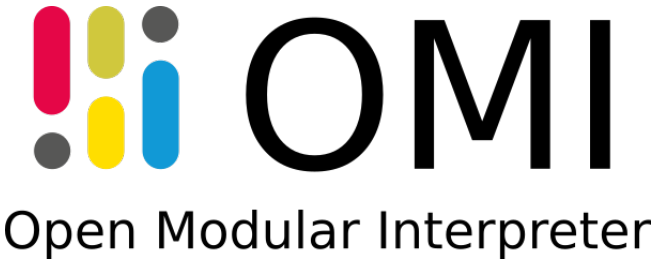

Requisitos funcionales

Fco. Javier Bohórquez Ogalla

Índice

1. Visión general

En esta sección se lleva a cabo un análisis de la situación actual en el estudio de la teoría de autómatas y los lenguajes formales. Se describirán las necesidades presentes en este campo, y que serán objetivo de las soluciones planteadas. A partir de estos objetivos se llevará acabo una descripción de los requisitos que debe cumplir la solución tomada.

2. Situación actual

El estudio de los lenguajes formales es anterior a la concepción de las computadoras. Las matemáticas, la lógica y otras ciencias venían haciendo uso de los conceptos de los lenguajes formales para la solución de problemas.

Una computadora desde un punto de vista teórico es un autómata o máquina de estados, que es capaz de ejecutar una serie de instrucciones descritas en lenguaje máquina. Por tanto fueron los avances en la rama de la teoría de autómata y los lenguajes formales, entre otros campos, los que permitieron la concepción de las primeras computadoras. Actualmente la mayoría de las ingenierías de la información se estudian los conceptos teóricos detrás de las computadoras y los lenguajes de programación.

Lo más común es que un estudiante de informática comience sus estudios en estos campos con los autómatas: los tipos que existen, cómo se definen y para qué se utilizan. Llegando a comprender conceptos como el de estado, alfabeto, etc. Incluso estudiando definiciones de algunos tipos de lenguajes formales como los lenguajes regulares.

Posteriormente el alumno podría utilizar los conceptos aprendidos para estudiar distintos modelos de computación, los cuales son definidos formalmente y desde un punto de vista teórico. La máquina de turing y el cálculo lambda son piezas esenciales en este punto del aprendizaje.

Una vez que se es poseedor del conocimiento base en la teoría de autómatas y los lenguajes formales, el alumno puede aplicar estos conocimientos para estudiar las estructuras, mecanismos y demás conceptos detrás de los lenguajes de programación. En este punto se estudian los interpretes y compiladores, y los conceptos básicos que hay detrás estos. Conceptos como el léxico, sintaxis y la semántica de los lenguajes de programación, las tablas de símbolos, o las distintos tipos de gramáticas.

Hasta aquí se habrá obtenido los conocimientos teóricos necesarios y el alumno podría comenzar a dar soluciones prácticas a problemas, aplicando así los conocimientos obtenidos. Así es común que se comience con el desarrollo de analizadores léxicos y sintácticos

sencillos y concretos, para luego aplicarles una semántica. Ejercicios como el desarrollo de una calculadora suelen ser habituales. Además se estudian algunas de las herramientas que asisten al proceso del desarrollo de este tipo de tecnologías.

Después del proceso descrito al alumno se le ha brindado la oportunidad de profundizar en un campo con multitud de ramas, técnicas, metodologías y conceptos, que son fruto de años de estudio de expertos y apasionados. Podría profundizar en el proceso de compilación o traducción, en las distintas gramáticas, en técnicas de optimización o diseñar sus propias herramientas de traducción o interpretación de lenguajes formales.

Por otro lado, en la industria de la tecnología de la información se hace uso de lenguajes muy completos, con gran diversidad de mecanismos y bien consolidados, que son efecto de la evolución y las necesidades en el sector. En la mayoría de cursos académicos se estudian estas herramientas desde un punto de vista práctico y de uso.

La teoría de autómatas y lenguajes formales presenta la base para el estudio de los compiladores e interpretes que son parte fundamental de la industrial actual. A pesar de ello no existen herramientas divulgativas, colaborativas e interactivas que, a partir de los conocimientos básicos, ayude a comprender cómo se desarrollan los distintos mecanismos y herramientas presentes en la tecnología actual.

Los lenguajes de programación han evolucionado mucho desde que comenzaron ha alejarse del lenguaje máquina. El alumnado actual trabaja sobre los conceptos que le ayudan a entender esta evolución, pero no dispone de herramientas o medios para ver cómo estos conceptos son trasladados a un producto real y presente en el día a día de un programador actual.

3. Necesidades

Dada la situación actual se precisa de una herramienta que ayude a comprender cómo se implementa y construye un intérprete para un lenguaje de programación. Es condición necesaria que este proceso quede correctamente documentado. La herramienta elaborada deberá ser accesible por cualquier interesado en el tema, que desee profundizar en la práctica del desarrollo de interpretes y lenguajes de programación.

Se partirá de los conceptos básicos de la teoría de autómatas y los lenguajes formales, así como de la teoría de compiladores e intérpretes. Se asume pues que el usuario dispone de este conocimiento.

4. Objetivos

Se llegará a construir un lenguaje de programación completo con características presente en la tecnología actual. Este proceso quedará correctamente documentado y se pondrá a disposición pública. Las características que serán contempladas son:

- Distintos tipos de datos simples y compuestos.
- Expresiones lógicas y aritméticas.
- Expresiones y funciones sobre cadenas.
- Expresiones y funciones sobre vectores de datos.
- Expresiones y funciones sobre expresiones regulares.
- Operadores de conversión de tipos.
- Mecanismos de entrada/salida.
- Creación de procesos.
- Manipulación de ficheros.
- Funciones de fecha y tiempo.
- Definición y uso de variables de tipado dinámico.
- Ámbito de variables.
- Sentencias de control de flujo condicionales e iterativas.
- Definición y uso de funciones y procedimientos.
- Mecanismos y técnicas de la programación funcional.
- Definición y uso de clases de objetos.
- Mecanismos y técnicas de la programación orientada a objeto.
- Integración de módulos que extienden el lenguaje.

El interprete desarrollado podrá ser usado de una forma interactiva, permitiendo así la ejecución de instrucciones bajo demanda. Además deberá tener la capacidad de dar información estructurada de todo el proceso de interpretación. Otros sistemas podrán utilizar esta información para ilustrar este proceso de una forma flexible. El intérprete también

se deberá poder ejecutar en modo servidor, recibiendo así cualquier cadena que hace de código fuente por un puerto TCP e interpretándola.

Toda la documentación generada a partir del proceso de desarrollo deberá ser estructurada y cumplimentada para formar una parte de una aplicación web que la haga accesible. La aplicación web además permitirá el uso online del intérprete, y mostrará información relativa al proceso de interpretación.

El proyecto presentará una licencia de uso libre para que pueda ser usado abiertamente por la comunidad, a la vez que se nutrirá de las contribuciones de la misma.

5. Requisitos funcionales

5.1. Intérprete

5.1.1. Léxico

Número: 22 de noviembre de 2015

Nombre: Léxico.

Categoría: Intérprete.

Descripción: El sistema debe fijar el léxico del lenguaje conformado por una conjunto de palabras y expresiones bien definidas y acotadas.

5.1.2. Gramática

Número: 0001

Nombre: Gramática.

Categoría: Intérprete.

Descripción: El sistema debe definir una gramática que representará el lenguaje. La gramática debe ser libre de contexto, clara y uniforme en toda su extensión. Además debe estar libre de ambigüedades.

5.1.3. Semántica

Número: 0002

Nombre: Interpretación semántica.

Categoría: Intérprete.

Descripción: Dado un contenido fuente el sistema debe analizarlo en función al léxico (análisis léxico) y la gramática (análisis sintáctico) del lenguaje y producir el resultado semántico asociado.

5.1.4. Comentarios

Número: 0003

Nombre: Comentarios.

Categoría: Intérprete.

Descripción: Se ha de contemplar un mecanismo para añadir comentarios al contenido fuente que serán ignorados durante la tarea de interpretación.

Los comentarios comprenderán desde un carácter "#", o bien "//", hasta fin de línea.

Por otro lado se ha de contemplar los comentarios de múltiples líneas, que deberán estar contenidos entre "/*" y "*/".

5.1.5. Contenido fuente

Número: 0004

Nombre: Fuente desde línea de comandos.

Categoría: Intérprete.

Descripción: El intérprete debe ser capaz de obtener contenido fuente desde una línea de comandos.

Número: 0005

Nombre: Fuente desde entrada estándar.

Categoría: Intérprete.

Descripción: El intérprete debe ser capaz de obtener contenido fuente desde la entrada estándar del sistema.

Número: 0006

Nombre: Fuente desde fichero.

Categoría: Intérprete.

Descripción: El intérprete debe ser capaz de obtener contenido fuente desde un fichero.

Número: 0007

Nombre: Fuente desde puerto TCP.

Categoría: Intérprete.

Descripción: El intérprete debe poder ser ejecutado en modo servidor, recibiendo una cadena de caracteres por un puerto TCP, e interpretándola. Esto ocasionará que el servidor devuelva por otro puerto una estructura de datos que representa el proceso de interpretación.

5.1.6. Salida

Número: 0008

Nombre: Salida por la salida estándar.

Categoría: Intérprete.

Descripción: El intérprete debe poder utilizar la salida estándar del sistema para imprimir el resultado de la interpretación según el código fuente. Para ello se deberá presentar sentencias que permitan escribir en esta.

Número: 0009

Nombre: Salida estructurada.

Categoría: Intérprete.

Descripción: El intérprete debe de tener la capacidad de producir una serie de datos estructurados que representen el proceso de interpretación llevado a cabo para un determinado código fuente. Estos datos podrán ser almacenados en un fichero o devueltos por un puerto TCP.

5.1.7. Entorno

Número: 0010

Nombre: Entorno de ejecución.

Categoría: Intérprete.

Descripción: El intérprete debe definir un entorno de ejecución en el que se controlen parámetros de entrada, variables de entornos del sistema operativo e información sobre el proceso como número de línea actual y los errores producidos.

5.1.8. Parámetros

Número: 0011

Nombre: Parámetros al programa.

Categoría: Entrada.

Descripción: Se debe facilitar un mecanismo para que el contenido fuente pueda recibir parámetros de entrada desde la invocación a su interpretación. Estos parámetros deberán ser copiados a símbolos variables accesibles desde el contenido fuente. Adicionalmente se tratará otro parámetro que se corresponderá con el número de parámetros dados.

5.2. Ejecución

5.2.1. Sentencias

Número: 0012

Nombre: Sentencia.

Categoría: Ejecución.

Descripción: Son las unidades interpretables más pequeña en las que se divide un contenido fuente. Las sentencias están sujetas a unas reglas sintácticas y encierran un significado semántico. El intérprete debe definir la gramática de cada sentencia y dotarlas de significado semántico. Toda sentencia debe finalizar con el carácter ";", excluyendo las sentencias formadas por bloques de sentencias. Aunque carezca de sentido práctico, para evitar posibles errores de codificación y mantener coherencia en la sintaxis y la definición del lenguaje, se debe contemplar la sentencia vacía que sólo conste del carácter ";".

Número: 0013

Nombre: Bloques de sentencias.

Categoría: Ejecución.

