Cubik: Una herramienta de apoyo en la enseñanza de la Programación.

Juan Cristóbal García - Perla Señas - Norma Moroni

Departamento de Ciencias de la Computación
Instituto de Ciencias e Ingeniería de Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina.
e-mail: jcgarcia@uns.edu.ar, jgarcia@uncoma.edu.ar,
ccsenas@criba.edu.ar, ccmoroni@criba.edu.ar

Resumen

Se presenta un entorno para el desarrollo de Algoritmos. Este entorno controla con naturalidad la edición de forma tal que no se incurra en errores sintácticos ni errores semánticos. Permite usar del nombre de otros Algoritmos como primitivas lo que promueve el concepto de modularidad. Además el entorno soporta que los Algoritmos pueden ser traducidos a programas en lenguaje Pascal los cuales luego pueden ser ejecutados. Esta característica permite la prueba de los Algoritmos editados. El Editor maneja datos de entrada, salida, y datos locales. Asimismo maneja *acciones* para modificar los datos, estas acciones son representaciones abstractas de las sentencias existentes en la mayoría de los Lenguajes de Programación Estructurados.

1. Introducción

Cubik es un Editor de Algoritmos Estructurados que fue desarrollado en base al trabajo de la Lic. Norma Moroni y la Prof. Perla Señas, "Un entorno para el Aprendizaje de la Programación".

Se pone énfasis en el uso de una interfaz amigable, gráfica y compatible con los nuevos Sistemas Operativos orientados a ventanas.

La aplicación se desarrolló con una metodología orientada a objetos debido a su conveniencia en el manejo de las interfaces gráficas. Se utilizó la Librería de Objetos que provee el paquete Borland Pascal 7.0 en su OWL(Object Windows Library).

El objeto principal que provee su interface es la Ventana de Edición de Algoritmos. Dicha Ventana se encarga de realizar el Análisis Léxico, Sintáctico y Semántico de los Algoritmos cada vez que se agrega o elimina un elemento. Cuando el Analizador Sintáctico detecta que se ha ingresado una palabra clave para una estructura, agrega por sí misma el resto de la estructura de forma tal que resulta imposible la aparición de un error sintáctico. El mismo proceso ocurre cuando se ingresa un operador en una expresión.

Una vez logrado un algoritmo correcto, el Traductor asociado lo puede traducir a un subprograma Pascal.

El otro objeto que se utiliza es la Ventana de Edición de Textos que simplemente permite ver y modificar los subprogramas Pascal generados por el Traductor a partir de los Algoritmos creados en la Ventana de Algoritmos.

Cubik permite el manejo de Múltiples Ventanas, tanto de Edición de Algoritmos como de Edición de Textos. En la Ilustración 1 se muestra un ejemplo del Entorno Cubik con una Ventana de Edición de Algoritmos y una Ventana de Edición de Textos.

