Ejercicio 1: analizador morfológico

- **Analizador morfológico de verbos regulares de la 1ª conjugación en modo indicativo (tiempos simples)**
  - Identificar el infinitivo del verbo, el tiempo, la persona y el número
  - Por ejemplo
    - Hablará
    - Hablar, futuro, 3ª persona, singular

Ejercicio 1: diccionario

- Necesitamos incluir información en el diccionario
  - Verbos regulares de la 1ª conjugación
    - Infinitivo = Raíz ++ “ar”
    - Conjugaciones = Raíz ++ Terminación
  - Debemos identificar las terminaciones

```
% es_terminación(Termin, Tiempo, Persona, Número)

es_terminación(o, presente, 1, singular).
es_terminación(as, presente, 2, singular).
...
es_terminación(aré, futuro, 1, singular).
...
```

Ejercicio 1: diccionario

- Necesitamos incluir información en el diccionario
  - Verbos regulares de la 1ª conjugación
    - Infinitivo = Raíz ++ “ar”
    - Conjugaciones = Raíz ++ Terminación
  - Debemos enumerar las raíces de los verbos que queremos reconocer conjugados

```
% es_verbo(Raíz, Infinitivo)
es_verbo(am, amar).
es_verbo(habl, hablar).
es_verbo(jug, jugar).
es_verbo(peg, pegar).
...
```

Ejercicio 1: DCG analizador morfológico

- Sólo necesitamos una regla
  - Verbos regulares de la 1ª conjugación
    - Infinitivo = Raíz ++ “ar”
    - Conjugaciones = Raíz ++ Terminación
  - Estructura básica

```
verbo --> [V].
```

  - Necesitamos parámetros para devolver el resultado y hacer todo el trabajo de descomposición

```
verbo(Inf, Tpo, Pers, Num) --> [V], { ... }.
```

  - Leeremos un átomo y tenemos que descomponerlo. Para ello, necesitamos saber cómo hacerlo en Prolog
    - Transforma un átomo en una lista de caracteres (códigos ascii)

```
% name(?Átomo0Int, ?Cadena)
```

Necesita al menos 1 argumento instanciado: sirve en los 2 sentidos

Así podemos usar predicados de listas para la descomposición

Ejercicio 1: DCG analizador morfológico

- Necesitamos parámetros para devolver el resultado y hacer todo el trabajo de descomposición

```
verbo(Inf, Tpo, Pers, Num) -->
        [V],
        {
          name(V, VerboCad),
          append(RaízCad, TermCad, VerboCad),
          name(Raíz, RaízCad),
          es_verbo(Raíz, Inf),
          name(Term, TermCad),
          es_terminación(Term, Tpo, Pers, Num)
        }.
```

Ejercicio 1: uso de la DCG

- Como el símbolo inicial de la gramática es

```
verbo
```

y tiene 4 parámetros, el uso en Prolog es

```
?- verbo(Inf, Tpo, Pers, Num, [hablarás], []).
Inf = hablar
Tpo = futuro
Pers = 2
Num = singular ;

No
```

Ejercicio 2: concordancia

- **Construir una DCG que compruebe la concordancia pronombre-verbo en frases sencillas del tipo**

  <pronombre_personal> <verbo>

  utilizando el analizador desarrollado en el ejercicio anterior

Ejercicio 2: diccionario

- Aparte de lo ya incluido en el ejercicio anterior, hemos de añadir al diccionario
  - Información sobre pronombres personales

```
% es_pronombre(Pronombre, Persona, Número)

es_pronombre(yo, 1, singular).
es_pronombre(tú, 2, singular).
es_pronombre(ella, 3, singular).
es_pronombre(él, 3, singular).
es_pronombre(nosotros, 1, plural).
es_pronombre(vosotros, 2, plural).
es_pronombre(ellas, 3, plural).
es_pronombre(ellos, 3, plural).
```

Ejercicio 2: DCG

- Aparte de la regla del ejercicio anterior, hemos de añadir más
  - Símbolo inicial de la gramática: reconoce la estructura de frase

```
frase --> pronombre, verbo(Inf, Tpo, Pers, Num).
```