Descripción: Son un conjunto de sentencias que deberán ser interpretadas y ejecutadas secuencialmente. La disposición de sentencias en el bloque determinan el flujo de ejecución que se llevará a cabo cuando se intérprete el bloque. El contenido fuente en si mismo es un bloque de sentencias. Todo bloque de sentencias de más de una sentencia (con excepción del contenido fuente en si mismo) debe ir delimitado mediante llaves. Aunque no sea de uso común, para mantener coherencia en la sintaxis y la definición del lenguaje, se debe contemplar el bloque de sentencias vacío.

5.2.2. Expresiones

Número: 0014

Nombre: Expresiones.

Categoría: Ejecución.

Descripción: El intérprete debe ser capaz de evaluar expresiones. Estas son secuencias de datos, operadores, operandos, elementos de puntuación y/o palabras clave, que especifican una unidad computacional. Generalmente encierran un valor que se asocia a la expresión después de ser evaluada. Una sentencia puede estar formada por una o varias expresiones que deberán ser evaluadas o interpretadas para dotarla de significado. Una sentencia puede constar únicamente de una expresión en ese caso la sentencia es considerada la evaluación de dicha expresión.

La expresión más simple equivale a un único dato, en este caso el valor de la expresión será el del dato.

Número: 0015

Nombre: Expresiones de tipo definido.

Categoría: Ejecución.

Descripción: Son expresiones cuyo valor es de un tipo definido y fijo. El sistema debe interpretar estas expresiones para determinar el valor asociado a la mismas en un momento dado.

Nombre: Expresiones de tipo no definido.

Categoría: Ejecución.

Número: 0016

Descripción: Son expresiones cuyo valor no tiene un tipo definido ni fijo, sino que es durante la interpretación cuando se determina el tipo. El sistema debe interpretar estas expresiones para determinar, además del valor asociado a la mismas, el tipo de dato que guardan en un momento dado.

5.2.3. Datos

Nombre: Datos.

Categoría: Ejecución.

Número: 0017

Descripción: El intérprete deberá operar sobre datos. El contenido fuente definirá cómo se han de construir y/o acceder a los datos y las operaciones que se realizarán sobre ellos durante la ejecución.

Un dato será tratado en función de su tipo. El tipo de dato lo dota de una semántica, un significado. Así, todo dato deberá ser considerado un objeto, por lo que tendrán unas propiedades y funcionalidad ligadas al tipo como el que es tratado.

5.2.4. Operadores

Número: 0018

Nombre: Operadores.

Categoría: Operadores.

Descripción: Se ha de facilitar una serie de operadores que permitan manipular los datos. Los operadores son en si mismo expresiones, por los que estos tendrán un valor asociado tras ejecutarse. Los operadores constarán de una serie operandos que intervendrán en la operación y que serán a su vez otras expresiones.

Los operadores se clasificarán en función del tipo de valor que tendrán tras la ejecución, los tipos de los operandos y/o la naturaleza del operador en si.

Un operador puede presentarse en forma de función, los operandos serán los parámetros de esta. La ejecución de la función conllevará la realización de la operación asociada al operador.

5.3. Tipos de datos

Número: 0019

Nombre: Tipos de datos.

Categoría: Tipos de datos.

Descripción: El sistema debe ser capaz de interpretar y operar sobre diferentes tipos de datos. Las expresiones pueden tener un tipo de dato asociado que puede o no ser definido y fijo. Los tipos de datos pueden ser simples o compuestos.

Un dato debe tratarse como diferente tipo en función del contexto en el que se utilice. Así un dato de un tipo concreto puede ser tratado como otro tipo de dato si fuese necesario. Un dato por si mismo siempre será considerado del tipo de dato con el que se creó, sin embargo cuando interviene en una operación es posible que se precise una conversión o equivalencia de tipos. Para ello debe tomar su valor como si de otro tipo se tratase. Si en la operación no es posible convertir el tipo en el tipo requerido se debe producir un error de tipos.

Se debe establecer un mecanismo de conversión de tipos. La relación de conversión de tipo debe ser transitiva, así si un dato de tipo lógico puede verse como un dato de tipo aritmético y un dato aritmético como un cadena de caracteres, entonces el dato de tipo lógico también puede verse como una cadena.

5.3.1. Nulo

Número: 0020

Nombre: Tipo de dato nulo.

Categoría: Tipo de dato.

Descripción: Se debe contemplar el tipo de dato nulo. Este tipo de dato tendrá un único valor posible. El valor nulo deberá representar un elemento no definido. Una expresión puede tomar el valor nulo cuando sea evaluada si se hace uso de elementos no definidos.

5.3.2. Lógico

Número: 0021

Nombre: Tipo de dato lógico.

Categoría: Tipo de dato.

Descripción: El sistema debe ser capaz de interpretar y operar sobre datos de tipo lógicos. Este tipo de dato sólo contempla dos posibles valores: falso y verdadero. Este será el tipo de dato más simple. Un dato lógico puede ser tratado como un tipo de dato aritmético tomándose falso como el valor cero, y verdadero como el valor uno.

5.3.3. Aritmético

Número: 0022

Nombre: Tipo de dato aritmético.

Categoría: Tipo de dato.

Descripción: El sistema debe ser capaz de interpretar y operar sobre datos de tipo aritméticos. Este tipo de dato contempla valores numéricos racionales que puedan ser represetnados mediante notación en coma flotante. Todo dato aritmético además tiene asociado un valor lógico cuando se utiliza como este tipo de dato, tal que cualquier número distinto de cero tiene valor verdadero y el cero tiene el valor falso. Además cuando un dato aritmético es tratado como una cadena de caracteres se tomará la cadena que representa al número. El intérprete debe tener la capacidad de configurarse con una representación interna distinta para los datos aritméticos.

5.3.4. Cadena de caracteres

Número: 0023

Nombre: Tipo de dato cadenas de caracteres

Categoría: Tipo de dato.

Descripción: El sistema debe ser capaz de interpretar y operar sobre datos de tipo cadena de caracteres. Este tipo de dato contempla cualquier sucesión de caracteres alfanuméricos, secuencias de escape, u otros signos o símbolos. Esta sucesión pude ser vacía. Una cadena de caracteres vendrá delimitada mediante comillas dobles o simples. Toda cadena de caracteres además tiene asociado un valor aritmético cuando se utiliza como este tipo de dato, tal que, si la cadena representa un número racional el valor será el del propio número, por otro lado, si la cadena no representa un número racional el valor aritmético de la misma será el número de caracteres que la conforman. No se ha de considerar el tipo de dato carácter simple, pudiéndose tratar este como una cadena de un solo elemento.

5.3.5. Expresiones regulares

Número: 0024

Nombre: Tipo de dato expresión regular.

Categoría: Tipo de dato.

Descripción: El sistema debe ser capaz de interpretar y operar sobre datos de tipo expresión regular. Una expresión regular consiste en una cadena de caracteres que representan un patrón. Las expresiones regulares tendrán una sintaxis PERL. Una expresión regular se delimita mediante caracteres acento grave (`). El tipo de dato expresión regular no debe ser tratado como otro tipo de dato.

5.3.6. Arrays

Número: 0025

Nombre: Tipo de dato array.

Categoría: Tipo de dato.

Descripción: El sistema debe ser capaz de interpretar y operar sobre datos de tipo array. Este tipo de dato contempla cualquier sucesión de elementos. Estos elementos pueden ser pares de expresiones clave/valor donde la clave servirá para referenciar el valor dentro de la sucesión. También pueden ser simples expresiones, por lo que se tomará automáticamente una clave numérica y secuencial según el orden del array y como valor el de la expresión. El significado semántico de las claves en un array puede ser numérico (array numérico) o cadenas de caracteres (array asociativo). Una definición de array se delimita mediante llaves y sus elementos se denotarán mediante un listado de expresiones o pares de estas. Un dato de tipo array solo puede ser tratado como un tipo de dato booleano, siendo falso si se encuentra vacío y verdadero en caso contrario.

5.4. Sentencias

5.4.1. Inclusión de ficheros

Número: 0026

Nombre: Inclusión de ficheros.

Categoría: Sentencias de control de flujo.

Descripción: Se ha de facilitar un mecanismo para incluir, en un punto de la ejecución, contenido fuente localizado en recurso externo. El recurso consistirá en un fichero con sentencias interpretables.

5.4.2. Saltar a etiqueta

Número: 0027

Nombre: Saltar a etiqueta.

Categoría: Sentencias de control de flujo.

Descripción: Se ha de facilitar un mecanismo para llevar el flujo de ejecución a la sentencia referenciada por una etiqueta.

5.4.3. Sentencia if

Número: 0028

Nombre: Sentencia if.

Categoría: Sentencias de control de flujo.

Descripción: Deben de existir una serie de sentencias condicionales que alteren el flujo de ejecución. Las sentencias if deberán estar construidas por bloques de sentencias y una serie de expresiones denominadas "condiciones". La interpretación de una sentencia de este tipo debe consistir en la evaluación lógica de las "condiciones" para determinar el bloque de sentencias que se ejecutará. Las formas de la sentencia if que el intérprete debe aceptar son las siguientes:

- if (cond) stmts
- if (cond) stmts else stmts
- if (cond) stmts elif (cond) stmts ... else stmts

5.4.4. Sentencia switch

Número: 0029

Nombre: Sentencia switch.

Categoría: Sentencias de control de flujo.

Descripción: El intérprete debe ser capaz de interpretar sentencias del tipo switch case. Estas constan de una lista de bloques de sentencias precedidas de una expresión denominada "caso". Dada una expresión base esta debe ser comparada mediante la operación de igualdad con cada uno de los casos, ejecutando el bloque correspondiente al "caso" cuya comparación sea positiva y todos los bloques siguientes. Se deberá poder especificar un bloque denominado "default" que no dispondrá de expresión "caso" y será ejecutado sin aplicar condición alguna.

5.4.5. Sentencia while

Número: 0030

Nombre: Sentencia while.

Categoría: Sentencias de control de flujo.

Descripción: El intérprete debe ser capaz de interpretar sentencias del tipo while. Esta es una sentencia de control iterativa que consta de una expresión denominada "condición" y un bloque de sentencias. El bloque de sentencias debe ser ejecutado mientras que "condición" permanezca verdadera.

5.4.6. Sentencia do...while

Número: 0031

Nombre: Sentencia do...while.

Categoría: Sentencias de control de flujo.

Descripción: El intérprete debe ser capaz de interpretar sentencias del tipo do while. Esta es una sentencia de control iterativa que consta de una expresión denominada "condición" y un bloque de sentencias. El bloque de sentencias debe ser ejecutado mientras que "condición" permanezca verdadera, llevándose a cabo la ejecución al menos una vez.

5.4.7. Sentencia for

Número: 0032

Nombre: Sentencia for.

Categoría: Sentencias de control de flujo.

Descripción: El intérprete debe ser capaz de interpretar sentencias del tipo for. Esta es una sentencia de control iterativa que consta de tres expresiones denominadas "inicialización", "condición" y "paso", además de un bloque de sentencias. Primero se ha de evaluar la expresión "inicialización", luego el bloque de sentencias se ejecutará mientras "condición" se valore como verdadera. La expresión "paso" se deberá ejecutar al finalizar cada iteración.

5.4.8. Sentencia foreach

Número: 0033

Nombre: Sentencia foreach.

