پروژه درس کامپایلر

اعضای گروه: آتنا متولیان ، مهدی علیزاده ، سارا شیبانی

فاز اول:

بخش الف)

بخش الف)

پیاده سازی عملیات ریاضی

ابتدا توکن های مورد نیاز را به فایل lexer.h اضافه میکنیم که این توکن ها شامل mod,

power,

lt,

gt,

gteq,

lteq,

isEqual,

notEqual,

logicalAnd

logicalOr

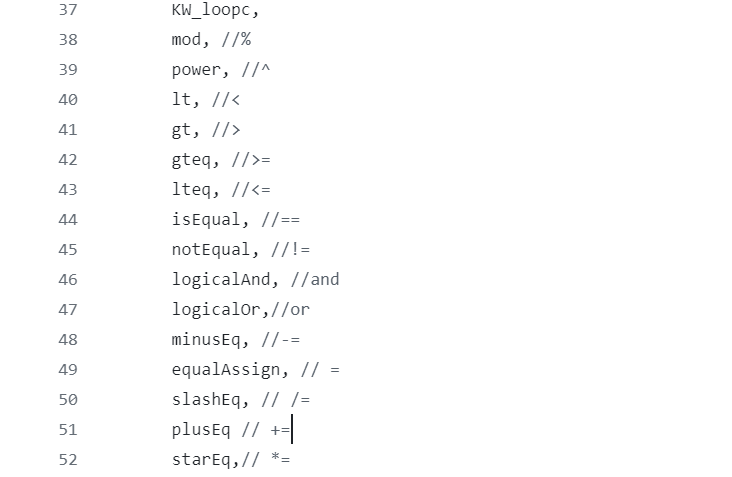
minusEq,

equalAssign

