|  |
| --- |
| Materia:algoritmos y estructura de datos |
| Parcial Practico Nº1 |
|  |
|  |
|  |
|  |

**14/05/2015**

Alumnos:

Herrera, Nicolás 35135164

Malano, Leandro 38883701

Quinteros, Nicolás 34351470

**DEfensa del algoritmo realizado**

En este trabajo se desarrolló un programa que nos permite calcular las operaciones elementales de suma (+), multiplicación (\*), resta (-) y división (/) con anotación Posfija, teniendo en cuenta las estructura Pilas y listas. La Pila utilizada en nuestro algoritmo la llamamos Tokens y en donde se almacenan los operandos y el operador.

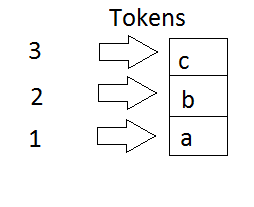
Analizando nuestro código, tenemos las clases Nodo, Lista y Tokens (Pila).

En la clase Nodo, tenemos en la parte privada un tipo de dato entero largo (definido como calculadora) llamado dato y un puntero llamado next .Luego pasando en la parte publica tenemos los métodos de la Clase Nodo, que contiene la inicialización de Nodo para cuando el puntero next, está o no vacío, luego tenemos la función void set\_next(Nodo \*n) que ingresa un parámetro del tipo nodo n y con next que tiene el valor del nodo n. Sigue la función calculadora get\_dato(), la cual retorna el un elemento del tipo calculadora llamado dato, la función Nodo \*get\_next(), la cual retorna un puntero next y por ultimo esta la función bool vacio() la cual retorna si next está en NULL.

En la Clase Lista, tenemos como privado al Nodo puntero czo, y en la parte publica tenemos la lista se inicializa en czo con un nuevo nodo creado si no se ingresó ningún nodo a dicha lista, si la lista posee un nodo puntero n , czo se inicializa con el valor del puntero n. La función void anadir (calculadora a) crea un nodo nuevo cuyo nodo utiliza el método set\_next(czo) y czo toma el valor de nuevo, con el cual forma una lista. La función bool verifvacia (void) devuelve un booleano si encuentra o no el puntero czo vacio utilizado la función bool vacio() heredara de la Objeto Nodo. En la función void del(void) la función elimina un nodo de la lista, utilizando un nodo puntero aux y se llama a la función get\_next() para czo. En calculadora vercabeza(void) (el cual ingresa un número del tipo calculadora),con un if si la lista es vacia(usando la función verifvacia()), el programa tira un mensaje de error y luego retorna czo el cual llama a la función get\_dato(). En la función Lista \*resto(void) se crea una lista en donde czo usa la función get\_next() y retorna dicha lista.En la función Void Impre(void) se crea un nodo aux el cual almacena a czo , y si aux retorna un valor(en esta caso un nodo) distinto de NULL se muestra el valor de aux la cual utiza la función get\_dato() y luego aux se redefine como aux llamando a la función get\_next(). Luego tenemos las funciones void suma(), void resta(), void mult() y void div() la cual funcionan de manera parecidas: se verifica con un verifvacia() si la lista esta vacía y retorna 0, sino se retorna la función vercabeza adicionada, restada, multiplicada o dividida con la función resto() que llama de manera recursiva a dicha función( sea suma, resta,mult o div).

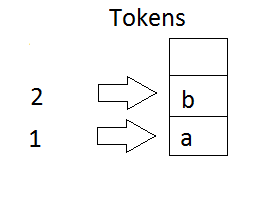
Luego en la Clase Tokens (Pila), se implementan las funciones heredara de la Clase Lista. En la función void apilar(calculadora x) se implementa la función heredada anadir(calculadora x). La función void desapilar(void) implementa la función del() de la clase Lista.

En la Función principal (int Main() ) se lleva a cabo la creación de un Tokens (Pila), luego declaramos variables enteras llamadas a, b y d y creamos una variable char c.

Al correr el programa, nos pide que ingresemos el valor de nuestra variables numéricas a y b, donde a es el primer operador y b es el segundo operador y en c se introduce el operando, luego estos valores se apilan en Tokens.

En la imagen vemos que el orden de apilamiento de cada una de las variables.

Luego el programa en el Main, ingresamos a un ciclo switch donde verifica el valor de c .

Si en switch se ingresa c como un carácter +,-,\* o/, se ejecuta el case, de acuerdo al operando elegido, y dentro del case, se desapila c (ver imagen siguiente), para que el valor que toma c (al ser un char y la función espera un int, toma el valor del código ASCII de c) no se considere en la cuenta y realice la operación correspondiente, y muestra el valor obtenido con anotación postfija (juntos con el operando y ambos operadores) y en d se almacena el valor de la operación realizada.

Si c no es un operando indicado, el programa por defecto, da un mensaje de error, haciendo que el programa no funcione.

Saliendo del switch, se desapila a y b para apilarse d en un token y mostrar su valor desde el token, utilizando la función Void Impre(void).

