



Universidad Nacional Autónoma de México

Facultad de Ingeniería



Asignatura:

Estructura de Datos y Algoritmos I

Actividad 1: Repaso de lo que aprendí en la asignatura de Fundamentos de Programación.

Nombre del Alumno:

Sánchez Estrada Angel Isaac

Fecha:

24/febrero/2021



REPASO DE LO QUE APRENDÍ EN LA ASIGNATURA DE FUNDAMENTOS DE PROGRAMACIÓN

El conocimiento que adquiriré en la materia de Fundamentos de Programación fue esencial en mi desarrollo como estudiante de ingeniería ya que sienta las bases de un programador. En base a ello aprendí que el programador se va a ir especializando a las diferentes ramas que existe en la Programación como Analizador de Códigos, Desarrollador de aplicaciones, Programador orientada a Videojuegos, etc. en base a gustos personales, esa especialización se lograría gracias a que la ingeniería es multidisciplinaria ya que los conocimientos y experiencias adquiridas son adaptadas y utilizadas para distintos ámbitos de nuestras vidas.

Lo primero que aprendí fue acerca de el Sistema de Gestión de Calidad integrado en la página web del Laboratorio de Salas A y B de La facultad de Ingeniería UNAM, el cual Observe y Analice para adquirir información de suma importancia como lo sería los Reglamentos para el Uso de los laboratorios, los Protocolos de Seguridad, etc. Pero a su vez también contenía material que servía para alumnos y maestros, como lo seria las Caratulas para Entrega de Practicas, donde aprendí sobre la entrega de Prácticas en la cual se tiene que seguir siempre un formato adecuado y bien planeado que debe contener aspectos como Caratula, Objetivo, Introducción, Contenido visto en la Práctica, Conclusión y Referencias para una entrega completa.

The screenshot displays the website of the Laboratorio de Computación Salas A y B. At the top, there is a header with contact information and a navigation menu. The main content area features a large menu titled 'SISTEMA DE GESTIÓN DE LA CALIDAD' with various options. A document titled 'REGLAMENTO GENERAL DE USO DE LABORATORIOS Y TALLERES' is prominently displayed, including its approval date and the first section on objectives and definitions.

laboratorio.computacion.ab@gmail.com Ciudad Universitaria, Circuito Exterior
Anexo de Ingeniería, Edif. "Luis G. Valdés Vallejo"
Planta Baja, Salas A y B

PRINCIPAL ▾ SISTEMA DE GESTIÓN DE LA CALIDAD ▾ ASIGNATURAS ▾ SERVICIOS ▾ BOLETÍN CPI ▾ APRENDAMOS ▾

ISO 9001:2015

Política de Calidad
Buzón de sugerencias (FODO-26)
Protocolos de Seguridad
Resultados de evaluación práctica (FODO-30)
Plan de contingencia (ver. 1)
Encuestas
Misión y Visión
Plan de la calidad (PLDO-01)
Calendarios de prácticas (FODO-05)
Carátula para entrega de prácticas
Guía para la impartición de prácticas (GUDO-01)
Reglamentos

INGENIERÍA NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

**REGLAMENTO GENERAL DE USO DE
LABORATORIOS Y TALLERES¹**
Aprobado por el Consejo Técnico en sesión ordinaria del 30 de agosto de 2018

1. OBJETIVO Y DEFINICIONES

OBJETIVO

El presente reglamento tiene por objeto establecer el marco general al que debe sujetarse el

Lo siguiente que aprendí fue la Historia de la Programación en el cual comprendí que todo comenzó con Maquinas que ejecutaban operaciones básicas como suma, resta, multiplicación y división después fue actualizando teniendo Maquinas que aparte de hacer operaciones básicas también hacían cálculos de polinomios, después se empezaron a automatizar y crear sistemas de almacenamiento como tarjetas perforadas o un ejemplo muy conocido era la maquina de Turing que funcionaba con tubos de vacío la cual fue capaz de resolver problemas, haciendo también aportes a la lógica matemática y se puede observar mas detalladamente su funcionamiento en la película “El Código Enigma”. Después empezaron a surgir computadoras digitales y a través de esos sucesos fue aumentando y evolucionando la tecnología de la computación de ese modo surgiendo avances tecnológicos que en su tiempo eran muy grandes como las memorias para guardar información hasta llegar a lo que hoy se conoce como supercomputadoras donde la UNAM tiene en su poder una de ellas.

A su vez en este apartado aprendí que a lo largo del tiempo se han desarrollado diversos Lenguajes de Programación los cuales se clasificaban en:

