UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS Y ALGORITMOS I (1227)

Profesor: ***M.I. Marco Antonio Martínez Quintana***

Semestre 2021-2

Actividad Asíncrona #1 Viernes 26 de Febrero

Nombre del alumno: Cadena Luna Iván Adrián

Grupo: 15

Fecha: (10/03/2021)

- Realizar un acordeón del Lenguaje C con lo visto en las prácticas de Laboratorio de Fundamentos de Programación.

UN PROGRAMA ELEMENTAL

```c
#include <stdio.h>
main() {
 printf("Hola");
}

/*
Programa de Ejemplo
Fecha_
Autor_
*/
#include ____
#define ____
typedef ____
[Prototipos]
int main(void)
{
[variables] /* descripcion */
[instrucciones]
return 0;
}
```

TIPOS DE DATOS BÁSICOS

- Números enteros: **int**
- Números reales: **float** (simple precisión) y double (doble precisión)
- Caracteres: **char**
- Operaciones habituales: suma (+), resta (-), multiplicación (*), división (/), resto de la división o "módulo" (%).
- Operaciones abreviadas: incremento en una unidad (++), decremento en una unidad, incremento en varias unidades (x+=2), decremento en varias unidades (z-=3) y abreviaturas similares para las demás operaciones (x*=5; z/=4; y%=2;).

ESTRUCTURAS BÁSICAS

if

- Comprueba una condición.
- Ejemplo: if (x==5) printf("Vale 5");
- Tipos de condiciones posibles: Mayor que (>), menor que (<), mayor o igual que (>=), menor o igual que (<=), distinto de (!=), igual a (==).
- Si no se cumple la condición: else -> if (x==5) printf("Vale 5"); else printf("No vale 5");
- Para enlazar varias condiciones: y (&&), o (||), no (!): if ((x==5) && (y==1)) printf("Son 5 y 1");

while

- Repite mientras se cumpla una condición (la condición se comprueba antes de empezar a repetir).
- Ejemplo: while (x!=correcto) { printf("Vuelve a intentar"); scanf("%d", &x); }
- Tipos de condiciones y forma de enlazarlas: igual que para "if".

do-while

- Repite mientras se cumpla una condición (la condición se comprueba después de haber dado la primera "vuelta").
- Ejemplo: do {printf("Vuelve a intentar"); scanf("%d", &x); } while (x!=correcto);
- Tipos de condiciones y forma de enlazarlas: igual que para "if".

MANEJO BÁSICO DE PANTALLA Y TECLADO

printf

- Escritura formateada en pantalla
- Ejemplo: printf("El resultado es %d", x);
- Formatos más habituales: %d = número entero en decimal, %f = número real (coma flotante), %c = caracter, %s = cadena de texto, %x = número entero en hexadecimal
- Devuelve: el número de caracteres escritos

scanf

- Lectura formateada desde teclado
- Ejemplo: scanf("%d", &x);
- Formatos más habituales: similares a "printf"
- Devuelve: el número de datos leidos (0 = ninguno, EOF = error)
- Observaciones: en general, el nombre de la variable se debe preceder de & (no es necesario si se trata de un array)

BUCLES

Bucle **for**

```
for(inicialización, condición, instrucción_final)
    {
    [instrucciones]
}

Ejemplo: for(i=0; i<10; i++)
```

Bucle **while**

```
while (condición) {
    [instrucciones]
}
```

Bucle **do-while**

```
do {
    [instrucciones]
} while(condición);
```

Bucle **if**

```
case 1:
if (condición) {
    [instrucciones]
}

case 2:
if (condición) {
    [instrucciones_1]
} else {
    [instrucciones_2]
}

case 3:
if (condición_1) {
    [instrucciones_1]
} else if (condición_2) {
    [instrucciones_2]

...

} else if (condición_n) {
    [instrucciones_n]
} else {
    [instrucciones]
}
```

SINTAXIS DEL SWITCH

```
switch(expresión_entera) {
case constante_1:
    [instrucciones_1]
    break;
case constante_2:
    [instrucciones_2]
    break;
...
case constante_3:
    [instrucciones_3]
    break;
default:
    [instrucciones]
}
```

CADENAS

```
//Lectura:

scanf("%s",cadena);
//lee una palabra

gets(cadena);
//lee una frase hasta fin de linea

fgets(cadena, N, stdin);
/*lee una frase con control de tamaño.
También lee \n */


//Escritura:

printf("%s",cadena);
/*escribe una cadena por pantalla,
vale para frase o palabra */
```

ESTRUCTURAS

```
//Declaración de un tipo estructura
typedef struct persona {
    char nombre [N];
    int edad;
    long dni;
} PERSONA;

//Declaración de variables
PERSONA p; //una estructura
PERSONA *pp; //puntero a estructuras
PERSONA vec[20]; //vector de estructuras

//Acceso a los miembros
p.edad=27;
pp->edad=30;
vec[7].edad=37;
```

MANEJO DE FICHEROS

Para manejar ficheros, siempre deberemos realizar tres operaciones básicas:

- Abrir el fichero.
- Leer datos de él o escribir datos en él.
- Cerrar el fichero.
- Para declarar fichero: **FILE* fichero;**
- Para abrir un fichero: **fichero = fopen(nombreFichero, modo);** Los modos son:
  - **r** Abrir sólo para lectura.
  - **w** Crear para escribir. Sobreescribe el fichero si existiera ya (borrando el original).
  - **a** Añade al final del fichero si existe, o lo crea si no existe.
  - **+** Se escribe a continuación de los modos anteriores para indicar que también queremos modificar. Por ejemplo: r+ permite leer y modificar el fichero.
  - **t** Abrir en modo de texto.
  - **b** Abrir en modo binario.
- Obtener un carácter del fichero: **caracter = fgetc(fichero)**
- Escribir un carácter en el fichero: **fputc(caracter,fichero)**
- Escribir datos formateados en el fichero: **fprintf(fichero,"formato", argumento1, argumento2...)**
- Leer de fichero: **fscanf(fichero, "formato", argumento1,....)**
- Cerrar fichero: **fclose(fichero)**
- Distinto de 0 si acaba el fichero: **feof(fichero)**
- Leer linea en cadena: **fgets(cadena, cantidad , fichero)**
- Escribir cadena: **fputs(cadena, fichero)**
- Saltar a una posición: **fseek(fichero, salto, desde)** (el "desde" puede ser: SEEK_SET, SEEK_CUR, SEEK_END, para principio, actual o final del fichero

- Investigar un lenguaje de programación cuya inicial sea la misma que la de su nombre, y si no existe, pueden usar una de sus apellidos.

Lua / El nombre significa «**luna**» en portugués.

Lua es un lenguaje de programación homónimo que presenta varias particularidades, como su extensibilidad independientemente de la plataforma, su pequeño tamaño y su alta velocidad. Sin, embargo, raras veces se usa como lenguaje de programación independiente: los desarrolladores suelen emplearlo como lenguaje de *scripting* integrado en programas individuales. Sus principales ámbitos de aplicación siempre han sido los videojuegos y los motores de juego, pero Lua también se utiliza para desarrollar una gran cantidad de programas de redes y sistemas.

Lua consiste básicamente en una biblioteca que los programadores pueden incorporar en su software para volverlo programable.

Comentarios

```
--Esta linea es comentario ya que empieza con doble guion
```

Saltos de línea

En Lua, los saltos de línea se añaden con “\“ o “\n“ y un salto de línea real:

```
1  print ('1 : Hello\
2          World')
3  print ("2: Hello\
4          World")
```

Se ejecuta la función print, la cual recibe un parámetro que es el string "hello world". En este caso, el resultado de salida tiene el siguiente aspecto:

```
1  1: Hello
2  World
3  2: Hello
4  World
```

Variables

No es obligatorio declarar variables en el lenguaje de programación Lua, sino que estas solo se crean si es necesario. Cuando se le asigna el valor inicial la misma comienza a existir.

Como recordatorio, los valores pueden ser de los siguientes tipos: "**nil**" es decir nulo (a la par que el valor de las variables no creadas), **números**, **literales** (caracteres, letras, palabras, etc.), **booleanos** (verdadero/falso o true/false), **tablas** y **funciones**.

Un ejemplo de codificación simple sería el siguiente:

```
1  print ( var )
2  var = "Hello World"
3  print ( var )
```

Como las variables indefinidas resultan en nulo ("nil"), el resultado es el siguiente:

```
1  nil
2  Hello Worldprint ( var )
```

Operaciones matemáticas

Como todo lenguaje de programación, Lua soporta operaciones matemáticas: +, -, *,/

```
1  numerador = 4*3
2  denominador = 5 - 2 + 7
3  print(numerador / denominador)
```

Condicionales

Los valores true y false permiten la ejecución condicional:

```
1  if true then
2     print("es cierto")
3  end
```

```
1  if true then
2     print("es cierto")
3  else
4     print("esto no se tiene que imprimir")
5  end
```

Adicionalmente se pueden usar operadores de comparacion: >, >=, <, <=, == y operadores lógicos: and, or y not

```
1  miValor = 20
2  if miValor >= 0 and miValor <= 10 or miValor == 20 then
3     print(miValor .. " esta entre 0 y 10 o es 20")
4  end
```

Ciclos

Para contar se usa el ciclo **for**:

```
1  --todos los valores (cuenta de 1 en 1)
2  for valor = 0, 10 do
3   print(valor);
4  end
5
6  --Solo los pares (cuenta de 2 en 2)
7  for valor = 0, 10, 2 do
8   print(valor);
9  end
```

Para ciclar por una condición se puede usar **while**:

```
1  valor = 0
2  while valor < 10 do
3   print(valor)
4   valor = valor + 1
5  end
```

Funciones

Para reutilizar código se pueden hacer funciones. Esto evita cortar y pegar el código y generar programas más prolijos. Una función puede recibir parámetros y devuelve un valor.

Función sin parámetros que **no devuelve nada**:

```
1  --aca defino la funcion
2  function helloWorld()
3   print ("Hola mundo")
4  end
5
6  --aca uso la funcion
7  helloWorld()
```

Función con parámetros que **devuelve un elemento**:

```
1  --aca defino la funcion
2  function minimo(a,b)
3   if a < b then
4   return a
5   else
6   return b
7   end
   end

   --aca uso la funcion con distintos parametros
   print(minimo(5,7))
   print(minimo(9,4))
```

- **BIBLIOGRAFÍA CONSULTADA**

Zambrano, A., Gassman L. (s.f.). Ejemplos en lua muy básicos. Recuperado el 07 de marzo de 2021, de https://sites.google.com/site/tvdunq/home/material/ejemplos/ejemplos-lua-muy-basicos

Desconocido (2020). ¿Qué es Lua?: una introducción al lenguaje de programación. Recuperado el 07 de marzo de 2021, de https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/que-es-lua/