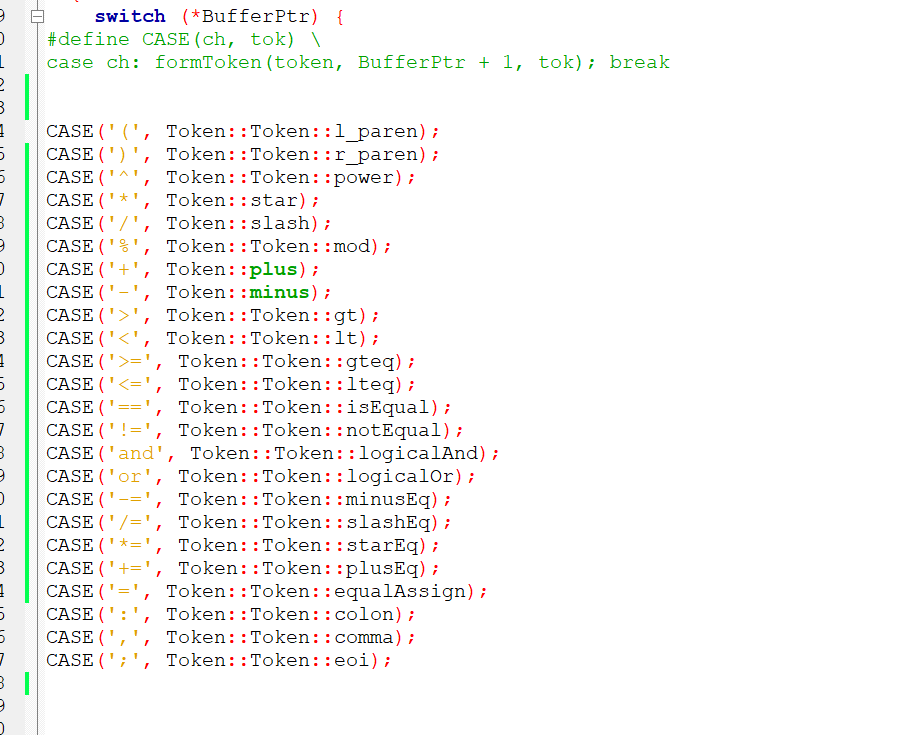
slashEq,

plusEq

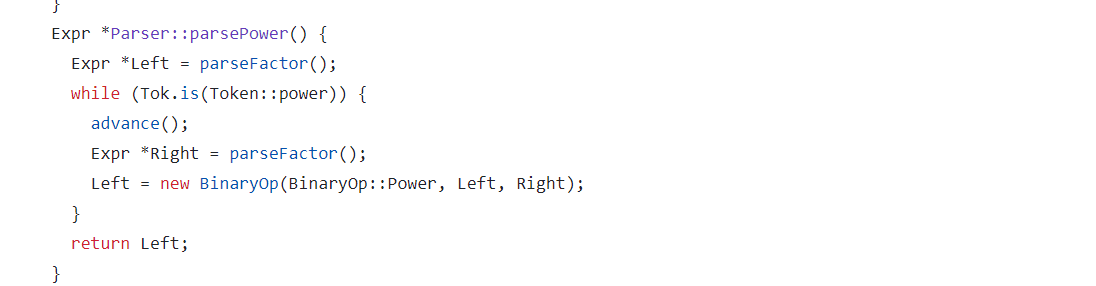
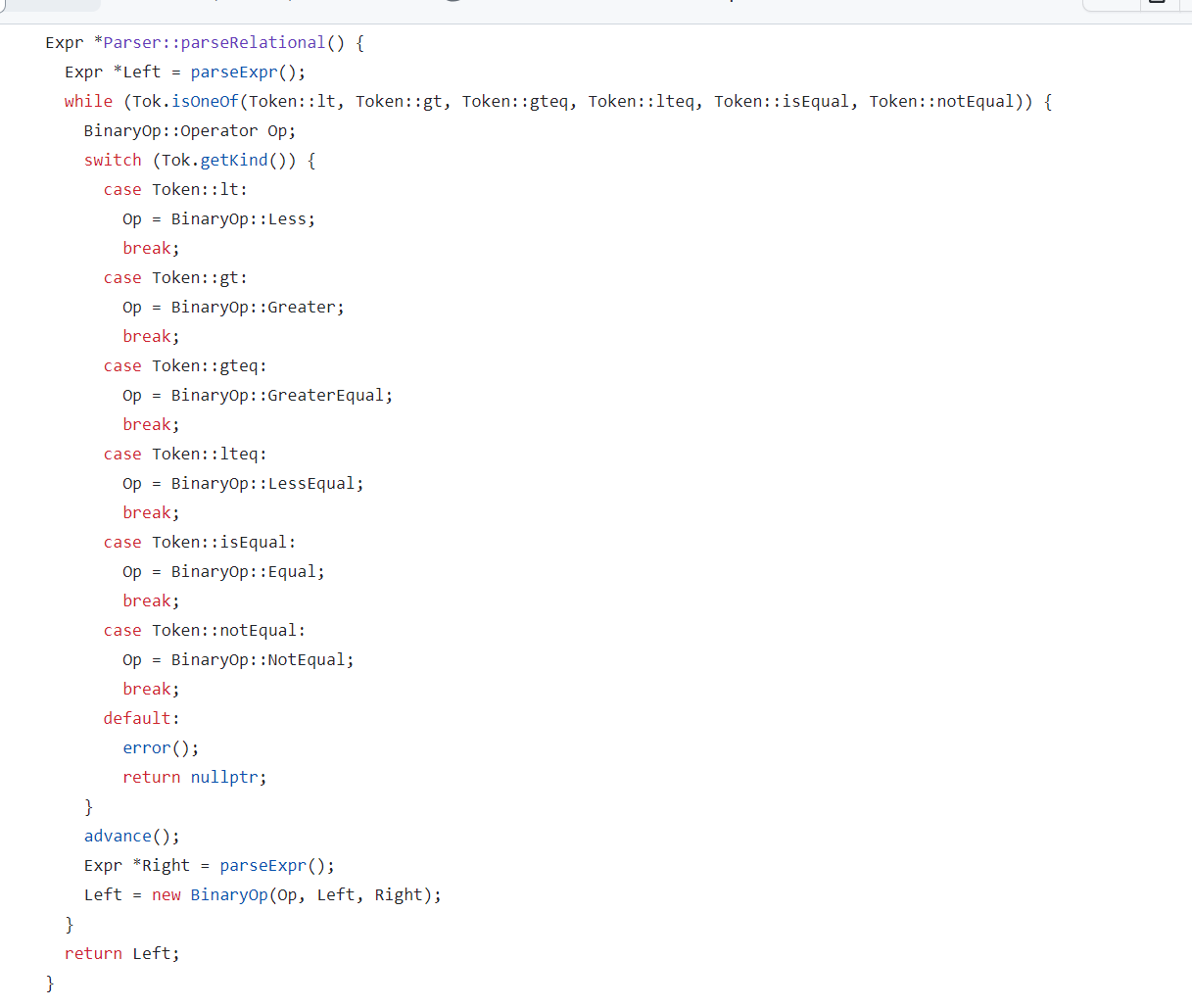
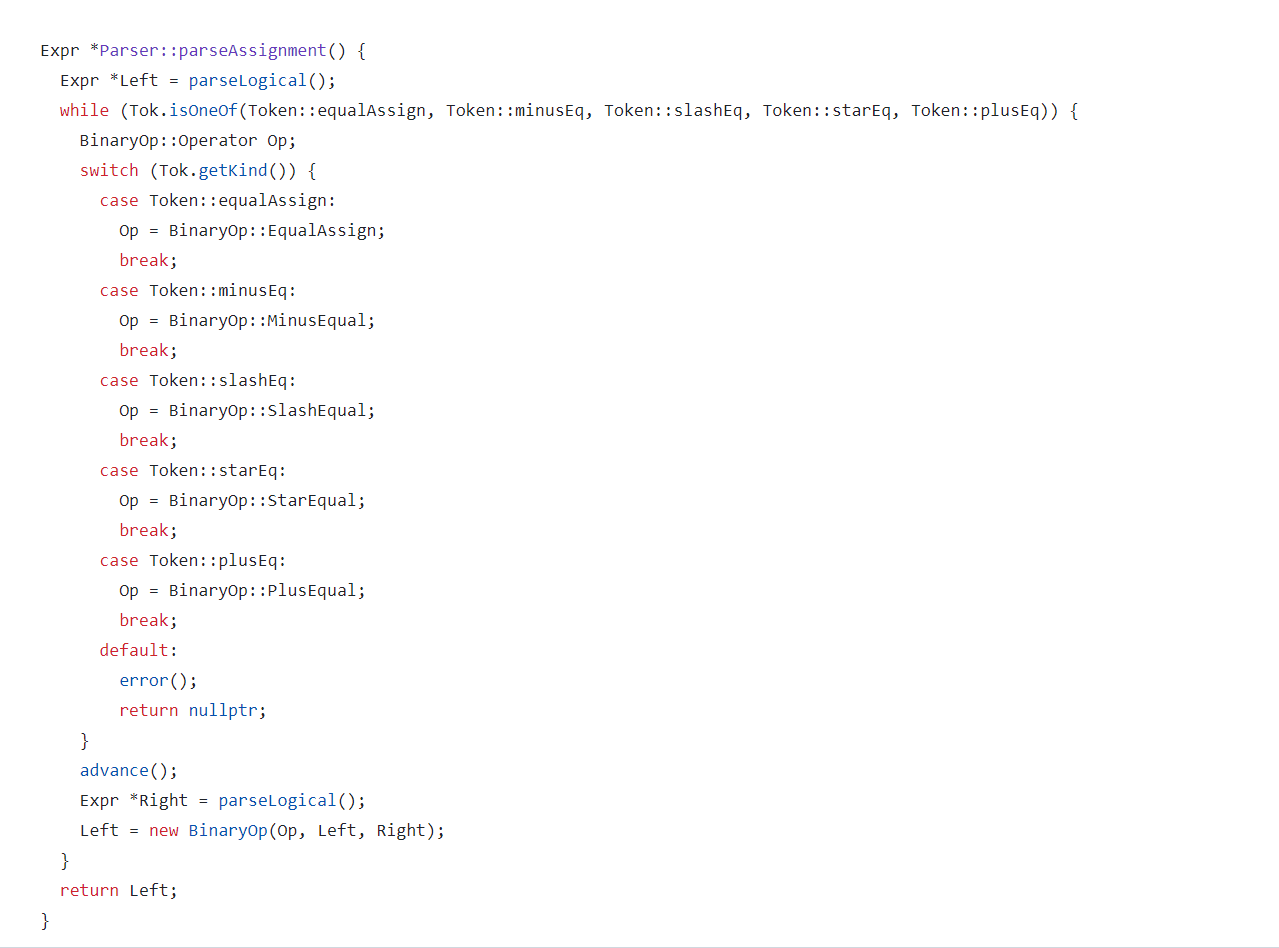
starEq, هست