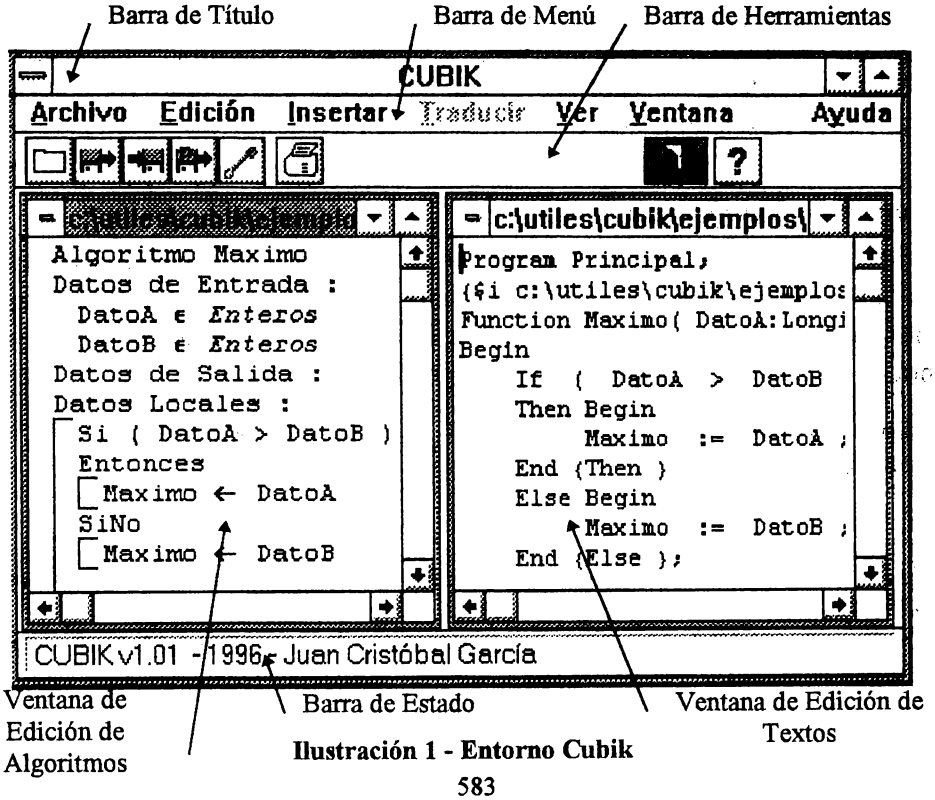

Ilustración 1 - Entorno Cubik

Dentro de la Ventana de Edición de Algoritmos existen elementos que pueden ser reemplazados, estos elementos se denominan *sugerencias*. La palabra **Acción** puede ser reemplazada por palabras claves o Hot Keys que representen Acciones Simples o Estructuradas. Existen sugerencias que representan Expresiones Libres (**Exp.**) que son expresiones las cuales no pertenecen a ningún conjunto determinado; Expresiones Enteras, Reales y Lógicas (**Exp.Ent.**, **Exp.Rea.**, **Exp.Lóg.** respectivamente); Caracteres (**Carácter**); Nombres de Datos (**Nombre**) y algunas más que se verán posteriormente.

2. El Entorno Cubik

2.1 El Lenguaje Algorítmico.

El Lenguaje Algorítmico es aquel que se utiliza para la descripción del Algoritmo. Se considera al Algoritmo compuesto por una sección de Declaraciones seguida de una sección de Acciones. El Lenguaje Algorítmico que acepta Cubik, consta de Acciones Simples como Asignaciones o invocaciones a otros Algoritmos y de Acciones Estructuradas tales como: Bloque de Acciones (Secuencia de dos o más Acciones), Acciones Condicionales Simples (Si *Condición* Entonces *Acción*), Acciones Condicionales Compuestas (Si *Condición* Entonces *Acción1* Sino *Acción2*), Acciones Repetitivas (Repetir Desde *LímiteInferior* Hasta *LímiteSuperior Acción*; Repetir Mientras *Condición Acción* y Repetir *Acción* Hasta *Condición*).

Las Asignaciones, las invocaciones a otros Algoritmos y las Condiciones hacen uso de expresiones. Dentro de las expresiones se utilizan *nombres*, los cuales representan datos que estarán asociados a algún conjunto en particular y que además soportan algún tipo de operación sobre ellos. Los conjuntos a los cuales se puede asociar un dato en Cubik son **Enteros**, **Reales**, **Caracteres** y **Lógicos.** Además se pueden definir **Sucesiones** de elementos pertenecientes a cualquiera de los conjuntos anteriores.

Los datos en un algoritmo pueden ser Datos de Entrada, que representan los valores que recibe el Algoritmo en primera instancia, Datos Locales, que se utilizan como auxiliares en la manipulación de los Datos de Entrada y los Datos de Salida que son el resultado del procesamiento de los Datos de Entrada.

2.2 Deducción de Conjuntos

Por deducción de conjuntos se entiende al análisis de una *expresión* para determinar a qué conjunto pertenece o si no pertenece a ninguno específicamente. Cubik realiza deducción de conjuntos cuando se especifica un nombre a la izquierda de una asignación y cuando se agrega un nuevo operador u operando en una expresión.

En el momento de especificar el nombre a la izquierda de una asignación, pueden ocurrir cuatros casos diferentes:

1. El nombre pertenece a algún conjunto y la expresión del lado derecho también pertenece a algún conjunto.
2. El nombre no pertenece a ningún conjunto aún y la expresión del lado derecho pertenece a algún conjunto.

3. El nombre pertenece a algún conjunto y la expresión del lado derecho no pertenece a ningún conjunto.
4. El nombre no pertenece a ningún conjunto y la expresión del lado derecho no pertenece a ningún conjunto.

En el caso 1, se deduce el conjunto al asociado al lado derecho y se verifica que sea compatible con el conjunto al cual pertenece el nombre del lado izquierdo. La asignación se permitirá sólo si los conjuntos asociados a ambos lados de la asignación coinciden o si la expresión del lado derecho pertenece a los Enteros y el nombre del lado izquierdo pertenece a los Reales.

