UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERIA

ESTRUCTURA DE DATOS Y ALGORITMOS I

ACTIVIDAD ASÍNCRONA #1

ALUMNA: RIOS HERRERA ELISA DANIELA

MIÉRCOLES 24 DE FEBRERO DE 2021

Fundamentos de Programación

En primera instancia de la asignatura se explicó que un problema es una cuestión que se plantea para hallar resultados a partir de un conjunto de datos conocidos, mientras que una estrategia es un conjunto de acciones de manera muy pensada o encaminada a un fin determinado. Para resolver tales conceptos se necesita realizar un análisis el cual es un examen detallado de algo para conocer sus características o cualidades y extraer conclusiones.

La composición del análisis son los **datos de entrada, restricciones y datos de salida.** En los datos de entrada se identifica cuantos datos se requieren para resolver el problema, de que tipo son (enteros, reales o alfanuméricos) y no numéricos (tangibles o intangibles) y se le debe asignar un identificador a cada uno para poder manipularlos. En las restricciones se identifica que valores no se pueden utilizar y que unidades de medida se deben emplear, todo esto con base en los datos de entrada. Por último, en los datos de salida se identifica cuantos datos se obtendrán al final, de que tipo y también se les asigna un identificador.

Existen algunas reglas para nombrar correctamente a los identificadores, algunas de ellas es que siempre deben empezar con una letra, a esta letra pueden suceder otras letras o números, no se deben emplear espacios ni caracteres especiales, y se nombran de acuerdo al uso que se les dará.

Es importante tener en cuenta los pasos para entender un problema, son los siguientes:

1. Entender el problema
2. Analizar el problema

3. Diseñar y desarrollar la solución más eficiente

4. Probar la solución

5. Implementar la solución y probarla

En la **caja negra** se desconoce el proceso, mientras que en la **caja blanca** si se conoce el proceso, diseño y se realizan pruebas de escritorio.

La **ingeniería en software** es la aplicación de un enfoque sistemático para desarrollar, operar y mantener el software, este software tiene un ciclo de vida el cual es un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto, y se define con base en la norma ISO 12207.

Su cronología es la siguiente: Definición de necesidades > análisis > diseño > codificación > pruebas > validación > mantenimiento y evolución.

La **programación** es el proceso de diseñar, codificar, depurar y mantener el código fuente.

Un **algoritmo** es un método para resolver un problema, sigue un conjunto de pasos ordenados y es finito, preciso y definido. Existen dos tipos de codificación algorítmica, la forma grafica que es a través de **diagramas de flujo** y la forma no gráfica que es en forma de texto a través de un **pseudocódigo**.

Un **diagrama de flujo** es la representación gráfica del algoritmo, por lo que emplea símbolos estandarizados para representar acciones como entrada, salida, conectores en la misma página o en diferente, subproceso o función y condición múltiple. Para diseñar un diagrama de flujo también hay que seguir algunas reglas.

Los temas que están a continuación corresponden al lenguaje de programación Lenguaje C.

Dentro del **lenguaje C** existen dos tipos de declaraciones: **globales y locales**. Las declaraciones globales son para variables o funciones que deben reconocerse en todo el programa y se colocan antes del main(), las declaraciones locales son identificadores que solo se reconocen en la función principal.

Las **instrucciones de selección** permiten que el control de programa se transfiera a un determinado punto de ejecución y esto depende si cierta condición empleada es verdadera o no.

En la sentencia de control **If-else** la condición es simple y es la expresión que será evaluada, si esta condición es verdadera se ejecuta la instrucción, sin embargo, si es falsa solo se ignora y el programa continúa con la siguiente instrucción. Se cumple cuando se ejecuta if y no se cumple cuando se ejecuta else. Si al evaluar las condiciones no hay sentencias en else, pueden omitirse las llaves al igual que cuando if o else cuentan con una sola sentencia. La **expresión condicional** es la alternativa al if-else, si al probar la expresión es diferente de 0 entonces se evalúa la primera expresión, si resulta 0 se evalúa la segunda expresión.

La sentencia **Switch** sirve para crear varias condiciones de igualdad, existen múltiples alternativas como posibles opciones pero solo una se realiza. Se puede probar un identificador de variable tipo entero o alfanumérico carácter pero nunca en un real ni un alfanumérico cadena.

Las **instrucciones de repetición** (iteración, bucles o ciclos) repiten un bloque de instrucciones mientras se conserva una condición verdadera, se requiere algún ciclo while, do-while o for, una expresión que establece condición y se coloca antes de ser evaluada.

