Lenguaje TIMBA

Resolución de Problemas y Algoritmos
Fundamentos de la Informática
Introducción a la Computación
Introducción a la Programación

Ingenierías en Computación, en Informática y Electrónica
Tecnicaturas Universitarias en Web, en Redes de Computadoras,
en Geoinformática, en Telecomunicaciones y en Electrónica
Profesorado en Tecnología Electrónica

Universidad Nacional de San Luis
Departamento de Informática
Año 2016

Índice

1. Génesis

El lenguaje "TIMBA" (`Terribly Imbecile Machine for Boring Algorithms`) fue desarrollado por un equipo de trabajo en la Universidad Nacional de San Luis, como respuesta a la necesidad de contar con un lenguaje sencillo de programación que permita trabajar con la estructura de pilas, estructurando los programas. El equipo fue dirigido por el Ing. Hugo Ryckeboer, de la Universidad Nacional del Centro de la Provincia de Buenos Aires, el que en la actualidad es aún Profesor Visitante de nuestra Casa.

2. Programación Estructurada

La programación estructurada surgió como un enfoque sintetizador de la labor de programación, sugiriendo el empleo de sólo tres estructuras de control: la *concatenación* o *secuenciación*, la *selección* y la *iteración*, como únicas componentes elementales de un programa.

En el lenguaje "TIMBA" se ha respetado la sugerencia, y por lo tanto el lenguaje cuenta con las tres estructuras de control:

- **Secuenciación:** la secuencialidad natural de las sentencias operativas.
- **Selección:** la sentencia `SI ... SINO ... NADA MAS`
- **Iteración:** la sentencia `MIENTRAS ... REPITA`

Estas tres estructuras de control permiten escribir programas en "TIMBA" de forma estructurada.

3. El Lenguaje "TIMBA"

"TIMBA" es un lenguaje de programación que maneja pilas de cartas, concebido como primer elemento de un curso elemental de programación para futuros profesionales de la disciplina. Consta sólo de tres operaciones elementales sobre pilas, y no maneja variables ni otras estructuras de datos. Por supuesto, así no es posible una gran capacidad de programación, pero esto es desde ya deseable en un lenguaje de uso didáctico.

"TIMBA" permite la definición de pilas, las operaciones elementales de tomar y agregar cartas a las pilas, e invertir la carta que en ese momento se analiza. Desde el punto de vista del programador, un programa en "TIMBA" es una secuencia de órdenes a un ejecutor, UCP, quién es responsable de los resultados y el análisis de errores. UCP reconoce las pilas por su nombre, las operaciones "tomar" y "depositar", las estructuras de control y un ente especial, llamado "CARTA" o "LA CARTA", que es operando implícito de las operaciones de `TOMAR` de una pila y `DEPOSITAR` en una pila, además de ser el objeto único de la operación `INVIERTA`.

4. Programa

Un programa “TIMBA” consta de dos partes, una primera definición del proceso algorítmico y la segunda de datos de pila. Las mismas deben escribirse contiguas, separadas por un punto y coma (`;`).

La definición del proceso debe encabezarse con las palabras claves: `DEFINICION DE PROGRAMA`, y todas las sentencias que constituyen la descripción del algoritmo a continuación, separadas entre sí por coma (`,`). Es decir, el elemento que evidencia el uso de la estructura de control de la *secuenciación* en un programa “TIMBA” es la coma.

Las sentencias del lenguaje “TIMBA” son de tres tipos:

1. Operativas;
2. De selección;
3. Iterativas.

1. Las sentencias operativas actúan sobre las pilas o la carta. Comienzan siempre por un verbo en imperativo, que es interpretado por UCP como una orden. Los verbos posibles son:
   a) `TOME`;
   b) `DEPOSITE`;
   c) `INVIERTA`.

2. Las sentencias de selección permiten ordenar a UCP cursos alternativos del flujo de proceso, bajo ciertas condiciones verificables en tiempo de ejecución. Una sentencia de selección comienza siempre por el adverbio `SI`, seguido de la condición (simple o no) y las sentencias a ejecutarse en caso de ser ésta verdadera.

   Si no hubiera dos secuencias alternativas, sino sólo un bloque de sentencias a ejecutar en caso de cumplirse la condición, o saltear si no se cumpliere, entonces la sentencia `SI` finaliza con las palabras claves `SINO NADA MAS`. Si hubiera una secuencia a realizar, en caso de no cumplirse la condición y sólo entonces, la secuencia deberá incluirse entre las palabras claves `SINO` y `NADA MAS`.

   Claramente las sentencias de selección del lenguaje "TIMBA", en cualquiera de sus variantes, permiten introducir la estructura de control de *selección* en la estructura del programa.

3. Las sentencias iterativas permiten ordenar a UCP la realización de una, ninguna o varias veces un bloque de sentencias, bajo control de una condición (simple o compuesta) verificable en tiempo de ejecución. Una sentencia iterativa comienza siempre por el adverbio `MIENTRAS`, que es seguido por la condición, el bloque a ejecutar y la palabra clave `REPITA`.

   Así, la sentencia de iteración del lenguaje "TIMBA" permite introducir la estructura de control de *iteración* en la estructura de un programa.

La descripción de datos de pila comienza por la frase de palabras clave "`UCP EJECUTE CON LAS SIGUIENTES CARTAS:`", seguida por la descripción de las pilas, cada una separada de la anterior por coma (`,`).

Una descripción de pila es de una de las dos formas:

1. descripción de pila vacía;
2. descripción de contenido de pila.

La descripción de pila vacía tiene siempre la siguiente forma: la palabra clave `PILA` seguida del nombre de la pila, seguido de las palabras claves `NO TIENE CARTAS`.

La descripción de contenido de pilas es semejante, cambiando `NO TIENE CARTAS` por `TIENE` y una lista de cartas. Al finalizar la segunda parte de un programa se coloca punto (`.`).

Cada una de las sentencias posibles y algunas de sus componentes están explicadas con detalle en las páginas que siguen. Para una descripción formal de la gramática que genera "TIMBA", se refiere al lector al Apéndice C de este manual.

Una lista completa de las palabras claves se encuentra en el Apéndice A.

Antes de leer las páginas que siguen, se recomienda referirse al Apéndice B, "Simbología Utilizada", para aprovechar mejor el material.

