpre288

Lab 1 Programming Exercise