INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
DEPARTAMENTO DE CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN
ACADEMIA DE CIENCIAS DE LA COMPUTACIÓN

CARRERA:
Ingeniería en Sistemas Computacionales

PROFESOR:
José Carlos Dávalos López.

REACTIVOS DE EXAMEN PARA LA UNIDAD DE APRENDIZAJE:
Algoritmia y Programación Estructurada

NIVEL:
Primero

OBJETIVO GENERAL DE LA UNIDAD DE APRENDIZAJE:
Resolver problemas computacionales, de dificultad similar al ordenamiento, utilizando algoritmos, pseudo-código y buenas prácticas de programación para adquirir la capacidad de construir y desarrollar programas en un lenguaje de alto nivel.

CONTENIDOS:

Unidad Temática I. Conceptos básicos y herramientas de programación.
Unidad Temática II. Modularidad
Unidad Temática III. Control de flujo
Unidad Temática IV. Arreglos y tipos estructurados.
Unidad Temática V. Archivos e integración de conceptos.

Bibliografía

LÓPEZ ROMÁN, LEOBARDO
Programación estructurada y orientada a objetos: un enfoque algorítmico
3ra. Edición
Alfaomega

DEITEL / DEITEL
Cómo programar en C/C++
2da. Edición
Prentice Hall

CEBALLOS, FRANCISCO JAVIER
Enciclopedia del Lenguaje C
Primera Edición
Alfaomega

DAVIES, PAUL
The Indispensable Guide to C
Primera Edición
Addison-Wesley

ÍNDICE

PÁGINA

UNIDAD TEMÁTICA I
NOMBRE:
Conceptos básicos y herramientas de programación
OBJETIVO PARTICULAR DE LA UNIDAD
Reproducir en una computadora algunos ejemplos sencillos escritos en pseudo-código para operar las herramientas de desarrollo de programas utilizando variables, diferentes tipos de datos, expresiones, documentación y siguiendo un estilo de escritura del código.

1.1 Reactivos de opción múltiple

1.1.1. Es un elemento básico de una computadora:
a) Unidad central de procesamiento
b) Memoria
c) Unidad de entrada
d) Todas las anteriores

1.1.2. Es un ejemplo de algoritmo:
a) Manual de usuario
b) Receta de cocina
c) Instructivo para armar un mueble
d) Todas las anteriores

1.1.3. Un algoritmo se puede representar de esta forma:
a) Lenguaje Natural
b) Pseudo-código
c) Diagramas de flujo
d) Todas las anteriores

1.1.4. Una expresión unida por operadores relacionales evalúa a:
a) El valor entero o real de la expresión aritmética.
b) Valores de falso o verdadero.
c) Números reales.
d) Datos de tipo carácter.

1.1.5. Es una característica que debe presentar un buen programa:
a) Operatividad
b) Legibilidad
c) Claridad
d) Todas las anteriores

1.1.6. Es un nombre que define a una variable, una función o un tipo de datos:
a) Operador
b) Identificador
c) Evaluador
d) Compilador

1.1.7. Una sentencia en lenguaje C debe terminar en
a) punto y coma
b) cerrar llaves
c) cerrar comillas
d) punto y aparte

1.1.8. Las normas mínimas que debe cumplir un compilador C están dadas por:
a) IEEE
b) NOM
c) ANSI
d) ANCE

1.1.9. Es un ejemplo de operador lógico:
a) ?:
b) ++
c) ||
d) %

1.1.10. Es un nombre inválido para una variable:
a) x3
b) zzz
c) r2
d) 4y

1.2 Reactivos de falso / verdadero

1.2.1. ( ) Un programa transportable es el que puede ejecutarse en un entorno diferente (como un sistema operativo distinto) sin hacerle modificaciones importantes.
1.2.2. ( ) C++ es un lenguaje apropiado para la programación estructurada .
1.2.3. ( ) Un algoritmo forzosamente debe tener un inicio, pero no es necesario que tenga fin.
1.2.4. ( ) En C se pueden emplear operadores de asignación compuestos.
1.2.5. ( ) Los operadores relacionales tienen mayor precedencia que los operadores aritméticos.
1.2.6. ( ) Los operadores de incremento y decremento son operadores unarios.
1.2.7. ( ) ASCII significa Código Estándar Americano para el Intercambio de Información.
1.2.8. ( ) El nombre de una variable puede comenzar con un dígito.
1.2.9. ( ) Una variable que se declaró, pero no inicializó aún, tiene un valor de cero.
1.2.10. ( ) Los operadores unarios tienen una precedencia baja.
1.2.11. ( ) Un decremento es una operación binaria.
1.2.12. ( ) Una asignación se lee de derecha a izquierda.

1.3 Reactivos de relación de columnas

1.3.1. ( ) Este tipo de dato ocupa un byte de memoria	a) *
1.3.2. ( ) La división de dos enteros, arroja como resultado un	b) ?:
1.3.3. ( ) Operador de asignación	c) int
1.3.4. ( ) El resultado de esta operación es el residuo de una división entera	d) =
1.3.5. ( ) Así se representa la multiplicación	e) ==
1.3.6. ( ) Es un operador relacional	f) char
1.3.7. ( ) Es el operador lógico con mayor precedencia	g) &&
1.3.8. ( ) Se conoce como operador ternario	h) define
1.3.9. ( ) Valor con decimales de precisión sencilla	i) %
1.3.10. ( ) Se puede usar para declarar constantes	j) float

1.4 Reactivos para completar la sentencia

1.4.1. La arquitectura _____ referencia una organización de la computadora que utiliza el mismo dispositivo de almacenamiento tanto para las instrucciones como para los datos.
1.4.2. La arquitectura _____ referencía una organización de la computadora que utiliza memorias físicamente separadas para las instrucciones y para los datos.
1.4.3. Una ventaja importante de la representación de un algoritmo mediante _____ es que es más sencilla la tarea de pasar a un lenguaje de programación formal.
1.4.4. Para representar el inicio y el fin en un diagrama de flujo se utiliza un _____.
1.4.5. Para representar una decisión en un diagrama de flujo se usa un _____.
1.4.6. Para representar un paso, actividad o instrucción en un diagrama de flujo se usa un _____.
1.4.7. Para declarar una constante en lenguaje C, se puede usar la directiva define o la palabra reservada _____.
1.4.8. El formato de impresión _____ se usa para imprimir un entero sin signo.
1.4.9. El formato de entrada _____ se emplea para recibir una cadena.
1.4.10. _____es la secuencia de escape del carácter diagonal invertida.
1.4.11. _____ es la secuencia de escape que mueve el cursor a la siguiente posición de tabulador horizontal.
1.4.12. _____ es la ejecución de una instrucción (o conjunto) mientras una variable sea 'verdadera'. Esta estructura lógica también se conoce como ciclo o bucle.

1.5 Reactivos de preguntas abiertas (conceptos)

1.5.1. ¿Qué es una computadora?
1.5.2. ¿Qué es la arquitectura de una computadora?
1.5.3. ¿Qué función tiene la memoria RAM (Random Access Memory)?
1.5.4. ¿Qué es un algoritmo?
1.5.5. Mencione las formas para representar un algoritmo
1.5.6. ¿De dónde proviene la palabra algoritmo?
1.5.7. ¿Qué es un lenguaje de programación?
1.5.8. ¿Qué es un programa?
1.5.9. ¿Qué es un “Diagrama de Flujo”?
1.5.10. ¿Qué es un “Pseudocódigo”?
1.5.11. ¿Qué es un compilador?
1.5.12. ¿Cuál es el valor de la siguiente expresión: 6*6-2+16/8
1.5.13 Si se declararon int a=2,b=8; ¿Cuál es el valor de la siguiente expresión:
(5*9%++a)+(5*(b++/(3%2)))
1.5.14. ¿Cuáles son los símbolos que se utilizan para agregar comentarios en lenguaje C?

1.6 Reactivos para escribir la salida de código

1.6.1. Indique cuál es la salida del siguiente código:

```
#include <stdio.h>
#include <conio.h>

int a = 5, b = 6, c, d;
int x;

main( )
{
c = (15%10);
d = (58-57);
x = a < b || a < c && c < d;
printf("\nLa expresión sin paréntesis evalúa a %d", x);
x = (a < b || a < c) && c < d;
printf("\nLa expresión con paréntesis evalúa a %d", x);
getch( );
}
```

1.6.2. Indique cuál es la salida del siguiente código si se ingresan los valores 25, 14:

```
#include <stdio.h>
#include <math.h>
#include <conio.h>

 main ()
{
        float a=0, b=0, a1=0, a2=0, h=0;
        printf ("\nIngrese el valor de c1: ");
        scanf ("%f", &a);
        printf ("Ingrese el valor de c2: ");
        scanf ("%f", &b);
        h=sqrt(a*a+b*b);
        a1=180.0*acos(a/h)/M_PI;
        a2=180.0*asin(b/h)/M_PI;
        printf ("Valor de a1: %g\n", a1);
        printf ("Valor de a2: %g\n", a2);
        printf ("Valor de h: %g\n", h);
        printf ("\n");
getch ();
return 0;
}
```

1.6.3. Indique cuál es la salida del siguiente código si se ingresa el valor 23:

```
#include <stdio.h>
#include <conio.h>


main ()
{
    int n;
```

```
    printf ("Ingresa un numero entero:\n");
    scanf ("%d",&n);
    printf ("el %d es numero %s", n,((n%2)?"impar":"par"));
    getch ();
}
```

1.7 Reactivos de problemas para elaborar algoritmos y para programar

1.7.1. Elaborar un algoritmo para cambiar la llanta ponchada de un automóvil.

1.7.2. Elaborar un algoritmo que permita cambiar el vidrio roto de una ventana.

1.7.3. Elaborar el algoritmo para obtener la solución de una ecuación de segundo grado de una variable.

1.7.4. Hacer un algoritmo (mostrar en lenguaje natural y diagrama de flujo) para copiar un CD de música a una memoria USB.

1.7.5. Crear un algoritmo (representar diagrama de flujo y pseudocódigo) que lea un valor de temperatura en grados Celsius (centígrados) y muestre como resultado su equivalente en grados Fahrenheit. Considerar la siguiente fórmula:

ºF = °C 9/5 + 32

1.7.6. Realizar un algoritmo (mostrar con diagrama de flujo y pseudocódigo) que calcule el volumen de un cilindro de radio $(X + 4) * X^2$ y de altura $(X + 6)^{-x}$
Observaciones:

- El rango de valores de X los dará el usuario al comenzar el programa y deberá ser un rango positivo. Por ejemplo $(10 < X < 200)$
- Recuerde que el volumen de un cilindro se calcula multiplicando el área de la base por la altura
- El área de un círculo es: $\pi * radio^2$

1.7.7. Escribir un programa que muestre los datos del alumno en la pantalla.

1.7.8. Hacer un programa que reciba dos números enteros, calcule su suma, resta, multiplicación, división y división modular e imprima los resultados.

1.7.9. Elaborar un programa que calcule e imprima el costo de un terreno cuadrado o rectangular, ingresando como datos la anchura y la longitud en metros, y el costo del metro cuadrado.

1.7.10. Hacer un programa que lea una cantidad en horas, e imprima su equivalente en centésimas de segundo, segundos, minutos, días y semanas.

1.7.11. Escribir un programa que calcule el área de un triángulo rectángulo, dada la altura y la base.

1.7.12. Escribir un programa que muestre el equivalente en ASCII de un carácter recibido desde el teclado.

UNIDAD TEMÁTICA II
NOMBRE:
Modularidad
OBJETIVO PARTICULAR DE LA UNIDAD
Conocer el concepto de modularidad para que las implementaciones que realice de los algoritmos sean reutilizables aplicando los conceptos de función, procedimiento o módulo.

2.1 Reactivos de opción múltiple

2.1.1. Son los datos que recibe una función del exterior:
a) argumentos
b) resultados
c) tipos
d) mensajes

2.1.2. Es la función que se usa para calcular una potencia:
a) sqrt (x)
b) exp (x)
c) log (x)
d) pow (x, y)

2.1.3. La función gets está contenida en la biblioteca:
a) conio.h
b) string.h
c) stdio.h
d) math.h

2.1.4. La función log está contenida en la biblioteca:
a) stdlib.h
b) stdio.h
c) string.h
d) math.h

2.1.5. La Programación Estructurada se refiere a:
a) A programar en un lenguaje adecuado a las necesidades de estructura.
b) A un paradigma de programación no orientado a objetos basado en procedimientos.
c) A un paradigma de programación que implementa estructuras de control secuenciales, iterativas y selectivas.
d) A un paradigma de programación que requiere el uso de lenguajes de medio nivel.

2.1.6. Para que una variable local mantenga su valor entre llamadas, se utiliza el modificador:
a) unsigned
b) register
c) short
d) static

2.1.7. Esta función es necesaria en cualquier programa en C:
a) include
b) main
c) return
d) exit

2.1.8. El cuerpo de una función debe estar encerrado entre:
a) { }
b) ( )
c) [ ]
d) /* */

2.2 Reactivos de falso / verdadero

2.2.1. ( ) La función sqrt() sirve para generar números aleatorios
2.2.2. ( ) La biblioteca math.h está diseñada para operaciones matemáticas básicas
2.2.3. ( ) La función fabs sirve para ordenar dos caracteres.
2.2.4. ( ) La definición de una función tiene un encabezado y el cuerpo de la función.
2.2.5. ( ) Una función no puede llamar a otra función.
2.2.6. ( ) C permite la definición de una función dentro de otra función.
2.2.7. ( ) Los nombres de variable de los argumentos en un prototipo de función son opcionales.
2.2.8. ( ) Se recomienda que una función sea lo más extensa posible.
2.2.9. ( ) Cada que se llama a una función, se le mandan los mismos valores.
2.2.10. ( ) Para llamar a una función, se utiliza la palabra reservada call.