En términos de la simbología descripta en el Apéndice B un programa en "TIMBA" es:

`DEFINICION DE PROGRAMA` ⟨sentencia⟩ [`,` ⟨sentencia⟩]* `; UCP EJECUTE CON LAS SIGUIENTES CARTAS:` ⟨descripción de pila⟩ [`,` ⟨descripción de pila⟩]* `.`

Los ítems "sentencia" y "descripción de pila" son desarrollados a continuación.

4.1. Sentencias

Las sentencias son el elemento constitutivo de “TIMBA”.

Una sentencia en “TIMBA” es la mínima unidad lógica.

Las sentencias son de dos grandes tipos, en “TIMBA”: *Operativas* y *de Control*.

Las sentencias operativas realizan cambios sobre la configuración de las pilas, o estado de la carta que UCP analiza en ese momento, lo que se simboliza por la “carta que UCP tiene en la mano” (ver Sección 4.1.1 Sentencias Operativas).

Las sentencias de control son comandos condicionales, y en sí no alteran la configuración de las pilas ni el estado de la carta, sino por las sentencias que contienen a su vez. Las sentencias de control permiten controlar la secuencia de operaciones, sin ser operaciones en sí.

Las hay de dos tipos: de selección (ver `SI`) y de iteración (ver `MIENTRAS`).

Cuando en una descripción formal aparece la palabra sentencias, se quiere simbolizar:

$$\langle \text{sentencia} \rangle \, [, \langle \text{sentencia} \rangle]^*$$

Donde sentencia es, formalmente:

$$\left\{ \begin{array}{l} \langle \text{sentencia operativa} \rangle \\ \langle \text{sentencia de selección} \rangle \\ \langle \text{sentencia de iteración} \rangle \end{array} \right\}$$

4.1.1. Sentencias Operativas

Las sentencias operativas son tres, formalmente:

`TOME` [`UNA` [`CARTA`]] `DE` [`LA`] `PILA` ⟨nombre de pila⟩

o

$$\left\{ \begin{array}{l} \texttt{DEPOSITE LA CARTA} \\ \texttt{DEPOSITELA} \end{array} \right\} \texttt{EN} \, [\texttt{LA}] \, \texttt{PILA} \, \langle \text{nombre de pila} \rangle$$

o

$$\left\{ \begin{array}{l} \texttt{INVIERTA LA CARTA} \\ \texttt{INVIERTALA} \end{array} \right\}$$

o escrito en la misma notación que se viene utilizando:

$$\left\{\begin{array}{l} \texttt{TOME}\ [\texttt{UNA}\ [\texttt{CARTA}]]\ \texttt{DE}\ [\texttt{LA}]\ \texttt{PILA}\ \langle\text{nombre de pila}\rangle \\ \left\{\begin{array}{l}\texttt{DEPOSITE LA CARTA}\\ \texttt{DEPOSITELA}\end{array}\right\}\texttt{EN}\ [\texttt{LA}]\ \texttt{PILA}\ \langle\text{nombre de pila}\rangle \\ \left\{\begin{array}{l}\texttt{INVIERTA LA CARTA}\\ \texttt{INVIERTALA}\end{array}\right\} \end{array}\right\}$$

La ejecución por UCP de una sentencia operativa `TOME` presupone que la pila cuyo nombre figura en la sentencia ha sido definida en la segunda parte del programa.

La carta que figura al tope de la pila es tomada por UCP, y cualquier referencia posterior que se haga al ente `CARTA` se interpretará como una referencia a la última carta tomada por UCP (siempre que no hubiera sido depositada).

`DEPOSITE` ordena a UCP dejar la carta que tiene en la mano en ese momento, en una pila determinada.

La pila debe ser declarada en la segunda parte del programa. La ejecución de un `DEPOSITE` presupone que UCP tiene, efectivamente, una carta en la mano; lo que se consigue por medio de la ejecución de una sentencia `TOME`.

Por obvias razones, UCP detectará un error de ejecución si no tuviera una carta en la mano y se le ordenara `DEPOSITE`. Por lo mismo, UCP no admite la ejecución de dos `TOME` sin un `DEPOSITE` entre ellos, detectando también error de ejecución.

Por último, UCP reconoce error de ejecución al intentar tomar una carta de una pila vacía.

El verbo `INVIERTA` se refiere a la carta que UCP tiene en la mano. Éste reconoce dos estados: `NO BOCA ABAJO` y `BOCA ABAJO`. La ejecución por UCP de una sentencia `INVIERTA` altera el estado de `CARTA`: si éste era `NO BOCA ABAJO` pasa a ser `BOCA ABAJO` y viceversa. El verbo `INVIERTA` presupone la existencia de carta, sino UCP detecta un error de ejecución.

Mensajes de error:

Ejemplo 1: `ME PIDE UD. QUE TOME DE PILA` ⟨nombre⟩ `QUE ESTA VACIA.`

Ejemplo 2: `ME PIDE UD. QUE TOME PILA` ⟨nombre⟩ `Y YO YA TENGO UNA CARTA`: el ⟨número⟩ de ⟨palo⟩.

Ejemplo 3: `ME PIDE UD. QUE DEPOSITE EN PILA` ⟨nombre⟩ `Y YO NO TENGO CARTA.`

Ejemplo 4: `ME PIDE UD. QUE INVIERTA LA CARTA, PERO YO NO TENGO CARTA.`

4.1.2. Sentencia de Selección `SI`

La sentencia `SI` permite al programador solicitar a UCP que evalúe una condición y proceder, de acuerdo con ella, a la ejecución de un bloque u otro, por lo que la secuencia normal de ejecución puede depender de las condiciones halladas en las pilas; esto es, de la situación o el estado de las pilas, o bien del estado o atributos de la carta.

Formalmente una sentencia `SI` debe escribirse:

$$\texttt{SI}\ \langle\text{condición}\rangle\ \langle\text{sentencias}\rangle \left\{ \begin{array}{l} \texttt{SINO}\ \langle\text{sentencias}\rangle\ \texttt{NADA MAS} \\ \texttt{SINO NADA MAS} \end{array} \right\}$$

Si la condición (ver Sección 4.2 Condición) es verdadera en el momento de analizarla UCP, se procederá con la ejecución del primer bloque de sentencias; esto es, las sentencias que preceden al `SINO`.

