INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

**PRACTICA 4**

“MAQUINA VIRTUAL DE PILA”

ALUMNO:

LOPEZ PEREZ ALBERTO ANDREI

PROFESOR: TECLA PARRA ROBERTO

GRUPO: 5CM4

MATERIA: COMPILADORES

# Introducción

En esta práctica se implementaron los conceptos base de compiladores y la implementación de YACC, donde se da continuidad a la práctica número 3, la cual fue una una "Calculadora de Vectores con Tabla de Símbolos". Lo que nos atañe realizar en esta práctica es:

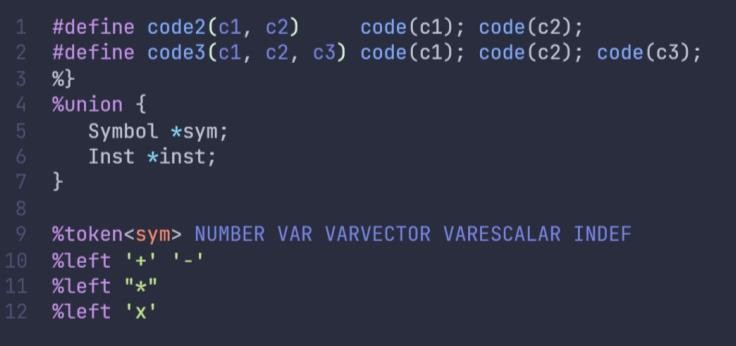
Agregar Máquina Virtual de Pila

De modo que, la calculadora bajo la perspectiva de la ejecución no cambiará con respecto a la anterior práctica. Sin embargo, lo que sí cambia es la implementación, ya que ahora no usaremos solamente la pila de YACC para generar la funcionalidad, si no nos apoyaremos de una *máquina virtual* que generar código de ciertas instrucciones y valores que se guardaran en una RAM virtual para que con ayuda de una pila después se ejecuten dando así la misma funcionalidad que teníamos antes.

Para poder realizar lo anterior se nos proporciona el código del **HOC4.y** visto en clase donde se implementa precisamente una *máquina virtual de pila* en una calculadora normal*,* por lo que con ayuda de ese nos apoyamos para construir nuestra propia *máquina virtual de pila* acomodada al contexto de la "Calculadora de Vectores".

# Desarrollo

Lo primero que modificamos de nuestra anterior implementación fue la sección de definiciones, ya que cambiamos ahora los tipos de los elementos de la pila de YACC a Symbol\* e Inst\* los cuales son para los símbolos normales e instrucciones de nuestra máquina virtual. Además agregamos unas ciertas macros que nos ayudaran a mandar a llamar la función code() de **code.c** la cual nos ayuda a instalar instrucciones y/o operandos en la RAM de nuestra máquina virtual.



