Raul-Mihai Rusu

Group 937

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

1. Symbol table

The symbol table is represented internally as a hash table. The number of buckets it's constant but each bucket is represented as a dynamic linked. The hash function used is djb2. The implementation of the hash table is very simplistic, the load factor is not taken in consideration and no rehashing will be done if the underlying hash table becomes inefficient.

For now, only a single function is exposed in the symbol table class "Position". It takes a token (represented as a string) and the function returns the position of that token in the internal representation of the symbol table, in this case hash table. If the token is not present it will be added, and the position of this newly added element is returned.

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

Where HashTablePosition represents the Bucket and Index form the hash table

3. Regex for tokens

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".

4. Class diagram