IntroducciÃ³n al lenguaje de programaciÃ³n C++

WebQuest Description: Un lenguaje de programaciÃ³n es aquella estructura que, con una cierta base sintÃ¡ctica y semÃ¡ntica, imparte distintas instrucciones a un programa de computadora.

A la hora de establecer el origen del lenguaje de programaciÃ³n tenemos que hacer referencia, sin lugar a dudas, a Ada Lovelace que estÃ¡ considerada como la primera programadora de computadoras conocida en todo el mundo. De ahÃ-, curiosamente que se hablara en su honor del lenguaje de programaciÃ³n Ada. Y es que dicha figura llevÃ³ a cabo no sÃ³lo la manipulaciÃ³n de una serie de sÃ-mbolos para una mÃ¡quina del cientÃ-fico britÃ¡nico Charles Babbage sino tambiÃ©n la consecuciÃ³n del establecimiento de las instrucciones necesarias para que un computador pudiera realizar una serie de cÃ¡lculos iniciales.

Dentro de lo que es el lenguaje de programaciÃ³n es muy importante subrayar que los profesionales que se dedican a desarrollar este trabajan con un conjunto de elementos que son los que dan forma y sentido al mismo, los que permiten que aquellos funcionen y logren sus objetivos. Entre los mismos se encontrarÃ-an, por ejemplo, las variables, los vectores, los bucles, los condicionantes, la sintaxis o la semÃ¡ntica estÃ¡tica.

Las secuencias de programaciÃ³n para las acciones mÃ¡s usuales fueron asociadas para ser denominadas con nombres fÃ¡ciles de memorizar (como ADD o MUL). Al conjunto de instrucciones se lo denomina lenguaje ensamblador.
Grade Level: 9-12
Curriculum: Technology
Keywords: Ada Lovelace,
ADD o MUL,
Charles Babbage,
lenguaje,
sintaxis.
Published On:
Last Modified: 2018-09-12 20:57:52
WebQuest URL: http://zunal.com/webquest.php?w=388485

Introduction

en este WebQuest podrÃ¡s empaparte con todo lo relacionado al lenguaje C++ ...El lenguaje C es bastante cuestionado por los nuevos estudiantes hoy en dÃ-a, muchos no entienden el por quÃ© su aprendizaje, lo ven innecesario debido a que hay otros mÃ¡s fÃ¡ciles de aprender, otros con los que se puede lograr cosas mÃ¡s vistosas, como un gran juego, por ejemplo. AdemÃ¡s de que C no es un lenguaje orientado a objetos, llega esto a pensar â€œÂ¿para quÃ© aprenderlo si no lo vamos a usar?â€• y es que esto quizÃ¡ tenga al de sentido, por lo general las personas que estudian programaciÃ³n tienen diferentes objetivos, donde C, no es el lenguaje mÃ¡s conveniente de usar. Pero para ver la utilidad e importancia del mismo, debemos de verlo desde un punto de vista mÃ¡s tÃ©cnico.