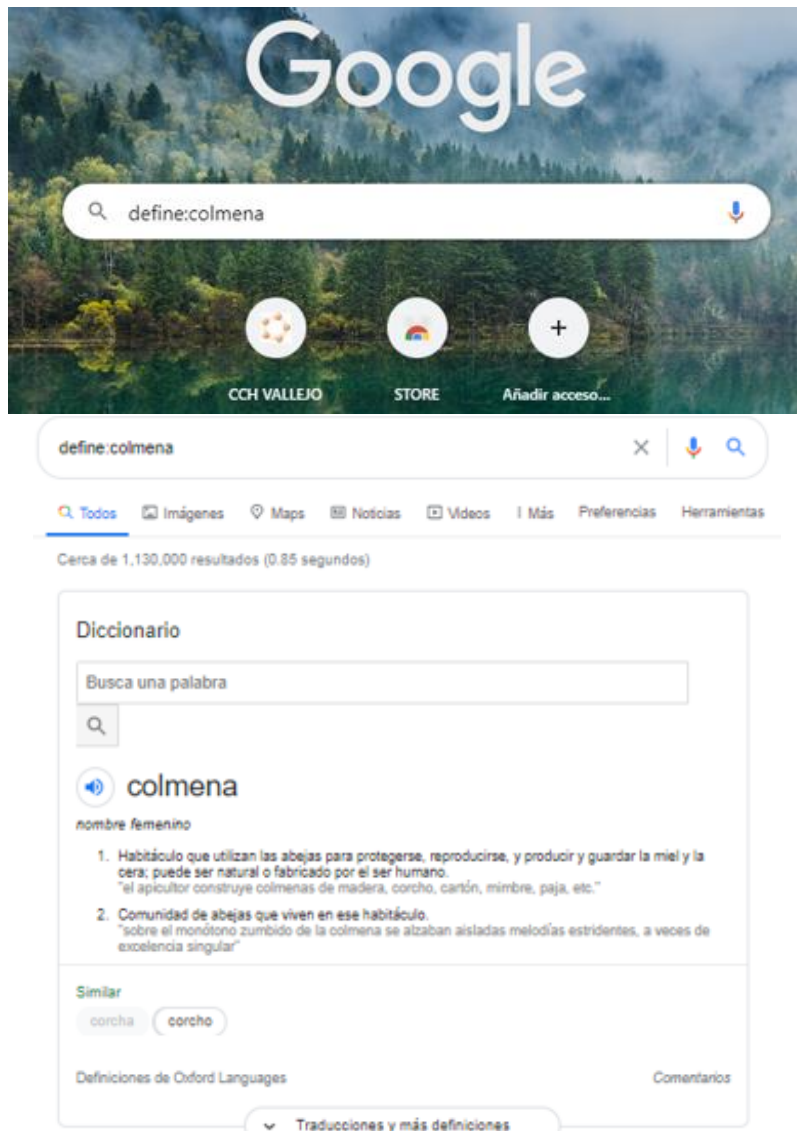
- Lenguaje Maquina
- Lenguaje de bajo nivel
- Lenguaje de alto nivel

El que comprendí mejor fue el Lenguaje de alto nivel ya que se caracterizan por su estructura semántica que es muy similar a la que escriben los humanos, lo que permite codificar los algoritmos de manera más natural, los ejemplos más representativos de lenguajes de alto nivel son:

- C++
- Fortran
- Java
- Perl
- PHP
- Python

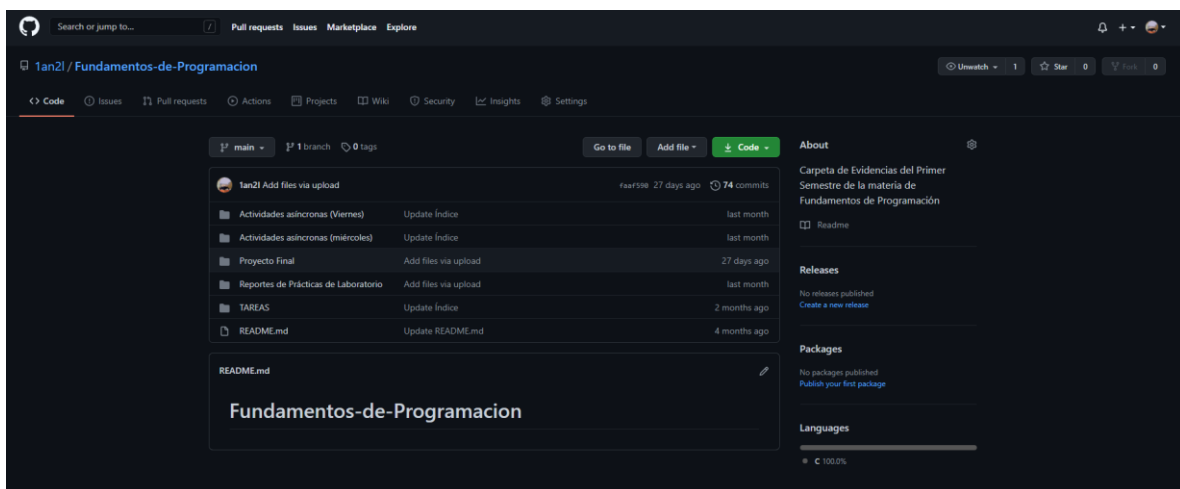
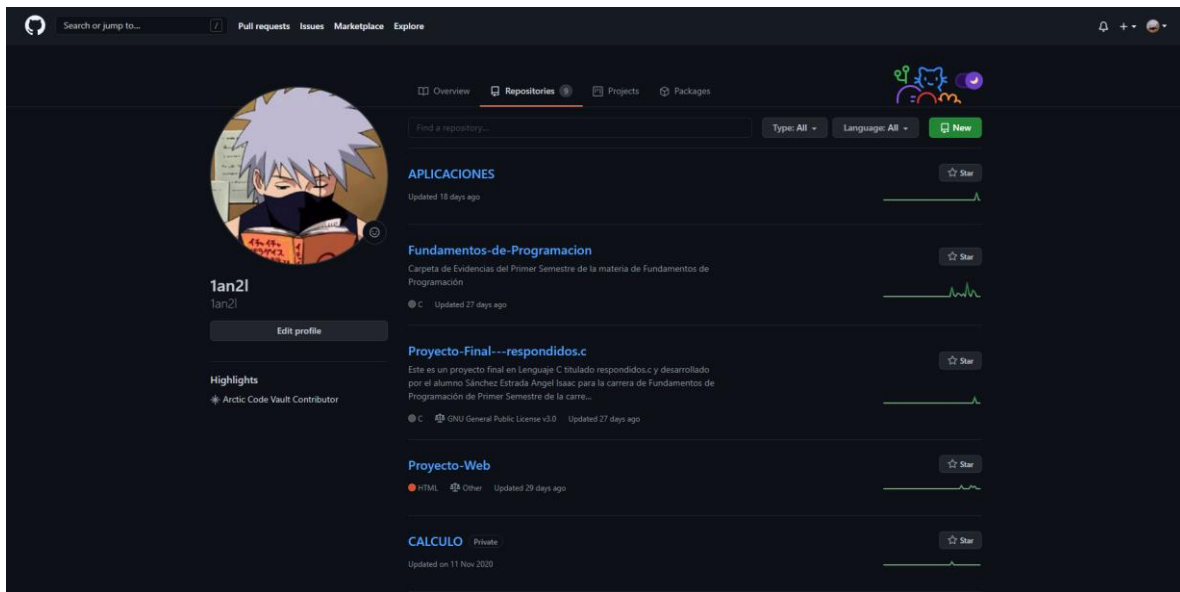
Al utilizar la computadora como herramienta de trabajo, Aprendí, descubrí y utilicé herramientas de software que se ofrecían en internet, que me permitieron realizar actividades y trabajos académicos de una forma organizada y profesional, las cuales eran el manejo de repositorios de almacenamiento como lo eran el Repositorio local o algún Repositorio Remoto el cual su alojamiento es un servidor externo como la nube ejemplos de este puede ser el GitHub, Google Drive, SkyDrive, etc. y buscadores con funciones avanzadas.

