```
//                              Calc v1.0
+ ... // Подключение заголовков, объявления using, определение пользовательских типов, описание программы

+ class Error { ... }                                                    // Класс работы с ошибками
+ struct token  { ... }                                                  // Структура для хранения лексемы
typedef vector<token>::iterator  vectok_iter;
+ void print_help { ... }
+ inline strsz_t dot_ix(const string &str) { ... }                       // Нахождение положения точки (десятичного разделителя)
+ double str2num(const string &str) { ... }                              // Преобразование строки в число типа double
+ inline vectok_iter right_pair(vectok_iter  iter) { ... }               // Нахождение итератора парной скобки
+ void error_check(const  vector<token>  &tvec) { ... }                  // Функция обработки смысловых ошибок
+ double expr_calc(vectok_iter  beg, vectok_iter  end) { ... }           // Вычисление значения выражения из вектора лексем,
+ void tokens_vec_create(vector<token>  &tvec, const string &exprn)  { ... }  // Создание вектора лексем из строки exprn

int main()
{
    cout << "\t\tCalc v1.0\n\tType '\\help' to print help\n" << endl;
    string expression;                                                   // Переменная для хранения введенного выражения
    vector<token> tokens_vec;                                            // Вектор лексем
    while (getline(cin, expression)) {                                   // Главный цикл программы
        if (expression.empty())
            continue;
        if (expression == "\\exit" || expression == "\\Exit")            // Выход
            break;
        else if (expression == "\\help")                                 //
            print_help();                                                // Вывод справки
        try {
            if (expression[expression.size()  - 1] == '=')               // Обработка выражения
                tokens_vec_create(tokens_vec,  expression), cout << "\t\t\t"  //
                << expr_calc(tokens_vec.begin(),  tokens_vec.end()) << endl;
            // cout << endl;                                             //
            // for (auto tok : tokens_vec)                               // Отладка
            // cout << tok.type << " : " << tok.value << endl;           //
        }
        catch (Error err) {                                              // Обработка ошибок
            cout << err.what() << endl;                                  //
            continue;                                                    //
        }
    }
    cout << "\n\n" << endl;
    return 0;
}
```