TE
UVG
Comp.
D67
1989
c. 2

UNIVERSIDAD EL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

DISEÑO E IMPLEMENTACION DE UN GENERADOR DE INTERFASES BASADO EN LENGUAJE NATURAL

LUIS FERNANDO DIAZ VILLALTA

Trabajo de investigación presentado para optar
al grado académico de
Ingeniero en Computación
en el grado de Licenciado

Guatemala

1989

Vo. Bo. :

(f)_____
Licenciado Rodrigo Arias Palomo
Asesor

Tribunal:

(f)_____
Licenciado Rodrigo Arias Palomo

(f)_____
Ingeniero Luis Furlán

(f)_____
Licenciado Héctor Echeverría

Fecha de Aprobación: 10 de noviembre de 1989.

A mis padres

DISEÑO E IMPLEMENTACION DE UN GENERADOR DE INTERFASES BASADO EN LENGUAJE NATURAL

RESUMEN

Este trabajo de graduación se puede dividir en dos partes importantes. La primera es el diseño de un generador de interfases. Se presentan las consideraciones teóricas de un compilador y de una gramática cómoda y versátil que permita al usuario una óptima utilización del generador. Se plantean los pasos de diseño del proyecto y la manera en que se toman en cuenta las características que optimizan un compilador.

La segunda parte es la presentación del desarrollo de los programas que constituyen el sistema. Se exponen las distintas herramientas posibles que existen para llevar a cabo este trabajo y se describe el criterio de selección de las más adecuadas. Las estructuras de datos utilizadas, son descritas en detalle y al final se propone una evaluación de los objetivos del proyecto, la cual consiste en exponer la manera en que son incluidos en el sistema todos los objetivos que se habían planteado. Además se adjunta un ejemplo del resultado del sistema donde se muestran las bondades del producto final.

Como parte también importante del sistema, dentro de las utilerías con que se cuenta, se diseñó y desarrolló un

manejador de ventanas que puede ser utilizado no solo en este sistema, sino en cualquier otro programa de cualquier índole, desarrollado en lenguaje C. Puede incluso ser utilizado por los usuarios del generador de interfases para lograr sistemas más versátiles y flexibles.

El proyecto utiliza varios tópicos de las ciencias de la computación tales como:

- Inteligencia artificial
- Organización de lenguajes de programación
- Teoría de compiladores
- Teoría de grafos
- Sistemas operativos
- Estructuras de datos
- Bases de datos
- Análisis de sistemas
- Programación en Pascal y lenguaje C

Estas, entre otras, son las áreas cuyos conceptos más se utilizaron a través del proyecto.

El sistema está enfocado a un usuario final que pueda entablar un diálogo con la computadora, con base en preguntas y respuestas, que tengan sentido dentro del ambiente en que se trabaja. La generalización de este tipo de ambiente es uno de los objetivos principales de este proyecto. Se puede diseñar un interfase de cualquier índole para satisfacer distintas necesidades a un costo bajo de tiempo, recurso humano y dinero, obteniendo así, un

considerable beneficio por la facilidad de uso y por el poco tiempo que se invierte en el aprendizaje del mismo.

PREFACIO

En la actualidad se ha hablado mucho de lo que es la inteligencia artificial, sin embargo, a pesar que el trabajo de investigación que se ha desarrollado hasta la fecha en este tema es muy amplio, los resultados prácticos que éstos han dejado son realmente pocos.

Son diversas las causas que han provocado este fenómeno, a pesar de ello la importancia que ha tomado en los últimos 3 años es muy grande. Se ha descubierto que las aplicaciones de esta rama de la computación abarcan tanto el ámbito comercial como el científico.

El proyecto que se describe a continuación tiene ambos enfoques, científico y comercial. El primero es porque se utilizan muchas técnicas convencionales de ciencias de la computación que ayudan al desarrollo integral de este proyecto. El segundo, se deriva de la idea original de este sistema: está basada en la facilidad de acceso al usuario a un sistema de información. Todo originado en una rama de la inteligencia artificial que es el reconocimiento de lenguaje natural.

CONTENIDO

I. INTRODUCCION

El proyecto que se decidió desarrollar es un precompilador de lenguaje C. Se describen a continuación los aspectos más sobresalientes acerca del diseño y el desarrollo de este sistema. Se utilizaron diversos criterios y técnicas aprendidas teóricamente y otras basadas en la experiencia en el campo de la computación en nuestro país.

La organización del informe se presenta de una manera similar al proceso de diseño y desarrollo de este programa. La intención de esto es poder llevar de la mano al lector hacia la comprensión total del sistema, desde sus inicios hasta poder utilizar el producto final.

Se utilizan ciertos conceptos que, por cuestiones de tiempo y espacio, no son incluidos en este condensado. Sin embargo se incluye una bibliografía adecuada para que sean satisfechas todas las dudas que surgan alrededor de algún concepto utilizado.

II. LENGUAJE NATURAL

La importancia de la inteligencia artificial en este proyecto, es porque una de sus áreas de estudio principales, es el reconocimiento de lenguaje natural. A continuación se incluye un poco de historia de lo que han sido los lenguajes a través de la historia de la computación moderna.

A. Historia

A través de la historia de la computación moderna, las maneras de comunicarse con la máquina, han ido evolucionando rápidamente. En un inicio, la programación de una computadora, consistía en conectar directamente alambre con alambre para lograr que ésta, hiciera lo que el usuario deseara. Sin embargo, esta técnica comienza a quedar atrás cuando aparece el lenguaje de máquina. Este, está compuesto de instrucciones que la máquina entiende muy bien pero el usuario tiene que invertir mucho tiempo para poder escribir programas, por la incomodidad de manejar un sistema de comunicación que no tiene relación alguna con lo que se piensa.

Posteriormente, aparece el Assembler que es un lenguaje relativamente más fácil de entender para el usuario, que el anterior. El tiempo de programación se reduce considerablemente, aunque no lo suficiente para disminuir el costo y tiempo de desarrollo significativamente.

Luego de este avance aparente en lo que respecta a lenguajes de programación, aparecen los de tercera

generación. Se basan en series de palabras entendibles para el usuario, y que serán convertidas a lenguaje de máquina por medio de un compilador. En este momento, pareciera que el problema de desarrollo de programas está solucionado, sin embargo, aún es posible delegar más acciones a la máquina, puesto que se logra rapidez y eficiencia.

Los compiladores comienzan a tomar mucha importancia, debido a que se implementan lenguajes específicos para muchas actividades comerciales y científicas. Con ellos, es más sencillo familiarizarse que con un lenguaje de fines generales. Esto se debe, en gran parte, a que para aprender un lenguaje especializado en una rama, el usuario conoce muy bien términos y actividades que puede describir con una herramienta que maneje similares conceptos.

