**عنوان گزارش: پروژه نهایی کامپایلر**

**نام استاد: دکتر فرزانه زارعی**

**تدریس ‌یار: آقای امیر سجاد حسین پور**

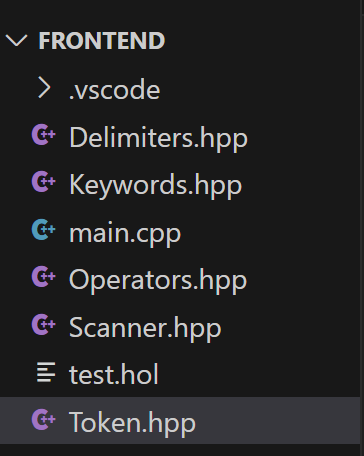
**ارائه دهنده: گلسا قاسمی (شماره دانشجویی: 401130673)**

**تاریخ ارائه گزارش: 14/11/1403**

**مقدمه**

**در این فاز از پروژه کامپایلر به پیاده سازی تحلیلگر لغوی می‌پردازیم. ما یک فایل را به عنوان ورودی دریافت می‌کنیم وآن را به صورت توکن در لیستی ذخیره می‌کنیم.**

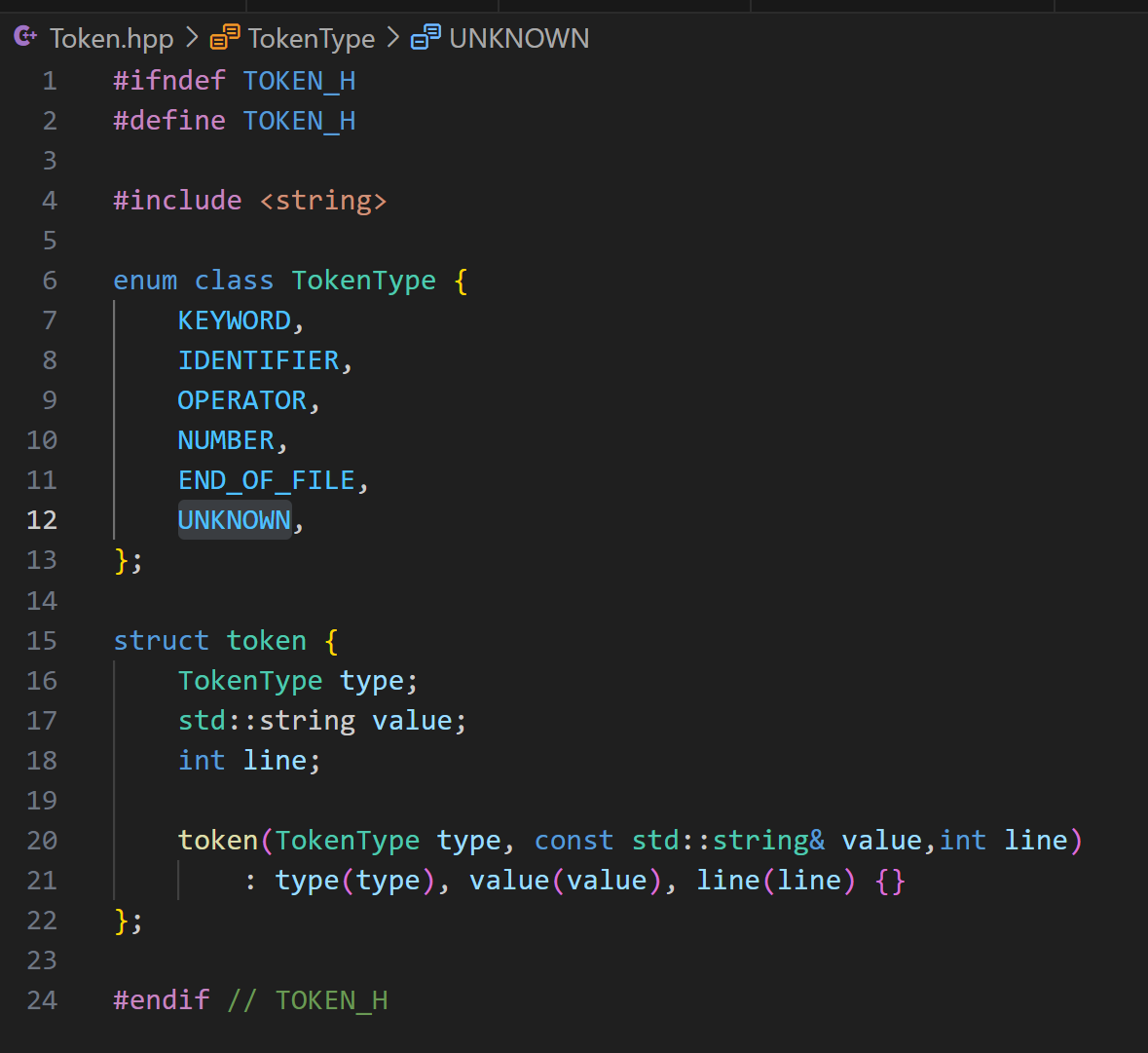
**فایل های ساخته شده در فولدر Frontend**