En el caso 2, se deduce el conjunto del lado derecho y se registra al nombre como perteneciente a dicho conjunto. Si es un dato de Entrada y/o Salida ya definido, se actualiza en la lista de Declaraciones, si no se agrega una nueva declaración de Dato Local para dicho nombre.

El caso 3 se puede dar únicamente cuando la expresión del lado derecho está conformada sólo por la sugerencia **Exp.** que representa a una expresión que no pertenece a ningún conjunto aún. En este caso, la sugerencia es reemplazada por otra sugerencia que indica el conjunto al cual pertenece el nombre.

En el caso 4 no se realiza ninguna deducción de tipos, postergándose hasta que se posea más información.

En el caso de reemplazar un operador o un operando sobre una expresión, se pueden dar los siguientes casos:

1. El elemento insertado es un nombre que aún no tiene asociado un conjunto y tampoco se ha determinado el conjunto correspondiente a la expresión.
2. El elemento insertado es un nombre que aún no tiene asociado un conjunto y ya se ha determinado el conjunto correspondiente a la expresión.
3. El elemento insertado pertenece a algún conjunto (puede ser un nombre, una constante o un operador) y aún no se ha determinado el conjunto correspondiente a la expresión.
4. El elemento insertado pertenece a algún conjunto (puede ser un nombre, una constante o un operador) y ya se ha determinado el conjunto correspondiente a la expresión.

En el caso 1, se genera un error ya que si se permite que el nombre reemplace a la expresión, posteriormente no será posible deducir directamente el conjunto del nombre.

En el caso 2, se deduce el conjunto de la expresión y se registra el nombre como perteneciente a dicho conjunto en la lista de Declaraciones.

En el caso 3, directamente se reemplaza la expresión (que debe ser una sugerencia **Exp.**) por el elemento insertado si el elemento es un nombre o una constante o bien por el elemento insertado y otros elementos adicionales si es un operador.

En el caso 4, se deduce el conjunto de la expresión y se verifica que sea compatible con el conjunto al que pertenece el elemento insertado. El reemplazo será permitido sólo si el conjunto del elemento insertado es igual al conjunto de la expresión o bien si el conjunto del elemento insertado es el de los Enteros y el conjunto de la expresión el de los reales.

La forma en que se deduce el conjunto de una expresión es el siguiente.

- Se analiza cada operador y operando en la expresión.
- Si la expresión contiene algún operador relacional, entonces la expresión pertenece al del los Lógicos.
- De lo contrario, si existe algún operando con conjunto asociado entonces la expresión pertenece a dicho conjunto.
- De lo contrario, si existe alguno de los operadores **Y, O, NO,** la expresión pertenece al conjunto de los Lógicos.
- De lo contrario, si existe alguno de los operadores +, -, *, y los operandos son todos constantes enteras, nombres enteros, o bien existe alguno de los operandos **MOD, DIV,** la expresión pertenece al conjunto de los Enteros
- De lo contrario, si existe alguno de los operadores +, -, *, /, la expresión pertenece al conjunto de los Reales.
- De lo contrario no se puede deducir el conjunto de la expresión, es decir se deduce una expresión libre.

2.3 Edición de los Algoritmos

El Editor de Algoritmos es el elemento principal dentro de Cubik. Utiliza una ventana como la que se muestra en la Ilustración 2 que permite crear o modificar algoritmos. Un algoritmo es representado con *elementos editables* los cuales pueden ser reemplazados y por *elementos fijos*. Éstos últimos se caracterizan por tener color Azul Oscuro, los elementos editables son de alguno de estos colores: Blanco (Acciones), Azul Claro (Nombres de Datos o de Algoritmos) , Verde Lima (Caracteres) o Rojo (Números). Cada elemento dentro de la Ventana de Algoritmos puede ser señalado por otro elemento llamado Punto de Inserción, éste puede moverse con las teclas de movimiento o bien ubicando el cursor del ratón y presionado el botón izquierdo.

2.3.1 Reemplazando elementos editables

Para crear un algoritmo, se van reemplazando *elementos editables*, los elementos editables pueden ser *sugerencias* o los reemplazos de las sugerencias, es decir una vez que se ha reemplazado una sugerencia, el elemento obtenido puede ser reemplazado nuevamente.