Pero la computación sigue evolucionando. Se persigue la comodidad del usuario y la eficiencia en el desarrollo, y se pone más énfasis en el diseño de los programas, reduciendo así la dificultad en la programación de sistemas. Aparecen los lenguajes de cuarta generación. Estos poseen una sintaxis más mnemónica y cada una de sus instrucciones es muy poderosa, puesto que abarca varias instrucciones de un lenguaje de tercera generación, y bastante más instrucciones de un lenguaje de máquina. Esto disminuye el esfuerzo de aprendizaje de un programador y el documento fuente es más predecible a simple vista. El éxito de esta generación es la similitud de las instrucciones con el lenguaje que se utiliza normalmente.

Con esta idea, nacen los lenguajes de quinta generación. Es el utilizado normalemente por las personas y ahora, entendido por la computadora. Se reduce, y casi se elimina, el aprendizaje de una sintáxis rígida y especializada. Se regresa a los lenguajes de fines generales, pero con mayor poder que cualquier otro antes implementado. Este se conoce como lenguaje natural.

A pesar de todas las ventajas que ofrece esta nueva generación, la implementación de un compilador que abarque todas las posibles combinaciones de las palabras de un idioma se hace casi imposible. Se han desarrollado técnicas en inteligencia artificial que persiguen este fin. Actualmente, el reconocimiento de lenguaje natural es posible aunque las técnicas que se utilizan están aún bajo investigación para lograr más eficiencia y generalidad.

Conociendo los objetivos de esta generación, se pueden implementar sistemas basados en este concepto utilizando ciertas variantes del mismo. Esto produce sistemas muy cómodos y muy específicos de utilizar para el usuario, pero con las ventajas ofrecidas por los lenguajes de quinta generación.

B. ¿ Qué es el interfase de lenguaje natural vía menús sensibles al contexto ?

La finalidad principal de un lenguaje de quinta generación es la fácil comunicación entre el usuario y la máquina. Originado en esta idea, nace el interfase de

lenguaje natural, vía menús sensibles al contexto. Es un programa que por medio de un diálogo con el usuario, va escribiendo instrucciones en lenguaje natural y paralelamente en un lenguaje de generaciones anteriores con el fin de que la máquina ejecute una acción deseada. Este diálogo se origina en las opciones que aparecen en un menú seleccionadas por el usuario en su momento.

Para una mejor comprensión de este sistema, en el apéndice B se muestra un ejemplo de corrida de este programa. Especificamente, está desarrollado para un sistema de control de personal. Maneja datos de empleados. Las pantallas a las que se hace referencia, están tomadas de un prototipo del interfase de lenguaje natural. Se trata de navegar a través de la red para observar la ejecución de los códigos especificados para cada uno de sus elementos. Con este ejemplo se persigue que se entienda mejor el objetivo de este proyecto.

III. DISEÑO

Para poder llevar a cabo el proyecto, fueron necesarias varias fases de análisis y de diseño. Se comenzó con un prototipo que incluye el desarrollo de un interfase basado en lenguaje natural programado completamente en Modula 2. Con base en este programa, se determinaron las partes del mismo, que podían ser generadas a partir de la implementación de un precompilador de un lenguaje de alto nivel.

El siguiente paso, consiste en la determinación de la gramática a utilizar para poder especificar cómodamente las características de las rutinas en lenguaje C que deben ser creadas por este precompilador.

Para lograr esta fase de diseño de la gramática, es necesario, primeramente, considerar ciertos parámetros importantes tales como:

1. Especificación de las características que debe poseer el lenguaje.
2. Determinar los objetivos del mismo.
3. Diseño de los tipos de bloque de especificaciones que contiene el lenguaje.
4. Detalle sintáctico y semántico de cada uno de los bloques.
5. Detalle del código a generar con base en la descripción reconocida.
6. Evaluación de los objetivos planteados respecto del diseño realizado.

En las siguientes líneas se amplía cada uno de los puntos enumerados anteriormente. En cada uno de ellos se incluye los aspectos más relevantes de esa fase.

A. Características

Con base en la experiencia del trabajo con lenguajes de programación, se determinaron ciertas características que se consideran importantes, debido a las desventajas encontradas en éstos. Los puntos deben ser considerados en el momento de diseñar la gramática a utilizar. Estas características son completamente generales y deben formar parte de una gramática eficiente y funcional. Los más importantes son:

1. Estructurado

La gramática que se utilice debe ser estructurada para que el usuario pueda escribir sus especificaciones muy fácilmente y pueda darles mantenimiento de igual forma.

2. Compacto

Se debe evitar, en lo posible, la redacción de muchas palabras para poder utilizar eficientemente la gramática. De esta manera se evita, también, que el aprendizaje del mismo sea demasiado largo, en tiempo. Puede además trabajar en pocos párrafos o bloques, gran cantidad de información.

3. Fácil de entender

La gramática debe utilizar una síntaxis que permita, cuando se lea la especificación fuente, determinar el contenido de la misma sin mayor problema de entendimiento o traducción.

B. Objetivos

Además de poseer ciertas características, la gramática a utilizar en este caso específico, persigue ciertos objetivos que son propios del proyecto. Algunos de estos, son de suma importancia; otros, de menor importancia pero deben ser considerados también. Los de mayor relevancia son:

1. Sensible al contexto

En el lenguaje objeto o en el interfase, que es el resultado del precompilador, debe existir la posibilidad de manejar ciertas banderas o indicadores que limiten conjuntos de expresiones que tengan coherencia semática entre ellas. Esto se debe a que en lenguaje natural es muy importante mantener esta congruencia en todo momento para no perder el sentido de las oraciones o de las palabras.

Estos filtros, funcionan con base en las opciones seleccionadas de antemano por el usuario. Sintácticamente, si se está hablando en singular, las opciones posteriores deben ser completamente congruentes con esta propiedad del lenguaje. Además, semánticamente, se debe conservar el

sentido de la oración. Si se habla de días de la semana, no se puede incluir dentro de las siguientes opciones los nombres de los meses del año. Por ejemplo, es incorrecto decir: " El día junio ... ".

La gramática debe considerar, en su diseño, la posibilidad de hacer el diálogo sintáctica y semánticamente correcto.

2. Facilidad en la descripción de redes

Basado en la estructura que se debe generar para la ejecución del interfase, la gramática considera plenamente la facilidad de descripción de grafos. Debe poder especificar nodos y transiciones unificadas, de tal manera que se pueda describir una red completa.

3. Descripción de "ATN"

Entrando un poco más a lo que es el interfase con el usuario que será generado, aparece el concepto de "Augmented Transition Network" (ATN). Una red es un grafo que posee estados y transiciones. Los "ATN" además, incluyen ciertas condiciones que son atribuidas a los nodos como a las transiciones. Una variante de este tipo de redes son las que se utilizan en el interfase. Por esta razón, la gramática debe facilitar la descripción de estas propiedades para cada uno de los elementos.

