UAEM | Universidad Autónoma del Estado de México

Universidad Autónoma del Estado de México

Centro Universitario UAEM Atlacomulco

“Calculadora en notación polaca inversa”

Presenta:

Segundo Antonio Elías Edgardo

Unidad de Aprendizaje:

Estructuras de datos

Fecha de entrega:

13 de septiembre de 2017

1 Introducción

En esta práctica se realizará una calculadora polaca inversa en lenguaje C, para ello se manipulará una cadena de entrada y discriminar números de operadores para después utilizarlos en el proceso de derivación de resultados.

En esta práctica no se introduce ningún concepto nuevo de programación, sin embargo, sirve como preludio para el tema de pilas, la solución óptima para este problema hace uso de las mismas, sin embargo, se buscó crear una solución sin el uso pilas y finalmente se consiguió implementar solo con arreglos y un indicador, resultando una suerte de pila en esencia, pero sin la implementación de la misma de manera explícita.

Por otra parte, esta práctica también sirve como repaso al tema de prevalencia de operadores en C, dado que el cambio de dinámica en la asociación de operadores requerirá evaluar su similitud con la notación algebraica y la notación polaca.

También se utilizaron algunas funciones de las librerías estándar de C para manipulas las cadenas de entrada y convertirlas a números flotantes o evaluar sin son operadores en su defecto, fue así como se logró aceptar números enteros en un rango del 0 al 9 con 5 cifras, esto es 0-99999 números para realizar las operaciones.

Una excepción que no se tomó en cuenta fueron los números negativos, dado que el propio signo negativo de un numero puede significar un operador de resta, por esta razón se decidió solo trabajar con números positivos y sin puntos decimales, puesto que el manejo de cadenas se dificultaría por esta razón.

A continuación, se muestra el marco teórico con los conceptos relacionados a la notación polaca.

2 Marco teórico

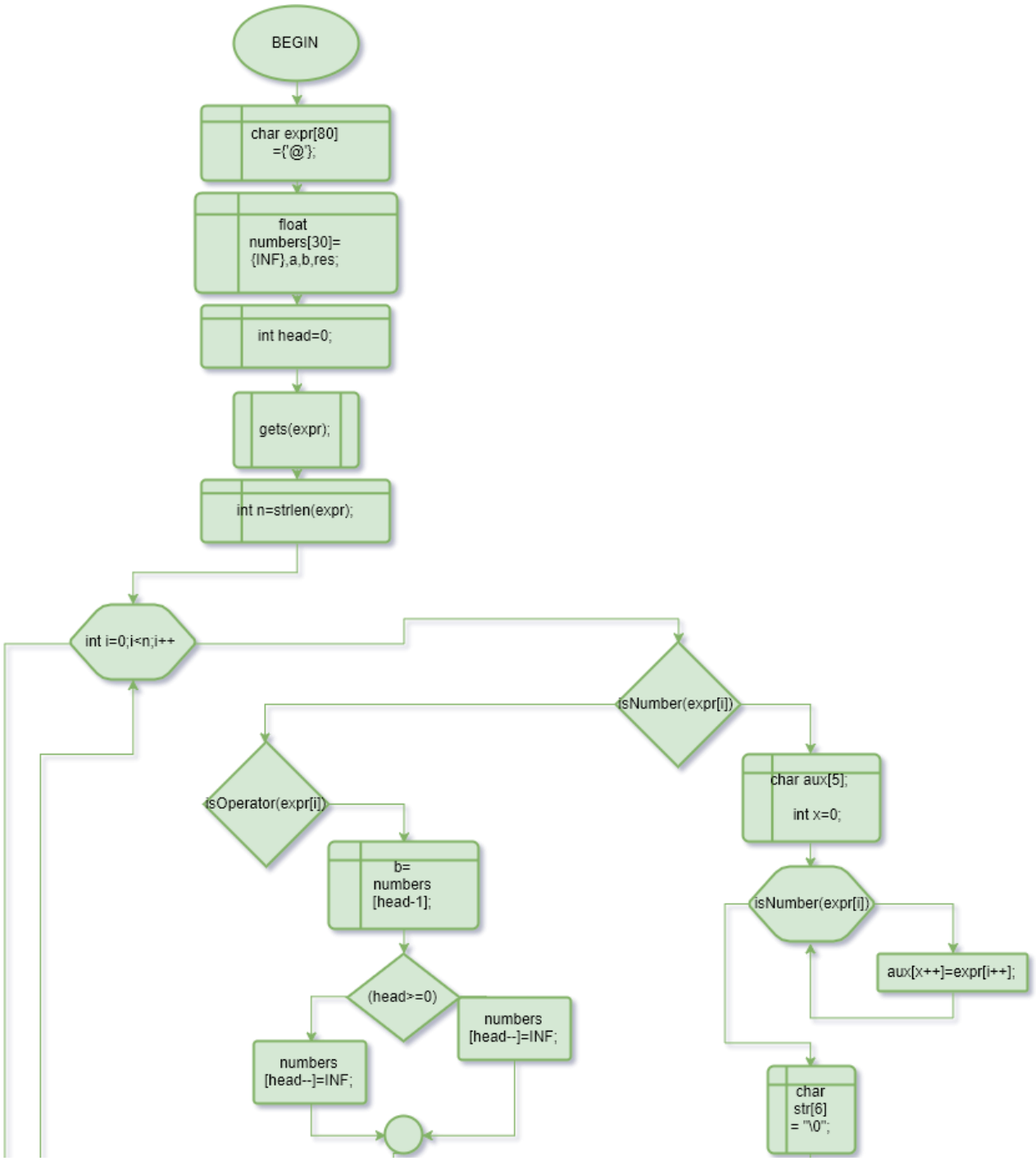
El único concepto nuevo utilizado en esta práctica fue la introducción de la notación polaca, en este caso en su forma inversa, en realidad solo se trata de un cambio de orden en el ingreso de los operandos y los operadores y para llevarlo a cabo se utiliza una pila auxiliar, el siguiente ejemplo muestra el procedimiento:

