UNIVERSIDAD NACIONAL AUTONÓMA DE MÉXICO

FACULTAD DE INGENIERIA

Estructuras de Datos y Algoritmos I

Actividad #1 “Acordeón del lenguaje C y Linux”

Alumno: García Gallegos Luis

Grupo:15 **SEMESTRE 2021-2**

Fecha de entrega 01/03/2021

Práctica #1

Actualmente google nos facilita las búsquedas en internet con las herramientas del software que ofrece permitiendo realizar actividades, trabajos académicos de una manera más sencilla y organizada, tales como el manejo de repositorios de almacenamiento (google drive), buscadores con funciones avanzadas (site:).

Algunos de los comandos para realizar búsquedas avanzadas son:

- site: que restringe los resultados de la búsqueda a solo aquellos de un sitio en específico. Ejemplo: site: el universal
- “Termino de búsqueda”, realiza una búsqueda exacta, donde coincida con la palabra que buscamos. Por ejemplo: “Linux”
- OR, el buscador que devolverá resultados de un término que buscamos o ambos. Ejemplo: Dev c++ OR Dev
- -, excluye el término de la búsqueda, por ejemplo: lenguajes de programación – Linux, esto quiere decir que buscaremos lenguajes de programación, pero excluye de la búsqueda a Linux.
- Define: busca el significado de la palabra indicada, ejemplo: define: programación.
- +, para forzar una búsqueda que coincida con la palabra o frase exacta que se le indicó.

Otras de las funciones de google es realizar cálculos que le puedes indicar en la barra de búsquedas (5+5, 4-2, 5*3, 3/2), al igual que puede realizar conversiones, graficas en 2D y 3D.

Google académico te permite obtener de manera más sencilla la descripción, el link y el formato APA de páginas web o pdf’s confiables.

Github y google drive son repositorios que te permiten trabajar con varias personas al mismo tiempo en un mismo proyecto y tener un control sobre las modificaciones que se le hacen al mismo.

Práctica #2

Comandos básicos de GNU/Linux y la importancia de el sistema operativo.

Sistema operativo: es el que se encarga de cómo funciona la computadora.

Comandos básicos:

date: da la hora y fecha

(ctrl + d) 2 veces para cerrar el archivo

(Esc) :wq terminar y guardar el documento

./ (nombre del archivo sin extensión) para correr el programa

:q! forzar cierre

:w guardar el documento actual

cat > nombre del archivo: crea uno o sobre escribe el archivo

cat >> (archivo): para agregar al final del texto

cat y (archivo): para ver contenido

cd nombre de la carpeta: para entrar a una carpeta

cd.. : para salir de la carpeta

clear: limpia la pantalla

cp (archivo)(carpeta): para copiar

gcc (nombre con la extensión del archivo) –o nombre del archivo sin la extensión para compilar el archivo

i: para comenzar a escribir en el archivo

ls –l mostrar todos los archivos donde estos y te muestra la opción de escritura y lectura

ls -la muestra todos los archivos y aparte los archivos raíz, también la fecha de creación

ls –li muestra detalle de la fecha y hora de creación

ls: te muestra el conjunto de archivos y carpetas que se encuentran

mkdir “nombre de la carpeta (con espacios)”: para crear una nueva carpeta

mkdir nombre de la carpeta (sin espacios): para crear una nueva carpeta

mv (archivo)(carpeta): para mover

mv (archivo)(nuevo nombre): para renombrar

rm (archivo): para borrar archivo

rm –r (carpeta): borrar carpeta

vi archivo: crea o abre el archivo

ls *(extensión) para buscar una extensión

rm* elimina todos los archivos contenidos en el directorio lista

cp -r (directorio1) (directorio2), el directorio (directorio1) se copió al directorio (directorio2) junto con los archivos guardados en (directorio1).

Práctica #3

Un algoritmo debe ser preciso, indicar el orden de cada paso, definido, numero finitos de pasos, en las entradas se especifica todo lo que se necesita para el desarrollo del proceso que se va a realizar, en la salida lo que se va obtener de este proceso y las restricciones son las limitaciones que deben estar dentro de este proceso.

