ABSTRACCION DE CODIGO

Abstraer (R.A.E.):

*tomar un rasgo o cualidad de algo para analizarlo en su pura escencia y noción

*tomar algo y aislarlo de su contexto

*Generalizar

Abstracción Informática: concentrarse en «¿Qué hace el código?» más que en el «¿Cómo lo hace?»

EJEMPLO

Ver el programa ANSI C disponible en:

http://www.grch.com.ar/docs/poo/2020/espagueti/espagueti.c

Observe su código, puede descargarlo y compilarlo como (MingW/Cygwin Windows):

```
$ gcc –Wall –o espagueti.exe espagueti.c
$ espagueti.exe
```

ANALISIS DE CODIGO

Conceptualmente, ¿Qué hace este código?

Identificar líneas de código repetitivas

Identificar líneas de código relacionadas para hacer una tarea específica

Ejemplo de sección de código

```
char scodigo[10];
int ncodigo;
printf("Ingrese Codigo:");
fgets(scodigo,10,stdin);
ncodigo = atoi(scodigo);
```

¿Qué Hace este código?

ingreso_entero

¿Cuál es su «entrada»?

¿Cuál es su «salida»?

¿Cómo podría usarlo?

int ncodigo = ingreso_entero(«Ingrese Codigo»,10)

int ingreso_entero(char *titulo,int largo)

¿Cuál es su «prototipo»?

Implementación de Función

```
int ingreso_entero(char *titulo,int largo)
{
    char scodigo[largo];
    int ncodigo;
    printf(«%s»,titulo);
    fgets(scodigo,largo,stdin);
    ncodigo = atoi(scodigo);
    return ncodigo;
}
```

Utilizo parte del código del programa para implementar la función

Reemplazo esas líneas de código del programa por la llamada a la función:
`int ncodigo = ingreso_entero(«Ingrese Codigo»,10);`

Uso la función ingreso_entero() en TODO el programa

Antes de la función main() escribo el prototipo de la función

Después de la función main() escribo el código de la función

GENERALIZACIÓN DE FUNCIONES

¿Qué otros ingresos por teclado hay?

¿Son similares?

¿Qué tienen en común?

¿Cuál es la lógica subyacente?

Por ejemplo, si implemento la función ingreso_double(), seguramente va a ser muy similar a ingreso_entero()

GENERALIZACIÓN DE FUNCIONES

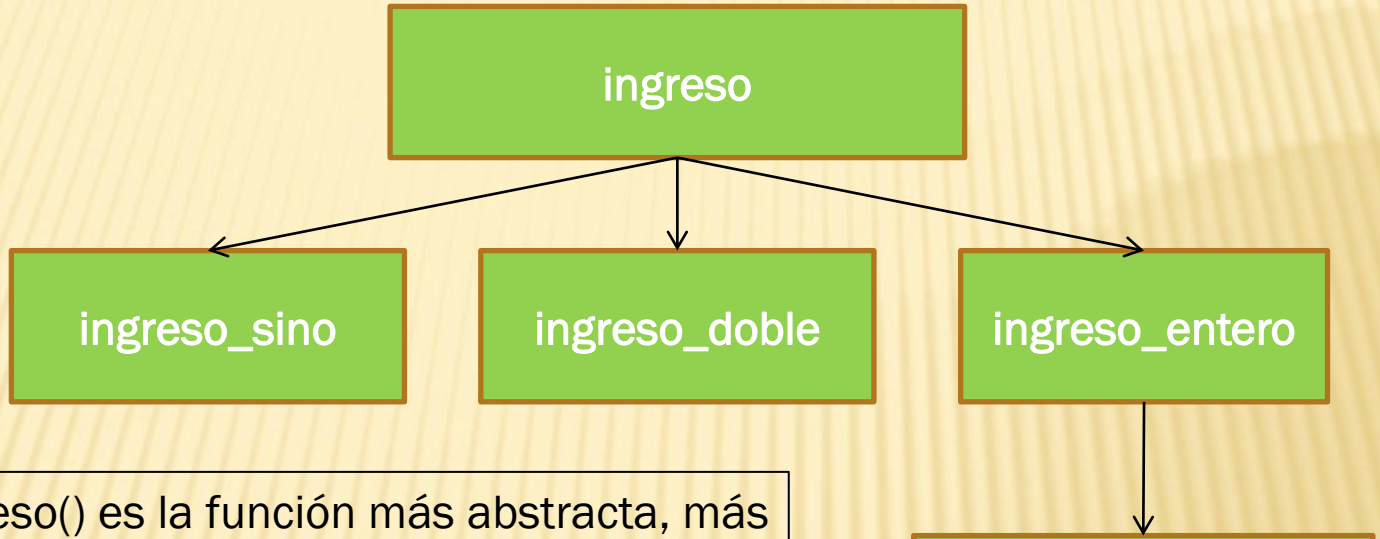
Todas hacen lo mismo:

Estan basadas en `fgets()`

En realidad ingresan una cadena de caracteres por teclado y luego convierten esa cadena en **`int`**, **`double`**, **`float`**, etc. si fuese necesario

Entonces, ¿Qué puedo hacer?

GENERALIZACIÓN DE FUNCIONES



ingreso() es la función más abstracta, más genérica, más simple y potente: simplemente ingresa unos caracteres por teclado

ingreso_entero() llama, usa, a función ingreso() y lo que ésta le devuelve lo convierte a int

ingreso_enterodh() llama, usa, a función ingreso_entero() y lo que ésta le devuelve, verifica que el valor sea >= desde y <= hasta

IMPLEMENTACIÓN DE ingreso() e ingreso_entero()

```
void ingreso(char *titulo,int largo,char *buffer){
   printf("%s",titulo);
   fgets(buffer,largo,stdin);
}

int ingreso_entero(char *titulo,int largo) {
   char teclas[largo];
   ingreso(titulo,largo,teclas);
   return atoi(teclas);
}
```

IMPLEMENTACIÓN DE ingreso_enterodh()

```
int ingreso_enterodh(char *titulo,int largo,int desde,
int hasta) {
   int nro;
   do {
      nro = ingreso_entero(titulo,largo);
   } while(nro < desde || nro > hasta);
   return nro;
}
```

AGRUPACION DE FUNCIONES

Una vez que hemos abstraído un conjunto de funciones a partir de un código de programa espagueti, no modular, una vez que hemos establecido una serie de relaciones y reúso de código entre las funciones: AGRUPAR LAS FUNCIONES por FUNCIONALIDAD (Para este ejemplo, se deberían agrupar las funciones de manejo de consola por un lado y las funciones de manejo de archivos de acceso directo por otro)

AGRUPACION DE PROTOTIPOS

Los prototipos de las funciones de consola se guardaran en consola.h

Quitar todos los prototipos del programa principal y antes de main() agregar:

#incluce <consola.h>

AGRUPACION DE FUNCIONES

La implementación de las funciones (su código) de consola se guardaran en consola.c y se quitan del programa principal

En consola.c, agregar todos los #include's que se necesiten, mas:

#include <consola.h>

COMPILAMOS Y CONSTRUIMOS LIBRERIA

Compilamos la librería consola de la siguiente forma:

```
$ gcc -Wall -I . -c consola.c
```

(-I es «i latina en mayúsculas, de directorio de Include's)

Esto crea objeto compilado consola.o que usamos para crear libería estática:

```
$ ar rcs libconsola.a consola.o
```

Luego compilar programa principal con librería:

```
$ gcc -Wall -I . -L . -o espagueti.exe espagueti.c -lconsola
```

(-I es «i latina en mayúsculas, de directorio de Include's)
(-L es «ele en mayúsculas, de directorio de Librerias)
(«. punto» Se refiere al directorio actual)

QUE HEMOS LOGRADO

Pasar de un código espagueti, repetitivo, no mantenible, no reusable, extenso, confuso, etc. a un código modular, no repetitivo, mantenible, reusable, más corto y que hace exactamente lo mismo lo que anterior , pero ha extraído y aprovechado la experiencia, construyendo código de librería reusable y aprovechable en otros proyectos evitando tener que escribir una y otra vez el mismo código.

Hay dos cosas para distribuir: espagueti.exe y la librería consola (consola.h, libconsola.a + manual de la liberia), conservando nuestra propiedad intelectual (consola.c, espagueti.c)

CODIGO EJEMPLO PROGRAMA espagueti

En la carpeta:

http://www.grch.com.ar/docs/poo/2020/espagueti/

Esta el código espagueti.c original, el cual fue sometido a un proceso de abstracción y paso a paso fui transformando el código hasta extraer del mismo dos librerías: una para manejo de consola y otra para manejo de archivo de acceso directo. Cada paso se documenta en una subcarpeta enumerada de 1 a 11.