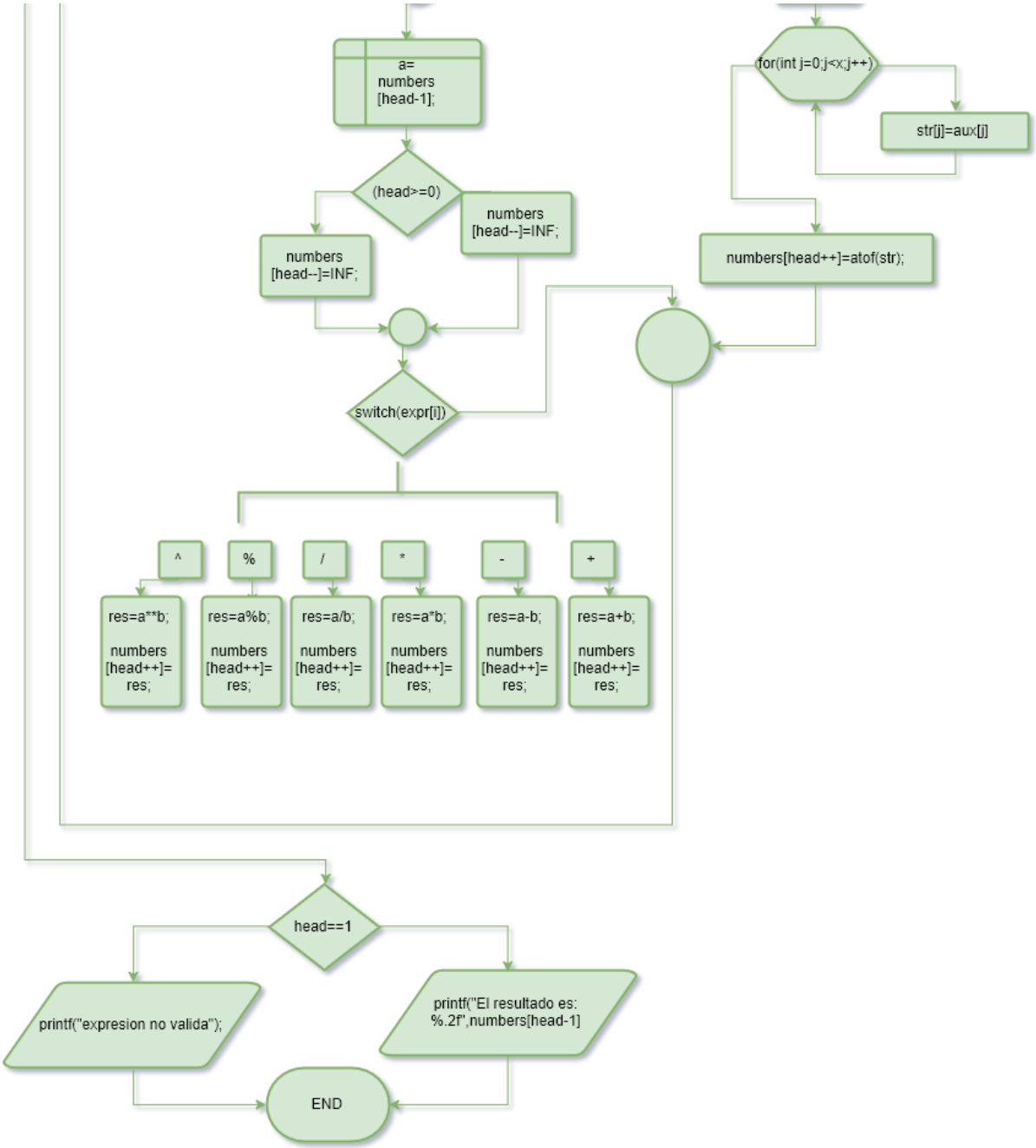
Expresión= 5 4 + 1 * 2 +	
+	Toma los números 4 5 y los suma, resultando un 9
*****	Toma el 9 y el 1 y los multiplica, resultando un 9
+	Toma el 9 y el 2 y los suma, obteniéndose un resultado de 11
Resultado=	11