Práctica #4

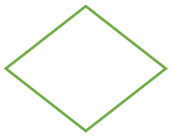
Los diagramas de flujo pueden representar una serie de acciones que comprendan un proceso que solucione un problema

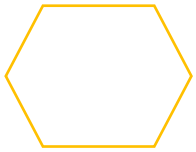
 Marca el inicio o el fin de un diagrama de flujo

 Representa un proceso, se expresan asignaciones, operaciones aritméticas, cambios de valor de celdas, etc.

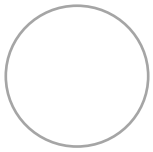
 Muestra la dirección del diagrama.

 Impresión de un resultado.

 Selección doble si/entonces

 Decisión múltiple

 Representa lectura de datos

 Conexión dentro de una página

 Conexión entre páginas diferentes

Práctica #5

Un ciclo es una secuencia de instrucciones o partes del código que se ejecutan repetidamente hasta que una condición deje de cumplirse.

En general, un ciclo estará compuesto de las siguientes partes:

- **Iteración de condición:** Es la condición que deben cumplirse para que nuestra secuencia de instrucciones se repita.
- **Iteración inicial:** Este será el valor inicial en el cual nuestro ciclo comenzará.
- **Iteración final:** Este será el valor final hasta el cual nuestro ciclo llegará. **Incremento de iteración:** Es la cantidad de unidades que avanzará nuestra iteración desde el valor de iteración inicial hasta el final.
- **Secuencia de instrucciones:** Son las instrucciones o bloques de código que se encuentran contenidos dentro de nuestro ciclo y se repetirá hasta que la condición de iteración no se cumpla.

Encabezado de un ciclo: Estas son las iteraciones que determinan cuántas veces se repite nuestro ciclo.

Cuerpo de un ciclo: Es el código que indica las instrucciones que se repetirá, la secuencia de instrucciones.

Práctica #6

Un compilador es un programa que se podría decir que es como el intermediario entre dos lenguajes donde al primer lenguaje que normalmente es un lenguaje de programación lo transforma a otro que puede ser texto o lenguaje máquina, también checa si hay errores en el código del primero.

Pasos para desarrollar un programa:

a. Definir el problema. Es lo que nos pide el problema, hay que tener en cuenta que es exactamente lo que nos pide para desarrollar.

b. Análisis del problema. Ya que sabemos que debemos hacer, debemos ver qué datos tenemos como entrada, salida y si hay restricciones.

c. Diseño de algoritmo. Comenzamos a hacer el algoritmo, para ello debemos tener en cuenta que debe de ser preciso, que pueda responder a variantes del usuario, que sea finito en extensión y tiempo.

d. codificación. En esta parte escribimos el algoritmo en algún lenguaje para que la computadora lo pueda leer. Lo que se codifica será conocido como archivo fuente.

e. prueba y depuración. la depuración nos puede ayudar a conocer errores en el programa, para una vez que han sido identificados se puedan corregir, la prueba es correr el programa, probándolo en varias situaciones, con distintos datos hasta que no presente errores.

f. documentación. la documentación son esquemas que se utilizan para guía de apoyo para comprender futuras modificaciones.

g. mantenimiento. se realiza después de haber hecho todo el proceso, por lo general se hace cuando se necesita hacer algún cambio. para esto es importante que el programa esté bien documentado.

Cuando se compila un programa se produce un archivo ejecutable y un archivo fuente.

Práctica #7

Comandos básicos del lenguaje C

Printf- para imprimir texto en la consola

Scanf- lee datos para después utilizarlos

Las variables se pueden definir como:

float- números decimales y enteros

int- números enteros

double- es igual que el float solo que la capacidad es mucho mayor

char- solo caracteres

Para que realice una operación primero se pone la variable que obtendrá el resultado, después igual y al final las operaciones, ejemplo x=2x+3

También se puede predefinir una variable con un valor así: (tipo de número variable=valor)

Práctica #8

if-else permite que el diagrama de flujo y el pseudocódigo siga un camino especifico si se cumple una condición o conjunto de condiciones. La estructura en Dev C++ es:

```
if (condiciones) {

//bloque de instrucciones}

else{

//bloque de instrucciones}
```

