UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Estructura de Datos y Algoritmos

MI. Marco Antonio Martínez Quintana

Asignatura:
Estructura de Datos y Algoritmos 1

Actividad Viernes
#1 Acordeón: Lenguaje C y C#

Alumna
Citlali Cuahtepitzi Cuatlapantzi

Fecha
(28/febrero/2021)

Acordeón: Lenguaje C y C#

El lenguaje C, es un lenguaje de programación, este se define como la combinación de símbolos y reglas que permiten la elaboración de programas con los cuales la computadora puede resolver tareas o resolver problemas de manera eficiente.

Lenguaje C

ESTRUCTURA DE UN PROGRAMA	
/*Programa que…*/	Comentarios
#include <stdio.h> **#include <conio.h>** #include <math.h> #include <string.h> #include <ctype.h>	Declaración de archivos de cabecera (librerías)
	Prototipo de las funciones
	Declaración de variables globales y constantes
main()	Programa principal
{	
	Declaración de contantes (locales)
	Declaración de variables (locales)
	Cuerpo del programa (estructuras de control)
}	
	Declaración de funciones
{	
***	
}	

TIPOS DE DATOS			
Tipo	**Significado**	**Ancho en bit**	**Rango de valores (IBM PC)**
char	carácter	8	-128 a127
int	entero	16	-32,768 a 32,767
short	entero corto	16	-32,768 a 32,767
long	entero largo	32	-2,147,483,648 a 2,147,483,647
unsigned char	carácter sin signo	8	0 a 255
unsigned	entero sin signo	16	0 a 65,535
unsigned short	entero corto sin signo	16	0 a 65,535
unsigned long	entero largo sin signo	32	0 a 4,294,967,295
enum	enumerado	16	0 a 65,535
float	real	32	3.4E-38 a 3.4E+38
double	real doble	64	1.7E-308 a 1.7E+308
Long double	real doble largo	64	1.7E-4932 a 1.7E+4932

OPERADORES									
Aritméticos:			**Lógicos:**		**Relacionales:**		**De asignación compuesta:**		
								Ejemplo	**Equivalencia**
+	Suma	a+b	||	o	**<**	Menor que	**+ =**	s+ = 10	s=s+10
-	Resta	a-b	&&	y	**>**	Mayor que	**- =**	r- = 3	r=r-3
*****	Producto	a*b	!	no	**<=**	Menor o igual que	*** =**	m* =4	m=m*4
/	División	a/b			**>=**	Mayor o igual que	**/ =**	d/ = 2	d=d/2
%	Módulo (residuo de la división entera)	a% b (Residuo de a entre b)			**!=**	Distinto a	**% =**	x% = 33	x=x%33
pow	Potencia	Pow(a,b) (a elevado a la b)			**==**	Igual que			

INCREMENTO Y DECREMENTO		
++	**++i**	Se incrementa i en 1 y a continuación se utiliza el nuevo valor de i en la expresión en la cual esté i
++	**i++**	Utiliza el valor actual de i en la expresión en la cual esté i y después se incrementa en 1
– –	– –**i**	Se decrementa i en 1 y a continuación se utiliza el nuevo valor de i en la expresión en la cual esté i
– –	**i**– –	Utiliza el valor actual de i en la expresión en la cual esté i y después se decrementa en 1

PRIORIDAD DE LOS OPERADORES ARIMÉTICOS	
Prioridad	**Operación**
1ª.	( ) ,[ ]
2ª.	!
3ª.	*, /, %
4ª.	+, -
5ª.	<, >, <=, >=
6ª.	= =, !=
7ª.	&&, ¦¦
8ª.	=, +=, -=, *=, /=, %=

SECUENCIAS DE ESCAPE PARA PRINTF MÁS COMUNES	
Secuencia	**Significado**
\n	Nueva línea
\t	Tabulador
\b	Espacio atrás
\r	Retorno de carro
\f	Comienzo de página
\'	Comilla simple
\' '	Comilla doble
****	Barra invertida
\xdd	Código ASCII en notación hexadecimal (cada d representa un dígito)
\ddd	Código ASCII en notación octal (cada d representa un dígito)