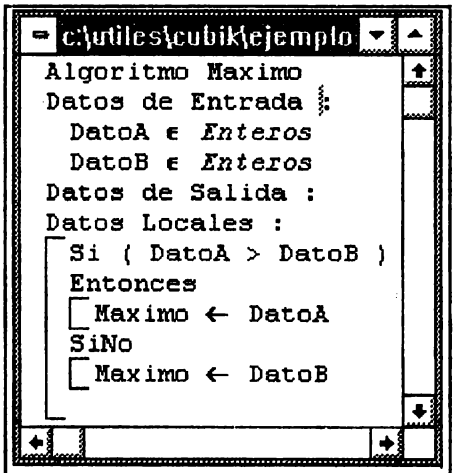

Ilustración 2 Ventana de Edición de Algoritmos

Para reemplazar un elemento editable se debe colocar el punto de inserción sobre el éste y comenzar a escribir. Se borrará el valor anterior del elemento y se comenzará a ver lo que se escribe. Para finalizar, se puede presiona *Enter*, la barra espaciadora o se hace un doble click con el ratón fuera del área de edición. El nuevo valor será analizado por Cubik, si es aceptado, se reemplazará por alguno de los valores en la Tabla 1 de lo contrario, si se ha incurrido en un error, se notificará de ello y se restaurará el elemento a su valor anterior.

Los diferentes tipos de sugerencias se muestran en la Tabla 1.

Sugerencia	Puede ser Reemplazada por	Efecto resultante
Acción	Si	Acción Condicional. Para obtener una Acción condicional simple, es decir una estructura *Si-Entonces*, se puede eliminar el Sino sobrante ubicando el punto de inserción sobre dicho elemento y presionando Delete o Eliminar del menú Edición.
	Repetir	Menú de selección de alguno de los tres tipo de Acciones repetitivas.
	Mientras	Acción Repetir Mientras.
	Hasta	Acción Repetir Hasta
	Desde	Acción Repetir Desde Hasta.
	Ctrl+A	Acción de Asignación.
	Un nombre	Invocación al algoritmo con dicho nombre.
Exp.	Operador Aritmético	Expresión Aritmética, es decir se expande el operador más los operandos necesarios.
	Operador Lógico	Expresión Lógica, es decir se expande el operador Lógico más los operandos necesarios.
	Operador Relacional	Expresión Lógica, se expande el operador relacional más dos sugerencias de expresiones a ser comparadas.
	Un nombre	El nombre puede corresponder a un dato de entrada, salida o local o bien al nombre de un algoritmo que devuelva algún valor. Si el nombre corresponde a una sucesión, se genera una referencia a alguno de sus elementos, si el nombre es el de un algoritmo, se genera una invocación a dicho algoritmo.
Exp.Arit. **Exp.Ent.** **Exp.Real.**	Operador Aritmético	El operador más los operandos requeridos para dicho operador.
	Un nombre	Si es una dato local, de entrada o de salida, el nombre mismo, si es un algoritmo, se genera una llamada al algoritmo. Si el nombre corresponde a una sucesión, se genera una referencia a alguno de sus elementos.
Exp.Log.	Operador Lógico	El operador lógico más los operando requeridos para el operador.
	Operador Relacional	El operador relacional más las expresiones que se compararán.
	Un nombre	Si es una dato local, de entrada o de salida, el nombre mismo, si es un algoritmo, se genera una llamada al algoritmo. Si el nombre corresponde a una sucesión, se genera una referencia a alguno de sus elementos.
Declaración	Un nombre	Se reemplaza por el nombre mismo seguido del símbolo $\in$, seguido de la sugerencia **Conjunto**
Nombre	Un nombre	El nombre mismo.
Carácter	Comilla simple	Se reemplaza por **'C.C.'** para poder ingresar una constante carácter.
	Un nombre	Si es una dato local, de entrada o de salida, el nombre mismo, si es un algoritmo, se genera una llamada al algoritmo. Si el nombre corresponde a una sucesión, se genera una referencia a alguno de sus elementos

Tabla 1 Reemplazos de Sugerencias

Sugerencia	Puede ser Reemplazada por	Efecto resultante
C.C.	Un carácter	El carácter mismo
Suc.Ent. Suc.Log. Suc.Rea. Suc.Car.	Un nombre	El nombre mismo. El nombre debe ser el de una sucesión de elementos del tipo indicado.
Iter.	Un nombre	El nombre mismo
Número	Un número	El número mismo.
Conjunto	Nombre de Conjunto	Los nombres posibles son Enteros, Reales, Lógicos, Caracteres. Se reemplaza por el nombre mismo.
	Sucesión	Se reemplaza por ***es una sucesión de Número Elementos.***
Elementos	Nombre de Conjunto	Los nombres posibles son Enteros, Reales, Lógicos, Caracteres. Se reemplaza por el nombre mismo.