Si la condición fuese falsa, UCP procederá con la ejecución de las sentencias que hubiere entre el `SINO` y el `NADA MAS`, y si no las hubiera, con la sentencia que sigue a la sentencia `SI`; es decir, la que empieza después de la coma (`,`) que sigue a `NADA MAS`. [1]

Las sentencias que pueden escribirse en un `SI` son todas las de "TIMBA", por lo tanto, un `SI` puede contener otro `SI` (y aún un `MIENTRAS`). Tal estructura de un `SI` dentro de otro se llama usualmente "Nidos de SI". (Ver Sección 4.1 Sentencias).

Ejemplo: `SI LA PILA A ESTA VACIA TOME UNA DE PILA B SINO TOME DE PILA A NADA MAS, DEPOSITELA EN PILA C.`

UCP verificará si hay cartas en la pila `A`, y procederá, de acuerdo con ello, a tomar una carta de `B`, si `A` fuese vacía o, una carta de `A` si ésta no lo fuera.

4.1.3. Sentencia de Iteración `MIENTRAS`

La sentencia `MIENTRAS` permite al programador ordenar a UCP la realización repetida y condicionada de un bloque de sentencias, gobernado por una condición verificable en la ejecución. (Ver Sección 4.2 Condición)

Formalmente, la sentencia `MIENTRAS` debe escribirse:

$$\texttt{MIENTRAS}\ \langle\text{condición}\rangle\ \langle\text{sentencias}\rangle\ \texttt{REPITA}$$

Cuando UCP encuentra durante la ejecución de un programa una sentencia `MIENTRAS`, evalúa primero la condición (ver Sección 4.2 Condición), y si ésta es verdadera, ejecuta normalmente las sentencias que siguen hasta el `REPITA`. Al hallar el `REPITA`, vuelve a evaluar la condición, volviendo a ejecutar todas las sentencias si la condición es aún verdadera, y así siguiendo hasta que la condición resulte falsa.

Cuando la condición resulta falsa, ya sea en la primera, segunda o enésima iteración, UCP se saltea la ejecución de las sentencias que siguen hasta el `REPITA`, prosiguiendo con la sentencia que sigue al `MIENTRAS`, esto es, la que comienza después de la coma (`,`) que sigue al `REPITA`. [2]

Ejemplo: `MIENTRAS LA PILA B NO ESTA VACIA TOME DE PILA B, DEPOSITE EN PILA C REPITA, TOME DE PILA D.`

[1] Se coloca coma (`,`) después un `NADA MAS` siempre que continúe después del mismo otra sentencia (concatenación o secuenciación).
[2] Se coloca coma (`,`) luego del `REPITA` siempre que a continuación continúe otra sentencia (concatenación o secuenciación).

UCP vaciará la pila `B`, pasando a la pila `C` las cartas de `B` de a una. Una vez vacía la pila `B`, UCP tomará la carta del tope de la pila `D`.

4.2. Condición

Una condición en "TIMBA" es una proposición lógica, simple o compuesta, que emite un juicio sobre el estado de las pilas o la carta, y que puede por lo tanto ser evaluado como "Verdadero" o "Falso".

Formalmente, una condición en "TIMBA" es:

$$\langle \text{proposición} \rangle \left[\left\{ \begin{array}{c} \texttt{Y} \\ \\ \texttt{O} \end{array} \right\} \langle \text{proposición} \rangle \right]^{*}$$

Donde proposición es:

$$\left\{ \begin{array}{l} \texttt{PILA}\ \langle \text{nombre} \rangle \left\{ \begin{array}{l} \texttt{ESTA} \\ \texttt{NO ESTA} \end{array} \right\} \texttt{VACIA} \\ \\ [\texttt{LA}] \left\{ \begin{array}{l} \texttt{CARTA} \left\{ \begin{array}{l} \left\{ \begin{array}{l} \texttt{ESTA} \\ \texttt{NO ESTA} \end{array} \right\} \texttt{BOCA ABAJO} \\ \\ \left\{ \begin{array}{l} \texttt{ES} \\ \texttt{NO ES} \end{array} \right\} [\texttt{DEL PALO}]\ \langle \text{palo} \rangle \\ \\ \texttt{ES DE} \left\{ \begin{array}{l} \texttt{IGUAL} \\ \texttt{DISTINTO} \end{array} \right\} \texttt{PALO QUE TOPE DE PILA}\ \langle \text{nombre} \rangle \\ \\ \left\{ \begin{array}{l} \texttt{ES} \\ \texttt{NO ES} \end{array} \right\} [\texttt{DE VALOR}] \left\{ \begin{array}{l} \texttt{DISTINTO DE} \\ \\ \left\{ \begin{array}{l} \texttt{MENOR} \\ \texttt{MAYOR} \end{array} \right\} \texttt{QUE} \\ \\ \left\{ \begin{array}{l} \texttt{IGUAL} \\ \left\{ \begin{array}{l} \texttt{MENOR} \\ \texttt{MAYOR} \end{array} \right\} \texttt{O IGUAL} \end{array} \right\} \texttt{A} \end{array} \right\} \langle \text{número} \rangle \\ \\ \left\{ \begin{array}{l} \texttt{ES} \\ \texttt{NO ES} \end{array} \right\} \texttt{DE} \left\{ \begin{array}{l} \texttt{IGUAL} \\ \texttt{DISTINTO} \\ \texttt{MENOR} \\ \texttt{MAYOR} \\ \\ \left\{ \begin{array}{l} \texttt{MENOR} \\ \texttt{MAYOR} \end{array} \right\} \texttt{O IGUAL} \end{array} \right\} \texttt{VALOR QUE TOPE DE PILA}\ \langle \text{nombre} \rangle \end{array} \right\} \end{array} \right\} \end{array} \right\}$$

4.2.1. Pila Vacía

Formalmente, una condición de `PILA VACIA` se escribe:

$$[\texttt{LA}]\ \texttt{PILA}\ \langle\text{nombre}\rangle \left\{ \begin{array}{l} \texttt{ESTA} \\ \texttt{NO ESTA} \end{array} \right\} \texttt{VACIA}$$

UCP reconoce si una pila está vacía cuando no tiene cartas en la pila. Si hubiera una, dos o más cartas en la pila `PILA` ⟨nombre⟩ `ESTA VACIA` sería reconocido como "Falso" por UCP, mientras que `PILA` ⟨nombre⟩ `NO ESTA VACIA` sería "Verdadero".

UCP reconoce un error de ejecución si se intenta tomar cartas de una pila vacía.

Por lo tanto, la condición de pila vacía permite construir sentencias que eviten tomar de pilas sin cartas.