4. Descripción de códigos de ejecución

El funcionamiento del interfase se basa en el recorrido de un grafo a través de varias transiciones y de varios nodos. Para cada uno de estos elementos transitados, además de existir condiciones para pasar por ellos, existen códigos de ejecución que se dan en el momento en que se visitan. Una vez más, la gramática debe ofrecer esta facilidad de descripción.

5. Generador de instrucciones de propósito general

El resultado del precompilador es un interfase con el usuario. El del interfase con el usuario es una serie de instrucciones en un lenguaje que lleva a la generación de un reporte o de una consulta, o de cualquier otro comando que sirva para ejecutar una actividad. Esto hace que el interfase sea completamente de propósito general para cualquier lenguaje que se desee utilizar. Esta variabilidad, en que puede resultar el interfase debe ser descrita en el lenguaje a diseñar.

6. Código de ejecución de retroceso

"Errar es de humanos, para ser perfecto se necesita una computadora". Este dicho se aplica en este caso y es muy cierto. Nadie es perfecto. Considerando la posibilidad de indecisión en un momento determinado por parte del usuario, debe existir la posibilidad de rehacer el diálogo.

Para poder retroceder, no se puede preestablecer el recorrido que llevará el usuario en el grafo. Esta generalidad se abarca si el lenguaje posee alguna manera de poder variar los códigos de retroceso para cada uno de los elementos que hayan sido transitados. Nuevamente, es necesario que este detalle esté incluido en la gramática.

7. Especificación de ventanas

El interfase, funciona con base en un diálogo. Este, necesita ofrecer al usuario varias opciones a escoger en cada punto de decisión, en este caso, en cada nodo por el que se transite. Para mostrar al usuario estas posibilidades de selección, se utilizan ventanas en la pantalla de la computadora. Para poder variar las características de cada una de las ventanas que aparecen a criterio del usuario, la gramática debe considerar la descripción de ellas y sus características respectivas como atributos, tamaño, etc.

C. Diseño de bloques de especificación

Profundizando en el diseño de la gramática, primero se determinó en términos generales los bloques que contiene:

1. Descripción del código inicial a ejecutar

El usuario que utiliza este precompilador, necesita tener la posibilidad de inicializar sus variables, ambientes y algunos otros detalles de tipo general. El primer bloque

de la gramática consiste en un bloque opcional de código en C que el usuario, libremente, puede escribir. Este será ejecutado al inicio del interfase.

2. Descripciòn de la red

Para tener en el código fuente una vista general de lo que es la red, primeramente de describe de qué nodo a qué nodo va cada una de las transiciones. Esto es el primer bloque de descripción.

3. Descripción de los elementos de la red

Dentro de si misma, la red debe ser detallada en sus propiedades más específicas. El siguiente bloque consiste en la descripción de las características de cada elemento de la red. Los nodos y las transiciones son descritas en detalle y asignadas cada una de sus propiedades.

4. Descripción de las ventanas

Una vez descritas todas las redes con todos sus elementos, se pasa al bloque de descripción de todas las ventanas a las cuales se hizo referencia en la especificación de la red.

Estos constituyen los bloques globales que representan la gramática. Dentro de cada uno de ellos existe una serie de consideraciones que abarca muchos de los objetivos planteados.

D. Detalle sintáctico y semántico

Dentro de la gramática existen los bloques descritos anteriormente. No existe una metodología universal preestablecida para la descripción semántica y sintáctica de un lenguaje. Sin embargo, a continuación se describe en detalle estas características utilizando convenciones propias que se encuentran descritas en el apéndice A.

1. Código inicial a ejecutar

sintáxis

[Initial Code]

[<código en lenguaje C>]

semántica

- Es código en C que genera una rutina que será ejecutada al inicio del interfase.

2. Descripción de redes

sintáxis

{ ATN }

{ net }

{ [<nodo> - (<transición>) -> <nodo>,] }

{ [(atn:<nombre>)] }

{ [<nodo> [generate].] }

semántica

- Define un "ATN" con sus transiciones y nodos
- Si una transición es un "ATN", se especifica el nombre que debe aparecer definido posteriormente
- Se indica si el nodo es terminal o no. Con este dato se determina en que momento se debe generar el código como resultado del interfase. Se debe distinguir el estado de aceptación del estado de generación. El primero únicamente indica que la red finaliza y el segundo que se genera código al momento de llegar a este nodo.

3. Descripción de elementos

sintáxis

```
{ features                                  }
{ (default trans)                           }
{ (<transición> ):                          }
{ [ cond : <expresión booleana en C>]       }
{ [ trans: <código en C>                  ] }
{ [ back : <código en C>                  ] }
{ [ generate : <código en C >             ] }
{ [ window : <nombre>                     ] }
{ (default state)                           }
{ (<nodo>       ):                          }
{ [ arrive : <código en C>                ] }
{ [ leave  : <código en C>                ] }
```

{ [back : <código en C>] }

{ [generate : <código en C >] }

semántica

- Se define el detalle de los elementos de la red
- Para las transiciones
  - Se definen las transiciones o especificaciones "default"
  - Se define la condición de aceptación de la transición
  - Código de ejecución si se transita por esa transición
  - Código de recuperación si se transita de retroceso
  - Código que genera el interfase para esta transición
  - Se define la ventana en la que debe aparecer en el interfase
- Para los nodos
  - Se define el nombre del nodo o un "Default"
  - Código de ejecución cuando se llega a èl
  - Código de ejecución cuando se abandona
  - Código de retroceso si se anula su tránsito
  - Código de generación de código como salida del interfase

4. Descripción de características de ventanas

sintáxis

Windows

{ (default) }

{ (<nombre>): }

{ offset : <fila>, <columna> }

{ length : <largo> }

{ [ptitle : <posición del título>] }

{ [tframe : <tipo de marco que rodea la ventana>] }

{ [title : <caracter>/<fondo>, <caracter>/<fondo>] }

{ [frame : <caracter>/<fondo>, <caracter>/<fondo>] }

{ [options: <caracter>/<fondo>, <caracter>/<fondo>] }

semántica

- Se definen las ventanas a las cuales se hizo referencia en la descripción de la red
- Se especifica el nombre de la ventana o características "default"
- Se indica relativamente en la pantalla, la posición de la esquina superior izquierda de la ventana
- Se especifica el largo máximo que puede tener la ventana
- El título puede estar al centro o a la izquierda, en la ventana. En este punto se especifíca ese dato.
- El tipo de marco puede ser de doble línea, de

línea sencilla o puede no tener marco. Aquí se especifica este dato.

