CAPÍTULO 11

OBJETOS COMPUESTOS

Los objetos compuestos contienen otros objetos dentro de ellos. A diferencia con las series (ARRAYS) los diferentes tipos de objetos forman un objeto compuesto y a la vez éste puede contener otro objeto compuesto. Ya se mencionó los objetos compuestos en la introducción, cuando usamos un secundario para agrupar varios órdenes en uno solo.

Todos los objetos compuestos son similares en su estructura: empiezan con una palabra que varía, dependiendo del tipo del objeto y acaba con la palabra SEMI.

Además de los secundarios, son objetos compuestos también las listas, los objetos simbólicos (descrito en el capítulo 14) y los objetos de unidad (descrito en el capítulo 13).

Usted puede crear una lista empezando con {, y terminando con }. Dentro de él ingrese cualquier tipo de objeto. Los secundarios se delimitan con :: y ;

Para concatenar (unir) dos objetos compuestos use &COMP, o simplemente agregue a la cabeza (inicio) o cola (final) de un objeto compuesto, primero ponga el objeto compuesto en la pila, luego el objeto y llama >HCOMP o >TCOM respectivamente. Para conseguir la longitud del objeto compuesto (el número de objetos, en números binarios), simplemente ponga el objeto compuesto en nivel uno y use el orden LENCOMP. Para extraer todo los elementos del objeto compuesto y un recuento de objetos use el comando NNERCOMP, etc.

11.1 REFERENCIA:

11.1.1 OPERACIONES GENERALES:

DIRECCIÓN	NOMBRE	DESCRIPCIÓN
0521F	&COMP	(comp1 comp2 → comp1 + comp2) Une (concatena) dos objetos compuestos, es parecido cuando tenemos dos listas en la pila y pulsas mas (+).
052FA	>TCOMP	(comp1 obj → comp1 + obj) Inserta un objeto al lugar n+1 de la lista, es decir el objeto insertado será el objeto del último lugar en el objeto compuesto.
052C6	>HCOMP	(obj comp1 → obj + comp1) Añade el objeto al inicio del objeto compuesto, inserta el objeto al lugar uno de la lista.
05089	CARCOMP	(comp1 → obj_inicial) (comp_null → comp_null) Extrae el primer objeto, saca el objeto del primer lugar si en el objeto compuesto hay elementos, si el objeto compuesto es nulo (no tiene elementos) entonces devuelve (tal vez no hace nada) el objeto compuesto nulo.

361C6	?CARCOMP	(comp T → obj_inicial) (comp F → comp) Depende de la bandera (Flag) si es TRUE saca el primer elemento, si es falso devuelve el objeto compuesto.
05153	CDRCOMP	(comp → comp-ob_inicial) (comp_null→ comp_null) Devuelve el objeto compuesto excepto el primer elemento si el objeto compuesto tiene elementos, en caso contrario no hace nada.
2BC006	^LASTCOMP	(comp → obj_final) Obtiene el último elemento de un objeto compuesto.
0567B	LENCOMP	(comp → #(número de elementos)) Retorna el número de elementos de un objeto compuesto.
3627A	DUPLENCOMP	(comp → comp #n) Igual que el anterior, pero antes ejecuta DUP (duplica el objeto del primer nivel)
055B7	NULLCOMP?	(comp → flag) Pregunta si es un objeto compuesto nulo, es decir que no tiene electos.
36266	DUPNULLCOMP?	(comp → comp flag) Igual que el anterior, pero antes duplica el objeto compuesto.
056B6	NTHELCOMP	(comp #i → obj T) (comp #i → F) Retorna el elemento indicado si el índice(#i) es menor o igual a la longitud del objeto compuesto y TRUE (verdadero), de lo contrario devuelve FALSE.
35BC3	NTHCOMPDROP	(comp #i → obj) Hace lo mismo que el comando anterior, pero además borra el indicador (flag).
25D58	NTHCONDDUP	(comp #i → ob ob) Ejecuta NTHCOMPDROP y DUP
376EE	POSCOMP	(comp ob pred → #i) (comp ob pred→ #0) (eg: pred= ‘ %<) Este comando compara cada elemento del objeto compuesto con “ob”, dependiendo de “pred” será el tipo de comparación, termina el proceso cuando encuentra una primera comparación verdadera, devolviendo el índice, de lo contrario si todas las comparaciones dan falso al finalizar devuelve #0. En este ejemplo el número devuelto será #4 “:: { %1 %2 %3 %-4 %-5 %6 %7} %0 ‘ %< POSCOMP ; @ “
3776B	EQUALPOSCOMP	(comp ob → #ipos) (comp ob → #0) Ejecuta POSCOMP con test(pred) EQUAL, es decir busca en todo los elementos del objeto compuesto “ob” y devuelve el índice que indica la posición del objeto, de lo contrario devuelve #0.