Tabla 1 Reemplazos de Sugerencias (*Continuación*)

2.3.2 Insertando nuevas sugerencias

Se pueden insertar nuevas sugerencia en el algoritmo para poder ser reemplazadas y extender el algoritmo. Las sugerencias que pueden ser insertadas son **Acción**, lo que permite definir un secuencia de acciones, y **Declaración**, la cual permite agregar una nueva declaración para un dato de entrada, salida o local.

Para insertar una nueva **Acción** se ubica el punto de inserción sobre algún elemento y se presiona *Insert* (o Insertar\Nuevo Elemento en la Barra de Menú o Insertar en el Menú Flotante). Dependiendo de la posición donde se ubique el punto de inserción, se insertará encima o debajo de la línea actual. Si se ubica al principio de la línea (o a la izquierda del primer elemento en la línea), se insertará encima de la línea actual, si se ubica al final (o sobre algún elemento en la línea) se insertará debajo de la línea actual.

La inserción no tendrá efecto si el primer elemento en la línea actual corresponde a alguna acción estructurada y se ha ubicado el punto de inserción sobre dicho elemento.

2.3.3 Eliminando elementos

Cada elemento puede ser eliminado de una manera coherente, si se ubica el punto de inserción sobre el elemento en cuestión y se presiona Delete (o Eliminar en el menú flotante), se reemplazará el elemento por una sugerencia que indica cuáles serán los elementos que se aceptarán.

En las expresiones, la única forma de eliminar una operación es que ambos operandos ya hayan sido eliminados, cuando se elimina la operación, esta es reemplazada por una sugerencia correspondiente al tipo de expresión al que pertenece la operación.

2.4 La Lista de Algoritmos Incluidos

La caja de diálogo que se muestra en la Ilustración 3 permite agregar y/o eliminar algoritmos para que puedan ser usados como acciones simples desde el algoritmo que se está editando actual mente. Cada algoritmo tendrá una lista de algoritmos asociada. Estos algoritmos son traducidos completamente cuando se traduzca el algoritmo principal, por ello se requiere que los algoritmos incluidos tengan al

menos el encabezamiento definido.

Para insertar un nuevo algoritmo, seleccionar el nombre del archivo, en la lista de Archivos, posteriormente presionar el botón Agregar, el algoritmo se sumará a la lista de Algoritmos Incluidos. Para quitar un algoritmo, seleccionar el nombre del algoritmo de la lista de Algoritmos Incluidos y posteriormente presionar el botón Quitar, automáticamente el algoritmo es borrado de la lista de Algoritmos Incluidos. Asimismo cualquier referencia a un algoritmo que se quita de la lista, es borrada en el algoritmo.

Para Actualizar la información referente a un algoritmo ya insertado, se procede de igual manera que cuando se lo insertó por primera vez, Cubik se encarga de consultar si se desea ejecutar dicha operación.