Categoría: Sentencia de control de flujo.

Descripción: Se han de de interpretar sentencias del tipo forearch. Esta es una sentencia de control iterativa que consta un bloque de sentencias, de una expresión denominada "conjunto" y un identificador denominado "valor". Se debe poder, aunque de forma opcional, especificar otro identificador que se denominará "clave". El bloque de sentencias será ejecutado de forma iterativa en función el tipo de dato y el valor de "conjunto". El "conjunto" será evaluado para determinar el número de iteraciones y el valor que se le asignará como variables a los identificadores en cada iteración. Dependiendo del tipo de la expresión "conjunto" la sentencia foreach deberá actuar como sigue:

- **Tipo lógico:** El bloque de sentencias se ejecutará mientras "conjunto" sea verdadero. El identificador "valor" tomará el valor verdadero. En el caso en el que se especifique un identificador "clave" a este no se le asignará ningún valor.
- **Tipo aritmético:** Si "conjunto" representa un número mayor que cero el bloque de sentencias se ejecutará tantas veces como el valor numérico que representa. En cada iteración "valor" se le asignará el número de la iteración comenzando por cero. Si se presenta un identificador "clave" a este no se le asignará ningún valor. Si el valor de "conjunto" es menor o igual a cero el bloque no deberá ejecutarse.
- **Tipo cadena de caracteres:** Si "conjunto" es una cadena de caracteres que representa un número racional la ejecución deberá ser como si de un tipo aritmético se tratase. Si el "conjunto" es una cadena que no representa un número racional el bloque de sentencias se ejecutará por cada carácter en la cadena. En este último caso a "valor" se le asignará el carácter contemplado en cada iteración. Si se dio un identificador "clave" este no será asignado.
- **Tipo array:** Si "conjunto" es un array, u otro tipo de dato derivado de este como un objeto, el bloque de sentencias se ejecuta por cada elemento en el mismo. Al identificador "valor" se le asignará el valor del elemento en el array correspondiente a la iteración. En el caso de que se facilite un identificador "clave" este deberá tomar la clave del elemento en el array.
- **Otros tipos:** No se llevará a cabo ninguna operación.

5.4.9. Sentencia de iteración ágil

Número: 0034

Nombre: Sentencia de iteración ágil.

Categoría: Sentencia de control de flujo.

Descripción: Se ha de facilitar una sentencia de control que permita iterar un bloque de sentencias en función una expresión "conjunto" de forma ágil y sencilla. Para ello esta sentencia deberá operar igual que la sentencia foreach pero sin ser necesario, aunque posible, dar un identificador "valor" sobre el que se realizará la asignación. En lugar de ello la asignación que se produce en cada iteración se deberá realizar sobre un símbolo con identificador fijo y contenido variable denominado iterador. El iterador debe ser accesible desde el bloque de sentencias. Además se debe contemplar el acceso al iterador de varias sentencias de ciclo ágil cuando estas se presentan de forma anidada.

5.4.10. Sentencia with

Número: 0035

Nombre: Sentencia with.

Categoría: Sentencia de control de flujo.

Descripción: Se deberá facilitar un mecanismo que permita establecer una estructura compuesta como contexto. Así todo acceso que se realice, y cuya definición no exista, se deberá hacer sobre los elementos que de la estructura compuesta utilizada como contexto. Esta sentencia se deberá construir a partir del dato compuesto y un bloque de sentencias sobre el que se aplicará el contexto.

5.4.11. Finalizar bloque de sentencias

Número: 0036

Nombre: Finalizar bloque de sentencias.

Categoría: Sentencias de control de flujo.

Descripción: Se ha de disponer de un mecanismo para indicar que el flujo debe salir de una sentencia de control. Se ha de contemplar las sentencias anidadas.

5.4.12. Finalizar iteración

Número: 0037

Nombre: Finalizar iteración.

Categoría: Sentencias de control de flujo.

Descripción: El sistema debe facilitar algún recurso que permita finalizar la iteración actual de una sentencia de control en ejecución y comenzar con la siguiente. Este mecanismo debe contemplar la posibilidad de salir de varias sentencias de control anidadas.

5.4.13. Finalizar ejecución

Número: 0038

Nombre: Finalizar ejecución.

Categoría: Sentencias de control de flujo.

Descripción: Se ha de disponer de un mecanismo para que el sistema finalice de forma inmediata de interpretar el contenido fuente.

5.4.14. Capturar excepción

Número: 0039

Nombre: Capturar excepción.

Categoría: Sentencias de control de flujo.

Descripción: Se ha de disponer de un mecanismo para definir un bloque de sentencias en el que puede suceder una excepción, a la vez que permita definir un bloque de sentencias en el que esta será tratada. La excepción podrá devolver un valor que debe ser capturado. No se ha de contemplar bloques en los que se trate cada excepción según su tipo, en lugar de ello se usará un único bloque de sentencias y el tipo deberá ser evaluado dentro de este.

5.4.15. Lanzar excepción

Número: 0040

Nombre: Lanzar excepción.

Categoría: Sentencias de control de flujo.

Descripción: Se ha de disponer de un mecanismo para lanzar excepciones. Cuando una excepción es lanzada esta podrá ser capturada por un bloque de sentencias destinado para ello. Si la excepción no es capturada se producirá un error. Cuando una excepción es lanzada se deberá de dar un valor que podrá ser utilizado en el bloque que la captura.

5.5. Definiciones

Número: 0041

Nombre: Identificadores.

Categoría: Definiciones.

Descripción: El intérprete debe facilitar mecanismos para que el usuario defina e identifique expresiones, datos, bloques de sentencias, y otras construcciones y elementos del lenguaje. Se precisa una manera unívoca de nombrar estos elementos. Un identificador válido debe estar formado por una secuencia de caracteres alfanuméricos de al menos un carácter, donde el primer carácter a de ser una letra.

Número: 0042

Nombre: Tabla de símbolos.

Categoría: Definiciones.

Descripción: El intérprete debe ser capaz de gestionar tablas de símbolos. Los símbolos hacen referencias a valores, funciones y otras expresiones del lenguajes. Para acceder a estos símbolos se debe utilizar un identificador. Se hace necesario el acceso y uso de los símbolos según el contexto de ejecución, determinado por el ámbito y el tipo símbolo, para ello deben poder coexistir diferentes tablas de símbolos globales. Para evitar conflictos en el uso de identificadores algunos conceptos deben disponer de su propia tabla de símbolos.

5.5.1. Variables

Número: 0043

Nombre: Variables.

Categoría: Definiciones.

Descripción: El intérprete debe ser capaz gestionar una serie de símbolos denominados variables. Estos relacionan un identificador con un valor que puede variar durante el proceso de ejecución. El tipo de una variable dependerá del tipo del valor al que referencia (tipado dinámico), este podría ser de cualquiera de los tipos de datos soportados. La tabla de símbolos de variables debe adaptarse al contexto de ejecución.

Número: 0044

Nombre: Variables globales.

Categoría: Definiciones.

Descripción: Aunque la tabla de símbolos de variables es dependiente del contexto de ejecución se ha de facilitar algún mecanismo para que un dato esté disponible independientemente del contexto en el que se acceda.

5.5.2. Funciones

5.5.2.1 Definición de función

Número: 0045

Nombre: Definición de función

Categoría: Definiciones.

Descripción: Se necesita de un mecanismo que permita definir y nominar bloques de sentencias. Estos bloques podrán recibir unos valores de entrada y producir una salida. Las sentencias en el bloque podrán operar sobre los parámetros de entrada, representados por unos símbolos variables que tomarán distintos valores en cada ejecución. Tras interpretarse el bloque de sentencias se podrá tomar un valor considerado de salida.

La definición de una función representará en si misma un dato, por lo que podrán formar parte de operaciones y otras expresiones. Una función se define mediante un bloque de sentencias, una lista de identificadores que nominan a los parámetros de entrada y un identificador que le da nombre a la propia función, aunque este último no debe ser necesario (funciones anónimas).

5.5.2.2 Llamada a función

Número: 0046

Nombre: Llamada a función

Categoría: Definiciones.

Descripción: Dada una función, se debe disponer de un mecanismo que permita la ejecución del bloque de sentencias que la forma, mediante el uso de unos valores concretos como parámetros de entrada, y con la posibilidad de tomar el valor de salida.

Una llamada a función se deberá componer de un identificador relativo a su definición, y una lista de expresiones que determinarán los valores de los parámetros. La llamada deberá ser en si misma una expresión que tomará como valor la salida de la función tras la ejecución.

Los valores de los parámetros se corresponderán con los parámetros de la definición de la función de forma posicional.

5.5.2.3 Valor de retorno

Número: 0047

Nombre: Valor de retorno

Categoría: Definiciones.

Descripción: Se necesita de un mecanismo en forma de sentencia que, dada una función, determine el valor del salida que se tomará en la llamada a la misma. La sentencia return se compondrá de una expresión correspondiente al valor salida. Al ser interpretada la ejecución de la función deberá finalizar, y tomará el valor de la expresión dada.

5.5.2.4 Valores de parámetros por defecto

Número: 0048

Nombre: Valores de parámetros por defecto.

Categoría: Definiciones.

Descripción: Dada la definición de una función, debe existir un mecanismo para que los parámetros de esta puedan tener valores por defecto. Estos valores serán asignado a los parámetros cuando los valores no sean dados en una llamada a función.

Como la correspondencia entre parámetros en una llamada a función se hace de forma posicional, los valores por defecto deberán ser especificados desde el final de la lista de parámetros hasta el inicio.

5.5.2.5 Parámetros por valor

Número: 0049

Nombre: Parámetros por valor.

Categoría: Definiciones.

Descripción: Cuando una función es ejecutada todos los símbolos variables que se definan y utilicen deben tratarse de forma local al bloque de sentencias de la función. De esta forma los símbolos variables definidos fuera de la función no serán accesibles desde el cuerpo de la misma y viceversa. Cuando se realice una llamada a función los valores de los parámetros deben ser copiados a los símbolos variables correspondientes.

5.5.2.6 Parámetros por referencia

Número: 0050

Nombre: Parámetros por referencia.

Categoría: Definiciones.

Descripción: Se necesita de un mecanismo que permita que los parámetros de una función referencien valores definidos fuera del cuerpo de la misma. De esta forma se podrá acceder y/o modificar datos externos a la función.

Cuando en una llamada a función se especifiquen expresiones que sean símbolos variables como algunos de sus parámetros, si estos se definieron en la función como parámetros por referencia, el valor del símbolo en la llamada será referenciado por el símbolo correspondiente de la función.

5.5.2.7 Función lambda

Número: 0051

Nombre: Función lambda.

Categoría: Definiciones.

Descripción: Se debe dar la posibilidad de crear funciones anónimas. Estas funciones carecerán de nombre y normalmente se utilizarán en la asignación de variables, como parámetros de otras funciones o como valor de retorno. Las funciones lambda deberán ser en si misma una expresión que toma como valor el dato correspondiente a la función.

5.5.2.8 Expresiones parametrizadas

Número: 0052

Nombre: Función lambda simple.

Categoría: Definiciones.

Descripción: Se debe de facilitar un mecanismo para crear funciones simples, que solo consten de una lista de parámetros y de una única expresión que será devuelta y que constituirá el cuerpo de la función.