2.3 Reactivos de relación de columnas

2.3.1. ( ) Recibe y regresa un entero
2.3.2. ( ) Se declara fuera de una función
2.3.3. ( ) Es válida sólo dentro de la función donde se declaró
2.3.4. ( ) Biblioteca para manejo de cadenas
2.3.5. ( ) La función sin está en esta biblioteca
2.3.6. ( ) Esta biblioteca no tiene funciones
2.3.7. ( ) Esta función no muestra la tecla que se presionó
2.3.8. ( ) Muestra la tecla que se presionó en pantalla
2.3.9. ( ) Es un fragmento de un programa que se desarrolla de forma independiente del resto
2.3.10. ( ) Recibe dos enteros y no regresa nada

a) assert.h
b) getche();
c) módulo
d) int cubo (int n);
e) variable global
f) getch();
g) void tcp (int n, int x);
h) math.h
i) variable local
j) string.h

2.4 Reactivos para completar la sentencia

2.4.1. Las variables _____ son automáticas por default.
2.4.2. Las variables _____ se pueden usar en cualquier parte del programa.
2.4.3. La función printf está contenida en la biblioteca_____.
2.4.4. La función tan está contenida en la biblioteca_____.
2.4.5. Para que sean visibles las bibliotecas al programa, se añade el comando del preprocesador_____.

2.4.6. Las “funciones” que no retornan nada, y que se llaman únicamente para que ejecuten su código, se les llama _____.
2.4.7. _____ es la directiva que se usa para declarar una macro.
2.4.8. Para regresar un valor de una función se usa la palabra clave _____.
2.4.9. La biblioteca de entrada y salida estándar de C es _____.
2.4.10. _____ es el prototipo de una función llamada func, que regresa un carácter y recibe un flotante llamado z.

2.5 Reactivos de preguntas abiertas (conceptos)

2.5.1. En un programa, ¿en qué consiste el método divide y vencerás?
2.5.2. ¿Qué es una función en lenguaje C?
2.5.3. ¿Qué es un prototipo de función?
2.5.4. ¿Cómo está formado el prototipo de una función?
2.5.5. ¿Qué valor regresa la siguiente función y cuáles recibe: void función (int r, int c)?
2.5.6.¿Cuál es la diferencia entre una variable global y una local?
2.5.7. ¿Cuál es la diferencia entre una función y un procedimiento?
2.5.8. ¿Qué es una biblioteca?
2.5.9. ¿Cuál es el uso de las funciones?.
2.5.10. ¿Qué es una sentencia compuesta?
2.5.11. ¿Qué es un dato sin valor (void)?
2.5.12.¿Para qué sirve return?

2.6 Reactivos para escribir la salida de código

2.6.1. Indique cuál es la salida del siguiente código:

```
#include <conio.h>
#include <stdio.h>
#include <math.h>

void imprimir(){
}
double suma(double valor1, double valor2){
   double valor3;
   valor3=valor1+valor2;
   return valor3;
}

double multiplicacion(double x, double y){
   double z;
   z=x*y;
   return z;
}

int main(void){
   double a=2.5,b=3.0,resultado;
   imprimir();
   resultado=suma(a,b);
   printf("suma: %e\n",resultado);
   resultado=multiplicacion(a,b);
   printf("multiplicacion: %e\n",resultado);
   imprimir();
   getch();
}
```

2.6.2. Indique cuál es la salida del siguiente código si se introducen los valores 53, 22:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int compara (int a, int b);
main ()
{
    int num1, num2;
    int resultado;
    printf("Introduzca dos numeros: \n");
    scanf("%i %i", &num1, &num2);
    resultado=compara(num1,num2);
    printf("\nEl mayor de los dos es: %i", resultado);
    printf("\n\n\t\t\t***Oprime Enter para terminar***");
getch();
}
int compara (int a, int b)
{
   return (a > b)?a:b;
}
```

2.6.3. Indique cuál es la salida del siguiente código:

```
#include <stdio.h>
#include <conio.h>

void funcion0 (void);

main( )
{

printf (" Primero: ");
funcion0 ( );
printf (" Segundo: ");
funcion0 ( );
printf (" Tercero: ");
funcion0 ( );
printf (" Cuarto: ");
funcion0 ( );
getch();
}
void funcion0 (void)
{
static int x = 12;
int y = 12;
printf ("x = %2d, y = %d\n", x--, y--);
}
```

2.7 Reactivos de problemas para programar

2.7.1. Escribir un programa que implemente y utilice una función para determinar si un número es positivo o negativo. El número entero se lee por teclado e imprime en pantalla si el número leído es positivo o negativo haciendo uso de la función definida.

2.7.2. Hacer un programa con una función llamada promedio, que tome tres números reales como parámetros, y no devuelva nada. Esta función debe calcular el promedio de los tres números pasados como parámetros y mostrar con un mensaje cuál es el resultado del cálculo.

2.7.3. Escribir un programa con dos funciones: una llamada pedir, que no toma parámetros, y devuelve un número entero; y otra llamada triple, que toma un número entero como parámetro y devuelve un número entero. La función pedir debe solicitar por teclado un número entero, y devolverlo. La función triple, debe calcular el triple del número que recibe como parámetro y devolver el resultado.

2.7.4. Elaborar un programa que reciba un valor de radio y calcule el perímetro, área y volumen para este valor de radio usando funciones para cada cálculo.

2.7.5. Las resistencias electrónicas suelen ir identificadas por un código de colores que permite marcar cada resistencia con su valor (en Ohms) y su Tolerancia (en %). Este código de colores viene representado en la siguiente tabla:

Dígito	Color	Multiplicador	Tolerancia
	Ninguno		20%
	Plata	0.01	10%
	Oro	0.1	5%
0	Negro	1	
1	Marrón	10	
2	Rojo	10^2	2%
3	Naranja	10^3	
4	Amarillo	10^4	
5	Verde	10^5	
6	Azul	10^6	
7	Violeta	10^7	
8	Gris		
9	Blanco		

El código que suele emplearse en las resistencias es un código de 4 colores, es decir, cada resistencia está marcada con 4 bandas y cada una de ellas puede ser de diferente color. Cada banda tiene un significado, que depende de cada color:
-Las primeras 2 bandas indican un número de 2 dígitos: Esos dos dígitos vienen dados por el color de esas bandas, según la columna "Dígito" de la tabla.
-La tercera banda es un valor por el que se multiplicará el número obtenido por las bandas anteriores. Una vez multiplicados ambos valores, obtenemos el valor de la resistencia en Ohms.
-La cuarta banda indica la tolerancia de la resistencia y, como puede verse en la tabla, no puede ser de cualquier color.
Ejemplo: Las resistencias con los siguientes colores, tienen los siguientes valores de resistencia y tolerancia:
Verde-Azul-Amarillo-Oro 560 kOhms, 5%
Rojo-Negro-Rojo-Rojo 2 kOhms, 2%
Rojo-Rojo-Marrón-Plata 220 Ohms, 10%

Según todo lo anterior, implemente un programa que pida los colores de las cuatro bandas que serán números naturales, y que indicarán el color de las bandas según la columna "Dígito". Los colores Oro, Plata y Ninguno tomarán los valores 10, 11 y 12 respectivamente.
Con una función se debe calcular la resistencia y la tolerancia dados los colores, la función tendrá, como mínimo, 4 argumentos. Esta función debe llamar a otra función que muestre en pantalla el color que le corresponde a cada dígito (incluyendo los dígitos 10, 11 y 12) y los resultados que se obtuvieron en la función anterior.

UNIDAD TEMÁTICA III
NOMBRE:
Control de flujo
OBJETIVO PARTICULAR DE LA UNIDAD
Reproducir algoritmos que resuelvan problemas iterativos en una computadora con la finalidad de incrementar la habilidad del estudiante para programar tomando en cuenta las técnicas de la programación estructurada y modular.

3.1 Reactivos de opción múltiple

3.1.1. Es un operador de asignación compuesto:
a) x= y+2;
b) x=*3;
c) z=25+1;
d) x*=5;

3.1.2. Como parte de este ciclo está especificado un incremento o variación de la variable de control:
a) while
b) for
c) repeat
d) do/while

3.1.3. El operador ternario es equivalente a una estructura:
a) if/else
b) if
c) case
d) do/while

3.1.4. Las estructuras de repetición son:
a) while, do/while, if
b) switch, if/else
c) while, do/while, for
d) repeat, continue

3.1.5. Este ciclo no está definido en lenguaje C:
a) while
b) for
c) repeat
d) do/while

3.2 Reactivos de falso / verdadero

3.2.1. ( ) La estructura switch es de selección simple.
3.2.2. ( ) La estructura switch puede tomar como argumento un flotante.
3.2.3. ( ) No es posible anidar estructuras for.
3.2.4. ( ) Break se puede aplicar solamente fuera de un bucle.
3.2.5. ( ) Continue se puede aplicar solamente dentro de un ciclo.
3.2.6. ( ) No es posible utilizar el ciclo while para resolver problemas cuya solución esté dada con for.
3.2.7. ( ) En C un falso se representa únicamente con el valor 0 (cero).
3.2.8. ( ) La opción default dentro de la estructura switch es obligatoria.

3.3 Reactivos de relación de columnas

3.3.1. ( ) Estructuras de selección	a) continue
3.3.2. ( ) Estructuras de repetición	b) if , if-else
3.3.3. ( ) Finaliza la ejecución del ciclo donde esta incluida	c) x++;
3.3.4. ( ) Obliga a ejecutar la siguiente iteración del ciclo.	d) switch
3.3.5. ( ) Es la más adecuada para programar un menú.	e) x = x+y;
3.3.6. ( ) Se puede usar para hacer un contador.	f) for , while
3.3.7. ( ) Se puede usar para hacer un acumulador.	g) break
3.3.8. ( ) Permite un rango de repeticiones que va de 1 a n.	h) do/while

3.4 Reactivos para completar la sentencia

3.4.1. Las _____ permiten modificar el flujo de ejecución de las instrucciones de un programa.
3.4.2. Los 3 tipos de estructuras de control son: secuencia, _____ e _____.
3.4.3. Las estructuras de selección son if, _____ y _____.
3.4.4. Un _____ o _____ es un grupo de instrucciones que la computadora ejecuta de forma repetida mientras se cumple cierta condición.
3.4.5. La estructura de repetición _____ se ejecuta al menos una vez porque la condición se evalúa al final.
3.4.6. La estructura de selección _____ ejecuta una acción cuando la condición es verdadera, de lo contrario, la acción no se ejecuta.
3.4.7. La estructura de selección _____ ejecuta una acción cuando la condición es verdadera, de lo contrario, se ejecuta una acción diferente.
3.4.8. Cuando se tienen varios enunciados for anidados, el que se mueve más lento es el _____.

3.5 Reactivos de preguntas abiertas (conceptos)

3.5.1. ¿Qué dice el “Teorema de la Programación Estructurada”?
3.5.2. ¿Qué es indentación, para qué se utiliza?
3.5.3. ¿Que es una estructura de repetición?
3.5.4. La selección se implementa en 3 instrucciones, ¿cuales son?
3.5.5. La repetición se implementa en 3 instrucciones, ¿cuales son?

3.5.6. ¿Cuál es la sintaxis de la estructura IF?
3.5.7. ¿Cuál es la sintaxis de la estructura IF/ELSE?
3.5.8. ¿Cuál es la sintaxis de la estructura FOR?
3.5.9. ¿Cuál es la sintaxis de la estructura WHILE?
3.5.10. ¿Cuál es la sintaxis de la estructura DO/WHILE?
3.5.11. ¿Qué es un contador?
3.5.12. ¿Qué es un acumulador?
3.5.13. ¿Que hace la estructura de decisión múltiple (switch)?
3.5.14. ¿Cuál es al sintaxis de la estructura SWITCH?
3.5.15. ¿Para qué sirve break?
3.5.16. ¿Para que sirve continue?
3.5.17. ¿Qué es un contador?
3.5.18. ¿Qué es un acumulador?
3.5.19. Encontrar el valor final de x, si su valor inicial es 0

```
if(x>=0)
        x++;
if(x>=1);
        x++;
if(x<=1)
        x--;
else
        x++;
```

3.5.20. Encontrar el valor final de x, si su valor inicial es 0

```
if(x>=0)
        x++;
if(x>0 && x<7)
        x+=2;
else if(x>0 && x<5)
        x+=3;
```

3.6 Reactivos para escribir la salida de código

3.6.1. Indique cuál es la salida del siguiente código:

```
#include <stdio.h>
#include <conio.h>
main ()
{
    int n;
    for (n=0; n<=100; n++)
    {
       if (n%5!=0)
       continue;
       printf ("%d ",n);
    }
    getch ();
}
```

3.6.2. Indique cuál es la salida del siguiente código si se introduce el valor 8:

```
#include <stdio.h>
#include <conio.h>

main()
{
   int calif;
   printf("Ingresa una calificacion: ");
   scanf ("%d",&calif);
   if(calif>=6)
            {
              printf("pasaste!! :D\n");
             }
  else{
       printf("reprobaste :(\n");
       }
   getch ();
}
```

3.6.3. Indique cuál es la salida del siguiente código si se ingresan los valores 5, 7, 9, 3, 4, 8:

```
#include <conio.h>
#include <stdio.h>
int main()
{
int num, n, m, i;
float s=0, p;
    printf ("Ingrese la cantidad: ");
    scanf ("%i", &n);
    for(i=0;i<n;i++)    {
        printf ("Numero %i: ", i+1);
        scanf ("%i", &num);
        s+=num;    }
        p= s/n;
        printf ("El resultado es: %f\n", p);
getch ();
}
```

3.6.4. Indique cuál es la salida del siguiente código si se ingresa el valor 13:

```
#include <stdio.h>
#include <conio.h>
main ()
{
    int mes = 0;
    printf("\nIntroduce un numero:");
    scanf("%i", &mes);
    switch (mes)
```