Tasks

Primero te mencionamosÂ algunos errores tÃ-picos que se cometen al momento de escribir en lenguaje C++ para que no los cometas y tengas cuidado de identificarlos:Error de sintaxis: Estos errores son producidos, cuando se hace mal uso de las reglas del lenguaje de programaciÃ³n, y se violan las normas de sintaxis del lenguaje en C; estos errores son fÃ¡ciles de detectar por que generalmente es el compilador, que los identifica (Y hasta muestra la lÃ-nea donde se encuentra dicho error, pero eso depende de la versiÃ³n del compilador que estemos usando).Â Errores de ejecuciÃ³n: Estos errores se producen cuando le indicamos a la computadora realizar una determinada acciÃ³n, y esta la comprende, pero no puede ejecutarla. Por ejemplo: indicarle a la computadora una divisiÃ³n entre cero, sumar dos variables a las cuales no se les ha signado valor alguno, etc. Errores de LÃ³gica: Cuando estamos programando, el compilador no nos indica errores de sintaxis, ni de lÃ³gica; pero el resultado de nuestro programa, esta fuera del rango esperado, esto es producto de un error de lÃ³gica en el cÃ³digo de nuestro programa. Este tipo de errores son muy difÃ-ciles de identificar y por supuesto de corregir, ya que generalmente hay que revisar lÃ-nea por lÃ-nea de nuestro programa. Ejemplo: El sueldo negativo de un empleado, etc. Ahora que sabes los errores principales puedes ver su estructuraLa estructura de un programa en C, consta de algunas partes esenciales las cuales son uno o mÃ¡s mÃ³dulos llamadas funciones, siendo main() la primera funciÃ³n que es llamada cuando empieza la ejecuciÃ³n del programa . Cada funciÃ³n debe contener : - Directivas de pre-procesador (instrucciones que se le dan al compilador #include antes de compilar) #define Ejemplo: #include   Lo que se le esta indicando, es que de las librerÃ-as, "Incluya" en nuestro programa, la cual contiene las funciones de entrada y salida de datos. Si necesitamos las funciones matemÃ¡ticas, debemos especificarlo con la declaratoria: #include   Si necesitamos las funciones de cadenas: #include   Es necesario aclarar que esto se hace al inicio del programa, y las declaratorias deben llevar el sÃ-mbolo de numeral (#) seguido de la sentencia "include", y entre signos de mayor y menor que () el nombre de la directiva. -Â Declaraciones Globales pueden ser: *Prototipos de Funciones: TambiÃ©n llamadas declaraciones de funciones, lo cual se tratarÃ¡ mÃ¡s adelante *Declaraciones de Variables cabe destacar, que esto se hace seguido de los #include y los #define. -Â FunciÃ³n Principal main() Esta es la funciÃ³n principal de nuestro programa, su cuerpo, por ello NUNCA debe faltar, ya que en ella van contenidas todas las instrucciones de nuestro programa. main() { declaraciones locales /*Comentarios */ sentencias } la funciÃ³n main() va al inicio, luego abrimos llaves y dentro de ellas van las declaraciones de variables, las sentencias de lectura, cÃ¡lculos, asignaciones e impresiones, y con la Ãºltima llave (}), le indicamos el final del programa. Ejemplo:Â Â Programa que a partir del radio, calcula el Ã¡rea de un circulo #include #include main() { float radio, area; Â Â Â printf("Radio=\n"); Â Â Â Â Â scanf("%f", &radio); Â Â Â Â Â Â Â area=3.14159*radio*radio Area es %f\n\n", area); getch(); return 0; } ExplicaciÃ³n: Le indicamos al compilador, que usaremos las bibliotecas y , Â¿por quÃ© ?,

por que esta biblioteca, contiene las funciones getche(), getch(), etc, y de una de ellas hacemos uso en este pequeÃ±o ejemplo. Luego, le indicamos a nuestro programa el incio de nuestro programa (funciÃ³n main()). Declaramos, como valores reales, las variables radio y area (de esto se hablarÃ¡ mÃ¡s adelante). Luego, con la instrucciÃ³n printf(), mostramos en pantalla el mensaje (Radio=) y scanf se encarga de leer el valor digitado por el usuario. Posteriormente area, es igual al la multiplicaciÃ³n de pi (3.14159), el radio al cuadrado. Se muestra en pantalla ese resultado, luego el programa espera que se presiones cualquier tecla (getch()) y no retorna ningÃºn valor (return 0).