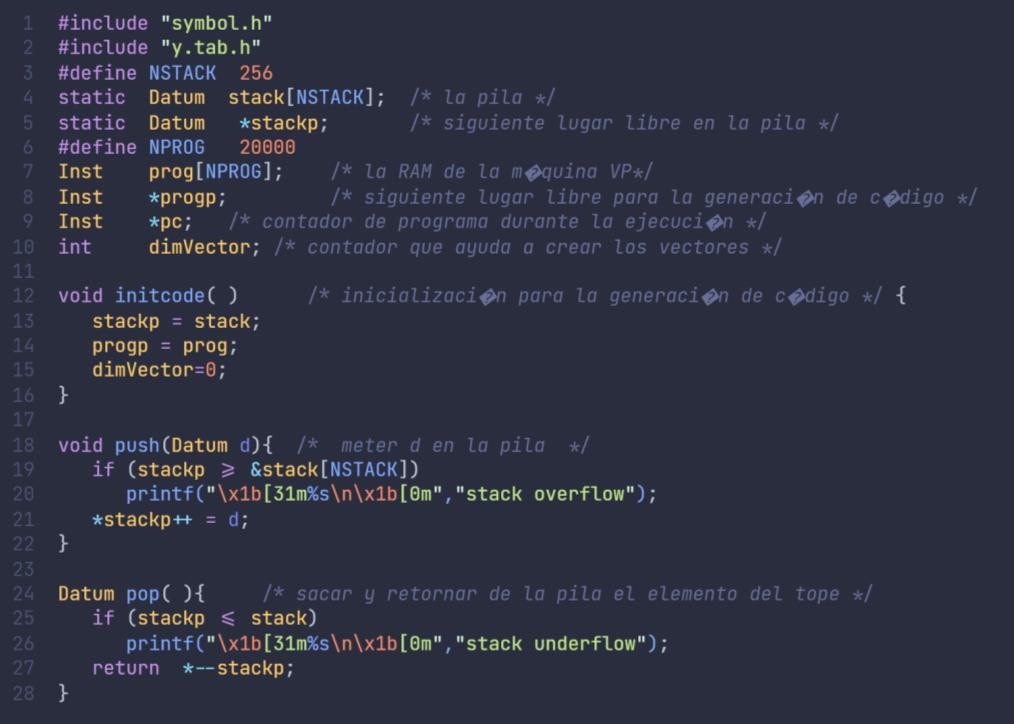
Después, algo que cambiamos fueron las acciones gramaticales de nuestra sección de reglas de YACC, donde ya no hacemos uso de la pila de YACC para ir generando la funcionalidad, si no solo usamos las acciones para poder generar el código de la RAM.



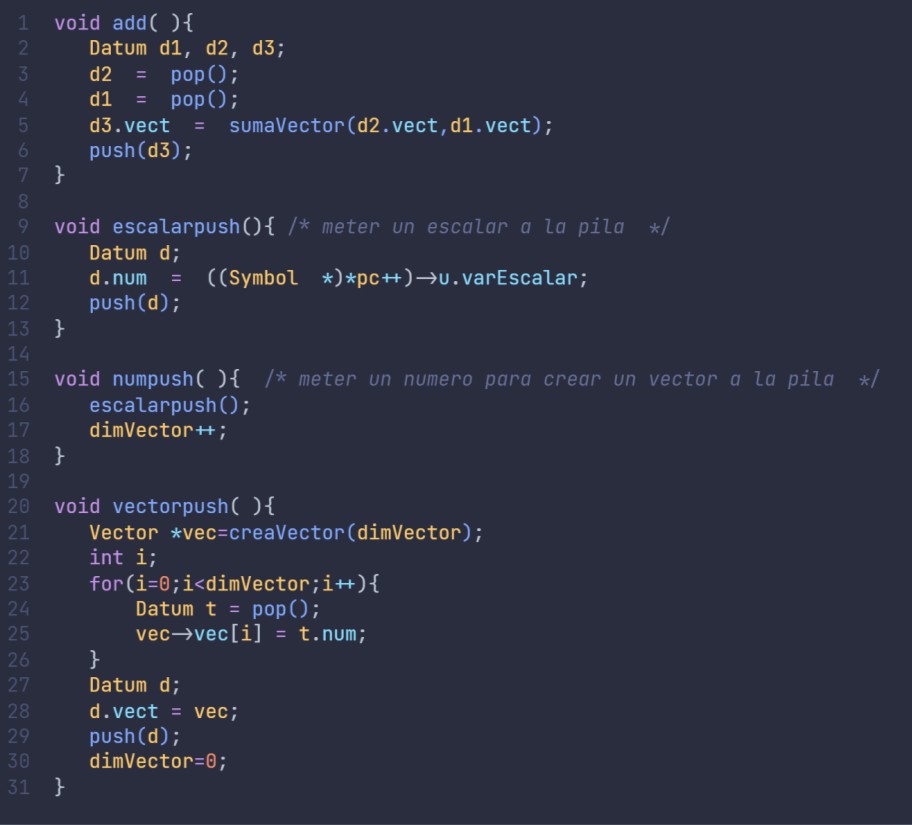
Cada una de las instrucciones instaladas en la RAM de la máquina virtual son funciones que se implementaron en **code.c**

ℹ Hacemos énfasis en que en la RAM instalamos instrucciones, nombres de variables o números, donde los números pueden ser **elementos de un vector** para crearlo

o simples **valores escalares**.

