Hernández Torres Mario Ivan

Ingeniería Eléctrica

Electrónica

EDA I

Grupo 21

Actividad 1. Repaso de lo que aprendí en Fundamentos de programación

25 de febrero de 2021

Repaso de Fundamentos de Programación

Antes que nada, vimos el panorama general sobre la programación.

Primero vimos la evolución de la programación, que es programar, la programación en informática, esta se refiere a la elaboración de un programa de computadora para que esta realice una actividad deseada. Además, checamos el concepto de software, que son el conjunto de programas usados para ejecutar una computadora.

Conceptos importantes que se vieron al principio son: Lenguaje de programación, el lenguaje diseñado para indicar instrucciones a una máquina, particularmente a una computadora; el código fuente, que son las instrucciones escritas en algún lenguaje de programación y son almacenadas en un archivo de texto. Dentro del los lenguajes de programación existen de bajo nivel y de alto nivel.

Se vieron los beneficios de la programación, estos se presentan dentro de la mayoría de los campos de la sociedad (la industria, la educación, la salud, la investigación, el gobierno, el ambiente, la investigación, el comercio) de forma indirecta.

Dentro del primer tema además se realizaron algoritmos en la solución de problemas y sus retos. Se vio el concepto de algoritmo el cual es un conjunto de pasos, procedimientos o acciones que permiten alcanzar un resultado o resolver un problema. Las tres características principales de un algoritmo son: la precisión, el determinismo y la finitud. Los algoritmos están presentes en la vida diaria ya que día con día enfrentamos situaciones que hay que resolver, desde lo muy simple a lo muy complicado.

Aprender a programar nos permite encontrar soluciones mediante una forma de pensamiento: estructurada a partir de pasos básicos, flexible, abierta.

Dentro del segundo tema vimos la resolución de problemas.

Las etapas generales para la resolución de un problema son:

- Análisis profundo del problema
- Construcción del algoritmo
- Verificación del algoritmo

Vimos la resolución de problemas desde el enfoque sistemático. Esta propone que cualquier problema puede analizarse asociado al concepto de sistema.

Se checo lo que es la ingeniería de software, esta es una aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software. El objetivo es llevar un mejor control del desarrollo del softeare y tener mayor certeza de que cumplirá con las expectativas planteadas de todos los interesados. Los métodos de la ingeniería de software se refieren a las técnicas

para realizar las actividades del proceso de software, es decir, el como realizarlas: como obtener los requerimientos, como realizar el análisis de requerimientos, como construir el programa, como realizar las pruebas.

Dentro de la ingeniería de software esta el proceso de software. Este es una serie de pasos definidos de acuerdo al tipo de software a desarrollar. Abarca procesos de las siguientes categorías:

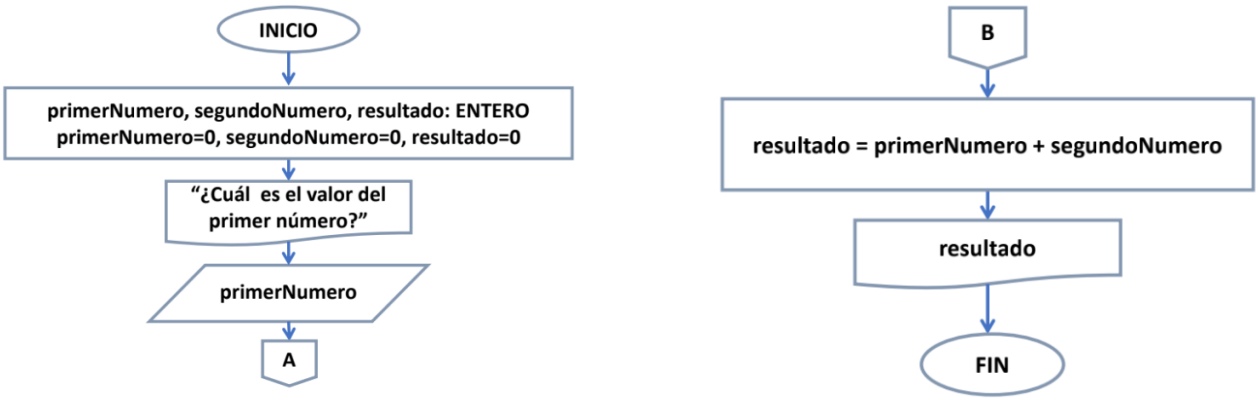
- Comunicación
- Planeación
- Modelado
- Construcción
- Despliegue

Se vieron los tipos de datos en un algoritmo de computación:

- Datos numéricos
- Datos de tipo carácter
- Datos de tipo cadena de caracteres
- Datos de tipo boolaneo

Se vieron los diagramas de flujo, que son representaciones graficas de un algoritmo. A partir de un algoritmo se puede escribir un programa en pseudocódigo o en algún lenguaje de programación.

