C for PIC

E155
Outline

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

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

### C Code Example: Data Types

```c
// 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);
    printMax(max);
}
# Operators

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>==</td>
<td>equals</td>
<td>y == 2</td>
</tr>
<tr>
<td></td>
<td>!=</td>
<td>not equals</td>
<td>x != 7</td>
</tr>
<tr>
<td></td>
<td>&lt;</td>
<td>less than</td>
<td>y &lt; 12</td>
</tr>
<tr>
<td></td>
<td>&gt;</td>
<td>greater than</td>
<td>val &gt; max</td>
</tr>
<tr>
<td></td>
<td>&lt;=</td>
<td>less than or equal</td>
<td>z &lt;= 2</td>
</tr>
<tr>
<td></td>
<td>&gt;=</td>
<td>greater than or equal</td>
<td>y &gt;= 10</td>
</tr>
<tr>
<td>Bitwise</td>
<td>&amp;</td>
<td>bitwise AND</td>
<td>y = a &amp; 15;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bitwise OR</td>
<td>y = a</td>
</tr>
<tr>
<td></td>
<td>^</td>
<td>bitwise XOR</td>
<td>y = 2 ^ 3;</td>
</tr>
<tr>
<td>Logical</td>
<td>&amp;&amp;</td>
<td>Boolean AND</td>
<td>x &amp;&amp; y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Ternary      | ? :      | ternary operator | y = x ? a : b; // if x is TRUE,  
<p>|              |          |             | // y=a, else y=b       |</p>
<table>
<thead>
<tr>
<th>Category</th>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment</td>
<td>=</td>
<td>assignment</td>
<td>x = 22;</td>
</tr>
<tr>
<td></td>
<td>+=</td>
<td>addition and assignment</td>
<td>y += 3;</td>
</tr>
<tr>
<td></td>
<td>-=</td>
<td>subtraction and assignment</td>
<td>z -= 10;</td>
</tr>
<tr>
<td></td>
<td>*=</td>
<td>multiplication and assignment</td>
<td>x *= 4;</td>
</tr>
<tr>
<td></td>
<td>/=</td>
<td>division and assignment</td>
<td>y /= 10;</td>
</tr>
<tr>
<td></td>
<td>%=</td>
<td>modulo and assignment</td>
<td>x %= 4;</td>
</tr>
<tr>
<td></td>
<td>&gt;&gt;=</td>
<td>bitwise right-shift and assignment</td>
<td>x &gt;&gt;= 5;</td>
</tr>
<tr>
<td></td>
<td>&lt;&lt;=</td>
<td>bitwise left-shift and assignment</td>
<td>x &lt;&lt;= 2;</td>
</tr>
<tr>
<td></td>
<td>&amp;=</td>
<td>bitwise AND and assignment</td>
<td>y &amp;= 15;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=</td>
<td>bitwise OR and assignment</td>
</tr>
<tr>
<td></td>
<td>^=</td>
<td>bitwise XOR and assignment</td>
<td>x ^= y;</td>
</tr>
</tbody>
</table>
## C Code Example: Operator Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 / 14</td>
<td>3</td>
<td>Integer division truncates</td>
</tr>
<tr>
<td>44 % 14</td>
<td>2</td>
<td>44 mod 14</td>
</tr>
<tr>
<td>0x2C &amp;&amp; 0xE //0b101100 &amp;&amp; 0b1110</td>
<td>1</td>
<td>Logical AND</td>
</tr>
<tr>
<td>0x2C</td>
<td></td>
<td>0xE //0b101100</td>
</tr>
<tr>
<td>0x2C &amp; 0xE //0b101100 &amp; 0b1110</td>
<td>0xC (0b001100)</td>
<td>Bitwise AND</td>
</tr>
<tr>
<td>0x2C</td>
<td>0xE //0b101100</td>
<td>0x2E (0b101110)</td>
</tr>
<tr>
<td>0x2C ^ 0xE //0b101100 ^ 0b1110</td>
<td>0x22 (0b100010)</td>
<td>Bitwise XOR</td>
</tr>
<tr>
<td>0xE &lt;&lt; 2 //0b1110 &lt;&lt; 2</td>
<td>0x38 (0b111000)</td>
<td>Left shift by 2</td>
</tr>
<tr>
<td>0x2C &gt;&gt; 3 //0b101100 &gt;&gt; 3</td>
<td>0x5 (0b101)</td>
<td>Right shift by 3</td>
</tr>
<tr>
<td>x = 14; x += 2;</td>
<td>x=16</td>
<td></td>
</tr>
<tr>
<td>y = 0x2C; // y = 0b101100 y &amp;= 0xF; // y &amp;= 0b1111</td>
<td>y=0xC (0b001100)</td>
<td></td>
</tr>
<tr>
<td>x = 14; y = 44; y = y + x++;</td>
<td>x=15, y=58</td>
<td>Increment x after using it</td>
</tr>
<tr>
<td>x = 14; y = 44; y = y + ++x;</td>
<td>x=15, y=59</td>
<td>Increment x before using it</td>
</tr>
<tr>
<td>y = (a &gt; b) ? a : b;</td>
<td>if (a &gt; b) y = a; else y = b;</td>
<td></td>
</tr>
</tbody>
</table>
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");
}
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.

```c
int sum3(int a, int b, int c);
void printPrompt(void);
```
// 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");
}
if/if else

```c
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");
// 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++;
}
// 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");
/*
 * 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");
    print("This is original array\n");

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

<table>
<thead>
<tr>
<th>Address (Byte #)</th>
<th>Data</th>
<th>Variable Name</th>
<th>Address (Byte #)</th>
<th>Data</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x7B</td>
<td>0x70</td>
<td></td>
<td>0x7B</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x7A</td>
<td>0x70</td>
<td></td>
<td>0x7A</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x79</td>
<td>0x70</td>
<td></td>
<td>0x79</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x78</td>
<td>0x00</td>
<td>ptr</td>
<td>0x78</td>
<td>0x70</td>
<td></td>
</tr>
<tr>
<td>0x77</td>
<td>0x00</td>
<td></td>
<td>0x77</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x76</td>
<td>0x00</td>
<td></td>
<td>0x76</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>0x75</td>
<td>0x0B</td>
<td>salary2</td>
<td>0x75</td>
<td>0x0B</td>
<td></td>
</tr>
<tr>
<td>0x74</td>
<td>0x94</td>
<td>salary2</td>
<td>0x74</td>
<td>0x94</td>
<td></td>
</tr>
<tr>
<td>0x73</td>
<td>0x00</td>
<td></td>
<td>0x73</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x72</td>
<td>0x01</td>
<td></td>
<td>0x72</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>0x71</td>
<td>0x07</td>
<td></td>
<td>0x71</td>
<td>0x07</td>
<td></td>
</tr>
<tr>
<td>0x70</td>
<td>0xAC</td>
<td></td>
<td>0x70</td>
<td>0xAC</td>
<td></td>
</tr>
</tbody>
</table>

(a) Memory

(b) Memory
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