- En caso que el monitor utilizado sea de colores, se indica el color de los caracteres y del fondo del título, el marco y las opciones que aparecen en una ventana cuando está activa y cuando no lo está. Si el monitor utilizado es monocromo, en este mismo lugar, se describen los atributos especiales del título, el marco y el contenido de la ventana

E. Generación de código

La traducción de la gramática a lenguaje objeto, en este caso C, es el centro de importancia alrededor del cual giran todas las estructuras, características, ventajas y propiedades del precompilador. Para describir la generación de código que se produce a partir de las especificaciones fuente, utilizando la gramática anteriormente expuesta, se expone el esqueleto de las mismas.

Estructurando de una manera eficiente y funcional, el código generado se puede dividir en cuatro partes. Una de ellas, quizá la más interesante, es la definición de características "default". Para facilidad del usuario se definen propiedades en los elementos de la red, incluyendo las ventanas, que abarquen varios renglones de las especificaciones. Estas propiedades pasarán a formar parte

de las características de un elemento, si éste al momento de definirse no cuenta explícitamente con ellas.

Para clarificar y ejemplificar los conceptos antes mencionados, se describe en detalle cada una de estas partes, mostrando además, la estructura del código generado:

1. Código de transiciones

Para las transiciones, se generan cuatro rutinas diferentes. Todas son similares, a excepción de la primera que es el código de condición. Todas las rutinas tienen una instrucción "CASE" y recibe como único parámetro la identificación en la tabla de la transición que se desee.

Estas son incluidas en el programa interfase y son llamadas cuando se necesite ejecutar determinado código. El esqueleto de las mismas se describe a continuación:

```
<nombre de la rutina>( ntrans )
begin
switch( ntrans )
 begin
  case 1: begin
            < expresión booleana especificada en el
              fuente o código en C a ejecutar >
    .
    .
 end
end
```

Los nombres de las rutinas son :

- Exec_t_cond : son las expresiones booleanas en C que determinan si la transición debe aparecer o no en un momento dado.
- Exec_t_tran : es el código en C que se ejecuta al pasar por una transición en el interfase.
- Exec_t_back : son las instrucciones que se ejecutan cuando se anula el paso por una transición.
- Exec_t_gene : es el código que genera las instrucciones en el lenguaje objeto del interfase.

2. Código de nodos

Similarmente a las rutinas generadas para la ejecución de los códigos referentes a las transiciones, se producen, a partir de las especificaciones fuente, otras para el manejo de los códigos de los nodos. Los esqueletos son muy similares y el parámetro es el mismo. Se describe a continuación, en detalle, el esqueleto de las rutinas generadas:

```
< Nombre de la Rutina > ( nstate )
begin
switch( nstate )
 begin
  case 1: begin
```

```
                < código en C especificado en el
                  fuente >
      .
      .
    end
  end
```

Los nombres de las rutinas son :

- Exec_s_arri : código que se ejecuta al llegar al nodo, especificado en la navegación por el interfase.
- Exec_s_leav : código que se ejecuta al abandonar el nodo en tiempo de corrida del interfase.
- Exec_s_back : código de retroceso de un nodo. Se ejecuta cuando se anula el tránsito a través de este nodo.
- Exec_s_gene : instrucciones que generan el código objeto del interfase.

3. Código de definiciones "default"

Para hacer versátil el sistema, se pueden definir en el documento fuente, características que serán incluidas en los elementos de las redes, cuando no sean definidos explícitamente. Por ejemplo, si se desea que varios nodos ejecuten el mismo código cuando se salga de ellos, se define un código de salida como "default" y será asignado a cada nodo que aparezca después de esta definición hasta que el

"default" sea redefinido o termine la red. Se pueden definir todas las características de las transiciones y de los nodos.

Para el manejo de esto, se utilizan estructuras internas en las cuales se escribe, temporalmente, el código "default". Además, existe una tabla índice sobre esta estructura que determina las posiciones exactas donde se encuentra el código actual, si lo hay.

Se muestran a continuación los esqueletos de las rutinas que se generan, temporalmente, en tiempo de compilación:

- Definiciones "default" en transiciones y nodos

```
case default: begin
                < código en C >
                break;
              end
.
.
```

4. Código de inicialización de estructuras

Esta es la rutina generada de mayor importancia. Es la que debe llevar la mayor eficiencia posible y la que determina el funcionamiento del interfase. Es la rutina que inicializa las tablas de datos del mismo.

Para la generación de estas rutinas se tomó en consideración el tipo de estructuras de datos que utiliza el

interfase. Basado en esto, las tablas de elementos en el compilador, sufren varias modificaciones en su forma, antes de ser escritas al archivo objeto. Dentro de éste, existen varias rutinas que son descritas a continuación:

a. Rutina de inicialización de "ATN"

```
initatn()
begin
 inic_atn( A, B );
 .
 .
end
```

- Parámetros:
    - A. Corresponde al número de posición que lleva dentro de la tabla que se maneja en el interfase.
    - B. El número de posición en la tabla de nodos en que inicia esta red.

b. Rutina de inicialización de nodos

```
initnodos()
begin
 inic_nodos( A, B, C, D, E );
 .
 .
end
```

- Parámetros:

A. Corresponde al número de posición que lleva dentro de la tabla que se maneja en el interfase.

B. El número de posición en la tabla de transiciones donde inician las que salen de este nodo. En caso sea cero, significa que la transición es un "ATN".

C. Es el indicador de generación de código en un nodo.

D. Es el número de referencia con el cual se llama a las rutinas de ejecución de código.

E. Indica si el nodo es terminal o no.

c. Rutina de inicialización de transiciones

```
inittrans()
begin
 inic_trans( A, B, C, D, E, F, G );
 .
 .
end
```

- Parámetros:

A. Número correlativo de transición, el cual se refiere a la posición de la tabla que ocupa en el interfase.

B. Número de referencia de "ATN" al que

pertenece la transición.

C. Número de nodo en que finaliza la transición. En caso de ser 0, la transición es una llamada a un "ATN".

D. Número de ventana a la cual pertenece o en la cual debe aparecer la transición.

E. Nombre o título de la transición.

F. Si se trata de un "ATN", este parámetro indica el número de referencia en la tabla de "ATN".

G. Este, indica el número de referencia de la transición con la cual se deben llamar a las rutinas de ejecución de código.

d. Rutina de inicialización de ventanas

```
initwindows()
begin
 inic_windows( A, B, C, D, E, F, G, H, I, J, K, L,
               M, N, O, P, Q, R, S, T );
 .
 .
end
```

- Parámetros:

A. Número de referencia en la tabla utilizada por el interfase.

B, C, D, E. Coordenadas de la esquina superior izquierda y la esquina inferior

derecha respectivamente, con base en las dimensiones de la pantalla.

F. Título o nombre de la ventana.

G. Posición en que debe llevar la ventana escrito el título.

H. Tipo de línea que debe formar el recuadro de la ventana.

I, J, K, L. Atributos de texto y fondo del título de la ventana, cuando está activa y cuando no lo está, respectivamente.