PALABRAS RESERVADAS	
abs()	Calcula el valor absoluto
char	Tipo de dato carácter
case	Si se cumple un caso
default	Ninguna opción de la selectiva múltiple
typedef	Crea un nuevo nombre para un tipo de dato ya definido
for	Estructura repetitiva o de ciclo
int	Tipo de dato entero
}	Fin del programa o de un bloque
do	Estructura repetitiva
printf	Imprime en pantalla
puts	Imprime una cadena
{	Inicio del programa o de un bloque
scanf	Lee una variable
gets	Lee una cadena de caracteres
clrscr	Borra el contenido de la pantalla
while	Estructura repetitiva
void	Valor nulo
main	Programa principal
sqrt	Calcular raíz cuadrada
float	Tipo de dato real
struct	Registro o estructura
return	Regresa valor a otra función
break	Terminar el caso
switch	Estructura selectiva múltiple
if	Estructura selectiva
else	La parte falsa de la selectiva

FORMATOS PARA PRINTF Y SCANF		
Formato	**printf()**	**scanf()**
Carácter simple	**%c**	**%c**
Cadena	**%s**	**%s**
Entero decimal con signo	**%d**	**%d**
Entero decimal con signo	**%i**	
Entero decimal, hexadecimal u octal		**%i**
Punto flotante (notación decimal)	**%f**	**%f o %e**
Punto flotante (notación expenencial)	**%e**	**%f o %e**
Punto flotante (%f o %e, el más corto)	**%g**	
Entero decimal sin signo	**%u**	**%u**
Entero hexadecimal sin signo	**%x**	**%x**
Entero octal sin signo	**%o**	**%o**

EJEMPLOS	
printf("El promedio es %f", N);	Imprime " El promedio es N"
scanf("%d",&EDAD);	Lee EDAD
if (A==2 && B<=8)	Si A=2 Y B es menor o igual a 8
if (A==2 ¦¦ B<=8)	Si A=2 o B es menor o igual a 8
if (!a)	Si No

INSTRUCCIONES DE SELECCIÓN

Sentencia "if"

```
if (condición) {
* * *
}
```

Sentencia "if- else"

```
if (condicion) {
* * *
}
else {
* * *
}
```

Sentencia "switch – case "

```
switch (Expresión entera o carácter)
{
case 'a':
Instrucciones para a;
break;
case 'b':
Instrucciones para b;
break; …
default: /* opcional de no ser ninguna de las anteriores*/
Instrucciones para default;
}
```

INSTRUCCIONES DE REPETICIÓN

Bucle "for"

Inicializa La variable | Comprueba la variable | Incrementa la variable

```
for ( contador=0;contador<10;contador++){
    printf (" contador= %d\n", contador);
    * * *
    }
```

Bucle "while"

```
while (condición) {
* * *
}
```

Bucle "do"

```
do {
* * *
} while (condición);
```

ARREGLOS UNIDIMENSIONALES

Tipo de dato | Nombre del arreglo | Número de variables que contendrá el arreglo

```
Int nombre[7]
```

Otra forma es definir los valores iniciales que tendrá el mismo

```
Int nombre[]={ 28, 2, 30, 33, 90, 21,34}
```

(Ubicación gráficamente)

28	2	30	33	90	21	34
0	1	2	3	4	5	6

Para imprimir la tercera expresión (30)

```
printf ("%d", nombre[2]);
```

FUNCIONES

- **Tipo_de_retorno:** es el tipo del valor devuelto por la función, o, en caso de que la función no devuelva valor alguno, la palabra reservada void.
- **Nombre_de_la_función:** es el nombre o identificador asignado a la función.
- **Lista_de_parámetros:** es la lista de declaración de los parámetros que son pasados a la función. Éstos se separan por comas. Debemos tener en cuenta que pueden existir funciones que no utilicen parámetros.
- **Cuerpo_de_la_función:** está compuesto por un conjunto de sentencias que llevan a cabo la tarea específica para la cual ha sido creada la función.
- **Return expresión:** mediante la palabra reservada return, se devuelve el valor de la función, en este caso representado por expresión.

