Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería

MICROCONTROLADORES

Práctica No. 1 Introducción

Docente: García López Jesús Adán

Alumno:

Gómez Cárdenas Emmanuel Alberto 01261509

Indice

Objetivo:

El alumno se familiarizará con algunos usos típicos del Lenguaje C en sistemas embebidos.

Material:

- Computadora Personal

Teoría:

- Lenguaje C y su uso en sistemas embebidos
- Compilador Cruzado (Cross-Compiler)

Desarrollo:

Investigar

Tamaño de las primitivas y uso de *inttypes.h*.

El tamaño de las primitivas en C puede variar dependiendo de la arquitectura del sistema en el que se esté compilando el programa. Es importante conocer el tamaño de las primitivas para garantizar la portabilidad del código entre diferentes plataformas, por lo que es recomendable utilizar librerías como inttypes.h la cual define enteros con tamaño garantizado, lo que permite escribir código que funcione de manera consistente en diferentes arquitecturas.

Tamaño de las primitivas

Tipo de Dato	Tamaño (en bytes)
char	1
signed char	1
unsigned char	1
short	2
int	4
long	4 o 8 (dependiendo de la plataforma)
long long	8
float	4
double	8
long double	8 o 16 (dependiendo de la plataforma)

Unos de los tipos de datos definidos en Inttypes.h

Tipo de Dato	Tamaño (en bytes)
int8_t	1
int16_t	2
int32_t	4
int64_t	8
int_least8_t	1
int_least16_t	2
int_least32_t	4
int_least64_t	8
intmax_t	Depende de la implementación

Representación de literales (1, 0x1, 0b1, 0)

Son valores fijos que se especifican en el código fuente y no cambian durante la ejecución del programa. La representación de los literales depende del tipo de datos utilizado, y puede tener limitaciones en términos de precisión y rango. Para representar números enteros, se pueden utilizar valores decimales, octales o hexadecimales. Por ejemplo, el número decimal 1 se puede representar como “1” en decimal, “0x1” en hexadecimal, “0b1” en binario y 01 en octal.

Para representar números de punto flotante, se puede usar notación decimal con o sin notación científica. Los caracteres se representan entre comillas simples y las cadenas de caracteres entre comillas dobles. Es importante tener en cuenta que la representación de los literales puede variar dependiendo del tipo de datos utilizado.

Operadores Bitwise ({&, |, ~, ^, >>, << } vs {&&, ||, !})

En C, los operadores bitwise, como & (AND), | (OR), ~ (NOT), ^ (XOR), << (desplazamiento a la izquierda) y >> (desplazamiento a la derecha), se utilizan para manipular bits individuales de un número entero y son aplicados (somo su nombre lo indica) a nivel de bit. Por otro lado, los operadores lógicos, como && (AND lógico), || (OR lógico) y ! (NOT lógico), se utilizan para evaluar expresiones lógicas, estos son aplicados a nivel de expresión. Los operadores bitwise son útiles en situaciones en las que se desea manipular bits individuales, como en la creación de máscaras de bits o en el manejo de puertos de E/S. Por otro lado, los operadores lógicos son útiles para evaluar expresiones booleanas y controlar el flujo de ejecución del programa.

Apuntadores (char *x, int (*cb)(void))

Un apuntador es una variable que almacena la dirección de memoria de otra variable. Los apuntadores se utilizan para manipular datos indirectamente y para realizar operaciones avanzadas en memoria, como la asignación dinámica de memoria.

Estos pueden ser utilizados para apuntar a variables de cualquier tipo de datos, como enteros, flotantes, caracteres, arreglos y estructuras. Por ejemplo, el tipo de dato char *x indica que x es un apuntador a un carácter, mientras que el tipo de dato int (*cb)(void) a diferencia de un apuntador normal que apunta a datos, un apuntador a funcion punta a codigo, en este caso int (*cb)(void) apunta a una función que no toma argumentos y devuelve un entero.