M, N, O, P. Atributos de texto y fondo del marco de la ventana cuando está activa y cuando no lo está, respectivamente.

Q, R, S, T. Atributos de texto y fondo de las opciones dentro de la ventana, cuando está activa y cuando no lo está, respectivamente.

F. Evaluación del diseño

Al terminar el diseño de la gramática, se considera necesario, para una mayor claridad en la comprensión del mismo, una evaluación de sus características y de sus objetivos. A continuación se describen las consideraciones de diseño que se tuvo con cada una de ellas:

BIBLIOTECA
DE LA
UNIVERSIDAD DEL VALLE DE GUATEMALA

1. Objetivos

Se comienza describiendo los objetivos, para que al finalizar su evaluación se incluyan las características que deben mantenerse en todo el diseño de la gramática, independiente de cualquier consideración adicional que se tenga que hacer.

a. Sensible al contexto

Para que se logre delimitar ambientes en el interfase, se introduce un código de condición en cada una de las transiciones. Este evalúa una estructura que contiene las especificaciones de las expresiones que el usuario del interfase ha elegido hasta el momento. Esta estructura, varía conforme se avanza en la navegación a través del interfase. Son los códigos de ejecución de los nodos y de las transiciones los que manejan esta estructura interna.

b. Facilidad de descripción de redes

En un bloque inicial de la gramática, se hace una descripción global de las redes, para luego proceder al detalle de los elementos de la misma. Con este bloque, se logra una vista global, sencilla y fácil de escribir.

c. Descripción de las características de los elementos de la red

Para asignar las características a cada uno de los elementos de la red descrita, se asigna en el siguiente

bloque de especificación. En este se describe, por separado, cada uno de los elementos de la red, incluyendo las transiciones, con lo que la red se convierte en una variante del "ATN".

d. Descripción de códigos de ejecución

Al describir en otro bloque las propiedades de cada elemento, se incluyen los códigos que se deben ejecutar cuando en el interfase se transita por alguno de estos. De esta manera queda considerada la posibilidad de poder manejar rutinas propias del usuario.

e. Generador de instrucciones de propósito general

Para generalizar el uso de los interfases, existe dentro de las propiedades de cada elemento, un renglón que se denomina en la gramática "generate". Este es código en C que se utiliza para indicar como armar las instrucciones del código objeto del interfase con base en las opciones seleccionadas en el mismo. Con este código variable se logra introducir, al interfase, instrucciones para cualquier tipo de lenguaje que el usuario desee.

f. Código de ejecución de retroceso

Similarmente al código que se utiliza para la generación de intrucciones para el interfase de tipo general, existe un código de retroceso que se ejecuta cuando se transita de regreso por alguno de los elementos de la

red. Este código se especifica de manera similar que el de instrucciones.

g. Especificación de ventanas

El último bloque de especificación de la gramática abarca la descripción de las ventanas. Todos los parámetros que se utilizan de atributos, tamaño, título y otros, se especifica en este bloque. Esto permite utilizar las ventanas a voluntad del usuario del precompilador.

2. Características

Se procede ahora a describir las consideraciones realizadas para cumplir con las características programadas. A través de la inclusión de todos los objetivos, estas características se mantienen, ya que abarcan toda la gramática y no únicamente consideraciones aisladas.

a. Estructurado

La gramática está dividida en bloques de especificación, y dentro de cada uno de ellos, existen subbloques que mantienen, a través de todo el código fuente de la descripción, un tipo de documento estructurado. Además, los bloques contienen una entrada y una salida que facilita el trabajo del precompilador en el reconocimiento de los elementos y de sus propiedades.

b. Compacto

La escritura de una descripción de un interfase se limita a los parámetros que son necesarios para que éste pueda funcionar bien. El código que el usuario escribe puede auxiliarse con las definiciones "default" para hacerlo aún más compacto.

c. Fácil de entender

Al ver en pantalla o en papel una descripción de un interfase, se encuentra que es agradable a la vista por su misma estructura de bloques. Si está indentado correctamente, se entiende lo que está escrito. Las palabras reservadas son muy explícitas, no se hicieron abreviaciones en ellas para poder trabajarlo más cómodamente. Es muy consistente en el sentido en que para todos los elementos se sigue el mismo patrón de descripción.

IV. IMPLEMENTACION

La segunda parte de este proyecto, consiste en la implementación eficiente de un precompilador que reconozca y traduzca las especificaciones a un interfase. Para la implementación se tomaron decisiones que llevan a una mejor realización del sistema. Los criterios utilizados se tomaron con base en muchas técnicas y a experiencia en el desarrollo de este tipo de programas.

El funcionamiento eficiente del precompilador, gira alrededor de un código objeto muy rápido y funcional, aparte de considerar mucho el espacio en memoria que este código debe ocupar. En este sentido, el interfase debe ser muy especial, pues es el producto final y el que se va a utilizar. Este está orientado para usuarios cuyo conocimiento de computación es mínimo, por ello, el interfase debe funcionar lo más eficientemente posible.

A. Ambiente de desarrollo

Para la programación del sistema en una computadora personal, existen cantidad inmensa de lenguajes que se pueden utilizar, sin embargo se pueden reducir a unos cuantos cuando se hacen consideraciones tales como:

- Lenguaje estructurado
- Rapidez en cuanto a ejecución de código
- Rapidez en cuanto a tiempo de compilación
- Eficiencia en el manejo de estructuras de datos complejas.

Además de determinar el lenguaje a utilizar, se debe también elegir herramientas de desarrollo que respondan a las necesidades específicas del programa. En este caso, para auxiliar el código objeto, se debe utilizar un manejador de ventanas. La eficiencia en rapidez y memoria de este programa debe ser muy buena. Por el número de rutinas auxiliares que se necesitaban de este manejador se decidió mejor implementar uno con las rutinas necesarias para este programa. Para profundizar más en las consideraciones realizadas en estas dos decisiones se describen a continuación en detalle, cada una de ellas:

1. Lenguaje Utilizado

El lenguaje a utilizar debe tener las características antes mencionadas. Existen varios que las tienen, razón por la cual entraron en consideración otros factores que también son determinantes al desarrollar. Los lenguajes posibles de desarrollo son:

- Pascal
- Lenguaje C
- Módula 2

Debido a la similitud en la estructura de estos lenguajes, se consideró que el conocimiento entre ellos es mayor en C y en Pascal que en Modula 2. La decisión está entonces, entre estos dos lenguajes y para esta parte del proyecto, por el sencillo manejo de estructuras de datos sofisticadas se eligió el Pascal. El C significaría mucho

esfuerzo en cuanto a programación eficiente, debido a la variedad de instrucciones que posee. Para determinar cuál es la mejor, era preciso una mayor investigación del lenguaje y la eficiencia que se logra es similar a la del Pascal.