****

**Token.hpp: در این فایل کلاسی از نوع enum ساختیم که یک لیست از توکن ها رh داخل آن تعریف کردیم:**

1. **KEYWORD: برای کلمات کلیدی داخل برنامه**
2. **IDENTIFIE: برای نامی که به متغیر می‌دهیم**
3. **OPERATOR: برای عملگرها**
4. **NUMBER: برای اعداد integer**
5. **END\_OF\_FILE: برای بررسی زمانی که به انتهای فایل می‌رسیم**
6. **UNKNOWN**

**و در ادامه struct برای توکن‌ها تعریف می‌کنیم.**

****

**Delimiters.hpp: در این فایل عباراتی که برای مشخص کردن محدوده مورد استفاده قرار می‌گیرند را تعریف میکنیم مثل پرانتز، آکولاد، نقطه و ...**

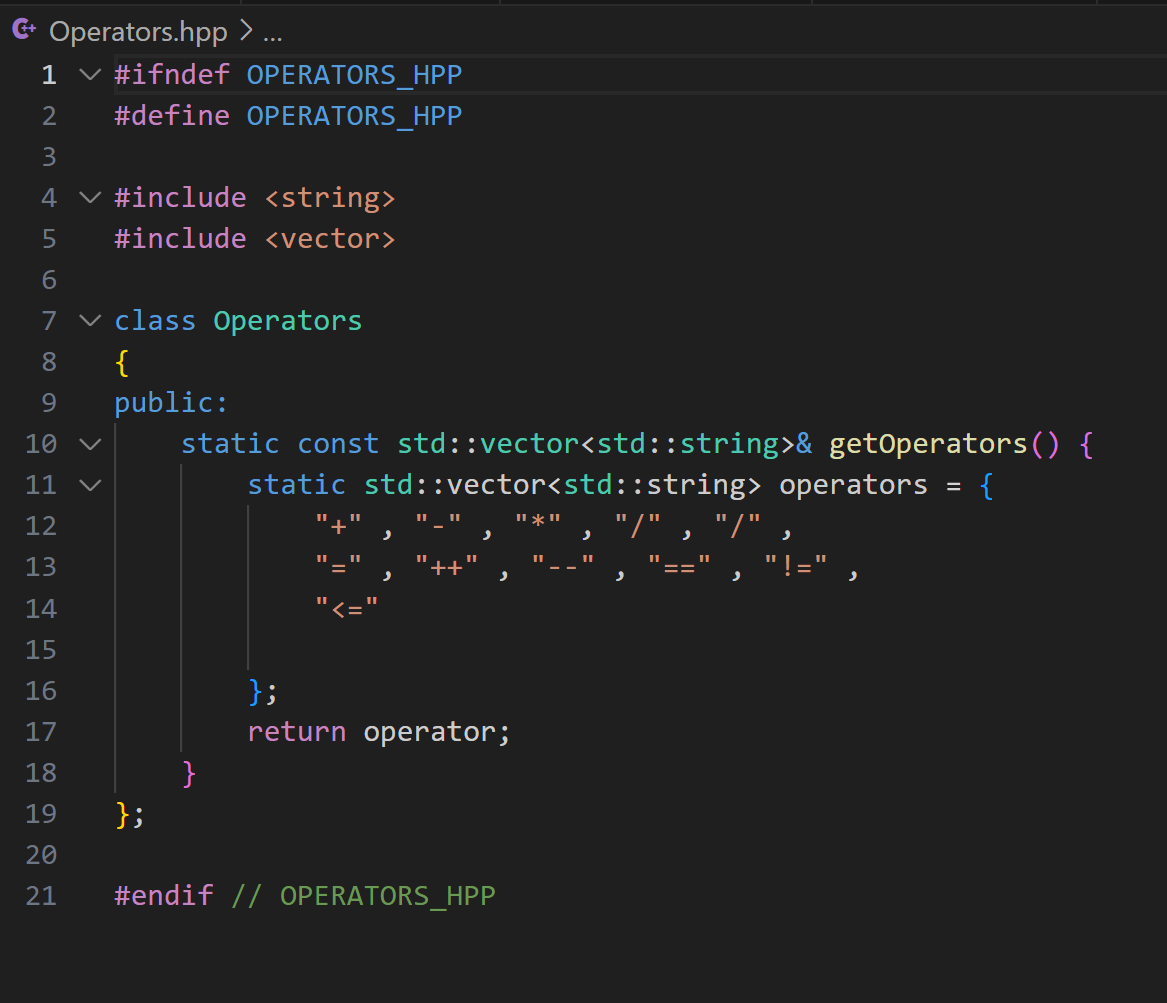
**A computer screen shot of text

Description automatically generated**

**Keywords.hpp: در این فایل کلمات کلیدی اولیه را تعریف می‌کنیم.**

****

**Operators.hpp: در این فایل عملگرها مثل ضرب ، جمع را تعریف می‌کنیم.**

****

**Scanner.hpp: در این فایل کلاسی به نام Scanner تعریف می‌کنیم. این کلاس شبیه DFD عمل می‌کند با این تفاوت که ساختار آن خیلی ساده تر است.**

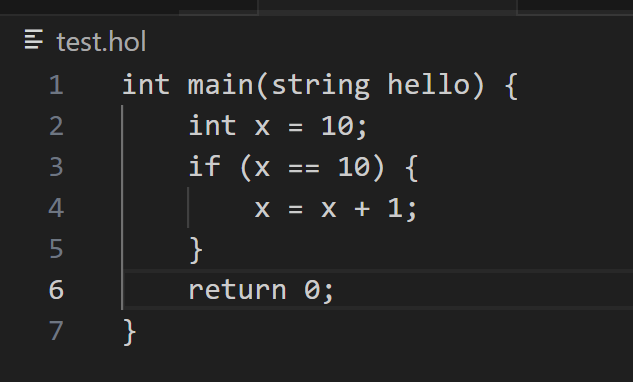
**متد های تعریف شده در کلاس Scanner**

1. **متد isKeyword: چک می‌کند آیا توکن ورودی جز keyword هست یا خیر**
