```
        /// <summary>
        /// Implements CFG Rule: <simple expression> -> [<sign>] <term> {<addop> <term>}*
        /// </summary>
        /// <returns>The result of the expression.</returns>
        private int SimpleExpression()
        {
            if (IsError)
                return -1;

            Debug(true, "SimpleExpression()");

            int left, right, temp, opcode;
            int signVal = 0;

            if (isSign())
            {
                signVal = Sign();
            }

            left = Term();

            if (signVal == -1)
            {
                Quads.AddQuad(MULTIPLY, left, Minus1Index, left);
            }

            while (isAddOp() && !IsError)
            {
                opcode = AddOp();
                right = Term();
                temp = GenSymbol();
                Quads.AddQuad(opcode, left, right, temp);
                left = temp;
            }

            Debug(false, "SimpleExpression()");
            return left;
        }

        /// <summary>
        /// Implements CFG Rule: <term> -> <factor> {<mulop> <factor> }*
        /// </summary>
        /// <returns></returns>
        private int Term()
        {
            if (IsError)
                return -1;
            int left, right, opCode, temp;

            Debug(true, "Term()");
            left = Factor();

            while (isMulOp() && !IsError)
            {
                opCode = MulOp();
                right = Factor();
                temp = GenSymbol();
                try
                {
                    Quads.AddQuad(opCode, left, right, temp);
                }
                catch (DivideByZeroException e)
                {
                    Console.WriteLine(e.Message);
                }
                left = temp;
            }

            Debug(false, "Term()");
            return left;
        }

        /// <summary>
        /// Implements CFG Rule: <factor> -> <unsigned constant> | <variable> | $LPAR <simple expression> $RPAR
        /// </summary>
        /// <returns></returns>
        private int Factor()
        {
            if (IsError)
```

```
                    return -1;

                Debug(true, "Factor()");

                int index = 0;

                if (isUnsignedConstant())
                {
                    index = UnsignedConstant();
                }
                else if (isVariable())
                {
                    index = Variable();
                }
                else if (Scanner.TokenCode == LPAR)
                {
                    GetNextToken();
                    index = SimpleExpression();
                    if (Scanner.TokenCode == RPAR)
                        GetNextToken();
                    else
                        UnexpectedTokenError("RPAR");
                }
                else
                    UnexpectedTokenError("UNSIGNED CONSTANT or VARIABLE or LPAR");

                Debug(false, "Factor()");
                return index;
            }
```