Process

CaracterÃ-sticas del Lenguaje C++ 1.-Tiene un conjunto completo de instrucciones de control. 2.-Permite la agrupaciÃ³n de instrucciones. 3.-Incluye el concepto de puntero (variable que contiene la direcciÃ³n de otra variable). 4.-Los argumentos de las funciones se transfieren por su valor. 5.- E/S no forma parte del lenguaje, sino que se proporciona a travÃ©s de una biblioteca de funciones. Permite la separaciÃ³n de un programa en mÃ³dulos que admiten compilaciÃ³n independiente. Originalmente el Lenguaje C estuvo muy ligado al sistema operativo UNIX como se habÃ-a mencionado antes que, en su mayor parte, estÃ¡ escrito en C. MÃ¡s adelante se comenzÃ³ a utilizar en otros sistemas operativos para programar editores, compiladores, etc. Sintaxis de Algunos Elementos de Un Programa en C Como su nombre lo indica, estos son los nombres, con los que identificamos las variables, constantes, funciones, vectores, etc, de nuestro programa. Para ello debemos tener presente algunas reglas: -pueden tener de 1 hasta un mÃ¡ximo de 31 caracteres -Debe de iniciar con una letra o subrayado Ejemplo: (Correctos) c2 _c2 (Incorrectos) 2c 2 c -No es lo mismo una minÃºscula que una mayÃºscula, ya que c distingue de entre ellas. Ejemplo: BETA Â¹ Beta Â¹ beta Â¹ BeTa -No son vÃ¡lidos los identificadores de palabras reservadas. En un inicio hablamos que c posee 32 palabras reservadas, entre ellas estÃ¡n: float char while int else return Estas palabras no pueden ser utilizadas para identificar variables, constantes, funciones etc b. identificadores: En todo programa que estemos diseÃ±ando en C (o en cualquier otro lenguaje de programaciÃ³n); es necesario insertar ciertos comentarios en el cÃ³digo, para que en posteriores modificaciones y cuando se realice el mantenimiento, podamos recordar cosas importantes ya que, en los comentarios, podemos incluir aspectos importantes del programas, explicaciones del funcionamiento de las sentencias, etc. El formato de los comentarios en C, es el siguiente: /*este es un comentario en C */ /*Podemos colocar mucha informaciÃ³n importante de nuestro Programa */ c. Comentarios Permite que, el pre-procesador, incluya funciones proporcionadas por el fabricante, a nuestro programa. Ejemplo: #include /* le decimos al compilador que incluya la librerÃ-a stdio.h */ d. La Directiva #include :permite definir constantes simbÃ³licas. Pero hasta ahora ha sido poco lo que hemos hablado acerca de las constantes, es por ello que en aprovechando, este especio; dedicarÃ© unas cuantas lÃ-neas para aclarar ello. Las variables pueden cambiar de valor, durante la ejecuciÃ³n del programa, por eso es que se llaman variables. Y las constantes como su nombre lo indica, son valores que permanecen constantes durante toda la ejecuciÃ³n del programa, un ejemplo de ello, es el valor de p (pi) que equivale a 3.14159.... En C existen diferentes tipos de variables, entre ellas tenemos: 1. Constates NumÃ©ricas: Son valores numÃ©ricos, enteros o de reales (de punto flotante). Se permiten tambiÃ©n constantes octales y hexadecimales. 2. Constantes SimbÃ³licas: las constantes simbÃ³licas tiene un nombre (identificador), y en esto se parecen las variables. Sin embargo, no pueden cambiar de valor a lo largo de la ejecuciÃ³n del programa. En C, se pueden definir mediante el preprocesador. (Tomado del Manual "Aprenda Lenguaje ANSI C como si estuviera en Primero" Escuela superior de Ingenieros Industriales. Universidad de Navarra. Febrero de 1998). Ejemplo: #define N 100 #define PI 3.1416 #define B 45 Esta directiva (#define) va, inmediatamente despuÃ©s de los #include. Se escribe la directiva, se deja un espacio y se escribe el identificador de la constante, otro espacio y su valor. e. la directiva #define / ! % ^ & * () - + {} [] \ ; : <> Â¿ . Signos de PuntuaciÃ³n y de SeparaciÃ³n Al momento de programar en C, esta es una regla de oro, y la causa por la cual nuestro programa puede darnos muchos errores de sintaxis, cuando se omite, al final de cada sentencia un punto y coma (;). Ya que con ello le indicamos al compilador que ha finalizado una sentencia. NOTA: el lector no debe confundirse, las directivas: #include, #define. Main(), no llevan