COMENTARIOS: edcivilic@lycos.es

37784	NTHOF	(ob comp → #i) (ob comp → #0) Antes hace SWAP, luego ejecuta EQUALPOSCOMP
0FD006	^ListPos	(ob {} → #i / #0) Equivalente a NTHOF, pero solo funciona con listas.
37752	# =POSCOMP	(comp # → #i) (comp # → #0) Ejecuta POSCOMP con #= como prueba. Éste trabaja con número binarios.
05821	SUBCOMP	(comp #m #n → comp’) Retorna la aparte especificada del objeto compuesto
376B7	matchob?	(ob comp → T) (ob comp → ob F) Retorna TRUE si el objeto es igual a cualquier elemento del objeto compuesto.
371B3	Embedded?	(bo1 ob2 → flag) Retorna TRUE si ob2 se encuentra dentro o es igual a ob1, de lo contrario devuelve FALSE
37798	FindlstTrue	(comp test → ob T) (comp test → F) Verifica el test para cada elemento, si se encuentra el elemento devuelve el objeto y TRUE, en caso contrario FALSE. Por ejemplo, este programa devuelve %-4 y TRUE :: {%1 %2 %3 %-4 %-5 %6 %6} ‘ %< FindlsTrue ; @
377C5	Lookup	(ob test comp → nextob T) (ob test comp → ob F) : Hace las pruebas con cada elemento de lugar impar del objeto compuesto, si se verifica verdadero en uno de las pruebas entonces devuelve el elemento siguiente y TRUE, en caso de no encontrar verdadero devuelve el objeto y FALSE.
377DE	Lookup.1	(ob test → nextob T) (ob test → ob F) Ejecuta Lookup con el objeto compuesto en la pila.
37829	EQLookup	(ob comp → nextob T) (ob comp → ob F) Ejecuta Lookup, buscando una igualdad

11.1.2 CONSTRUYENDO

También existen comandos para construir listas, especificando el número de elementos, los cuales se describen a continuación.

DIRECCIÓN	NOMBRE	DESCRIPCIÓN
05459	{}N	((obn…ob1 #n→ { obn…ob1}) Necesita los objetos y la cantidad de objetos en números binarios en la pila lo cual retorna en una lista.
05445	**::N**	((ob1 … obn #n → :: ob1 … obn ;) Necesita los objetos y la cantidad de objetos en números binarios en la pila lo cual retorna en un secundario.
0546D	SYMBN	(ob1 … obn #n → symb) Crea un objeto simbólico.
05481	EXTN	(ob1 … obn #n → u) Devuelve un objeto de unidad.
293F8	P{}N	(ob1 … obn #n → { }) La lista de la figura con la posible colección de basura

