Programación en C para sistemas embebidos

Arquitectura 2023

Ventajas de programar en C

Posee las ventajas del diseño "Top-Down":

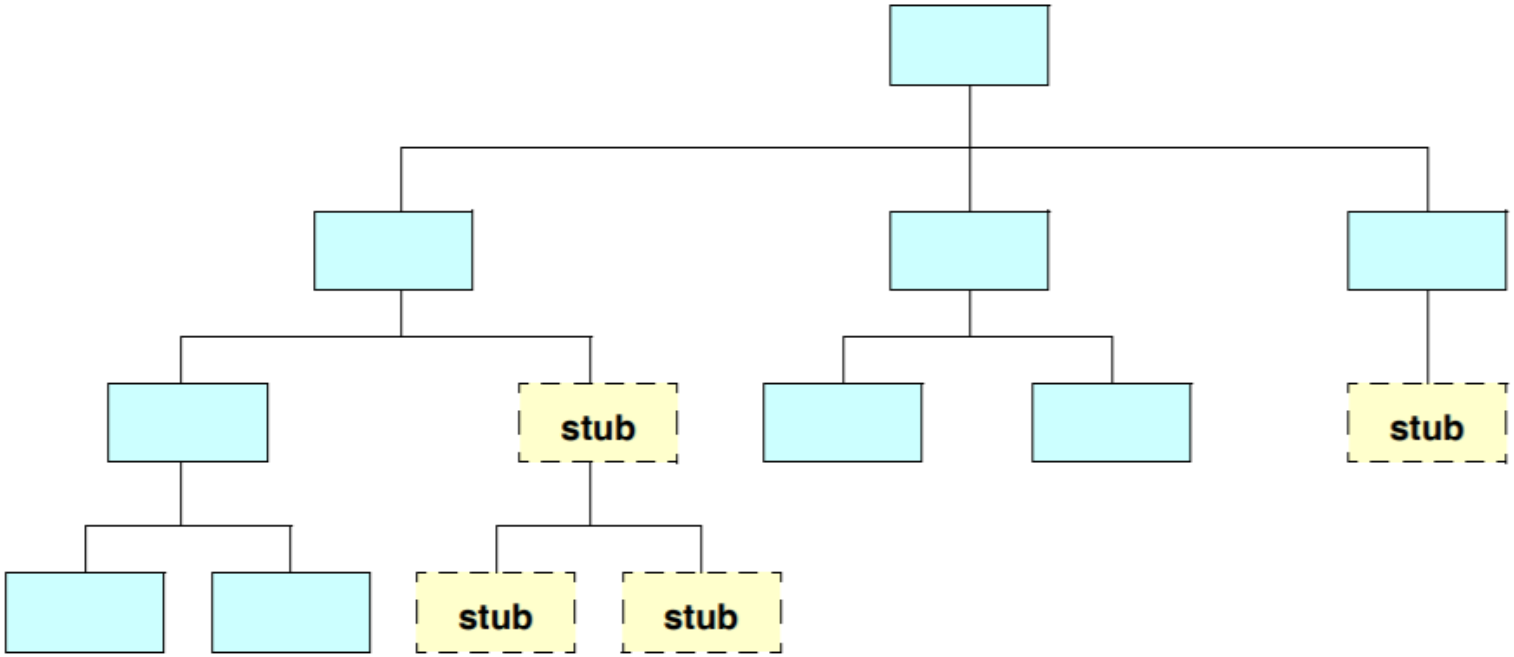
-Enfoque orientado a la problemática (No es necesario que el desarrollador conozca el funcionamiento interno del sistema).

-Tiempos de desarrollo reducidos.

-Se simplifica la detección de errores.

-Facilita el seguimiento del desarrollo

División de un problema



Objetivos de la Programación en C

- Fácil de entender
- Fácil de depurar
- Fácil de verificar
- Fácil de mantener

Regla de Oro: **“Escribe software para otros como te gustaría que lo escriban para vos.”**

Diseño de Código Reusable

- Código Documentado
- Abstracción
- Modularidad
- Software en capas

Código documentado

- *Comentar el código sirve para entender programas escritos con mucha anterioridad*
- *Sirve para que otro pueda modificar nuestro código*
- *Existen software que en base al código te realizan comentarios automáticamente.*

Código documentado

En el Mantenimiento de software se requiere:

- Corrección de errores,
- Nuevas características,
- Optimizaciones de velocidad de ejecución o uso de memoria,
- Migración hacia otro hardware,
- Adaptación a distintas situaciones.

