



**Instituto Politécnico Nacional
Escuela Superior de Cómputo**

Compiladores

**Práctica #4
“HOC 4”**

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

Agregar una Máquina Virtual de Pila al HOC3, esto requiere agregar varias funciones y modificar la sección de reglas, así como la parte del analizador léxico.

DESARROLLO

La parte de inicialización y del manejo de tabla de símbolos quedó prácticamente igual, y en la biblioteca “hoc.h” sólo se agregaron los prototipos de las funciones que la máquina virtual de pila ocupa:

```
extern Inst prog[];
extern Datum pop();
extern void execerror(char *, char *);
extern void eval(), ADD(), SUB(), MUL(), DIV(), negate(), power();
extern void assign(), bltin(), varpush(), constpush(), print(), complexpush(),
imagipush(), cflag();
```

Esta es la parte cambiada más significativa, la sección de reglas donde se esta usando máquina virtual de pila, además de que se agregan nuevos tokens:

```
%token <sym> NUMBER VAR BLTIN INDEF
%right '='
%left '+' '-'
%left '*' '/'
%left UNARYMINUS
%right '^' 'i'

%%

list:
    | list expr '\n' {code2(print, STOP); return 1;}
    | list asgn '\n' {code2(pop, STOP); return 1;}
    | list error '\n' {yyerrok;}
    | list '!' '\n' {exit(0);}
    | list '\n'
    ;

asgn:
    VAR '=' expr {code3(varpush, (Inst)$1, assign);}
    ;

complex:
    'i' {code2(cflag,
imagipush);}
    | NUMBER 'i' {code2(imagipush,
(Inst)$1);}
    | NUMBER '+' NUMBER 'i' {code3(complexpsh, (Inst)$1,
(Inst)$3);}
    | NUMBER '-' NUMBER 'i' {code4(cflag, complexpsh,
(Inst)$1, (Inst)$3);}
    ;

expr:
    complex
    | VAR
    {code3(varpush, (Inst)$1, eval);}
    | NUMBER
    {code2(constpush, (Inst)$1);}
    | BLTIN '(' expr ')' {code2(bltin, (Inst)$1-
>u.ptr);}
    | expr '+' expr {code(ADD);}
    | expr '-' expr {code(SUB);}
    | expr '*' expr {code(MUL);}
    | expr '/' expr {code(DIV);}
    | expr '^' expr {code(power);}
    | '-' expr %prec UNARYMINUS {code(negate);}
    | '(' expr ')'
    ;

%%
```

En la máquina de pila hubo pequeños cambios, sólo se modificaron lo datos de la pila, como lo lee, y también se agregó un par de funciones para los complejos:

```
void complexpush()
{
    double a, b;
    Datum d;
    a = ((Symbol *) *pc++)->u.num;
    b = ((Symbol *) *pc++)->u.num;
    if(flag)
        d.val = cmalloc(a, -b);
    else
        d.val = cmalloc(a, b);
    flag = 0;
    push(d);
}

void constpush() /*meter una constante a la pila*/
{
    Datum d;
    double a;
    a = ((Symbol *) *pc++)->u.num;
    d.val = cmalloc(a, 0);
    push(d);
}

void imagipush()
{
    Datum d;
    double a = 1.0;
    if (flag == 0)
        a = ((Symbol *) *pc++)->u.num;
    d.val = cmalloc(0, a);
    flag = 0;
    push(d);
}

void varpush() /*meter una variable a la pila*/
{
    Datum d;
    d.sym = (Symbol *) (*pc++);
    push(d);
}
```

```

void eval() /*evaluar una variable en la pila*/
{
    Datum d;
    d = pop();
    if (d.sym->type == INDEF)
        execerror("undefined variable", d.sym->name);
    d.val = d.sym->u.val;
    push(d);
}

void ADD() /*sumar los dos elementos superiores de la pila*/
{
    Datum d1, d2;
    d2 = pop();
    d1 = pop();
    d1.val = complexAdd(d1.val, d2.val);
    push(d1);
}

void SUB()
{
    Datum d1, d2;
    d2 = pop();
    d1 = pop();
    d1.val = complexSub(d1.val, d2.val);
    push(d1);
}

void MUL()
{
    Datum d1, d2;
    d2 = pop();
    d1 = pop();
    d1.val = complexMul(d1.val, d2.val);
    push(d1);
}

void DIV() //Division
{
    Datum d1, d2;
    d2 = pop();
    if (d2.val->real == 0.0 && d2.val->imag == 0.0)
        execerror("division by zero", (char *)0);
    d1 = pop();
    d1.val = complexDiv(d1.val, d2.val);
    push(d1);
}