punto y coma, por que no son sentencias. Recordemos el ejemplo 1.1, y vea que al final de cada sentencia lleva su correspondiente punto y coma: #include #include main() { float radio, area; printf("Radio=\n"); scanf("%f", &radio); area=3.14159*radio*radio; printf("El Area es %f\n\n", area); getch(); return 0; } g. Todas las Instrucciones o sentencias del programa terminan con un punto y coma (;) Esta consideraciÃ³n toma mayor auge, cuando veamos las instrucciones anidadas en condiciones, ciclos, etc. Ejemplo: { ... printf("Hola\n\b"); ... } h. Todo Bloque de Instrucciones debe ir entre llaves i. En una lÃ-nea se pueden escribir mÃ¡s de una instrucciÃ³n separada por un punto y coma Esto es posibles, por que con el punto y coma, le estamos indicando al compilador el fin de una sentencia o instrucciÃ³n. Ejemplo: b = c + d; d = 2*k;Tipos de Datos en CUn tipo de dato, se define como un conjunto de valores que puede tener una variables, junto con ciertas operaciones que se pueden realizar con ellastres tipos de datos bÃ¡sicos: - NÃºmeros enteros definidos con la palabra clave int - Letras o caracteres definidos con la palabra clave char - NÃºmeros reales o en coma flotante definidos con las palabras claves float o double Enteros Se definen con â€œintâ€• y admiten de forma opcional dos prefijos modificadores: â€œshortâ€• y â€œlongâ€•: Modifica el tamaÃ±o en bits del entero. Existen por tanto tipos de enteros: â€œintâ€•, â€œshort intâ€• (que se puede abreviar como â€œshortâ€•), y â€œlong intâ€• (que se puede abreviar como â€œ define tamaÃ±os fijos para sus tipos de datos bÃ¡sicos. Lo Ãºnico que garantiza es que un short int tiene un tamaÃ±o menor o igual que un int y este a su vez un tamaÃ±o menor o igual a un long int. Esta caracterÃ-stica del lenguaje ha complicado la creaciÃ³n de programas que sean compatibles entre varias plataformas. â€œunsignedâ€•: define un nÃºmero natural (mayor o igual a cero).Letras y cadenas Las variables de tipo letra se declaran como â€œcharâ€•. Para referirse a una letra se rodea de comillas simples: 'M'. Como las letras se representan internamente como nÃºmeros, el lenguaje C permite realizar operaciones aritmÃ©ticas como 'M' + 25. Las cadenas de texto o strings son simplemente tablas de â€œcharâ€•. Las funciones de biblioteca para manipular estas cadenas asumen que el Ãºltimo byte tiene valor cero. Las cadenas de texto se escriben en el programa rodeadas de dobles comillas y contienen el valor cero al final. NÃºmeros reales Los nÃºmeros reales se definen con â€œfloatâ€• o â€œdoubleâ€•. La diferencia entre ambas es la precisiÃ³n ofrece su representaciÃ³n interna. Hay un nÃºmero infinito de reales, pero se representan con un nÃºmero finito de bits. A mayor nÃºmero de bits, mayor nÃºmero de reales se representan, y por tanto, mayor precisiÃ³n. Los reales definidos con â€œdoubleâ€• tienen un tamaÃ±o doble a los definidos con â€œfloatâ€•. Al igual que en el caso de los enteros, el tamaÃ±o de estas representaciones varÃ-a de una plataforma a otra. Algunas plataformas ofrecen nÃºmeros reales con tamaÃ±o mayor al â€œdoubleâ€• que se definen como â€œlong double tamaÃ±os tÃ-picos para los tipos â€œfloatâ€•, â€œdoubleâ€• y â€œlong doubleâ€• son 4, 8 y 12 bytes respectivamente.Tablas Las tablas en prÃ¡cticamente idÃ©nticas a las de Java, con el tamaÃ±o entre corchetes a continuaciÃ³n del nombre. Al igual que en Java, los Ã-ndices de la tabla comienzan por cero.Los elementos de la tabla se acceden con el nombre de la tabla seguido del Ã-ndice entre corchetes. Una de las diferencias entre C y Java es que el acceso a una tabla en C no se verifica. Cuando se ejecuta un programa en Java si se accede a una tabla con un Ã-ndice incorrecto, se genera una excepciÃ³n de tipo â€œArrayIndexOutOfBoundsâ€•. Estas comprobaciones no se hacen nunca en C (a no ser que se escriban explÃ-citamente en el programa). Si se accede a una tabla con un Ã-ndice incorrecto se manipulan datos en una zona de memoria incorrecta y el programa continua su ejecuciÃ³n. Tras este acceso incorrecto pueden suceder dos cosas. La primera es que la memoria a la que ha accedido por error estÃ© fuera de los lÃ-mites del programa. En este