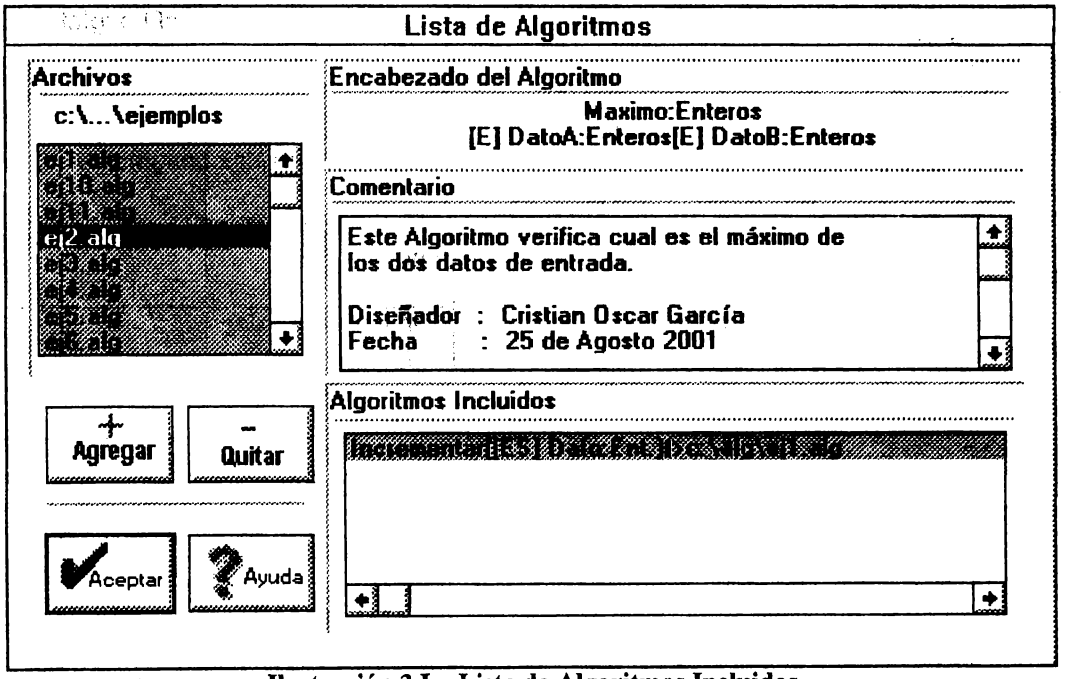

Ilustración 3 La Lista de Algoritmos Incluidos

2.5 Traducción de los Algoritmos

2.5.1 El Traductor

Cuando se termina de editar un algoritmo, éste puede ser traducido automáticamente a Lenguaje Pascal. Cada acción del algoritmo se traduce en una sentencia de dicho lenguaje. Si la acción es una invocación a otro algoritmo, el Traductor, se encarga de insertar la traducción correspondiente a dicho algoritmo como una declaración de procedimiento o función, dependiendo según el caso. En la Tabla se muestran trozos de algoritmos y su correspondiente traducción.

Porción de Algoritmo	Traducción
Algoritmo Ejemplo1 Datos de Entrada : A ∈ Enteros B ∈ Enteros Datos de Salida : B ∈ Enteros Datos Locales : C ∈ Reales D ∈ Caracteres E ∈ Lógicos F es una sucesión de 12 Enteros	Procedure Ejemplo1(A:LongInt;var B:LongInt); Var C:Real; D:Char; E:Boolean; F:array[0..11] of LongInt; ...

Algoritmo Ejemplo2
Datos de Entrada :
Datos de Salida :
Datos Locales :
Ejemplo2 ← 1
	Function Ejemplo2:LongInt;
	Begin
	Ejemplo2:=1
	End;

Repetir Desde I = 0 Hasta I = 11
A ← A + I
	For y:=0 to 11 do
	Begin
	A:=A+I
	End;

Repetir A ← A + 2 Hasta (A > 200)	Repeat A:=A+2 Until (A>200);

Repetir Mientras (A < 200)
A ← A + 2
	While (A<200) do
	Begin
	A:=A+2
	End;

Algoritmo Ejemplo3
Datos de Entrada :
A es una sucesión de 10 Reales
Datos de Salida :
A es una sucesión de 10 Reales
Datos Locales :
I ∈ Enteros
Repetir Desde I = 0 Hasta I = 9
A [I] ← A [I] + 1
	Procedure Ejemplo3(var A:array of Real);
	Var
	I:LongInt;
	Begin
	For I:=0 to 9 do
	Begin
	A[I]:=A[I]+1
	End;
	End;

Si (A = 0)
Entonces
Factorial ← 1
SiNo
Factorial ← Factorial (A - 1) * A
	if (A=0)
	then begin
	Factorial:=1
	end
	else begin
	Factorial:=Factorial(A-1)*A
	end;

Tabla 2 Ejemplos de Traducción

Porción de Algoritmo	Traducción

Algoritmo Potencia
Datos de Entrada :
X ∈ *Reales*
N ∈ *Enteros*
Datos de Salida :
Datos Locales :
Acumulador ∈ *Reales*
Acumulador ← 1
Repetir Mientras (N > 0)
Acumulador ← Acumulador * N
Decrementar (N)
Potencia ← Acumulador
	Program Principal;
	{$i c:\utiles\cubik\ejemplos\ej5.inc}
	Procedure Decrementar(var Dato:Longint);
	Begin
	Dato := Dato - 1 ;
	End {Decrementar};
	Function Potencia(X:Real; N:Longint):Real;
	Var
	Acumulador:Real;
	Begin
	Acumulador := 1 ;
	While (N > 0) Do
	Begin
	Acumulador := Acumulador * N
	;
	Decrementar (N) ;
	End {While };
	Potencia := Acumulador ;
	End {Potencia};
	Begin {Programa Principal}
	if CargarDatos
	then begin
	Var_Potencia:=Potencia(Var_X,Var_N);
	MostrarSalida;
	end;
	End. {Programa Principal}

Tabla 2 Ejemplos de Traducción (*Continuación*)

La estructura general de un programa traducido se muestra en la Tabla 3

Programa Pascal.	Explicación
;ram Principal;	Declaración del programa.
Archivo.inc}	Inclusión del archivo con rutinas de soporte.
ıducción de Algoritmos}	Traducción del algoritmo y cada uno de los algoritmos que son usados por el algoritmo principal. Los algoritmos se traducen como se muestra en la Tabla .
in {Programa Principal}	Comienzo del programa principal.
if CargarDatos then begin	Se consultan los datos de entrada. Sin no hay ningún error en la entrada, se pasa a la siguiente instrucción.
{ Ejecución del Algoritmo} { Principal}	Se ejecuta el algoritmo principal. Si el algoritmo principal devuelve algún valor, entonces se asigna a una variable para ser mostrada en la siguiente instrucción.
MostrarSalida; end;	Se muestran los resultados.
.. {Programa Principal}	Fin del programa principal.