4.2.2. Carta Boca Abajo

Formalmente, una condición de carta boca abajo se escribe:

$$[\texttt{LA}]\ \texttt{CARTA} \left\{ \begin{array}{l} \texttt{ESTA} \\ \texttt{NO ESTA} \end{array} \right\} \texttt{BOCA ABAJO}$$

UCP reconoce dos estados de la `CARTA`: `BOCA ABAJO` y `NO BOCA ABAJO`.

Si la `CARTA` estuviera `BOCA ABAJO`, UCP no puede hacer comparaciones, por lo que esta condición permite construir sentencias que la inviertan condicionalmente.

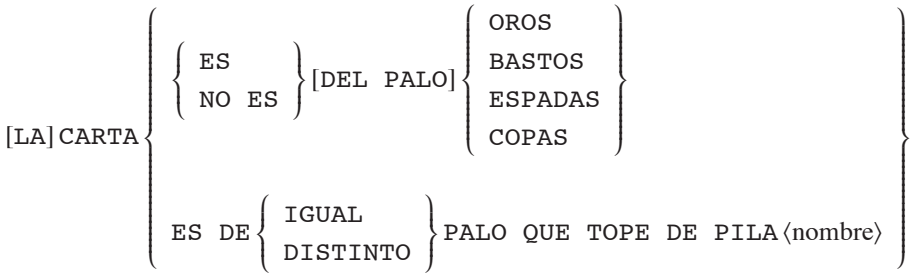
Si no hubiera `CARTA` definida para UCP, esto es, si ninguna `CARTA` hubiere sido tomada (ver Sección 4.1.1 Sentencias Operativas), UCP reconoce un error de ejecución.

Mensaje de error:

Ejemplo 5: `UD. QUIERE SABER COMO ESTA LA CARTA, Y YO NO TENGO CARTA.`

4.3. Palo de Carta

Una carta en "TIMBA" es un elemento biestable (ver Sección 4.2.2 Carta Boca Abajo) con dos atributos: *Palo* y *Valor*. Palo de una carta es uno de los cuatro palos de la baraja española: `OROS`, `BASTOS`, `ESPADAS` o `COPAS`. Valor es un número del `1` al `7` o del `10` al `12`.



En la primera de las dos opciones, UCP compara el palo de `LA CARTA` contra uno de los cuatro palos de la baraja española: `OROS`, `BASTOS`, `ESPADAS`, `COPAS`; detectando valor "Verdadero" si la condición se escribió con `ES` y los palos coinciden o si la condición se escribió con `NO ES` y los palos no coinciden; y "Falso" en el caso inverso: se escribió con `ES` y los palos son distintos o se escribió con `NO ES` y los palos coinciden. UCP detecta un error de ejecución si no tiene `CARTA` (ver Sección 4.1.1 Sentencias Operativas) o si ésta estuviere `BOCA ABAJO` (ver Sección 4.2.2 Carta Boca Abajo).

En la segunda opción, UCP compara el palo de la `CARTA` contra el palo de la carta que está al tope de la pila "nombre", detectando un valor "Verdadero" si la condición se escribió con `IGUAL` y los palos coinciden o si se escribió con `DISTINTO` y los palos no coinciden, siendo para UCP "Falsa" la condición si se escribió ésta con `IGUAL` y los palos son distintos o si se escribió con `DISTINTO` y el palo es el mismo.

UCP detecta un error de ejecución si no tiene carta en la mano (ver Sección 4.1.1 Sentencias Operativas) o si cualquiera de las dos está `BOCA ABAJO` (ver Sección 4.2.2 Carta Boca Abajo), o la pila `ESTA VACIA` (ver Sección 4.2.1 Pila Vacía).

Mensajes de Error:

Ejemplo 6: `USTED QUIERE SABER SI LA CARTA ES DE` ⟨palo⟩ `Y YO NO TENGO CARTA.`

Ejemplo 7: `USTED QUIERE SABER SI LA CARTA ES DEL PALO DEL TOPE` ⟨nombre⟩ `Y YO NO TENGO CARTA.`

Ejemplo 8: `USTED QUIERE SABER SI LA CARTA ES DEL PALO DEL TOPE DE` ⟨nombre⟩ `PERO LA CARTA ESTA BOCA ABAJO.`

Ejemplo 9: `USTED QUIERE SABER SI LA CARTA ES DEL PALO DEL TOPE DE` ⟨nombre⟩ `PERO EL TOPE ESTA BOCA ABAJO.`

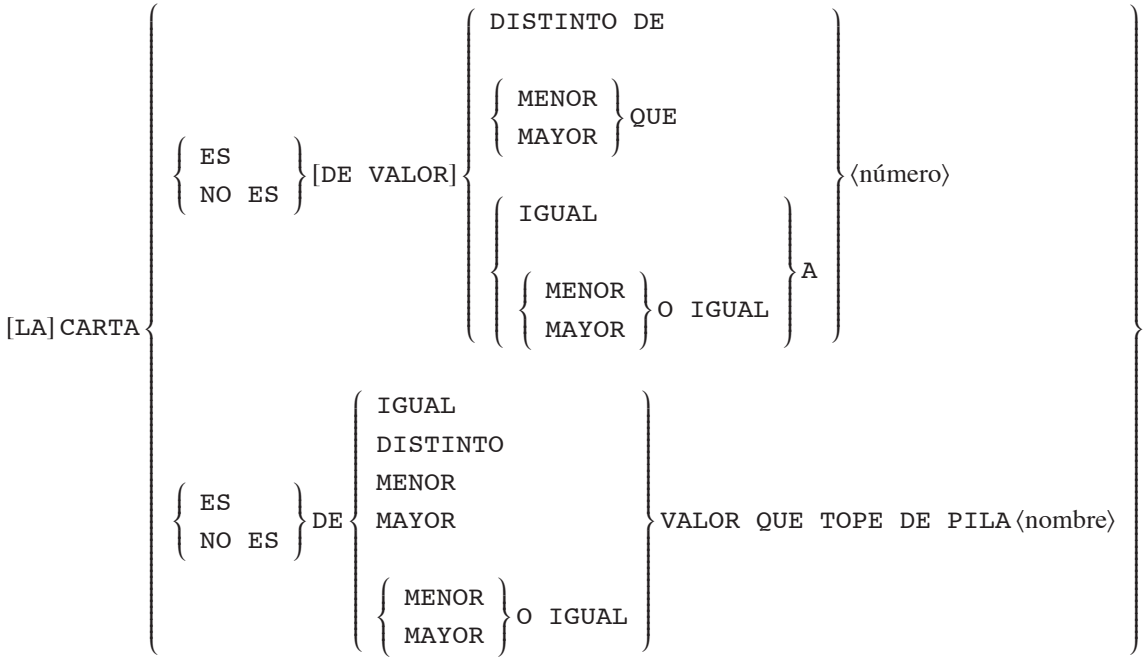
Ejemplo 10: `USTED QUIERE SABER SI LA CARTA ES DEL PALO DEL TOPE DE` ⟨nombre⟩ `PERO LA PILA ESTA VACIA.`