Estructuras (struct, union, bitfield)

Una estructura (struct) es un tipo de dato definido por el usuario que agrupa varios tipos de datos en un solo objeto. Una unión (union) es un tipo de estructura que permite a los miembros compartir la misma área de memoria, lo que significa que sólo un miembro de la unión puede tener un valor asignado a la vez. Un campo de bits (bitfield) es una estructura que permite acceder y manipular datos en términos de bits individuales en lugar de bytes completos. Ejemplos:

```
struct Libro {
  char titulo[50];
  char autor[50];
  int paginas;
  float precio;
};
```

```
union Numero {
  int entero;
  float flotante;
  char cadena[50];
};
```

```
struct Estado {
  unsigned int encendido: 1;
  unsigned int puerta_abierta: 1;
  unsigned int frenos_activos: 1;
  unsigned int velocidad: 4;
};
```

Palabras claves (keywords: const, static, volatile)

Son palabras reservadas en un lenguaje de programación que tienen un significado especial para el compilador o intérprete del lenguaje. Estas palabras clave no pueden ser utilizadas como nombres de variables, funciones o cualquier otro identificador del programa.

Keywords	Descripcion	Ejemplo
const	Se utiliza para definir una variable cuyo valor no puede ser modificado después de su inicialización. Esto puede ser útil para definir constantes, como el valor de PI en una aplicación de matemáticas.	const int edad = 20;
static	Se utiliza para definir una variable o función con un ámbito de visibilidad limitado a un archivo en particular. Esto puede ser útil para evitar conflictos de nombres en programas grandes y complejos que utilizan múltiples archivos fuente.	static int contador = 0;
volatile	Se utiliza para indicar que el valor de una variable puede cambiar en cualquier momento, incluso sin una asignación explícita. Esto puede ser útil en situaciones donde el valor de una variable puede ser modificado por un hardware externo o un proceso concurrente.	volatile int *direccion = 0x1000;

Condicionales del pre-compilador (#)

Las directivas de preprocesador, se usan normalmente para hacer que los programas de origen sean más fáciles de modificar y compilar en diferentes entornos de ejecución, estas indican al preprocesador que realice acciones especificas. El numeral (#) debe ser el primer carácter de la directiva

Condicional	Descripción	Ejemplo
#define, #undef	Define una etiqueta temporal #Undef anula la definición de una etiqueta	#define WIDTH 80 #define LENGTH (WIDTH + 10)

#if, #elif, #else, #endif	Estas directivas son utilizadas para controlar el proceso de ensamblado.
	Si la condición de la directiva #if no es verdadera la instrucciones no generan código hasta encontrar una directiva #endif o #else.
		#if defined(CREDIT)
		credit();
		#elif defined(DEBIT)
		debit();
		#else
		printerror();
		#endif

#error	La directiva **#error** emite un mensaje de error especificado por el usuario en tiempo de compilación y después finaliza la compilación.	#error “Error”
#ifdef, #ifndef	Las directivas de preprocesador #ifdef y #ifndef tienen el mismo efecto que la directiva #if cuando se usa con el operador defined. La directiva #ifndef comprueba lo contrario de la condición que comprueba #ifdef. Si el identificador no se ha definido o si su definición se ha quitado con #undef, la condición es true (distinto de cero). De lo contrario, la condición es false (0).	#ifndef test #define final #endif
#include	Se utiliza para insertar contenido de un archivo fuente en un punto especifico	#include <stdio.h>
#message	Emite un mensaje especificado por el usuario en tiempo de compilacion	#message “mensaje de prueba”
//, //**	Se utilizan para comentar secciones del listado, /**/ comenta todo lo que se encuentre dentro de estos, mientras que // marca el resto de la línea como comentario	/* Este es un comentario de bloque */ //Este es un comentario sencillo

Casteo (type casting)

El casteo en C (también conocido como "casting" en inglés) es un operador que permite convertir un valor de un tipo de dato a otro tipo de dato compatible. Esto se logra mediante la sintaxis "(tipo de dato) valor", donde el tipo de dato entre paréntesis indica el tipo de dato al que se quiere convertir el valor. (ej. cuando a una variable entera se le asigna un valor flotante).