```
{
      case 1:
          printf("\n***Enero***");
          break;
      case 2:
          printf("\n***Febrero***");
          break;
      case 3:
          printf("\n***Marzo***");
          break;
      case 4:
          printf("\n***Abril***");
          break;
      case 5:
          printf("\n***Mayo***");
          break;
      case 6:
          printf("\n***Junio***");
          break;
      case 7:
          printf("\n***Julio***");
          break;
      case 8:
          printf("\n***Agosto***");
          break;
      case 9:
          printf("\n***Septiembre***");
          break;
      case 10:
          printf("\n***Octubre***");
          break;
      case 11:
          printf("\n***Noviembre***");
          break;
      case 12:
          printf("\n***Diciembre***");
          break;
      default:
            printf("\n\a Solo hay 12 meses!!! ");
      }
switch (mes)
{
      case 1: case 2: case 3:
          printf("\n\n***Es parte del Primer Trimestre***");
          break;
      case 4: case 5: case 6:
          printf("\n\n**Es parte del Segundo Trimestre***");
          break;
      case 7: case 8: case 9:
          printf("\n\n**Es parte del Tercer Trimestre***");
          break;
      case 10: case 11: case 12:
          printf("\n\n**Es parte del Cuarto Trimestre***");
```

```
            break;
        default:
              printf("\n\a Solo hay 4 Trimestres!!! ");
    }

        printf("\n\nOprime una tecla para finalizar*");
getch ();
}
```

3.6.5. Indique cuál es la salida del siguiente código:

```
#include<stdlib.h>
#include<stdio.h>

int main(void){
   int valor1=0,valor2=10;
   while(valor1<valor2){
   printf(" estoy dentro del cliclo %d\n",valor1);
      valor1++;
   }
   printf("ya estoy fuera del ciclo %d\n",valor1);
  system ("color 1F");
  system("pause");
}
```

3.6.6. Indique cuál es la salida del siguiente código si se ingresan los valores 2, 2000:

```
#include <stdio.h>
#include <conio.h>

main ()
{
    int d=0, m=0, a=0;
    printf ("introduzca numero del mes y el año:\n");
    scanf ("%d %d",&m, &a);
    switch (m)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
          d=31;
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            d=30;
            break;
```

```
        case 2:
            if ((a%4==0) && (a%100!=0) || (a%400==0))
            d=29;
            else
            d=28;
            break;
        default:
              printf ("\n El mes no es valido\n");
              break;
              }
        if (m>=1 && m<=12)
        printf ("\nEl mes %d del año %d tiene %d dias\n",m, a, d);
        getch ();
        }
```

3.6.7. Indique cuál es la salida del siguiente código si se ingresan los valores 614, 123:

```
#include <stdio.h>
#include <conio.h>

main ()
{
   int n1,n2;
   printf("Ingrese un numero: ");
   scanf("%d", &n1);
   printf("Ingrese otro numero: ");
   scanf("%d", &n2);
   if(n1>n2){
      printf("%d es mayor que %d.\n",n1,n2);
   }else if(n2>n1){
      printf("%d es mayor que %d.\n",n2,n1);
   }else{
      printf("Los dos numeros son iguales.\n");
  }
   getch ();
}
```

3.6.8. Indique cuál es la salida del siguiente código si se ingresa el valor 10:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

void numeros (int n);
main()
{
     int numero;
     printf("\nEscribe un numero entero\n");
     scanf("%d", &numero);
     numeros(numero);
     return 0;
}
```

```
void numeros(int numero)
{
    int cont1=1, cont2;

    while (cont1<=numero)
    {
        cont2=1;
        printf("%d.",numero);
            while (cont2<=cont1)
            {
                printf("%d",cont2);
                cont2++;
            }
        printf("\n");
        cont1++;
    }
    system("pause");
}
```

3.6.9. Indique cuál es la salida del siguiente código si se ingresa el valor 79:

```
#include <stdio.h>
#include <conio.h>

void cr (int);
main ()
{
    int (x);
    printf("Escriba un numero: \n");
    scanf("%d",&x);
cr (x);
}
 void cr (int x)
{
int mod1=0, mod2=0, x1=0, x2=0, x3=0, x4=0;
mod1=x%100;
x1=x-mod1;
x2=x1/100;
switch(x2)
{
            case 1 : printf ("C");
            break ;
            case 2 : printf ("CC");
            break ;
            case 3 : printf ("CCC");
            break ;
            case 4 : printf ("CD");
            break ;
            case 5 : printf ("D");
            break ;
            case 6 : printf ("DC");
            break ;
            case 7 : printf ("DCC");
```

```
            break ;
            case 8 : printf ("DCCC");
            break ;
            case 9 : printf ("CM");
            break ;
            case 10 : printf ("M");
            break ;

}
        mod2=mod1%10;
        x3=mod1-mod2;
        x4=x3/10;
        switch(x4)
{
    case 1 : printf ("X");
    break ;
    case 2 : printf ("XX");
    break ;
    case 3 : printf ("XXX");
    break ;
    case 4 : printf ("XL");
    break ;
    case 5 : printf ("L");
    break ;
    case 6 : printf ("LX");
    break ;
    case 7 : printf ("LXX");
    break ;
    case 8 : printf ("LXXX");
    break ;
    case 9 : printf ("XC");
    break ;
}
  switch(mod2)
{
  case 1 : printf ("I");
  break ;
  case 2 : printf ("II");
  break ;
  case 3 : printf ("III");
  break ;
  case 4 : printf ("IV");
  break ;
  case 5 : printf ("V");
  break ;
  case 6 : printf ("VI");
  break ;
  case 7 : printf ("VII");
  break ;
  case 8 : printf ("VIII");
  break ;
  case 9 : printf ("IX");
  break ;
}
getch ();
}
```