Los **prototipos** de las funciones que se utilizan en un programa se incluyen generalmente en la cabecera del programa y presentan la siguiente sintaxis:
tipo_de_retorno nombre_de_la_función(lista_de_parámetros)

Definición

```
tipo_de_retorno  nombre_de_la_función(lista_de_parámetros)
{
sentencias;
}
```

Por ejemplo:

```
int cubo(int base)
{
int potencia;
potencia = base * base * base;
return potencia;
}
```

ARCHIVOS

```
#include<stdio.h>
int main(void){
FILE *Apun_a_Archivo; /*Declara apuntador a Archivo */
Apun_a_Archivo=fopen("Archivo.txt","w"); /*Se crea un Archivo.txt y se le asigna su dirección   Apun_a_Archivo */
putc('B',Apun_a_Archivo); /*Se escribe una letra al archivo abierto*/
fclose(Apun_a_Archivo); /*Se cierra el archivo creado */
}
```

Funciones		“MODOS” de abrir un archivo	
Nombre	**Función**	**Modo**	**Significado**
fopen()	Abre un archivo.	**“r”**	Abre un archivo existente sólo para lectura.
fclose()	Cierra un archivo.	**" w "**	Abre un nuevo archivo sólo para escritura. Si existe un archivo con el NombreDelArchivo especificado, será destruido y creado uno nuevo en su lugar.
fgets()	Lee una cadena de un archivo.		
fputs()	Escribe una cadena en un archivo	**"a "**	Abre un archivo existente para añadir, agrega información al final del archivo. Se creará un archivo nuevo si no existe un archivo con el NombreDelArchivo especificado.
fseek()	Busca un byte específico de un archivo.		
fprintf()	Escribe una salida con formato en el archivo.		
fscanf()	Lee una entrada con formato desde el archivo	**"r+"**	Abrir un archivo existente tanto para lectura como para escritura.
feof()	Devuelve cierto si se llega al final del archivo.	**"w+"**	Abre un archivo nuevo para lectura y escritura. Si existe un archivo con NombreDelArchivo especificado, será destruido y creado uno nuevo en su lugar.
ferror()	Devuelve cierto si se produce un error.		
rewind()	Coloca el localizador de posición del archivo al principio del mismo.	**"a+"**	Abre un archivo existente para leer y añadir. Se creará un nuevo archivo si no existe un archivo con el NombreDelArchivo especificado.
remove()	Borra un archivo.		
fflush()	Vacía un archivo.		

C#

Es un lenguaje de programación multiparagdima desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

CONCEPTOS	
Título	**Descripción**
Ensamblados en .NET	Describe cómo crear y usar ensamblados.
Programación asincrónica con async y await (C #)	Describe cómo escribir soluciones asincrónicas mediante las palabras clave async y await en C #. Incluye un tutorial.
Atributos (C #)	Analiza cómo proporcionar información adicional sobre elementos de programación como tipos, campos, métodos y propiedades mediante el uso de atributos.
Colecciones (C #)	Describe algunos de los tipos de colecciones que proporciona .NET. Demuestra cómo utilizar colecciones simples y colecciones de pares clave / valor.
Covarianza y contravarianza (C #)	Muestra cómo habilitar la conversión implícita de parámetros de tipo genérico en interfaces y delegados.
Árboles de expresión (C #)	Explica cómo puede utilizar árboles de expresión para habilitar la modificación dinámica del código ejecutable.
Iteradores (C #)	Describe iteradores, que se utilizan para recorrer las colecciones y devolver elementos de uno en uno.
Consulta de lenguaje integrado (LINQ) (C #)	Analiza las poderosas capacidades de consulta en la sintaxis del lenguaje de C # y el modelo para consultar bases de datos relacionales, documentos XML, conjuntos de datos y colecciones en memoria.
Reflexión (C #)	Explica cómo usar la reflexión para crear dinámicamente una instancia de un tipo, vincular el tipo a un objeto existente u obtener el tipo de un objeto existente e invocar sus métodos o acceder a sus campos y propiedades.
Serialización (C #)	Describe conceptos clave en serialización binaria, XML y SOAP.