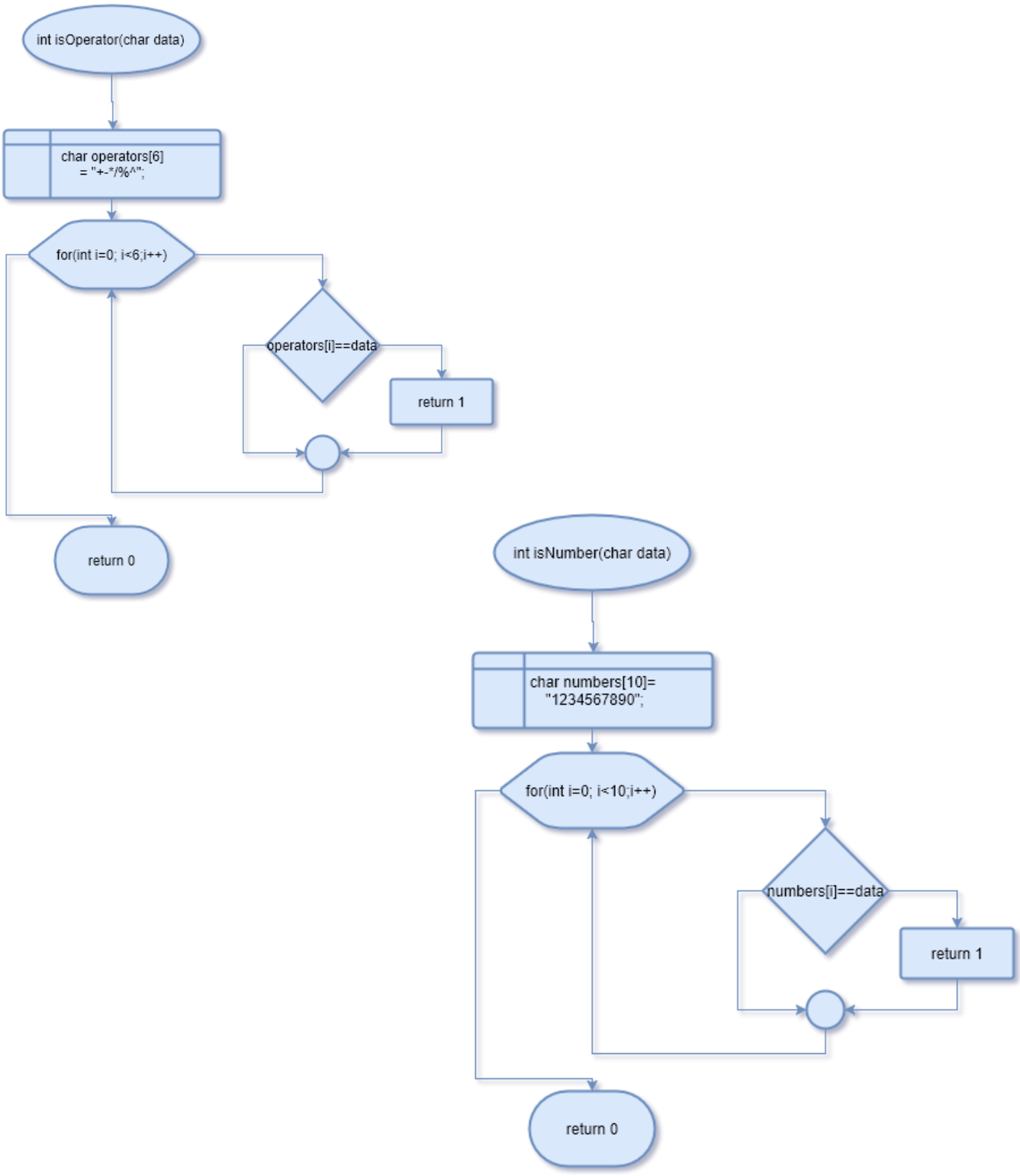
Como se mencionó en la introducción la forma correcta de solucionarlo (al menos la óptima) es haciendo uso de pilas, sin embargo, solo se usó un arreglo con un indicador de la cabeza del arreglo, con esto se emulo el funcionamiento de una pila básica, pero sin uso de apuntadores ni la implementación de las funciones.