4.4. Valor de Carta

Una carta en "TIMBA" es un elemento biestable (ver Sección 4.2.2 Carta Boca Abajo) con dos atributos: *Palo* y *Valor*. Palo de una carta es uno de los cuatro palos de la baraja española: `OROS`, `BASTOS`, `ESPADAS` o `COPAS`. Valor es un

número del 1 al 7 o del 10 al 12.

Formalmente, la condición de valor de carta se escribe:



En la primera de las dos opciones, UCP compara el valor (1 al 7 o 10 al 12) de la CARTA contra el "número". La comparación se realiza de acuerdo con las reglas comunes para los números naturales, y cada uno de los operadores relacionales (DISTINTO DE, IGUAL A, MENOR QUE, MAYOR QUE, MENOR O IGUAL A y MAYOR o IGUAL A) representan exactamente lo que significan en lenguaje diario.

UCP reconoce la proposición como "Verdadera" si ha sido escrita con la opción ES y la relación se verifica o si ha sido escrita con NO ES y la relación no se verifica. UCP reconoce un error de ejecución si CARTA no está definida, (ver Sección 4.1.1 Sentencias Operativas) o si está BOCA ABAJO (ver Sección 4.2.2 Carta Boca Abajo).

En la segunda opción, UCP compara el valor de CARTA, en el sentido de los números naturales, contra la carta al tope de la PILA nombrada.

UCP reconoce la proposición como "Verdadera" si ha sido escrita con la opción ES y la relación se verifica o si ha sido escrita con NO ES y la relación no se verifica. UCP reconoce un error de ejecución si CARTA no está definida (ver Sección 4.1.1 Sentencias Operativas), si la PILA ESTA VACIA (ver Sección 4.2.1 Pila Vacía) o si por lo menos una de las dos cartas de la comparación está BOCA ABAJO.

Mensajes de error:

Ejemplo 11: USTED QUIERE SABER SI LA CARTA ES DE VALOR ⟨número⟩ , Y YO NO TENGO CARTA.

Ejemplo 12: USTED QUIERE COMPARAR EL VALOR DE LA CARTA CON EL DEL TOPE DE ⟨nombre⟩ , Y YO NO TENGO CARTA.

Ejemplo 13: USTED QUIERE COMPARAR EL VALOR DE LA CARTA CON EL DEL TOPE DE ⟨nombre⟩ ,

`Y LA PILA ESTA VACIA.`

Ejemplo 14: `USTED QUIERE COMPARAR EL VALOR DE LA CARTA CON EL DEL TOPE DE` ⟨nombre⟩ `, PERO LA CARTA ESTA BOCA ABAJO.`

Ejemplo 15: `USTED QUIERE COMPARAR EL VALOR DE LA CARTA CON EL DEL TOPE DE` ⟨nombre⟩ `, PERO EL TOPE ESTA BOCA ABAJO.`

4.5. Descripción de Pila

Formalmente, la descripción de pila se escribe:

$$[\texttt{LA}]\ \texttt{PILA}\ \langle\text{nombre}\rangle \left\{ \begin{array}{l} \texttt{NO TIENE CARTAS} \\ \texttt{TIENE} \left\{ \begin{array}{l} \langle\text{carta boca abajo}\rangle \\ \langle\text{carta boca abajo}\rangle\ \uparrow \end{array} \right\} \left[- \left\{ \begin{array}{l} \langle\text{carta boca abajo}\rangle \\ \langle\text{carta boca abajo}\rangle\ \uparrow \end{array} \right\} \right]^{*} \end{array} \right\}$$

La primera opción hace que UCP asocie al nombre una `PILA` vacía.

La segunda opción hace que UCP asocie al nombre una `PILA` con las cartas (una, dos o más) que se describen después de la palabra clave `TIENE`. La última carta descripta se ubicará al tope de la `PILA`, la penúltima debajo de ésta y así hasta ubicar todas las cartas.

Todas las cartas son descriptas por su palo y su número, y supuestas `BOCA ABAJO`. Si se desea invertir la carta al definir la `PILA`, esto es, se desea que la o las cartas aparezcan `NO BOCA ABAJO` en la `PILA`, después de cada descripción de carta que se desee `NO BOCA ABAJO` debe incluirse el símbolo especial ↑.

Formalmente, carta boca abajo debe escribirse:

$$\left\{ \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 10 \\ 11 \\ 12 \end{array} \right\} \texttt{DE} \left\{ \begin{array}{l} \texttt{OROS} \\ \texttt{BASTOS} \\ \texttt{ESPADAS} \\ \texttt{COPAS} \end{array} \right\}$$

UCP detecta error de previa ejecución si una `PILA` utilizada en la descripción del proceso no fuera descripta, y da un mensaje de atención si fuera descripta una `PILA` no utilizada en la descripción del proceso.

Previo a la Ejecución (PE):

```
0    LA PILA ⟨nombre⟩ NO FUE DESCRIPTA.
```

Atención (AT):

```
1    LA PILA ⟨nombre⟩ FUE DESCRIPTA SIN NECESIDAD.
```

Reconocimientos

El presente apunte es una transcripción mejorada del Apunte “Lenguaje TIMBA”, realizado originalmente para la asignatura Diagramación y Programación de la antigua Licenciatura en Programación de Sistemas (carrera que luego se transformó en la Licenciatura en Ciencias de la Computación).

El dibujo original del apunte era:

5. Ejemplos

Ejemplo 1:

Dado el siguiente problema: *Se tiene una pila de cartas, llamada* `A`*, conteniendo cartas de los cuatro palos. Se pretende obtener una pila donde se encuentren intercaladas la mayor cantidad posible de cartas de palo* `OROS` *y de palo* `COPAS`.