Existen dos tipos de repeticiones: controlada por contador o definida la cual se emplea antes de que inicie el ciclo y se conoce el numero de repeticiones, la segunda es la controlada por centinela o indefinida, pues el numero de iteraciones se desconoce.

El **ciclo for o bucle for** repite el conjunto de instrucciones que se encuentran dentro de sus llaves y ejecuta la secuencia de instrucciones un numero determinado de veces. El ciclo for aninado contiene otro ciclo for, que este a su vez contiene otro y así sucesivamente, se emplea cuando se requiere manejar variaciones rápidas y lentas.

En el **ciclo while** se ejecuta una secuencia un número de veces indeterminado mientras la condición o condiciones sean verdaderas, este repite el conjunto de instrucciones que se encuentra entre las llaves y para ingresar la condición siempre debe ser verdadera, de lo contrario el ciclo se rompe. Es posible que en while no haya condición, sino un identificador que se esté evaluando, en este caso si el identificador es 0 la condición es falsa y no se ejecuta el while.

El **ciclo do-while** repite instrucciones tantas veces se desee, estas instrucciones que se encuentran dentro de las llaves se realizan al menos una vez y no dependen de si se cumple una función o no.

Es posible sustituir un ciclo for por un ciclo while o do-while, pero jamás al revés.

Los **arreglos** en lenguaje C son una secuencia de datos del mismo tipo que ocupan un lugar continuo en la memoria, las posiciones son consecutivas y se denominan elementos, empezando a numerarse desde el 0. Existen dos tipos: estáticos y dinámicos. Los arreglos estáticos se crean al momento de programar, existen y se indica su tamaño y número de elementos y no cambian. Los arreglos dinámicos se crean al ejecutarse el programa, se indica el tipo y el usuario debe definir el tamaño y se pueden cambiar, pero no siempre existen. Si un arreglo es unidimensional quiere decir que tiene 1 renglón y m columnas, mientras que si es multidimensional tiene n renglones y m columnas.

Para los arreglos estáticos, si son unidimensionales numéricos contienen una serie de valores de números de forma consecutiva en la memoria, se accede con un índice o posición y deben declararse antes de usarse. Para almacenar varios valores en un arreglo estos se deben ingresar uno por uno para asignar la celda, se emplean repeticiones como en el ciclo for.

Si son unidimensionales alfanuméricos se debe indicar el numero de elementos del arreglo y sumarle un elemento que indicaría en fin de la cadena.

Los **apuntadores, punteros o pointers** tienen un espacio específico en la dirección de memoria para una variable declarada, la cual apunta a otra posición en la memoria donde se almacenan datos, debe declararse antes de usar, no almacena datos enteros y nunca se representa en el algoritmo. A través de ellos se puede acceder de forma más rápida a la información almacenada. Dentro de las aplicaciones más útiles de los apuntadores son las que tienen que ver con los arreglos.

El acceso de un arreglo puede suceder a través de un índice de localidad del mismo y otra forma es recorrer un arreglo mediante el apuntador.

Para declarar un apuntador, primero se debe definir el tipo de dato y el nombre de la variable precedida de un asterisco, por ejemplo: TipoDeDato *apuntador, variable;

Debemos saber y recordar siempre que los apuntadores solo deben apuntar a variables del mismo tipo de dato con el que fueron declarados. Para poder asignarle un valor a nuestro apuntador, se debe acceder a la localidad de memoria de la variable haciendo uso de un ampersand (&), de la siguiente forma: apuntador = &variable;

Los **archivos, files o ficheros** en lenguaje C son una parte de almacenamiento de la memoria que representa un conjunto de caracteres o bytes, se emplean como entrada (lectura) o salida (escritura). Para abrir un archivo, ya sea para lectura o escritura se utiliza fopen, si el nombreArchivo tiene extensión se debe incluir, y si no tiene entonces se agrega, es importante que el archivo se encuentre en la misma ubicación que el código fuente.

Se utilizan los siguientes comandos para: **Lectura** (fgetc para leer un carácter, fgets (para leer palabras o frases), fscanf (leer cualquier tipo de dato numérico o alfnum) **Escritura**: fputc (para escribir un carácter), fputs (para escribir frases y palabras), fprintf (escribir cualquier tipo de dato numérico o alfnum). Se debe cerrar el archivo al termino de usarse y se utiliza fopen.