C for PIC

E155

Outline

- Welcome to C
- Syntax
- Variables
- Operations
- Function calls
- Control statement
- More data type
- Timer examples

Welcome to C

- Invented in 1973 by Dennis Ritchie of Bell Labs
- Its popularity stems from a number of factors including:
  - Availability for a tremendous variety of platforms, from supercomputers down to embedded microcontrollers
  - Relative ease of use, with a huge user base
  - Moderate level of abstraction providing higher productivity than assembly language, yet giving the programmer a good understanding of how the code will be executed

Welcome to C

- Suitability for generating high performance programs
- Ability to interact directly with the hardware
- In summary, C allows the programmer to directly access addresses in memory, illustrating the connection between hardware and software
- C is a practical language that all engineers and computer scientists should know. Its wide usages make proficiency in C a vital and marketable skill.

Simple Program

C Code Example C.1. Simple C program

```
// Writes "Hello world!" to the console
#include <stdio.h>

int main(void){
  printf("Hello world!\n");
}
```

Console Output

```
Hello world!
```

Comments/Define

```
// this is an example of a one-line
comment.

/* this is an example
   of a multi-line comment */

#define TRISD 0xFE000044
```

Primitive data types and sizes

Type	Size (bits)	Minimum	Maximum
char	8	$-2^{-7} = -128$	$2^7 - 1 = 127$
unsigned char	8	0	$2^8 - 1 = 255$
short	16	$-2^{15} = -32{,}768$	$2^{15} - 1 = 32{,}767$
unsigned short	16	0	$2^{16} - 1 = 65{,}535$
long	32	$-2^{31} = -2{,}147{,}483{,}648$	$2^{31} - 1 = 2{,}147{,}483{,}647$
unsigned long	32	0	$2^{32} - 1 = 4{,}294{,}967{,}295$
long long	64	-2^{63}	$2^{63} - 1$
unsigned long long	64	0	$2^{64} - 1$
int	machine-dependent		
unsigned int	machine-dependent		
float	32	$\pm 2^{-126}$	$\pm 2^{127}$
double	64	$\pm 2^{-1023}$	$\pm 2^{1022}$

C Code Example: Data Types

```
// Examples of several data types and their binary representations
unsigned char x = 42;                  // x = 00101010
short y = -10;                         // y = 11111111 11110110
unsigned long z = 0;                   // z = 00000000 00000000 00000000 00000000
```

Global vs Local Variable

C Code Example: Global Variables

```
// Use a global variable to find and print the maximum of 3 numbers

int max;                    // global variable holding the maximum value

void findMax(int a, int b, int c) {
  max = a;
  if (b > max) {
    if (c > b) max = c;
    else       max = b;
  } else if (c > max) max = c;
}

void printMax(void) {
  printf("The maximum number is: %d\n", max);
}

int main(void) {
  findMax(4, 3, 7);
  printMax();
}
```

Global vs Local Variable

C Code Example C.5. Local Variables

```
// Uses local variables to find and print the maximum of 3 numbers

int getMax(int a, int b, int c) {
  int result = a;    // local variable holding the maximum value

  if (b > result) {
    if (c > b) result = c;
    else       result = b;
  } else if (c > result) result = c;

  return result;
}

void printMax(int m) {
  printf("The maximum number is: %d\n", m);
}

int main(void) {
  int max;

  max = getMax(4, 3, 7);
  printMax(max);
}
```

Operators

	Category	Operator	Description	Example
Highest	Monadic	++	post-increment	a++; // a = a+1
		--	post-decrement	x--; // x = x-1
		&	memory address of a variable	x = &y; // x = the memory // address of y
		~	bitwise NOT	z = ~a;
		!	Boolean NOT	!x
		-	negation	y = -a;
		++	pre-increment	++a; // a = a+1
		--	pre-decrement	--x; // x = x-1
		(type)	casts a variable to (type)	x = (int)c; // cast c to an // int and assign it to x
		sizeof()	size of a variable in bytes	long int y; x = sizeof(y); // x = 4
	Multiplicative	*	multiplication	y = x * 12;
		/	division	z = 9 / 3; // z = 3
		%	modulo	z = 5 % 2; // z = 1
	Additive	+	addition	y = a + 2;
		-	subtraction	y = a - 2;
	Bitwise Shift	<<	bitshift left	z = 5 << 2; // z = 0b0001 0100
Lowest		>>	bitshift right	x = 9 >> 3; // x = 0b0000 0001