5.5.2.9 Referencia a función

Número: 0053

Nombre: Referencia a función.

Categoría: Definiciones.

Descripción: Se debe facilitar un mecanismo para referenciar funciones ya creadas, de forma que puedan ser asignadas a variables, pasadas como parámetros o devueltas como valor de retorno.

5.5.2.10 Funciones de orden superior

Número: 0054

Nombre: Funciones de orden superior.

Categoría: Definiciones.

Descripción: Se debe poder definir funciones de orden superior, estas pueden recibir otras funciones como parámetros o tomar como valor de retorno otra función.

5.5.2.11 Funciones clausura

Número: 0055

Nombre: Funciones clausura.

Categoría: Definiciones.

Descripción: Se debe poder definir funciones dentro del contexto de otras. La función de clausura puede tener variables que dependan del entorno en el que se ha definido la función. De esta forma cuando la función es llamada podrá acceder al valor de la variable en el contexto en el que se definió.

5.5.2.12 Aplicación parcial

Número: 0056

Nombre: Aplicación parcial.

Categoría: Definiciones.

Descripción: Se debe facilitar un mecanismo que permita, a partir de una función, obtener otra equivalente donde se ha dado valor a un subconjunto de los parámetros.

5.5.2.13 Decoradores

Número: 0057

Nombre: Decoradores.

Categoría: Definiciones.

Descripción: Se debe facilitar un mecanismo para definir decoradores. Un decorador será un tipo especial de función. Al igual que una función se define mediante un identificador que lo nomina, una lista de parámetros y un bloque de sentencias.

A diferencia de las funciones ordinarias, la llamada a un decorador deberá tener como parámetro una función que será decorada, como resultado se deberá obtener una función que tendrá las siguientes características:

- La lista de parámetros que admite será la misma que la lista con la que se definió el decorador
- El bloque de sentencias será el del decorador pero haciendo uso de la función que ha sido decorada

Se debe facilitar un mecanismo para referenciar la función que se va a decorar dentro del decorador. Para ello se utilizará la función de contexto.

5.5.2.14 Función de contexto

Número: 0058

Nombre: Función de contexto.

Categoría: Definiciones.

Descripción: Se debe facilitar un mecanismo para acceder a la función de contexto. Esta será una función cuyo valor dependerá del contexto en el que se ejecute:

- En el primer nivel de ejecución la función de contexto no estará definida.
- En el cuerpo de una función será la propia función.
- En el cuerpo de un decorador será la función que se decorará.

5.5.3. Clases de objetos

5.5.3.1 Clase de objeto

Número: 0059

Nombre: Clase de objeto.

Categoría: Definiciones.

Descripción: El lenguaje debe contemplar el paradigma de la programación orientada a objetos. Una clase se ha de ver como una definición estática e inmutable que será utilizada para la creación de objetos.

Las clases definirán tipos de objetos que tendrán métodos y atributos comunes. Una clase se construye mediante un identificador que le da nombre y un bloque de sentencias que contendrá una serie de funciones (métodos) y símbolos variables (atributos).

Las características de la programación orientada a objetos que se deberán contemplar son:

Abstracción: Un objeto por si mismo representará una entidad abstracta que podrá tener cierta funcionalidad asociada, disponer de atributos que establezcan su estado interno o comunicarse con otros objetos.

Encapsulamiento: Un objeto podrá contener todos los elementos correspondiente a su definición, estado y funcionalidad.

Principio de ocultación: Un objeto podrá tener atributos y/o métodos privados, de forma que sólo sean accesibles desde el propio objeto.

Polimorfismo: Se debe permitir a objetos de distinto tipo se le pueda enviar mensajes sintácticamente iguales, de forma que se pueda llamar un método de objeto sin tener que conocer su tipo.

Herencia: Se debe contemplar la herencia simple entre clases de forma que una clase se pueda definir mediante otra.

5.5.3.2 Objeto

Número: 0060

Nombre: Objeto.

Categoría: Definiciones.

Descripción: El sistema debe ser capaz de interpretar y operar sobre objetos. Los objetos serán vistos como estructuras de datos funcionales, formado tanto por datos de diferentes tipos (atributos), como por funciones (métodos).

Se podrá crear objetos a partir de una clase ya definida. Las clases de objetos pueden definir un método constructor que será utilizado cuando se cree un objeto a partir de la misma.

Dentro del bloque de sentencias que conforma un método es posible hacer referencia a los demás valores del objeto mediante la expresión especial "this".

Las clases de objetos podrán definir un método constructor que será llamado cuando el objeto sea instanciado.

Un objeto podrá disponer de un método que será llamado cuando se precise su conversión a un tipo de dato cadena de caracteres. También podrán disponer de métodos que serán llamados cuando se acceda a un atributo o método no existente.

5.5.3.3 Elementos privados

Número: 0061

Nombre: Elementos privados.

Categoría: Clases de Objetos.

Descripción: Se debe facilitar un mecanismo para definir atributos y métodos de una clase de objetos como privados. Estos elementos solo serán accesibles desde métodos del propio objeto. Se deberá contemplar el acceso a estos elementos sobre objetos del mismo tipo dentro de métodos de la clase.

5.5.3.4 Elementos estáticos

Número: 0062

Nombre: Elementos estáticos.

Categoría: Clases de Objetos.

Descripción: Se debe facilitar un mecanismo para definir atributos y métodos de una clase de objetos pertenecientes a la propia clase. Estos elementos no serán trasladados a los objetos instanciados.

5.5.3.5 Herencia de clases

Número: 0063

Nombre: Herencia de clases.

Categoría: Clases de Objetos.

Descripción: Se debe de disponer de un mecanismo que permita establecer una relación de herencia entre unas clases dadas. Así será posible la definición de nuevas clases partiendo de otras. La clase derivará de otra extendiendo su funcionalidad y definición.

En la definición de una clase se debe de disponer de un mecanismo que permita especificar la clase que se extenderá. La nueva clase tendrá todos los atributos y métodos de la extendida y añadirá los suyos propios, pudiendo sobrescribirse los ya existentes.

5.5.3.6 Instanciación de clases

Número: 0064

Nombre: Instanciación de clases.

Categoría: Clases de Objetos.

Descripción: Dada una clase, se debe de disponer de un mecanismo que permita crear objetos a partir de la misma. Para construir un objeto a partir de la instanciación de una clase se deben llevar las funciones y variables definidas en el cuerpo de la clase a métodos y atributos del objeto.

Una clase pude definir un método constructor que deberá ser llamado sobre el objeto recién creado cuando la clase es instanciada.

La instanciación se deberá realizar mediante un operador que, a partir de un identificador correspondiente a la clase y una lista de expresiones correspondientes a los parámetros del método constructor, tome como valor el objeto recién creado.

5.5.3.7 Acceso al objeto en ejecución

Número: 0065

Nombre: Acceso al objeto en ejecución.

Categoría: Clases de Objetos.

Descripción: Se debe de disponer de un mecanismo que permita acceder a los atributos y métodos de un objeto desde la ejecución de un método del mismo. Este mecanismo, correspondiente a una expresión, deberá tomar como valor el objeto en ejecución.

5.5.3.8 Acceso al objeto en ejecución como clase padre

Número: 0066

Nombre: Acceso al objeto en ejecución como clase padre.

Categoría: Clases de Objetos.

Descripción: Se debe de disponer de un mecanismo que permita acceder a los atributos y métodos de la clase padre de un objeto desde la ejecución de un método del mismo. Este mecanismo, correspondiente a una expresión, deberá tomar como valor el objeto en ejecución, pero tomando como métodos y atributos los de la clase padre de la cual deriva.

5.5.3.9 Enlace estático en tiempo de ejecución

Número: 0067

Nombre: Enlace estático en tiempo de ejecución.

Categoría: Clases de Objetos.

Descripción: Se debe de disponer de un mecanismo que permita acceder a los atributos y métodos estáticos de una clase hija desde un método estático de la clase padre.

5.5.3.10 Obtener clase

Número: 0068

Nombre: Obtener clase

Categoría: Clases de Objetos.

Descripción: Se debe de disponer de un mecanismo para, a partir de un objeto, obtener la clase a la que pertenece como una cadena correspondiente al nombre.

5.5.4. Listas

Nombre: Listado.

Categoría: Definiciones.

Número: 0069

Descripción: Se ha de facilitar un mecanismo que permita agrupar expresiones para darles un significado operacional común. Este deberá consistir en una serie de expresiones separadas por comas.

5.5.5. Pares clave/valor

Nombre: Pares clave/valor.

Categoría: Definiciones.

Número: 0070

Descripción: Se ha de facilitar un mecanismo que permita relacionar un par de expresiones para darles un significado estructural. Este deberá consistir en el par de expresiones separadas por el carácter ":".

5.5.6. Etiqueta

Número: 0071

Nombre: Etiqueta.

Categoría: Definiciones.

Descripción: El sistema debe ser capaz de interpretar y operar sobre etiquetas. Una etiqueta es una referencia a una sentencia concreta dentro del contenido fuente. Las etiquetas dependen quedarán definidas dentro de un contexto determinado por el bloque de sentencias en el que se encuentren.

5.5.7. Listas por comprensión

Número: 0072

Nombre: Listas por comprensión.

Categoría: Definiciones.

Descripción: Se necesita de un mecanismo que sea una expresión por si mismo y que permita generar arrays desde una sentencia iterativa. Este mecanismo se formará mediante una expresión seguida da una sentencia for. La expresión será ejecutada tras iteración del bucle y será asignada como último elemento de un array. Al final de la ejecución la expresión tomará el valor del array generado.

5.5.8. Expresiones reflexivas

Número: 0073

Nombre: Expresiones reflexivas.

Categoría: Definiciones.

Descripción: Se ha de disponer de un mecanismo de reflexión, que permita utilizar expresiones del lenguaje para definir otras expresiones como identificadores.

5.6. Asignaciones

5.6.1. Asignación

Número: 0074

Nombre: Asignación.

Categoría: Asignaciones.

Descripción: Se hace necesario la gestión de los símbolos variables creados durante la ejecución, lo que implica la asignación de valores a las variables que serán definidas y utilizadas por el contenido fuente dado por el usuario. El valor que es asignado a una variable puede ser cualquier tipo de dato contemplado, incluso funciones o objetos. El valor asignado puede ser determinado a partir de cualquier expresión que tenga un valor asociado después de su ejecución. La operación de asignación debe ser en si misma una expresión que toma como valor tras su ejecución el valor asignado.

5.6.2. Asignación de referencia

Número: 0075

Nombre: Asignación de referencia.

Categoría: Asignaciones.

Descripción: Se debe facilitar un mecanismo para que dos símbolos variables distintos referencien al mismo valor. Para ello se ha de facilitar un mecanismo para obtener la referencia de un símbolo variable, de forma que esta pueda ser usada en una asignación.

5.7. Operadores aritméticos

5.7.1. Suma

Número: 0076

Nombre: Suma.

Categoría: Operadores aritméticos.

Descripción: Se debe contemplar la expresión correspondiente a la operación aritmética "suma". Para realizar esta operación se deberá tomar el valor aritmético de cada operando. Tras realizarse la operación, el valor de la expresión deberá ser el resultado aritmético de la misma.

5.7.2. Diferencia

Número: 0077

Nombre: Diferencia.

Categoría: Operadores aritméticos.