2. Implementación del manejador de ventanas independiente

Respecto de la creación de rutinas para el manejo de ventanas en el interfase, se eligió el lenguaje C por el proceso de estructuras de datos dinámicas y el manejo, también, de memoria que se hace más sencillo en este lenguaje que en Pascal o en Modula 2. La rapidez del lenguaje en el proceso de este tipo de estructuras es mejor. La memoria en tiempo de corrida es más eficiente en un programa en C.

B. Lenguaje objeto

El lenguaje que produce el precompilador es C. Esto se debe a que únicamente crea rutinas que son variables de un interfase a otro. El interfase en sí, está desarrollado en lenguaje C y posee rutinas fijas, es decir, no variables para cualquiera. Las tablas de elementos son las que varían, al igual que los códigos de ejecución de cada uno de ellos.

La pregunta es ¿ por qué no se creó un interpretador de tablas de elementos ? La respuesta es sencilla: para lograr mejores resultados es bueno hacer un interfase específico para evitar alguna sobrecarga de proceso que sería causada por la demasiada generalidad del programa.

De una manera u otra, las rutinas generadas son, en su mayoría, código que se debe ejecutar eventualmente, no tan seguido como las rutinas de funcionamiento fijo en el interfase. La otra parte de lo que se genera son rutinas de carga de tablas que se ejecutan, una vez en el programa, y es al inicio del mismo.

C. Estructura del precompilador

La estructura básica del programa del precompilador es la siguiente:

- Inicio de la Compilación
- Inicialización de variables
- Apertura de Archivos Objeto
- Inicialización de archivos objeto
- Proceso del codigo inicial de ejecución
- Proceso del "ATN"
    - Inicialización de las estructuras objeto de "default"
    - Reconocimiento de la red
    - Reconocimiento de las características de los nodos
        - Características normales
        - Características "default"
    - Reconocimiento de las características de las transiciones
        - Características normales

- Características "default"
- Proceso de ventanas
- Cierra archivos objeto
- Generación de código objeto
    - Rutina de inicialización de "ATN"
        - Ordenación de Nodos
    - Rutina de inicialización de nodos
        - Ordenación de transiciones
    - Rutina de inicialización de transiciones
    - Rutina de inicilización de ventanas
- Cierra el archivo objeto de inicializaciones
- Final de compilación

Estos son los pasos seguidos a través del precompilador. Se asume en estos pasos el funcionamiento constante del "scanner" , las rutinas de reconocimiento de "tokens" y del "Parser". Estos están implícitos en el funcionamiento y no se mencionan, puesto que no tienen ninguna relevancia especial respecto de los demás pasos.

D. Estructuras de datos

En la parte del precompilador, son utilizadas tablas de datos, donde están contenidos los nombres, y, algunas veces, las propiedades de los elementos en cuestión. Cada una de éstas, posee distintos datos. Todos los arreglos o vectores de registros simulan un árbol binario para efectos de búsqueda.

Otro aspecto importante que abarca todas las estructuras, es que se ingresan en la tabla en el momento en que se reconoce la red. Sin embargo, se les debe definir varias propiedades. Si el elemento únicamente aparece referenciado en la red, pero no se le definen sus características, existen campos dentro de las estructuras que son evaluadas para determinar esta carencia de definición. Las tablas utilizadas son:

1. Tabla de "ATN"

Los campos que contiene el registro son:

- Left : Hijo izquierdo del registro
- Right : Hijo derecho del registro
- Nombre : Nombre del "ATN"
- Atn_ref : Número de referencia en la tabla
- Nodo_inicial : La referencia en la tabla de nodos donde comienza este "ATN"
- Adefinido : Variable lògica que indica si el "ATN" ya está definido

2. Tabla de transiciones

Los campos que componen el registro de transiciones son:

- Left : Hijo izquierdo del registro
- Right : Hijo derecho del registro
- Desde : Nodo en el que parte la transición
- Hasta : Nodo al que llega la transición

- Atn_ref : Número de referencia en la tabla de "ATN" al que pertenece esta transición
- Nombre : Nombre o título de la transición
- Window : Número de ventana en que debe aparecer en el interfase
- Tcond : Variable lógica que indica si el código de condición fue definido para esta transición
- Ttrans : Variable lógica que indica si el código de ejecución de la transición ya fue definido
- Tback : Variable lògica que indica si el código de retroceso ya fue definido
- Tgenerate : Variable lógica que indica si el código de generación de código objeto del interfase ya fue definido
- Tatn : Número de referencia en la tabla de "ATN" si la transición es al mismo tiempo un "ATN"
- Tdefinido : Es una variable lógica que indica si esta transición ya fue definida
- Trans_ref : Debido al ordenamiento que sufre la tabla de transiciones, se guarda el número original de la transición para poder referenciar a sus códigos de ejecución que fueron generados con el número de referencia con el que se introdujeron a la tabla.

3. Tabla de Nodos

La estructura del registro es la siguiente:

- Left : Hijo izquierdo del registro
- Right : Hijo derecho del registro
- Atn_ref : Número de "ATN" al que pertenece el nodo
- Trans_ini : Referencia de la transición inicial que parten de este nodo
- Aceptado : Indicación si el nodo es o no de aceptación, es decir, si es o no terminal
- Genera : Indicación si al transitar este nodo es necesario ejecutar los códigos de generación de instrucciones
- Nombre : Nombre del nodo
- Sarrive : Indicación booleana si el nodo tiene definido o no el código de ejecución de "arrive"
- Sleave : Indicación lógica si este nodo tiene o no definido el código de "leave"
- Sback : Variable lógica que indica si el nodo tiene o no definido el código de retroceso
- Sgenerate : Variable lògica que indica si está definido el código de generación de instrucciones.
- Sdefinido : Indica si el nodo está o no definido.
- Nodo_ref : después del ordenamiento que sufren

los nodos, este campo guarda el número de referencia original.

4. Tabla de ventanas

El detalle de la estructura es el siguiente:

- Left : Hijo izquierdo del registro
- Right : Hijo derecho del registro
- Nombre : Nombre o título de la ventana
- Woffset : Arreglo de 2 posiciones que contiene la fila y la columna superior izquierda de la ventana respecto de las dimensiones de la pantalla
- Wlength : Largo máximo que puede tener la ventana
- Wtitle : Arreglo de cuatro posiciones que contiene respectivamente:
    - Atributo del título (ventana está activa)
    - Atributo del fondo del título (ventana está activa)
    - Atributo del título (ventana no está activa)
    - Atributo del fondo del título (ventana no está activa)
- Wframe : Es un arreglo similar al del título, pero se refiere al marco de la ventana
- Woptions : Es un arreglo igual al anterior, pero referente a las opciones dentro de la ventana

Por último, otra estructura de datos implementada es la que maneja las palabras reservadas. Estas están contenidas en una constante. Las búsquedas que se hacen en ella son secuenciales. Además de estar en esta forma las palabras reservadas, también lo están las definiciones de atributos de las ventanas, las posiciones posibles del título en las ventanas y el tipo de marco. Estas estructuras son utilizadas porque se conoce el tamaño exacto y son estáticas a través del programa. Además, da una gran facilidad al usuario de poder escribir el nombre de los atributos que desea, en lugar de especificarlos con números.

