



Tecnológico de Monterrey - Campus Monterrey

Uso de AI para el desarrollo de Babyduck

Desarrollo de aplicaciones avanzadas de ciencias computacionales

Profa. Elda Guadalupe Quiroga González

Daniela Ramos García A01174259

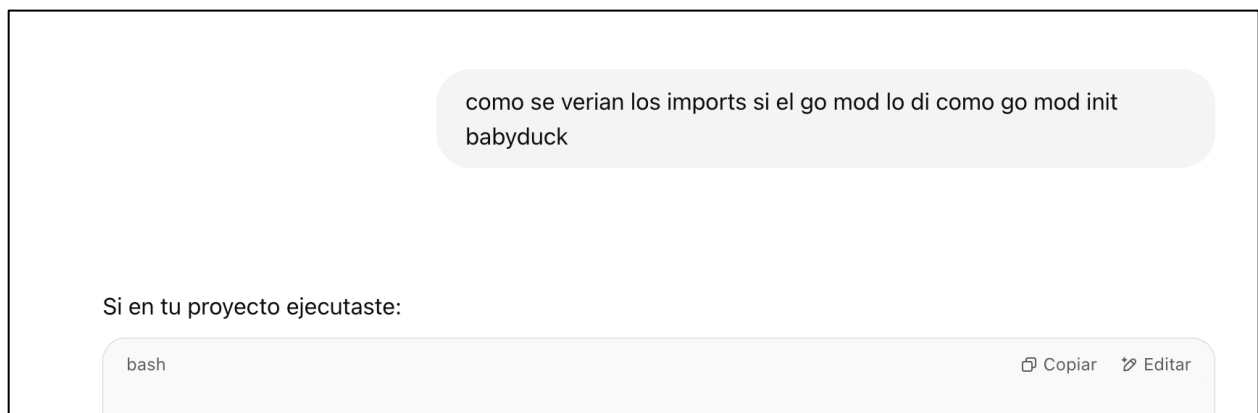
4 de junio del 2025

Uso de IA.

Durante el desarrollo de BabyDuck he recurrido a la inteligencia artificial como una herramienta en diversas etapas clave del proyecto. El servicio que he utilizado para desarrollar mi proyecto del compilador para el lenguaje Babyduck fue ChatGPT de OpenAI y algunas ocasiones que Microsoft Copilot hacia sugerencias para completar código repetitivo que estaba a punto de implementar. A continuación, listaré por sección de mi entregable final, la ayuda que recibí de dicha herramienta y fotos puntuales de los prompts.

Consideraciones de léxico y sintaxis.

En la fase inicial, me ayudó a estructurar mi directorio de carpetas y modulos en go por que no estaba familiarizada en como funcionan los modulos y paquetes, ademas me ayudó con el syntax de las reglas en el BNF que eran necesarias para que Gocc generara correctamente el scanner y el parser ya que no estaba 100% familiarizada con el syntax de Gocc: me indicó cómo declarar tokens, cómo encadenar las producciones de la gramática y cómo encapsular la lógica de análisis semántico en las acciones incrustadas de la BNF, de esta forma pude implementar todos esos elementos acorde a los requerimientos de mi lenguaje de manera independiente ya con el conocimiento de la sintaxis exigida en el BNF. Ej.



por que me da este error en el archivo productionstable.go que genera gocc. package babyduck/data_structures is not in std (/usr/local/go/src/babyduck/data_structures) (compile)go-staticcheck

Pensó durante 12 segundos >

Ese error no viene de Gocc, sino de Go Staticcheck al ver un import que no contiene ningún punto y que no es parte de la librería estándar. Cuando pones

go

Copiar Editar

```
import "babyduck/data_structures"
```

Go y Staticcheck buscan en GOROOT (la std lib) una carpeta babyduck/data_structures y, al no existir, se quejan de que "no está en std"

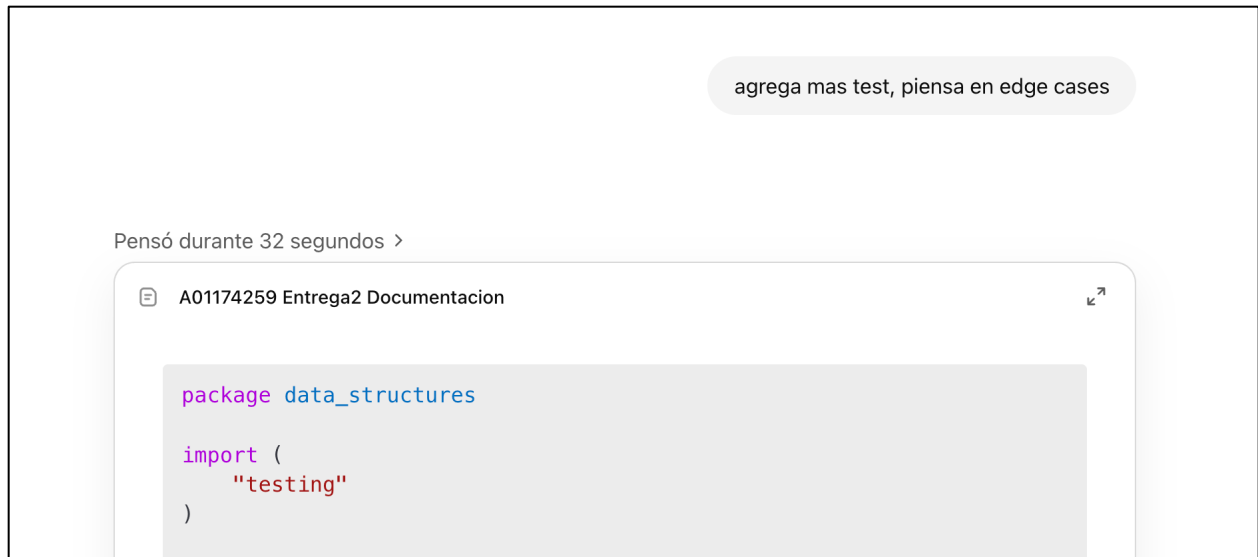
Análisis semántico.

