UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERIA

ESTRUCTURA DE DATOS Y ALGORITMOS I

ACTIVIDAD ASÍNCRONA #1

ALUMNA: RIOS HERRERA ELISA DANIELA

VIERNES 26 DE FEBRERO DE 2021

Acordeón: Lenguaje C

FUNCIÓN / ACCIÓN	SINTAXIS
Lectura de datos (#include<stdio.h>)	scanf("Formato",&identificador); scanf("Formato,Formato",&ident,&ident);
Impresión de datos almacenados en un identificador (#include<stdio.h>)	printf("Formato",&identificador);

If else (#include<stdio.h>)	**if**(condición)
	{
	sentencias
	}
	else
	{
	sentencias
	}

Switch (#include<stdio.h>)	**switch**(identificador)
	{
	case valor1:
	sentencias
	break;
	case valor2:
	sentencias
	break;
	case valorN:
	sentencias
	break;
	default:
	sentencias
	break;
	}

for (#include<stdio.h>)	for(i=0;i<=1;i++)

while (#include<stdio.h>)	**while**(condición)
	{
	Instrucciones
	}

do-while (#include<stdio.h>)	**do**
	{
	Instrucciones
	}
	while(condición)

Toupper (mayúsculas) y **Tolower** (minúsculas) (#include<ctype.h>)	IAC=**toupper**(valorCarácter); IAC=**tolower**(valorCarácter);
Números aleatorios (#include<stdlib.h>) y (#include<time.h>)	**srand**(time(NULL)); identificador=rand()%N; identificador=(rand()%N)-M
Arreglo estático unidimensional numérico (#include<stdio.h>)	tipo Arreglo[Tamaño];
Almacenar un valor en un arreglo (#include<stdio.h>)	**scanf**("EF",&Arreglo[posición]);
Imprimir un valor en un arreglo (#include<stdio.h>)	**printf**("EF",Arreglo[posición]);
Arreglo estático unidimensional alfanumérico (#include<stdio.h>)	tipo Arreglo[Tamaño+1];
Almacenar un valor en un arreglo alfanumérico: un carácter (#include<stdio.h>)	**scanf**("%s",&Arreglo[posición]);

Almacenar un valor en un arreglo alfanumérico: una frase (#include<stdio.h>)	**scanf**(" %[^\n]",&Arreglo); **fflush**(stdin); **gets**(Arreglo);
Almacenar un valor en un arreglo alfanumérico: una palabra (#include<stdio.h>)	**scanf**("%s",&Arreglo);
Imprimir un carácter de un arreglo (#include<stdio.h>)	**printf**("%s", Arreglo[posición]);
Imprimir una frase de un arreglo (#include<stdio.h>)	**printf**("%s", Arreglo);
Iniciar un arreglo numérico (#include<stdio.h>)	tipo Arreglo[]={valor1,valor2,...,valorN};
Iniciar un arreglo alfanumérico (#include<stdio.h>)	**char** Arreglo[]="cadena"; **char** Arreglo[]={'c','c','c','c','c',...,**0**}; **char** Arreglo[]={'c','c','c','c','c',...,**'\0'**};
Contabilizar los caracteres de un arreglo incluidos espacios (#include<string.h>)	IE=**strlen**(Arreglo);
Comparar 2 cadenas (#include<string.h>)	IE=**strcmp**(Arreglo1,Arreglo2);
Concatenar 2 cadenas (#include<string.h>)	**strcat**(Arreglo1,Arreglo2);
Copiar en otro arreglo lo que ya hay en uno (#include<string.h>)	**strcpy**(Arreglo1,Arreglo2);
Convertir el contenido de un arreglo a MAYÚSCULAS – IDE (#include<string.h>)	**strupr**(Arreglo);
Convertir el contenido de un arreglo a MINÚSCULAS – IDE (#include<string.h>)	**strlwr(**Arreglo);
Declarar arreglo multidimensional (#include<stdio.h>)	tipo Arreglo[Renglones][Columnas];
Almacenar valores en un arreglo multidimensional: un carácter (#include<stdio.h>)	**scanf**("%s",&Arreglo[posiciónRenglón][posiciónColumna];
Almacenar valores en un arreglo multidimensional: un número (#include<stdio.h>) **Imprimir con printf y misma sintaxis**	**scanf**("EF",&Arreglo[posiciónRenglón][posiciónColumna];
Operaciones con arreglos (#include<stdio.h>)	Identificador=Arreglo+Arreglo; Arreglo=**sqrt**(Arreglo);
Iniciar arreglo estático multidimensional numérico (#include<stdio.h>)	tipo Arreglo[][Columnas]={{valor1,valor2,...,valorN},{valor1, valor2,...,valorN}};
Iniciar un arreglo estático multidimensional alfanumérico (#include<stdio.h>)	**char** Arreglo[][Columnas+1]={{' c','c','c','c','c',...,**0**},{' c','c','c','c','c',...,**0**}};

