Etapas de compilación

Como *usuarios del compilador* GNU Pascal, cada vez que lancemos una compilación se inicia un proceso de varias etapas (sólo una es la compilación) y, mediante directivas, podemos detenerlo en cualquier momento.

(ejercicio 01Algor\programas\etapas.pas)

1. *Preprocesado* . Son operaciones de inclusión, búsqueda y sustitución de textos, borrado de otros... Son como las realizadas por un procesador de textos. (directiva -E)
   gpc --extended-pascal -E etapas.pas
2. *Compilación* a lenguaje *ensamblador* . Da un archivo en ensamblador con extensión s. (Usa la directiva -S).
   gpc --extended-pascal -S etapas.pas
3. *Compilación* a código *máquina* . Genera código máquina no ejecutable que pone en un archivo con extensión o. (directiva -c).
   gpc --extended-pascal -c etapas.pas
4. *Enlace* con las bibliotecas disponibles en código gmáquina para generar el ejecutable.

NOTA. Es posible que algunos compiladores no incluyan todas las etapas, tengan alguna más o resuman varias etapas en una sola.

NOTA. La compilación a ensamblador en lugar de ir directamente a código máquina permite enlazar código compilado cuyo fuente fue escrito en distintos lenguajes: C, Ada, Pascal...

Por tanto la *orden de compilación habitual*

```
      directiva       entrada       salida
gpc --extended-pascal etapas.pas      {a.exe}
```

equivale a esta secuencia de órdenes

```
   +---- directivas ----+  entrada      salida
gpc --extended-pascal -E   etapas.pas > etapas.i    {preprocesado}
gpc --extended-pascal -S   etapas.pas  {etapas.s}   {ensamblador }
gpc --extended-pascal -c   etapas.s    {etapas.o}   {objeto      }
gpc --extended-pascal      etapas.o    {a.exe   }   {ejecutable  }
```

También se puede lanzar esta orden y se obtendrán los archivos anteriores

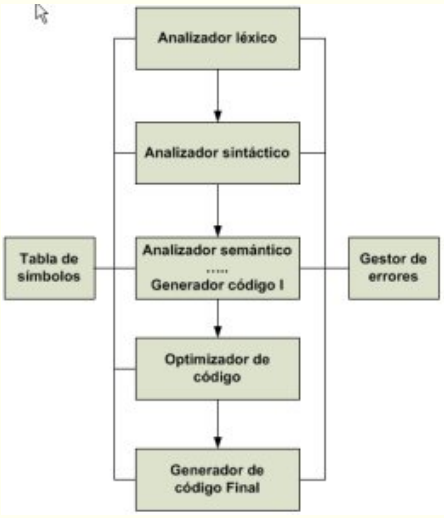
```
gpc --extended-pascal -save-temps etapas.pas
```

La directiva *--extended-pascal* le pide a gpc que compile según Pascal ISO 10206.

Etapa de compilación: subetapas

Vamos a hablar de la *etapa de compilación* de la figura anterior.

El constructor de compiladores sabe que un compilador suele tener las siguientes etapas :



- *Análisis léxico* . Consiste en agrupar caracteres para formar las unidades que ya tienen significado por si mismas (tokens).
  NOTA. Token es la moneda de menor valor. Nosotros diríamos céntimo.
- *Análisis sintáctico* . Se analizan las sentencias, que están compuestas de tokens, de acuerdo con la sintaxis del lenguaje.
- *Análisis semántico* y *Generador de código intermedio* son dos etapas que se representan juntas porque la generación de código intermedio utiliza el mismo árbol del análisis semántico.
- *Optimización del código* . Es una sección optativa pero muy difundida y demandada entre los compiladores profesionales. Por ello, Los constructores de compiladores gastan mucho dinero en ella. Trata de optimizar el código en tamaño y/o velocidad.
- *Generador de código final* . Se encarga de generar el código en el lenguaje objeto del compilador.
  NOTA. Los objetos del programa pueden ser simples (enteros, reales) o complejos (matrices, registros, funciones, procedimientos). Tienen un identificador asociado, una posición en memoria, un valor y unos atributos. El valor es el almacenado en la zona de memoria que ocupa el objeto y los atributos determinan los valores que puede tomar el objeto, las operaciones a las que se puede someter...

Las *tablas de símbolos* tienen al menos la siguiente información respecto de cada objeto del programa.

Su nombre o *denotador,*

Los *atributos* (el más importante es el tipo),

Su *posición en memoria* o la forma de encontrarla...

El *gestor de errores* trata de optimizar las compilaciones de un programa. El gestor

Divide el código por *tramos* delimitados por marcas de bloque (*begin* y *end* en Pascal).

Si en un *tramo* de código encuentra *demasiados errores* (entra en *modo panic*) lo ignora y pasa a compilar el siguiente tramo. En consecuencia, cuando el compilador nos indica los errores que ve tenemos que ser conscientes de que son consecuencia de las partes compiladas y de las no compiladas.

Modo de corregir los errores

¡ *No* hay que *asustarse* !

- Seguramente no aparecen todos los *errores que hay*
- Es posible que aparezcan *errores que no existen* (el compilador está bien)

Corregir los errores de menor a mayor número de línea . Esto puede

- *Hacer que desaparezcan* errores que 'veía' el compilador en líneas posteriores
- *Hacer que aparezcan otros* que antes no había considerado porque entró en el modo panic

Pregunta de examen

Cuando se lanza la compilación en pascal con gpc, ¿cuantas compilaciones se realizan en realidad?

A. 1
B. 2
C. 3