



**UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO**



FACULTAD DE INGENIERIA

Estructuras de Datos y Algoritmos I

Actividad #1 “Repaso de Lenguaje C”

Alumno: García Gallegos Luis

Grupo:15

SEMESTRE 2021-2

Fecha de entrega 09/06/2021

Repaso de Lenguaje C

Definición de lenguaje de programación:

El lenguaje de programación es la combinación de símbolos y reglas, los cuales, permiten la creación de programas, mediante estos programas se pueden realizar o resolver problemas desde una computadora de manera eficiente. Existen tres tipos distintos:

- Lenguaje de maquina: Las instrucciones son directamente entendible por la computadora y no necesitan traductor para que la CPU pueda entender y ejecutar el programa. Utiliza un código binario (0 y 1), se basa en bits.
- Lenguaje de bajo nivel: Las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos.
- Lenguaje de alto nivel: Es semejante al lenguaje humano (en general en inglés), lo que facilita la elaboración y comprensión del programa. Por ejemplo, Basic, Pascal, Cobol, Fortran, C, etcétera.

Definición de algoritmo:

Se le denomina algoritmo al conjunto de pasos ordenados y finitos, mediante los cuales podemos resolver una tarea o un problema específico. Los algoritmos son independientes del lenguaje de programación y de la computadora que se use para ejecutarlo.

Todo algoritmo debe ser:

- Precisión: los pasos a seguir en el algoritmo deben ser precisos
- Determinismo: el algoritmo, dado un conjunto de datos idénticos entrada, siempre debe arrojar los mismos resultados
- Finitud: el algoritmo siempre debe ser de longitud finita

Absolutamente toda actividad que realizamos la podemos interpretar como un algoritmo. Existen dos tipos de algoritmos, los que se desarrollan para ser ejecutados por una computadora, llamados algoritmos computacionales, y los

que realiza el ser humano, llamados algoritmos no computacionales. Por ejemplo, cambiar un neumático o calcular el área de un triángulo.

El algoritmo consta de tres secciones o módulos principales

- Módulo 1: representa la operación o acción que permite el ingreso de los datos del problema
- Módulo 2: representa la operación o conjunto de operaciones secuenciales cuyo objetivo es obtener la solución al problema
- Módulo 3: representa una operación o conjunto de operaciones que permiten comunicar los resultados alcanzados

Palabras reservadas:







Lenguaje C	Función
aba()	Calcula el valor absoluto
char	Tipo de dato carácter
case	Si se cumple un caso
default	Ninguna opción de la selectiva múltiple
type def	Crea un nuevo nombre de tipo para un tipo de dato ya definido
for	Estructura repetitiva (o de ciclo)
int	Tipo de dato entero
}	Fin del programa o bloque
do	Estructura repetitiva
printf	Imprime en pantalla
puts	Imprime una cadena
{	Inicio de un programa o un bloque
scanf	Lee una variable
gets	Lee una cadena de caracteres
clrscr	Borra el contenido de la pantalla
while	Estructura repetitiva
void	Valor nulo
main	Programa principal

sqrt	Calcula raíz cuadrada
float	Tipo de dato real
struct	Registro o estructura
return	Regresa valor a otra función
break	Terminar el caso
switch	Estructura selectiva múltiple
If	Estructura selectiva
else	La parte falsa de la selectiva

Diagramas de flujo

Un diagrama de flujo representa la esquematización gráfica de un algoritmo. Muestra gráficamente los pasos a seguir para alcanzar la solución. Su construcción correcta es muy importante porque a partir del mismo se escribe un programa en lenguaje de programación, además hace más fácil y directo el transpasar a lenguaje de programación

Símbolos

	Símbolo usado para marcar el inicio y el fin
	Símbolo usado para introducir datos de entrada. Expresa lectura.
	Símbolo utilizado para representar un proceso. En su interior se expresan asignaciones como operaciones aritméticas cambios de celdas en memoria, etc.
	Lo utilizado para expresar la dirección del flujo del diagrama
	Símbolo utilizado para expresar conexión dentro de una misma página
	Símbolo utilizado para expresar conexión entre diferentes