Descripción: Se debe contemplar la expresión correspondiente a la operación aritmética "resta". Para realizar esta operación se deberá tomar el valor aritmético de cada operando. Tras realizarse la operación, el valor de la expresión deberá ser el resultado aritmético de la misma.

5.7.3. Producto

Número: 0078

Nombre: Producto.

Categoría: Operadores aritméticos.

Descripción: Se debe contemplar la expresión correspondiente a la operación aritmética "producto". Para realizar esta operación se deberá tomar el valor aritmético de cada operando. Tras realizarse la operación, el valor de la expresión deberá ser el resultado aritmético de la misma.

5.7.4. División

Número: 0079

Nombre: División.

Categoría: Operadores aritméticos.

Descripción: Se debe contemplar la expresión correspondiente a la operación aritmética "división". Para realizar esta operación se deberá tomar el valor aritmético de cada operando. El segundo operando debe ser distinto de 0. Si el segundo operando tiene valor aritmético 0 se deberá mostrar un error que informe del caso. Tras realizarse la operación, el valor de la expresión deberá ser el resultado aritmético de la misma.

5.7.5. Potencia

Número: 0080

Nombre: Potencia.

Categoría: Operadores aritméticos.

Descripción: Se debe contemplar la expresión correspondiente a la operación aritmética "potencia". Para realizar esta operación se deberá tomar el valor aritmético de cada operando. Tras realizarse la operación, el valor de la expresión deberá ser el resultado aritmético de la misma.

5.7.6. Módulo

Número: 0081

Nombre: Módulo.

Categoría: Operadores aritméticos.

Descripción: Se debe contemplar la expresión correspondiente a la operación aritmética "módulo". Para realizar esta operación se deberá tomar el valor aritmético de cada operando. El segundo operando debe ser distinto de 0. Si el segundo operando tiene valor aritmético 0 se deberá mostrar un error que informe del caso. Tras realizarse la operación, el valor de la expresión deberá ser el resultado aritmético de la misma.

5.7.7. Tamaño

Número: 0082

Nombre: Tamaño.

Categoría: Operadores aritméticos.

Descripción: Se precisa de algún mecanismo que dado un dato calcule el tamaño de este. Este operador calculará el tamaño dependiendo del tipo de dato del operando. Tras ejecutase la expresión el valor que tome será de tipo aritmético.

Lógico: Si es verdadero el tamaño es uno, si es falso será cero.

Aritmético: Tomará el número de dígitos decimales.

Cadena: El tamaño será el número de caracteres de la cadena.

Array: Para el tipo de dato array u otros derivados se el tamaño será el número de elementos contenidos en el mismo.

Otro tipo de dato: Se deberá dar un error de tipos.

5.7.8. Incremento y asignación

Número: 0083

Nombre: Incremento y asignación.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, el valor de esta se debe poder incrementar y reasignar. Para ello se tomará el valor aritmético de la variable, se incrementará en una unidad y se reasignará a la misma. El valor de la expresión será el de la variable incrementada.

5.7.9. Asignación e incremento

Número: 0084

Nombre: Asignación e incremento.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, el valor de esta se debe poder incrementar y reasignar. Para ello se tomará el valor aritmético de la variable, se incrementará en una unidad y se reasignará a la misma. El valor de la expresión será el de la variable antes de ser incrementada.

5.7.10. Decremento y asignación

Número: 0085

Nombre: Decremento y asignación.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, el valor de esta se debe poder decrementar y reasignar. Para ello se tomará el valor aritmético de la variable, se decrementará en una unidad y se reasignará a la misma. El valor de la expresión será el de la variable decrementada.

5.7.11. Asignación y decremento

Número: 0086

Nombre: Asignación y decremento.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, el valor de esta se debe poder decrementar y reasignar. Para ello se tomará el valor aritmético de la variable, se decrementará en una unidad y se reasignará a la misma. El valor de la expresión será el de la variable antes de ser decrementada.

5.7.12. Suma y asignación

Número: 0087

Nombre: Suma y asignación.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, al valor de esta se ha de poder sumar otra expresión y reasignarle el resultado. Para ello se tomará el valor aritmético de la variable, se le sumará el valor aritmético de la expresión y se reasignará a la variable el resultado. El valor de la expresión será el resultado de la suma aritmética.

5.7.13. Diferencia y asignación

Número: 0088

Nombre: Diferencia y asignación.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, al valor de esta se ha de poder restar otra expresión y reasignarle el resultado. Para ello se tomará el valor aritmético de la variable, se le restará el valor aritmético de la expresión y se reasignará a la variable el resultado. El valor de la expresión será el resultado de la resta aritmética.

5.7.14. Producto y asignación

Número: 0089

Nombre: Producto y asignación.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, al valor de esta se ha de poder multiplicar otra expresión y reasignarle el resultado. Para ello se tomará el valor aritmético de la variable, se calculará el producto y se reasignará a la variable el resultado. El valor de la expresión será el resultado del producto aritmético.

5.7.15. División y asignación

Número: 0090

Nombre: División y asignación.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, al valor de esta se ha de poder dividir por otra expresión y reasignarle el resultado. Para ello se tomará el valor aritmético de la variable, se realizará la división y se reasignará a la variable el resultado. La expresión no puede tener un valor aritmético de cero. El valor de la expresión será el resultado de la división aritmética.

5.7.16. Potencia y asignación

Número: 0091

Nombre: Potencia y asignación.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, al valor de esta se ha de poder elevar a otra expresión y reasignarle el resultado. Para ello se tomará el valor aritmético de la variable, se elevará y se reasignará a la variable el resultado. El valor de la expresión será el resultado de la potencia.

5.7.17. Módulo y asignación

Número: 0092

Nombre: Módulo y asignación.

Categoría: Operadores aritméticos.

Descripción: Dado un identificador o expresión que referencia a una dato variable, al valor de esta se ha de poder dividir por otra expresión y reasignarle el resto originado. Para ello se tomará el valor aritmético de la variable, se realizará la división por el valor aritmético de la expresión y se reasignará a la variable el resto obtenido. La expresión no puede tener un valor aritmético de cero. El valor de la expresión será el resultado de la operación módulo.

5.8. Operadores lógicos

5.8.1. AND lógico

Número: 0093

Nombre: AND lógico.

Categoría: Operadores lógicos.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica AND. Para ello se deberá tomar el valor lógico de cada uno de los operandos. La evaluación de la operación lógica AND debe ser de cortocircuito, tomándose el valor del último elemento evaluado. Así, aunque esta expresión se corresponde con un operador lógico, el valor de la misma será el del último elemento evaluado.

5.8.2. OR lógico

Número: 0094

Nombre: OR lógico.

Categoría: Operadores lógicos.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica OR. Para ello se deberá tomar el valor lógico de cada uno de los operandos. La evaluación de la operación lógica OR debe ser de cortocircuito, tomándose el valor del último elemento evaluado. Así, aunque esta expresión se corresponde con un operador lógico, el valor de la misma será el del último elemento evaluado.

5.8.3. NOT lógico

Número: 0095

Nombre: NOT lógico.

Categoría: Operadores lógicos.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica NOT. Para ello se deberá tomar el valor lógico de su único operando y negarlo. La expresión deberá tomar un valor de tipo booleano tras realizarse la operación.

5.8.4. Vacío

Número: 0096

Nombre: Vacío.

Categoría: Operadores lógicos.

Descripción: Se necesita de un operador que determine si un dato se considera vacío. Este operador tendrá un único operando y funcionará igual que el operador lógico NOT. El valor que tomará la expresión será lógico.

5.8.5. Es nulo

Número: 0097

Nombre: Es nulo.

Categoría: Operadores lógicos.

Descripción: Se necesita de un operador que determine si un dato o expresión contiene el valor nulo.

5.8.6. Menor que

Número: 0098

Nombre: Menor que.

Categoría: Operadores de comparación.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica "menor que". Para ello se deberá tomar el valor aritmético de cada operando. La expresión deberá tomar un valor de tipo booleano tras realizarse la operación.

5.8.7. Menor o igual que

Número: 0099

Nombre: Menor o igual que.

Categoría: Operadores de comparación.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica "menor o igual que". Para ello se deberá tomar el valor aritmético de cada operando. La expresión deberá tomar un valor de tipo booleano tras realizarse la operación.

5.8.8. Mayor que

Número: 0100

Nombre: Mayor que.

Categoría: Operadores de comparación.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica "mayor que". Para ello se deberá tomar el valor aritmético de cada operando. La expresión deberá tomar un valor de tipo booleano tras realizarse la operación.

5.8.9. Mayor o igual que

Número: 0101

Nombre: Mayor o igual que.

Categoría: Operadores de comparación.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica "mayor o igual que". Para ello se deberá tomar el valor aritmético de cada operando. La expresión deberá tomar un valor de tipo booleano tras realizarse la operación.

5.8.10. Igual que

Número: 0102

Nombre: Igual que.

Categoría: Operadores de comparación.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica "igual que". La operación de igualdad debe ser independiente de los tipos de datos de los operandos, aplicándose en función del tipo de dato más completo que compartan. Por ejemplo si se compara un dato cadena que no representa un número racional con uno aritmético, como el tipo de dato común a ambos es el booleano, ambos tomarán su valor lógico para la comparación. Si ambos datos son tipos compuestos, se ha de comprobar mediante la operación de igualdad todos los elementos simples que lo componen por pares y de forma posicional. Como valor de la expresión se toma el valor booleano de la operación.

5.8.11. Idéntico que

Número: 0103

Nombre: Idéntico que.

Categoría: Operadores de comparación.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica "idéntico que". Esta operación se refiere a una operación lógica de igualdad pero contemplando además que los datos tengan el mismo tipo. Como valor de la expresión se debe tomar el valor booleano resultado de aplicar la operación.

5.8.12. Distinto que

Número: 0104

Nombre: Distinto que.

Categoría: Operadores de comparación.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica "distinto que". Esta operación debe ser independiente de los tipos de datos de los operandos, aplicándose en función del tipo de dato más completo que compartan. Por ejemplo si se compara un dato cadena que no representa un número racional con uno aritmético, como el valor más completo que ambos pueden tomar es el booleano, tomarán su valor lógico para la comparación. Si ambos datos son tipos compuestos, se ha de comprobar mediante la operación de igualdad todos los elementos simples que lo componen por pares y de forma posicional. Como valor de la expresión se toma el valor booleano de la operación.

5.8.13. No idéntico que

Número: 0105

Nombre: No idéntico que.

Categoría: Operadores de comparación.

Descripción: Se debe contemplar la expresión correspondiente a la operación lógica "no idéntico que". Esta operación se corresponde con la operación inversa de la operación "idéntico que". Como valor de la expresión se debe tomar el valor booleano resultado de aplicar la operación.

5.9. Operadores sobre cadenas

5.9.1. Concatenación

Número: 0106

Nombre: Operador concatenación.

Categoría: Operadores sobre cadena de caracteres.

Descripción: La expresión que simboliza una operación de concatenación precisa de dos operandos que serán tratados como cadena de caracteres. El valor que tomará la expresión será la cadena resultante de concatenar ambas.

5.9.2. explode

Número: 0107

Nombre: Función explode.

Categoría: Operadores sobre cadena de caracteres.