11.1.3 EXPLOTANDO

DIRECCIÓN	NOMBRE	DESCRIPCIÓN
054AF	INNERCOMP	(comp. → ob1 … obn #n) Retorna todos los elementos del objeto compuesto y el recuento de esos elementos.
3622A	DUPINCOMP	(comp. → comp ob1 … obn #n) Lo mismo que el comando anterior, pero antes ejecuta DUP
3623E	SWAPINCOMP	(comp. ob → ob ob1 … obn #n) Ejecuta INNERCOMP pero antes hace SWAP.
35BAF	INCOMPDROP	(comp. → ob1 … obn) Ejecuta INNERCOMP y luego DROP
35C68	INNERDUP	(comp. → ob1 … obn #n #n) Ejecuta INNERCOMP luego DUP.
2F0EC	ICMPDRPRTDRP	(comp. → obn …. ob4 ob2 ob1) Ejecuta INNERCOMP luego ROTDROP
366E9	INNER#=	(comp. → obn … ob1 falg) Lo mismo que INNERCOMP luego evalua si el número e electos es uno en enteros binarios.
157006	^SYMBINCOMP	(symb. → ob1 … obn #n) (ob → ob #1) ({}→ {} #1) Retorna todos los elementos de un objeto simbólico y un recuento de todos ellos, si es cualquier objeto o una lista devuelve el objeto y #1
12A006	^2 SYMBINCOMP	(ob1 ob2 → meta1 meta2) Ejecuta ^SYMBINCOMP para dos objetos
158006	^CKINNERCOMP	({}→ ob1 … obn #n) (ob → ob #1) Explota una lista en un objeto meta, otros

		objetos se convierten en metas y es empujado #1 a la pila

11.1.4 LISTAS

DIRECCIÓN	NOMBRE	DESCRIPCIÓN
055E9	NULL{}	(→ {}) Ingresa una lista vacía a la pila.
36ABD	DUPNULL{}?	({}→ {} flag) Pregunta si es una lista nula.
159006	^DUPCKLEN{}	({}→ {} #n) (ob → ob #1) Duplica y devuelve el número de elementos de la lista, si no es una lista devuelve el objeto y #1
29D18	ONE{}N	(ob→ { ob }) Devuelve un objeto en una lista.
36202	TWO{}N	(ob1 ob2→ { ob1 ob2 }) Devuelve dos objetos en una lista
36216	THREE{}N	(ob1 ob2 ob3→ { ob1 ob2 ob3}) Devuelve tres objetos en una lista.
361EE	#1-{}N	(ob1 … obn #n+1→ {}) Devuelve en una lista los elementos desde uno hasta el nivel #n.
2B42A	PUTLIST	(ob #i {}→ {}') Reemplaza el objeto del lugar #i, #i debe ser como máximo la longitud de la lista.
2FC006	^INSERT{}N	({} ob #→ {}') Inserta el objeto a la lista en la posición indicada.
2FB006	^NEXTPext	(list → list1 list2) Extrae el primer elemento de la lista y devuelve la lista inicial menos el primer elemento y ese primer elemento en una lista, es decir hace: list1=list – list2
2FD006	^COMPRIMext	({}→ {}') Devuelve una lista modificada, suprime los elementos que se repite
15A006	^CKCARCOMP	({}→ ob1) (ob → ob) Devuelve el primer elemento de la lista, si no es una lista devuelve el objeto.
2EF5A	apndvarlst	({} ob → {}') Añade el objeto a la lista, éste será el último lugar. Antes verifica si el objeto existe en la lista, si existe no lo añade y si no existe lo adiciona.
0FE006	^AppendList	({} ob → {}') Igual que el comando anterior, pero más rápidamente
4EB006	^prepvarlist	({} ob → {}') Añade el objeto a la lista, éste será el primer elemento de la lista, antes verifica la existencia del elemento y sólo adiciona

		si el objeto no existe en la lista.
100006	^SortList	(L pred → L’) El comando se comporta según pred. “pred” es un programa que prueba dos elementos, devuelve FALSE o TRUE según la comparación de esos dos elementos, esto sirve por ejemplo para ordenar números.
18A006	^PIext	({} → ob) Retorna el producto entre todos los elementos de la lista.
25ED3	EqList?	(ob →) ¿el objeto es una lista de ecuaciones?, devuelve TRUE si el objeto es de por lo menos dos elementos o FALSE si es otro objeto.