Dando un vistazo al archivo **code.c**, vemos que aquí se implementa la *máquina virtual de pila* donde se crea su RAM, su pila y se implementan sus funciones que la manipulan, como initcode() para inicializarla, push() y pop() para manipular la Pila, o como execute() para empezar a leer las instrucciones de la RAM y ejecutarlas con ayuda del contador de programa.

Y también aquí se implementan todas las instrucciones que hacen referencia a todas las operaciones que se realizan en la calculadora, como lo es crear un vector, sumar vectores, imprimir un vector o un escalar, crear una variable, meter un número o un vector en la pila, etc.



Normalmente todas las funciones que conciernen a una operación de la calculadora lo que realizan es sacar los operandos de la pila que fueron previamente insertados por instrucciones anteriores y los manipulan generando la funcionalidad requerida, donde si se arroja un resultado (como la suma, resta, etc) se inserta el resultado en la pila para que este sea manipulado después por otra instrucción.

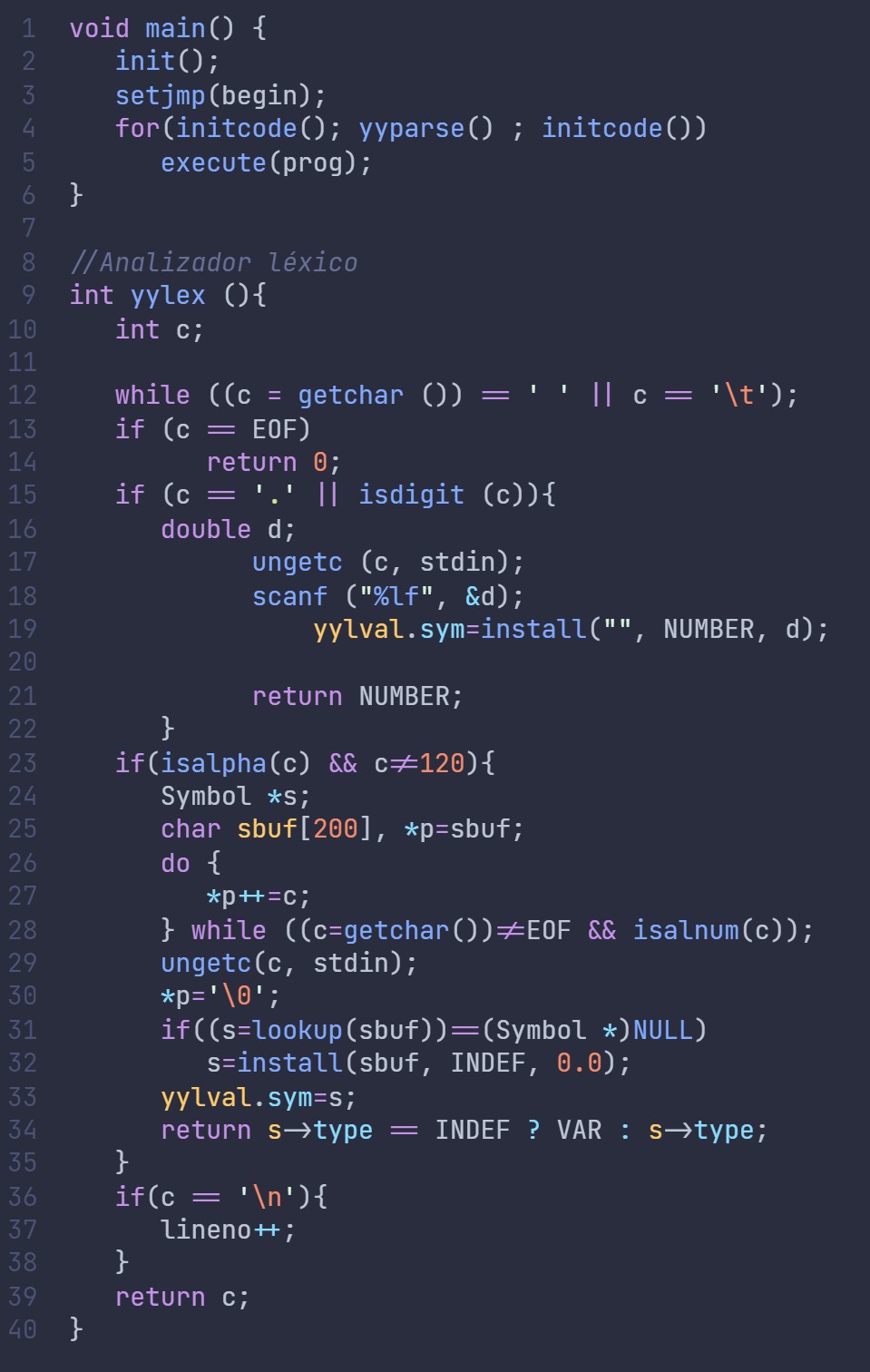
ℹ Como podemos observar, vectorpush() es la instrucción para crear un vector, donde previamente se debieron haber puesto los elementos del vector en la pila con numpush() donde nos ayudamos de una variable global de la máquina llamada dimVector para saber cuantos elementos tendrá el vector y así crearlo. Por lo que nos daremos cuenta que en esta práctica **ya no usamos nuestra propia**

