



**Instituto Politécnico Nacional
Escuela Superior de Cómputo**

Compiladores

**Práctica #3
“HOC 3”**

3CM7

Alumno: Zepeda Flores Alejandro de Jesús

Profesor: Tecla Parra Roberto

INTRODUCCIÓN

HOC es un acrónimo para **High Order Calculator**, es un lenguaje de programación interpretado que fue usado en 1984 en el libro *“El Entorno de Programación de UNIX”* para demostrar como construir interpretes usando una herramienta llamada YACC y lenguaje C.

HOC fue desarrollado por Brian Kernighan y Rob Pike como una grandiosa calculadora interactiva. Su función básica es evaluar expresiones numéricas de puntos flotantes e.g. `“1+2*sin(0.7)”`. Después variables fueron agregadas, expresiones condicionales, ciclos, funciones definidas por el usuario, simple entrada/salida y más, todo esto usando una sintaxis parecida a lenguaje C.

Hasta ahora, las 6 etapas de HOC son:

- HOC1: Calculadora Básica
- HOC2: Calculadora con 26 variables
- **HOC3: Calculadora Científica**
- HOC4: Máquina Virtual de Pila
- HOC5: Ciclos / Decisiones
- HOC6: Funciones y Procedimientos

OBJETIVO

Usando el HOC1 proporcionado por el profesor, elegir un tipo de calculadora y modificarlo para que pueda hacer lo de una calculadora científica como en HOC3 pero con el tipo de datos que se eligió.

Para esta práctica (y las que siguen) se eligió **números complejos**.