3 Desarrollo

A continuación, se mostrarán los diagramas de flujo de la práctica realizada:




a=
numbers
[head-1];
(head>=0)
numbers
[head--]=INF;
numbers
[head--]=INF;
switch(expr[i])
^
%
/
*
-
+
res=a**b;
numbers
[head++]=
res;
res=a%b;
numbers
[head++]=
res;
res=a/b;
numbers
[head++]=
res;
res=a*b;
numbers
[head++]=
res;
res=a-b;
numbers
[head++]=
res;
res=a+b;
numbers
[head++]=
res;
for(int j=0;j<x;j++)
str[j]=aux[j]
numbers[head++]=atof(str);
head==1
printf("expresion no valida");
printf("El resultado es:
%.2f",numbers[head-1]
END


int isOperator(char data)
char operators[6]
= "+-*/%^";
for(int i=0; i<6;i++)
operators[i]==data
return 1
return 0
int isNumber(char data)
char numbers[10]=
"1234567890";
for(int i=0; i<10;i++)
numbers[i]==data
return 1
return 0

polaca.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define INF 100000

int isNumber(char data)
{
    char numbers[10]= "1234567890";
    for(int i=0; i<10;i++)
    {
        if(numbers[i]==data) return 1;
    }
    return 0;
}

int isOperator(char data)
{
    char operators[6]= "+-*/%^";
    for(int i=0; i<6;i++)
    {
        if(operators[i]==data) return 1;
    }
    return 0;
}

/*void printArray(float a[],int n)
{
    for(int i=n-1; i>=0;i--)
    {
        printf("%.1f , ",a[i]);
    }
    printf("\n");
}*/


int main(){
    char expr[80]={'@'};
    float numbers[30]={INF},a,b,res;
    int head=0;
    gets(expr);
    int n=strlen(expr);
    for(int i=0;i<n;i++)
    {
        if(isNumber(expr[i]))
        {
            char aux[5];
            int x=0;
            while( isNumber(expr[i]) )
            {
                aux[x++]=expr[i++];
            }
```