Descripción: La función explode deberá tomar dos cadenas de caracteres como operandos denominadas "texto" y "separador". El valor será el array resultante de separar la cadena "texto" en diferentes cadenas en función la cadena "separador".

5.9.3. implode

Número: 0108

Nombre: Función implode.

Categoría: Operadores sobre cadena de caracteres.

Descripción: Representa la operación inversa a explode. La función implode deberá tomar como argumentos un array denominado "listado" y una cadena denominada "separador". El valor de la expresión será una cadena resultado de concatenar cada uno de los elementos de "listado" separados por la cadena "separador".

5.9.4. sprintf

Número: 0109

Nombre: Función de formato.

Categoría: Operadores sobre cadena de caracteres.

Descripción: Se hace necesario un mecanismo que permita generar cadenas formateadas. Este deberá consistir en una cadena de caracteres denominada "formato" y un listado de expresiones. La cadena formato contendrá una serie de directivas de formato. Estas directivas serán sustituidas por el valor correspondiente, según posición, de la lista de expresiones. Cuando se realiza cada sustitución el valor es formateado según la directiva.

Las directivas de formato tienen el siguiente forma:

$$\%[operador][precisión][formato]$$

Los posibles operadores serán los siguientes:

+: Fuerza la impresión del símbolo + cuando se formatean números positivos.

ˆ : Convierte el caracteres a mayúsculas cuando se formatean cadenas de texto.

#: Añade el carácter 0x cuando se formatean números hexadecimales y el carácter 0 cuando se formatean octales.

La precisión se refiere al número de decimales que se imprimirán en el caso de formatear números o el número de caracteres en el caso de formatear cadenas.

El carácter de formato indica que tipo de formato se le dará al valor:

i|d: Número entero.

u: Sin signo.

f: Coma flotante.

%: Carácter %.

e: Notación científica.

o: Octal.

x: Hexadecimal.

s|c: Cadena de texto.

5.9.5. Buscar subcadena

Número: 0110

Nombre: Función de búsqueda de subcadena.

Categoría: Operadores sobre cadena de caracteres.

Descripción: Esta es una función básica en el tratamiento de cadenas. Opera sobre dos argumentos que serán tratados como cadenas de caracteres, uno denominado "texto" y otro "subcadena". La función toma como valor un dato aritmético relativo a la posición de la primera ocurrencia de "subcadena" dentro de "texto". Si no se encuentra ningún resultado se tomará el valor nulo. Adicionalmente se puede dar otro operando denominado "offset" que simbolice la posición dentro de "texto" a partir de la cual se comenzará a buscar.

5.9.6. Buscar y remplazar

Número: 0111

Nombre: Función de remplazo de subcadena.

Categoría: Operadores sobre cadena de caracteres.

Descripción: Se necesita de un mecanismo para buscar y remplazar subcadenas dentro de otra. Este debe buscar las ocurrencias de una subcadena "búsqueda" en una cadena "texto", sustituyéndolas por una cadena de "remplazo". Esta función debe admitir el número máximo de sustituciones que se llevarán a cabo. Tras la ejecución debe tomar como valor la cadena resultante de sustituir en la cadena principal las ocurrencias de la subcadenas por la cadena de remplazo.

Adicionalmente la subcadena de "búsqueda" puede ser una expresión regular, en cuyo caso se buscará subcadenas que pertenezcan al conjunto de las palabras definido por la expresión regular.

Si se utiliza una expresión regular como patrón de búsqueda deberá ser posible utilizar en la cadena de remplazo parte de la subcadena que concuerda con la expresión regular. Para ello se ha de formar la expresión regular mediante subexpresiones delimitadas por "()". En la cadena de remplazo se debe poder hacer referencia, de forma posicional, a las subcadenas correspondientes a cada una de las subexpresiones.

5.9.7. Remplazar subcadena

Número: 0112

Nombre: Función de remplazo de subcadena mediante posiciones.

Categoría: Operadores sobre cadena de caracteres.

Descripción: Se necesita de un mecanismo para buscar y remplazar subcadenas dentro de otra. Este debe sustituir en una cadena "texto" la subcadena comprendida entre dos posiciones dadas por expresiones numéricas, sustituyendo las subcadena correspondiente por una cadena de "remplazo". Tras la ejecución debe tomar como valor la cadena resultante de sustituir, en la cadena principal, la subcadena correspondiente a las pociones dadas por la cadena de remplazo.

5.9.8. Convertir a mayúsculas

Número: 0113

Nombre: Función conversión a mayúsculas.

Categoría: Operadores sobre cadena de caracteres.

Descripción: Dada una cadena de caracteres se necesita de un mecanismo que convierta todos los caracteres alfabéticos que la conforman en mayúsculas. El valor que se tomará será la cadena resultante de la operación.

5.9.9. Convertir a minúsculas

Número: 0114

Nombre: Función conversión a minúsculas.

Categoría: Operadores sobre cadena de caracteres.

Descripción: Dada una cadena de caracteres se necesita de un mecanismo que convierta todos los caracteres alfabéticos que la conforman en minúsculas. El valor que se tomará será la cadena resultante de la operación.

5.10. Operadores sobre array

5.10.1. Dividir en fragmentos

Número: 0115

Nombre: Función dividir array en fragmentos.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array y un valor aritmético. Se divida el array en varios subarray de tantos elementos como indique el valor (a excepción del último). La expresión tomará como valor un array que contiene cada uno de los subarray.

5.10.2. Reducir mediante función

Número: 0116

Nombre: Función reducir array.

Categoría: Operadores sobre array.

Descripción: Se necesita de un mecanismo que dado un array y una función reduzca el array a un solo valor. La función de reducción deberá recibir como parámetro dos valores correspondiente al valor acumulado y al nuevo valor. La función de reducción se ejecutará por cada elemento del array (excepto para el primero) tomando el valor acumulado y el nuevo valor, y devolviendo el próximo valor acumulado. Como valor este operador tomará el valor de la reducción.

5.10.3. Obtener último

Número: 0117

Nombre: Función obtener último elemento de array.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array se obtenga el último elemento de este, o el valor nulo si este está vacío.

5.10.4. Obtener primero

Número: 0118

Nombre: Función obtener primer elemento de array.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array se obtenga el primer elemento de este, o el valor nulo si este está vacío.

5.10.5. Insertar en posición

Número: 0119

Nombre: Función para insertar un elemento en una posición de array.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array, un elemento y una expresión aritmética que hace de posición, se inserte el elemento en dicha posición del array. Si la posición se encuentra fuera de rango se dará un error de acceso.

5.10.6. Eliminar posición

Número: 0120

Nombre: Función eliminar elemento de array de una posición.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array y una expresión aritmética que hace de posición, se elimine el elemento que ocupa dicha posición dentro del array. Si la posición se encuentra fuera de rango se dará un error de acceso.

5.10.7. Insertar al inicio

Número: 0121

Nombre: Función insertar al inicio de un array.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array y un elemento de un tipo arbitrario, se inserte el elemento al comienzo del array.

5.10.8. Insertar al final

Número: 0122

Nombre: Función insertar al final de un array.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array y un elemento de un tipo arbitrario, se inserte el elemento al final del array.

5.10.9. Eliminar del inicio

Número: 0123

Nombre: Función eliminar del comienzo de un array.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array, se elimine el primer elemento del mismo. Como valor se tomará el elemento eliminado del array, o el valor nulo si este se encontraba vacío.

5.10.10. Eliminar del inicio

Número: 0124

Nombre: Función eliminar del final de un array.

Categoría: Operadores sobre array.

Descripción: Se precisa de un mecanismo que, dado un array, se elimine el último elemento del mismo. Como valor se tomará el elemento eliminado del array, o el valor nulo si este se encontraba vacío.

5.11. Operadores sobre expresiones regulares

5.11.1. Crear expresión regular

Número: 0125

Nombre: Operador creador de expresión regular.

Categoría: Operadores sobre expresiones regulares.

Descripción: Dada una cadena de caracteres se necesita de un operador que convierta esta en una expresión regular. El valor del operador será la expresión regular.

5.11.2. Comprobar expresión regular

Número: 0126

Nombre: Función comprobación de expresión regular.

Categoría: Operadores sobre expresiones regulares.

Descripción: Se precisa un operador que, dada una expresión regular y una cadena de caracteres, compruebe si esta pertenece al lenguaje definido por la expresión regular. Se deberá tomar como valor un dato de tipo lógico resultado de la operación.

Para que el resultado sea positivo la cadena debe pertenecer al conjunto de palabras delimitadas por la expresión regular. Si tan solo existe correspondencia parcial el resultado será negativo.

5.11.3. Buscar mediante expresión regular

Número: 0127

Nombre: Función de búsqueda estructurada.

Categoría: Operadores sobre expresiones regulares.

Descripción: Se precisa de un mecanismo que lleve a cabo una búsqueda estructurada, es decir, obtener una estructura de datos array condicionada por una expresión regular denominada "patrón de búsqueda" y una cadena de caracteres "texto" sobre la que se comprueba.

En la ejecución de este operador se deberá buscar en "texto" subcadenas que pertenezcan al conjunto definido por la expresión regular, originando un array con cada una de las coincidencias, que será el valor que tome la operación.

Adicionalmente la expresión regular podría estar formada por subexpresiones delimitadas por "()". En dicho caso se buscará en "texto" subcadenas que pertenezcan al conjunto delimitado por la expresión regular. Por cada subcadena encontrada se creará un array con las correspondencias de cada subexpresión. Cada uno de los arrays resultantes se deberán guardar en otro que será el valor que tome la operación.

Si se utiliza una expresión regular formada por subexpresiones los arrays correspondientes a cada subcadena deberán tener índices numéricos. Sin embargo debe darse la posibilidad de especificar una lista ordenada de cadenas claves para crear un array asociativo.

Además se ha de contemplar la búsqueda estructurada sobre un array de cadenas "texto".

5.12. Operadores de conversión de tipo

5.12.1. Conversión a numérico

Número: 0128

Nombre: Conversión a numérico.

Categoría: Operadores de conversión de tipo.

Descripción: Se ha de facilitar un operador que permita convertir un dato a tipo aritmético. Esta conversión se deberá realizar en función al tipo de dato origen y de la forma descrita en el requisito en el que se hace referencia al mismo. El valor que tomará la operación deberá ser el dato tras la conversión de tipos. La conversión se podrá realizar a un número entro o flotante.

5.12.2. Conversión a lógico

Número: 0129

Nombre: Conversión a lógico.

Categoría: Operadores de conversión de tipo.

Descripción: Se ha de facilitar un operador que permita convertir un dato a tipo lógico. Esta conversión se deberá realizar en función al tipo de dato origen y de la forma descrita en el requisito en el que se hace referencia al mismo. El valor que tomará la operación deberá ser el dato tras la conversión de tipos.

5.12.3. Conversión a cadena de caracteres

Número: 0130

Nombre: Conversión a cadena de caracteres.

Categoría: Operadores de conversión de tipo.

Descripción: Se ha de facilitar un operador que permita convertir un dato a tipo cadena de caracteres. Esta conversión se deberá realizar en función al tipo de dato origen y de la forma descrita en el requisito en el que se hace referencia al mismo. El valor que tomará la operación deberá ser el dato tras la conversión de tipos.

5.13. Operadores de entrada/salida

5.13.1. Escribir en la salida estándar

Número: 0131

Nombre: Escribir en la salida estándar.