DESARROLLO

Lo primero que hice fue tomar el HOC3 del profesor y modificar la sección de reglas, para que pudiera crear variables de más de 2 letras y además funciones built-in:

```
%%  
  
list:  
  | list '\n'  
  | list asgn '\n'  
  | list expr '\n'  
{imprimirC($2);} ;  
  
asgn:  
  VAR '=' expr      { $$ = $1->u.val = $3; $1->type = VAR; }  
  ;
```

```

complejo:
    NUMBER '+' NUMBER 'i'    {$$ = creaComplejo($1, $3);}
| NUMBER '+' 'i'             {$$ = creaComplejo($1, 1);}
| NUMBER '-' 'i'             {$$ = creaComplejo($1, -1);}
| NUMBER '-' NUMBER 'i'     {$$ = creaComplejo($1, -$3);}

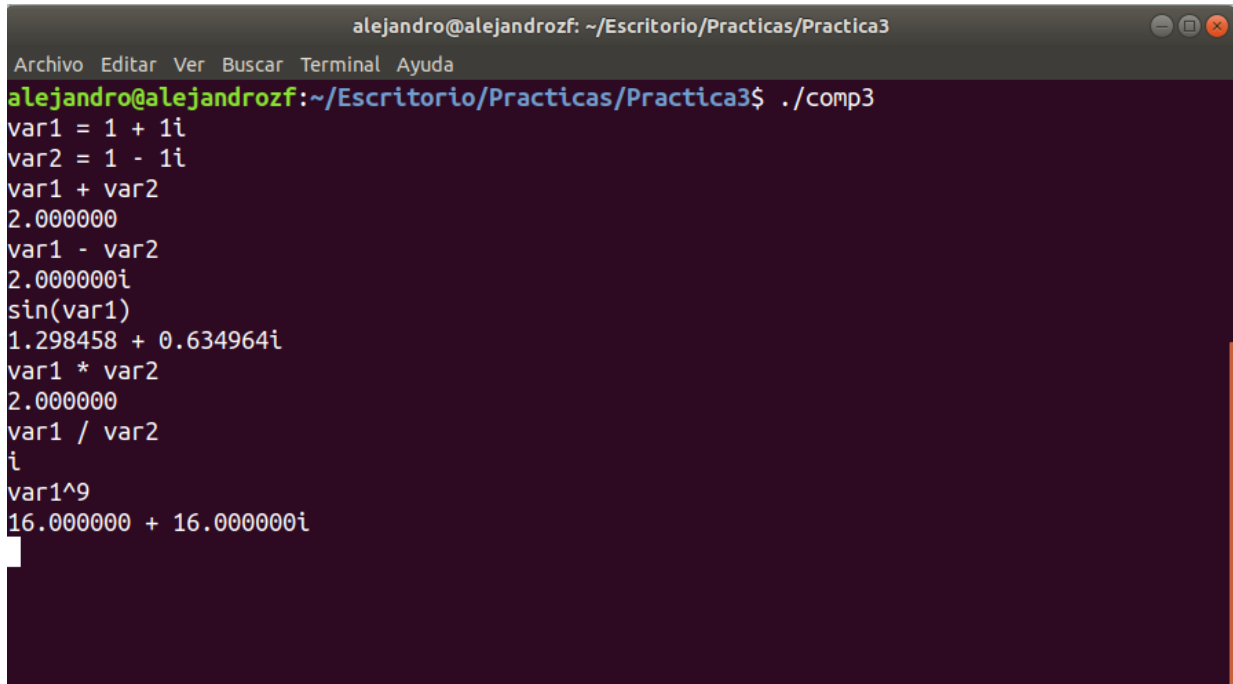
    | '-' NUMBER %prec USIGNE '+' NUMBER %prec USIGNE 'i'    {$$ =
creaComplejo(-$2, $4);}
    | '-' NUMBER %prec USIGNE '-' NUMBER %prec USIGNE 'i'    {$$ =
creaComplejo(-$2, -$4);}
    | '-' NUMBER %prec USIGNE '+' %prec USIGNE 'i'            {$$ =
creaComplejo(-$2, 1);}
    | '-' NUMBER %prec USIGNE '-' %prec USIGNE 'i'            {$$ =
creaComplejo(-$2, -1);}
;

expr:
    complejo{$$ = $1;}
| VAR
    {
        if($1->type == INDEF)
            execerror("variable no definida", $1->name);
        $$ = $1->u.val;
    }
| BLTIN '(' expr ')'        {$$ = (*($1->u.ptr)) ($3);}
| expr '+' expr             {$$ = Complejo_add($1,$3);}
| expr '-' expr             {$$ = Complejo_sub($1,$3);}
| expr '*' expr             {$$ = Complejo_mul($1,$3);}
| expr '/' expr
    {
        if($3->real == 0.0 && $3->img == 0.0)
            execerror("division por cero", "");
        $$ = Complejo_div($1,$3);
    }
| expr '^' NUMBER           {$$ = Complejo_pot($1,$3);}
| '(' expr ')'              {$$ = $2;}
;
%%

```

En el hoc.h se modificó la estructura, en el init.c los built-in serán los que escribí y se agregaron esas funciones.

Pruebas de funcionamiento:

A terminal window titled 'alejandro@alejandrozf: ~/Escritorio/Practicas/Practica3'. The window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The prompt is 'alejandro@alejandrozf:~/Escritorio/Practicas/Practica3\$'. The user has entered './comp3'. The output shows several complex number operations: 'var1 = 1 + 1i', 'var2 = 1 - 1i', 'var1 + var2' resulting in '2.000000', 'var1 - var2' resulting in '2.000000i', 'sin(var1)' resulting in '1.298458 + 0.634964i', 'var1 * var2' resulting in '2.000000', 'var1 / var2' resulting in 'i', and 'var1^9' resulting in '16.000000 + 16.000000i'.

```
alejandro@alejandrozf: ~/Escritorio/Practicas/Practica3
Archivo Editar Ver Buscar Terminal Ayuda
alejandro@alejandrozf:~/Escritorio/Practicas/Practica3$ ./comp3
var1 = 1 + 1i
var2 = 1 - 1i
var1 + var2
2.000000
var1 - var2
2.000000i
sin(var1)
1.298458 + 0.634964i
var1 * var2
2.000000
var1 / var2
i
var1^9
16.000000 + 16.000000i
```

CONCLUSIONES

Ahora podemos que tanto potencial tiene la creación de intérpretes de esta forma, aunque sus funcionalidades son limitadas, aún así se puede hacer mucho más que con HOC1.