Tabla 3 Ejemplo del programa Pascal Genrado

2.5.2 Nombres de Archivos y de Algoritmos

Cada algoritmo debe tener un nombre para identificarlo. Un algoritmo se almacena en un archivo el cual contendrá un nombre que debe adaptarse a las restricciones del sistema operativo. De esta manera cada algoritmo tendrá dos nombres asociados, el nombre del algoritmo mismo y el nombre del archivo donde se almacena.

2.5.3 Proceso de Traducción

Primeramente es necesario que el archivo *visual.inc* se encuentre en el directorio de Cubik. Este archivo se provee en la instalación por lo tanto la mayoría de las veces no será necesario tenerlo en cuenta. Otro archivo importante es el archivo *visual.res*, el cual es un archivo que contiene los recursos usados en la creación de la aplicación que permitirá la ejecución del algoritmo. Este archivo debe estar ubicado en el directorio de trabajo el cual es solicitado al usuario cuando se instala Cubik o bien se puede cambiar desde el entorno desde el menú Ver/Opciones.

Si el archivo que contiene el algoritmo tiene el nombre *archivo1.alg*, en la traducción Cubik generará los archivos *archivo1.pas* que contiene el la traducción del algoritmo y *archivo1.inc* que contiene todas las rutinas de soporte para la ejecución del algoritmo. Ambos archivos son almacenados en el directorio de trabajo.

2.5.4 Ejecución de programas

Una vez generados los dos archivos, es necesario compilarlos para poder ejecutarlos, para ello se debe abrir el entorno Borland Pascal 7.0 y se debe colocar como archivo principal (Primary File) en el menú de Compilación (Compile) el nombre del archivo con extensión *.pas* que ha generado Cubik.

Es necesario que el entorno Borland Pascal tenga acceso a su Librería de Objetos para Windows (OWL) para tener éxito en la compilación.

El módulo ejecutable del algoritmo se podrá crear con el comando *Make* del menú de compilación y ejecutar con el comando *Run*.

2.6 La interface de Cubik

2.6.1 La Barra de Menú

A continuación se enumeran cada menú y submenú de la Barra de Menú.

Archivo: Nuevo Algoritmo, Abrir Algoritmo, Abrir Programa Pascal, Guardar, Guardar Como, Cerrar, Imprimir, Especificar Impresora, Abandonar, Acerca De.

Edición: Deshacer, Cortar, Copiar, Pegar.

Insertar: Nuevo elemento, Repetir(*Mientras, Desde Hasta, Hasta*), Si, Asignación, Operadores Aritméticos(+,-,/,*,DIV,MOD), Operadores Relacionales(<, >, <=, >=, <>, =), Operadores Lógicos(Y, O y NO).

Traducir

Ver: Comentario, Lista de Algoritmos, Opciones.

Ventana: Cascada, Mosaico, Arreglar Iconos, Cerrar todo.

Ayuda

2.6.2 El Menú Flotante.

El menú flotante se despliega cuando se presiona el botón derecho del ratón, desde este menú es posible acceder a la mayoría de los comandos de la barra de menú pero de una forma más directa.

Además, el menú flotante es sensible al contexto, dependiendo de la posición dentro del algoritmo donde se despliegue, mostrará distintos items.

En la Ilustración 4 se muestra cada una de las posibles configuraciones del Menú Flotante, en (a) se puede ver el Menú Flotante General, en (b) el Menú Flotante para Conjuntos, y en (c) se menú flotante cuando se reemplaza una sugerencia **Acción** con "repetir".