Algunos de los Buscadores de Internet mas conocidos que aprendí a identificar fueron Yahoo!, Google, DuckDuckgo. Algunos de los comandos que aprendí a utilizar con mayor frecuencia fueron el el Define, las comillas, Google Scholar y el sistema de conversiones que incluye Google.



Un tema que fue importante durante el Proceso de la materia de Fundamentos de Programación fue la creación de repositorios en la nube el cual el elegido fue GitHub en donde tuve la posibilidad de aprender como crear un repositorio como crear carpetas y subcarpetas, así como crear y modificar códigos a través de la plataforma de GitHub.

También me percate que el repositorio permitía el tener un registro de lo que se añadía, modificaba, etc. así como de esta una herramienta perfecta para la modificación de códigos, para equipos colaborativos, sin incluir que realice un proyecto de una página web que se hostiaba a través de GitHub page en donde podía modificar en tiempo real las páginas.



Respecto a GNU aprendí que es un sistema operativo de tipo Unix que está formado en su totalidad por software libre, ya que respeta la libertad de los usuarios de ejecutar, editar, contribuir y compartir, que esta mayoritariamente bajo términos de copyleft. De igual forma aprendí que la FSF o mejor conocida como Free Software Foundation o Fundación del Software Libre es una organización sin fines de lucro con la misión mundial de promover la libertad de los usuarios de computadoras. Y gracias a obtener ese conocimiento pude percatarme que a mi parecer todo el software debería ser libre para poder tener la libertad de modificarlo al gusto de que el usuario desee. Pude probar GNU/Linux atreves de servidores de la UNAM conocido como samba donde comencé a explorar los comandos por primera vez de un sistema Linux. Como el cd, pwd, clear, etc. permitiéndome así conectarme de una computadora a otra y poder navegar entre sus archivos de una manera rápida y sencilla.



Aprendí de igual forma que el ciclo de vida de un software que es un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso. A su vez aprendí que la diferencia del ciclo de vida de un software y la vida de un software, es que la vida de un software inicia cuando se instala en un dispositivo que posteriormente se empieza a utilizar y termina cuando se desinstala.

Dentro del ciclo de vida del software, en el análisis se busca comprender la necesidad, es decir, entender el problema. Entendí que la etapa del análisis consiste en conocer qué es lo que está solicitando el usuario. Para ello es importante identificar los dos conjuntos que conforman un sistema la entrada y la salida.

En cuanto al Algoritmo una vez realizado el análisis, es decir, ya que se entendió qué es lo que está solicitando el usuario y ya identificado el conjunto de entrada y el conjunto de salida, se puede proceder al diseño de la solución, de esta manera es que se genera el algoritmo. Durante el diseño del algoritmo se busca proponer una o varias alternativas viables para dar solución al problema y con base en esto tomar la mejor decisión para iniciar la construcción.

Las principales características que aprendí que debe llevar un algoritmo son:

- Preciso
- Definido
- Finito
- Debe tener al menos una salida
- Sencillo
- Legible
- Eficiente
- Eficaz

Al realizar los diagramas de Flujo Aprendí que es una herramienta fundamental para la elaboración de un procedimiento, porque a través de ellos podemos ver gráficamente y en forma consecutiva el desarrollo de un proceso o una actividad determinada.

Algo que aprendí sobre los diagramas de flujo es que poseen símbolos los cuales permiten estructurar la solución respecto a un problema planteado de manera gráfica. Los elementos que aprendí que conforma un diagrama son:

1. Tiene un inicio y un final.
2. Las líneas que indican la dirección del flujo del diagrama son verticales u horizontales.
3. Todas las líneas antes mencionadas se conectan a los símbolos.
4. Se construye de arriba abajo y de izquierda a derecha.
5. A cada línea del flujo solo le puede llegar una línea de dirección del flujo.
6. Para nombrar variables y nombres de funciones se debe hacer uso de la notación de camello.



También aprendí a realizar un pseudocódigo el cual me ayudo a representar una solución o algoritmo, de este modo expresaba los pasos de una solución lo mas detallada posible y lo más cercano a un lenguaje de programación sin ser tan estricto en cuanto a la sintaxis, creando una especie de lenguaje intermedio entre el lenguaje natural y el lenguaje de programación.

Algunas ventajas que me permitió el pseudocódigo fueron:

- Permite centrarse en aspectos lógicos de la solución.
- Abstracción de la sintaxis de un lenguaje de programación.
- Es un código escrito el cual lo entiende cualquier ser humano.
- Este es independiente del lenguaje de programación que se vaya a utilizar el cual en este semestre se utilizó el Lenguaje C.

Las reglas que llegue a utilizar más común mente al realizar los Pseudocódigos fueron:

1. Esta limitado por las etiquetas de INICIO y FIN. Dentro de estas etiquetas se deben escribir todas las instrucciones del programa.
2. Todas las palabras propias del pseudocódigo deben de ser escritas en mayúsculas.

3. El pseudocódigo debe tener diversas alineaciones para que el código sea más fácil de entender y depurar.
4. Para indicar lectura de datos se utiliza la etiqueta LEER. Para indicar escritura de datos se utiliza la etiqueta ESCRIBIR. La lectura de datos se realiza, por defecto, desde el teclado, que es la entrada estándar del sistema. La escritura de datos se realiza, por defecto, en la pantalla, que es la salida estándar del sistema.