Un programa que resuelve el problema, utilizando el lenguaje TIMBA y como nombres de PILAS sólo las primeras letras mayúsculas del alfabeto castellano y que considera la siguiente situación inicial de las pilas en la declaración: `1 DE OROS ↑ - 2 DE OROS ↑ - 1 DE COPAS ↑ - 3 DE OROS - 4 DE COPAS - 3 DE ESPADAS ↑ - 1 DE BASTO - 1 DE ESPADA - 5 DE ESPADA ↑ - 5 DE OROS ↑ - 6 DE COPAS - 10 DE COPAS ↑`
se muestra a continuación:

```
DEFINICION DE PROGRAMA
MIENTRAS LA PILA A NO ESTA VACIA
  TOME UNA CARTA DE LA PILA A,
  SI LA CARTA ESTA BOCA ABAJO
    INVIERTA LA CARTA
  SINO NADA MAS,
  SI LA CARTA ES DEL PALO OROS
    DEPOSITE LA CARTA EN LA PILA B
  SINO
    SI LA CARTA ES DEL PALO COPAS
      DEPOSITE LA CARTA EN LA PILA C
    SINO
      DEPOSITE LA CARTA EN LA PILA D
    NADA MAS
  NADA MAS
REPITA,
MIENTRAS LA PILA B NO ESTA VACIA Y LA PILA C NO ESTA VACIA
  TOME UNA CARTA DE LA PILA B,
  DEPOSITE LA CARTA EN LA PILA A,
  TOME UNA CARTA DE LA PILA C,
  DEPOSITE LA CARTA EN LA PILA A
REPITA;
UCP EJECUTE CON LAS SIGUIENTES CARTAS: PILA A TIENE 1 DE OROS ↑ - 2 DE OROS ↑ -
                                       1 DE COPAS ↑ - 3 DE OROS - 4 DE COPAS -
                                       3 DE ESPADAS ↑ - 1 DE BASTO -1 DE ESPADA -
                                       5 DE ESPADA ↑ - 5 DE OROS ↑ - 6 DE COPAS -
                                       10 DE COPAS ↑, PILA B NO TIENE CARTAS,
                                       PILA C NO TIENE CARTAS,
                                       PILA D NO TIENE CARTAS.
```

Se muestra aquí una posible solución al problema. Existen muchas otras posibilidades de declaraciones y de procesos algorítmicos que resuelvan el problema adecuadamente.

Tal como lo establece la sintaxis del lenguaje TIMBA, el programa está compuesto de dos partes bien diferenciadas: el *proceso algorítmico* y la *descripción de datos de pila*. El proceso algorítmico en sí mismo comienza luego de las palabras clave `DEFINICION DE PROGRAMA`, donde cada una de las sentencias que lo componen se separan entre sí por ",", y finaliza con ";". La descripción de datos de pila comienza luego del ";" y de las palabras clave `UCP EJECUTE CON LAS SIGUIENTES CARTAS:`, donde el contenido de cada pila que compone la descripción de datos se separa entre sí por "," y finaliza con ".".

El estado final obtenido en cada una de las pilas luego de ejecutar el programa con la declaración presentada sería:

```
PILA A TIENE 1 DE OROS ↑ - 1 DE COPAS ↑ - 2 DE OROS ↑ - 4 DE COPAS ↑ - 3 DE OROS ↑ - 6 DE
COPAS ↑ - 5 DE OROS ↑ - 10 DE COPAS ↑, PILA B NO TIENE CARTAS, PILA C NO TIENE CARTAS, PILA
D TIENE 5 DE ESPADA ↑ - 1 DE ESPADA ↑ - 1 DE BASTO ↑ - 3 DE ESPADAS ↑
```

Ahora, si se mantiene el mismo proceso algorítmico y se cambia la descripción de datos de pila del programa, combinando el proceso algorítmico con cada una de las alternativas que se muestran:

- `UCP EJECUTE CON LAS SIGUIENTES CARTAS: PILA A TIENE 1 DE OROS ↑ - 2 DE OROS ↑ - 1 DE COPAS ↑ - 3 DE OROS - 4 DE COPAS - 3 DE ESPADAS ↑ - 1 DE BASTO - 1 DE ESPADA - 5 DE ESPADA ↑ - 5 DE OROS ↑ - 6 DE COPAS - 10 DE COPAS ↑, PILA B NO TIENE CARTAS, PILA C NO TIENE CARTAS, PILA D NO TIENE CARTAS.`

  El estado final de las pilas, luego de ejecutar el programa sería:

  ```
  PILA A TIENE 1 DE OROS ↑ - 1 DE COPAS ↑ - 2 DE OROS ↑ - 4 DE COPAS ↑ - 3 DE OROS ↑ - 6
  DE COPAS ↑ - 5 DE OROS ↑ - 10 DE COPAS ↑, PILA B NO TIENE CARTAS, PILA C NO TIENE CARTAS,
  PILA D TIENE 5 DE ESPADA ↑ - 1 DE ESPADA ↑ - 1 DE BASTO ↑ - 3 DE ESPADAS ↑
  ```

- `UCP EJECUTE CON LAS SIGUIENTES CARTAS: PILA A TIENE 1 DE OROS ↑ - 2 DE OROS ↑ - 1 DE COPAS ↑ - 3 DE OROS - 4 DE COPAS - 3 DE ESPADAS ↑ - 1 DE BASTO - 1 DE ESPADA - 5 DE ESPADA ↑ - 5 DE OROS ↑, PILA B NO TIENE CARTAS, PILA C NO TIENE CARTAS, PILA D NO TIENE CARTAS, PILA E NO TIENE CARTAS.`

  El estado final de las pilas, luego de ejecutar el programa sería:

  ```
  PILA A TIENE 1 DE OROS ↑ - 1 DE COPAS ↑ - 2 DE OROS ↑ - 4 DE COPAS ↑, PILA B TIENE 5 DE
  OROS ↑ - 3 DE OROS ↑, PILA C NO TIENE CARTAS, PILA D TIENE 5 DE ESPADA ↑ - 1 DE ESPADA
  ↑ - 1 DE BASTO ↑ - 3 DE ESPADAS ↑, PILA E NO TIENE CARTAS
  ```

  Además, UCP da el siguiente mensaje de atención:

  ```
  LA PILA E FUE DESCRIPTA SIN NECESIDAD
  ```