Algunas reglas son:

1. Todo diagrama de flujo debe tener un inicio y un fin
2. Las líneas utilizadas deben ser rectas verticales y horizontales
3. El diagrama de flujo debe ser construido de arriba hacia abajo y de izquierda a derecha
4. Estación utilizada en el diagrama debe ser independiente de lenguaje de programación
5. Conveniente poner comentarios que expresen ayuden a entender lo que hicimos
6. No puede llegar más de una línea un símbolo

Tipos de datos

Los diferentes objetos de información con los que un algoritmo o programa trabaja se conocen colectivamente como datos. Todos los datos tienen un tipo asociado con ellos; el tipo de un dato es el conjunto de valores que este puede tomar durante una variable. El tipo de dato asociado a una variable limita el conjunto de datos que puede almacenar, así como las operaciones aplicables sobre dicha variable limita el conjunto de datos que puede almacenar, así como las operaciones aplicables sobre esa variable. Por lo tanto, una variable que pertenece a un tipo de dato int no podrá almacenar datos de tipo char; tampoco se podrán realizar o calcular operaciones de otros tipos de datos.

Los datos que utilizan los algoritmos y programas los podemos clasificar en simples o compuestos, un dato simple es indivisible y no se puede descomponer, mientras que, y un dato compuesto está integrado por varios datos, por lo tanto, es posible descomponerlo.

Los tipos de datos predefinidos son: numéricos, lógicos, caracteres y cadenas. De las anteriores mencionadas, tan solo el tipo cadena es compuesto y las demás son los tipos de datos simples.

Hay cinco tipos básicos de lenguaje C: entero, coma flotante, coma flotante con doble precisión, carácter y sin valor.

- ***Datos numéricos***

Este tipo de datos solo se divide entre enteros y reales.

- **Tipos enteros:** Son aquellos números que no tienen fracciones o decimales. Pueden ser negativos o positivos y el rango es de -32,768 a 32,767. Aunque este rango puede variar de un compilador a otro.
- **Tipos reales o de coma flotante:** Los tipos de datos flotante contienen una coma decimal, tal como 3,1416, pueden ser negativos o positivos formando el subconjunto de los números reales.
- **Datos lógicos o booleanos:** Hay lenguajes que solo pueden tomar uno de dos valores: verdadero (true) o falso (false). En lenguaje C no existe el tipo lógico, pero se puede implementar con un número entero, por ejemplo, 0 es falso y cualquier número diferente de 0 es verdadero.
- **Caracteres:** El almacenamiento de caracteres en el interior de las computadoras se hace en “palabras” de 8 bits (1 byte). Esto representa valores enteros en el rango de -128 a +127, el lenguaje C proporciona el tipo unsigned char para representar valores de 0 a 255. Contiene letras mayúsculas, minúsculas y dígitos, cada uno de ellos ordenado en su forma natural, gracias a esto, podemos manejar datos de caracteres bien definidos.
Existe también el dato de tipo cadena, que es una sucesión de caracteres que se encuentran delimitados por comillas.
- **Tipo Void:** Son datos vacíos o sin valor. Por ejemplo, la función main no representa valor alguno, debido a que tampoco tiene parámetros. Debemos de tener cuidado ya que esta característica es propia de algunos compiladores.

Declaración de variables:

Todas las variables deben de ser declaradas antes de ser usadas. Cada variable, por lo tanto, tiene asociado un tipo, un nombre y un valor. No se admiten como identificadoras palabras reservadas del lenguaje de programación que se esté utilizando.

Las variables del mismo tipo pueden definirse con una definición múltiple, separándolas mediante “,”: `int x, y, z;`

Una variable puede declararse en cuatro lugares diferentes del algoritmo o programa:

- Fuera de todos los subprogramas o funciones (global).
- Dentro de un subprograma o función (local a la función).
- Dentro de un bloque enmarcado con llaves {} (local al bloque).
- Dentro de una instrucción, por ejemplo: for (int i=0; 10; i++).
- Como parámetro formal (local a la función).

