

Trabajo Practico numero 1 Sistemas digitales

Lenguaje C

a) Explique cuales son los tipos de variables de C, el tamaño en bits y el rango de representación en cada caso ? Como puede evaluarse el tamaño de una variable en C

Tipos de variables en C:

name	size	range
char	8-bits	[0,255]
signed char	8-bits	[-127,127]
unsigned char	8-bits	[0,2 ^{CHAR_BIT} -1]
short int	16-bits	[-32767,32767]
unsigned short	16-bits	[0,65535]
int	16-bits	[-32767,32767]
unsigned int	16-bits	[0,65535]
long	32-bits	[-2147483647,2147483647]
unsigned long	32-bits	[0,4294967295]
long long	64-bits	[-9223372036854775807,9223372036854775807]
unsigned long long	64-bits	[0,18446744073709551615]
float	single-precision	IEEE 754
double	double-precision	IEEE 754-double
long double	quadruple-precision	IEEE 754-quadruple

El tamaño puede evaluarse simplemente haciendo: `sizeof (var)`

b) Explique para que sirven los modificadores: `volatile`, `const`, `static` y `extern`

volatile: Sirve para hacerle saber al compilador que esa variable que estamos declarando puede sufrir modificaciones(en cualquier tiempo) sin que sea por alguna acción del código mismo: Una variable debe ser declarada **volatile** cuando su valor puede cambiar inesperadamente. Solo tres tipos de variables pueden tener este comportamiento:

- Registros mapeados en memoria
- Variables globales modificadas por un servicio de interrupcion
- Variables globales accedidas por un programa de muchos hilos(multithread)

const: Sirve para declarar una variable que sabemos que no debe ser cambiado su valor la cual si en algun momento del programa es cambiada el compilador nos lo va a indicar.

static: Si declaramos **static** dentro de una funcion, sirve para que el valor persista entre llamadas a ella, si declaramos una funcion/variable global como **static** esta solo sera “visible” en el archivo que fue declarada.

extern: Sirve para declarar variables que pueden ser vistas en otros archivos(Por ejemplo en un .h declaramos una de estas variables y cuando lo incluimos, ella ya es accesible desde el programa que llamo al .h)

- c) Explique para que sirven las sentencias de precompilacion: **#include**, **#define**, **ifndef**

#include: Sirve para incluir archivos de cabecera y archivos normales.

#define: Sirve para definir constantes literales o macros(pedazos de codigo que se pegan tal cual los definimos en el codigo que las llama)

ifndef: Una vez que el archivo .h(header) es incluido con esa directiva el compilador se fija si esta definido o no para no volver a incluir el archivo. Esto previene declaraciones dobles.

d)Explique **typedef**, de algunos ejemplos de usos.

typeded: Sirve para poder renombrar **types** por ejemplo: **typedef unsigned char BYTE;**

Tambien para renombrar los **types** del usuario, por ejemplo cuando creamos un **struct**:

```
typedef struct Books
{
    char title[37];
    char author[37];
}Book;
```

- e) Eplique el tipo de datos **enum** y proponga algunos ejemplos

enum: Nos permite crear nuestros propia lista de simbolos que estan relacionados de alguna manera, por ejemplo estados de un sistema:

```
typedef enum{GREEN, YELLOW, RED}state; // define the states
```

- f) Explique que es una estructura, como se define, como se declaran variables y punteros a estructuras. Como se acceden al campo interno de las mismas utilizando punteros?.

Las estructuras son agrupamientos de datos definidos por el usuario. Se definen:

```
struct Mistruct
{
    int a;
    float b;
    char c;
};
```

Se accede a los campos de las estructuras con punteros de la siguiente manera: Mistruct->a=0; con el operador ->

Modularizacion por clases

- a) Explique que entiende por modularizacion de software y que ventajas tiene

La modularizacion de software es una manera de organizar el codigo que hace que sea mas eficiente , portable, menos propenso a errores.

- 4) nvic: Nested vectored Interrupt Controller