```
            char str[6] = "\0";
            for(int j=0;j<x;j++) str[j]=aux[j];
            numbers[head++]=atof(str);
        }

        else if (isOperator(expr[i]))
        {
            b=numbers[head-1];
            if(head>=0)
            {
                numbers[head--]=INF;
            } else { printf("No valida"); return 1;}
            a=numbers[head-1];
            if(head>=0)
            {
                numbers[head--]=INF;
            } else { printf("No valida"); return 1;}
            switch(expr[i])
            {
            case '+':
                res=a+b;
                numbers[head++]=res;
                break;
            case '-':
                res=a-b;
                numbers[head++]=res;
                break;
            case '*':
                res=a*b;
                numbers[head++]=res;
                break;
            case '/':
                res=a/b;
                numbers[head++]=res;
                break;
            case '%':
                res=fmod(a,b);
                numbers[head++]=res;
                break;
            case '^':
                res=pow(a,b);
                numbers[head++]=res;
                break;
            }

        }
        /*printArray(numbers,head);*/
    }
    if(head==1){
    printf("El resultado es: %.2f",numbers[head-1] );}
    else printf("expresion no valida");
    return 0;
```

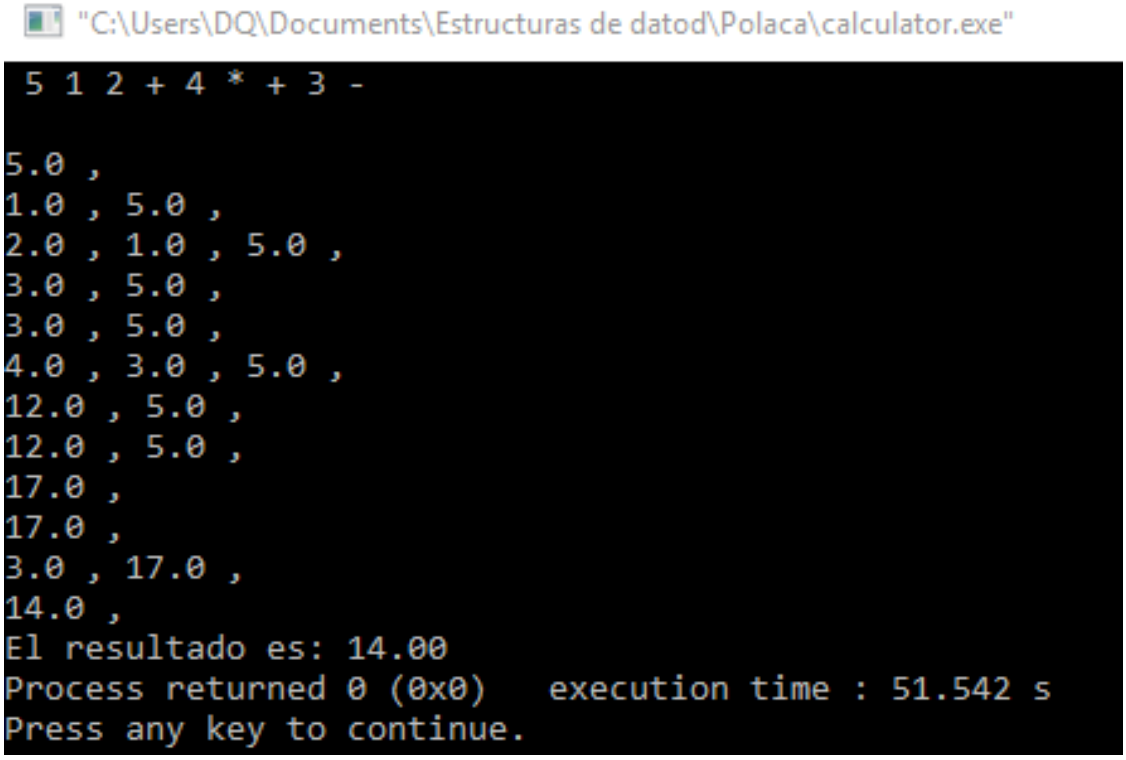
```
}
```

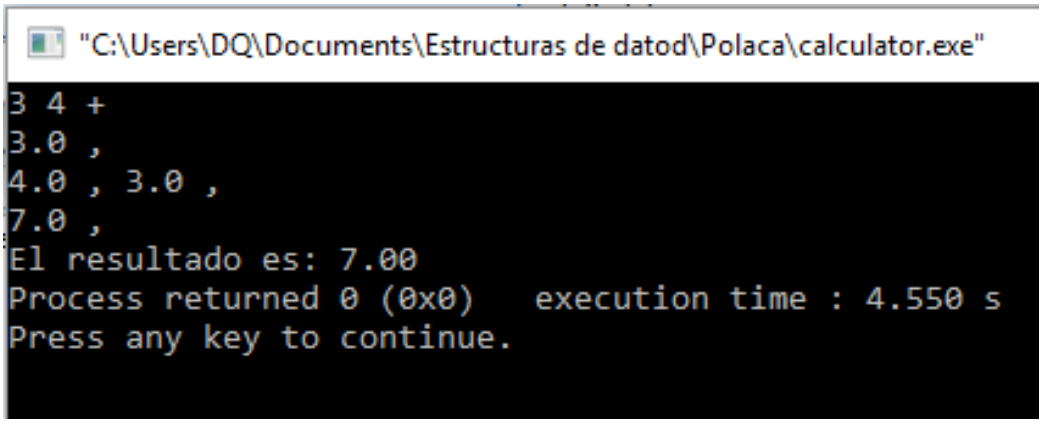
Polish.cpp

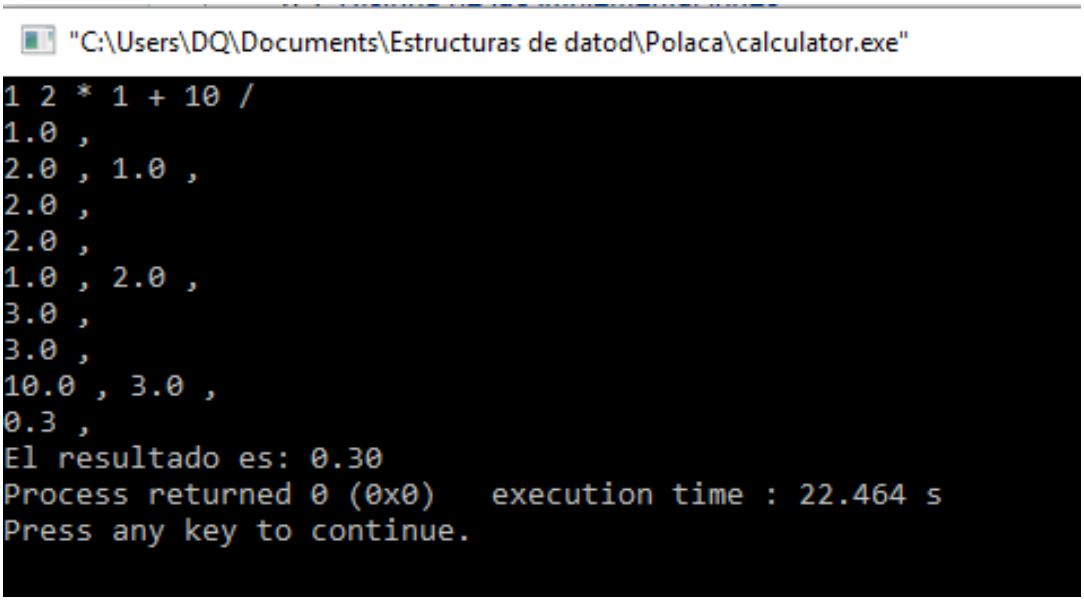
```
#include <bits/stdc++.h>
using namespace std;

bool isNumber(char data)
{
   string numbers= "1234567890";
   for(char n:numbers)
   {
      if(data==n) return true;
   }
   return false;
}
bool isOperator(char data)
{
   string operators= "+-*/";
   for(char n:operators)
   {
      if(data==n) return true;
   }
   return false;
}
void printStack(stack<double> d)
{
   while(!d.empty())
   {
      cout<<d.top()<<",";
      d.pop();
   }
   cout<<"\n";
}