  - Hay que comprobar la concordancia añadiendo parámetros

```
frase -->
        pronombre(Persona, Número),
        verbo(_, _, Persona, Número).
```

  - Y añadir otra regla para reconocer pronombres y obtener su persona y número

```
pronombre(Persona, Número) -->
        [P],
        { es_pronombre(P, Persona, Número) }.
```

Ejercicio 2: uso de la DCG

- Como el símbolo inicial de la gramática es

```
frase
```

y no tiene parámetros, el uso en Prolog es

```
?- frase([tú, hablarás], []).

Yes

?- frase([tú, hablaré], []).

No
```

Ejercicio 3: analizador sintáctico

- **Construir un analizador Prolog basado en una DCG que permita analizar frases como las siguientes**
  - *David habla con Ana*
  - *Julia lee libros en el jardín*
  - *Los niños leen*
  - *Pedro juega*
  - *Elvira juega en la piscina*

Ejercicio 3: analizador sintáctico

- **Cada verbo puede llevar como máximo un complemento circunstancial**
  - Los complementos circunstanciales siempre empiezan por una preposición
  - Para cada verbo se establece una única proposición permitida o ninguna (si el verbo no admite complementos)
  - Aunque el verbo admita un complemento circunstancial puede no llevar ninguno

Ejercicio 3: analizador sintáctico

- **Las frases con verbos transitivos pueden llevar o no complemento directo**
  - Si hay complemento directo, el complemento circunstancial aparecerá después
- **Las frases con verbos intransitivos no pueden tener complemento directo**

Ejercicio 3: analizador sintáctico

- **Se debe garantizar**
  - La concordancia sujeto-verbo
  - La compatibilidad verbo-complemento circunstancial
- **Si el análisis de la frase es correcto, se devolverá el árbol de análisis sintáctico correspondiente**

Ejercicio 3: estructura básica

```
frase       --> g_nominal, g_verbal.

g_nominal   --> nombre.
g_nominal   --> articulo, nombre.
g_nominal   --> nombre_propio.

g_verbal    --> verbo_trans, g_nominal, complemento.
g_verbal    --> verbo_trans, g_nominal.
g_verbal    --> verbo_trans, complemento.
g_verbal    --> verbo_trans.

g_verbal    --> verbo_int, complemento.
g_verbal    --> verbo_int.
```

Estructura de frase

Posibles formas de grupos nominales

Posibles formas de grupos verbales, distinguiendo tipos de verbos

Ejercicio 3: árbol de análisis sintáctico

```
frase(f(GN, GV))  --> g_nominal(GN), g_verbal(GV).

g_nominal(gn(N))          --> nombre(N).
g_nominal(gn(A, N))       --> articulo(A), nombre(N).
g_nominal(gn(N))          --> nombre_propio(N).

g_verbal(gv(V, CD, C))  --> verbo_tr(V),
                            g_nominal(CD), complem(C).
g_verbal(gv(V, CD))       --> verbo_tr(V), g_nominal(CD).
g_verbal(gv(V, C))        --> verbo_tr(V), complem(C).
g_verbal(gv(V))           --> verbo_tr(V).

g_verbal(gv(V, C))        --> verbo_int(V), complem(C).
g_verbal(gv(V))           --> verbo_int(V).
```

Ejercicio 3: concordancia

```
frase(f(GN, GV)) --> g_nominal(GN, Num), g_verbal(GV, Num).
```

concordancia sujeto-verbo

```
g_nominal(gn(N), Num)       --> nombre(N, Num, _).
g_nominal(gn(A, N), Num)    --> articulo(A, Num, Gen),
                                nombre(N, Num, Gen).
g_nominal(gn(N), singular) --> nombre_propio(N).

g_verbal(gv(V, CD, C), Num)--> verbo_tr(V, Num),
                               g_nominal(CD, _), complem(C).
g_verbal(gv(V, CD), Num)    --> verbo_tr(V, Num),
                                g_nominal(CD, _).
g_verbal(gv(V, C), Num)     --> verbo_tr(V, Num), complem(C).
g_verbal(gv(V), Num)        --> verbo_tr(V, Num).



g_verbal(gv(V, C), Num)     --> verbo_int(V, Num), complem(C).
g_verbal(gv(V), Num)        --> verbo_int(V, Num).
```

