Universidad Nacional Autónoma de México
Facultad de Ingeniería

Asignatura: Estructura de Datos y Algoritmos I

Actividad 1: Repaso de Lenguaje C

Alumno: Miranda González José Francisco

Fecha: Lunes 7 de Junio del 2021

Repaso Lenguaje C

Conocimos y usamos los ambientes y herramientas para el desarrollo y ejecución de programas en Lenguaje C, como editores y compiladores en diversos sistemas operativos.

Durante el curso de Fundamentos de Programación revisamos conceptos muy importantes para poder realizar diferentes programas, a continuación, resumiré los aspectos más importantes de cada tema visto en esa materia.

Comentarios:

El comentario por línea inicia cuando se insertan los símbolos '//' y termina con el salto de línea (hasta donde termine el renglón). El comentario por bloque inicia cuando se insertan los símbolos '/*' y termina cuando se encuentran los símbolos '*/'.

Bibliotecas:

Al momento de iniciar el programa se deben agregar todas las bibliotecas que se van a utilizar en el mismo, es decir, funciones externas necesarias para ejecutar el programa. En lenguaje C la biblioteca estándar de entrada y salida está definida en 'stdio.h' (standard in out).

Variables:

Declaramos el tipo de dato que esta puede contener (Caracteres, Enteros, Flotantes, Dobles) al igual que el identificador (nombre o etiqueta) con el que se va a manejar el valor.

El especificador de dato se usa para guardar o imprimir el valor de una variable.

Entero: %d, %i, %ld, %li, %o, %x , Flotante: %f, %lf, %e, %g, Carácter: %c, %d, %i, %o, %x, Cadena de caracteres: %s

Funciones:

printf es una función para imprimir con formato, es decir, se tiene que especificar entre comillas el tipo de dato que se desea imprimir.

scanf, la cual es una función que sirve para leer datos de la entrada estándar (teclado), para ello únicamente se especifica el tipo de dato que se desea leer entre comillas y en qué variable se quiere almacenar. Al nombre de la variable le antecede un ampersand (&), esto indica que el dato recibido se guardará en la localidad de memoria asignada a esa variable.

Teclas de escape:

\a: carácter de alarma, \b: retroceso, \f: avance de hoja, \n: salto de línea, \r: regreso de carro, \t : tabulador horizontal, \v: tabulador vertical, '\0': carácter nulo.

Para imprimir con formato nuestros códigos.

Tipos de operadores:

Operadores aritméticos: +, -, *, /, %

Operadores lógicos a nivel de bits: >>, <<, &, |, ~

Operadores de relación permiten comparar elementos numéricos, alfanuméricos, constantes o variables: ==, ¡=, <, >, <=, >=

Operadores lógicos permiten formular condiciones complejas a partir de condiciones simples: ¡, &&, ||

Estructuras de selección:

if, if-else y switch. Las estructuras de selección (o condicionales) nos permiten realizar una u otra acción con base en una expresión lógica.

Tres estructuras de repetición:

while, do-while y for. Las cuales nos permiten ejecutar un conjunto de instrucciones de manera repetida (o cíclica) mientras que la expresión lógica a evaluar se cumpla (sea verdadera).

Depuración:

Sirve para revisar de manera precisa el flujo de ejecución de un programa y el valor de las variables.

Depurar es muy importante pues nos ayuda a optimizar el programa, ver si este tiene algún fallo o error de ejecución.

Arreglos unidimensionales y multidimensionales:

Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse. A cada elemento (dato) del arreglo se le asocia una posición particular, el cual se requiere indicar para acceder a un elemento en específico. Esto se logra a través del uso de índices.

Para los unidimensionales, aprendimos la forma en la que estos se almacenan y la sintaxis para definirlo: tipoDeDato nombre[tamaño].

Para los arreglos multidimensionales aprendimos la sintaxis para definirlos: tipoDato nombre [tamaño][tamaño]…[tamaño];

Apuntadores:

Estos son una variable que contiene la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable.

La sintaxis para declarar un apuntador y para asignarle la dirección de memoria de otra variable es la siguiente:

TipoDeDato *apuntador, variable;

apuntador = &variable;

Los apuntadores solo pueden apuntar a direcciones de memoria del mismo tipo de dato con el que fueron declarados; para acceder al contenido de dicha dirección, a la variable apuntador se le antepone *.

Funciones:

La sintaxis básica para definir a una función es:

valorRetorno nombre (parámetros){

// bloque de código de la función

}

Una función puede recibir parámetros de entrada, los cuales son datos de entrada con los que trabajará la función, dichos parámetros se deben definir dentro de los paréntesis de la función, separados por comas e indicando su tipo de dato.

El valor de retorno de una función indica el tipo de dato que va a regresar la función al terminar el bloque de código de la misma. El valor de retorno puede ser de tipo entero, real, carácter o arreglo. Aunque también se puede regresar el elemento vacío (void).

La firma de una función está compuesta por tres elementos: el nombre de la función, los parámetros que recibe la función y el valor de retorno de la función; finaliza con punto y coma (;).

Archivos de texto:

Apuntadores en archivos:

Los apuntadores a un archivo se manejan en lenguaje C como variables apuntador de tipo FILE que se define en la cabecera stdio.h. La sintaxis para obtener una variable apuntador de archivo es la siguiente: FILE *F;

Funciones en archivos:

fopen() abre una secuencia para que pueda ser utilizada y la asocia a un archivo. Su estructura es la siguiente: *FILE fopen(char *nombre_archivo, char *modo);

fclose() cierra una secuencia que fue abierta mediante una llamada a fopen(). Escribe la información que se encuentre en el buffer al disco y realiza un cierre formal del archivo a nivel del sistema operativo: int fclose(FILE *apArch);

Donde apArch es el apuntador al archivo devuelto por la llamada a fopen().

fgets() y fputs() pueden leer y escribir, respectivamente, cadenas sobre los archivos.

Sus estructuras son:

char *fgets(char *buffer, int tamaño, FILE *apArch);

char *fputs(char *buffer, FILE *apArch);

fprintf() y fscanf() se comportan exactamente como printf() (imprimir) y scanf() (leer), excepto que operan sobre archivo. Sus estructuras son:

int fprintf(FILE *apArch, char *formato, ...);

int fscanf(FILE *apArch, char *formato, ...);

Donde apArch es un apuntador al archivo devuelto por una llamada a la función fopen().

fread permite leer uno o varios elementos de la misma longitud a partir de una dirección de memoria determinada (apuntador). Su estructura es la siguiente:

int fread(void *ap, size_t tam, size_t nelem, FILE *archivo)

fwrite permite escribir hacia un archivo uno o varios elementos de la misma longitud almacenados a partir de una dirección de memoria determinada. Su estructura es la siguiente:

int fwrite(void *ap, size_t tam, size_t nelem, FILE *archivo)

Bibliografía: Manual de prácticas del Laboratorio de Fundamentos de programación.