3.6.10. Indique cuál es la salida del siguiente código si se ingresan los valores 4, 13:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
main()
{    system("color 4f");
     int x,y,c,res=1,coc,i,i1,i2;
     printf("\nIntroduce primer numero:");
     scanf("%d",&x);
     printf("Introduce segundo numero:");
     scanf("%d",&y);
     i1=x;
     i2=y;
     if(x<y)
     {
          i=x;
          x=y;
          y=i;
     }
     while(res!=0)
     {
          coc=x/y;
          res=x%y;
          if(res<0)
          {
            if(coc<0)
            coc-=1;
            if(coc>=0)
            coc+=1;
            res=x-(y*coc);
 }
          printf("%d = %d*%d + %d \n ",x,y,coc,res);
          x=y;
          y=res;
     }
     if(x<0) x=-x;
     printf("\n El resultado es: %d",x);
     getch();
}
```

3.7 Reactivos de problemas para programar

3.7.1. Dibujar el diagrama de flujo, escribir el pseudocódigo y el código en C de un algoritmo que lea cinco números (cada uno de ellos entre 1 y 40). Para cada uno de los números se debe imprimir el número y una línea conteniendo dicho número en asteriscos adyacentes. Por ejemplo, si se lee el número 7, deberá imprimir 7 *******.

3.7.2. Escribir un programa que muestre una tabla de conversiones de temperatura de grados Fahrenheit a grados Celsius, de 0ºF a 300ºF de 20 en 20ºF.

3.7.3. Hacer un programa que reciba un número entre 0 y 30 e imprima una pirámide de asteriscos. Debe considerar que si ingresa un número menor a 0 o mayor a 30 debe marcar error. Por ejemplo, si se ingresa el número 5 debe imprimir en pantalla:

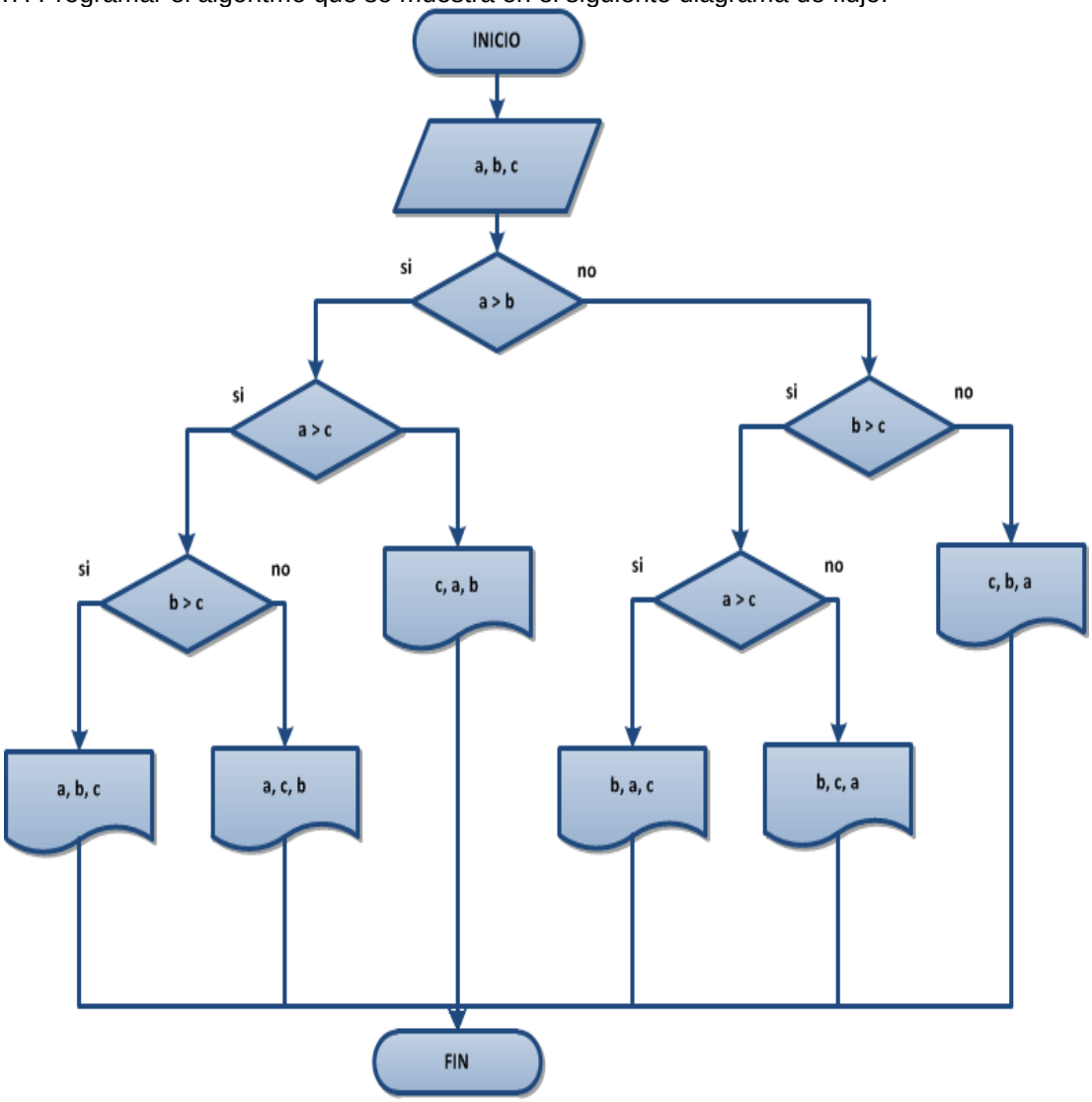
```
    **
   ****
  ******
 ********
**********
```

3.7.4. Escribir un programa que reciba 10 números enteros desde teclado, debe mostrar cuantas veces se repitió algún número de ser el caso.

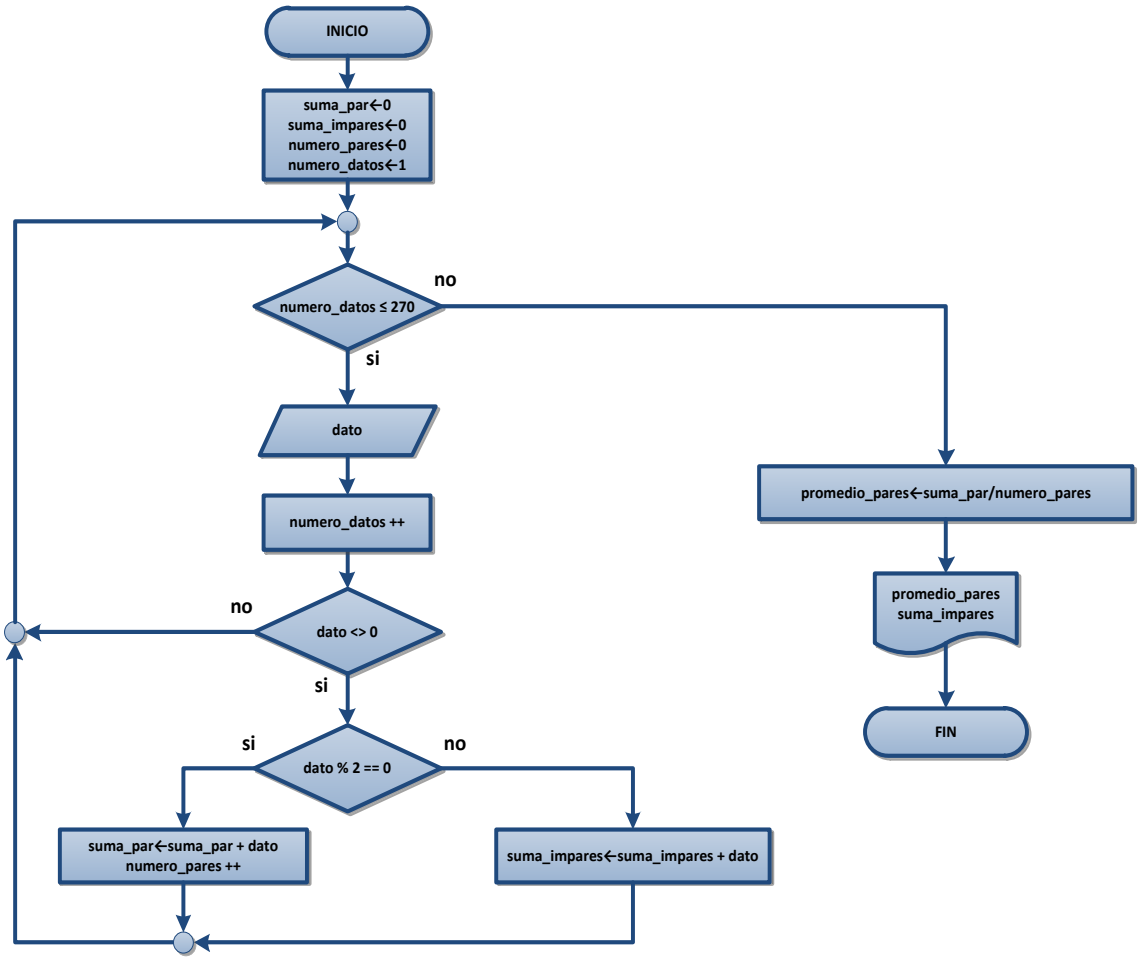
3.7.5. Hacer un programa que imprima la tabla de código ASCII.

3.7.6. Escribir un programa que reciba dos enteros y calcule la suma de los números múltiplos de 3 que se encuentren en el intervalo delimitado por los enteros recibidos.

3.7.7. Programar el algoritmo que se muestra en el siguiente diagrama de flujo:



3.7.8. Programar el algoritmo que se muestra en el siguiente diagrama de flujo:



3.7.9. Hacer un programa para encontrar las raíces de una ecuación de segundo grado de una variable real. Debe considerar los tres casos posibles, discriminante igual a cero (dos raíces reales e iguales), discriminante mayor a cero (dos raíces reales diferentes) y discriminante menor a cero (dos raíces imaginarias).

3.7.10. Escribir un programa para calcular la suma de los números primos menores o iguales a un numero n, introducido por el usuario.

UNIDAD TEMÁTICA IV
NOMBRE:
Arreglos y tipos estructurados
OBJETIVO PARTICULAR DE LA UNIDAD
Resolver problemas que requieran de arreglos de tipos de datos compuestos para representar información mixta construyendo módulos documentados que mantengan las definiciones y las operaciones necesarias.

4.1 Reactivos de opción múltiple

4.1.1. A los arreglos en C también se les conoce como:
a) arrays
b) structs
c) vectores
d) matrices

4.1.2. Es el tercer elemento del arreglo declarado como: char otro[6];
a) otro[3%1]
b) otro[3-1]
c) otro[2+1]
d) otro[3*2]

4.1.3. El acceso directo a los elementos de un arreglo puede hacerse:
a) Usando los subíndices.
b) Usando apuntadores a los elementos del arreglo.
c) Usando dobles apuntadores.
d) Todas las anteriores.

4.1.4. Este operador permite el acceso a los miembros de una estructura mediante un apuntador:
a) ->;
b) *
c) &;
d) =

4.1.5. Regresa el tamaño en bytes de su argumento
a) tam ()
b) malloc ()
c) byte ()
d) sizeof ()

4.1.6. Si tenemos lo siguiente: int *apa; es equivalente a:
apa=&a;
a) int apa=&a;
b) int *apa=a;
c) int *apa=&a;
d) int apa=a;

4.1.7. Si apa y apb son dos apuntadores, la siguiente operación es la única permitida:
a) apa*apb;
b) apa-apb;
c) apa/apb;
d) apa%apb;

4.1.8. Para pasar un arreglo a una función, el argumento de la función puede ser:
a) un apuntador a int
b) un doble apuntador
c) el nombre del arreglo
d) un triple apuntador

4.1.9. Este tipo de datos puede incluirse dentro de una estructura:
a) enteros
b) apuntadores
c) estructuras
d) Todas las anteriores

4.1.10. Permite declarar un sinónimo de tipo de dato:
a) asm
b) typedef
c) assert
d) define

4.1.11. Un CAST se refiere a:
a) conversión de tipo
b) conversión de unidades
c) conversión de funciones
d) unión de arreglos

4.1.12. Para guardar una cadena de 8 caracteres en un arreglo, con cuántos elementos se debe declarar éste:
a) 1
b) 8
c) 9
d) indefinido

4.1.13. Escribir char cadena[5]="hola"; es equivalente a:
a) char cadena[5]={'h','o','l','a'};
b) char cadena[5]={ '\0','h','o','l','a'};
c) char cadena[5]={'h','o','l','a','\0'};
d) char cadena[5]={'hola'};

4.1.14. La función strlen regresa:
a) una cadena invertida
b) una cadena en mayusculas
c) la cantidad de caracteres de la cadena
d) la cantidad de espacios en una cadena

4.2 Reactivos de falso / verdadero

4.2.1. ( ) En C el nombre de un arreglo siempre es un apuntador.
4.2.2. ( ) Es posible y conveniente aplicar la operación de multiplicación entre apuntadores.
4.2.3. ( ) No hay límite para declarar apuntadores que apunten a otros apuntadores.
4.2.4. ( ) %s es el formato de impresión para un apuntador.
4.2.5. ( ) Con el operador “coma” se accesan los elementos en una estructura.
4.2.6. ( ) Con un apuntador se hacen accesos directos.
4.2.7. ( ) Para accesar al elemento al que está apuntando un apuntador se usa una indirección.
4.2.8. ( ) Si se usa la función printf con un salto de línea o la función puts para imprimir una cadena, el resultado es exactamente el mismo.
4.2.9. ( ) En C no es posible incrementar o decrementar un apuntador.
4.2.10. ( ) En C está permitido comparar dos apuntadores usando == ó ¡=.
4.2.11. ( ) La definición de una estructura no ocupa espacio en la memoria.
4.2.12. ( ) El operador de flecha se puede representar indistintamente como -> ó =>
4.2.13. ( ) Los arreglos en lenguaje C tienen un límite de tres dimensiones.
4.2.14. ( ) El subíndice del primer elemento de un arreglo en C siempre es 1.
4.2.15. ( ) Si se intenta escribir datos más allá de los límites de un arreglo se indica un error.

4.3 Reactivos de relación de columnas

4.3.1. ( ) Declara una matriz de tipo entero con 5 filas y 3 columnas	a) tolower ()
4.3.2. ( ) Declara una matriz de tipo entero con 3 filas y 5 columnas	b) char * 1x;
4.3.3. ( ) Es el primer elemento del arreglo ejemplo	c) ->
4.3.4. ( ) Dirección de	d) struct d20 *apd;
4.3.5. ( ) Indirección	e) int bidi [3][5];
4.3.6. ( ) Un apuntador a 4 enteros	f) isalpha ()
4.3.7. ( ) Arreglo de 4 apuntadores	g) char *cadena;
4.3.8. ( ) Un apuntador a tipo char	h) int (*ap)[4];
4.3.9. ( ) Inválido	i) *apx
4.3.10. ( ) Apuntador a función	j) int (*ap)(int z);
4.3.11. ( ) Arreglo de 20 estructuras	k) struct x dir[20];
4.3.12. ( ) Operador de apuntador de estructura	l) ejemplo[0]
4.3.13. ( ) Apuntador a una estructura	m) &x
4.3.14. ( ) Pregunta si lo que está dentro de los paréntesis es una letra	n) int *ap [4];
4.3.15. ( ) Convierte el carácter a minúscula	o) int b [5][3];

4.4 Reactivos para completar la sentencia

4.4.1. El arreglo declarado como char ejemplo [8][6][9][3]: tiene un total de _____ elementos.
4.4.2. El arreglo declarado como int x[ ]={10, 20, 30, 40, 50}; tiene un total de _____elementos.
4.4.3. Además de printf, se puede usar la función _____ para imprimir una cadena.
4.4.4. El operador _____ es para obtener la dirección en memoria de un variable y el operador _____ es para indireccionar un apuntador.
4.4.5. Para que un apuntador no apunte a ningún lado, se utiliza la constante _____.
4.4.6. La función _____ permite liberar el espacio que había asignado malloc.

4.4.7. El paso de parámetros a una función se puede hacer por_____ y por _____.
4.4.8. La función _____ de la biblioteca string.h sirve para copiar dos cadenas.
4.4.9. La función _____ de la biblioteca string.h sirve para concatenar dos cadenas.
4.4.10. La función _____ de la biblioteca ctype.h sirve para convertir un caracter a mayúsculas.

4.5 Reactivos de preguntas abiertas (conceptos)

4.5.1. ¿Qué es un arreglo?
4.5.2. ¿Cuál es la diferencia entre una cadena y un arreglo de caracteres?
4.5.3. Además de scanf, ¿qué otra función sirve para recibir cadenas?
4.5.4. Escriba dos maneras para obtener la dirección del primer elemento de un arreglo llamado matriz.
4.5.5. ¿Qué es un apuntador o puntero?
4.5.6. Escriba la sintaxis para la declaración de un apuntador
4.5.7. Mencione cuales son las operaciones que se pueden hacer entre apuntadores
4.5.8. ¿Qué función realiza la función malloc?
4.5.9. ¿Qué es la función calloc()?
4.5.10. Escriba un ejemplo de asignación dinámica de memoria para 10 flotantes
4.5.11. ¿Qué pasa si malloc no puede hacer la asignación de memoria?
4.5.12. ¿Qué es una estructura?
4.51.3. ¿Cuál es la finalidad de una estructura?
4.5.14. Escriba la definición de una estructura de tipo datos con un entero, un flotante y un caracter.
4.5.15. Con la definición de la pregunta anterior declare una estructura de tipo datos que se llame a.
4.5.16. Mencione un ejemplo típico de una estructura y escribir el código de su definición y declaración de una estructura de este tipo.
4.5.17. ¿Cuál es el operador que se utiliza para el acceso a los elementos de una estructura?
4.5.18. ¿Qué es una unión?
4.5.19. ¿Cuál es la diferencia entre una unión y una estructura?
4.5.20. ¿Qué es la memoria estática?
4.5.21. ¿Qué es la memoria dinámica?
4.5.22. Escriba una función para permutar dos valores por referencia:
4.5.23. ¿Qué es una cadena?
4.5.24. ¿Cuáles son las desventajas de un arreglo estático?
4.5.25. ¿Cuáles son las ventajas de un arreglo estático?
4.5.26. ¿Cuáles son las ventajas de un arreglo dinámico?
4.5.27. ¿Cuáles son las ventajas de un arreglo dinámico?
4.5.28. ¿Para qué se usa la librería string.h?
4.5.29. ¿Para qué se usa la librería ctype.h?

4.6 Reactivos para escribir la salida de código

4.6.1. Indique la salida del siguiente programa si se introducen los valores 2, 5, 8, 1, 9, 9, 0:

```
#include <stdio.h>
#include <conio.h>
main ()                 {
        int n;
        float vector[3];
        float p;
        printf("\n\t Numero = ");
        scanf("%i", &n);
                for(int i=0; i<n; i++)                          {
                        printf("\n Numero %i ", i+1);
                        for(int j=0; j<3; j++)                          {
                                printf("\n\t Dato %i = ", j+1);
                                scanf("%f", &vector[j]);                                        }
                p = 0.0f;
                for(int i=0; i<3; i++)                  {
                        p = p + vector[i];
                        printf("\t %f", vector[i]);                  }
        printf("\n\n\t ~El resultado es %f\n", p/3);                      }
        printf("\n\t\t\t***Oprime Enter para terminar***");
getch();   }
```

4.6.2. Indique la salida del siguiente programa si se introducen los valores 2, 2, 2, 2, 2, 2:

```
#include<stdio.h>
#include<stdlib.h>

int diag(int **maz, int a,int b);

main(){
  int y,x,i,j;
  puts("Introduzca el ancho de la matriz: ");
  scanf("%d",&x);
  puts("Introduzca el largo de la matriz: ");
  scanf("%d",&y);
  int **mat;
  mat=(int**)malloc(y*sizeof(int*));
  for(i=0;i<x;i++){
        mat[i]=(int*)malloc(y*sizeof(int));
  }
  puts("Introduzca los valores de la matriz: ");
  for(i=0;i<x;i++){
        for(j=0;j<y;j++){
           scanf("%d",&mat[i][j]);
        }
  }
  putchar ('\n');
```

```
  puts ("La matriz resultante es: ");
  diag(mat,x,y);
}

int diag(int **maz, int a,int b){
    int i,j;
    for(j=0;j<b;j++){
        for(i=0;i<a;i++){
           if(i==j){
             printf("   ");
           }
           else{
           printf(" %d ",maz[j][i]);
           }
        }
        printf("\n");
    }
    system ("pause");
}
```

4.6.3. Indique la salida del siguiente programa si se introducen los valores 2, 2, 2, 2, 2, 2:

```
#include<stdio.h>
#include<stdlib.h>

int compara(int **maz, int anc,int larg);

main(){
  int y,x,i,j,cc;
  puts("Introduzca el largo de la matriz: ");
  scanf("%d",&x);
  puts("Introduzca el ancho de la matriz: ");
  scanf("%d",&y);
  int **mat;
  mat=(int**)malloc(y*sizeof(int*));
  for(i=0;i<x;i++){
        mat[i]=(int*)malloc(y*sizeof(int));
  }
  puts("Introduzca los valores de la matriz: ");
  for(i=0;i<x;i++){
        for(j=0;j<y;j++){
           scanf("%d",&mat[i][j]);
        }
  }

  printf("\n\n");
  for(i=0;i<x;i++){
        for(j=0;j<y;j++){
           printf ("%d\t", mat[i][j]);
        }
        putchar ('\n');
  }
  compara(mat,x,y);
```

```
}

int compara(int **maz, int anc,int larg){
    int i,j,n=0;
    for(i=0;i<anc-1;i++){
        for(j=i+1;j<larg;j++){
            if(maz[j][i]==0){
            }
            else{
                n=n+1;
            }
        }
    }
    if(n>0){
         puts("La matriz NO es triangular superior: 0");
    }
    else{
         puts("La matriz SI es triangular superior: 1");
    }
    system ("pause");
}
```

4.6.4. Indique la salida del siguiente programa si se introducen los datos Juan, Estructuras de datos, 9.2:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

struct Materia {
    char nombre [MAX];
    char materia [MAX];
    float calif;
};

struct Materia nuevo(void);
void imprimeDatos(struct Materia MATERIA);

 main ()
{
    struct Materia recibe;
    recibe = nuevo();
    imprimeDatos(recibe);
    system ("pause");
}

struct Materia nuevo(void)
{
    struct Materia MATERIA;
    puts("Introduzca un nombre: ");
    gets(MATERIA.nombre);
    puts("Introduzca una unidad academica: ");
    gets(MATERIA.materia);
```

```
    puts("Introduzca la calificacion obtenida: ");
    scanf("%f", &MATERIA.calif);
    fflush(stdin);
    return(MATERIA);

}

void imprimeDatos(struct Materia MATERIA)
{
    puts("DATOS DEL ALUMNO");
    printf("Nombre: %s \n", MATERIA.nombre);
    printf("Materia: %s \n", MATERIA.materia);
    printf("Calificacion: %f \n", MATERIA.calif);
    if (MATERIA.calif<6)
        puts("REPROBADO");
    else
        puts ("APROBADO");
}
```

