Universidad Nacional Autónoma de

México

Facultad de Ingeniería

Asignatura: Estructura de datos y algoritmos I.

Actividad 2: Acordeón de C y lenguaje con la letra inicial del nombre.

Alumno: Carrasco Ruiz Alan Uriel.

Fecha (26/02/2021)

Parte de C

El lenguaje de programación C fue creado por Brian Kernighan y Dennis Ritchie a mediados de los años 70. Además, el lenguaje C no es rígido en la comprobación de tipos de datos, permitiendo fácilmente la conversión entre diferentes tipos de datos y la asignación entre tipos de datos diferentes.

El lenguaje C es un lenguaje estructurado, en el mismo sentido que lo son otros lenguajes de programación tales como el lenguaje Pascal, el Ada o el Modula-2, pero no es estructurado por bloques.

Mi acordeón contendrá las cosas básicas acerca de todo lo que contiene el lenguaje de programación C.

Todo programa de C consta, básicamente, de un conjunto de funciones, y una función llamada main, la cual es la primera que se ejecuta al comenzar el programa, llamándose desde ella al resto de funciones que compongan nuestro programa.

Posteriormente de este método principal puede contener cualquiera de las siguientes estructuras o funciones que se pueden utilizar.

Estructuras de secuencia están formadas por una secuencia de instrucciones que se ejecutan en orden una a continuación de la otra.

Cada una de las instrucciones están separadas por el carácter punto y coma (;). No obstante, en algunos casos nos interesará agrupar en un bloque una serie de instrucciones. El bloque de sentencias se define por el carácter llave de apertura ({) para marcar el inicio de este, y el carácter llave de cierre (}) para marcar el final.

{instrucción 1;
instrucción 2;
.....
instrucción N;}

Sin embargo, en caso de que el bloque de sentencias está constituido por una única sentencia no es obligatorio el uso de las llaves de apertura y cierre ({}).

Estructuras de control: Las estructuras condicionales controlan si una sentencia o bloque de sentencias se ejecutan, en función del cumplimiento o no de una condición o expresión lógica.

C tiene dos estructuras de control para la selección, if y switch.

Esta instrucción (If) hace que se ejecuten unas sentencias u otras dependiendo del valor que toma una condición. La instrucción if puede ser simple o doble.

Las instrucciones if-else se pueden anidar obteniéndose una estructura condicional múltiple; En este caso se evalúa la condición 1; si es cierta, se ejecuta la *instrucción 1* y se continúa por la *instrucción 6* después del bloque de if-else anidados. Si la condición 1 es falsa, se evalúa la condición 2, y así sucesivamente.

En caso de no ser cierta ninguna de las condiciones, la sentencia que se ejecuta es la del último else, es decir, la *instrucción 5*.

Switch En una instrucción switch, expresión debe ser una expresión con un valor entero, y constante 1, constante 2, ..., deben ser constantes enteras, constantes de tipo carácter o una expresión constante de valor entero.

La instrucción switch evalúa la expresión entre paréntesis y compara su valor con las constantes de cada case. Se ejecutarán las instrucciones de aquel case cuya constante coincida con el valor de la expresión, y continúa hasta el final del bloque o hasta una instrucción que transfiera el control fuera del bloque del switch (una instrucción break, o return).

Estructuras de repetición

While. - Ejecuta una instrucción o un bloque de instrucciones cero o más veces, dependiendo del valor de la condición.

Se evalúa la condición, y si es cierta, se ejecuta la instrucción o bloque de instrucciones y se vuelve a evaluar la condición; pero si la condición es falsa, se pasa a ejecutar la siguiente instrucción después del while.

do-while. - Ejecuta una instrucción o un bloque de instrucciones, una o más veces, dependiendo del valor de la condición.
Se ejecuta la instrucción o bloque de instrucciones y a continuación se evalúa la condición. Si la condición es cierta, se vuelve a ejecutar la instrucción o bloque de instrucciones, y si es falsa, pasa a ejecutarse la siguiente instrucción después del do-while.
Cuando se utiliza una instrucción do-while el bloque de instrucciones se ejecuta al menos una vez, ya que la condición se evalúa al final.

For. -Un bucle for hace que una instrucción o bloque de instrucciones se repitan un número determinado de veces mientras se cumpla la condición.
Hablamos de ciclos anidados cuando se incluyen instrucciones for, while o do-while unas dentro de otras.

Funciones

sintaxis

tipo_función nombre_función (tipo y nombre de argumentos) {
bloque de sentencias
}

Puede ser de cualquier tipo de los que conocemos. El valor devuelto por la función será de este tipo. Por defecto, es decir, si no indicamos el tipo, la función devolverá un valor de tipo entero (int). Si no queremos que retorne ningún valor deberemos indicar el tipo vacío (void). Los argumentos de una función no son más que variables locales que reciben un valor. Este valor se lo enviamos al hacer la llamada a la función. Pueden existir funciones que no reciben argumentos.
Las funciones pueden ser llamadas desde la función main o desde otras funciones. Nunca se debe llamar a la función main desde otro lugar del programa.