Uso de #define: creación de macros:

El compilador C tiene un procesador incorporado. Si las líneas

```
#define LIMITE 100
```

```
#define PI 3.14159
```

Se encuentran en un archivo que está compilando, el preprocesador cambia primero todos los identificadores LIMITE por 100 y todos los PI por 3.14159, excepto los que estén en cadenas entre comillas. Una línea #define puede estar en cualquier parte del programa, pero debe empezar en la columna 1 y solo tendrá efecto en las líneas del archivo que le siguen.

Declaración archivos de cabecera o encabezado (librerías o bibliotecas)

Indican al compilador que en esta posición se incluyan las líneas de sentencias que están dentro del archivo que se declara; son archivos estándar proporcionados por el fabricante del compilador, y se suelen declarar funciones, variables y constantes que van a ser utilizadas por las sentencias que el programador va a manejar en las siguientes líneas del programa. Para llamar un archivo de inclusión o cabecera es necesario hacer uso de la directiva #include, la cual tiene la sintaxis:

```
#include nombre_ejemplo_cabecera.
```

Las comillas le dicen a C que busque primero en el directorio de trabajo actual el archivo de inclusión; si no lo encuentra busca entonces en el directorio

especificado en la línea de órdenes, y finalmente si aún no lo ha encontrado busca en el directorio estándar que se haya definido durante la instalación.

Bibliotecas o librerías más utilizadas en C:

stdio.h contiene declaraciones de rutinas entrada/salida.

math.h contiene declaraciones de funciones matemáticas.

string.h contiene funciones con cadenas.

conio.h consola y puerto de E/S.

ctype.h clasificadro de caracteres.

Declaración de variables globales y constantes.

Las constantes y variables son elementos que tienen el objetivo de identificar los datos que se utilizan en las operaciones y cálculos que realiza el programa y que se almacenan en la memoria de la computadora. Si se declaran en dicha parte, pueden ser usadas en todo el programa.

El programa principal o función principal main():

El programa principal contiene el flujo del programa llamando a las funciones necesarias para su funcionamiento. La función `main()` indica dónde empieza el programa, cuyo cuerpo principal es un conjunto de instrucciones delimitadas por dos llaves, una inmediatamente después de la declaración `main()`, “ { “, y otra que finaliza el estado, “ } “.

Ciclos o estructuras repetitivas

if: Estructura de control que se dirige a la computadora para ejecutar una o más instrucciones solamente si la condición es verdadera. Si la condición es falsa no realiza ninguna acción.

Se declara de la siguiente manera:

```
if (condición) {  
  
}
```


si/si-no o if/else: Estructura de control que dirige a la computadora para ejecutar una acción si la condición es verdadera, y otra acción en caso de que sea falsa. Las instrucciones deberán de ser diferentes en cada caso, ya que si fueran iguales no se requeriría una estructura selectiva, con la estructura secuencial se resolvería el problema.

Se declara de la siguiente manera:

```
if (condición) {  
  
} else {  
  
}
```

for: es una estructura algorítmica adecuada para utilizar en un ciclo que se ejecutará un número definido de veces en el lenguaje c se escribe como (i=0; i<n; i+m) n es el número de veces que se va a repetir el ciclo y m cómo tiene que avanzar el ciclo.

While: es de estructura adecuada para utilizar en un ciclo cuando no sabemos el número de veces que éste se va a repetir dicho número depende de las proposiciones dentro del ciclo.

La estructura while se distinguen dos partes:

Ciclo: conjunto de instrucciones que se ejecutarán repetidamente.

Condición de terminación: la evaluación de esta condición permite decidir cuándo finalizará el ciclo la condición se evalúa al inicio (mientras se cumpla la condición continuar el ciclo).

En lenguaje c se escribe de la siguiente manera:

```
While(condición){  
  
}
```

Do-While: Esta estructura le permite al programador especificar que se repita una acción en tanto cierta condición se a verdadera ; cuando sea esta falsa se sale del ciclo. La condición la revisa después del ciclo o bucle.

Existen algoritmos y programas que requiere que por lo menos se ejecute el ciclo una vez y al final se revise la condición. Es semejante al ciclo while como el do-while puede utilizarse cuando no se conoce de antemano el número de veces que se repite el ciclo.

Características:

- Siempre entra por lo menos una vez al ciclo, ya que la condición esta después del ciclo.
- Si la condición es verdadera entra de nuevo al ciclo y regresa a revisar la condición, hasta que esta sea falsa sale del bucle.
- Debe existir una instrucción dentro del ciclo que modifique la condición, de lo contrario se hace infinita.
- Si tiene más de una instrucción, necesita obligadamente del inicio- fin({- }).

Arreglos

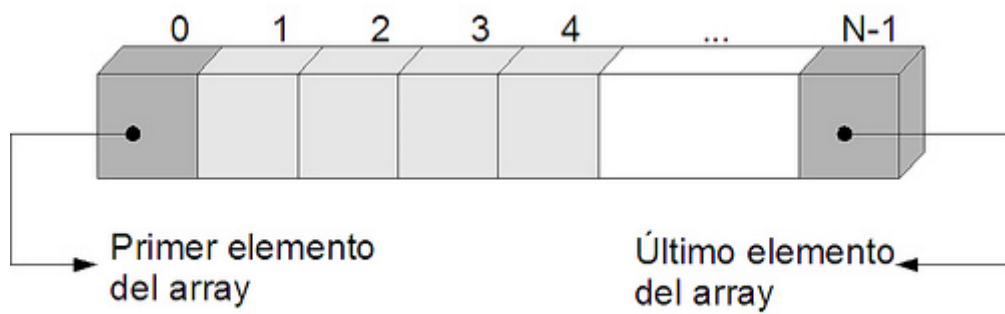
Arreglos unidimensionales: se define como una colección finita homogénea y ordenada de elementos.

Finita: todo arreglo tiene un límite, es decir qué debe estar bien definido.

Homogénea: todos los elementos de un arreglo son del mismo tipo.

Ordenada: se puede determinar cuál es el primer elemento el segundo el tercero etcétera.

el arreglo puede representarse gráficamente así:



Un arreglo tiene la característica de almacenar n elementos del mismo tipo y además permite el acceso a cada uno de ellos. Así se distinguen dos partes en los arreglos:

- Los componentes: son los elementos que componen o forman el arreglo
- Los índices: son los que permiten acceder a los componentes del arreglo en forma individual y el índice siempre inicia de cero en el lenguaje c

Para acceder a un elemento del arreglo se necesita el nombre del arreglo y el índice del elemento.

Se define de la siguiente manera un arreglo:

(Tipo de elementos) (Nombre el arreglo[número de elementos]); o

(Tipo de elementos) (Nombre el arreglo[número de elementos])={elementos};

Para poder recorrer todo el arreglo tanto como para leer datos e imprimir datos es necesario hacer lo siguiente:

```

3 #include <stdio.h>
4 #define p printf
5 #define s scanf
6 int main(){
7     int i, A[10];
8     p("\n\n\n\t ingresa los elementos al arreglo ");
9     for(i=0; i<10; i++){
10         p("\n\n\n\t Elemento [%d]", i+1);
11         s("%d", &A[i]);
12     }
13     system("cls");
14     p("\n\n\n\t Los numeros que ingresaste al arreglo son:");
15     for (i=0; i<10; i++){
16         p("\n\t %d", A[i]);
17     }
18 }

```

Todos los arreglos pueden ser combinados, con los ciclos mencionados anteriormente.

Arreglos Multidimensionales

Arreglo Bidimensional: Es un conjunto de datos homogéneos finito y ordenado, donde se hace referencia a cada elemento por medio de dos índices el primero de los índices se utiliza para indicar el renglón y el segundo para indicar la columna. Un arreglo bidimensional también puede definirse como un arreglo de arreglos.