سپس در فایل lexer.cpp در قسمت switch case هر کدام از عملگرهارا اضافه میکنیم.



سپس در قسمت parser.cpp برای پارس کردن assignmentها و عملگر ها توابعی را تعریف میکنیم.



A computer code with many text

Description automatically generated with medium confidenceA computer code on a white background

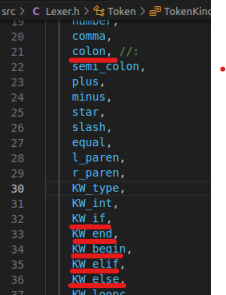
Description automatically generatedA computer code with many letters

Description automatically generated with medium confidence

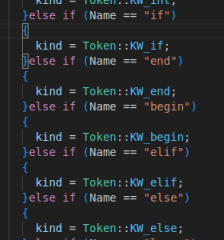
بخش ب)

* پیاده سازی شرط:

برای پیاده سازی شرط ابتدا باید توکن های مورد نیاز را به فایل lexer.h اضافه کنیم که این توکن ها شامل KW\_if ، : ، KW\_elif و KW\_else و begin و end هست.

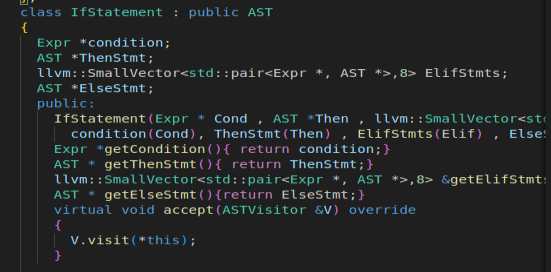


سپس در فایل lexer.cpp باید هر توکن را در تابع next() چک کنیم:



در فایل AST.h نیز باید یک کلاس ifStatement ایجاد کنیم که دارای شرط ، بدنه ، else و یک وکتور است که elif ها را ذخیره میکند.

همینطور دارای یک تابع visit است برای visit کردن نود شرط:



در تابع parseCalc باید یک شرط اضافه کنیم در صورتی که توکن با KW\_if شروع شده بود عملیات مربوط به پارسه کردن شرط رو انجام بدیم :

با استفاده از advance()، به توکن بعدی منتقل می‌شویم و if رو میخونیم.

Expr \*Condition = parseExpr(); نشان‌دهنده گرفتن شرط مربوط به if است که خود این شرط هم parse میشود.

با if(expect(Token::colon)) و advance()، بررسی می‌شود که آیا بعد از شرط، (:) وجود دارد یا نه(به بخش ارور میرود)

if(expect(Token::KW\_begin)) و advance() بررسی می‌کند که (begin) وجود دارد یا نه.

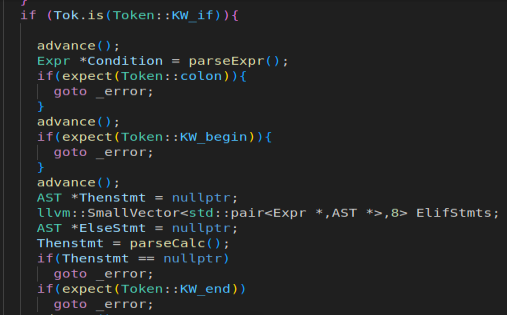
در اینجا از AST \*Thenstmt = parseCalc(); استفاده می‌شود که یک تابع دیگر برای parse کردن داخل شرط استفاده میشه .

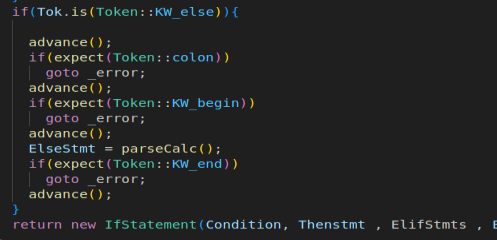
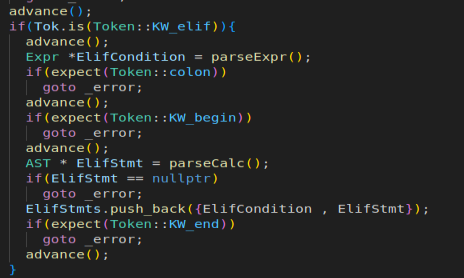
با if(expect(Token::KW\_end)) و advance()، بررسی می‌شود که آیا توکن (end) وجود دارد یا خیر.