Arreglos. - Los arreglos son estructuras de datos consistentes en un conjunto de datos del mismo tipo. Los arreglos tienen un tamaño que es la cantidad de objetos del mismo tipo que pueden almacenar. Los arreglos son entidades estáticas debida a que se declaran de un cierto tamaño y conservan todo a lo largo de la ejecución del programa en el cual fue declarado, esto es el espacio que se reserva en la memoria en la cual se apartara ese espacio para que los valores que se ingresen sean almacenados uno después de otro.
int arreglo1 [30]declara que arreglo1 es un arreglo que puede contener 30 enteros.
#define TAMANIO 100

Archivos. -Es un conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros que son de igual tipo y constan a su vez de diferentes entidades de nivel más bajos denominadas campos. Hay dos tipos de archivos, archivos de texto y archivos binarios. Un archivo de texto es una secuencia de caracteres organizadas en líneas terminadas por un carácter de nueva línea. Un archivo binario es una secuencia de bytes que tienen una correspondencia uno a uno con un dispositivo externo. Así que no tendrá lugar ninguna traducción de caracteres. Se puede accederá cada archivo y ya sea lo que necesitemos, agregar, crear nuevo, o escribir, esto tan solo con una letra diferente a la hora de que entremos al archivo dentro del programa.

Parte de ActionScript

Ada es un lenguaje de programación estructurado y fuertemente tipado de forma estática que fue diseñado por Jean Ichbiah de CII Honeywell Bull por encargo del Departamento de Defensa de los Estados Unidos. Es un lenguaje multipropósito, orientado a objetos y concurrente, pudiendo llegar desde la facilidad de Pascal hasta la flexibilidad de C++.

Ada se usa principalmente en entornos en los que se necesita una gran seguridad y fiabilidad como la defensa, la aeronáutica (Boeing o Airbus), la gestión del tráfico aéreo (como Indra en España) y la industria aeroespacial entre otros.
Un compilador de Ada muy usado es GNAT, originalmente desarrollado por la Universidad de Nueva York bajo patrocinio del DoD. Está basado en la tecnología de GCC y es software libre.

Características de Ada:

- La sintaxis, inspirada en Pascal, es bastante legible incluso para personas que no conozcan el lenguaje. Es un lenguaje que no escatima en la longitud de las palabras clave, en la filosofía de que un programa se escribe una vez, se modifica decenas de veces y se lee miles de veces (legibilidad es más importante que rapidez de escritura).
- Identificadores y palabras claves son equivalentes sea cual sea el uso de mayúsculas y minúsculas, es decir es un lenguaje *case-insensitive*.
- En este caso, todo el programa es un único procedimiento, que puede contener subprogramas (procedimientos o funciones) (en este caso: la función Ack).
- Cada sentencia se cierra con end *qué cerramos*. Es un modo de evitar errores y facilitar la lectura. No es necesario hacerlo en el caso de subprogramas, aunque todos los manuales lo aconsejan y casi todos los programadores de Ada lo hacen.
- El operador de asignación es: =, el de igualdad =. A los programadores de C y similares les puede confundir este rasgo inspirado en Pascal.
- La sintaxis de atributos predefinidos es Objeto'Atributo (o Tipo'Atributo) (nota: esto sólo aplica a atributos predefinidos por el lenguaje, ya que no es el concepto de atributo típico de OOP).
- Se distingue entre "procedimientos" (subrutinas que no devuelven ningún valor, pero pueden modificar sus parámetros) y "funciones" (subrutinas que devuelven un valor y no modifican los parámetros). Muchos lenguajes de

programación no hacen esta distinción. Las funciones de Ada favorecen la seguridad al reducir los posibles efectos colaterales, pues no pueden tener parámetros in out.

Tipos de variables:

El tipo Natural incluye a los enteros positivos y al 0.
El tipo Positive incluye a los enteros positivos sin el 0.
El tipo Boolean solo incluye dos valores: True y False.
Un tipo y sus subtipos son compatibles, por lo que pueden mezclarse en expresiones. Dos tipos diferentes siempre son incompatibles. Natural y Positive no son tipos propiamente, sino subtipos de Integer Al mezclar tipos y subtipos puede ocurrir que un resultado se salga del rango de valores permitidos para un subtipo, lo que provoca un error en tiempo de ejecución.

Punteros: Los punteros son un mecanismo presente en la mayoría de los lenguajes de programación para gestionar espacio en memoria de una forma indirecta. El uso de punteros es una herramienta potente pero peligrosa: cuando los punteros se usan incorrectamente se pueden producir erratas muy difíciles de detectar. Algunos lenguajes de programación llaman a los punteros con otros nombres. Así en Ada a los punteros se les llama accesos.

Estructuras de control: Las condiciones son expresiones lógicas: operaciones cuyo resultado es True o False. Las partes elsif ...y else son opcionales. Aunque en una de las alternativas haya varias sentencias NO hay que poner begin y end entre ellas. Las expresiones lógicas pueden usar los operadores and, or, xor, not, y ser tan complejas como se quiera. Los operadores lógicos son los de menor precedencia de todos, por eso en la expresión del ejemplo no son necesarios paréntesis (aunque pueden aumentar la legibilidad).

Archivos: El paquete Ada.Text_IO, además de permitir trabajar con la entrada/salida estándar (por defecto, teclado y ventana de terminal), sirve también para trabajar con ficheros de textos. Ada.Text_IO permite leer y escribir en ficheros de texto utilizando los mismos subprogramas que para la entrada/salida estándar, pero utilizando un primer parámetro adicional que representa el fichero que se lee o escribe.

Referencias:

https://informatica.uv.es/estguia/ATD/apuntes/laboratorio/Lenguaje-C.pdf

https://tecnologia.fandom.com/wiki/Ada_(Lenguaje_de_programaci%C3%B3n)