E. Otras consideraciones

Como parte de la implementación, existen otros puntos que son relevantes y que ayudan al mejor funcionamiento del compilador del punto de vista que facilita el uso al usuario. Hay varias consideraciones adicionales como estas:

1. Nombres con blancos

Tanto los nombres de las ventanas como los nombres de las transiciones, son al mismo tiempo, el título de ellas. Ambas deben aparecer en algún momento en la pantalla. Para poder ampliar el campo de operación, estos nombres pueden llevar espacios en blanco y en tiempo de compilación, además estos espacios son reducidos a uno solo entre cada palabra. Para comparaciones y otras operaciones de búsqueda se trabajan con un espacio, sin importar cuántos de hayan

dejado en la especificación fuente.

2. Nombres repetidos

Dentro de cada una de las redes se pueden utilizar, para comodidad del usuario, los mismos nombres de las transiciones y de los nodos. Esto facilita al usuario la programación estructurada y consistente.

3. Estructuras "default"

Para poder manejar las definiciones "default" de una a otra red, se utilizan estructuras internas que se inicializan cada vez que se empieza a recorrer una nueva red. Esto se hace con el fin de poder limitar el uso de los "defaults" hasta que termine una red o la especificación completa.

4. Búsquedas sobre árboles binarios

Las búsquedas sobre las tablas de datos se hacen como en un árbol binario, puesto que los vectores simulan este tipo de estructuras.

V. CONCLUSIONES

1.- El óptimo funcionamiento de un compilador es bastante dependiente de la gramática que se está utilizando.

2.- El funcionamiento del producto final, en este caso el interfase, depende de la facilidad de uso que tenga el diálogo entre el usuario y la computadora.

3.- El tiempo invertido en el diseño del proyecto es el más importante. Es en esta fase donde se determinan todas las bondades que el sistema debe incluir y la manera en que se debe desarrollar.

4.- La selección de herramientos es muy importante para que se puedan incluir dentro del sistema todas las características que se plantearon en el diseño.

5.- Como interfase con el usuario, el lenguaje natural es el método más sencillo ya que no debe aprender nada. Es algo que hace todos los días y está acostumbrado a ello. El impacto psicológico sobre la persona es menor debido a la familiaridad con el lenguaje que ya posee.

VI. BIBLIOGRAFIA

Aho, A. and Ullman, J. Principles of Compiler Design.
1979 Addison-Wesley Publishing Company. USA.

Tenembaun, A. Data Structures and Problem Solving in Pascal.
1976 Prentice Hall. USA.

LOGITECH, Inc. MODULA 2/86. Third Edition.
1986

Patrick Henry Winston. Artificial Inteligence. Second
1984 Edition. Addison-Wesley Publishing Company.

Winograd, Terry. Language as a cognitive Process.
1981 Addison-Wesley Publishing Company.

Aho, A., John Hopcroft, Jeffrey Ullman. The Desing and
1974 Analysis of Computer Algorithms. Addison-Wesley
Publishing Company.

Borland International. TURBO PASCAL, Owner's Handbook.
1987 Borland International.

Borland International. TURBO C, User's Guide. Borland
1987 International.

Borland International. TURBO C, Reference Guide. Borland
1987 International.

APENDICE A

SIMBOLOS UTILIZADOS EN NOTACION SINTACTICA

- Palabras sin ningún símbolo a su alrededor

  Palabras como ATN, net, Initial code, son palabras que deben aparecer en la especificación fuente de la misma manera que aparecen descritas en la gramática. No pueden sufrir ninguna variante.

- Palabras entre los signos : <>

  Es el significado de lo que se encuentra entre ellos. Al escribir la especificación fuente, se debe incluir lo que allí se describe.

- Expresiones entre llaves : {}

  Estos símbolos indican que lo que está entre ellos es repetitivo. Puede aparecer varias veces dentro de las especificaciones fuente.

- Palabras entre paréntesis: ()

Estos símbolos aparecen delimitando palabras que están una sobre otra. Esto significa que es opcional cuál de las posibilidades se elige. Sin embargo debe existir alguna de ellas.

- Expresiones entre corchetes : []

Lo que aparece entre estos símbolos, es opcional. Puede o no ser incluido en las especificaciones.

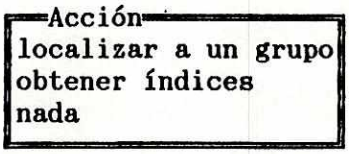
APENDICE B

EJEMPLO DE INTERFASE DE LENGUAJE NATURAL VIA MENUS SENSIBLES AL CONTEXTO

En las siguientes páginas se encuentra una muestra del interfase de lenguaje natural vía menús sensibles al contexto. Para cada una de las pantallas se hacen las observaciones que se consideran necesarias para su mejor entendimiento.

PANTALLA 1

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1



¿ Qué desea hacer ?

-

- Pantalla 1: Muestra las primeras transiciones y la pregunta que aparace en la parte inferior es desplegada gracias al código inicial del interfase. Se elige la opción de localizar un grupo.

PANTALLA 2

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1

Selección
selección de empleado
todos los empleados

¿ Qué desea hacer ?

- Localizar a un grupo de empleados.

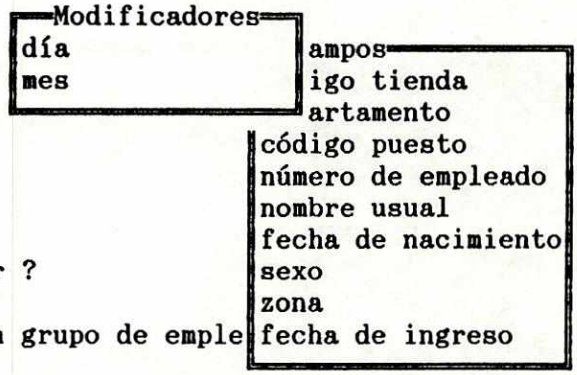
¿ Cuál es el grupo a trabajar ?

-

- Pantalla 2 : La pregunta inicial esta contestada. El código que genera la pregunta es el de ejecución cuando se pasa por esa transición. También el código de llegada del nodo escribe la siguiente pregunta. Las rutinas que se utilizan para escribir estas frases son parte de la librería que posee el generador. Se elige la opción de selección de empleados.

