Universidad Nacional Autónoma de México

Facultad de Ingeniería

Asignatura:

Estructura de Datos y Algoritmos I

Actividad #1 - lunes | Repaso de Conocimientos en C

Nombre del Alumno:

Sánchez Estrada Angel Isaac

Maestro:

M.I. Marco Antonio Martínez Quintana

Grupo:

15

Fecha:

07/06/2021

REPASO DE CONOCIMIENTOS EN C

Fundamentos de Programación

Repase de nuevo como es el funcionamiento del Git Hub, pero ahora utilizando herramienta de Git proporcionado por el editor de Código VS Studio y lo repase a través de un curso en la plataforma de Udemy

https://www.udemy.com/share/104exWAEYfeV1bQ3wJ/

Repace los Diagramas de Flujo en donde cheque el cómo se desarrollan y el cómo se conforman apoyándome a través de los apuntes que realice el semestre pasado

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 repasé el Pseudocódigo en donde recordé que me ayuda a representar los pasos de una solución y que tiene que ser lo mas detallada posible y lo más cercano a un lenguaje de programación sin ser tan estricto en cuanto a la sintaxis,

haciendo de esa forma un lenguaje intermedio entre el lenguaje natural (humano) y el lenguaje de programación

Las ventajas que note que daban los Pseudocódigo fueron:

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

Al igual repase algunas de las reglas que se utilizan para la realización de los Pseudocódigos.

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.

También realices un pequeño repaso de la ejecución del compilador GCC el cual es un conjunto de compiladores de uso libre para sistemas operativos basados en UNIX.

Sus comandos o sintaxis para su compilación son los siguientes:

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
```

Y su forma para ejecutar los programas seria:

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
```

Repase los tipos de Datos que existe en el Lenguaje C 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.

Sus variables enteras que existen en Lenguaje C son:

Tipo	Bits	Valor Mínimo	Valor Máximo
signed char	8	-128	127
unsigned char	8	0	255
signed short	16	-32 768	32 767
unsigned short	16	0	65 535
signed int	32	-2 147 483 648	2 147 483 647

unsigned int	32	0	4 294 967 295
signed long	64	9 223 372 036 854 775 808	9 223 372 036 854 775 807
unsigned long	64	0	18 446 744 073 709 551 615
enum	16	-32 768	32 767

Las variables de tipo real son:

Tipo	Bits	Valor Mínimo	Valor Máximo
float	32	3.4 E-38	3.4 E38
double	64	1.7 E-308	1.7 E308
Long double	80	3.4 E-4932	3.4 E4932

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

- Entero: %d, %i, %ld, %li, %o, %x
- Flotan: %f, %lf, %e, %g
- Carácter: %c, %d, %i, %o, %x
- Cadena de Caracteres: %s

Repase a su vez algunas de las funciones que se necesitan para lo básico en Lenguaje C como:

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

Otra parte del Lenguaje C son los operadores los cuales existen los aritméticos los cuales son:

Operador	*Operación*	*Uso*	*Resultado*
+	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:

Operador	*Operación*	*Uso*	*Resultado*
>>	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

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:

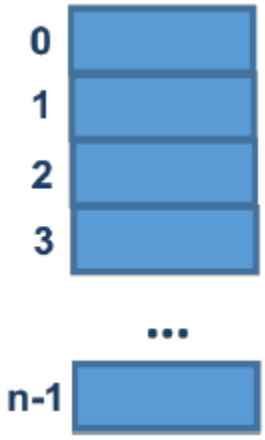
- **While:** Esta estructura valida la condición lógica dada y en caso de ser verdadera ejecutará el bloque de instrucciones contenido en ella, por el

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 ter-mina 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.

También di repaso sobre los arreglos en Lenguaje C los cuales son:



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.

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

tipoDeDato nombre[tamaño]

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.

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

tipoDato nombre [tamaño][tamaño]…[tamaño];

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.

Lectura y escritura de datos

- fopen(): Abre una secuencia para que pueda ser utilizada y la asocia a un archivo.

  *FILE fopen(char *nombre_archivo, char *modo);

  Modos de abrir los archivos

  - r: Abre un archivo de texto para lectura.
  - w: Crea un archivo de texto para escritura.
  - a: Abre un archivo de texto para añadir.
  - r+: Abre un archivo de texto para lectura / escritura.
  - w+: Crea un archivo de texto para lectura / escritura.
  - a+: Añade o crea un archivo de texto para lectura / escritura.
  - rb: Abre un archivo en modo lectura y binario.
  - wb: Crea un archivo en modo escritura y binario

- fclose(): Cierra una secuencia que fue abierta mediante una llamada a fopen().

  int fclose(FILE *apArch);

- fgets() y fputs(): Pueden leer y escribir, respectivamente, cadenas sobre los archivos.

  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.

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

- fread y fwrite: Son funciones que permiten trabajar con elementos de longitud conocida. fread permite leer uno o varios elementos de la misma longitud a partir de una dirección de memoria determinada (apuntador).

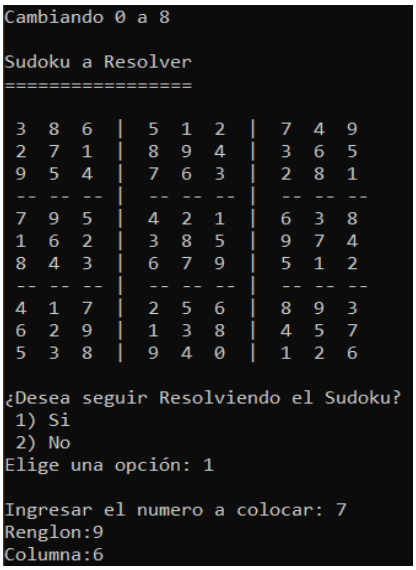
  int fread(void *ap, size_t tam, size_t nelem, FILE *archivo)

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

Estructura de Datos y Algoritmos I

Sudoku y Cifrado de Cesar

Repase el cómo desarrolle el sudoku y viendo como ocupe los arreglos y matrices para su desarrollo.