int main(){
   stack<double> numbers;
   string expr;
   double a,b,res;
   getline(cin, expr, '\n');

   for(int i=0; i<expr.size(); i++)
   {
      if(isNumber(expr[i]))
      {
         numbers.push(int(expr[i]-'0'));
      }
      else if (isOperator(expr[i]))
      {
         b=numbers.top();
         if(!numbers.empty())numbers.pop();
         a=numbers.top();
```

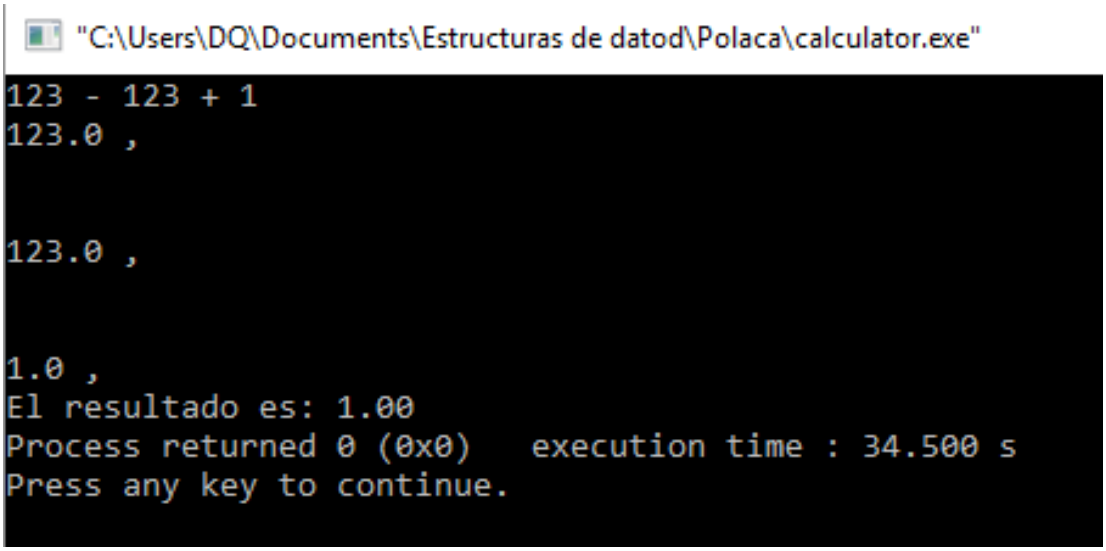
```
            if(!numbers.empty())numbers.pop();
            switch(expr[i])
            {
            case '+':
               res=a+b;
               numbers.push(res);
               break;
            case '-':
               res=a-b;
               numbers.push(res);
               break;
            case '*':
               res=a*b;
               numbers.push(res);
               break;
            case '/':
               res=a/b;
               numbers.push(res);
               break;
            }
         }
            printStack(numbers);
      }

}
```

4 Resultados (código en C)









5 Conclusiones

El programa tuvo un funcionamiento adecuado para los objetivos de la práctica, un área de oportunidad a mejorar es el ingreso de números negativos y números con punto decimal, también se desarrolló una segunda versión implementada en C++ con las pilas proporcionadas por la STL, lo que marca una diferencia notable entre la facilidad de

implementar entre ambos lenguajes, sin embargo, ambas versiones arrojaron el mismo resultado.

6 Referencias

Joyanes Aguilar, L. (2005). *Programacion en C.* Madrid: Mc Graw Hill.

Langsam, Y., Augenstein, M., & Tenenbaum, A. (1997). *Estructuras de datos con C y C++.* Mexico: Prentice Hall.

Marquez Fausto, T. G., Osorio Angel, S., & Olvera Perez, E. N. (2011). *Introduccion a la programacion estructurada en C.* Mexico: Prentice Hall.

Tenebaum, A., Langsam, Y., & Augenstein, M. (1993). *Estructuras de datos en C.* Mexico: Prentice Hall.