Ya al saber lo esencial en cuanto a lo teórico empezamos a realizar el entorno de C el cual empecé a conocer para poder utilizar los ambientes y herramientas para el desarrollo y Ejecución de programas en lenguaje C como editores y compiladores de diversos sistemas operativos.

Algunos de los Editores de Texto que llegue a conocer para identificar a cuál me adaptaba más fueron: Notepad++, Visual Studio y GitHub atom. Una vez que un programa es codificado en C en alguno de los antes mencionados editores de texto debía ser leído por un programa que produjera el ejecutable. A este programa se le conoce como compilador y depende totalmente del hardware de la computadora y el sistema operativo que corre sobre ella.

En este semestre el compilador que se utilizó fue el GCC (GNU Compiler Collection) el cual es un conjunto de compiladores de uso libre para sistemas operativos basados en UNIX. Entre sus compiladores existe el que sirve para programas escritos en C. Ya que mi sistema operativo fue Windows tuve que utilizar una versión modificada para correr y crear programas para plataformas Windows en el paquete llamado MinGW. Ya que estuvo instalado y ambientado al sistema que preferí utilizar Windows empecé a ver como serian las sintaxis para su compilación o mejor dicho comandos.

Suponiendo que se tiene un programa escrito en C y se le llamó *calculadora.c* la manera de compilarlo es localizándose mediante la línea de comandos en la ruta donde el archivo se encuentra y ejecutando el comando:

```
gcc calculadora.c
```

Esto creará un archivo *a.out* (en Windows *a.exe*) que es el programa ejecutable resultado de la compilación.

Si se desea que la salida tenga un nombre en particular, debe definirse por medio del parámetro *-o* de gcc, por ejemplo, para que se llame *calculadora.out* (en Windows *calculadora.exe*):

```
gcc calculadora.c -o calculadora.out
```

A veces, para realizar un programa más complejo, se necesitan bibliotecas que se instalaron en el equipo previamente y se definió su uso en el programa escrito en C pero al momento de compilar es necesario indicar a GCC que se está usando bibliotecas que no se encuentran en su repertorio de bibliotecas estándar. Para ello es necesario utilizar el parámetro *-l* seguido inmediatamente por el nombre de la biblioteca, sin dejar espacio alguno:

```
gcc calculadora.c -o calculadora -lnombre_libreria
```


Al tener el Ejecutable el mismo GCC tenía métodos para hacerlo correr que serian los siguientes:

```
Considerando que se tiene un programa compilado en un sistema base Unix cuyo nombre es
calculadora.out, para ejecutar debe teclearse en línea de comandos:

./calculadora.out

Y en Windows, teniendo un programa llamado calculadora.exe debe teclearse en símbolo de
sistema:

calculadora.exe

En ambos casos localizándose previamente en la ruta donde se encuentra el ejecutable. En
Windows a veces puede omitirse mencionar .exe.
Si el programa realizado necesita tener una entrada de información por medio de argumentos,
éstos se colocan así:

calculadora argumento1 argumento2
```

Y de este modo ya tuve mi entorno en cd a lo cual me llevo a investigar sobre los comandos para la navegación entre archivos a través del símbolo de sistema como era el de dir, cd, cls, etc. que posteriormente utilizaría en mi proyecto llamando a la librería de Windows para su ejecución.

Ya que obtuvimos el entorno en C empecé a descubrir cómo se realizaban las escrituras de pantalla con printf pero antes de eso teníamos que declarar el inicio y el final con int main() y return 0 pero antes de esto teníamos que llamar a las librerías y en este caso la principal es #include<stdio.h> y con eso pude realizar la escritura de pantalla.

Ante esto empecé a descubrir diferentes comandos y librerías que se podían ocupar en el Lenguaje en C, pero una cosa que aprendí que le suma valor a cualquier código son los comentarios los cuales especifican que acción o que parte es lo que se lleva a cabo en cierta sección del código.

El Lenguaje en C permite las combinaciones de distintas funciones para hacer un código mas completo y obtener una solución a alguna problemática.

En el Lenguaje C existen tipos de Datos los cuales son:

- Caracteres: codificación definida por la máquina.
- Enteros: números sin punto decimal.
- Flotantes: números reales de precisión normal.
- Dobles: números reales de doble precisión.

Las variables enteras que existes en Lenguaje C son:

<i>Tipo</i>	<i>Bits</i>	<i>Valor Mínimo</i>	<i>Valor Máximo</i>
<i>signed char</i>	8	-128	127
<i>unsigned char</i>	8	0	255
<i>signed short</i>	16	-32 768	32 767
<i>unsigned short</i>	16	0	65 535
<i>signed int</i>	32	-2,147,483,648	2 147 483 647
<i>unsigned int</i>	32	0	4 294 967 295
<i>signed long</i>	64	9 223 372 036 854 775 808	9 223 372 036 854 775 807
<i>unsigned long</i>	64	0	18 446 744 073 709 551 615
<i>enum</i>	16	-32 768	32 767

Existen otras variables las cuales son de tipo real y son:

<i>Tipo</i>	<i>Bits</i>	<i>Valor Mínimo</i>	<i>Valor Máximo</i>
<i>float</i>	32	3.4 E-38	3.4 E38
<i>double</i>	64	1.7 E-308	1.7 E308
<i>long double</i>	80	3.4 E-4932	3.4 E4932

Para poder acceder al valor de una variable se requiere especificar el tipo de dato. Los especificadores que tiene lenguaje C para los diferentes tipos de datos son:

<i>Tipo de dato</i>	<i>Especificador de formato</i>
<i>Entero</i>	%d, %i, %ld, %li, %o, %x
<i>Flotante</i>	%f, %lf, %e, %g
<i>Carácter</i>	%c, %d, %i, %o, %x
<i>Cadena de caracteres</i>	%s

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

En cuanto a los Identificadores es el nombre el nombre con el que se va a almacenar en memoria un tipo de dato. Los identificadores siguen las siguientes reglas:

- Debe iniciar con una letra [a-z].
- Puede contener letras [A-Z, a-z], números [0-9] y el carácter guion bajo (_).

