Raul-Mihai Rusu

Group 937

Github Link: https://github.com/RaulRusu/FLCD2023/tree/feature/scanner

PIF: represented as a list (System.Collection.Generics.List) of tuples (string, HashTablePosition)

Where HashTablePosition represents the Bucket and Index form the hash table

1. Reserved token

```
1 reference
private bool isReserved(string token)
{
    return reservedTokens.Any(reservedToken => reservedToken.Equals(token));
}
```

reservedTokens are read from file, and we check if the specific token is in the list of reservedTokens.

2. Identifier

```
//^[a-zA-Z][a-zA-Z0-9]*$
1 reference
private bool IsIdentifier(string token)
{
    var match = Regex.Match(token, @"^[a-zA-Z][a-zA-Z0-9]*$");
    return match.Success;
}
```

Using regex we check if the token is an Identifier. From the start of the string we math for a letter and then for a letter or a digit 0 or more times until the end of the string to avoid partial matches.

3. Int

```
//^[-+]?[0-9]+$
1 reference
private bool IsInt(string token)
{
    var match = Regex.Match(token, @"^[-+]?[0-9]+$");
    return match.Success;

}
```

Chek for + or – 0 or 1 time then digits one or more times. Again we match from the start to the end of string in order to avoid partial matches.

4. Float

```
//^[-+]?[0-9]+\.[0-9]+$
1 reference
private bool IsFloat(string token)
{
    var match = Regex.Match(token, @"^[-+]?[0-9]+\.[0-9]+$");
    return match.Success;
}
```

From start to the end of the string: match + or – 0 or 1 time, digits one or more times, next “.”, and after digits one or more.

5. Char

```
//^[0-9a-zA-Z]$
1 reference
private bool IsChar(string token)
{
    var match = Regex.Match(token, @"^[0-9a-zA-Z]$");
    return match.Success;
}
```

From start to end of string: match any digit or letter.

6. Bool

```
1 reference
private bool IsBool(string token)
{
    return token == "true" || token == "false";
}
```

Chek if the token is “true” or “false”.