Se declara un arreglo bidimensional en lenguaje C de la siguiente manera: (Tipo de dato) (Nombre del arreglo) [Número de elementos en filas] [Número de elementos en columnas]; o (Tipo de dato) (Nombre del arreglo) [Número de elementos en filas] [Número de elementos en columnas]={elementos};

Un arreglo bidimensional se lee y se imprime con dos for anidados como se muestra a continuación:

```
#include <stdio.h>

#define p printf
#define s scanf

int main(){

    int i, j, A[4][4];

    //lee el arreglo

    p("\n\n\n\t Ingrese los valores del arreglo bidimensional");

    for(i=0;i<4;i++)

        for(j=0;j<4;j++){

            p("\n\t Ingrese el valor [%d][%d]", i+1, j+1);

            s("%d", &A[i][j]);

        }

}
```

```

system("cls");

//imprime el arreglo

p("\n\n\n\t Los elementos del arreglo son \n");

for(i=0; i<4; i++){

    for(j=0; j<4; j++){

        p("%3d", A[i][j]);

    }

    p("\n");

}

}

```

Un ejemplo con el uso de arreglo bidimensional es el siguiente:

```

1  /*En un edificio de 4 pisos y de 4 departamentos por piso se hara una encuesta
2  para encontrar en que depto viven mas niños*/
3  #include <stdio.h>
4  #define p printf
5  #define s scanf
6  int main(){
7      int i, j, A[4][4], mayor, piso, depto;
8      p("\n\n\n\t Ingrese la informacion que le solicita");
9      for(i=0; i<4; i++){
10         for(j=0; j<4; j++){
11             p("\n\t En el piso %d, depto %d Cuantos ninos viven", i+1, j+1);
12             s("%d", &A[i][j]);
13         }
14         system("cls");
15         piso=0;
16         depto=0;
17         mayor=A[0][0];
18         for (i=0; i<4; i++)
19             for(j=0; j<4; j++)
20                 if (A[i][j]>mayor){
21                     mayor=A[i][j];
22                     piso=i;
23                     depto=j;
24                 }
25         p("\n\n\n\t Los niños que viven en el edificio son: \n");
26         for(i=3; i>=0; i--){
27             for(j=3; j>=0; j--){
28                 p("%3d",A[i][j]);
29             }

```

Funciones

Un recurso practico para el ahorro de líneas de código son las funciones, que sirven en ocasiones cuando necesitas repetir una acción en diversos puntos del pseudocódigo, se definen de la siguiente manera, primero se hace un prototipo (tipo de dato que regresa, nombre de la función, tipo de dato que se le envía o tipos de dato(Tipo de dato, tipo de dato)) para que el programa sepa que hay una función, después se pone el main y lo que necesite dentro de este para que pueda ser llamada a la función, y regrese el dato solicitado, la función solo puede regresar un solo dato. Ejemplo:

```
1  /*Programa que musltiplica 2 numeros con una función*/
2  #include <stdio.h>
3  float multi(float,int);
4  int main(){
5      float n1;
6      int n2;
7      float resultado;
8      printf("\n\n\t Dame 2 numeros para multiplicarlos: \n");
9      scanf("%f %d", &n1, &n2);
10     resultado=multi(n1,n2);
11     printf("Multiplicar %f x %d = a %f", n1, n2, resultado);
12     printf("\n\n\t Hasta luego");
13 }
14 float multi(float a,int b){
15     float producto;
16     producto=a*b;
17     return (producto);
18 }
```

Prototipos de funciones

Un prototipo es la declaración de una función que solo contiene la cabecera; a diferencia de la declaración completa, al final lleva punto y coma. El prototipo le avisa al compilador que existe una función que va a regresar un determinado tipo de dato y que parámetros utilizara.

Se declara de la siguiente manera: (Tipo de dato que regresara la función)
(nombre de la función) (Tipo o tipos de datos que se enviara);

Declaración de Funciones

Se declaran las funciones que utiliza el programa y que no están definidas en las librerías. Las funciones son un conjunto de sentencias que realizan una tarea

concreta y que retorna un dato como resultado de las operaciones que ha realizado. Las funciones se pueden mandar llamar varias veces al main.

Bibliografía

- Cairó Osvaldo. (2005). Metodología de la programación. México: Alfaomega.
- Corona Nakamura María Adriana y Ancona Valdez María de los Ángeles. (2011). Diseño de algoritmos y su codificación en lenguaje c. México: McGraw Hill.