concordancia artículo-nombre

Ejercicio 3: complementos circunstanciales

```
g_verbal(gv(V, CD, C), N)  --> verbo_trans(V, N, Lprep),
                               g_nominal(CD, _N),
                               complemento(C, Lprep).
g_verbal(gv(V, CD), N)     --> verbo_trans(V, N, _L),
                               g_nominal(CD, _N).
g_verbal(gv(V, C), N)      --> verbo_trans(V, N, Lprep),
                               complemento(C, Lprep).
g_verbal(gv(V), N)         --> verbo_trans(V, N, _L).
g_verbal(gv(V, C), N)      --> verbo_int(V, N, Lprep),
                               complemento(C, Lprep).
g_verbal(gv(V), N)         --> verbo_int(V, N, _L).

complemento(c(P, GN), [Prep]) --> preposicion(P, [Prep]),
                                  g_nominal(GN, _N).
```

¡sólo lista unitaria en este ej.!

```
preposicion(p(P), [P])     --> [P], {es_preposicion(P)}.
```

Ejercicio 3: resto de la DCG

```
nombre_propio(np(P))      --> [P],
                              {es_nombre_propio(P)}.

nombre(n(P), Num, Gen)    --> [P],
                              {es_nombre(P, Num, Gen)}.

articulo(a(P), Num, Gen) --> [P],
                              {es_articulo(P, Num, Gen)}.

verbo_int(v(P), Num, L) --> [P],
                              {es_verbo_intr(P, Num, L)}.

verbo_trans(v(P), Num, L) --> [P],
                              {es_verbo_trans(P, Num, L)}.
```

Ejercicio 3: diccionario

```
es_nombre_propio(ana).
es_nombre_propio(julia).
es_nombre_propio(david).
es_nombre_propio(pedro).
es_nombre_propio(elvira).

es_nombre(libros, plural, masc).
es_nombre(niños, plural, masc).
es_nombre(jardín, singular, masc).
es_nombre(piscina, singular, fem).

es_articulo(el, singular, masc).
es_articulo(la, singular, fem).
es_articulo(los, plural, masc).
```

Ejercicio 3: diccionario

```
es_verbo_intr(habla, singular, [con]).
es_verbo_intr(juega, singular, [en]).

es_verbo_trans(lee, singular, [en]).
es_verbo_trans(leen, plural, [en]).
es_verbo_trans(escriben, plural, []).

es_preposicion(en).
es_preposicion(con).
```

Ejercicio 3: uso de la DCG

- Como el símbolo inicial de la gramática es

```
frase
```

y tiene 1 parámetro, el uso en Prolog es

```
?- frase(Árbol, [david, habla, con, ana], []).
Árbol = frase(gn(np(david)),
              gv(v(habla), c(p(con), gn(np(ana)))))

?- frase(Árbol, [los, niños, leen], []).
Árbol = frase(gn(a(los), n(niños)), gv(v(leen)))

?- frase(Árbol, [pedro, juega], []).
Árbol = frase(gn(np(pedro)), gv(v(juega)))

?- frase(A, [elvira, juega, en, la, piscina], []).
A = frase(gn(np(elvira)),
   gv(v(juega), c(p(en), gn(a(la), n(piscina)))))
```

Ejercicio 4: analizador y traductor a voz pasiva

- **Construir una DCG que permita analizar y pasar a voz pasiva frases como las siguientes**
  - *El niño dibujó una flor en el cuaderno*
    - *Una flor fue dibujada por el niño en el cuaderno*
  - *Yo tomé la decisión*
    - *La decisión fue tomada por mí*
- **Estructura de frase: sujeto, verbo, complemento directo**
  - Puede haber o no complemento circunstancial
    - Si lo hay, va al final

Ejercicio 4: analizador y traductor a voz pasiva

- **El analizador determinará la corrección de la frase de entrada, comprobando las concordancias sintácticas**
  - Aquí no se pide el árbol
- **Y, en caso de ser correcta, producirá como salida la misma frase pero en voz pasiva**