```

```

void negate()
{
    Datum d;
    d = pop();
    d.val = cmalloc(-d.val->real, -d.val->imag);
    push(d);
}

void cflag()
{
    flag = 1;
}

void power()
{
    Datum d1, d2;
    d2 = pop();
    d1 = pop();
    d1.val = complexPow(d1.val, d2.val);
    push(d1);
}

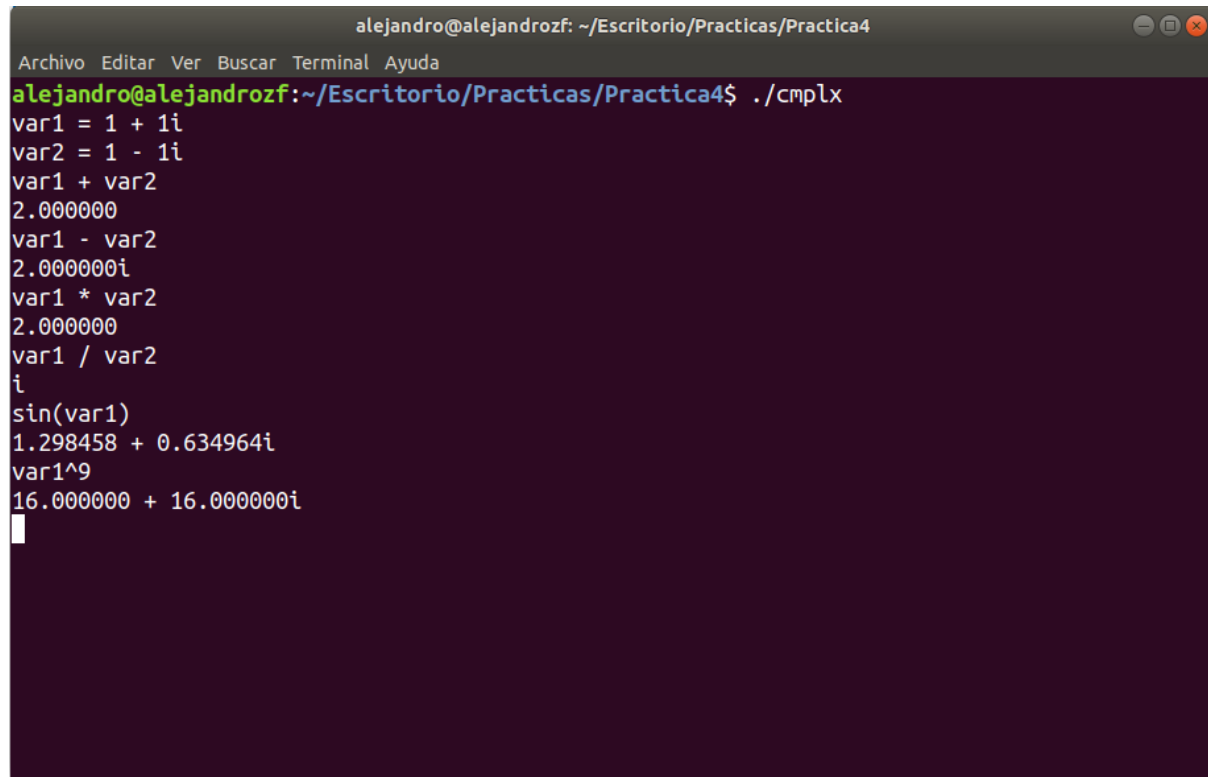
void assign() /*asignar el valor superior al siguientevalor*/
{
    Datum d1, d2;
    d1 = pop();
    d2 = pop();
    if (d1.sym->type != VAR && d1.sym->type != INDEF)
        execerror("assignment to non-variable", d1.sym->name);
    d1.sym->u.val = d2.val;
    d1.sym->type = VAR;
    push(d2);
}

void print() /*sacar el valor superior de la pila e imprimirlo */
{
    Datum d;
    d = pop();
    printc(d.val);
}

void bltin() /*evaluar un predefinido en el tope de la pila*/
{
    Datum d;
    d = pop();
    d.val = (*(complex **)(*)())(*pc++)(d.val);
    push(d);
}

```

Pruebas de funcionamiento:



```
alejandro@alejandrozf: ~/Escritorio/Practicas/Practica4
Archivo Editar Ver Buscar Terminal Ayuda
alejandro@alejandrozf:~/Escritorio/Practicas/Practica4$ ./cmlx
var1 = 1 + 1i
var2 = 1 - 1i
var1 + var2
2.000000
var1 - var2
2.000000i
var1 * var2
2.000000
var1 / var2
i
sin(var1)
1.298458 + 0.634964i
var1^9
16.000000 + 16.000000i
```

CONCLUSIONES

Esta fue probablemente la práctica que me costo más trabajo, ya que el concepto de la máquina virtual de pila tiene que estar bien claro para poder saber como funciona el HOC4, pero una vez que supe como funciona modificarlo fue mucho más rápido de lo que esperaba, incluso agregué unas cosas, como un símbolo para que terminara el programa, este es el signo de admiración.