2. **متد isOperator: چک می‌کند آیا توکن ورودی جز operatorهست یا خیر**
3. **متد isDelimiter: چک می‌کند آیا توکن ورودی جز Delimiter هست یا خیر**

**و در ادامه چک می‌کنیم که توکنی که داریم حرف ، عدد یا فضای خالی است.**

**اگر فضای خالی باشد آن را رد می‌کند. اگر ترکیبی از عدد یا حروف الفبا بود متغیر شروع را مشخص می‌کند و تا وقتی که به انتهای کد نرسیده باشد به سمت جلو حرکت می‌کند.**

**Test.hol: در این فایل رشته ای که میخواهیم برای بررسی به کامپایلر بدهیم را تعریف می‌کنیم.**

****

**Main.cpp: در این فایل برنامه اصلی را پیاده‌سازی می‌کنیم.**

**متد :scanFileدر ابتدا تابعی برای اسکن فایل ایجاد می‌کنیم تا آدرس فایل را بگیرد و آن را اجرا کند و اگر فایلا باز نشد ارور بدهد.**

**سپس کلاس اسکنر را مدل میکنیم. و رشته ای به نام line می‌سازیم. تا وقتی که خطوط کد داخل فایل وجود داشته باشد لیست توکن ها را نگه می‌داردو با کلاس Scanner آن را اسکن می‌کند و خروجی را داخل کلاس tokens می‌ریزد و وقتی عملیات اسکن تمام شود جزئیات آن را پرینت می‌کند.**

**متد main: در این تابع با آرگومان‌های ورودی خط فرمان مسیر فایل را به برنامه می‌دهیم.**

خروجی برنامه بعد از اجرای فایل main.cpp بر اساس فایل Test.hol:

