1.- ¿Qué es un puntero y para que se utiliza? Un puntero es una variable que almacena la dirección de memoria de un objeto. Los punteros se usan ampliamente en C y C++ para tres propósitos principales:

Para asignar nuevos objetos en el montón,
Para pasar funciones a otras funciones
Para iterar sobre elementos en matrices u otras estructuras de datos.

2.- ¿Qué es un arreglo o array? Los arrays son variables estructuradas, donde cada elemento se almacena de forma consecutiva en memoria. Las cadenas de caracteres son declaradas en C como arrays de caracteres y permiten la utilización de un cierto número de notaciones y de funciones especiales.

Un array (unidimensional, también denominado vector) es una variable estructurada formada de un número "n" de variables simples del mismo tipo que son denominadas los componentes o elementos del array.

Un array en C puede tener una, dos o más dimensiones. Por ejemplo, un array de dos dimensiones también denominado matriz, es interpretado como un array (unidimensional) de dimensión "f" (número de filas), donde cada componente es un array (unidimensional) de dimensión "c" (número de columnas). Un array de dos dimensiones contiene, pues, "f*c" componentes.}

3.- ¿Qué tipo de variable son los arreglos? De estructura.

4.- ¿Cómo se obtiene el tamaño de un arreglo? Para obtener la longitud de un arreglo en C usamos el operador unario sizeof. Este operador devuelve el tamaño (en bytes) que determinada variable ocupa. Con este operador y algunas divisiones podremos obtener la longitud de un arreglo en C.

5.- ¿Qué información nos entrega el ampersand en un puntero? El ampersand (&) en C y C++ se utiliza para obtener la dirección de memoria de una variable. Cuando se utiliza antes de un nombre de variable, devuelve la dirección de memoria de esa variable en forma de puntero. Esta dirección de memoria es la ubicación física en la que se almacena la variable en la memoria del ordenador. Algunos usos comunes del ampersand en un puntero incluyen:

Pasar una variable por referencia a una función.
Asignar una dirección de memoria a un puntero.
Obtener la dirección de memoria de una variable para su manipulación directa.

Es importante tener en cuenta que el uso del ampersand depende del contexto en el que se emplee, ya que puede tener diferentes significados en diferentes situaciones dentro del lenguaje de programación.

6.- ¿Qué información nos entrega el símbolo asterisco en un puntero? El símbolo asterisco (*) en C y C++ se utiliza para declarar un puntero o para des referenciar un puntero existente.

Declaración de punteros:
Al utilizar el asterisco al declarar una variable, se indica que la variable es un puntero que apunta a una dirección de memoria específica.

Por ejemplo, al declarar int *ptr;, se está declarando un puntero llamado "ptr" que apunta a un entero.

Des referenciación de punteros:

Al utilizar el asterisco antes de un puntero existente, se accede al valor al que apunta el puntero, en lugar de a la dirección de memoria.

Por ejemplo, si se tiene int *ptr; y se desea acceder al valor al que apunta ptr, se utiliza *ptr.

En resumen, el asterisco se utiliza para declarar punteros y para acceder al valor al que apunta un puntero. Esto es fundamental para la manipulación de datos a través de punteros en C y C++.

7.- Escribe las definiciones y diferencias existentes entre los tipos de datos booleano, entero y carácter.
Booleano: El tipo de dato booleano representa un valor lógico que puede ser verdadero o falso. En la mayoría de los lenguajes de programación, se suelen representar como "true" (verdadero) o "false" (falso).

Diferencias: Los valores booleanos son utilizados para representar condiciones lógicas, como si algo es verdadero o falso. Por lo general, ocupan menos espacio en memoria que otros tipos de datos, ya que solo necesitan un bit para ser almacenados.

