C++ desk calculator

PROGRAMA BASADO EN EL LIBRO THE C++ PROGRAMMING LANGUAGE

PRESENTADO POR:
IVÁN MAMANI ARISACA

El programa presentado permite a un usuario definir funciones en la calculadora.

Sugerencia: defina una función como una secuencia de operaciones tal como las habría escrito un usuario.

Dicha secuencia se puede almacenar como una cadena de caracteres o como una lista de tokens. Luego se puede leer y ejecutar esas operaciones cuando se llama a la función.

Para definir una función se debe definir de la siguiente forma"$func(x) { x = x + 2; }“ y después llamar "$func(20)"

Partes principales del programa

- THE PARSER – Análisis sintáctico
- INPUT - Maneja la entrada y el análisis léxico
- SYMBOL TABLE – mantiene información permanentemente
- THE DRIVER – Maneja la inicialización, salida y errores.

The Parser

Usa los Token_stream que básicamente encapsula la lectura de caracteres y su composición en tokens que consta de un par de elementos siendo el primer el tipo de token y el segundo el elemento el valor.

Por ejemplo {number, 123.45}

Algunas funciones que se utiliza son el get y el current.

ts.get() – Para leer y retornar el siguiente token.

ts.current() – Para obtener el mas reciente token (“token actual”)

Algunas funciones adicionales

Expr() - maneja las adiciones y sustracciones

Term() - maneja la multiplicación y la división

Prim() – con esta función no es necesario realizar bucles

Input

En esta parte se define como se inicializa un Token_stream con un input stream.

El istream se pasa como un puntero y no como referencia.

Un token stream guarda 3 valores (un puntero al input stream, un buleano y el token actual)

El static_cast es necesario porque no hay una conversión implícita de char a kind.

```
class Token_stream {
public:
        Token_stream(istream& s) : ip{&s}, owns{false} { }
        Token_stream(istream* p) : ip{p}, owns{true} { }

        ~Token_stream() { close(); }

        Token get();            // read and return next token
        Token& current();       // most recently read token

        void set_input(istream& s) { close(); ip = &s; owns=false; }
        void set_input(istream* p) { close(); ip = p; owns = true; }

private:
        void close() { if (owns) delete ip; }

        istream* ip;                    // pointer to an input stream
        bool owns;                      // does the Token_stream own the istream?
        Token ct {Kind::end} ;          // current token
};
```

Symbol table

- Nos recomienda hacer uso de la librería estándar “map” como el symbol table.
- Esta recomendación es básicamente porque la librería estándar como otras que pueden existir han sido bien elaboradas y se ha prestado mucha atención en cuanto a su diseño e implementación.

Driver

El driver es utilizar para poner en marcha el programa.

En este caso el main() para manejar configuración y reporte de errores y la función calculate() para manejar el cálculo actual.

Como tarea principal del main loop es la lectura de las expresiones e imprimir posteriormente la respuesta. Esto se logra con la línea siguiente:

cout << expr(false) << '\n';

Donce false indica que no es necesario llamar a la función ts.get()

```
Token_stream ts {cin};    // use input from cin

void calculate()
{
        for (;;) {
                ts.get();
                if (ts.current().kind == Kind::end) break;
                if (ts.current().kind == Kind::print) continue;
                cout << expr(false) << '\n';
        }
}

int main()
{
        table["pi"] = 3.1415926535897932385;    // insert predefined names
        table["e"] = 2.7182818284590452354;

        calculate();

        return no_of_errors;
}
```

Error handling

- Aquí se hace uso de la función error() que básicamente cuenta los errores , manda o imprime el mensaje del error y/o retorna un valor
- También se usa el stream cerr que sirve como un reportardor de errores.

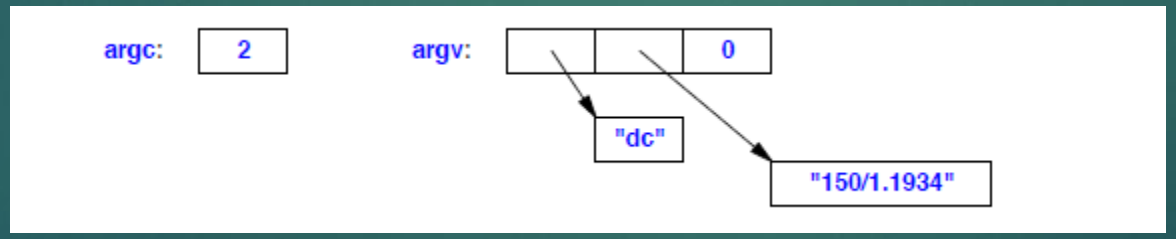
```
int no_of_errors;

double error(const string& s)
{
        no_of_errors++;
        cerr << "error: " << s << '\n';
        return 1;
}
```

Proceso del programa

El programa comienza llamando al main() dado con dos argumentos.

El primero nos dice la cantidad de argumentos llamado Argc y el Segundo es un array de argumentos llamado Argv (el tipo del Argv es **char∗[argc+1]**. El nombre del programa es pasado como Argv[0] asi el Argc siempre será almenos 1. También nos menciona que la lista de argumentos siempre termina en cero (**argv[argc]==0**.)



Conclusiones

La calculadora propuesta por el autor del libro nos muestra como estructurar y usar diferentes funciones junto con conceptos nuevos para que esta sea más óptima.

El programa realizado toma como base los conceptos del libro para poder realizar la calculadora y se implementa la operaciones con funciones.

Aún se necesita aprender mucho más en temas de estructuración mas que en la implementación en sí del programa.