Los pseudocodigos son la representación de los pasos indicados en un diagrama de flujo en instrucciones genéricas, similares a las de un lenguaje de programación.




```
INICIO
   primerNumero, segundoNumero, resultado: ENTERO;
   primerNumero=0, segundoNumero=0, resultado=0;
   IMPRIMIR: "¿ Cuál es el valor del primer número?";
   LEER: primerNumero;
   IMPRIMIR : "¿Cuál es el valor del segundo número?";
   LEER: segundoNumero;
   resultado = primerNumero + segundoNumero;
   IMPRIMIR: resultado;
FIN
```

Pseudocodigo.

INICIO

conSalsa, 3depapa: CADENA DE CARACTERES;

3depapa = “Si”;

conSalsa = “Si”;

IMPRIMIR: “¿3 de papa joven?”

LEER: 3depapa;

3depapa == “Si”;

Si 3depapa == “Si”, ENTONCES: Poner 3 tacos de papa;

SI 3depapa == “No”, ENTONCES: Poner 3 tacos de frijol;

IMPRIMIR: “¿Con salsa joven?”;

LEER: conSalsa;

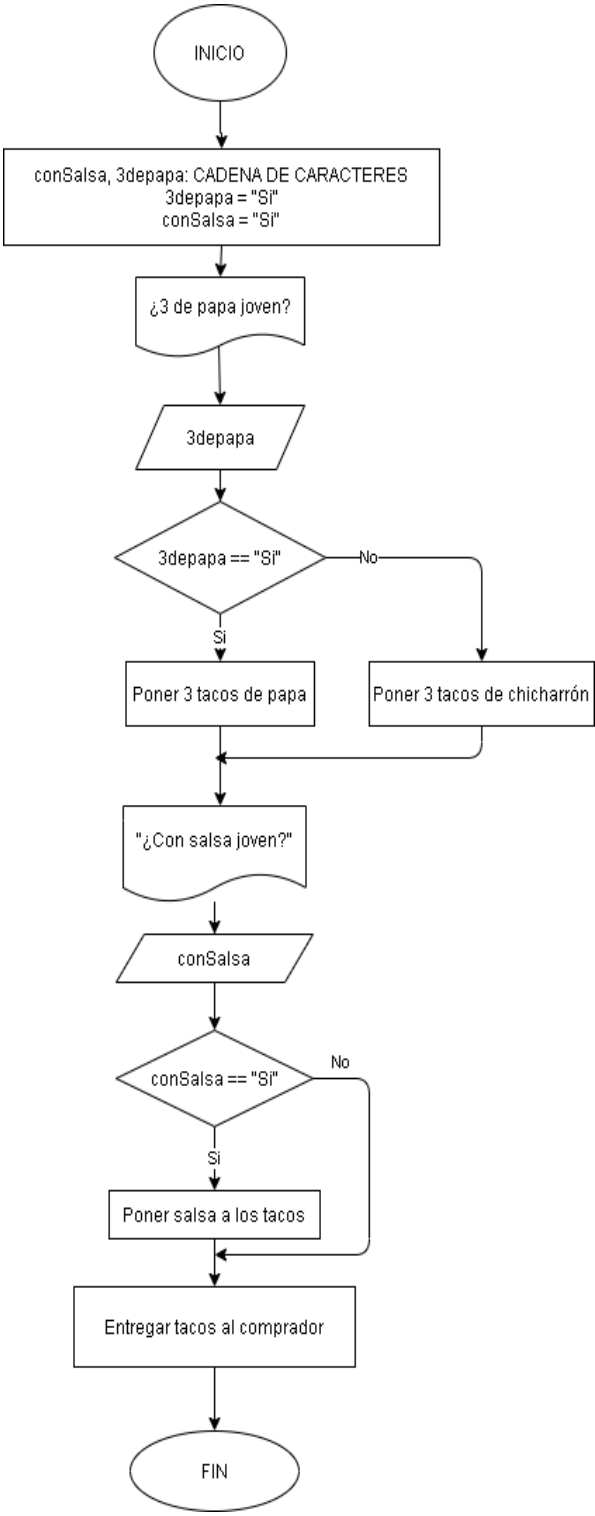
conSalsa==”Si”;

Si: conSalsa == “Si”, ENTONCES: poner salsa;

:conSalsa == “No”;

Entregar tacos al comprador

FIN



Se vio además todos los sistemas numéricos posicionales (base 10, base 2, base 8, base 16) y sus distintas conversiones.