VARIABLES

- Los nombres de las variables deben empezar con letra.
- Es posible colocar números en los nombres de las variables, pero no empezar el nombre con ellos.
- Los nombres de las variables no pueden llevar signos a excepción del guión bajo _ .
- Las variables no pueden llevar acentos en sus nombres.

MANERAS DE USAR UNA VARIABLE	
TIPO	**INFORMACIÓN QUE GUARDA**
bool	Es una variable booleana, es decir que solamente puede guardar los valores verdadero o falso
byte	Puede guardar un byte de información. Esto equivale a un valor entero positivo entre 0 y 255.
sbyte	Guarda un byte con signo de información. Podemos guardar un valor entero con signo desde –128 hasta 127.
char	Puede guardar un carácter de tipo Unicode.
decimal	Este tipo puede guardar un valor numérico con decimales. Su rango es desde ±1.0 ? 10?28 hasta ±7.9 ? 1028.
double	También nos permite guardar valores numéricos que tengan decimales. El rango aproximado es desde ±5.0 ? 10?324 hasta ±1.7 ? 10 308.
float	Otro tipo muy utilizado para guardar valores numéricos con decimales. Para este tipo el rango es desde ±1.5 ? 10?45 hasta ±3.4 ? 1038.
int	Cuando queremos guardar valores numéricos enteros con signo, en el rango de -2,147,483,648 hasta 2,147,483,647.
uint	Para valores numéricos enteros positivos, su rango de valores es desde 0 hasta 4,294,967,295.
long	Guarda valores numéricos enteros realmente grandes con un rango desde –9,223,372,036,854,775,808 hasta 9,223,372,036,854,775,807.
uling	Guarda valores numéricos enteros positivos. Su rango de valores varía desde 0 hasta 18,446,744,073,709,551,615.
short	Guarda valores numéricos enteros, pero su rango es menor y varía desde -32,768 hasta 32,767.
ushort	Puede guardar valores numéricos enteros positivos con un rango desde 0 hasta 65,535.
string	Este tipo nos permite guardar cadenas.

ESTRUCTURA BÁSICA DE UN PROGRAMA EN C#

```
// A skeleton of a C# program
using System;
namespace YourNamespace
{
   class YourClass
   {
   }

   struct YourStruct
   {
   }

   interface IYourInterface
   {
   }

   delegate int YourDelegate();

   enum YourEnum
   {
   }

   namespace YourNestedNamespace
   {
      struct YourStruct
      {
      }
   }

   class YourMainClass
   {
      static void Main(string[] args)
      {
         //Your program starts here...
      }
   }
}
```

Operadores relacionales		Expresiones lógicas		Operadores aritméticos		
<	Menor que	**OPERADOR**	**SIGNIFICADO**	**OPERADOR DESCRIPCIÓN**	**OPERADOR DESCRIPCIÓN**	**Precedencia**
>	Mayor que	&&	y	=	Asignación. Este operador ya es conocido por nosotros.	
<=	Menor que igual	||	o	+	Suma. Nos permite sumar los valores de las variables o los números	4
>=	Mayor que igual	!	no	-	Resta. Para restar los valores de las variables o los números.	5
!=	Distinto a			*	Multiplicación. Multiplica los valores de las variables o los números.	1
==	Igual que			/	División. Divide los valores de las variables o los números.	2
				%	% Módulo. Nos da el residuo de la división.	3

COMENTARIOS Y MENSAJES DEL COMPILADOR	
//	Comentario hasta el final de la línea
/* */	Comentario entre barras
Write()	Permite mostrar un mensaje en la consola
WriteLine()	Permite mostrar un mensaje en la consola e inserta un salto de línea

Referencias

Arrioja, N.,(2010).C# Guía total del programador. Recuperado de: https://www.bibliadelprogramador.com/2017/07/c-guia-total-del-programador.html

Corona, M. A. y Ancona, M. A., (s.f.), Diseño de algoritmos y su codificación en lenguaje C, Guadalajara, México: McGRAW-HILL.

Microsoft.(2017).Guía de programación de C#. Recuperado de: https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/