Operators

	Category	Operator	Description	Example
Highest Lowest	Relational	==	equals	y == 2
		!=	not equals	x != 7
		<	less than	y < 12
		>	greater than	val > max
		<=	less than or equal	z <= 2
		>=	greater than or equal	y >= 10
	Bitwise	&	bitwise AND	y = a & 15;
		|	bitwise OR	y = a | b;
		^	bitwise XOR	y = 2 ^ 3;
	Logical	&&	Boolean AND	x && y
		||	Boolean OR	x || y
	Ternary	? :	ternary operator	y = x ? a : b;// if x is TRUE, // y=a, else y=b

Operators

	Category	Operator	Description	Example
Highest	Assignment	=	assignment	x = 22;
		+=	addition and assignment	y += 3; // y = y + 3
		-=	subtraction and assignment	z -= 10; // z = z – 10
		*=	multiplication and assignment	x *= 4; // x = x * 4
		/=	division and assignment	y /= 10; // y = y / 10
		%=	modulo and assignment	x %= 4; // x = x % 4
		>>=	bitwise right-shift and assignment	x >>= 5; // x = x>>5
		<<=	bitwise left-shift and assignment	x <<= 2; // x = x<<2
		&=	bitwise AND and assignment	y &= 15; // y = y & 15
		|=	bitwise OR and assignment	x |= y; // x = x | y
Lowest		^=	bitwise XOR and assignment	x ^= y; // x = x ^ y

C Code Example: Operator Examples

Expression	Result	Notes
44 / 14	3	Integer division truncates
44 % 14	2	44 mod 14
0x2C && 0xE //0b101100 && 0b1110	1	Logical AND
0x2C ll 0xE //0b101100 ll 0b1110	1	Logical OR
0x2C & 0xE //0b101100 & 0b1110	0xC (0b001100)	Bitwise AND
0x2C l 0xE //0b101100 l 0b1110	0x2E (0b101110)	Bitwise OR
0x2C ^ 0xE //0b101100 ^ 0b1110	0x22 (0b100010)	Bitwise XOR
0xE << 2 //0b1110 << 2	0x38 (0b111000)	Left shift by 2
0x2C >> 3 //0b101100 >> 3	0x5 (0b101)	Right shift by 3
x = 14; x += 2;	x=16	

y = 0x2C; // y = 0b101100
y &= 0xF; // y &= 0b1111
	y=0xC (0b001100)	

x = 14; y = 44; y = y + x++;	x=15, y=58	Increment x after using it
x = 14; y = 44; y = y + ++x;	x=15, y=59	Increment x before using it
y = (a > b) ? a : b;		if (a > b) y = a; else y = b;

Function Calls

C Code Example C.8. sum3()function

```
// Returns the sum of the three input variables
int sum3(int a, int b, int c) {
  int result = a + b + c;
  return result;
}
```

After the following call to sum3(), y holds the value 42.

```
 int y = sum3(10, 15, 17);
 // printPrompt() prints a user prompt to the
console.
void printPrompt(void)
{
  printf("Please enter a number from 1-3:\n");
}
```

Function Prototype

- A function must be declared in the code before it is called by another function. This may be done by placing the called function earlier in the file. For this reason, main() is often placed at the end of the C file after all the functions it calls.
- Alternatively, a function *prototype* can be placed in the program before the function is defined.

```
int sum3(int a, int b, int c);
void printPrompt(void);
```

Function Prototype

```
// function prototypes
int sum3(int a, int b, int c);
void printPrompt(void);

int main(void)
{
  int y = sum3(10, 15, 20);

  printf("sum3 result: %d\n", y);
  printPrompt();
}

int sum3(int a, int b, int c) {
        int result = a+b+c;
        return result;
}

void printPrompt(void) {
        printf("Please enter a number from 1-3:\n");
}
```

Conditional Statements

- if/if else

```
int dontFix = 0;

if (aintBroke == 1)
  dontFix = 1;

if (a > b)        y = a;
else              y = b;
```

Conditional Statements

- switch/case

```
// assign amt depending on the value of option
switch (option) {
  case 1:      amt = 100; break;
  case 2:      amt = 50;  break;
  case 3:      amt = 20;  break;
  case 4:      amt = 10;  break;
  default:     printf("Error: unknown option\n");
}

// assign amt depending on the value of option
if      (option == 1)  amt = 100;
else if (option == 2)  amt = 50;
else if (option == 3)  amt = 20;
else if (option == 4)  amt = 10;
else    printf("Error: unknown option\n");
```