La biblioteca 'stdio.h' contiene diversas funciones tanto para imprimir en la salida estándar (monitor) como para leer de la entrada estándar (teclado).

printf es una función para imprimir con formato, es decir, se tiene que especificar entre comillas el tipo de dato que se desea imprimir, también se puede combinar la impresión de un texto predeterminado:

```
printf("El valor de la variable real es: %lf", varReal);
```

scanf 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.

```
scanf ("%i", &varEntera);
```

Para imprimir con formato también se utilizan algunas secuencias de caracteres de escape, C maneja los siguientes:

- \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

Otro aspecto importante son los Modificadores de Alcance

Los modificadores que se pueden agregar al inicio de la declaración de variables son const y static.

El modificador const impide que una variable cambie su valor durante la ejecución del programa, es decir, permite para crear constantes. Por convención, las constantes se escriben con mayúsculas y se deben inicializar al momento de declararse.

Otra parte de el Lenguaje C son los operadores los cuales existen los aritméticos los cuales son:

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
+	Suma	125.78 + 62.5	188.28
-	Resta	65.3 - 32.33	32.97
*	Multiplicación	8.27 * 7	57.75
/	División	15 / 4	3.75
%	Módulo	4 % 2	0

Los operadores lógicos a nivel de bits que maneja el lenguaje C se describen en la siguiente tabla:

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
>>	Corrimiento a la derecha	8 >> 2	2
<<	Corrimiento a la izquierda	8 << 1	16
&	Operador AND	5 & 4	4
	Operador OR	3 2	3
~	Complemento ar-1	~2	1

Como ultimo apartado que lleve a ver respecto a lo esencial en Lenguaje se fueron las Expresiones Lógicas las cuales están constituidas por números, caracteres, constantes o variables que están relacionados entre sí por operadores lógicos. Una expresión lógica puede tomar únicamente los valores verdadero o falso.

Los operadores de relación permiten comparar elementos numéricos, alfanuméricos, constantes o variables.

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
==	Igual que	'h' == 'H'	Falso
!=	Diferente a	'a' != 'b'	Verdadero
<	Menor que	7 < 15	Verdadero
>	Mayor que	11 > 22	Falso
<=	Menor o igual	15 <= 22	Verdadero
>=	Mayor o igual	20 >= 35	Falso

Los operadores lógicos permiten formular condiciones complejas a partir de condiciones simples.

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>
!	No	!p
&&	Y	a > 0 && a < 11
	O	opc == 1 salir != 0

Lenguaje C posee operadores para realizar incrementos y decrementos de un número.

El operador ++ agrega una unidad (1) a su operando. Es posible manejar preincrementos (++n) o posincrementos (n++).

El operador -- resta una unidad (1) a su operando. Se pueden manejar predecrementos (-- n) o posdecrementos (n--).

Posteriormente Empecé a checar las Estructuras de Selección permiten realizar una u otra acción con base en una expresión lógica. Algunas de las Estructuras son las siguientes:

- Estructura de control selectiva if

La estructura de control de flujo más simple es la estructura condicional if, su sintaxis es la siguiente:

```
if (expresión_lógica) {  
    // bloque de código a ejecutar  
}
```

En esta estructura se evalúa la expresión lógica y, si se cumple (si la condición es verdadera), se ejecutan las instrucciones del bloque que se encuentra entre las llaves de la estructura. Si no se cumple la condición, se continúa con el flujo normal del programa

- Estructura de control selectiva if-else
Esta estructura evalúa la expresión lógica y si la condición es verdadera se ejecutan las instrucciones del bloque que se encuentra entre las primeras llaves, si la condición es falsa se ejecuta el bloque de código que está entre las llaves después de la palabra reservada 'else'. Al final de que se ejecute uno u otro código, se continúa con el flujo normal del programa. Es posible anidar varias estructuras if-else, es decir, dentro de una estructura if-else tener una o varias estructuras if-else.
- Enumeración
Existe otro tipo de dato constante conocido como enumeración. Una variable enumerador se puede crear de la siguiente manera:

```
enum identificador {VALOR1, VALOR2, ... , VALORN};
```

Para crear una enumeración se utiliza la palabra reservada enum, seguida de un identificador (nombre) y, entre llaves se ingresan los nombres de los valores que puede tomar dicha enumeración, separando los valores por coma. Los valores son elementos enteros y constantes (por lo tanto se escriben con mayúsculas).

- Estructura de control selectiva condicional
La estructura condicional (también llamado operador ternario) permite realizar una comparación rápida. Su sintaxis es la siguiente:

Condición ? SiSeCumple : SiNoSeCumple

Consta de tres partes, una condición y dos acciones a seguir con base en la expresión condicional. Si la condición se cumple (es verdadera) se ejecuta la instrucción que se encuentra después del símbolo '?'; si la condición no se

cumple (es falsa) se ejecuta la instrucción que se encuentra después del símbolo ':'.

Al ver estructuras de repetición me percate que nos permiten ejecutar un conjunto de instrucciones de manera repetida las veces que se requiera, mientras que la expresión lógica a evaluar se cumpla o sea verdadera.

En lenguaje C existen tres estructuras de repetición:

- While: Esta estructura valida la condición lógica dada y en caso de ser verdadera ejecutará el bloque de instrucciones contenido en ella, por el contrario, si la condición es falsa seguirá el flujo del programa. Las instrucciones dentro de la estructura while dejarán de ser ejecutadas cuando la condición sea falsa.
- do-while: A diferencia del while, esta estructura ejecutará una vez el conjunto de instrucciones y luego validará la condición lógica, si es falsa continuará ejecutando las instrucciones hasta que la condición se vuelva falsa.
- for: La estructura for consta de tres partes, en una de ellas se declaran variables, en la segunda se valida una condición lógica, la última parte son instrucciones que se realizan una vez se termina de ejecutar el conjunto de instrucciones, como dato curioso, la letra i es utilizada dentro de estas estructuras como referencia a iteración, pudiendo ser llamada variable de iteración.