De la misma forma repase lo del cifrado de cesar que es uno de los primeros métodos de cifrado conocidos históricamente. En el siglo I antes de Cristo, Julio César el célebre militar y político, usó el cifrado de César, también conocido como cifrado por desplazamiento, para enviar órdenes a sus generales en los campos de batalla, esta es una de las técnicas decodificación más simples y más usadas. El método consiste en desplazar el abecedario 3 posiciones, es decir, en lugar de iniciar en la letra A, el abecedario inicia en la letra D.

Alfabeto en claro:	A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z
Alfabeto cifrado:	D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z A B C

Punteros en C

Y repace los apuntadores en C que también son llamados punteros, esta es una variable que contiene la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable.

Se dice que los punteros “apuntan” a la variable cuyo valor se almacena a partir de la dirección de la memoria que contiene el apuntador.

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

```
TipoDeDato *apuntador, variable;

apuntador = &variable;
```

- El Operador de Dirección (&) regresa la dirección de una variable.
- El Operador de Indirección (*), toma la dirección de una variable y regresa el dato que contiene esa dirección.

La declaración de una variable apuntador inicia con el carácter (*). Cuando a una variable le antecede un ampersand, 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 (*).

Y los punteros se utilizan principalmente para la construcción de referencias, que a su vez son fundamentales para la construcción de casi todas las estructuras de datos, así como para pasar datos entre las diversas partes de un programa.

Debido a que los apuntadores trabajan directamente con la memoria, para acceder a ellos con rapidez para obtener un dato, se suelen utilizar para dar claridad y simplicidad a las operaciones a nivel de la memoria, por otra parte, se ocupan de igual forma para pasar parámetros por referencia. Esto es útil si el programador quiere modificaciones de una función a un parámetro.

Escítala Espartana

Repace que es un artefacto cilíndrico que resulta de proyectar un polígono regular de n lados, sobre el cual se enrosca una tira de cuero o papiro, es considerado uno de los primeros métodos de cifrado o sistema de criptografía, el cual fue utilizado en la guerra entre Atenas y Esparta para proteger el contenido de sus mensajes de los ojos del enemigo al ser enviados.

Para comunicarse los militares espartanos enrollaban en forma de espiral una tira de cuero o papiro en una escítala (palo o bastón). Sobre esa tira se escribía el mensaje longitudinalmente, luego se desenrollaba y se enviaba a destino. Si alguien interceptaba el mensaje, aunque leyera la tira no entendería su contenido debido a que las letras aparecerían mezcladas. Este método requería que el receptor dispusiera de una escítala idéntica a la utilizada para cifrar el mensaje, para que, enrollando la tira en el bastón, pudiera descifrarlo.

Asu vez vi ejemplos para repasar y su funcionamiento con los desarrollados durante el semestre.

Berrondo, R., Cabrera, N., Franco, G., Frederico, M., Mariani, F., & Rodríguez, L. (2015). Criptografía: una cuestión de códigos.

Material Utilizado para repasar

Para repasar los temas para GitHub y Git ocupe un curso de Udemy adjunto el certificado y link del curso:

https://www.udemy.com/share/104exWAEYfeV1bQ3wJ/

Para repasar los conocimientos de Lenguaje C ocupe un curso de Udemy de Fundamentos de C adjunto link

https://www.udemy.com/share/101Kj6AEYfeV1bQ3wJ/

Para reforzar conocimiento ocupe apuntes y las practicas de las materias que cursamos adjunto links de las prácticas

http://odin.fi-b.unam.mx/salac/practicasFP/MADO-17_FP.pdf

http://lcp02.fi-b.unam.mx

Bibliografía:

Apuntadores. (s. f.). webdelprofesor.ula.ve. Consultado el 08 de junio del 2021, de http://webdelprofesor.ula.ve/ingenieria/gilberto/pr2/01_Apuntadores.pdf

Arreglos - Programacion Basica VB. (s. f.). programacionbasica. Consultado el 08 de junio del 2021, de https://sites.google.com/site/programacionbasicavb/arreglos

Christian León. (2014, 1 enero). Aplicación de aritmética de apuntadores - Programación en C [Vídeo]. YouTube. https://www.youtube.com/watch?v=fYaBCgiG8xA&feature=youtu.be

Facultad de ingeniería - UNAM. (2018, 6 abril). Manual de prácticas del laboratorio de Fundamentos de programación. Consultado el 08 de junio del 2021de: http://odin.fi-b.unam.mx/salac/practicasFP/MADO-17_FP.pdf

P. (s. f.). El código ASCII Completo. El código ASCII Completo. Consultado el 08 de junio del 2021, de https://elcodigoascii.com.ar

Pineda, C. I. C. N. I. E. (s. f.). Definiciones | Arreglos. Arreglos. Consultado el 08 de junio del 2021, de http://www.utn.edu.ec/reduca/programacion/arreglos/definiciones1.html#:%7E:text=Un%20arreglo%20es%20una%20estructura,utilizaci%C3%B3n%20individual%20de%20sus%20elementos.&text=Un%20arreglo%20es%20en%20resumen,finito%20y%20del%20mismo%20tipo.

Sánchez, A. I. (s. f.). 1an2l - Overview. GitHub. Consultado el 08 de junio del 2021, de https://github.com/1an2l?tab=repositories

Sudoku. (s. f.). Sudoku generador. Consultado el 08 de junio del 2021, de https://www.sudoku-online.org