4.6.5. Indique la salida del siguiente programa si se introducen los valores 2, 2, 2, 2, 2, 2:

```
#include <stdio.h>
#include <conio.h>
#define F 10
#define C 10

main ()
{
    int m[F][C];
    int sf;
    int f, c;
    int i, j;

    do
    {
       printf ("introduce el numero de filas de la matriz:\n");
       scanf ("%d",&f);
       }
    while (f<1 || f>F);
    do
    {
         printf ("introduce el numero de columnas de la matriz:\n");
         scanf ("%d",&c);
         }
     while (c<1 || c>C);
     printf ("introducir los valores de la matriz\n");
     for   (i=0; i<f; i++)
     for (j=0; j<c; j++)
     {
        printf ("m[%d][%d]= ", i, j);
        scanf ("%d", &m[i][j]);
        }
     for (i=0; i<f; i++)
     {
```

```
        sf=0;
        for (j=0; j<c; j++)
        sf += m[i][j];
        printf ("el resultado es = %d=%d\n",i,sf);
        }
        getch ();
        }
```

4.6.6. Indique la salida del siguiente programa si se introducen los datos Ana, av. 522, 123456678, 11, 09, 1990:

```
#include <stdio.h>
#include <conio.h>

typedef struct {
      int d, m, a;
      } f;

typedef struct {
       char n [50];
       char d [50];
       int t;
       f fn;
       } d;
main ()
{
    d p, p2;
    printf ("NOMBRE:        \n");
    gets (p.n);
    printf ("DIRECCION:      \n");
    gets (p.d);
    printf ("TELEFONO:      \n");
    scanf ("%d", &p.t);
    printf ("FECHA DE NACIMIENTO:\n");
    printf (" Dia:       ");
    scanf ("%d", &p.fn.d);
    printf (" Mes:       ");
    scanf ("%d", &p.fn.m);
    printf (" Año:      ");
    scanf ("%d", &p.fn.a);

    p2=p;
    printf ("\n\nNOMBRE:        %s\n", p2.n);
    printf ("DIRECCION:        %s\n", p2.d);
    printf ("TELEFONO:        %d\n", p2.t);
    printf ("FECHA DE NACIMIENTO:\n");
    printf (" Dia:        %d\n",p2.fn.d);
    printf (" Mes:        %d\n",p2.fn.m);
    printf (" Año:        %d\n",p2.fn.a);

    getch();
    }
```

4.6.7. Indique la salida del siguiente programa si se introducen los datos El profesor se fue a comer, y ya mero termino esto =D:

```
#include <stdio.h>
#include <conio.h>

main ()
{
    const int SI=1;
    const int NO=0;
    int palabra=NO;
    int ncar=0, npal=0, nlin=0;
    char car;
    printf ("introduce una linea de texto, pulsando enter despues de cada linea:\n");
    printf ("pulse ctrl+z para mostrar\n\n");
    while ((car=getchar ())!=EOF)
    {
        ++ncar;
        if (car==' ' || car=='\n' || car=='\t')
        palabra=NO;
        else if (palabra==NO)
        {
            ++npal;
            palabra=SI;
            }
        if (car=='\n')
        ++nlin;
        }
        printf ("%d caracteres\n",ncar);
        printf ("%d palabras\n", npal);
        printf ("%d lineas \n", nlin);

        getch ();
        }
```

4.6.8. Indique la salida del siguiente programa si se introducen los valores 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1:

```
#include <stdio.h>
#include <conio.h>
#define TAM 100

int simetrica (int mat [TAM][TAM], int f,int c);
void imprime (int mat [TAM][TAM], int f, int c);
void ingresa (int mat [TAM][TAM], int f, int c);

main ()
{
    int arr[TAM][TAM];
    int fil, col;
    puts ("tamaño de la matriz");
    puts ("Ingrese numero de filas");
    scanf ("%d",&fil);
```

```
    puts ("Ingrese numero de columnas");
    scanf ("%d",&col);
    puts ("ingresa la matriz");
    ingresa (arr,fil,col);
    imprime (arr,fil,col);
    getch ();
    return 0;
    }

void ingresa (int mat [TAM][TAM], int f, int c)
{
    for (int i=0; i<f; i++)
    for (int j=0; j<c; j++)
    scanf ("%d",&mat [i][i]);
    }

void imprime (int mat [TAM][TAM], int f, int c)
{
    for (int i=0; i<f; i++)
    for (int j=0; j<c; j++)
    printf ("%d %c", mat[i][j], j==c-1?'\n':' ');
    }

int simetrica (int mat [TAM][TAM], int f,int c)
{
   for (int i=0; i<f; i++)
   for (int j=0; j<c; j++)
       if (mat [i][j]!=mat [j][i])
       return 0;
       return 1;
       }
```

4.6.9. Indique la salida del siguiente programa si se introduce: hola mundo como estamos todos por alla

```
#include <stdio.h>
#include <conio.h>

main ()
{
   char cadena[100]={'\0'};
   int c=0;
   int i;

   printf("Ingrese una cadena de texto: ");
   gets(cadena);
   for(i=0;cadena[i]!='\0';i++)
   {
       if(
       cadena[i]=='A'||
       cadena[i]=='a'||
       cadena[i]=='E'||
       cadena[i]=='e'||
       cadena[i]=='I'||
```

```
        cadena[i]=='i'||
        cadena[i]=='O'||
        cadena[i]=='o'||
        cadena[i]=='U'||
        cadena[i]=='u')
        {
           c++;
        }
    }
    printf("\nEl resultado es: %d\n",c);
    getch();
}
```

4.6.10. Indique la salida del siguiente programa

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(){
    int b=5;
    int * puntero1;
    int resultado;
    puntero1=&b;

    resultado=10+*puntero1;
    printf("resultado: %d\n",resultado);


    b=3;
    resultado=10+*puntero1;
    printf("resultado: %d\n",resultado);

    b=5;
    printf("b : %d\n",b);
    printf("puntero1 : %d\n",puntero1);
    printf("*puntero1 : %d\n",*puntero1);

 puts("");
  system("pause");
}
```

4.6.11. Indique la salida del siguiente programa

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void){
    char vector_caracteres[] ="Soy la vieja manera";
    printf(" %s\n",vector_caracteres);
```

```
    char * cad = "ESCUELA SUPERIOR DE COMPUTO";

    printf(" %s\n",cad);


    char * cad2;
    cad2= &vector_caracteres[0];

    printf(" %s\n",cad2);

 puts("");

  system("pause");
```

4.6.12. Indique la salida del siguiente programa si se introducen los valores 2, 3, 4:

```
#include <stdio.h>
#include <conio.h>

void funcion1(int a);
void funcion2(int b);
void funcion3(int c);

int main()
{
   void (*f[3])(int) = {funcion1, funcion2, funcion3};
   int eleccion;

   printf("Escoja un numero:\n");
   printf ("\n1. Opcion 1\n2. Opcion2\n3. Opcion3\n4. Salir\n");
   scanf("%d", &eleccion);

   while (eleccion > 0 && eleccion <4){

      (*f[eleccion-1])(eleccion);
      printf("Escoja un numero:\n");
      printf ("\n1. Opcion 1\n2. Opcion2\n3. Opcion3\n4. Salir");
      scanf("%d", &eleccion);
   }
getch ();
}
void funcion1(int a)
{
   printf ("\n 1\n");
}
void funcion2(int b)
{
   printf ("\n 2\n");
}
void funcion3(int c)
{
   printf ("\n 3\n");
}
```

4.6.13. Indique la salida del siguiente programa

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

void cs (char *ptrS);

int main ()
{
   char cadena [] = "La Escuela Superior de Computo";
   printf ("La cadena antes: %s", cadena);
   cs (cadena);

   printf ("\n\nLa cadena despues: %s \n", cadena);

   getch();
}

void cs (char *ptrS)
{
    while (*ptrS != '\0')
    {
        if (islower (*ptrS) )
        {
            *ptrS = toupper (*ptrS);
        }
        ++ptrS;
    }
}
```

4.6.14.Indique la salida del siguiente programa si se introduce: david

```
#include<stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>

main()
{
      char cad[80];
      int i,j;
      puts("\n\n\ Escribe una cadena: \n");
      gets(cad);
      fflush(stdin);
      puts("\n\n Ps: \n");
      for(i=0;i<strlen(cad);i++){
      printf("\nP%d : ",i+1);
          for(j=0;j<=i;j++)
              printf("%c",cad[j]);
      }
      puts("\n\n Ss: \n");
```

```
      for(i=strlen(cad);i>0;i--){

      printf("\nS%d : ",strlen(cad)-i+1);
         for(j=strlen(cad)-1;j>=i-1;j--)
            printf("%c",cad[j]);
      }
getch();
}
```

4.6.15. Indique la salida del siguiente programa

```
#include <stdio.h>
#include <conio.h>

void despliegaArreglo (const int a[][3]);

int main ()
{
    int arr1 [2][3] = {{1,2,3}, {4,5,6}};
    int arr2 [2][3] = {1, 2, 3, 4, 5};
    int arr3 [2][3] = {{1,2},{4}};

    printf ("Los valores en el Arreglo 1 por linea son: \n ");
    despliegaArreglo (arr1);

    printf ("\nLos calores en el Arreglo 2 por linea son: \n ");
    despliegaArreglo (arr2);


    printf ("\nLos valores en el Arreglo 3 por linea son: ");
    despliegaArreglo (arr3);
getch ();
}

void despliegaArreglo (const int a[][3])
{
    int i;
    int j;

    for (i = 0; i <= 1; i++)
    {
       for (j = 0; j <= 2; j++);
       {
          printf ("%d", a[i][j]);
       }
    }
}
```

4.6.16. Indique la salida del siguiente programa si se introducen los valores 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2:

```
#include <stdio.h>
#include <conio.h>

void llenar (int M[20][20], int f, int c);
void mostrar (int M[20][20], int f, int c);
void res (int R[20][20], int fr, int cr, int S[20][20], int fs, int cs);

int main()
{
    int R [20] [20];
    int S [20] [20];
    int fr, cr, fs, cs;
    printf("Cuantas filas en R: ");
    scanf("%d",&fr);
    printf("Cuantas columnas en R: ");
    scanf("%d",&cr);
    printf("Cuantas filas en S: ");
    scanf("%d",&fs);
    printf("Cuantas columnas en S: ");
    scanf("%d",&cs);

    printf("\nLlenando la Matriz R: \n");
    llenar (R, fr, cr);
    printf("\nLlenando la Matriz S: \n");
    llenar (S, fs, cs);
    printf("\nLa Matriz R : ");
    mostrar (R, fr, cr);
    printf("\n\nLa Matriz S : ");
    mostrar (S, fs, cs);
    res (R, fr, cr, S, fs, cs);
getch();
}

void llenar (int M[20][20], int f, int c)
{
 for (int i = 1 ; i <= f ; i++)
 {
  for (int j = 1 ; j <= c ; j++)
  {
   printf ("Ingresa la posicion [%d][%d]: ",i,j);
   scanf("%d",&M [i] [j]);
  }
 }
}

void mostrar (int M[20][20], int f, int c)
{
 for (int i = 1 ; i <= f ; i++)
 {
  printf("\n");
  for (int j = 1 ; j <= c ; j++)
```