Sistema de numeración octal

En el sistema de numeración octal:

- La posición menos significativa tiene un valor de 8^0

- La posición siguiente en significancia tiene un valor de 8^1

- La posición siguiente en significancia tiene un valor de 8^2

Decimal	Binario	Octal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
8	001 000	10
9	001 001	11
10	001 010	12
11	001 011	13
12	001 100	14
13	001 101	15
14	001 110	16
15	001 111	17
16	010 000	20

Decimal	Binario	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Decimal	Binario	Hexadecimal
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
19	0001 0011	13
20	0001 0100	14
21	0001 0101	15
22	0001 0110	16
23	0001 0111	17
24	0001 1000	18
25	0001 1001	19
26	0001 1010	1A
27	0001 1011	1B
28	0001 1100	1C
29	0001 1101	1D
30	0001 1110	1E
31	0001 1111	1F

Base 10	Complemento a 2 con 4 bits
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000

Base 10	Complemento a 2 con 4 bits
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001

En el tema 3 Fundamentos para la construcción de código a partir del algoritmo empezamos a trabajar con compiladores en lenguaje C. Vimos la sintaxis básica y semántica, las palabras reservadas en lenguaje C, la estructura general de un programa en lenguaje C las cuales son las siguientes:

- **Directivas del preprocesador**. Son las instrucciones al compilador antes de que se comple el programa principal. Las más utilizadas son #include y #define.
- **Declaraciones globales.** Indican al compilador que las funciones definidas por el usuario o
  variables así declaradas son comunes a todas las funciones de su
  programa. Estas se sitúan antes de la función main (). Pueden incluir la declaración de variables globales y de prototipos de las funciones en ese código fuente.
- **Función principal.** La cual es main (). Es la primera función que se ejecuta en un programa y solo debe existir un solo main en todo el programa.
- **Otras funciones.** Las funciones definidas por el usuario se invocan por su nombre y los parámetros que puedan tener. Cuando la función es llamada, el código asociado con la función se ejecuta, cuando termina de ejecutarse, el control se regresa a la función desde donde fue llamada para ejecutar las instrucciones siguientes. Las funciones requieren la declaración de prototipo en la sección de declaraciones globales. Estas declaraciones indican al compilador la forma por la que la función que será invocada en el programa.

Dentro de los tipos de datos del lenguaje c existen distintas formas de declarar una variable. Estas son de tipo char(8 bits), int(16 bits),, float(32 bits),, double(64 bits). Char se refiere a datos de tipo character. Int se refiere a valores enteros. Float y double se refieren a valores reales.

Código	Descripción
%d	El dato se considera como entero decimal.
%o	El dato se considera como octal.
%X, %x	El dato se considera como hexadecimal.
%u	El dato entero se toma como entero sin signo.
%c	El dato se considera de tipo carácter.
%e	El dato se considera de tipo float. Se convierte a notación científica, de la forma (-}n.mmmmmmE(+I-}dd.
%f	El dato se considera de tipo float. Se convierte a notación decimal, con parte entera y los dígitos de precisión.
%s	El dato ha de ser una cadena de caracteres.
%lf	El dato se considera de tipo double.

Secuencia especial	Símbolo	Secuencia especial	Símbolo
\a	Alarma	\V	Tabulación vertical
\b	Retroceso de espacio	\ \	\
\f	Avance de página	\ ?	?
\n	Salto de línea	\ "	"
\r	Retorno de carro	\0	Cero, nulo
\t	Tabulación		

Además se vieron los distintos operadores aritméticos.