En cuanto a las Estructura While:

La estructura repetitiva (o iterativa) while primero valida la expresión lógica y si ésta se cumple (es verdadera) procede a ejecutar el bloque de instrucciones de la estructura, el cual está delimitado por las llaves {}. Si la condición no se cumple se continúa el flujo normal del programa sin ejecutar el bloque de la estructura, es decir, el bloque se puede ejecutar de cero a ene veces. Su sintaxis es la siguiente:

```
while (expresión_lógica) {  
    // Bloque de código a repetir  
    // mientras que la expresión  
    // lógica sea verdadera.  
}
```


Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves.

En cuanto a las de do-while:

do-while es una estructura cíclica que ejecuta el bloque de código que se encuentra dentro de las llaves y después valida la condición, es decir, el bloque de código se ejecuta de una a nueve veces. Su sintaxis es la siguiente:

```
do {  
    /*  
    Bloque de código que se ejecuta  
    por lo menos una vez y se repite  
    mientras la expresión lógica sea  
    verdadera.  
    */  
} while (expresión_lógica);
```

Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves. Esta estructura de control siempre termina con el signo de puntuación ';'.

En cuanto a la de for:

Lenguaje C posee la estructura de repetición for la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer. La sintaxis que generalmente se usa es la siguiente:

```
for (inicialización ; expresión_lógica ; operaciones por iteración) {  
    /*  
    Bloque de código  
    a ejecutar  
    */  
}
```

La estructura for ejecuta 3 acciones básicas antes o después de ejecutar el bloque de código. La primera acción es la inicialización, en la cual se pueden definir variables e inicializar sus valores; esta parte solo se ejecuta una vez cuando se ingresa al ciclo y es opcional. La segunda acción consta de una expresión lógica, la cual se evalúa y, si ésta es verdadera, ejecuta el bloque de código, si no se cumple se continúa la ejecución del programa; esta parte es opcional. La tercera parte consta de un conjunto de operaciones que se realizan cada vez que termina de ejecutarse el bloque de código y antes de volver a validar la expresión lógica; esta parte también es opcional.

En él **Define** las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto, o incluir un archivo en otro archivo con #include.

define permite definir constantes o literales; se les nombra también como constantes simbólicas. Su sintaxis es la siguiente:

```
#define <nombre> <valor>
```

Al definir la constante simbólica con #define, se emplea un nombre y un valor. Cada vez que aparezca el nombre en el programa se cambiará por el valor definido. El valor puede ser numérico o puede ser texto.

En algunos casos es conveniente tener la posibilidad de abandonar un ciclo. La proposición **break** proporciona una salida anticipada dentro de una estructura de repetición, tal como lo hace en un switch. Un break provoca que el ciclo que lo encierra termine inmediatamente.

La proposición **continue** provoca que inicie la siguiente iteración del ciclo de repetición que la contiene.

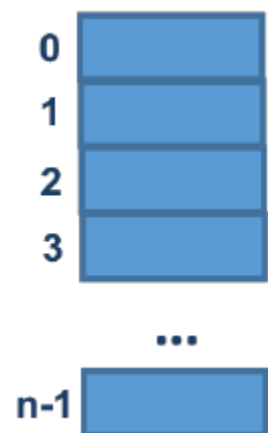
Algo de Suma importancia al Realizar código es la Depuración de Programas la cual es el proceso de identificar y corregir errores de programación y es útil cuando:

- Se desea optimizar un programa
- El programa tiene algún fallo
- El programa tiene un error de ejecución o defecto

Después de que Fue amentando tiempo del semestre nos adentramos a los arreglos unidimensionales y multidimensionales los cuales ayudaron para la elaboración de programas que resuelven problemas que requieran agrupar datos del mismo tipo.

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.

Los arreglos unidimensionales son un tipo de datos estructurado que está formado de una colección finita y ordenada de datos del mismo tipo. Es la estructura natural para modelar listas de elementos iguales. Están formados por un conjunto de elementos de un mismo tipo de datos que se almacenan bajo un mismo nombre, y se diferencian por la posición que tiene cada elemento dentro del arreglo de datos. Al declarar un arreglo, se debe inicializar sus elementos antes de utilizarlos. Para declarar un arreglo tiene que indicar su tipo, un nombre único y la cantidad de elementos que va a contener.



Los arreglos multidimensionales son un tipo de dato estructurado, que está compuesto por dimensiones. Para hacer referencia a cada componente del arreglo es necesario utilizar n índices, uno para cada dimensión. El término dimensión representa el número de índices utilizados para referirse a un elemento particular en el arreglo. Los arreglos de más de una dimensión se llaman arreglos multidimensionales.

Los arreglos con múltiples subíndices son la representación de tablas de valores, consistiendo de información arreglada en renglones y columnas. Para identificar un elemento particular de la tabla, deberemos de especificar dos sub-índices; el primero identifica el renglón del elemento y el segundo identifica la columna del elemento. A los arreglos que requieren dos subíndices para identificar un elemento en particular se conocen como arreglo de doble subíndice. Note que los arreglos de múltiples subíndices pueden tener más de dos subíndices. El estándar ANSI indica que un sistema ANSI C debe soportar por lo menos 12 subíndices de arreglo.

La sintaxis para definir un **arreglo unidimensional** en lenguaje C es la siguiente:

`tipoDeDato nombre[tamaño]`

Donde nombre se refiere al identificador del arreglo, tamaño es un número entero y define el número máximo de elementos que puede contener el arreglo. Un arreglo puede ser de los tipos de dato entero, real, carácter o estructura.