Cuando empezó la parte de análisis semántico, ChatGPT me corrigió errores que había realizado en la declaración de mi cubo semántico y también me ayudó con la sintaxis de Go para escribir ciclos y realizar operaciones como agregar, buscar y eliminar objetos dentro de arreglos (no necesitaba que me los hiciera todos pero si quería que me dijera como eran las reglas del lenguaje), esto me ayudó para poder realizar los métodos del directorio de funciones y tabla de variables. Además que me ayudó a generar tests que le indicaba e incluso edge cases que no había considerado.

divide el siguiente archivo de prueba de sintaxis para que sean test cases individuales para que pueda realizar un mejor debug de los test cases. package babyduck

```
import (  
    "babyduck/lexer"  
    "babyduck/parser"  
    "testing"  
)
```

```
// testData maps input programs to a boolean indicating whether  
parsing should succeed.  
var testData = map[string]bool{  
    // Test case 1: testing empty program  
    "program p; main { } end": true,  
    // Test case 2: testing variable declaration and assignment
```



Generación de código intermedio.

Más adelante, al implementar la generación de cuádruplos, ChatGPT me indicó cuales eran mis errores en la implementación de las pilas para operadores, operandos y tipos dentro del contexto cuando realizaba la integración de las partes anteriores. Me orientó para escribir los helpers `HandleOperand` y `HandleBinary`. Asimismo, redactó ejemplos de tests para verificar la corrección tanto del cubo semántico como de la secuencia de cuádruplos generados en casos sencillos y complejos. Cuando surgían errores de “unreachable code” o de conversiones de tipos, ChatGPT explicó por qué se producían.

en vez de agregarlo al bnf, voy a meter las funciones auxiliares a las funciones del contexto que ya existen en esos mismos puntos, ayudame, me marca error en la llamada al HandleOperand. // ResolveVarType consulta el tipo de la variable identificada por name.

```
func (ctx *Context) ResolveVarType(name string) (interface{}, error) {
    vt := ctx.CurrentVarTable()
    entry, ok := vt.GetByName(name)
    if !ok {
        return nil, fmt.Errorf("variable %q no declarada", name)
    }
    HandleOperand(entry.Name, entry.Type)
    return entry.Type, nil
}
```

undefined: HandleOperandcompilerUndeclaredName



Tienes que invocar tu helper como método del contexto (`ctx.HandleOperand`), no como función global.
Por ejemplo:



dame un archivo de pruebas donde pueda ver los pasos de esta entrega de las pilas y la pila de cuádruplos e imprima la pila

Pensó durante 21 segundos >

```
go

package data_structures

import "testing"
```



Copiar Editar

También produjo tablas de casos de prueba en formato CSV y guías de uso de los tests para asegurar la cobertura semántica y sintáctica. Gracias a ese apoyo, pude enfocarme en la arquitectura del compilador y dejar que la ChatGPT creará elementos de documentación que no requerían lógica en su producción como enlistar test cases.

dame un archivo de pruebas donde pueda ver los pasos de esta entrega de las pilas y la pila de cuádruplos e imprima la pila

Pensó durante 21 segundos >

go

 Copiar  Editar

```
package data_structures
```

```
import "testing"
```

ahora hazme una tabla con los 11 test cases nuevos que siga el formato de ID Descripción Entrada Resultado Esperado, damela en formato cvs

Pensó durante 18 segundos >

csv

 Copiar  Editar

Funciones.

Para la parte de funciones, ChatGPT me auxilió en poder generar los logs que me estaban fallando con razones de unreachable code que no me había dado cuenta, al igual me ayudó a generar test cases para verificar que estuviera correcta mi producción de cuádruplos.

```
por que no veo los logs? danielaramosgarcia@Danielas-MacBook-
Pro-2 babyduck % go test -v -run TestFunctionDeclaration
=== RUN TestFunctionDeclaration
    babyduck_test.go:139: Function declaration should parse, got:
1:48: error: variable a no declarada
--- FAIL: TestFunctionDeclaration (0.00s)
FAIL
exit status 1
FAIL babyduck 0.283s
```

Pensó durante 27 segundos >

El motivo por el que no estás viendo ni registrando nunca el parámetro `a` es que, con la gramática tal y como la tenías, la llamada a `ctx.RegisterFunction` (y cualquier `fmt.Printf` que pusieras dentro) se ejecuta sólo al final de la producción de `Func`, después de haber hecho todo el `Vars Body`. Eso

Máquina Virtual.

Para la parte de máquina virtual, ChatGPT me ayudó en una parte del desarrollo en la que mi código de ejecución, en específico con el switch donde manejaba el instructor pointer, no estaba manejando de manera correcta el manejo de memoria debido a que no me había dado cuenta que en los cuádruplos de PARAM, mi ejecución no estaba almacenando los valores de los argumentos para después asignarlos a los parámetros de la función a la que se iba a mover posteriormente el instructor pointer y por lo tanto, no asignaba el valor a los argumentos que estaban en la tabla de variables locales de la función, por lo que todas las operaciones dentro de las funciones no estaban siendo ejecutadas de manera correcta.

IDE.

Para realizar el IDE de mi compilador, ChatGPT me indico los elementos y sintaxis pertinentes para construir una UI sencilla usando Fyne, con esto pude construir la UI de manera rápida y personalizarla a mi gusto. De esta manera pude exponer mi trabajo de una forma más agradable visualmente.