CON parámetros y REGRESA un valor (#include<stdio.h>)	**void** Función(tipo parámetro1,tipo parámetro2,...,tipo
	parámetroN)
	{
	Sentencias
	return valor;
	}

CON parámetros y NO REGRESA un valor (#include<stdio.h>)	**void** Función(tipo parámetro1,tipo parámetro2,...,tipo
	parámetroN)
	{
	Sentencias
	}

SIN parámetros y REGRESA un valor (#include<stdio.h>)	`tipo Función(void);`
	`{`
	`Sentencias`
	`return valor;`
	`}`

SIN parámetros y NO REGRESA un valor (#include<stdio.h>)	`void Función(void)`
	`{`
	`Sentencias`
	`}`

Enumeraciones (#include<stdio.h>)	`enum Nombre{ID1,ID2,...,IDN};`
Declarar un apuntador (#include<stdio.h>)	`tipo *Apuntador;`
Iniciar un apuntador (#include<stdio.h>)	`Apuntador=&ID;`
Imprimir el valor de un apuntador (#include<stdio.h>)	`printf("%p",Apuntador);` `printf("%p",ID);`
Cambiar el contenido de una variable por medio de un apuntador (#include<stdio.h>)	`*Apuntador=valor;`
Declarar un apuntador de tipo archivo (#include<stdio.h>)	`FILE *apuntador;`
Abrir archivo (#include<stdio.h>) **Lectura**: r, r+t **Escritura**: w, w+t, a+t	`apuntador=fopen("archivo","acción");`

Archivo: leer un caracter (#include<stdio.h>)	`fgetc(apuntador);`
	
	`ID= fgetc(apuntador);`

Archivo: leer frases o palabras (#include<stdio.h>)	`fgets(arregloAlfa,tamañoArregloAlfa,apuntador);`
Archivo: leer cualquier tipo de dato numérico (#include<stdio.h>)	`fscanf(apuntador,"EF",&ID);`
Archivo: escribir un carácter (#include<stdio.h>)	`fputc(variableAlfa,apuntador);`
Archivo: escribir cadenas (#include<stdio.h>)	`fputs(arregloAlfa,apuntador);`
Archivo: escribir cualquier tipo de dato num o alfanum (#include<stdio.h>)	`fprintf(apuntador,"Texto EF",ID);`
Cerrar archivo (#include<stdio.h>)	`fclose(apuntador);`

Archivo: leer todos los renglones, no tiene fin (#include<stdio.h>)	`while((scanf(apuntador,"EF",&ID))!=EOF)`
	`{`
	`Sentencias`
	`}`

Archivo: leer todos los renglones, con o sin fin (#include<stdio.h>)	`while(feof(apuntador)==0)`
	`{`
	`Sentencias`
	`}`

Convertir un apuntador en un arreglo dinámico (#include<stdlib.h>)	`apuntador=(tipo*)malloc(tamaño*sizeof(tiempo));`
Regresar arreglo a apuntador (#include<stdlib.h>)	`free(apuntador);`

Declarar un apuntador del tipo del arreglo (#include<stdlib.h>)	tipo **apuntador;
Convertir un apuntador en un arreglo dinámico y crea renglones (#include<stdlib.h>)	apuntador=(tipo**)malloc(Renglones*sizeof(tipo*));

Convertir un apuntador en un arreglo dinámico y crea columnas (#include<stdlib.h>)	**for**(identificador=0;identificador<Renglones;identificador=
	++)
	{
	apuntador[identificador]=(tipo*)malloc(Columnas*sizeof(t
	ipo));
	}

Regresar arreglo a apuntador (#include<stdlib.h>)	**for**(identificador=0;identificador<Renglones;identificador=
	++)
	{
	free(apuntador[identificador]);
	}

Declarar una estructura (#include<stdio.h>)	**struct** estructura
	{
	tipo miembro1;
	tipo miembro2;
	tipo miembroN;
	};