Conditional Statements: Loops

```
// Compute 9! (the factorial of 9)
int i = 1, fact = 1;

// multiply the numbers from 1 to 9
while (i < 10) { // while loops check the condition first
  fact *= i;  //fact = fact * i;
  i++;
```

Conditional Statements: loops

```
// Query user to guess a number and check it against the
correct number.
#define MAXGUESSES 3
#define CORRECTNUM 7

int guess, numGuesses = 0;

do {
  printf("Guess a number between 0 and 9. You have %d more
guesses.\n",
     (MAXGUESSES-numGuesses));
  scanf("%d", &guess);   // read user input
  numGuesses++;
} while ( (numGuesses < MAXGUESSES) & (guess != CORRECTNUM) );
// do loop checks the condition after the first iteration

if (guess == CORRECTNUM)
  printf("You guessed the correct number!\n");
```

Conditional Statements: loops

```
 /*
 * helloworld.c: simple test application
 */

#include <stdio.h>
#include "platform.h"

void print(char *str);
int a[5]={100, 56, 20, 30, -5};
int i, j;
int tmp;
int flag;
int main()
{
    init_platform();

    print("Hello World\n\r");
    print("This is original array\n\r");

    for (i=0; i<5; i++)
            printf("%10d\n", a[i]);
```

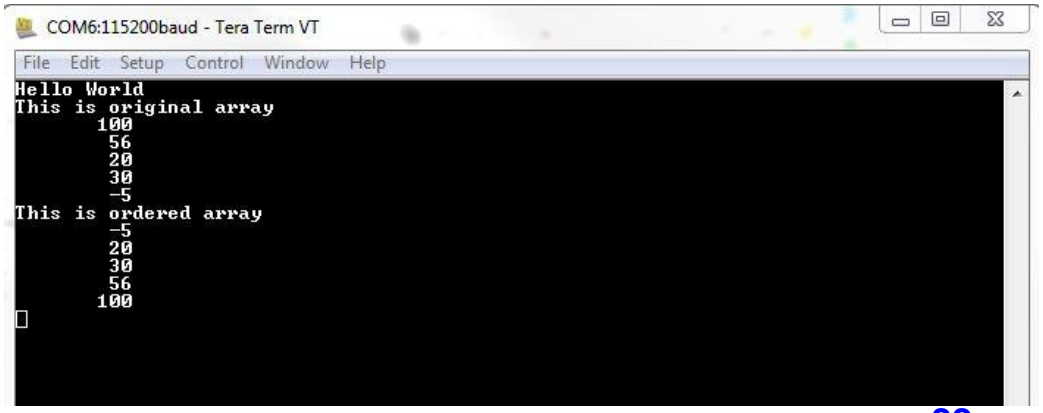
Conditional Statements: loops

```
    //bubble sort
    for  (j=0; j<4; j++){
            flag=0;
            for (i=0; i<4; i++){
                        if (a[i]>a[i+1]){
                                    tmp=a[i];
                                    a[i]=a[i+1];
                                    a[i+1]=tmp;
                                    flag=1;
                        }
            }
     if (flag=0)
              break;
     }
    print("This is ordered array\n\r");

       for (i=0; i<5; i++)
            printf("%10d\n", a[i]);

    cleanup_platform();

    return 0;
}
```