11.1.5 SECUNDARIOS

DIRECCIÓN	NOMBRE	DESCRIPCIÓN
055FD	NULL::	(→ :: ;) Retorna un secundario nulo.
37073	Ob>Seco	(ob → :: ob ;) Inserta #1 a la pila luego ejecuta ::N
3705A	?Ob>Seco	(ob → :: ob ;) Si el objeto no es un secundario, hace Ob>Seco.
37087	2Ob>Seco	(ob1 ob2 → :: ob1 ob2 ;) Inserta #2 a la pila luego ejecuta ::N
3631A	::NEVAL	(ob1… obn →) Hace **::N** luego eval

EJEMPLOS

- Este programa genera diez números reales aleatorios los cuales lo agrupa en una lista y luego lo ordena ascendentemente.

```
::                  @Inicia system
  CK0               @no necesita argumento
  NULL{}            @lista nula
  BINT10            @fin del bucle
  #1+_ONE_DO        @inicia bucle con inicio #1
  %RAN              @genera número real aleatorio
  >TCOMP            @añade al último lugar
  LOOP              @termina el bucle
  '
  %>                @compara
  ^SortList         @ordena
;                   @fin del programa
@                   @código para ensamblar con masd
```

- Este programa halla el producto de todos los elementos de la lista, la lista debe contener solamente números reales.

```
::
  CK1NOLASTWD       @verifica que existe un objeto
  CK1&Dispatch      @envía el tipo de objeto
  BINT5             @verifica que es una lista
  ^Plext            @ordena
;
@
```

- Mira esta otra forma:

```
::
  CK1NOLASTWD       @verifica que exista un objeto
  CK1&Dispatch      @envía el tipo de objeto
  BINT5             @verifica que el objeto es lista
  ::                @inicia subprograma
    %1              @número real uno
    SWAP            @cambia los niveles 1 y 2
    ^DUPCKLEN{}     @duplica y halla la longitud
    #1+_ONE_DO      @bucle DO
    DUP             @duplica
    INDEX@          @el índice
    NTHCOMPDROP     @saca un objeto de la lista
    ROT             @tercer nivel al primero
    %*              @multiplica dos números reales
    SWAP
    LOOP
    DROP            @elimina  objeto del nivel uno
```

```
  ;                    @termina subprograma
;                      @termina el programa
@
```

- y esta otra

```
::
  CK1NOLASTWD
  CK1&Dispatch
  BINT5
  ::
    INNERCOMP          @explota y hace un recuento
    ONE_DO
    %*
    LOOP
  ;
;
@
```

- Este programa reduce los elementos que se repiten en una lista, devolviendo la lista modificada.

```
::
  {
    BINT1              @número binario #1
    BINT1
    BINT2
    BINT2
    BINT3
    BINT3
    BINT4
    BINT4
    BINT5
    BINT5
    "X"                @cadena
  }
  ^COMPRIMext          @suprime
;
@
```

- Este programa halla la longitud de un objeto compuesto, el objeto compuesto debe estar en el primer nivel de la pila.

```
::
  CK1NOLASTWD
  DUPTYPELIST?                @pregunta si es una lista
  Case                        @decisión
  LENCOMP                     @si es verdad
  "Error: Ingrese lista"      @si es falso
  FlashWarning
;
@
```

- Programa que construye el triángulo de Pascal, necesita un número real en la pila, este número representa el exponente.

```
::                            @inicia system
  DUPTYPEREAL?                @duplica y pregunta si el objeto es número real
  Case                        @proceso que ejecuta si es verdad
  ::
    {  %1  }
    SWAP
    COERCE                    @convierte un número real a un entro bynario
    #1+_ONE_DO
    INDEX@
    1LAMBIND                  @nombre nulo
    ::
      NULL{}
      1GETLAM                 @se llama al contenido del nombre nulo
      #1+
      ZERO_DO
      1GETLAM
      INDEX@
      UNCOERCE2               @convierte dos número bynarios a reales
      %COMB                   @halla la combinatoria
      >TCOMP                  @inserta al final de la lista
      LOOP
    ;
    LOOP
    1GETLAM
    reversym
    SWAPDROP
    #1-
    reversym
    DROP
  ;
  "Error: Ingrese un número"  @si es falso
  FlashWarning
;
@
```

Autor: Canchari Gutiérrez, Edmundo

Comentarios: edcivilic@lycos.es