caso la ejecuciÃ³n termina de manera abrupta y en el intÃ©rprete de comandos se muestra el mensaje â€œsegmentation faultâ€•. La otra posibilidad es que se acceda a otro lugar dentro de los datos del programa. Esta situaciÃ³n seguramente producirÃ¡ un error cuyos sÃntomas sean difÃciles de relacionar con el acceso incorrecto. Tablas de mÃºltiples dimensiones C permite la definiciÃ³n de tablas de mÃºltiples dimensiones escribiendo los diferentes tamaÃ±os rodeados de corchetes y concatenados. El acceso se realiza concatenando tantos Ãndices como sea preciso rodeados de corchetes. Al igual que en el caso de las tablas unidimensionales, no se realiza ningÃºn tipo de comprobaciÃ³n de los Ãndices cuando se accede a un elemento.TamaÃ±o de los tipos de datos bÃ¡sicos En C, el tamaÃ±o de los tipos de datos bÃ¡sicos puede variar de una plataforma a otra. Esta caracterÃstica estÃ¡ detrÃ¡s de buena parte de las crÃticas que recibe este lenguaje, pues de ella se derivan problemas de compatibilidad (una aplicaciÃ³n se comporta de forma diferente cuando se ejecuta en plataformas diferentes).DeclaraciÃ³n de Variables Una Variable, como su nombre lo indica, es capaz de almacenar diferentes valores durante la ejecuciÃ³n del programa, su valor varÃa. Es un lugar en la memoria el cual, posee un nombre (identificador), y un valor asociado. La declaraciÃ³n de variables en C, se hace en minÃºsculas. Formato: Tipo_de_dato nombre_de_la_variable; Ejemplos: *Declare una variable de tipo entero y otra de tipo real, una con el nombre de "x" y otra con el identificador "y": int x; float y; *Declare una variable de tipo entero llamada moon, e inicialÃcela con un valor de 20 int x = 20; *Declare una variable de tipo real, llamada Pi, e inicialÃcela con una valor de 3.1415 float pi=3.1415; *Declare una variable de tipo caracrter y asÃgnele el valor de "M" char car = â€™Mâ€™; *Declare una variable llamada nombre, que contenga su nombre: char nombre[7]="Manuel"; ExplicaciÃ³n: En el apartado anterior, se explicÃ³, que C, no tiene el tipo de dato llamado string, o mejor conocido como cadenas de texto, pero nosotros podemos hacer uso de ellas, por medio de un arreglo, (de lo cual hablaremos con mÃ¡s detalle, posteriormente); pero para declarar este tipo de datos colocamos el tipo de datos, es decir la palabra reservada char luego el nombre, e inmediatamente abrimos, entre corchetes, va el nÃºmero de letras, que contendrÃ¡ dicha variable. Es muy importante que al momento de declarar el tamaÃ±o, sea un nÃºmero mayor, al verdadero nÃºmero de letras; por ejemplo, la palabra "Manuel", solo tiene 6 letras, pero debemos declararlo para 7 letras Â¿Por quÃ©?. VeÃ¡moslo grÃ¡ficamente, en la memoria, C crea un variable llamada nombre y esta posee la palabra Manuel, asÃ: en realidad, hay 7 espacios, pero la cuanta llega hasta 6, por que c, toma la primera posiciÃ³n como la posiciÃ³n cero, y para indicar el final de la cadena lo hace con un espacio en blanco. DeclaraciÃ³n de Constantes Las constantes, como su nombre lo indica, son valores que se mantiene invariables durante la ejecuciÃ³n del programa. Su formato es el siguiente: const tipo_de_dato nombre= valor; donde const, es una palabra reservada, para indicarle al compilador que se esta declarando una constante. Ejemplo: const int dia=7; const float pi=3.14159; const char caracter= â€˜mâ€™; const char fecha[]="25 de diciembre"; Caso Especial Constantes SimbÃ³licas Las constantes simbÃ³licas, se declaran mediante la directiva #define, como se explicÃ³ anteriormente. Funcionan de la siguiente manera, cuando C, encuentra el sÃmbolo que representa a la constante, lo sustituye por su respectivo valor. Ejemplo: #define N 150 #define PI 3.1416 #define P 50 NOTA: El lector debe comprender algunas diferencias fundamentales entre la declaratoria const y #define; la primera, va dentro del programa, es decir, dentro de la funciÃ³n main() o alguna funciÃ³n definida por el usuario, mientras que #define va en el encabezado, despuÃ©s de los #include, por eso estas no llevan al final el punto y coma (;). Entrada y Salida Por Consola Entrada y Salida por consola: se refiere a las operaciones que se producen en el teclado y en la pantalla de la computadora. En C no hay palabras claves para realizar las acciones de Entrada/Salida, estas se hacen mediante el uso de las funciones de la biblioteca estÃ¡ndar (stadio.h). Para utilizar las funciones de E / S debemos incluir en el programa el archivo de cabecera stdio.h, mediante la declaratoria: #include   Las Funciones de E / S mÃ¡s simples son getchar() que lee un carÃ¡cter del teclado, espera un retorno de carro (Â¿), es decir un enter y el eco aparece. Es decir la tecla presionada. *putchar(): Imprime un carÃ¡cter en la pantalla, en la posiciÃ³n actual del cursor. Algunas variaciones: *getche(): Aparece el Eco *getch(): No aparece el eco estas instrucciones se encuentran en la biblioteca conio.h Ejemplos: Programa que espera que se presiona una tecla, la muestra en pantalla, y ademÃ¡s muestra el carÃ¡cter siguiente: Ejemplo : #include   #include   main() { char car; clrscr(); /*Se encarga de borrar la pantalla por eso se llama claer screen*/ car=getchar(); putchar(car+1); getch(); return 0; } Ejemplo 2: #include   #include   main() { char x; /*Declaramos x como caracter*/ printf("Para Finalizar Presione cualquier Tecla:"); x= getchar();/*Captura y muestra el caracter presionado*/ getch();/*Espera a que se presione cualquier otra tecla para finalizar*/ return 0; } Entrada y Salida de Cadenas Una Cadena, es una frase, compuesta por varias palabras. En C, podemos hacer uso de las cadenas, mediante, la sentencia: *gets(): Lee una cadena de carÃ¡cter introducido por el teclado. Se puede introducir caracteres hasta que se de un retorno de carro, (enter); el cual no es parte de la cadena; en su lugar se coloca un terminador nulo \0. *puts(): Imprime en pantalla, el argumento guardado en la variable que se manda a impresiÃ³n. Ejemplo: DiseÃ±e un programa en C, que lea su nombre; lo salude y mande a impresiÃ³n su nombre, usando gets e y puts #include   #include   main() { char nombre[40]; puts("digite su nombre:"); gets(nombre); puts("BIENVENIDO:"); puts(nombre); getch(); return 0; } Secuencias de Escapes Indica que debe ejecutar algo extraordinario.Secuencia   Valor \n   Nueva lÃnea (new line). El cursor pasa a la primera posiciÃ³n de la lÃnea siguiente \r   (carriage return). El cursor pasa a la primera posiciÃ³n de la lÃnea donde nos encontremos. \t   Tabulador. El cursor pasa a la siguiente posiciÃ³n de tabulaciÃ³n. \a   Alerta. Crea un aviso bien de forma visible o bien mediante sonido. \b   Espacio atrÃ¡s (backspace). Hace retroceder el cursor una posiciÃ³n a la izquierda.   \f   AlimentaciÃ³n de pÃ¡gina (form feed). Crea una nueva pÃ¡gina.\\   Muestra la barra invertida. \â€•   Muestra la comilla doble. \?   Muestra un interrogante. \nÃºmero_octal   Muestra el carÃ¡cter ASCII correspondiente segÃºn el nÃºmero octal que se haya especificado. \xnÃºmero_hexadecimal   Muestra el carÃ¡cter ASCII correspondiente segÃºn el nÃºmero hexadecimal que se haya especificado. \v   TabulaciÃ³n vertical. \'   ApÃ³strofo o comilla simple.Cadena de ControlLas cadenas de caracteres (tambiÃ©n llamadas cadenas o strings) son un tipo