PANTALLA 3

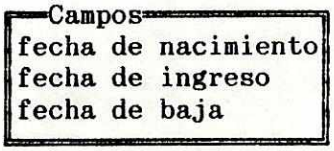
Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1



- Pantalla 3 : Se observa que se inicia la composición de una nueva frase. A la vez se muestran otras transiciones para poder dar características al grupo que se necesita trabajar. En este caso las transiciones disponibles son datos respecto a los empleados como el nombre de la persona, el número del empleado, etc. Las otras, se refieren a algún mes o día en específico que tiene relación con el empleado. Se elige el modificador mes.

PANTALLA 4

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1



¿ Qué desea hacer ?

- Localizar a un grupo de empleados.

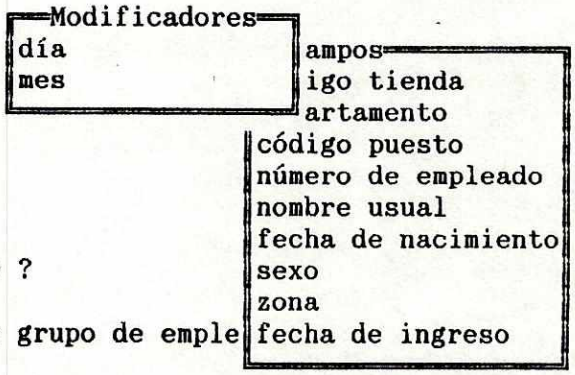
¿ Cuál es el grupo a trabajar ?

- El de los empleados con el mes de

- Pantalla 4 : Se observan varias acciones :
  - Se añaden palabras a la frase que se está formando.
  - Se observa la sensibilidad al contexto. Al referirse a un mes, las transiciones disponibles son únicamente fechas, donde el mes es procedente.

PANTALLA 5

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1



¿ Qué desea hacer ?

- Localizar a un grupo de emple

¿ Cuál es el grupo a trabajar ?

- El de los empleados con

- Pantalla 5 : Es igual a la 3. Es porque en la 4 se presionó la tecla de retroceso que indica al interfase que anule la última transición. Lo interesante, es observar que el código de retroceso de la transición elimina la parte de la frase que se agregó anteriormente. Se selecciona nuevamente el modificador mes para continuar la navegación.

PANTALLA 6

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1

Campos
fecha de nacimiento
fecha de ingreso
fecha de baja

¿ Qué desea hacer ?

- Localizar a un grupo de empleados.

¿ Cuál es el grupo a trabajar ?

- El de los empleados con el mes de

- Pantalla 6 : Se selecciona la fecha de ingreso para trabajar el grupo de empleados.

PANTALLA 7

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1

Comparadores
igual
diferente
mayor
mayor o igual
menor
menor o igual

¿ Qué desea hacer ?

- Localizar a un grupo de empleados.

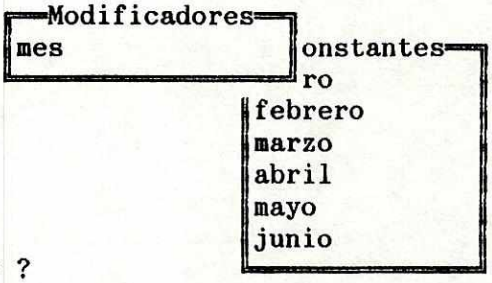
¿ Cuál es el grupo a trabajar ?

- El de los empleados con el mes de la fecha de ingreso

- Pantalla 7 : Nuevamente se observa la adición que se hizo a la frase en formación. Aquí mismo, se solicita al usuario que seleccione la política de búsqueda por medio de comparadores procedentes para el mes de la fecha de ingreso. Se elige la opción de igual.

PANTALLA 8

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1



¿ Qué desea hacer ?

- Localizar a un grupo de empleados.

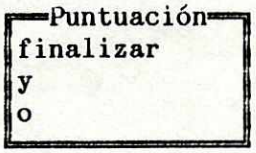
¿ Cuál es el grupo a trabajar ?

- El de los empleados con el mes de la fecha de ingreso igual a

- Pantalla 8 : El ambiente está reducido a trabajar con meses. Por lo tanto dentro de los modificadores aparece únicamente la opción de mes y en las constantes aparecen también nombres de los meses. Son las posibles transiciones para continuar la navegación. Se escoge la constante del mes de octubre.

PANTALLA 9

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1



¿ Qué desea hacer ?

- Localizar a un grupo de empleados.

¿ Cuál es el grupo a trabajar ?

- El de los empleados con el mes de la fecha de ingreso igual a octubre

- Pantalla 9 : Las opciones ahora son seguir con otra condición para seleccionar el grupo o finalizar. Se escoge la opción de finalizar.

PANTALLA 10

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1

¿ Qué desea hacer ?

- Localizar a un grupo de emple

¿ Cuál es el grupo a trabajar ?

- El de los empleados con el me

Campos
código tienda
departamento
código puesto
número de empleado
nombre usual
fecha de nacimiento
sexo
zona
fecha de ingreso

eso igual a octubre.

¿ Qué datos de los empleados desea ver ?

-

- Pantalla 10 : Se observa el punto en la frase. Ahora se deben elegir los datos de los empleados que se desean trabajar, del grupo que se escogió. Se elige inicialmente el nombre usual del empleado.

PANTALLA 11

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1

Anidación
fin
,
y

¿ Qué desea hacer ?

- Localizar a un grupo de empleados.

¿ Cuál es el grupo a trabajar ?

- El de los empleados con el mes de la fecha de ingreso igual a octubre.

¿ Qué datos de los empleados desea ver ?

- el nombre usual

- Pantalla 11 : Ahora, se elige si se desean más campos o no. Se toma la opción de seguir especificando. Se agregan el número del empleado y el departamento al que pertenece.

PANTALLA 12

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1

¿ Qué desea hacer ?

- Localizar a un grupo de empleados.

Salida
impresora
pantalla

¿ Cuál es el grupo a trabajar ?

- El de los empleados con el mes de la fecha de ingreso igual a octubre

¿ Qué datos de los empleados desea ver ?

- el nombre usual, el departamento y código de puesto .

¿ En dónde desea ver la información ?

-

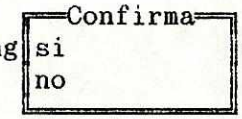
- Pantalla 12 : El siguiente paso en la navegación es elegir el dispositivo de salida. Se elige la pantalla.

PANTALLA 13

Tellme: INFORMACION DE RECURSOS HUMANOS Fase 2.1

¿ Cuál es el grupo a trabajar ?

- El de los empleados con el mes de la fecha de ing



ctubre

¿ Qué datos de los empleados desea ver ?

- el nombre usual .

¿ En dónde desea ver la información ?

- En la pantalla.

Voy a comenzar ... ¿ todo correcto ?

-

- Pantalla 13 : Para finalizar, se pide la confirmación de la acción, y finaliza. En este momento se generan las intrucciones en el lenguaje objeto del interfase y se devuelven como parámetros al sistema donde fue fue llamado.