Declaración de variable tipo estructura: local (#include<stdio.h>)	**struct** estructura variable1,variable2,variableN;
Declaración de una variable tipo estructura: global (#include<stdio.h>)	**struct** estructura { tipo miembro1; tipo miembro2; tipo miembroN; }variable1,variable2,variableN;
Declaración de un arreglo tipo estructura: global (#include<stdio.h>)	**struct** estructura { tipo miembro1; tipo miembro2; tipo miembroN; }arreglo1[tamaño],arreglo2[tamaño],arregloN[tamaño];
Declaración de un arreglo tipo estructura: local (#include<stdio.h>)	**struct** estructura arreglo1[tamaño],arreglo2[tamaño],arregloN[tamaño];
Emplear un identificador de tipo estructura, guardar datos si es variable (#include<stdio.h>)	**scanf**("EF",&variable.miembro);
Emplear un identificador de tipo estructura, guardar los datos si es arreglo (#include<stdio.h>)	**scanf**("EF",&arreglo[posición].miembro);
Emplear un identificador de tipo estructura: mostrar los datos si es variable (#include<stdio.h>)	**printf**("EF", variable.miembro);
Emplear un identificador de tipo estructura: mostrar los datos si es arreglo (#include<stdio.h>)	**printf**("EF",arreglo[posición].miembro);

Acordeón: Lenguaje Eiffel

Variables

TIPO	SIGNIFICADO	LITERALES	OPERADOR	POR DEFECTO
Boolean	Valor lógico	true, false	Lógicos	False
Character	Caracteres	'a' , '%N'	De relación	'%U'
Integer	Números enteros	-123	+,-,*,^,//,\\	0
Double	Números reales	123.45	+,-,*,^,/	0.0

Expresiones y operadores

TIPO	OPERADORES	VALORES	SE OBTIENE
Aritméticos	+ - *	Integer, double	Mismo tipo
	// para cociente \\ residuo	Integer	Integer
	/ división	Integer, double	Double
	^ potencia	Base: integer, double Exponente: integer:	Mismo tipo que la base
De relaciones	=, /=, , <=, >=	Character, integer, double	Boolean
Lógicos	Not, and, or, and then, or else, implies	Boolean	Boolean
Cadenas	+ (concatenación)	String	String

Métodos de utilidad para el lenguaje Eiffel

c: tipo carácter i: tipo integer d: tipo double s: tipo string

Salida por pantalla

```
io.put_boolean(b)
io.put_character(c)
io.put_integer(i)
io.put_double(d)
io.put_string(s)
io.put_new_line
```

Lectura por teclado

io.read_character
c := io.last_character

io.read_integer
i := io.last_integer

io.read_double
d := io.last_double

io.read_line
s := io.last_string

Conversión entre tipos

i := c.code
c := i.to_character — caracter del codigo ASCII almacenado en i
c2 := c1.to_upper — conversión a mayúsculas
c2 := c1.to_lower — conversión a minúsculas
if c1.same_as(c2)
s := i.to_string — Traduce el valor almacenado en i a una cadena de texto
s := i.to_string_format(8) — en este caso la cadena tendría un mínimo de 8 caracteres
s := d.to_string — Traduce el valor almacenado en r a una cadena de texto
s := d.to_string_format(6) — en tal caso se describen solo 6 decimales
i := d.floor — Devuelve el mayor entero menor o igual que el valor de d
i := d.ceiling — Devuelve el menor entero mayor o igual que el valor de d
i := d.rounded — Redondea al valor entero más cercano

Funciones matemáticas

i2 := i1.abs — Devuelve el valor absoluto
d2 := d1.abs — Devuelve el valor absoluto
i3 := i1.min(i2) — Devuelve el mínimo de valores almacenados en i1 e 12
i3 := i1.max(i2) — Devuelve el máximo de valores almacenador en i1 e i2
d3 := d1.pow(d2) — Devuelve la potencia del valor de d1 elevado al valor de d2

d2 := d1.sqrt — Devuelve la raíz cuadrada del valor almacenado en d1
d2 := d1.exp — Devuelve la exponencial del valor almacenado en d1
d2 := d1.log — Devuelve el logaritmo neperiano del valor almacenado en d1
d2 := d1.log10 — Devuelve el logaritmo decimal del valor almacenado en d1

Archivos de texto

Para creación, apertura o cierre:

!!fich.make — crea un archivo no asignado a ningún archivo físico
fich.connect_to(cad) — asigna el archivo a un archivo fisico descrito por la cadena cad
fich.disconnect — cierra el archivo
if fich.is_connected **then** — verifica si el archive ha sido conectado a un archivo físico que ya exista
if fich_ent.end_of_input **then** — verifica si se ha alcanzado el fin del archivo

Para lectura:

```
fich_ent.read_character;
c := fich_ent.last_character

fich_ent.read_integer;
n := fich_ent.last_integer

fich_ent.read_double;
d := fich_ent.last_double
```

Para escritura:

```
fich_sal.put_character(c)
fich_sal.put_integer(i)
fich_sal.put_double(d)
fich_sal.put_string(cad)
```

Bibliografía: Universidad de Valladolid. 2019. *Eiffel Estructurado*. [en línea] Disponible en: https://www.infor.uva.es/~cvaca/asigs/eiffel.pdf [Consultado el 3 de marzo de 2021].