Tipo de casteo	Descripción	Ejemplo
Casteo implícito	Conversión de un tipo de dato a otro sin necesidad de una operación explícita	**int a = 10;** **float b = a;**
Casteo explícito	Conversión de un tipo de dato a otro mediante el uso de operadores de casteo	**int a = 10;** **float b = (float)a;**
Casteo de punteros	Conversión de un puntero de un tipo de dato a otro tipo de dato	**int a = 10;** **int *ptr = &a;** **char *cptr = (char*)ptr;**
Casteo de enteros	Conversión de un tipo de dato entero a otro tipo de dato entero	**int a = 10;** **short b = (short)a;**
Casteo de punto flotante	Conversión de un tipo de dato de punto flotante a otro tipo de dato de punto flotante	**float a = 10.5;** **double b = (double)a;**
Casteo de caracteres	Conversión de un tipo de dato de carácter a otro tipo de dato de carácter	**char a = 'a';** **int b = (int)a;**
Casteo de estructuras	Conversión de una estructura a otra estructura con diferente definición	**struct person {** **char name[20];** **int age;** **};**

		struct student { **char name[20];** **int grade;** **};** **struct person p = {** **"John", 25** **};** **struct student s = *(struct student*)&p;**

Git

Git es un sistema distribuido de control de versiones distribuido que se utiliza para rastrear cambios en el código de fuente de un proyecto a lo largo del tiempo.

GitHub

GitHub es una plataforma de alojamiento de repositorios Git que se utiliza para la gestión de proyectos y la colaboración en el desarrollo de software. En GitHub, los usuarios pueden crear repositorios públicos o privados y trabajar en conjunto con otros desarrolladores mediante herramientas de colaboración como pull requests y problemas (issues).

Conclusiones y Comentarios.

En esta practica repasamos conocimentos previamente adquiridos sobre el lenguaje C, como lo son las primitivas, tipos de datos ,uso de operadores, entre otras cosas. Tambien se tocaron temas mas complejos como lo son el uso de punteros a funciones, el uso de las keywords (const, static, volatile) y demas. Esto es bastante importante conocerlo para poder desarrollar programas eficientes y portables. Esta práctica nos permitió ampliar nuestros conocimientos sobre dichos conceptos, lo que nos facilitará el desarrollo de futuras prácticas.

Bibliografía

(inttypes.h). cplusplus.com. (n.d.). Retrieved February 14, 2023, from https://cplusplus.com/reference/cinttypes/

C programming notes. (n.d.). Retrieved February 14, 2023, from https://eskimo.com/~scs/cclass/notes/top.html

Choudhary, V. (2018, March 29). *Type casting - C programming*. Developer Insider. Retrieved February 14, 2023, from https://developerinsider.co/type-casting-c-programming/

Constants. cplusplus.com. (n.d.). Retrieved February 14, 2023, from https://cplusplus.com/doc/tutorial/constants/

Corob-Msft. (n.d.). *Preprocessor directives*. Microsoft Learn. Retrieved February 14, 2023, from https://learn.microsoft.com/en-us/cpp/preprocessor/preprocessor-directives?view=msvc-170

Documentation on MQL5: Language basics / preprocessor / conditional compilation (#ifdef, #IFNDEF, #else, #endif). MQL5. (n.d.). Retrieved February 14, 2023, from https://www.mql5.com/en/docs/basis/preprosessor/conditional_compilation

Fundamental types. cppreference.com. (n.d.). Retrieved February 14, 2023, from https://en.cppreference.com/w/cpp/language/types

GeeksforGeeks. (2018, September 5). *Function pointer in C*. GeeksforGeeks. Retrieved February 14, 2023, from https://www.geeksforgeeks.org/function-pointer-in-c/

GeeksforGeeks. (2019, March 5). *Callbacks in C*. GeeksforGeeks. Retrieved February 14, 2023, from https://www.geeksforgeeks.org/callbacks-in-c/

GeeksforGeeks. (2022, October 25). *C++ Pointers*. GeeksforGeeks. Retrieved February 14, 2023, from https://www.geeksforgeeks.org/cpp-pointers/

Git. (n.d.). Retrieved February 14, 2023, from https://git-scm.com/

An introduction to github. Digital.gov. (2020, June 18). Retrieved February 14, 2023, from https://digital.gov/resources/an-introduction-github/

Operators. cplusplus.com. (n.d.). Retrieved February 14, 2023, from https://www.cplusplus.com/doc/tutorial/operators/

Structures in C. Structures in C - Cprogramming.com. (n.d.). Retrieved February 14, 2023, from https://www.cprogramming.com/tutorial/c/lesson7.html