Que de igual manera servirá con solo if sin embargo si no se cumple la condición no hará nada.

switch permite que el flujo del diagrama y el pseudocódigo se bifurque por varias ramas en el punto de la toma de decisiones, esto en función del valor que tome el selector. La estructura en Dev C++ es:

```
switch(variable){

case valor1: //bloque de instrucciones

break;

case valor2: //bloque de instrucciones

break;

case valor3: //bloque de instrucciones
```

break;

case valor n: //bloque de instrucciones

break;

default: //bloque de instrucciones

Práctica #9

La directiva del define (#define (palabra o letra) (palabra original)) se utiliza para que una algún carácter o palabra se vuelve una palabra reservada para ayudar ahorrar líneas de código o para reducir texto en el pseudocódigo, por ejemplo: #define s scanf

Do-while es una estructura de control cíclica, los cuales permiten ejecutar una o varias líneas de código de forma repetitiva sin necesidad de tener un valor inicial.

La estructura en Dev C++ es:

do{

//Bloque de instrucciones

}while(condición)

Práctica #10

La depuración sirve mucho ya que en proyectos grandes de programación si algo no está funcionando de manera correcta te podrá ayudar a identificarlo de manera más sencilla que si vas revisando línea por línea, para realizar este proceso se debe crear un archivo proyecto en Dev C++ para después marcar el punto de inicio o quiebre cliqueando sobre el número de la línea donde probablemente pueda iniciar el error, por consiguiente, añadiremos unos watch a algunas variables para ir checando como se desarrollan cada una a lo largo del programa e identificar donde está el problema, por último empezaremos con la depuración, para que después podamos corregir el pseudocódigo.

Práctica #11

Los arreglos ayudan a agrupar datos del mismo tipo para su posterior uso, para declarar un arreglo es necesario escribir el tipo de dato que se almacenara en el arreglo, luego se escribe el nombre del arreglo, y al final se escribe entre corchetes el número de datos a tratar. Ejemplo, float A[10] es un arreglo unidimensional y float A[10][10] es un arreglo bidimensional.

NOTA: los arreglos siempre empiezan desde la celda 0.

Para escribir datos dentro de un arreglo unidimensional se pueden escribir cuando se declara el arreglo, de esta manera float A[10]{1,2,3,4,5,6,7,8,9,10}; o recorrer el arreglo cuando no sabes exactamente qué datos vas a utilizar, de la siguiente manera:

```
for(i=0; i<10; i++){
        printf("\nEscriba el valor %d", i+1);
        scanf("%f", &A[i]);
}
```

Y para un arreglo bidimensional de la misma forma se puede escribir al mismo tiempo que se declara, ejemplo:

```
A[2][10]{1,2,3,4,5,6,7,8,9,10
          1,2,3,4,5,6,7,8,9,10};
```

o se puede recorrer el arreglo de la siguiente manera:

```
for(i=0; i<2; i++){
   for(j=0; j<10; i++){
        printf("\nEscriba el valor %d", i+1, j+1);
```

```
        scanf("%f", &A[i][j]);
    }
}
```

Práctica #12

Las funciones permiten automatizar tareas repetitivas, son un conjunto de procedimientos en capsulado en un bloque, cuyos valores son enviados para efectuar operaciones y posteriormente retornar un valor, esto permite que se reduzcan muchas líneas de códigos ya que puede ser llamada varias veces para no tener que escribirla tantas veces como se necesite, para declarar una función es necesario hacer un prototipo de la función. Ejemplo:

(el tipo de valor que regresara) (nombre de la función) ((tipo de valor que se enviara a la función));

char letra(int);

a continuación, se la mandara llamar a la función de esta manera:

(variable)= (nombre de la función) (variable que se le enviara);

después se deberá escribir la función y poner el bloque de acciones que realizará con los valores obtenidos para al final poder regresar un dato. Ejemplo:

(Tipo de dato que regresara) (nombre de la función) ((tipo de dato que regresara) (variable) {

//Bloque de acciones

return (variable que regresara);

}

Práctica #13

Un archivo es un elementó que te permite el almacenamiento de datos para su posterior uso.