Ejercicio 4: analizador y traductor a voz pasiva

- **Complementos circunstanciales**
  - Máximo uno por verbo
    - Puede no haber ninguno (aunque el verbo lo admita)
  - Cada verbo puede tener una única preposición permitida, todas o ninguna (si no admite complementos circunstanciales)
  - Usaremos una aproximación alternativa a la del anterior ejercicio corregido

Ejercicio 4: diccionario

```
es_articulo(el, singular, masc).
es_articulo(la, singular, fem).
es_articulo(los, plural, masc).
es_articulo(las, plural, fem).
es_articulo(uno, singular, masc).
es_articulo(una, singular, fem).
es_articulo(unos, plural, masc).
es_articulo(unas, plural, fem).

es_nombre(niños, plural, masc).
es_nombre(cuaderno, singular, masc).
es_nombre(decisión, singular, fem).
es_nombre(niño, singular, masc).
es_nombre(flor, singular, fem).

es_preposicion(en).
es_preposicion(con).
```

Como en el 3:

- artículo y nombre con información para concordancia
- + preposiciones

Ejercicio 4: diccionario

```
es_terminacion(o, presente, 1, singular).
es_terminacion(as, presente, 2, singular).
es_terminacion(a, presente, 3, singular).
es_terminacion(amos, presente, 1, plural).
es_terminacion(ais, presente, 2, plural).
es_terminacion(an, presente, 3, plural).

es_terminacion(é, pasado, 1, singular).
es_terminacion(aste, pasado, 2, singular).
es_terminacion(ó, pasado, 3, singular).
es_terminacion(amos, pasado, 1, plural).
es_terminacion(asteis, pasado, 2, plural).
es_terminacion(aron, pasado, 3, plural).
```

Como en 1 y 2:
terminaciones de los tiempos verbales

Ejercicio 4: diccionario

```
es_pronombre(yo, 1, singular, mí).
es_pronombre(tú, 2, singular, ti).
es_pronombre(él, 3, singular, él).
es_pronombre(ella, 3, singular, ella).
es_pronombre(nosotros, 1, plural, nosotros).
es_pronombre(nosotras, 1, plural, nosotras).
es_pronombre(vosotros, 2, plural, vosotros).
es_pronombre(vosotras, 2, plural, vosotras).
es_pronombre(ellos, 3, plural, ellos).
es_pronombre(ellas, 3, plural, ellas).
```

Identificación de pronombres:

junto con información de persona y número, así como la traducción para la construcción de la forma pasiva

Ejercicio 4: diccionario

```
participio(ada, fem, singular).
participio(ado, masc, singular).
participio(adas, fem, plural).
participio(ados, masc, plural).

verbo_ser(pasado, singular, fue).
verbo_ser(presente, singular, es).
verbo_ser(pasado, plural, fueron).
verbo_ser(presente, plural, son).

es_raiz(pint, [en]).
es_raiz(habl, [con]).
es_raiz(am, []).          % Ninguna prep. permitida
es_raiz(dibuj, [_P]).     % Cualquier prep. permitida
```

Para la construcción de la forma pasiva

Raíces + preposiciones

Ejercicio 4: DCG *(estructura de frase y resto)*

```
frase(Salida) -->
        gn(SUJ, _G, Pers, Num),
        % repres. de sujeto + info concordancia
        verbo(R, Tpo, Pers, Num, LPrep),
        % repres. de raíz + concordancia
        % + info tiempo verbal y preposiciones
        gn(CD, G, _P, N),
        % repres. de complemento directo
        % + info concordancia para paso a pasiva
        complemento(C, LPrep),
        % repres. de complemento circunstancial
        % (si hay) + preposición
        { componer(Salida, SUJ, R, Tpo, CD, G, N, C) }.
        % generación de la traducción a forma pasiva
```

Ejercicio 4: DCG

```
gn([A, N], Genero, 3, Numero) -->
            [A, N],
            { es_articulo(A, Numero, Genero),
              es_nombre(N, Numero, Genero) }.

gn([P], _, Persona, Numero) -->
            [Pal],
            { es_pronombre(Pal, Persona, Numero, P) }.

complemento([P|GN], [P]) -->
            [P], { es_preposicion(P) },
            gn(GN, _, _, _).

complemento([], _) --> [].    % Si no hay CC
```

