Proyecto 3

Eduardo Medina

Desktop calculator

Dividida en 4 partes:

- Parser
- Input
- Symbol table
- Driver

Lo más importante de cada parte:

Parser

```
class Token_stream {
public:
    explicit Token_stream(istream& s) : ip{&s},owns(false), ct{ .kind: Kind::end}{}
    explicit Token_stream(istream* p) : ip{p},owns(true), ct{ .kind: Kind::end}{}
    ~Token_stream(){close();}

    Token get(){...}

    Token & current(){...}

    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;
    bool owns;
    Token ct{ .kind: Kind::end};
};
```

Input

```
double term(bool);
double expr(bool get){...}
double prim(bool get){...}
double term(bool get){...}
```

Se declara de esta forma porque existen dependencias.

Symbol table

```
extern map<string,double> table;
```

Driver

```
#include "input.h"

int no_of_errors;
map<string,double> table;
Token_stream ts{ &: cin};

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

int main(int argc, char* argv[]){
    table["pi"]=3.14159265;
    table["e"]=2.718281828;
    calculate();
    return no_of_errors;
}
```

Conclusiones

- Se aprendió a utilizar extern keyword.
- Repaso de la organización de un programa en archivos.
- Nuestro nivel de programación parece bastante básico, si se compara al aplicado en este programa. Falta tiempo para practicar.