NOTA: es muy importante abrir y cerrar un archivo

Tipos de texto binarios y de texto.

Modos de apertura:

r-abre un texto para lectura

w-crea un archivo de texto para escritura

a-abre un archivo de texto para añadir

Funciones para manipular un archivo:

fopen()-abre un archivo

fclose()-cierra un archivo

fgets()- lee una cadena de archivo

fputs()-escribe una cadena de un archivo

fprintf()- escribe una salida con formato en el archivo

fscanf()- lee una entrada con formato desde el archivo

ferror()- devuelve cierto si se produce un error

Para trabajar con archivos se utiliza la siguiente estructura:

```
FILE *(Apuntador);
if(((Apuntador)=fopen("(nombre del archivo).txt","(modo de apertura)"))==NULL){
        p("\n\n\t ERROR: El archivo no se pudo abrir. \n");
        return 1;}
```

else{

//Bloque de acciones

}

fclose(Apun_a_archivo);

Lua:

Lua es un lenguaje de programación extensible diseñado para una programación procedimental general con utilidades para la descripción de datos.

Palabras reservadas:

- and
- break
- do
- else
- elseif
- end
- false
- for
- function
- if
- in
- local
- nil
- not
- or
- rep
- eat
- return
- then

- true
- until
- while

Todos los valores en Lua son valores de primera clase. Esto significa que todos ellos pueden ser almacenados en variables

Existen ocho tipos básicos en lua: nil, boolean, number, string, function, userdata, thread y table.

- nil- cuya principal propiedad es ser diferente de cualquier otro valor
- boolean- es el tipo de los valores false (falso) y true (verdadero).
- number- reales
- string- tira de caracteres (lua trabaja con 8 bits)
- userdata- permitir guardar en variables de lua datos arbitrarios en c
- thread- representa procesos de ejecución y es usado para implementar co-rutinas
- table- implementa arrays asociativos, esto es, *arrays* que pueden ser indexados no sólo con números, sino también con cualquier valor (excepto **nil**)

Lua puede convertir automáticamente entre valores *string* y valores numéricos en tiempo de ejecución.

Existen tres tipos de variables en Lua: globales, locales y campos de tabla.

Clase de caracteres:

- %a: representa cualquier letra.
- %c: representa cualquier carácter de control.
- %d: representa cualquier dígito.
- %l: representa cualquier letra minúscula.
- %p: representa cualquier carácter de puntuación.
- %s: representa cualquier carácter de espacio.
- %u: representa cualquier letra mayúscula.

- %w: representa cualquier carácter alfanumérico.
- %x: representa cualquier dígito hexadecimal.
- %z: representa el carácter con valor interno 0 (cero).

Bibliotecas estándar:

- biblioteca básica;
- biblioteca de empaquetado;
- manejo de strings;
- manejo de tablas;
- funciones matemáticas (sin, log, etc.);
- entrada y salida (I/O);
- interacción con el sistema operativo;
- utilidades de depuración.

Las estructuras de control if, while y repeat tienen el significado habitual y la sintaxis familiar:

sentencia ::= while exp do bloque end

sentencia ::= repeat bloque until exp

sentencia ::= if exp then bloque {elseif exp then bloque} [else bloque] end

break finaliza los bucles más internos que esten activos.

La forma numérica del bucle for repite un bloque mientras una variable de control sigue una progresión aritmética. Tiene la sintaxis siguiente:

sentencia ::= for nombre '=' exp1 ',' exp2 [',' exp3] do bloque end

La sentencia for genérica trabaja con funciones, denominadas iteradores. En cada iteración se invoca a la función iterador que produce un nuevo valor, parándose la iteración cuando el nuevo valor es nil. El bucle for genérico tiene la siguiente sintaxis:

sentencia ::= for lista_de_nombres in explist do bloque end

lista_de_nombres ::= nombre {',' nombre}

Bibliografía:

- Cairó Osvaldo. (2005). Metodología de la programación. México: Alfaomega.
- Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes. (2008). Manual de Referencia de Lua. 27/02/2021, de Lua Sitio web: https://www.lua.org/manual/5.1/es/manual.html