particular de vectores, que como su nombre lo dice son vectores de char, con la particularidad que tienen una marca en el fin del mismo (el caracter '\0'), ademÃ¡s el lenguaje nos permite escribirlas como texto dentro de comillas dobles si son simples no. Ejemplo:char cadena_hola[]="Hola"; char otro_hola[]={'H','o','l','a','\0'}; // Igual al anterior char vector[]={'H','o','l','a'}; /* Un vector de 4 elementos,   con los elementos 'H','o','l' y 'a' */ char espacio_cadena[1024]="Una cadena en C"; char cadena_vacia[]="";Operadores lÃ³gicosEstos son los que nos permiten unir varias comparaciones: 10>5 y 6==6. Los operadores lÃ³gicos son: AND (&&), OR (||), NOT(!).Ejemplo:Operador ! (NOT, negaciÃ³n): Si la condiciÃ³n se cumple NOT hace que no se cumpla y viceversa. Operador || (OR, en castellano O): Devuelve un 1 si se cumple una de las dos condiciones. printf("Resultado: %i", (10==10 && 5>2); Operador && (AND, en castellano Y): Devuelve un 1 si se cumplen dos condiciones. Operadores AritmÃ©ticosLos operadores aritmÃ©ticos son aquellos que "manipulan" datos numÃ©ricos, tanto enteros como reales. Hay 2 tipos de operadores aritmÃ©ticos: unarios y binarios. Los operadores unarios se anteponen a la expresiÃ³n aritmÃ©tica, y son los operadores de signo. Los operadores binarios se sitÃºan entre 2 expresiones aritmÃ©ticas.Ejemplo:Operadores aritmÃ©ticos binariosadmiten expresiones enteras y reales a excepciÃ³n de div y mod, que sÃ³lo admiten expresiones enteras, por lo que devuelven expresiones enteras. En el caso de los otros operadores, si los 2 operados a los que afecta son enteros, la expresiÃ³n resultante serÃ¡ entera, pero si alguno o ambos son reales, la expresiÃ³n resultado es de tipo real.Operador   OperaciÃ³n mod   MÃ³dulo (resto)   div   Divisiã³n entera /   DivisiÃ³n   *   MultiplicaciÃ³n   -   Resta   +   Suma   Operadores aritmÃ©ticos unariosLos operadores unarios devuelven expresiones del mismo tipo que la expresiÃ³n a la que afectan. Operador   OperaciÃ³n -   Signo negativo +   Signo positivo Operadores Unarios:incluye una clase de operadores que actÃºan sobre un solo operador para producir un nuevo valor. Por eso el nombre de unarios, por que para poder funcionar solo necesitan de un operador. Operador   PropÃ³sito   Operador Decremento: Hace que su variable, constante o expresiÃ³n disminuya en uno-- Operador Incremento: Hace que la variable, constante o expresiÃ³n se aumente en uno. ++ Menos Unario: Es el signo menos que va delante de una variable, constante o expresiÃ³n. - Operadores asignaciÃ³n:Donde el identificador representa por lo general una variable y una constante, una variable o una expresiÃ³n mÃ¡s compleja. identificador = expresiÃ³n; Se utilizan en forma de expresiones de asignaciÃ³n en los que se asigna en el valor de una expresiÃ³n a un identificador. El operador de asignaciÃ³n mÃ¡s utilizado es "=" y su formato es: Los Operadores de AsignaciÃ³n, como su nombre lo indica, se encargan de atribuirle, asignarle, confinarle, etc a una variable, el resultado de una expresiÃ³n o el valor de otra variable. Ejemplo: *Un valor en coma flotante puede ser truncado, se asigna a un identificador entero.   *Un valor de doble precisiÃ³n puede ser redondeado si se asigna a un identificador de coma flotante.   En C, estÃ¡n permitidas las asignaciones mÃºltiples, asÃ: Identificador1 = identificador2 = identificador3.....= identificadorn=expresiÃ³n   C, posee ademÃ¡s los siguientes operadores de asignaciÃ³n: Operador ExplicaciÃ³n P%n. Equivale a: p=p%%= K/=m, equivale a: k=k/m /= J*=2. Equivale a: j=j*2 *= i-=1. equivale a: i=i-1 -= ExpresiÃ³n1+=expresiÃ³n2. Equivale a: expresiÃ³n1=expresiÃ³n1 + expresiÃ³n2 +=