**pila de Pila.c** para crear los vectores.

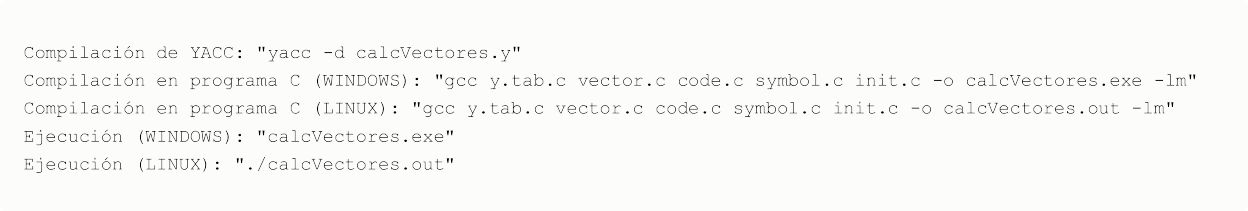
También algo importante a mencionar es que en **symbol.h** es donde definimos el tipo de elementos de la pila de la *máquina virtual* y hacemos posible que se pueda acceder a prog y a todas nuestras instrucciones desde otro archivo anteponiendo la palabra extern, todo esto con el fin de que puedan ser utilizadas en las acciones gramaticales cuando las instalamos con code().



Finalmente también modificamos nuestro main para que este ejecutando las instrucciones de la RAM con la función execute(prog) y también modificamos un tanto nuestro analizador léxico para que siga funcionando con la modificación de nuestra máquina de pila.



## Compilación y Ejecución

Para poder compilar y ejecutar:

## Programa en Ejecución



Ejecución de diversos ejemplos de operaciones, variables y errores posibles.

# Conclusión

Este nuevo método de implementar la calculadora de vectores puede que llegue a tornarse un poco complicado si no se logra abstraer el cómo funciona una máquina física, ya que añade todos los conceptos del funcionamiento para poder generar la solución solicitada. Puede que de momento se vea que es más sencillo implementar una calculadora como la teníamos anteriormente (solo usando la pila de YACC y sus acciones), sin embargo, esta nueva forma nos da la facilidad de poder escalar la calculadora con nuevas funcionalidades, ejecuciones, nuevos tipos y muchas cosas más generando simplemente nuevas reglas y nuevas instrucciones en code, por lo que iniciar la calculadora de esta manera es muy buena práctica si continuamente le estaremos agregando diversas cosas. Además, por lo que vemos nos resolvió el problema de crear nuestro vector sin usar una pila externa y ahora todo lo tenemos más englobado a una máquina.