Categoría: Operadores de entrada/salida.

Descripción: Se debe disponer de una serie de mecanismos que permitan llevar acabo la salida de datos. Esta permitirán que el contenido fuente pueda mostrar en la salida estándar los datos sobre los que opera. Estos datos debes tener una representación gráfica y ser imprimibles.

5.14. Leer de la entrada estándar

Número: 0132

Nombre: Leer de la entrada estándar.

Categoría: Operadores de entrada/salida.

Descripción: El intérprete debe implementar algún recurso que permita que el contenido fuente del usuario obtenga datos de la entrada estándar para operar. Este mecanismo deberá dar la posibilidad de mostrar un prompt. Además debe dar la opción de que la entrada finalice al introducir un salto de línea o al generarse una señal EOF (end-of-file).

5.15. Operadores informativos

5.15.1. Tipo de

Número: 0133

Nombre: Obtener tipo de dato.

Categoría: Operadores informativos.

Descripción: Debido al tipado dinámico se precisa de un mecanismo para conocer el tipo de dato relacionado con una variable. Este deberá mostrar en la salida estándar el tipo de la variable asociada a un identificador dado.

5.15.2. Tamaño de

Número: 0134

Nombre: Obtener tamaño de dato.

Categoría: Operadores informativos.

Descripción: Se precisa de un mecanismo para obtener el tamaño en memoria que ocupa el valor referenciado por una variable.

5.15.3. Información sobre

Número: 0135

Nombre: Obtener información interna de un dato.

Categoría: Operadores informativos.

Descripción: Se precisa de un mecanismo para obtener información estructural de un dato, cómo este es representado por el intérprete y el tamaño que ocupa.

5.16. Procesos

5.16.1. Ejecutar comando

Número: 0136

Nombre: Ejecutar comando.

Categoría: Procesos.

Descripción: Se debe facilitar un operador que ejecute un comando dado. Para ello se deberá usar el interprete de comandos definido por el entorno del sistema operativo. Este operador contemplará un único operando correspondiente al comando a ejecutar. Como valor se deberá tomar la cadena de caracteres correspondiente a la salida del comando.

5.16.2. Evaluar cadena

Número: 0137

Nombre: Evaluar cadena.

Categoría: Procesos.

Descripción: Deberá existir un operador que utilice el interprete para procesar una cadena de caracteres escrita en el léxico y con la sintaxis del propio lenguaje. Este operador tomará como valor una cadena de caracteres relativa a la salida generada por la interpretación.

5.16.3. Bifurcación de proceso

Número: 0138

Nombre: Bifurcación de proceso.

Categoría: Procesos.

Descripción: Se necesita de un mecanismo que bifurque el flujo de ejecución mediante la creación de un proceso hijo. El intérprete deberá crear un proceso clonado de si mismo, cuya ejecución proseguirá en el mismo punto. El operador de bifurcación tomará valor aritmético cero en el proceso hijo, mientras que en el padre tomará el valor aritmético correspondiente al identificador del proceso hijo.

5.16.4. Espera entre procesos

Número: 0139

Nombre: Espera entre procesos.

Categoría: Procesos.

Descripción: Se necesita de un mecanismo que permita hacer que la ejecución de un proceso padre espere a que todos o algunos de sus hijos finalicen su ejecución. Este podrá operar sobre un dato aritmético que referenciará al identificador de proceso del hijo que se ha de esperar. El valor que tomará consistirá en el código correspondiente a la señal de salida producida por el último proceso finalizado.

5.16.5. Obtener identificador de proceso

Número: 0140

Nombre: Obtener identificador de proceso.

Categoría: Procesos.

Descripción: Todo proceso tiene un identificador único en el sistema sobre el que se ejecuta. Se deberá disponer de un operador que tome el valor del identificador de proceso correspondiente al interprete.

5.16.6. Obtener identificador de proceso padre

Número: 0141

Nombre: Obtener identificador de proceso padre.

Categoría: Procesos.

Descripción: Debe existir algún mecanismo que permita obtener el identificador del proceso padre cuando el interprete se ejecuta como proceso hijo de otro.

5.16.7. Señales entre procesos

Número: 0142

Nombre: Señales entre procesos.

Categoría: Procesos.

Descripción: Debe existir algún mecanismo que permita mandar señales entre procesos. Estas señales se corresponderán con señales UNIX. Para mandar una señal a un proceso se deberá dar un identificador de proceso y un entero correspondiente a la señal a enviar.

5.16.8. Manejador de señales a procesos

Número: 0143

Nombre: Manejador de señales a procesos.

Categoría: Procesos.

Descripción: Debe existir algún mecanismo que permita especificar una función que será ejecutada cuando el proceso reciba una determinada señal.

5.16.9. Llamar a función como proceso

Número: 0144

Nombre: Llamar a función como proceso.

Categoría: Operadores sobre procesos.

Descripción: Se precisa de operador que mediante una función y un listado de parámetros realice una llamada a la misma mediante la creación de un proceso hijo.

5.17. Ficheros

5.17.1. Fichero

Número: 0145

Nombre: Fichero.

Categoría: Operadores sobre ficheros.

Descripción: El intérprete debe ser capaz de manipular ficheros, para ello se precisa de un tipo de dato que simbolice un puntero a un fichero del sistema de ficheros. Este tipo de dato no debe ser convertido a ningún otro tipo de dato ni viceversa. Además solo será tratado por algunos operadores dedicados.

No se tendrán en cuenta los ficheros binarios.

5.17.2. Abrir ficheros

Número: 0146

Nombre: Abrir ficheros.

Categoría: Operadores sobre ficheros.

Descripción: Es necesario un mecanismo que permita abrir ficheros para su manipulación. Este operará sobre una cadena de caracteres que simbolice la ruta al fichero y otra que determine el modo en el que será abierto. Se tomará como valor un dato de tipo puntero a fichero. Los posibles modos serán:

r: Lectura.

r+: Lectura y/o escritura.

w: Escritura truncando el contenido del fichero.

w+: Lectura y/o escritura truncando el contenido del fichero.

a: Escritura posicionando el puntero al final el fichero.

a+: Lectura y/o escritura posicionando el puntero al final del fichero.

Todos los modos a excepción de sólo lectura deberán crear el fichero si este no existe.

5.17.3. Cerrar ficheros

Número: 0147

Nombre: Cerrar ficheros.

Categoría: Operadores sobre ficheros.

Descripción: Es necesario un mecanismo que permita cerrar ficheros abiertos a partir de un puntero al mismo. Se deberá finalizar cualquier flujo de datos abierto y el fichero quedará cerrado. Como valor se deberá tomar un dato de tipo lógico que determine si la operación se ha realizado correctamente.

5.17.4. Escribir en fichero

Número: 0148

Nombre: Escribir en fichero.

Categoría: Operadores sobre ficheros.

Descripción: Se hace necesario un operador que, dado un dato de tipo puntero a fichero, pueda escribir datos en la posición referenciada por el mismo. Así este operador trabaja sobre dos operandos, un puntero a fichero y una cadena de caracteres que simbolizará el contenido a escribir. Como valor el operador toma el número de bytes que fueron escritos.

5.17.5. Leer de fichero

Número: 0149

Nombre: Leer de fichero.

Categoría: Operadores sobre ficheros.

Descripción: Se hace necesario un operador que, dado un dato de tipo puntero a fichero, lea desde la posición referenciada por el mismo hasta un carácter de nueva línea, o bien un número de caracteres dado. Así el operador deberá tomar como valor una cadena de caracteres que represente el contenido leído.

5.17.6. Cambiar posición de puntero a fichero

Número: 0150

Nombre: Cambiar posición de puntero a fichero.

Categoría: Operadores sobre ficheros.

Descripción: Una operación básica sobre punteros a ficheros es desplazar este dentro del contenido del mismo. Para ello se precisa de un operador que, dado un puntero a fichero, cambie la posición de este dentro del propio fichero. Así la nueva posición deberá ser una expresión numérica que represente un offset relativo al principio del fichero, el final o la posición actual del puntero. La expresión correspondiente al operador deberá tomar un valor booleano que determine si el cambio de posición se ha realizado correctamente.

5.17.7. Obtener la posición actual de puntero a fichero

Número: 0151

Nombre: Obtener la posición actual de puntero a fichero.

Categoría: Operadores sobre ficheros.

Descripción: Se necesita de un operador que dado un puntero a fichero tome el valor aritmético que represente la posición de este dentro del mismo.

5.17.8. Obtener contenido de un fichero

Número: 0152

Nombre: Obtener contenido de un fichero.

Categoría: Operadores sobre ficheros.

Descripción: Se precisa de un operador que simplifique la tarea de obtener el contenido completo de un fichero, sin que sea necesario disponer de un puntero al mismo. Para ello se deberá facilitar una cadena de caracteres que simbolice la ruta completa del fichero. El operador tomará como valor una cadena de caracteres que contenga todo el contenido del fichero. En el caso de que el fichero no exista se deberá tomar como valor la cadena vacía.

5.17.9. Cadena como contenido de un fichero.

Número: 0153

Nombre: Cadena como contenido de un fichero.

Categoría: Operadores sobre ficheros.

Descripción: Se precisa de un operador que simplifique la tarea de escribir una cadena de caracteres en un fichero, sin que sea necesario disponer de un puntero al mismo. Si el fichero existe su contenido deberá ser truncado, si no existe será creado. Este operador tendrá como operandos dos cadenas de caracteres que se correspondan con la ruta del fichero y la cadena a escribir. Como valor se tomará la cadena escrita.

5.17.10. Añadir cadena al contenido de un fichero

Número: 0154

Nombre: Añadir cadena al contenido de un fichero.

Categoría: Operadores sobre ficheros.

Descripción: Se precisa de un operador que simplifique la tarea de añadir una cadena de caracteres al final de un fichero, sin que sea necesario disponer de un puntero al mismo. Si el fichero no existe será creado. Este operador tendrá como operandos dos cadenas de caracteres que se correspondan con la ruta del fichero y la cadena a escribir. Como valor se tomará la cadena escrita.

5.18. Fechas y tiempo

5.18.1. Obtener fecha

Número: 0155

Nombre: Obtener fecha.

Categoría: Fechas y tiempo.

Descripción: Es necesario disponer de un operador que dé formato a la fecha/hora local. Este operador deberá tener como operando una expresión cadena de caracteres que contenga una serie de directivas de formato. El valor que tomará la operación será una cadena de caracteres que represente la fecha/hora en el formato dado. Las directivas de formato serán las siguientes:

%d: Día del mes con dos dígitos.

%j: Día del mes sin ceros iniciales.

%l: Día de la semana de forma alfabética completa.

%D: Día de la semana de forma alfabética y con tres letras.

%w: Día de la semana de forma numérica (0-domingo,6-sábado).

%z: Día del año de forma numérica.

%F: Mes de forma alfabética.

%m: Mes de forma numérica con dos dígitos.

%n: Mes de forma numérica sin ceros iniciales.

%M: Mes de forma alfabética con tres letras.

%Y: Año con cuatro dígitos.

%y: Año con dos dígitos.

%a: Periodo del día (am/pm) en minúsculas.

%A: Periodo del día (am/pm) en mayúsculas.

%g: Hora en formato 12h sin ceros iniciales.

%G: Hora en formato 24h sin ceros iniciales.

%h: Hora en formato 12h con dos dígitos.