```
  {
   printf ("[%d]",M [i] [j]);
  }
 }
}

void res (int R[20][20], int fr, int cr, int S[20][20], int fs, int cs)
{
 printf("\n\nResultado: ");
 for (int i = 1 ; i <= fr ; i++)
 {
  for (int j = 1 ; j <= cr ; j++)
  {
   for (int k = 1 ; k <= fs ; k++)
   {
    for (int l = 1 ; l <= cs ; l++)
    {
     if (R [i] [j] == S [k] [l])
        printf ("[%d]",R [i] [j]);
    }
   }
  }
 }
}
```

4.7 Reactivos de problemas para programar

4.7.1. Hacer un programa que cargue del teclado dos matrices enteras A y B de orden 3x3, y que visualice la matriz producto P = A · B.

4.7.2. Escribir un programa que reciba las dimensiones de una matriz, luego reciba los datos de ésta y posteriormente encuentre su matriz transpuesta.

4.7.3. Hacer un programa que reciba los datos de una matriz de 3x3 y la rote a la derecha 90° tantas veces como se quiera, debe comprobarse que después de 4 rotaciones se regresa a la matriz original.

4.7.4. Escribir un programa para leer una cadena y mostrar una estadística de las longitudes de las palabras, esto es, el número total de palabras de longitud 1 que hayan ocurrido, el total de longitud 2 y así sucesivamente.

4.7.5. Hacer un programa que reciba cinco cadenas desde teclado y las imprima en orden alfabético

4.7.6. Crear un programa que contenga una función llamada copiarArray que reciba dos arrays y el tamaño de los mismos (deben de ser del mismo tamaño) y que consiga copiar en el segundo array el contenido del primero.

4.7.7. Crear un programa que cree un array de 100 números aleatorios del 1 al 1000. Una vez creado, mostrar el contenido y después organizarlo de forma que estén juntos los elementos pares y los impares. Después, volver a mostrar el array.

4.7.8. Crear un programa que mediante un menú admita reservar o cancelar asientos de un avión, así como mostrar qué asientos están ocupados y libreas actualmente. El array tendrá 25 filas y 4 columnas.

4.7.9. Escribir un programa que cree y muestre en pantalla un cuadrado mágico de orden 3. Un cuadrado mágico es aquél en el que todas las filas y columnas suman lo mismo. Para ello, se coloca el valor 1 en el medio de la 1ª fila. Los siguientes valores (2, 3, 4, ...) se sitúan en la fila de arriba, columna de la derecha, salvo que esté ocupada, en cuyo caso se coloca inmediatamente debajo. Se supone que las filas y columnas de los extremos son adyacentes. Por ejemplo:

8	1	6
3	5	7
4	9	2

4.7.10. Hacer un programa que dada una cadena y dos caracteres, sustituya las apariciones del primer caracter por el segundo. Por ejemplo: (Esta es la cadena, a, x) producirá (Estx es lx cxdenx).

4.7.11. Hacer un programa que reciba dos caracteres desde el teclado y genere una cadena con todos los caracteres contenidos entre ellos, por ejemplo, si recibe la ‘a’ y la ‘d’, debe generar la cadena “abcd”, si recibe la ‘m’ y la ‘p’, debe generar “mnop”.

4.7.12. Hacer un programa que reciba una cadena desde el teclado y cambie el orden de las palabras de tal manera que la primera palabra sea la última y viceversa, por ejemplo, si se introduce la cadena “Algoritmia y programación estructurada”, debe producir “estructurada programación y Algoritmia”.

UNIDAD TEMÁTICA V
NOMBRE:
Archivos e integración de conceptos
OBJETIVO PARTICULAR DE LA UNIDAD
Construir un programa que utilice arreglos de estructuras con la capacidad de ordenar sus elementos bajo algún criterio específico y almacenar la información en un archivo para integrar los conceptos aprendidos utilizando buenas prácticas de programación.

5.1 Reactivos de opción múltiple

5.1.1. Esta función sirve para recibir datos especificando el flujo de entrada:
a) read
b) fscanf
c) scanf("%f")
d) fopen

5.1.2. Es un modo de apertura válido:
a) a
b) b
c) ap
d) bp

5.1.3. Los ficheros de texto finalizan con:
a) FILE
b) EFP
c) EOF
d) \0

5.1.4. La biblioteca a la que pertenece la definición de tipo FILE es:
a) stdio.h
b) stdlib.h
c) string.h
d) ctype.h

5.1.5. La función para cerrar un fichero es:
a) exit
b) fclose
c) close
d) no es necesario cerrar un fichero

5.1.6. En este tipo de archivo, los datos están guardados en bits, su acceso es directo:
a) binarios
b) de texto
c) .txt
d) .cpp

5.1.7. Mientras ejecuta, un programa en C guarda los datos en:
a) ROM
b) EPROM
c) RAM
d) HDD

5.1.8. Es una estructura cuyos miembros se utilizan para abrir una archivo:
a) NULL
b) FILE
c) EOF
d) AB+

5.2 Reactivos de falso / verdadero

5.2.1. ( ) fflush sirve para liberar los flujos.
5.2.2. ( ) Los tipos de flujos son de texto y binarios.
5.2.3. ( ) C ve a un archivo como un flujo secuencial de bytes.
5.2.4. ( ) El dispositivo que usa el flujo de error estándar son las bocinas.
5.2.5. ( ) El flujo de texto utiliza un conjunto de caracteres organizado en líneas.
5.2.6. ( ) En el nombre del archivo puede incluirse la ruta.
5.2.7. ( ) Los argumentos de la función fopen son apuntadores a carácter.
5.2.8. ( ) La entrada y salida directas se emplean con archivos en modo texto.
5.2.9. ( ) La función fread() regresa la cantidad de elementos que se pudieron leer.
5.2.10. ( ) Un fichero de texto almacena la información como secuencia de códigos ASCII.

5.3 Reactivos de relación de columnas

5.3.1. ( ) Modo de apertura para añadir
5.3.2. ( ) Modo de apertura para un archivo en modo binario
5.3.3. ( ) Función para archivos en modo texto
5.3.4. ( ) Función para archivos en modo binario
5.3.5. ( ) Permite leer de un archivo un carácter
5.3.6. ( ) Guarda una cadena en un archivo
5.3.7. ( ) Abre el archivo sólo para lectura
5.3.8. ( ) Abre el archivo para lectura y escritura
5.3.9. ( ) Permite borrar un archivo
5.3.310. ( ) Sirve para cambiar el nombre de un archivo

a) r+
b) fwrite
c) rename
d) fscanf
e) a
f) fgetc
g) remove
h) wb+
i) fputs
j) r

5.4 Reactivos para completar la sentencia

5.4.1. Para abrir un archivo se utiliza la función _____.
5.4.2. La función fprintf está contenida en la biblioteca_____.
5.4.3. La función fopen está contenida en la biblioteca_____.
5.4.4. Los argumentos de la función fopen son _____ y _____.
5.4.5. El flujo de entrada estándar es _____ y el dispositivo que utiliza es _____.
5.4.6. El flujo de salida estándar es _____ y el dispositivo que utiliza es _____.
5.4.7. El flujo de entrada estándar es _____ y el dispositivo que utiliza es _____.

5.7.8. _____ es el modo de apertura que abre el archivo para lectura y escritura. Si no existe, lo crea y si existe se sobrescriben los datos existentes.
5.4.9. La función fread() tiene _____ argumentos.
5.4.10. Un archivo en modo _____ no es legible para una persona.

5.5 Reactivos de preguntas abiertas (conceptos)

5.5.1. ¿Qué es un flujo de datos?
5.5.2. ¿Para qué se declara un apuntador a tipo FILE?
5.5.3. Escriba el prototipo de la función fopen:
5.5.4. ¿Qué regresa fopen si no puede abrir el archivo?
5.5.5. Mencione los modos de apertura de un archivo en modo texto:
5.5.6. Mencione las formas para entrada y salida (E/S) de datos de un archivo
5.5.7. De las formas anteriores, escribir las funciones que se utilizan para cada una:
5.5.8. ¿Para qué sirve la función fwrite()?
5.5.9. ¿Para qué sirve la función fread()?
5.5.10. ¿Qué indica la constante EOF?
5.5.11. Explique cómo funciona la función feof ()
5.5.12. ¿Qué hace la función fgetc()?
5.5.13. ¿Qué hace la función fputc ()?
5.5.14. ¿Qué hace la función fseek()?
5.5.15. ¿Cuáles son los argumentos de la función fseek?

5.6 Reactivos para escribir la salida de código

5.6.1. ¿Cuál es el resultado del siguiente programa si la cadena de entrada es: anita la huerfanita

```
#include <stdio.h>
#include <conio.h>
#define TAM 100

void fun (char cad[TAM], int tam, FILE *pa);
main ()
{
    FILE *ap;
    char val;
    char cadena [TAM];
    int lectura (char cad [TAM]);
    puts ("ingresa una cadena");
    gets(cadena);
    printf ("\n");
    ap=fopen("cadena.txt", "w+");
    fputs(cadena,ap);
    printf ("la cadena resultante es:\n");
    val=lectura(cadena);
    fun (cadena,val, ap);
    fclose(ap);
    getch();
    }
void fun (char cad [TAM], int tam, FILE *pa)
```

```
{
    int i;
    for (i=tam-1; i>=0; i--)
    {
        printf ("%c",cad[i]);
        fputc(cad[i],pa);
        }
        }
int lectura (char cad [TAM])
{
   int conta=0;
   while (cad[conta]!='\0')
   conta ++;
   return (conta);
}
```

5.6.2 Determine que es lo que escribe en el archive el siguiente programa, introduciendo los siguientes datos: 789456213, david, gomez,12003.63

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

struct datos{
      int numero;
      char nombre[50];
      char apellido[50];
      float cant;
      };

int main ()
{
    FILE *datos = NULL;
    struct datos x;
    x.numero = 2;

    while(x.numero!=0) {
       printf("Dame un numero: ");
       scanf("%d",&x.numero);

       if(x.numero != 0) {
          fflush(stdin);
          if(!datos) {
             if(!(datos=fopen("credito.txt","a+"))) {
                printf("\nError");
                system("pause");
                exit(0);
             }
          }
          printf("Dame un nombre :");
          scanf("%s",&x.nombre);
          printf("\nDame tu apellido: ");
          scanf("%s",&x.apellido);
          printf("\nDame la cantidad: ");
```

```
            scanf("%f",&x.cant);

            fwrite (&x,sizeof(struct datos),1,datos);
        }
    }
    fclose(datos);
    system("pause");
    return 0;
}
```

5.6.3. Determine la salida del siguiente programa si se le ingresan los siguientes datos: 2, david, gomez, dhf201

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

struct {
char matricula;
char nombre [30];
char apellido [15];
char curp [20];} alumno;

main() {

      fflush;

      printf ("Dame tu Matricula: ");
      scanf ("%s",&alumno.matricula);
      printf ("Dame tu Nombre: ");
      scanf ("%s", &alumno.nombre);
      printf ("Dame tu Apellido: ");
      scanf("%s",&alumno.apellido);
      printf ("Dame tu Curp: ");
      scanf("%s",&alumno.curp);

      FILE *archdisco;
      archdisco = fopen("al.txt","at+");
      fwrite(&alumno,sizeof(alumno),1,archdisco);
      fclose(archdisco);

      getch();

}
```

5.6.4. Determine cual es la salida tanto en consola como en el fichero, del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
```

```
    int hora = time(NULL);
    int li,ls;
    int cnt;
    int v= 21;
    FILE *a=NULL;
    int n;

    if(!(a=fopen("ex.txt","at+"))) {
        printf("\nerror");
        system("pause");
        exit(-1);
    }

    for(cnt = 0; cnt < v; cnt++) {
        n = ((rand()%1)+100);
        printf("%d,",n);
        printf("%d\n",(cnt%2 == 0 ? 1 : 0));
        fwrite(&n,sizeof(int),1,a);
    }

    fclose(a);
    printf("\n");
    system("pause");
    return 0;
}
```

5.6.5. Determine que es lo que escribe el siguiente programa en el archivo:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int hora = time(NULL);
    int li = 65,ls = 90;
    int cnt;
    int v;
    FILE *a=NULL;
    int n;

    if(!(a=fopen("da.txt","at+"))) {
        printf("\nerror");
        system("pause");
        exit(-1);
    }

    for(cnt = li; cnt < ls; cnt++)
        fwrite(&cnt,sizeof(int),1,a);

    fclose(a);
    printf("\n");
    system("pause");
    return 0;
}
```

5.6.6. Determine que es lo que el programa guarda en el archivo si se introduce el siguiente dato:
?

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int hora = time(NULL);
    int li = 65,ls = 90;
    int cnt;
    int v = 5,v2 = 10;
    FILE *a=NULL;
    int n;
    char c;
    int contador;

    if(!(a=fopen("sr.txt","at+"))) {
        printf("\nerror");
        system("pause");
        exit(-1);
    }

    printf("Escribe un simbolo: ");
    scanf("%c",&c);

    for(cnt = 0; cnt < v; cnt++ ){
        for(contador = 0; contador < v2; contador++)
            fprintf(a,"%c",c);
        fprintf(a,"\n");
    }

    fclose(a);
    printf("\n");
    system("pause");
    return 0;
}
```

5.7 Reactivos de problemas para programar

5.7.1. Programar una agenda, con opciones de agregar nuevo contacto, buscar algun contacto, ver la lista de contactos, modificar contacto, y borrar contacto. Cada registro (contacto) debe contener: nombre, apellido, teléfono, calle y número, y el e-mail. Los registros deben guardarse en un archivo en modo texto y recuerarse del mismo.

5.7.2. Hacer un programa que, dada una cadena, despliegue sus caracteres en forma de lista, indicando si es mayúscula o minúscula, así como su equivalente en ASCII, debe enviar esta información a un archivo en modo texto.

5.7.3. Hacer un programa que despliegue el calendario de un año dado (incluyendo los meses y días de la semana) y lo guarde en un archivo que lleve por nombre calendario_aaaa.

5.7.4. Hacer un programa que reciba dos cadenas desde el teclado en las que aparezcan dos números (enteros). Estos números se deben extraer de las cadenas para obtener su suma. Se debe generar una tercera cadena que indique el resultado y mandarse a un archivo. Por ejemplo, si se introducen las cadenas: “el primer número es el 124” y “el segundo número es el 216”, la cadena que se envía al archivo es “el resultado de la suma de 124 y 216 es 340”

5.7.5. Quizás el más famoso de todos los sistemas de codificación es el código Morse, desarrollado por Samuel Morse en 1832, para uso en el sistema telegráfico. El código Morse asigna una serie de puntos y rayas a cada letra del alfabeto, a cada dígito y a unos cuantos caracteres especiales. La separación entre palabras se indica por un espacio o por la ausencia de un punto o una raya. La versión internacional del código Morse aparece en la tabla siguiente:

Carácter	Código	Carácter	Código	Números	
A	.-	N	-.	1	.----
B	-...	O	---	2	..---
C	-.-.	P	.--.	3	...—
D	-..	Q	--.-	4	....-
E	.	R	.-.	5	.....
F	..-.	S	...	6	-....
G	--.	T	-	7	--...
H	....	U	..-	8	---..
I	..	V	...-	9	----.
J	.---	W	.--	0	-----
K	-.-	X	-..-		
L	.-..	Y	-.--		
M	–	Z	--..		

El programa deberá incorporar una función Menu() que muestre las siguientes opciones: 1) Pasar una frase a código Morse, lo cual se implementará en una función que se llame Frase2Morse(), 2) Pasar código Morse a una frase, implementando una función que se llame Morse2Frase() y 3) Salir.

Escribir un programa que lea una frase escrita en español, cifre dicha frase en código Morse y la envíe a un archivo en modo binario, además debe leer un archivo conteniendo una frase en código Morse y la debe convertir en el equivalente en español. Utilice un espacio en blanco entre cada letra codificada Morse y tres espacios en blanco entre cada palabra codificada en Morse.

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

DEPARTAMENTO DE CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN

ACADEMIA DE CIENCIAS DE LA COMPUTACIÓN

CARRERA:

Ingeniería en Sistemas Computacionales

PROFESOR:

José Carlos Dávalos López.

RESPUESTAS A LOS REACTIVOS DE EXAMEN PARA LA UNIDAD DE APRENDIZAJE:

Algoritmia y Programación Estructurada

NOTA:

Cabe señalar que los reactivos **de problemas para elaborar algoritmos y para programar**, no se da respuesta a ellos, ya que existen múltiples formas de diseño y de codificación para solventar o resolver el problema planteado, se deja a consideración tanto del profesor como del alumno.