Lenguaje C permite crear arreglos de varias dimensiones con la siguiente sintaxis:

`tipoDato nombre [tamaño][tamaño]...[tamaño];`

Donde nombre se refiere al identificador del arreglo, tamaño es un número entero y define el número máximo de elementos que puede contener el arreglo por dimensión (el número de dimensiones está determinado por el número de corchetes). Los tipos de dato que puede tolerar un arreglo multidimensional son: entero, real, carácter o estructura.

De manera práctica se puede considerar que la primera dimensión corresponde a los renglones, la segunda a las columnas, la tercera al plano, y así sucesivamente. Sin embargo, en la memoria cada elemento del arreglo se guarda de forma contigua, por lo tanto, se puede recorrer un arreglo multidimensional con apuntadores.

Después empezamos a adentrarnos a otra forma de guardar los datos de entrada de ese modo aprendí a identificar y utilizar los apuntadores los cuales son una variable que contiene la dirección de una variable, es decir, hace referencia a la

localidad de memoria de otra variable. Debido a que los apuntadores trabajan directamente con la memoria, a través de ellos se accede con rapidez a un dato.

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

```
TipoDeDato *apuntador, variable;
```

```
apuntador = &variable;
```

La declaración de una variable apuntador inicia con el carácter *. Cuando a una variable le antecede un comparador, lo que se hace es acceder a la dirección de memoria de la misma (es lo que pasa cuando se lee un dato con scanf).

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 *.

Un programa en C consiste en funciones, las cuales pueden estar definidas en la librería estándar o bien definidas por el propio usuario. Las funciones definidas por el usuario constan de, el tipo de dato de la función, su nombre, y los parámetros que tendrán de entrada, así como a final de esta, el valor de retorno que aportará al código principal.

La función es un fragmento de código que realiza una tarea bien definida. pueden ser utilizadas por el programador. Este tipo de funciones predefinidas son denominadas funciones de biblioteca. Sin embargo, cada programador puede definir sus propias funciones de acuerdo a sus necesidades. Las funciones que define el programador son conocidas como funciones de usuario.

La sintaxis básica para definir una función es la siguiente:

```
valorRetorno nombre (parámetros){
```

```
// bloque de código de la función
```

```
}
```

El nombre de la función se refiere al identificador con el cual se ejecutará la función; se debe seguir la notación de camello.

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, de la siguiente forma:

```
(tipoDato nom1, tipoDato nom2, tipoDato nom3...)
```

El tipo de dato puede ser cualquiera de los vistos hasta el momento (entero, real, carácter o arreglo) y el nombre debe seguir la notación de camello. Los parámetros de una función son opcionales.

Las variables declaradas dentro de un programa tienen un tiempo de vida que depende de la posición donde se declaren. En C existen dos tipos de variables con base en el lugar donde se declaren: variables locales y variables globales.

Las variables que se declaren dentro de cada función se conocen como variables locales (a cada función). Estas variables existen al momento de que la función es llamada y desaparecen cuando la función llega a su fin.

Las variables que se declaran fuera de cualquier función se llaman variables globales. Las variables globales existen durante la ejecución de todo el programa y pueden ser utilizadas por cualquier función.

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.

La función main también puede recibir parámetros. Debido a que la función main es la primera que se ejecuta en un programa, los parámetros de la función hay que enviarlos al ejecutar el programa. La firma completa de la función main es:

```
int main (int argc, char ** argv);
```

Lenguaje C permite definir elementos estáticos. La sintaxis para declarar elementos estáticos es la siguiente:

```
static tipoDato nombre;
```

```
static valorRetorno nombre(parámetros);
```

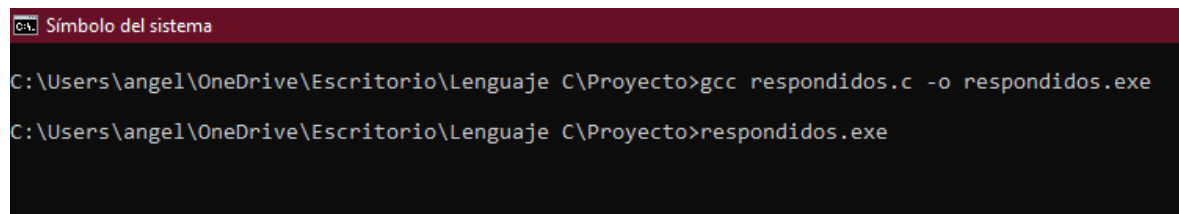
Es decir, tanto a la declaración de una variable como a la firma de una función solo se le agrega la palabra reservada static al inicio de las mismas. El atributo static en una variable hace que ésta permanezca en memoria desde su creación y durante toda la ejecución del programa, lo que quiere decir que su valor se mantendrá hasta que el programa llegue a su fin. El atributo static en una función hace que esa función sea accesible solo dentro del mismo archivo, lo que impide que fuera de la unidad de compilación se pueda acceder a la función.

El último tema que se vio en la clase de Fundamentos de Programación fue la Lectura y escritura de datos en el cual elaboramos programas en Lenguaje C que requerían el uso de archivos de texto plano en la resolución de problemas, lo que hacia era permitir la entrada y la salida de datos desde o hacia un archivo,

respectivamente, a través del uso de la biblioteca de funciones de la cabecera `stdio.h`.