%H: Hora en formato 24h con dos dígitos.

%i: Minutos con dos dígitos.

%U: Segundos desde la Época Unix (1 de Enero del 1970 00:00:00 GMT).

%%: Carácter %.

5.18.2. Tiempo Unix

Número: 0156

Nombre: Operador time.

Categoría: Fechas y tiempo.

Descripción: Se precisa de un operador que calcule el número de segundos desde la Época Unix (1 de Enero del 1970 00:00:00 GMT). Este operador no tendrá operandos y tomará el valor aritmético correspondiente.

5.18.3. Parar ejecución

Número: 0157

Nombre: Parar ejecución.

Categoría: Fechas y tiempo.

Descripción: Se deberá de proporcionar un mecanismo que permita suspender o bloquear la ejecución durante un tiempo dado. Constará de una expresión que represente el valor aritmético del tiempo en segundos.

5.19. Errores

Número: 0158

Nombre: Información de errores.

Categoría: Error.

Descripción: Si se produce un error el intérprete debe informar del tipo y causa del error, así como del contexto en el que se ha producido (nombre y línea de fichero).

5.20. Extensiones

5.20.1. Extensión

Número: 0159

Nombre: Extensión.

Categoría: Extensiones.

Descripción: La funcionalidad y características del intérprete deben ser extensible mediante módulos dinámicos. Estos módulos añadirán sentencias, operadores y demás elementos propios de un lenguaje de programación.

Para que un extensión pueda ser utilizada se deberá cargar.

5.20.2. Carga de extensiones mediante fichero de configuración

Número: 0160

Nombre: Carga de extensiones mediante fichero de configuración.

Categoría: Extensiones.

Descripción: Se debe facilitar un mecanismo que permita especificar un listado de extensiones que serán cargados al ejecutarse el intérprete. Estos módulos serán especificados en un fichero de texto plano separados mediante saltos de línea. Toda ejecución del intérprete conllevará la carga de las extensiones especificadas.

5.20.3. Carga de extensiones mediante sentencia

Número: 0161

Nombre: Carga de extensiones mediante sentencia.

Categoría: Extensiones.

Descripción: Se debe facilitar un mecanismo que permita cargar una extensión en tiempo de ejecución. Para ello se deberá facilitar la ruta del módulo correspondiente a la extensión como una cadena de caracteres. Tras la carga de la extensión las características de esta serán añadidas al intérprete.

5.20.4. Biblioteca GNU de internacionalización (gettext)

Número: 0162

Nombre: Extensión gettext.

Categoría: Extensión gettext.

Descripción: Se deberá facilitar a modo de extensión la funcionalidad y características de la biblioteca GNU de internacionalización (i18n), gettext.

5.21. Biblioteca de desarrollo OMI

5.21.1. Recursos del intérprete

Número: 0163

Nombre: Recursos del intérprete.

Categoría: Biblioteca OMI.

Descripción: Todos los recursos de desarrollo sobre los que se construye el intérprete tales como clases, funciones, etc. Deberán ser construidos para que puedan ser utilizados desde otros desarrollos, estableciéndose así un marco de trabajo para otros proyectos.

5.21.2. Interpretar desde otros proyectos

Número: 0164

Nombre: Intérpretar desde otros proyectos.

Categoría: Biblioteca OMI.

Descripción: Se debe facilitar un mecanismo para que otros proyectos software puedan interpretar código OMI de forma interna. Esto se hará mediante la llamada a una función de la biblioteca OMI.

5.22. Web del proyecto OMI

5.22.1. Sitio web

Número: 0165

Nombre: Sitio web.

Categoría: Web OMI.

Descripción: Se ha de disponer de una web que sirva de plataforma para acceder al proyecto y todos los recursos que este brinda. Esta web tendrá una estructura estática.

5.22.2. Home del sitio web

Número: 0166

Nombre: Home del sitio web.

Categoría: Web OMI.

Descripción: La web OMI debe dispone de una página de entrada en la que se recoge:

- Una descripción resumida del proyecto.
- Enlaces a las secciones principales.
- Un listado de noticias que será mantenida manualmente por el administrador
- Enlaces a las descargas de la última versión de los recursos.

5.22.3. Sobre OMI

Número: 0167

Nombre: Sobre OMI.

Categoría: Web OMI.

Descripción: Deberá recoger información sobre el proyeto, el alcance del mismo, la autoría y los organismos implicados.

5.22.4. Contacto

Número: 0168

Nombre: Contacto.

Categoría: Web OMI.

Descripción: deberá listar diferentes medios de contacto.

5.22.5. Índice de documentación

Número: 0169

Nombre: Índice de documentación.

Categoría: Web OMI.

Descripción: Se ha de dispone de una página en la que se enlace de forma ordenada las distintas secciones de la documentación, así como las distintas herramientas para navegar por esta.

5.22.6. Página de documentación

Número: 0170

Nombre: Índice de documentación.

Categoría: Web OMI.

Descripción: Cada documento relativo a la presente memoria deberá estar disponible desde la web del proyecto.

5.22.7. Características

Número: 0171

Nombre: Características.

Categoría: Web OMI.

Descripción: Se ha de disponer de una página en la que se liste las características del intérprete.

5.22.8. Navegador de gramática

Número: 0172

Nombre: Gramática.

Categoría: Web OMI.

Descripción: Se ha de disponer de una página que sirva para navegar por las reglas gramaticales que definen el intérprete.

5.22.9. Navegador de clases

Número: 0173

Nombre: Clases.

Categoría: Web OMI.

Descripción: Se ha de disponer de una página que sirva para navegar por las clases de programación sobre las que se construye el intérprete.

5.22.10. Navegador de archivos

Número: 0174

Nombre: Archivos .

Categoría: Web OMI.

Descripción: Se ha de disponer de una página que sirva para navegar por los archivos de código fuente que sirven para construir el intérprete.

5.22.11. Wiki

Número: 0175

Nombre: Wiki .

Categoría: Web OMI.

Descripción: Se ha de disponer de una wiki para el proyecto que recoja aspectos relativos al proyecto y las materias que este estudia.

5.23. RunTree

5.23.1. Intérprete online

Número: 0176

Nombre: Intérprete online.

Categoría: RunTree.

Descripción: Se ha de disponer de una herramienta que permita hacer uso del intérprete desde el navegador. Esta herramienta tomará un código fuente escrito en el lenguaje de programación y lo enviará a un intérprete OMI para su procesamiento. Luego mostrará, paso a paso, información relativa al proceso de interpretación.

5.23.2. Escribir código fuente

Número: 0177

Nombre: Escribir código fuente.

Categoría: RunTree.

Descripción: La herramienta deberá de disponer de un campo de formulario en el que se pueda escribir el código fuente que será enviado para interpretar. Este campo deberá mostrar un resaltado de sintaxis.

5.23.3. Árbol sintáctico

Número: 0178

Nombre: Árbol sintáctico.

Categoría: RunTree.

Descripción: La herramienta deberá de mostrar el árbol sintáctico resultado de la interpretación y permitir la navegabilidad por este.

5.23.4. Tabla de símbolos

Número: 0179

Nombre: Tabla de símbolos.

Categoría: RunTree.

Descripción: La herramienta deberá de mostrar el estado de las tablas de símbolos en cada paso del proceso.

5.23.5. Salida de datos

Número: 0180

Nombre: Salida.

Categoría: RunTree.

Descripción: La herramienta deberá mostrar la salida generada por cada paso del proceso.

5.23.6. Entrada de datos

Número: 0181

Nombre: Entrada de datos.

Categoría: RunTree.

Descripción: La herramienta deberá solicitar al usuario la entrada de datos cunado el proceso de interpretación lo requiera.

5.23.7. Limpieza de salida

Número: 0182

Nombre: Limpieza de salida.

Categoría: RunTree.

Descripción: La herramienta deberá dar la posibilidad de limpiar la salida generada por el proceso de interpretación.

5.23.8. Consola de información

Número: 0183

Nombre: Consola de información.

Categoría: RunTree.

Descripción: La herramienta deberá de disponer de una sección en la que se muestre textualmente la operación llevada a cabo.

5.23.9. Ejecución paso a paso

Número: 0184

Nombre: Paso a paso.

Categoría: RunTree.

Descripción: La herramienta deberá de disponer de acciones que permitan avanzar la ejecución un paso.

5.23.10. Ejecución sentencia a sentencia

Número: 0185

Nombre: Sentencia a sentencia.

Categoría: RunTree.

Descripción: La herramienta deberá de disponer de acciones que permitan avanzar la ejecución una sentencia.

5.23.11. Ejecución automática

Número: 0186

Nombre: Ejecución automática.

Categoría: RunTree.

Descripción: La herramienta deberá de disponer de la capacidad de ejecutar paso a paso el programa pero de una forma automática.

5.23.12. Guardar código fuente

Número: 0187

Nombre: Guardar código fuente.

Categoría: RunTree.

Descripción: La herramienta deberá de disponer de la capacidad de guardar el código fuente escrito en un fichero local.

5.23.13. Abrir código fuente

Número: 0188

Nombre: Abrir código fuente.

Categoría: RunTree.

Descripción: La herramienta deberá de disponer de la capacidad de leer el código fuente escrito en un fichero local para enviarlo a interpretar.

6. Requisitos no funcionales

6.1. Rendimiento

Es condición necesaria que el sistema software desarrollado presente un rendimiento óptimo. Para ello se medirá el rendimiento desde dos aspectos: tiempo y espacio

6.1.1. Tiempo

A pesar de que el objetivo del interprete no es servir para la producción de software, al ser un sistema que será utilizado para el desarrollo de otras aplicaciones, el rendimiento en cuanto al tiempo debe ser aceptable en comparación con las herramientas existentes en el mercado.

6.1.2. Espacio

El sistema debe ser óptimo en cuanto el espacio en memoria que ocupan sus estructuras. Debe hacer un uso correcto de la memoria. Si las estructuras que conforman el intérprete ocupan demasiado espacio, los datos de los que hagan uso el usuario en sus programas también lo harán.

6.2. Usabilidad

6.2.1. Reglas léxicas y sintácticas

El sistema deberá interpretar un lenguaje con reglas léxicas y sintácticas claras, que sean fáciles de comprender y asimilar. La estructura del lenguaje en cuanto a estos dos aspectos debe ser similar a los lenguajes de programación actuales.

6.2.2. Interfaz

La interfaz de entrada salida del interprete debe ser clara y presentar las opciones de una forma adecuada. Además la interfaz web debe ser amigable a la forma de operara del usuario que la utilice.

6.3. Accesibilidad

El sistema desarrollado debe ser accesible desde cualquier computadora con acceso a internet. Para ello se utilizará un navegador web. Por otro lado el interprete podrá ser instalado de forma local para un uso individual.

6.4. Estabilidad

Se requiere que el sistema desarrollado presente un umbral de fallos bajo. Debe adaptarse a los casos excepcionales, y en caso de error informar adecuadamente de los motivos y causas del mismo.

6.5. Mantenibilidad

El sistema desarrollado debe ser mantenible en el tiempo. Para ello debe estar correctamente documentado, modularizado y estructurado.

6.6. Concurrencia

El sistema deberá ser accesible por varios usuarios en el mismo marco de tiempo. Es por ello que deberá ser capaz de funcionar en un entorno de concurrencia.