NIVEL:

Primero

OBJETIVO GENERAL DE LA UNIDAD DE APRENDIZAJE:

Resolver problemas computacionales, de dificultad similar al ordenamiento, utilizando algoritmos, pseudo-código y buenas prácticas de programación para adquirir la capacidad de construir y desarrollar programas en un lenguaje de alto nivel.

CONTENIDOS:

Unidad Temática I. Conceptos básicos y herramientas de programación.
Unidad Temática II. Modularidad
Unidad Temática III. Control de flujo
Unidad Temática IV. Arreglos y tipos estructurados.
Unidad Temática V. Archivos e integración de conceptos.

UNIDAD TEMÁTICA I

1.1 Reactivos de opción múltiple

1.1.1. d)
1.1.2. d)
1.1.3. d)
1.1.4. b)
1.1.5. d)
1.1.6. b)
1.1.7. a)
1.1.8. c)
1.1.9. c)
1.1.10. d)

1.2 Reactivos de falso / verdadero

1.2.1. V
1.2.2. F
1.2.3. F
1.2.4. V
1.2.5. F
1.2.6. V
1.2.7. V
1.2.8. F
1.2.9. F
1.2.10. F
1.2.11. F
1.2.12. V

1.3 Reactivos de relación de columnas

1.3.1. (f)
1.3.2. (c)
1.3.3. (d)
1.3.4. (i)
1.3.5. (a)
1.3.6. (e)
1.3.7. (g)
1.3.8. (b)
1.3.9. (j)
1.3.10. (h)

1.4 Reactivos para completar la sentencia

1.4.1. Von Neumann
1.4.2. Harvard
1.4.3. Pseudo-código
1.4.4. Ovalo.
1.4.5. Rombo.

1.4.6. Rectángulo.
1.4.7. const
1.4.8. %u
1.4.9. %s
1.4.10. \\
1.4.11. \t
1.4.12. iteración

1.5 Reactivos de preguntas abiertas (conceptos)

1.5.1. Es una herramienta que se utiliza para representar cualquier situación de la realidad en forma de datos, los cuales se procesan después para generar información.
1.5.2. Es el conjunto de estructuras tanto físicas como lógicas que influyen de manera directa en las funciones y diseño del Hardware de una máquina, también influencia en el Software; siendo el objetivo primordial de la arquitectura el aumento del rendimiento de las computadoras.
1.5.3. Es en donde se cargan las instrucciones y programas que ejecuta el procesador y llevar a cabo su objetivo. Se denominan de acceso aleatorio porque se pueden sobrescribir datos.
1.5.4. Es un conjunto preescrito de instrucciones oreglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar la actividad.
1.5.5. Lenguaje natural, pseudo-código, diagramas de flujo y lenguajes de programación.
1.5.6. Del griego y latín dixit algorithmus y éste a su vez del matemático persa Al Juarismi.
1.5.7. Un lenguaje de programación es el medio a través del cual le comunicamos a la computadora la secuencia de instrucciones que debe ejecutar para llevar a cabo actividades, tareas o solución de problemas.
1.5.8. Es un conjunto de instrucciones que guían a la computadora para realizar alguna actividad o resolver algún problema; en el programa se ejecutan diferentes acciones de acuerdo con los datos que se estén procesando.
1.5.9. Es una representación grafica de un algoritmo. Estos diagramas utilizan símbolos con significados definidos que representan los pasos del algoritmo y representan el flujo de ejecución mediante flechas que conectan los puntos de inicio y de término.
1.5.10. Es una técnica para diseño de programas (algoritmos) que permite definir las estructuras de datos, las operaciones que se aplicarán a los datos y la lógica que tendrá e l programa para solucionar u determinado problema. Utiliza un pseudolenguaje muy parecido a nuestro idioma pero que respeta las directrices y los elementos de los lenguajes de programación.
1.5.11. Es un traductor que convierte un texto escrito en un lenguaje fuente de alto nivel en un programa objeto en código máquina.
1.5.12. 36
1.5.13. 40
1.5.14. /* */

1.6 Reactivos para escribir la salida de código

1.6.1.

La expresión sin paréntesis evalúa a 1
La expresión con paréntesis evalúa a 0

1.6.2.

Ingrese el valor de c1: 25
Ingrese el valor de c2: 14
Valor de a1: 29.2488
Valor de a2: 29.2488
Valor de h: 28.6531

1.6.3.

Ingresa un numero entero:
23
el 23 es numero par

1.7 Reactivos de problemas para elaborar algoritmos y para programar

Ver nota.

UNIDAD TEMÁTICA II

2.1 Reactivos de opción múltiple

2.1.1. a)
2.1.2. d)
2.1.3. c)
2.1.4. d)
2.1.5. c)
2.1.6. d)
2.1.7. b)
2.1.8. a)

2.2 Reactivos de falso / verdadero

2.2.1. F
2.2.2. V
2.2.3. F
2.2.4. V
2.2.5. F
2.2.6. F
2.2.7. V
2.2.8. F
2.2.9. F
2.2.10. F

2.3 Reactivos de relación de columnas

2.3.1. (d)
2.3.2. (e)
2.3.3. (i)
2.3.4. (j)
2.3.5. (h)
2.3.6. (a)
2.3.7. (f)
2.3.8. (b)
2.3.9. (c)
2.3.10. (g)

2.4 Reactivos para completar la sentencia

2.4.1. locales
2.4.2. globales
2.4.3. stdio.h
2.4.4. math.h
2.4.5. #include.
2.4.6. procedimientos
2.4.7. #define
2.4.8. return
2.4.9. stdio.h
2.4.10. char func (float z);

2.5 Reactivos de preguntas abiertas (conceptos)

2.5.1. El método está basado en la resolución recursiva de un problema dividiéndolo en dos o más sub-problemas de igual tipo o similar. El proceso continúa hasta que éstos llegan a ser lo suficientemente sencillos como para que se resuelvan directamente. Al final, las soluciones a cada uno de los sub-problemas se combinan para dar una solución al problema original.
2.5.2. Una función es un conjunto de declaraciones, definiciones, expresiones y sentencias bajo el mismo nombre que realizan una tarea específica.
2.5.3. Es la declaración de una función que se va a utilizar después (cabecera).
2.5.4. <tipo_retorno> <nombre> <argumentos>;
2.5.5. No regresa nada y recibe 2 enteros
2.5.6 Las variables globales se pueden usar en cualquier función y las locales sólo pueden usarse en la función en que se declaran.
2.5.7. La función devuelve uno o más valores y el procedimiento no.
2.5.8. Se define un conjunto de funciones, así como tipos relacionados y macros, que son proporcionados para la implementación. Todas las bibliotecas son declaradas en un fichero cabecera.
2.5.9. Como las funciones siempre retornan un valor, el uso de una función consiste en utilizar el valor de retorno. Se lo puede hacer de dos formas: Almacenar el valor de retorno en una variable que deberá ser del mismo tipo de dato que el tipo de dato de retorno de la función. Utilizar el valor de retorno en una expresión.
2.5.10. Es un conjunto de sentencias simples que se encierran entre los símbolos "{" y "}" para formar un bloque de código. Pueden aparecer en cualquier sitio en el que podría aparecer una sentencia simple.
2.5.11. Un dato sin valor (void en lenguaje C) es un dato que no puede tomar por valor ningún valor, es decir, es un dato vacío (nulo).

2.5.12. La instrucción de salto return es utilizada en lenguaje C para indicar el valor de retorno de una función. Por tanto, vamos a hacer uso de la instrucción return cuando definamos subprogramas de tipo función, que estudiaremos en el apartado 5. Llamadas a subprogramas.

2.6 Reactivos para escribir la salida de código

2.6.1.

suma: 5.500000e+000
multiplicacion: 7.500000e+000

2.6.2.

Introduzca dos numeros:
53
22

El mayor de los dos es:53
Oprime Enter para terminar

2.6.3.

Primero: x = 12, y = 12
Segundo: x = 11, y = 12
Tercero: x = 10, y = 12
Cuarto: x = 9, y = 12

2.7 Reactivos de problemas para programar

Ver nota.

UNIDAD TEMÁTICA III

3.1 Reactivos de opción múltiple

3.1.1. d)
3.1.2. b)
3.1.3. a)
3.1.4. c)
3.1.5. c)

3.2 Reactivos de falso / verdadero

3.2.1. F
3.2.2. V
3.2.3. F
3.2.4. F
3.2.5. V
3.2.6. F
3.2.7. V
3.2.8. F

3.3 Reactivos de relación de columnas

3.3.1. (b)
3.3.2. (f)
3.3.3. (g)
3.3.4. (a)
3.3.5. (d)
3.3.6. (c)
3.3.7. (e)
3.3.8. (h)

3.4 Reactivos para completar la sentencia

3.4.1. estructuras de control
3.4.2. selección, iteración
3.4.3. If/else, switch
3.4.4. ciclo, bucle
3.4.5. do-while
3.4.6. If
3.4.7. If/else
3.4.8. externo

3.5 Reactivos de preguntas abiertas (conceptos)

3.5.1. Establece que toda función computable puede ser implementada en un lenguaje de programación que combine sólo tres estructuras lógicas. Esas tres formas (también llamadas estructuras de control) específicamente son:
1. Secuencia
2. Selección
3. Iteración
3.5.2. En lenguajes de programación se pueden usar sangrías o llaves para indentar, se utiliza para delimitar bloques lógicos de código y hacer el código más legible.
3.5.3. Las computadoras están diseñadas especialmente para aquellas aplicaciones en las cuales una operación o conjunto de ellas deben de repetirse varias veces. A las estructuras que repiten una secuencia de instrucciones un número determinado de veces se les denomina bucles y se llama iteración al acto de repetir la ejecución de una secuencia de acciones.
3.5.4. If, if/else y switch.
3.5.5. While, do/while y for.
3.5.6.

if (condición)
(acción)
3.5.7.
if (condición)
(acción$_1$)
else
(acción$_2$)
3.5.8.
for (inicialización;condición;actualización)
{
(acción)
}
3.5.9.
while
(condición)
{
(acción)
}
3.5.10.
(Inicialización)
do
{
(acción)
(actualización)
}while (condición);
3.5.11. Los procesos repetitivos requieren contar los sucesos y acciones internas, una forma de hacerlo es mediante un contador. Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante en cada repetición.
3.5.12. Un acumulador o totalizador es una variable cuya función es almacenar cantidades resultantes de operaciones sucesivas. Realiza la misma función que un contador con la diferencia de que el incremento o decremento es variable en lugar de constante.
3.5.13. La estructura switch evalúa una expresión que puede tomar n valores distintos, según con cual de estos valores coincida, se ejecutaran ciertas acciones, es decir, el programa o algoritmo seguirá un determinado camino entre los n posibles.
3.5.14.

```
switch (expresión entera) {

    case exp_constante_1:
            acciones a realizar cuando la expresión tiene el valor exp_constante_1;
    break;

    case exp_constante_2:
            acciones a realizar cuando la expresión tiene el valor exp_constante_2;
    break;

    ...especificar todos los casos

    default:
            acciones a realizar cuando la expresión no coincide con ninguno de los casos;
    break;

    }
```

3.5.15. La instrucción de salto break se usa para interrumpir (romper) la ejecución normal de un bucle, es decir, la instrucción break finaliza (termina) la ejecución de un bucle y, por tanto, el control del programa se transfiere (salta) a la primera instrucción después del bucle.

3.5.16. La instrucción de salto continue siempre se usa para interrumpir (romper) la ejecución normal de un bucle. Sin embargo, el control del programa no se transfiere a la primera instrucción después del bucle (como sí hace la instrucción break), es decir, el bucle no finaliza, sino que, finaliza la iteración en curso, transfiriéndose el control del programa a la condición de salida del bucle, para decidir si se debe realizar una nueva iteración o no. Por tanto, la instrucción continue finaliza (termina) la ejecución de una iteración de un bucle, pero, no la ejecución del bucle en sí. De forma que, la instrucción continue salta (no ejecuta) las instrucciones que existan después de ella, en la iteración de un bucle.
3.5.17. Los procesos repetitivos requieren contar los sucesos y acciones internas, una forma de hacerlo es mediante un contador. Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante en cada repetición.
3.5.18. Un acumulador o totalizador es una variable cuya función es almacenar cantidades resultantes de operaciones sucesivas. Realiza la misma función que un contador con la diferencia de que el incremento o decremento es variable en lugar de constante.
3.5.19. 2
3.5.20. 6

3.6 Reactivos para escribir la salida de código

3.6.1.

0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

3.6.2.

Ingresa una calificación: 8
pasaste!! :D

3.6.3.

Ingrese la cantidad: 5
Numero 1: 7
Numero 2: 9
Numero 3: 3
Numero 4: 4
Numero 5: 8
El resultado es: 6.200000

3.6.4.

Introduce un numero:13

Solo hay 12 meses!!!
Solo hay 4 Trimestres!!!

Oprime una tecla para finalizar*

3.6.5.

```
 estoy dentro del ciclo 0
 estoy dentro del ciclo 1
 estoy dentro del ciclo 2
 estoy dentro del ciclo 3
 estoy dentro del ciclo 4
 estoy dentro del ciclo 5
 estoy dentro del ciclo 6
 estoy dentro del ciclo 7
 estoy dentro del ciclo 8
 estoy dentro del ciclo 9
ya estoy fuera del ciclo 10
Presione una tecla para continuar...
```

3.6.6.

```
introduzca numero del mes y el año:
2
2000

El mes 2 del año 2000 tiene 29 dias
```

3.6.7.

```
Ingrese un numero: 614
Ingrese otro numero: 123
614 es mayor que 123.
```

3.6.8

```
Escribe un numero entero
10
10.1
10.12
10.123
10.1234
10.12345
10.123456
10.1234567
10.12345678
10.123456789
10.12345678910
Presione una tecla para copntinuar...
```

3.6.9.

```
Escriba un numero:
79
LXXIX
```

3.6.10.

