FUNCIONES.

Para que un programa en C sea más fácil de manipular y entender, es recomendable que el programa se estructure en forma modular, esto se puede realizar a través de funciones, las cuales, cada una de ellas tenga una actividad que realizar, diferente de las demás.

Las ventajas de trabajar en forma modular son:

1. El programa esta compuesto de pequeños programas que se pueden analizar cada uno de forma más fácil y de manera independiente.
2. Si existe un error, este se indica en la función que lo contiene.
3. Se permite la reutilización de código.
4. Permite una administración de memoria más eficiente por parte del programa.

Las funciones con valor, regresan por medio de la palabra reservada “return” un resultado que puede ser un valor o una dirección de un tipo de dato básico o bien, una dirección de un nuevo tipo de dato generado por el programador. Un ejemplo de retorno de valores del tipo básico se presenta a continuación:

```
char nom_func1(parámetros de entrada)

int nom_func2(parámetros de entrada)

float nom_func3(parámetros de entrada)

double npm_func4(parámetros de entrada)
```

Como se observo en los ejemplos anteriores, para crear una función de programador, se requiere que se de un nombre a la función, un tipo de valor y los parámetros de entrada. Los parámetros de entrada son la información requerida por la función para que se pueda evaluar y genere un resultado.

```
void nom_funcion(parámetros de entrada)
```

La función anterior no retorna valor por ser de tipo “void”.

Para que se pueda construir una nueva función, se requiere de tres cosas:

1. Declarar la función entre los archivos de cabecera y el inicio de la función “main”, indicando el tipo de parámetros de entrada.
2. Crear el contenido de la nueva función, el cual se localizará en cualquier parte después de terminada la función “main”.
3. utilizarla en cualquier función, incluyendo a ella misma y a “main”. En donde se utilice la nueva función, deberá especificarse el número de argumentos que deben de ser iguales en cantidad que el número de parámetros.

Paso de parámetros por valor y referencia.

- Cuando se pasa una copia del valor de una variable a una función esto se conoce como, un paso de parámetro por valor.

- Cuando lo que pasa como información a la función, no es un valor sino una dirección de la variable que necesita la función, esto se conoce como un paso de parámetro por referencia.

- Al conocer las direcciones de estas variables se podrá tener acceso al contenido de estas direcciones y podrán modificarse los valores. No así si solo se conocen los valores (paso de parámetros por valor).

- Para llamar a una función y pasar un parámetro por referencia se utiliza el operador unario &. nom_funcion(&variable);

- Si un arreglo de dos dimensiones se pasa a una función, la declaración de parámetros en la función debe incluir el número de columnas; el número de renglones es irrelevante, puesto que lo que pasa es un apuntador a un arreglo de renglones, donde cada renglón contiene un arreglo de n valores.

  Ejemplo: si el arreglo

  ```
  int diasmes[2][13]={
  {0,31,28,31,30,31,30,31,31,30,31,30,31},{0,31,29,31,30,31,30,31,31,30,31,
  30,31},
  };
  ```

  Se pasa a una función f. La declaración puede ser como sigue:

  ```
  f(int diasmes[2][13]){...}

  f(int diasmes[][13]){...}

  f(int (*diasmes)[13]){...}
  ```

que indica que el parámetro es un apuntador a objetos (renglones), donde cada objeto es un arreglo de 13 enteros. Los paréntesis son necesarios, puesto que los corchetes [ ] tienen más alta precedencia que el operador de indirección (*). Sin paréntesis la declaración

int *diasmes[13]

indica que es un arreglo de 13 apuntadores a enteros.