Ejercicio 4: DCG

```
verbo(Rs, Tiempo, Persona, Numero, Lprep) -->
        [V],
        { name(V, Vs),
          % de átomo a lista de caracteres
          append(Rs, Ts, Vs),
          % descomposición en raíz y terminación
          name(Raiz, Rs),
          % de lista de caracteres a átomo
          es_raiz(Raiz, Lprep),
          % raíz y preposición en diccionario
          name(Termin, Ts),
          % de lista de caracteres a átomo
          es_terminacion(Termin, Tiempo, Persona, Numero)
        }.
```

Ejercicio 4: traducción a forma pasiva

```
componer(Salida, GN, Rs, Tiempo, CD, G, N, CC) :-
        % verbo a forma pasiva
        verbo_ser(Tiempo, N, Tiempo_ser),
        participio(Terminacion, G, N),
        name(Terminacion, Ts),
        append(Rs, Ts, Parts),
        name(Part, Parts),
        % construcción de la frase traducida
        append(CD, [Tiempo_ser, Part, por], Aux1),
        append(GN, CC, Aux2),
        append(Aux1, Aux2, Salida).
```

Ejercicio 4: ejemplos de uso

```
?- frase(Salida, [el, niño, dibujó, una, flor], []).

Salida = [una, flor, fue, dibujada, por, el, niño]

?- frase(Salida, [el, niño, dibujó, una, flor, en, el,
   cuaderno], []).

Salida = [una, flor, fue, dibujada, por, el, niño, en,
   el, cuaderno]

?- frase(Salida, [yo, tomé, la, decisión], []).

Salida = [la, decisión, fue, tomada, por, mí]
```

Ejercicio 5: interfaz en LN a BD

- **Se tiene una BD de empleados implementada con**

  empleado(Nombre, Atributo, Valor)

- **Construir una interfaz, utilizando una gramática semántica, que analice y responda a consultas como**
  - *¿Cuál es el salario de García?*
  - *Dime el departamento de Martínez*
  - *¿Qué edad tiene Martínez?*

Ejercicio 5: ciclo pregunta-respuesta

```
consulta :- write('Pregunta:        '), read(Pregunta),
            procesa(Pregunta).

procesa(Xs) :- pregunta1(Nombre, Atributo, Xs, []),
               empleado(Nombre, Atributo, Valor),
               write('  Respuesta:     '),
               write(Valor), nl, consulta.
```

Ejercicio 5: BD ejemplo

```
% empleado(Nombre, Atributo, Valor)

empleado(martínez, departamento, ventas).
empleado(martínez, salario, 1800).
empleado(martínez, edad, 30).

empleado(gutiérrez, departamento, compras).
empleado(gutiérrez, salario, 2130).
empleado(gutiérrez, factoría, madrid).

empleado(garcía, departamento, ventas).
empleado(garcía, salario, 1450).
empleado(garcía, factoría, madrid).
...
```

Ejercicio 5: análisis de consultas (G semántica)

```
pregunta1(Nombre, Atributo) -->
                    comienzo1,
                    ([el] ; [la]),
                    atributo(Atributo),
                    [de],
                    nombre(Nombre).

pregunta1(Nombre, Atributo) -->
                    [qué],
                    atributo(Atributo),
                    [tiene],
                    nombre(Nombre).

comienzo1 -->   [cuál, es] | [dime]
              | [dame] | [quiero, saber].
```

Ejercicio 5: análisis de consultas (G semántica)

```
atributo(Atributo) --> [Atributo],
                       { es_atributo(Atributo) }.

nombre(Nombre) --> [Nombre],
                   { es_nombre(Nombre) }.

valor(Valor) --> [Valor],
                 { es_valor(Valor) }.


% Cláusulas Prolog: acceso a campos
es_nombre(Nombre) :- empleado(Nombre, _, _), !.
es_atributo(Atributo) :- empleado(_, Atributo, _), !.
es_valor(Valor) :- empleado(_, _, Valor), !.
```

Ejercicio 5: ejemplos de uso

```
?- consulta.

Pregunta:       [qué, salario, tiene, martínez].
Respuesta:      1800


Pregunta:       [].

No
```