- `UCP EJECUTE CON LAS SIGUIENTES CARTAS: PILA A TIENE 1 DE OROS ↑ - 2 DE OROS ↑ - 1 DE COPAS ↑ - 3 DE OROS - 4 DE COPAS - 3 DE ESPADAS ↑ - 1 DE BASTO - 1 DE ESPADA - 5 DE ESPADA ↑ - 5 DE OROS ↑ - 6 DE COPAS, PILA B NO TIENE CARTAS, PILA C NO TIENE CARTAS.`

UCP detectaría el siguiente error previo a la ejecución:

```
LA PILA D NO FUE DESCRIPTA.
```

Ejemplo 2:

Dado el siguiente problema: *Se tiene una pila de cartas no vacía, llamada* `A`*, conteniendo cartas de los cuatro palos. Si en la pila* `A` *se encuentra el 4 de OROS, se pretende obtener una pila, llamada* `B`*, con todas las cartas que se encuentran por debajo del 4 de OROS en el mismo estado y en el mismo orden en el que se encuentran en la pila original, dejando al 4 de OROS como tope.*

Aunque existen muchas posibilidades de procesos algorítmicos que resuelvan el problema adecuadamente, una de las posibles soluciones que resuelve el problema utilizando el lenguaje TIMBA y como nombres de PILAS sólo las primeras letras mayúsculas del alfabeto, sería la siguiente:

```
DEFINICION DE PROGRAMA
MIENTRAS LA PILA A NO ESTA VACIA Y LA PILA C ESTA VACIA
  TOME UNA CARTA DE LA PILA A,
  SI LA CARTA ESTA BOCA ABAJO
    INVIERTA LA CARTA
  SINO NADA MAS,
  SI LA CARTA ES DE VALOR IGUAL A 4 Y LA CARTA ES DEL PALO OROS
    DEPOSITE LA CARTA EN LA PILA C
  SINO
    DEPOSITE LA CARTA EN LA PILA D
  NADA MAS
REPITA,
MIENTRAS LA PILA A NO ESTA VACIA
  TOME UNA CARTA DE LA PILA A,
  DEPOSITE LA CARTA EN LA PILA C
REPITA,
MIENTRAS LA PILA C NO ESTA VACIA
  TOME UNA CARTA DE LA PILA C,
  DEPOSITE LA CARTA EN LA PILA B
REPITA;
```

Cabe destacar que este *proceso algorítmico* podrá dar lugar a diferentes *programas* al combinarlo con distintas *descripciones de pilas*. Por lo tanto, el resultado de una ejecución, en caso de no existir errores en el programa completo, dará lugar a diferentes salidas posibles.

Ejercicio para el lector: Plantear diferentes programas con este proceso algorítmico, combinándolo con diferentes descripciones de pilas, de manera tal de comprobar distintas posibilidades de ejecución y distintas salidas posibles del programa.

A. Apéndice: Palabras reservadas de “TIMBA”

La siguiente es una lista de palabras reservadas de “TIMBA”. Estas palabras no pueden ser usadas como nombres de pila.

ABAJO	INVIERTA	SINO
BASTOS	INVIERTALA	TOME
BOCA	LA	↑
CARTA	LAS	TOPE
CARTAS	MAS	UCP
CON	MAYOR	UNA
COPAS	VACIA	VACIA
DE	MENOR	VALOR
DEL	MIENTRAS	Y
DEFINICION	NADA	1
DEPOSITE	NO	2
DEPOSITELA	O	3
DISTINTO	OROS	4
EJECUTE	PILA	5
EN	PROGRAMA	6
ES	QUE	7
ESPADAS	REPITA	10
ESTA	SI	11
IGUAL	SIGUIENTES	12

B. Apéndice: Simbología Usada en la Descripción de Sentencias

En la descripción formal de sentencias de “TIMBA” aparecen cuatro elementos: corchetes angulares (⟨ ⟩), corchetes ([ ]), llaves ({ }) y asteriscos (∗); las palabras claves de “TIMBA” (Ver Apéndice A), los signos de puntuación de “TIMBA” (`;`, `,`, `:` y `–`) y algunas palabras en minúscula las que van siempre entre corchetes angulares.

Las palabras que aparecen en mayúscula en una descripción formal son palabras claves de “TIMBA”, y por lo tanto deben escribirse tal como aparecen. Las palabras en minúscula, encerradas entre corchetes angulares deben reemplazarse por su definición que se encuentra siempre en algún punto de este Manual. Por ejemplo, la descripción formal (ver Sección 4.1.2 Sentencia de Selección `SI`):

`SI` ⟨condición⟩ ⟨sentencia⟩ { `SINO` ⟨sentencias⟩ `NADA MAS` | `SINO NADA MAS` }

contiene las palabras claves `SI`, `SINO`, `NADA` y `MAS`. Cuando un programador quiere escribir una sentencia de selección `SI` debe empezar por la palabra clave `SI`, escribir a continuación lo que se escribe en ⟨condición⟩ como una condición válida en “TIMBA”, luego las sentencias que él desee (ver Sección 4.1 Sentencias). Puesto que condición y sentencias están escritas con minúscula y entre corchetes angulares en la descripción formal, el programador las reemplaza por otros elementos más sencillos. Excepción a la regla de hallarse descripta en el manual es la palabra nombre, que cuando es hallada en una descripción formal debe ser reemplazada por una cadena de hasta diez caracteres alfabéticos o numéricos que no sea una palabra clave de “TIMBA”, por ejemplo `MI1`, `1000`, `X1`.

Siguiendo con el ejemplo de la sentencia de selección, veamos el uso de las llaves: cuando en una descripción aparecen dos o más ítems entre llaves, el programador debe elegir entre ellos aquél que se ajuste a su problema. En el ejemplo de la sentencia `SI`, el programador debe elegir entre dos alternativas: escribir la sentencia `SI` con la opción `SINO` ⟨sentencias⟩ `NADA MAS` o con la opción `SINO NADA MAS`. En el primer caso deberá elegir las sentencias que incluirá en el segundo deberá copiar las palabras claves en ese orden.