Introduce primer numero:4
Introduce segundo numero:13
13= 4*3 + 1
 4= 1*4 + 0

 El resultado es: 1

3.7 Reactivos de problemas para programar

Ver nota,

UNIDAD TEMÁTICA IV

4.1 Reactivos de opción múltiple

4.1.1. a)
4.1.2. b)
4.1.3. a)
4.1.4. a)
4.1.5. d)
4.1.6. c)
4.1.7. b)
4.1.8. c)
4.1.9. d)
4.1.10. b)
4.1.11. a)
4.1.12. c)
4.1.13. c)
4.1.14. c)

4.2 Reactivos de falso / verdadero

4.2.1. V
4.2.2. F
4.2.3. V
4.2.4. F
4.2.5. F
4.2.6. F
4.2.7. V
4.2.8. V
4.2.9. F
4.2.10. V
4.2.11. V
4.2.12. F
4.2.13. F
4.2.14. F
4.2.15. F

4.3 Reactivos de relación de columnas

4.3.1. (o)
4.3.2. (e)
4.3.3. (l)
4.3.4. (m)
4.3.5. (i)
4.3.6. (h)
4.3.7. (n)
4.3.8. (g)
4.3.9. (b)
4.3.10. (j)
4.3.11. (k)
4.3.12. (c)
4.3.13. (d)
4.3.14. (f)
4.3.15. (a)

4.4 Reactivos para completar la sentencia

4.4.1. 8 ó 1296
4.4.2. 5
4.4.3. puts
4.4.4. &, *
4.4.5. NULL
4.4.6. free ()
4.4.7. valor, referencia.
4.4.8. strcpy (char *s1, char *s2)
4.4.9. strcat (char *s1, char *s2)
4.4.10. toupper (char)

4.5 Reactivos de preguntas abiertas (conceptos)

4.5.1. Es una estructura compuesta por elementos del mismo tipo y almacenados consecutivamente en memoria.
4.5.2. El carácter nulo '\0'
4.5.3. gets
4.5.4. matriz, &matriz[0];
4.5.5. Es una variable que almacena una dirección física de memoria de otra variable del tipo especificado.
4.5.6. tipo *nombre;
4.5.7. Asignación, indirección, dirección, incremento, diferencia y comparación.
4.5.8. Permite asignar un bloque de memoria de n bytes consecutivos en memoria para almacenar datos.
4.5.9. Asigna un bloque de memoria para una matriz dado el número de elementos.
4.5.10. float *af;
af=(float *)malloc(10*sizeof(float));
4.5.11. Regresa un apuntador a NULL.
4.5.12. Es una colección de variables relacionadas que se referencia bajo un único nombre.
4.51.3. Agrupar una o más variables de diferentes tipos para hacer más fácil su manejo.

4.5.14. struct datos{
int x;
char c;
float f;
};
4.5.15. struct datos a;
4.5.16. Un ejemplo típico es el registro de una agenda
struct agenda{
char nombre [20];
char apellido [20];
int edad;
char calle [20];
int numero;
}datos;
4.5.17. Operador de miembro de estructura también conocido como operador de punto.
4.5.18. Una unión es un tipo de dato derivado, cuyos miembros comparten el mismo espacio de almacenamiento.
4.5.19. La unión solo se puede guardar un dato a la vez.
4.5.20. La memoria estática es memoria que se reserva en el momento de la compilación, antes de comenzar a ejecutarse el programa.
4.5.21. La memoria dinámica se hace en tiempo de ejecución, después de leer los datos y de conocer el tamaño exacto del problema a resolver.
4.5.22. void permutar(double* px, double* py) {
double temp;
temp = *px;
*px = *py;
*py = temp;
}
4.5.23. A diferencia de otros lenguajes de programación que emplean un tipo denominado cadena string para manipular un conjunto de simbolos, en C, se debe simular mediante un arreglo de caracteres, en donde la terminación de la cadena se debe indicar con nulo. Un nulo se especifica como '\0'. Por lo anterior, cuando se declare un arreglo de caracteres se debe considerar un caracter adicional a la cadena más larga que se vaya a guardar.
4.5.24. No se puede modificar el tamaño del vector o arreglo; desperdicio de memoria al ser estático, no se puede aprovechar al máximo.
4.5.25. Acceso directo hacia los elementos de la lista; almacenamiento continuo de los datos.
4.5.26. No es sencillo implementarlos; el acceso a las posiciones de memoria no es lineal; almacenamiento no continuo.
4.5.27. Facilita a la asignación de memoria y la liberación de la misma; mayor aprovechamiento de la memoria.
4.5.28. Es un archivo de la Biblioteca estándar del lenguaje de C que contiene la definición de macros, constantes, funciones y tipos de utilidad para trabajar concadenas de caracteres y algunas operaciones de manipulación de memoria.
4.5.29. Es un archivo de cabecera de la biblioteca estándar del lenguaje de C diseñado para operaciones básicas con caracteres.

4.6 Reactivos para escribir la salida de código

4.6.1.

```
        Numero = 2
 Numero 1
        Dato 1 = 5
        Dato 2 = 8
        Dato 3 = 1
        5.000000        8.000000        1.000000
        ~El resultado es 4.666667
 Numero 2
        Dato 1 = 9
        Dato 2 = 9
        Dato 3 = 0
        9.000000        9.000000        0.000000
        ~El resultado es 6.000000
                ***Oprime Enter para terminar***
```

4.6.2.

```
Introduzca el ancho de la matriz:
2
Introduzca el largo de la matriz:
2
Introduzca los valores de la matriz:
2
2
2
2
La matriz resultante es:
   2
 2
Presione una tecla para continuar . . .
```

4.6.3.

```
Introduzca el largo de la matriz:
2
Introduzca el ancho de la matriz:
2
Introduzca los valores de la matriz:
2
2
2
2

2       2
2       2
La matriz NO es triangular superior: 0
Presione una tecla para continuar . . .
```

4.6.4.

```
Introduzca un nombre:
Juan
Introduzca una unidad academica:
Estructuras de Datos
Introduzca la calificacion obtenida:
9.2
DATOS DEL ALUMNO
Nombre: Juan
Materia: Estructuras de datos
Calificacion: 9.200000
APROBADO
Presione una tecla para continuar . . .
```

4.6.5.

```
introduce el numero de filas de la matriz:
2
introduce el numero de columnas de la matriz:
2
introducir los valores de la matriz
m[0][0]= 2
m[0][1]= 2
m[1][0]= 2
m[1][1]= 2
el resultado es = 0=4
el resultado es = 1=4
```

4.6.6.

```
NOMBRE:
Ana
DIRECCION:
av. 522
TELEFONO:
12345678
```

FECHA DE NACIMIENTO:
Dia: 11
Mes: 09
Año: 1990

NOMBRE: Ana
DIRECCION: av. 522
TELEFONO: 12345678
FECHA DE NACIMIENTO:
Dia: 11
Mes: 9
Año: 1990

4.6.7.

introduce una linea de texto, pulsando enter despues de cada linea:
pulse ctrl+z para mostrar

El profesor se fue a comer
y ya mero termino esto =D
^Z
53 caracteres
12 palabras
2 lineas

4.6.8.

tamaño de la matriz
Ingrese numero de filas
3
Ingrese numero de columnas
3
ingresa la matriz
1
1
1
1
1
1
1
1
1
1 0 0
0 1 0
0 0 1

4.6.9.

Ingrese una cadena de texto: hola mundo como estamos todos por alla

El resultado es: 14

4.6.10.

resultado: 15
resultado: 13
b : 5
puntero1 : 2293572
*puntero1 : 5

Presione una tecla para continuar . . .

4.6.11.

Soy la vieja manera
ESCUELA SUPERIOR DE COMPUTO
Soy la vieja manera

Presione una tecla para continuar . . .

4.6.12.

Escoja un numero:

1. Opcion 1
2. Opcion2
3. Opcion3
4. Salir
2

2
Escoja un numero:

1. Opcion 1
2. Opcion2
3. Opcion3
4. Salir3

3
Escoja un numero:

1. Opcion 1
2. Opcion2
3. Opcion3
4. Salir4

4.6.13.

La cadena antes: La Escuela Superior de Computo

La cadena despues: LA ESCUELA SUPERIOR DE COMPUTO

4.6.14.

Escribe una cadena:

david

Ps:

P1 : d
P2 : da
P3 : dav
P4 : davi
P5 : david

Ss:

S1 : d
S2 : di
S3 : div
S4 : diva
S5 : divad

4.6.15.

Los valores en el Arreglo 1 por linea son:
454
Los calores en el Arreglo 2 por linea son:
42821880
Los valores en el Arreglo 3 por linea son: 42

4.6.16.

Cuantas filas en R: 2
Cuantas columnas en R: 2
Cuantas filas en S: 2
Cuantas columnas en S: 2

Llenando la Matriz R:
Ingresa la posicion [1][1]: 2
Ingresa la posicion [1][2]: 2
Ingresa la posicion [2][1]: 2
Ingresa la posicion [2][2]: 2

Llenando la Matriz S:
Ingresa la posicion [1][1]: 1
Ingresa la posicion [1][2]: 1
Ingresa la posicion [2][1]: 2
Ingresa la posicion [2][2]: 2

La Matriz R :
[2][2]
[2][2]

La Matriz S :
[1][1]
[2][2]

Resultado: [2][2][2][2][2][2][2][2]

4.7 Reactivos de problemas para programar

Ver nota.

UNIDAD TEMÁTICA V

5.1 Reactivos de opción múltiple

5.1.1. b)
5.1.2. a)
5.1.3. c)
5.1.4. a)
5.1.5. b)
5.1.6. a)
5.1.7. c)
5.1.8. b)

5.2 Reactivos de falso / verdadero

5.2.1. V
5.2.2. V
5.2.3. V
5.2.4. F
5.2.5. V
5.2.6. V
5.2.7. V
5.2.8. F
5.2.9. V
5.2.10. V

5.3 Reactivos de relación de columnas

5.3.1. (e)
5.3.2. (h)
5.3.3. (d)
5.3.4. (b)
5.3.5. (f)
5.3.6. (i)
5.3.7. (j)
5.3.8. (a)
5.3.9. (g)
5.3.310. (c)

5.4 Reactivos para completar la sentencia

5.4.1. fopen
5.4.2. stdio.h
5.4.3. stdio.h
5.4.4. el nombre del archivo, el modo de apertura
5.4.5. stdin, teclado
5.4.6. stdout, pantalla
5.4.7. stderr, pantalla
5.7.8. r+ ó rb+
5.4.9. 4
5.4.10. binario

5.5 Reactivos de preguntas abiertas (conceptos)

5.5.1. Es una abstracción del camino que siguen los datos desde alguna fuente (dispositivos de entrada o el programa) hasta un destino (el programa o dispositivos de salida). A la fuente se le llama productor, al destino se le llama consumidor. Esta abstracción es usada en varios lenguajes de programación, incluyendo el lenguaje C.
5.5.2. Se declara el apuntador para recibir lo que regresa fopen. Siempre y cuando el archivo se haya abierto
5.5.3. FILE * fopen(char nombreFichero [ ], char modoAcceso[ ])
5.5.4. NULL
5.5.5. r, w, a, r+, w+, a+
5.5.6. E/S formateada, E/S de carácter y E/S directa
5.5.7. E/S formateada: fprintf y fscanf
E/S de carácter: getc, fgets, putc, fputs.
E/S directa: fread y fwrite.
5.5.8. Escribe un bloque datos de memoria a un archivo en modo binario.
5.5.9. Lee un bloque de datos desde un archivo en modo binario hacia memoria.
5.5.10. Fin de archivo
5.5.11. feof () regresa 0 si no se ha llegado al final del archivo y si regresa cualquier valor diferente de 0 si ya llego al final.

5.5.12. Lee del archivo un carácter y lo almacena en una variable de tipo entero y almacena el contenido de una variable de tipo char esto se debe a que almacena el valor ascii de la variable leída en la variable carácter.
5.5.13. Permite agregar o modificar datos dentro del archivo siempre y cuando el archivo haya sido abierto para modificación y agregar datos al final.
5.5.14. Permite el movimiento dentro del archivo a partir de una referencia de posición.
5.5.15. fseek (puntero, tamaño_variable*nposiciones, posición_referencia)

5.6 Reactivos para escribir la salida de código

5.6.1.

Consola:
ingresa una cadena
anita la huerfanita

la cadena resultante es:
atinafreuh al atina
Archivo:
anita la huerfanitaatinafreuh al atina

5.6.2.

Consola:
Dame un numero: 789456213
Dame un nombre :david

Dame tu apellido: gomez

Dame la cantidad: 12003.63
Dame un numero: 0
Presione una tecla para continuar . . .
Archivo:
Todo es en binario, por lo que no es legible para una persona.

5.6.3.

Consola:
Dame tu Matricula: 2
Dame tu Nombre: david
Dame tu Apellido: gomez
Dame tu Curp: dhf201

Archivo:

2david gomez dhf201

5.6.4.

Consola:
100,1
100,0
100,1
100,0
100,1
100,0
100,1
100,0
100,1
100,0
100,1
100,0
100,1
100,0
100,1
100,0
100,1
100,0
100,1
100,0
100,1

Presione una tecla para continuar . . .

Archivo:
d d d d d d d d d d d d d d d d d d d d

5.6.5.

Archivo:
B C D E F G H I J K L M N O P Q R S T U V W X Y

5.6.6.

Consola:

Escribe un simbolo: ?

Presione una tecla para continuar . . .

Archivo:

??????????
??????????
??????????
??????????
??????????

5.7 Reactivos de problemas para programar

Ver nota.

Estadísticas

Unidad temática	Reactivos de opción múltiple	Reactivos de falso/ verdadero	Reactivos de relación de columnas	Reactivos para completar la sentencia	Reactivos de preguntas abiertas	Reactivos para escribir la salida de código	Reactivos de problemas para programar
I	10	12	10	12	14	3	12
II	8	10	10	10	12	3	5
II	5	8	8	8	20	10	10
IV	14	15	15	10	29	16	12
V	8	10	10	10	15	6	5
Subtotales	45	55	53	50	90	38	44
						Total	375