Código Documentado

Error Común:

- X = X + 4; /* sumo 4 a X */
- Flag = 0; /* pongo flag en cero */

Comentar cada línea del programa impide leer la función real que cumple el código

- •X = X + 4; /* Se suman 4 (mV) para corregir el offset del transductor */
- •Flag= 0; /* Significa que no se presionó ninguna tecla*/

Niveles de Abstracción

Modularidad

- La modularidad consiste en dividir un programa en bloques funcionales.
- Cada función cumplirá una tarea específica e indivisible.
- Esto es importante debido a que permite dividir un problema complejo es pequeños problemas.
- De esta manera, varios programadores podrían trabajar en el mismo problema a la vez y se aceleraría el tiempo de desarrollo de un sistema.

MODULARIDAD

Razones:

- Abstracción funcional: Permite reutilizar módulos de software desde múltiples lugares
- Abstracción en complejidad: Dividir un sistema complejo en componentes pequeños y simples.
- Portabilidad: Permite programar un mismo código en diferentes plataformas de HW.

Ejemplo de modularidad

- Supongamos que tenemos que diseñar una estación meteorológica que lee los datos de temperatura, humedad y presión desde 3 sensores diferentes y los almacena en una memoria externa. Además, supongamos que la estación debe transmitir los datos leídos desde los sensores a un servidor mediante un puerto serie asincrónico.
- Para la programación de este sistema deberíamos dividir la aplicación en pequeños módulos según las siguientes funciones:

- Lectura de temperatura
- Lectura de humedad
- Lectura de presión
- Almacenamiento en memoria
- Menú de comandos
- Transmisión y recepción de comandos

Modularidad y reusabilidad

- Estos módulos serán reutilizables, ya que si se requiere incorporar un sensor de temperatura a una aplicación, simplemente se debe

llaman a la porción de código que lee el sensor de temperatura. Ahora, para para poder reutilizar esos módulos en una plataforma diferente se deben seguir una serie de pasos.

1. En primer lugar se deberá crear un archivo con extensión .c por cada módulo definido. En estos archivos se escribirán las funciones que cumplan con la tarea definida por el módulo. Es indispensable no utilizar nombres de registros o direcciones que hagan alusión a una tecnología de microcontrolador para que los archivos .c sean completamente portables a otras plataformas.

Modularidad y reusabilidad

- Temperatura.c
  - Leer_temp ()
- Humedad.c
  - Leer_humedad ()
- Presion.c
  - Leer_presion ()
- d. Memoria.c
  - escribir_memoria(datos, direccion)
  - leer_memoria(datos, direccion)
  - Borrar_memoria(sector)
- e. Menú.c
  - Almacenar_sensor(sensor, destino)
  - Leer_sensor(sensor)
  - Transmitir_sensor(sensor)
- f. Comunicación_serie.c
  - Inicializar_modulo_serie()
  - Recibir_dato_serie()
  - Transmitir_dato_serie()
  - Dato_disponible()

Modularidad y reusabilidad

2. Los archivos .c van acompañados de un archivo .h (header) con el mismo nombre. El objetivo de los archivos .h es lograr la portabilidad del código, es decir, que el archivo .h contenga nombres de registros o características que lo vinculen a una tecnología de microcontrolador y el .c solo código puro.

- En cada .h se definirán:
  a. Mediante #define se definen nombres que representan la funcionalidad y utilizan algún registro. Ejemplo,
     a. #define temperatura ADR
     b. #define temp_disponible ADSCR_COCO
  b. También se pueden definir constantes como por ejemplo
     a. #define lectura_humedad 1
     b. #define lectura_presion 2
     c. #define lectura_temperatura 3
  c. Por último se deben declarar todas las funciones que se definirán en el archivo .c

Programación en capas o niveles

Reglas:

- Un módulo puede hacer una llamada a otro módulo en la misma capa
- Un módulo puede llamar a un módulo de una capa inferior solo utilizando la API
- Un módulo no puede acceder directamente a ninguna función o variable en otra capa (sin utilizar la API)
- Un módulo no puede llamar a una rutina de mayor nivel.

- Nota:API (application program interface)

Programación en capas o niveles