Operador	Operación	Ejemplo	Resultado
= =	Igual que	4 = = 4 "casa"= = "caza"	Verdadero Falso
!=	Diferente a	4 != 4 "casa" != "caza"	Falso Verdadero
<	Menor que	4 < 7	Verdadero
>	Mayor que	4 > 7	Falso
< =	Menor o igual que	5 < = 5 5.1 < = 5	Verdadero Falso
> =	Mayor o igual que	5 > = 5 5.1 > = 5	Verdadero Verdadero

Operador	Operación	Ejemplo	Resultado
&&	Se cumplan 2 o más condiciones indicadas al mismo tiempo	(5= =5)&&("Hola"= ="Adios")	Falso
||	Se cumpla una de dos condiciones indicadas	(5= =5) ||("Hola"= ="Adios")	Verdadero
!	Negación de sentencia indicada	!(5==5)	Falso

Vimos la sentencia switch, el ciclo while y do while

```
do
{
        …Bloque de instrucciones que se ejecutan mientras
        EXPRESIÓN sea VERDADERA
} while( expresión);
```

```
while( expresión)
{
        …Bloque de instrucciones que se ejecutan mientras
        EXPRESIÓN sea VERDADERA
}
```

Operadores de incremento y decremento.

Expresión	Acción
k = --i;	Resta a i una unidad y el resultado lo asigna a k
k = ++i;	Suma a i una unidad y el resultado lo asigna a k
k = i--;	Asigna el valor de i a k, y después decrementa el valor de i
k = i++;	Asigna el valor de i a k, y después incrementa el valor de i

El ciclo for.

```
for (VC=VI ; Condición de VC respecto a VF; VC=VC  + ID)
{
        …Bloque de instrucciones
}
```

Los arreglos que son una secuencia de datos del mismo tipo. Cada uno de estos datos se enumeran e identifican a través de un índice de valores enteros partiendo desde el 0.

Arreglos de dos dimensiones, se trata de un arreglo donde cada uno de los elementos es otro arreglo. Equivale a una matriz de 2 dimensiones.

```
for(indiceRenglon=0;indiceRenglon<m; indiceRenglon++)
{
        for(indiceColumna=0 ; indiceColumna < n ; indiceColumna++)
        {
                Operaciones con
                arregloBidimensional[indiceRenglon][indiceColumna];
        }
}
```

El manejo de archivos. Se utilizaba la funcion fopen para crear y abrir archivos. Existen distintos modos para abrir un archivo. Y la función fclose escribe el contenido y cierra el archivo previamente abierto con fopen.

Modo	Significado
r	Abre el archivo para lectura.
w	Crea un archivo para escritura. Sobreescribe el archivo si ya existe.
a	Abre el archivo para agregar contenido después de lo ultimo que contiene.
r+	Abre el archivo para lectura y escritura. No sobreescribe el archivo si ya existe.
w+	Crea un archivo para escritura y lectura. Sobreescribe el archivo si ya existe.
a+	Abre el archivo para agregar y leer contenido después de lo ultimo que contiene. Si no existe lo crea.

Existe también la funcion fprintf la cual escribe un valor con formato en un archivo se ocupa la siguiente sintaxis:

Fprintf(apuntadorAlArchivo, "cadena opcional e indicadores de formato", "argumentos");

La función fputc, escribe un carácter a la vez del archivo. Se utiliza la siguiente sintaxis.

fputc(caracter, apuntadorAlArchivo);

Funcion fgetc, lee un carácter desde un archivo. Se ocupa con la siguiente sintaxis.

fgetc(apuntadorAlArchivo);

La funcion feof sirve para determinar si se ha llegado al EOF. Tiene como valor de retorno un entero. Si el carácter leído es diferente de EOF regresa un valor de 0. Si el valor leído es EOF regresa un valor distinto de 0. A través de un ciclo recorre el contenido del archivo:

```
while ( feof(archivo) == 0) /*Mientras no se detecte el fin de archivo */
{
...
}
```

Los apuntadores Un apuntador es una variable que sirve para apuntar a otra variable. La manera en que una variable apuntador apunta a otra variable es almacenando la dirección de memoria donde reside ésta ultima. Para declarar un apuntador se usa el operador * antes del identificador de la variable. Para hacer referencia a la dirección de memoria de una variable se usa el operador &.

Paso de parámetros por valor. Cuando se invoca a una función y se le proporcionan argumentos, la función recibe una copia de los valores de los argumentos y los guarda en variables locales a ésta. Si se cambia el valor de una variable local en la función invocada, el cambio sólo afecta a la función y no tiene efecto en las variables que se le pasaron como argumentos.

Paso de parámetros por referencia. Se pasa a la función invocada las direcciones de memoria donde se almacenan los valores. La función invocada recibe las direcciones de memoria y las guarda en variables de tipo apuntador. Cuando en la función invocada se modifican los valores recibidos, éste valor queda almacenado en la misma dirección de memoria. Por lo tanto, los valores originales que se pasaron como argumentos se habrán modificado.