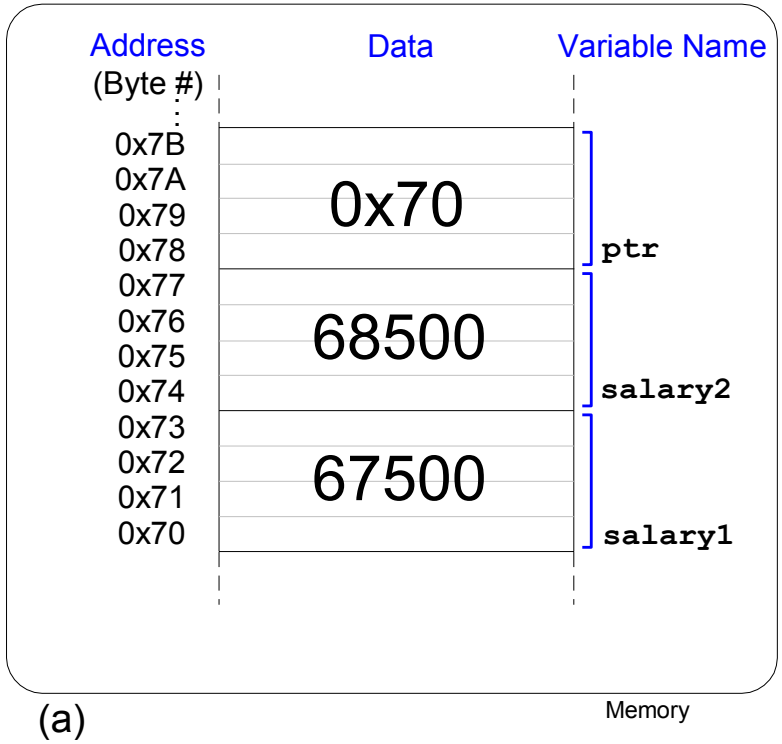
Data Types

- Pointers

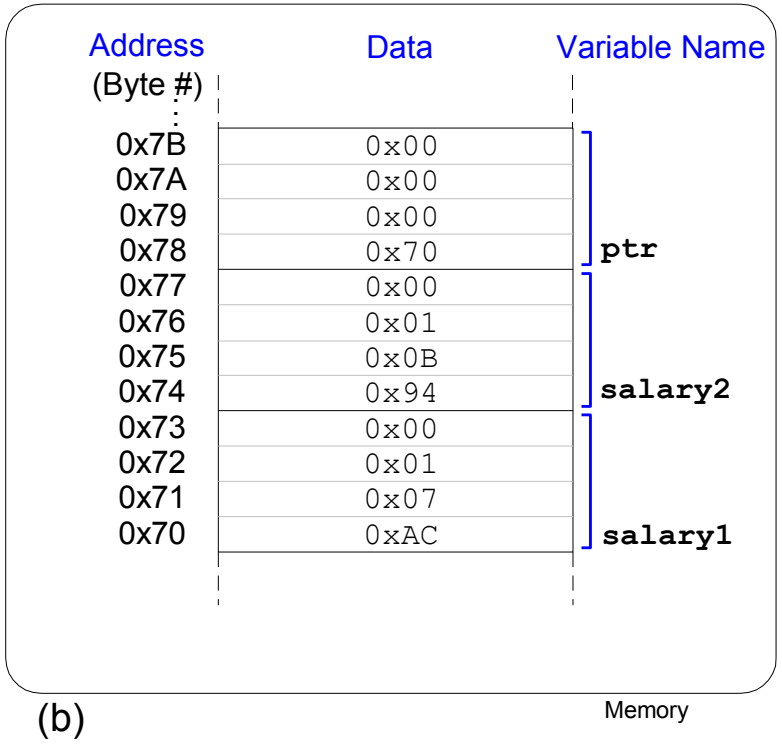
```
// Example pointer manipulations
int salary1, salary2;  // 32-bit numbers
int *ptr;              // a pointer specifying the
address of an int variable

salary1 = 67500;     // salary1 = $67,500 = 0x000107AC
ptr = &salary1;  // ptr = 0x0070, the address of salary1
salary2 = *ptr + 1000;
/* dereference ptr to give the contents of address 70 =
$67,500, then add $1,000 and set salary2 to $68,500 */
```

Pointers


Address
(Byte #)
Data
Variable Name
0x7B
0x7A
0x79
0x78
0x77
0x76
0x75
0x74
0x73
0x72
0x71
0x70
0x70
68500
67500
ptr
salary2
salary1
Memory

(a)


Address
(Byte #)
Data
Variable Name
0x7B
0x7A
0x79
0x78
0x77
0x76
0x75
0x74
0x73
0x72
0x71
0x70
0x00
0x00
0x00
0x70
0x00
0x01
0x0B
0x94
0x00
0x01
0x07
0xAC
ptr
salary2
salary1
Memory

(b)

Pointer

C Code Example: Passing an input variable by reference

```
// Quadruple the value pointed to by a.
#include <stdio.h>

void quadruple(int *a)
{
  *a = *a * 4;
}
 int main(void)
{
  int x = 5;

  printf("x before: %d\n", x);
  quadruple(&x);
  printf("x after: %d\n", x);
  return 0;
}
```

Console Output

x before: 5

x after: 20