Entero: El tipo de dato entero representa números enteros, es decir, aquellos que no tienen parte decimal. Pueden ser positivos, negativos o cero, y su rango de valores depende del número de bits utilizados para representarlos (por ejemplo, un entero de 32 bits puede representar valores de -2,147,483,648 a 2,147,483,647 en lenguaje Java).

Diferencias: Los enteros se utilizan para representar cantidades enteras, como conteos, índices, identificadores únicos, entre otros. Pueden ocupar más o menos espacio en memoria dependiendo del rango de valores que necesiten representar.

Carácter: El tipo de dato carácter representa un solo carácter alfanumérico, como una letra, número o símbolo. En la mayoría de los lenguajes de programación, se representan entre comillas simples ('a', 'b', '1', '&', etc.).

Diferencias: A diferencia de los booleanos y enteros, los caracteres están orientados a representar símbolos individuales, como letras del alfabeto, dígitos numéricos, signos de puntuación, entre otros. En muchos lenguajes, los caracteres también tienen una representación numérica según el estándar ASCII o Unicode.

8.- ¿Qué tipo de dato puede almacenar más de 1 carácter? El tipo de dato que puede almacenar más de un carácter es el tipo de dato "cadena de caracteres" o "string" en muchos lenguajes de programación. Este tipo de dato se utiliza para almacenar secuencias de caracteres, como palabras, frases o texto completo. Los strings pueden contener cualquier combinación de letras, números, espacios y símbolos, y su longitud puede variar desde una sola letra hasta miles

de caracteres, dependiendo de la implementación del lenguaje de programación y de la memoria disponible.

Los strings son muy utilizados en programación para representar información textual, como nombres, direcciones, mensajes, entre otros. Además, los lenguajes de programación suelen proporcionar una amplia gama de funciones y operaciones para manipular cadenas de caracteres, como concatenación, búsqueda, reemplazo, entre otras.

9.- Las palabras reservadas "using namespace std;" para que nos sirven en C++. El espacio de nombres "std" (abreviatura de standard) es donde se encuentran definidas las clases, objetos y funciones estándar del lenguaje, como por ejemplo "cout", "cin", "string", "vector", entre otros. Para utilizar estas entidades en un programa, normalmente se tendría que escribir el nombre completo con el espacio de nombres delante, por ejemplo "std::cout" o "std::cin". Al utilizar la declaración "using namespace std;", se le está indicando al compilador que se desea utilizar todas las entidades del espacio de nombres "std" de forma abreviada, es decir, ya no es necesario preceder cada entidad con "std::". Esto hace que el código sea más conciso y legible, especialmente en programas pequeños o ejemplos educativos. Es importante señalar que aunque usar "using namespace std;" puede hacer que el código sea más fácil de escribir, en proyectos más grandes o en contextos donde la colisión de nombres es un problema, es recomendable evitar su uso y en su lugar especificar explícitamente las entidades que se van a utilizar, por ejemplo utilizando "using std::cout;" para solo importar la entidad que se necesita.

10.- ¿Cuál es la definición formal de la palabra reservada If-Else?. En términos formales, la estructura "if-else" se conoce como una declaración condicional. Esta estructura permite ejecutar un bloque de código si una condición es verdadera (evaluada como "true"), y un bloque alternativo si la condición es falsa (evaluada como "false").

11.- ¿Cuántos y cuáles ciclos conoces? Ventajas y desventajas. Existen varios tipos de ciclos, también conocidos como bucles, que permiten ejecutar un bloque de código repetidamente.
Ciclo while:
Ventajas: Es útil cuando la cantidad de repeticiones no es conocida de antemano, ya que se ejecuta mientras una condición sea verdadera.

Desventajas: Existe el riesgo de entrar en un bucle infinito si la condición no se modifica correctamente dentro del bucle.

Ciclo do-while:
Ventajas: Garantiza que el bloque de código se ejecute al menos una vez, ya que la condición se evalúa al final del bucle.

Desventajas: Puede resultar en una ejecución adicional del bloque de código si la condición es verdadera desde el principio.

Ciclo for:
Ventajas: Es útil cuando se conoce de antemano la cantidad de repeticiones, ya que tiene una estructura más compacta que el ciclo while. Permite inicializar, verificar y actualizar una variable de control en una sola línea.

Desventajas: Puede resultar en un código menos legible si se abusa de su uso en situaciones donde no es la opción más clara.

Ciclo foreach (en algunos lenguajes):

Ventajas: Es útil para recorrer colecciones de elementos, como arreglos o listas, de una manera más sencilla y legible.

Desventajas: No es adecuado para todos los tipos de iteraciones, especialmente cuando se requiere un control más detallado del índice o la posición.

12.- Escribe la sintaxis de un Switch case.

```
switch (expresion) {
  case valor1:
    // Código a ejecutar si la expresion es igual a valor1
    break;
  case valor2:
    // Código a ejecutar si la expresion es igual a valor2
    break;
  case valor3:
    // Código a ejecutar si la expresion es igual a valor3
    break;
  // ...
  default:
    // Código a ejecutar si la expresion no coincide con ningún caso
}
```

En esta estructura:
Se evalúa la expresión entre paréntesis que sigue a la palabra "switch".
Se comparan los resultados de la expresión con los valores de los casos (case).
Si se encuentra un caso que coincida con el valor de la expresión, se ejecuta el bloque de código correspondiente a ese caso.
La palabra clave "break" se utiliza para salir del bloque switch-case después de ejecutar el código correspondiente a un caso. Si no se incluye "break", el programa continuará ejecutando los bloques de código de los casos siguientes, lo que puede no ser deseado.
Si ninguno de los casos coincide con el valor de la expresión, se ejecutará el bloque de código asociado a "default" (opcional). El bloque switch-case es una estructura de control que permite tomar decisiones basadas en el valor de una expresión, ofreciendo una alternativa más clara y concisa en comparación con múltiples estructuras if-else anidadas.

13.- ¿Qué es un IDE? Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) es una aplicación que proporciona herramientas integrales para el desarrollo de software. Algunas de las características comunes de un IDE son:

Editor de código: Permite escribir, editar y formatear el código fuente de manera efectiva.
Compilador/Interprete: Facilita la compilación y ejecución del código.
Depurador: Permite detectar y corregir errores en el código.
Administrador de archivos: Organiza y gestiona los archivos relacionados con el proyecto.
Herramientas de construcción y automatización: Ayuda a compilar, construir y probar el software de manera eficiente.
Integración con control de versiones: Permite gestionar las versiones del código fuente utilizando sistemas de control de versiones como Git o SVN.
Otras herramientas: Algunos IDEs ofrecen soporte para pruebas unitarias, generación de documentación, análisis estático de código, entre otros.

Los IDEs son utilizados por programadores para aumentar la productividad al proporcionar un entorno unificado que incluye todas las herramientas necesarias para el desarrollo de software. Además, suelen estar diseñados para trabajar con uno o varios lenguajes de programación específicos, lo que facilita el desarrollo de aplicaciones en esos lenguajes.

14.- La programación estructurada minimiza la complejidad de los programas y reduce la cantidad de errores. A) Verdadero.

15.- Escoja las fases que se consideran para el desarrollo de un programa.
C) Estudio del problema
E) Elaborar diagrama de flujo
A) Desarrollo de algoritmos de solución
D) Arreglos

16.- La metodología de programación es un conjunto de fases o etapas orientadas a resolver un problema específico. A) Verdadero.

17.- El algoritmo que provee de una información mínima es:
A) General.

18.- Un algoritmo es una secuencia ordenada de pasos o: Indicaciones.

19.- Los tipos de algoritmos son:
A) Bifurcación o decisión
B) Repetición o interacción
E) Lineales

20.- Los diagramas de flujo son representaciones gráficas de los: Algoritmos.