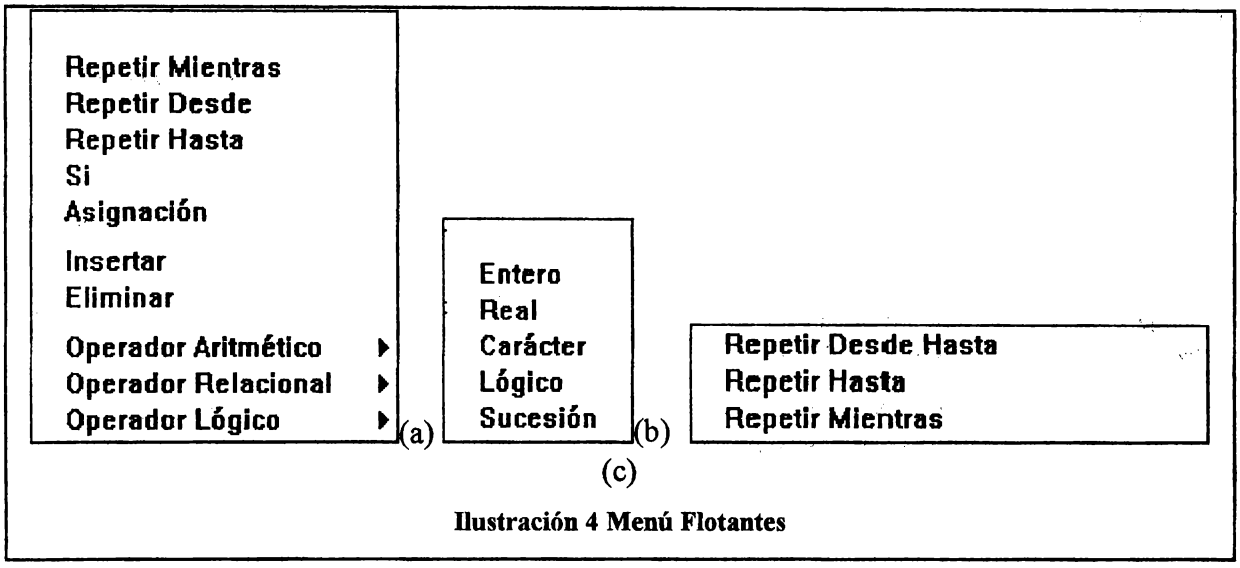

Ilustración 4 Menú Flotantes

2.6.3 La Barra de Herramientas

La barra de herramientas permite un acceso aún más directo a los comando más utilizados en la edición de Algoritmos y en la edición de textos así como los comandos utilizados para el manejo de archivos. A continuación se detallan los comandos asociados a cada botón en la barra de herramientas.

 Nuevo Algoritmo.

 Abrir Algoritmo.

 Guardar Algoritmo.

 Abrir Programa Pascal.

 Cerrar.

 Imprimir.

 Traducir.

 Ver Lista de Algoritmos Incluidos.

 Ver Comentario.

 Salir del Entorno.

 Ayuda.

3. Conclusiones

En el Desarrollo del Entorno Cubik se han aplicado técnicas tradicionales de Traducción, tanto de Compilación como de Interpretación. Se ha hecho uso de Técnicas Orientadas a Objetos para el Diseño y la Implementación. Se han usado Librerías de Interfaces Gráficas provistas en el entorno de programación Borland Pascal 7.0. mediante la aplicación de la herencia y el uso directo de las clases.

El Entorno Cubik solo representa un subconjunto de las funciones descriptas en [Moroni], pero representa una herramienta útil para la enseñanza de la programación estructurada básica.

Las propuestas de mejoras y extensiones son las siguientes: Ejecución y trazas de Algoritmos, Mejora en la edición del Algoritmo (Cortar y Pegar porciones del Algoritmo en una forma coherente) y almacenamiento de más de un algoritmo por archivo.

4. Bibliografía

[Aho] Aho, Alfred; Sethi, Ravi and Ullman, Jeffrey. *Compilers: Principles, Techniques and Tools*. Addison-Wesley. 1986.

[Borland] Borland International Inc. *Object Windows Programming Guide.* 1992

[Chavey] Darah Chavey. Beloit College. *A Structured Laboratory Component for the Introductory Programming Course*. SIGCSE BULLETIN ACM. 1991.

[Levy] Lisa Levy Kortright. *From Specific Problem Instances to Algoritms in the Introductory Course*. SIGCSE BULLETIN ACM. 1994.

[Moroni] Moroni, Norma y Señas, Perla.*Un entorno para el aprendizaje de la programación.* Enviado al 4to. Ateneo de Profesores Universitarios de Computación. 1996.

[Pittman] Pittman, Thomas; Peters, James; *The Art of Compiler Design, Theory and Practice*. University of Arkansas. Prentice-Hall International.1992.

[Swan] Tom Swan. *Turbo Pascal for Windows 3.0 Programming.* Bantam Books1991