Evaluation

No es necesario una evaluaciÃ³n enÂ este trabajo, por lo que continÃºe con el siguiente paso.

Category and Score					Score

Category and Score					Score
				Total Score	

Conclusion

-Los lenguajes de alto nivel se desarrollaron con el objetivo de ser mÃ¡s accesibles y entendibles por la mayorÃ-a de programadores, de manera que los programadores pudieran concentrarse mÃ¡s en resolver la tarea o los problemas y no en el lenguaje que la maquina tenÃ-a que entender. - C++ surge de fusionar dos ideas: la eficiencia del lenguaje C para poder acceder al hardware al ejecutar tareas que realmente demandaban recursos de memoria; y las ideas de abstracciÃ³n que representan las el nuevo conceptos de clases y objetos. - El lenguaje C++ presenta grandes herramientas de desarrollo para los programadores como las funciones, bibliotecas, clases y los objetos. De manera que el programador se ocupa de utilizar dichas herramientas para resolver un problema especÃ-fico. - El lenguaje C++ posee una serie de caracterÃ-sticas que lo hacen distinto del lenguaje C. Aunque es posible verlo como una simple extensiÃ³n del lenguaje C, en realidad implica un cambio en la forma de pensar por parte del programador.

Teacher Page

Standards

Credits

Other