Para terminar durante transcurría el semestre fuimos elaborando un proyecto final en el cual aplicamos todos los conocimientos adquiridos en la materia empezando planteando una problemática, pasando por el algoritmo, diagrama de flujo, y Pseudocódigo. Al pasar por todo eso empezamos a construir la solución que sería el código fuente donde ya no solo ocupaba una librería si no tres las cuales eran `#include<stdio.h>`, `#include<windows.h>`, `#include<conio.h>`, aprendí a comentar, a declarar variables a utilizar funciones del sistema Windows para limpiar pantalla y cambiar los colores de las letras para que tuviera una estética excelente ocupe repetidores como `do while`, una parte muy importante para solucionar la problemática de mi proyecto fueron los arreglos multidimensionales los cual los necesitaba para solicitar nombre y contraseña para realizar un login y algo de lo que podría decir que destaco mi proyecto fue que ocupe todos los conocimientos adquiridos durante el semestre y agregando a su vez investigaciones propias realizadas durante el paso del semestre. Cabe destacar que siempre se llevó un registro de las actividades que se llegaban a presentar atreves de un servidor externo que utilizaba la nube llamado Github, para percatarnos de los cambios de cuando iniciamos el proyecto y cuando lo terminamos llevamos una comparativa de lo antes esperado con lo ahora realizado a través de tablas comparativas de los recursos que pensábamos ocupar vs los que utilizamos, también la realización de diagramas de Gantt nos permitió visualizar la realizad de cuanto tarda y cuáles son los pasos para la realización de proyectos, lo más importante para un proyecto considero que es la documentación de ellos y existen muchos tipos de hacerlo como un trabajo o un video que fueron los medios que utilizamos para difundir nuestros proyectos y a su vez una parte importante de aprender es el ver como realmente es un proceso adecuado en ámbito laboral o escolar para la realización de proyectos tanto en equipo como individual.

FUNCIONAMIENTO DEL PROYECTO



```
C:\> Símbolo del sistema
C:\Users\angel\OneDrive\Escritorio\Lenguaje C\Proyecto>gcc respondidos.c -o respondidos.exe
C:\Users\angel\OneDrive\Escritorio\Lenguaje C\Proyecto>respondidos.exe
```

Se compila y ejecuta, como integre la función `system "cls"` limpia pantalla al ejecutar


```
♠ Bienvenidos a respondidos.c ♠

1) Registro con Usuario
2) Salir

Elige una opción de la lista: »
```

Empieza arrojando un menú realizado con do while, printf y scanf para que guarde el numero elegido y pueda ser pasada la indicación al primer switch

```
♠ Bienvenidos a respondidos.c ♠

1) Registro con Usuario
2) Salir

Elige una opción de la lista: »1

Escriba su nombre de usuario: »_
```

Al colocar la opción 1 te solicita poner el nombre de usuario que tienes en el juego

```
♠ Bienvenidos a respondidos.c ♠

1) Registro con Usuario
2) Salir

Elige una opción de la lista: »1

Escriba su nombre de usuario: »1an2l

Contraseña: »proyectoX

    Su registro a respondidos fue exitoso!!

Nombre de Usuario: 1an2l
PARTIDAS:
1) carlos.ram
2) marta.j
3) marco.fi
4) Actualizar partidas
5) Salir
```

Te arroja el segundo menú donde te muestra los nombres de los contrincantes

```

      ♠ Bienvenidos a respondidos.c ♠

1) Registro con Usuario
2) Salir

Elige una opción de la lista: »1

Escriba su nombre de usuario: »1an2l

Contraseña: »proyectoX

      Su registro a respondidos fue exitoso!!

Nombre de Usuario: 1an2l
PARTIDAS:
1) carlos.ram
2) marta.j
3) marco.fi
4) Actualizar partidas
5) Salir

Elige una opción de la lista: »1

-----
Oponente: carlos.ram
-----
Categoría:HISTORY

Pregunta: ¿Como se llama este politico estadounidense?

Respuesta: 2) Barack Obama
```

Al poner el primer contrincante me aparece la categoría de la pregunta la pregunta y la respuesta

```
Elige una opción de la lista: »1
-----
Oponente: carlos.ram
-----
Categoria: HISTORY
Pregunta: ¿Como se llama este politico estadounidense?
Respuesta: 2) Barack Obama
```

```
PARTIDAS:
1) carlos.ram
2) marta.j
3) marco.fi
4) Actualizar partidas
5) Salir
```

```
Elige una opción de la lista: »2
-----
Oponente: marta.j
-----
Categoria: ENTERTAINMENT
Pregunta: ¿Donde nacio Xavier Lopez Alias: Chavelo ?
Respuesta: 2) Chicago
```

Al poner 2 me arroja al segundo contrincante y ocurre lo mismo que paso en el primer contrincante

```

PARTIDAS:
1) carlos.ram
2) marta.j
3) marco.fi
4) Actualizar partidas
5) Salir

Elige una opción de la lista: »3

-----
Oponente: marco.fi
-----
Categoria: ARTS

Pregunta: ¿Quien es ella?

Respuesta: 4) Alicia

PARTIDAS:
1) carlos.ram
2) marta.j
3) marco.fi
4) Actualizar partidas
5) Salir

Elige una opción de la lista: »4

    Acabas de Actualizar partidas

PARTIDAS:
1) carlos.ram
2) marta.j
3) marco.fi
4) Actualizar partidas
5) Salir

Elige una opción de la lista: »5

    Elegiste la opción salir

1) Registro con Usuario
2) Salir

Elige una opción de la lista: »2

    Elegiste la opción salir

Gracias por usar respondidos.c regrese pronto =)

```

En el oponente 3 Ocurre lo mismo y a qui se muestra el funcionamiento de la función salir que nos proporciona el do while y al último un mensaje de gracias por usar la aplicación y que regrese pronto.

Algunas de las limitaciones es que no es posible obtener las respuestas en tiempo real por lo cual opte por definir las después y dejar libre el usuario y contraseña ya que no puedo obtener registro de los que si participan en tiempo real en el juego “Preguntados”

Referencias:

- Laboratorio Salas A y B. (s. f.). Laboratorio de Computación Salas A y B. Recuperado 24 de febrero del 2021, de <http://lcp02.fi-b.unam.mx>
- Sánchez, A. I. (s. f.). 1an2I - Overview. GitHub. Recuperado 24 de febrero del 2021, de <https://github.com/1an2I?tab=repositories>
- Facultad de ingeniería - UNAM. (2018, 6 abril). Manual de prácticas del laboratorio de Fundamentos de programación. Recuperado 24 de febrero del 2021, de: http://odin.fi-b.unam.mx/salac/practicasp/MADO-17_FP.pdf