Cuando en una descripción formal aparece un ítem entre corchetes ([ ]) el programador puede optar por escribir o no este ítem, sin que se modifique por ello la interpretación que UCP hace del programa. Por ejemplo, la descripción formal (ver Sección 4.1.1 Sentencias Operativas):

`TOME` [`UNA` [`CARTA`]] `DE` [`LA`] `PILA` ⟨nombre de pila⟩

contiene tres pares de corchetes, cada uno de los cuales encierra, respectivamente, `UNA CARTA`, `CARTA` y `LA`. Eso significa que el programador puede escribir una sentencia `TOME` omitiendo `UNA CARTA` o bien omitiendo `CARTA` o bien `LA` o bien las tres palabras claves: `TOME DE PILA XXX`, `TOME DE LA PILA XXX`, `TOME UNA DE LA PILA XXX`, `TOME UNA DE PILA XXX`, `TOME UNA CARTA DE PILA XXX` y `TOME UNA CARTA DE LA PILA XXX` son todas indistintas para UCP.

Por último, si un ítem se encuentra entre corchetes seguidos de un asterisco (∗), el programador puede omitir el ítem o escribirlo una, dos o más veces, de acuerdo con sus necesidades. Por ejemplo (ver Sección 4.2 Condición):

$$\langle \text{proposición} \rangle \left[\left\{ \begin{array}{c} \texttt{Y} \\ \\ \texttt{O} \end{array} \right\} \langle \text{proposición} \rangle \right]^*$$

significa que puede escribirse una, dos o más proposiciones, vinculadas por los operadores lógicos `Y` u `O`.

Para redactar un programa “TIMBA”, lea primero la Sección 4 Programa. Al finalizar dicha Sección encontrará la descripción formal de un programa “TIMBA”.

C. Apéndice: Descripción BNF de la gramática de “TIMBA”

BNF es la abreviatura de Backus-Normal Form, o Backus Naur Form.

⟨Programa timba⟩ ::= ⟨proceso⟩ **;** ⟨declaraciones⟩ **.**
⟨proceso⟩ ::= `DEFINICION DE PROGRAMA` ⟨sentencias⟩
⟨sentencias⟩ ::= ⟨sentencia⟩ | ⟨sentencias⟩ **,** ⟨sentencia⟩
⟨sentencia⟩ ::= ⟨operativa⟩ | ⟨de control⟩
⟨operativa⟩ ::= ⟨tomar⟩ `DE` ⟨pila⟩ ⟨nombre⟩ | ⟨depositar⟩ `EN` ⟨pila⟩ ⟨nombre⟩ | ⟨invertir⟩
⟨tomar⟩ ::= `TOME UNA CARTA` | `TOME UNA` | `TOME`
⟨depositar⟩ ::= `DEPOSITE LA CARTA` | `DEPOSITELA`
⟨invertir⟩ ::= `INVIERTA LA CARTA` | `INVIERTALA`
⟨de control⟩ ::= ⟨de selección⟩ | ⟨iterativa⟩
⟨de selección⟩ ::= `SI` ⟨condición⟩ ⟨sentencias⟩ `SINO` ⟨sentencias⟩ `NADA MAS` | `SI` ⟨condición⟩ ⟨sentencias⟩ `SINO NADA MAS`
⟨iterativas⟩ ::= `MIENTRAS` ⟨condición⟩ ⟨sentencias⟩ `REPITA`
⟨condición⟩ ::= ⟨condición simple⟩ | ⟨condición⟩ `Y` ⟨condición simple⟩ | ⟨condición⟩ `O` ⟨condición simple⟩
⟨condición simple⟩ ::= ⟨condición de pila vacía⟩ | ⟨condiciones de carta⟩
⟨condición de pila vacía⟩ ::= ⟨pila⟩ ⟨nombre⟩ `ESTA VACIA` | ⟨pila⟩ ⟨nombre⟩ `NO ESTA VACIA`
⟨condiciones de carta⟩ ::= ⟨estado⟩ | ⟨característica⟩
⟨estado⟩ ::= ⟨carta⟩ `ESTA BOCA ABAJO` | ⟨carta⟩ `NO ESTA BOCA ABAJO`
⟨característica⟩ ::= ⟨carta⟩ ⟨es o no es⟩ ⟨de palos⟩ | ⟨carta⟩ ⟨es o no es⟩ ⟨valor⟩ | ⟨carta⟩ `ES DE` ⟨relac⟩ `PALO QUE TOPE DE` ⟨pila⟩ ⟨nombre⟩
⟨es o no es⟩ ::= `ES` | `NO ES`
⟨carta⟩ ::= `LA CARTA` | `CARTA`
⟨pila⟩ ::= `LA PILA` | `PILA`
⟨de palos⟩ ::= `DEL PALO` ⟨palos⟩ | ⟨palos⟩
⟨palos⟩ ::= `OROS` | `BASTOS` | `ESPADAS` | `COPAS`
⟨relac⟩ ::= `IGUAL` | `DISTINTO`
⟨valor⟩ ::= `DE` ⟨rela⟩ `VALOR QUE TOPE DE` ⟨pila⟩ ⟨nombre⟩ | `DE VALOR` ⟨relación⟩ ⟨número⟩ | ⟨relación⟩ ⟨número⟩
⟨rela⟩ ::= `IGUAL` | `DISTINTO` | `MAYOR` | `MENOR` | `MAYOR O IGUAL` | `MENOR O IGUAL`
⟨relación⟩ ::= `IGUAL A` | `DISTINTO DE` | `MAYOR QUE` | `MENOR QUE` | `MAYOR O IGUAL A` | `MENOR O IGUAL A`
⟨número⟩ ::= `1` | `2` | `3` | `4` | `5` | `6` | `7` | `10` | `11` | `12`
⟨declaraciones⟩ ::= `UCP EJECUTE CON LAS SIGUIENTES CARTAS:` ⟨lista de pilas⟩
⟨lista de pilas⟩ ::= ⟨descripción de pila⟩ | ⟨lista de pilas⟩ **,** ⟨descripción de pila⟩
⟨descripción de pila⟩ ::= ⟨pila⟩ ⟨nombre⟩ ⟨contenido⟩
⟨contenido⟩ ::= `NO TIENE CARTAS` | `TIENE` ⟨lista de cartas⟩
⟨lista de cartas⟩ ::= ⟨descripción de carta⟩ | ⟨lista de cartas⟩ **–** ⟨descripción de carta⟩
⟨descripción de carta⟩ ::= ⟨número⟩ `DE` ⟨palos⟩ | ⟨número⟩ `DE` ⟨palos⟩ ↑
⟨nombre⟩ es una cadena que no tiene blancos ni símbolos especiales de diez o menos caracteres alfabéticos o numéricos que no formen una palabra reservada.