در صورتی که Tok.is(Token::KW\_elif) باشد، بلوک‌های elif پردازش می‌شوند و اطلاعات مربوط به هر یک از آنها در llvm::SmallVector<std::pair<Expr \*,AST \*>,8> ElifStmts; ذخیره می‌شود.

اگر بلوک else وجود داشته باشد، ابتدا با advance() به توکن بعدی منتقل می‌شویم، سپس با if(expect(Token::colon)) و advance() بررسی می‌شود که بعد از else، : وجود دارد یا خیر. در ادامه با AST \*ElseStmt = parseCalc(); بلوک else ،parse میشود.

در نهایت، یک instance جدید از نوع IfStatement با استفاده ایجاد شده و برگردانده می‌شود.





در فایل codeGen.cppنیز تابع visit برای ifStatement اضافه شده، ابتدا شرط if ارزیابی شده و نتیجه آن با مقدار صفر مقایسه می‌شود. نتیجه این مقایسه در conditionResult ذخیره می‌شود.

سپس بلوک‌های LLVM متناظر با بلوک‌های if، elif و else ایجاد می‌شوند.

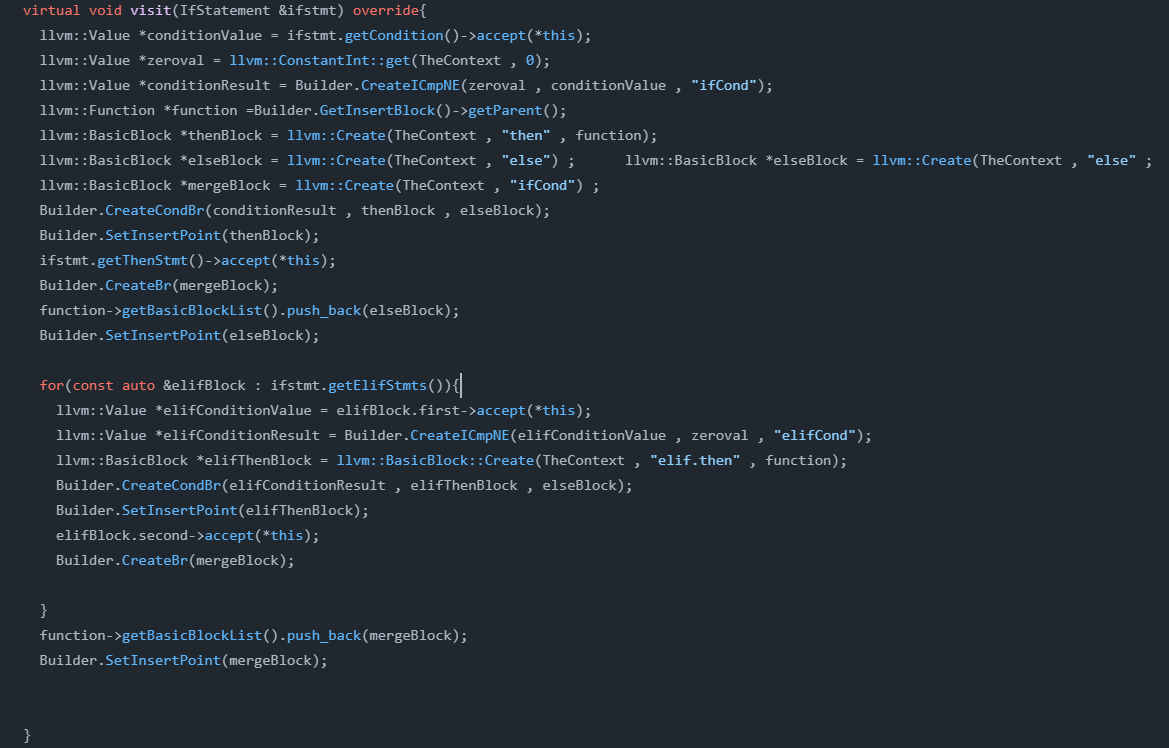
یک دستور شرطی با استفاده از Builder.CreateCondBr ایجاد می‌شود که به توجه به نتیجه شرط، به بلوک then یا else منتقل می‌شود.

در بلوک then، دستورات مربوط به تنها شرط if اجرا می‌شوند و پس از آن به بلوک ادغامی (mergeBlock) می‌رود.

در بلوک else، دستورات مربوط به بلوک else اجرا می‌شوند و نیز به بلوک ادغامی می‌رود.

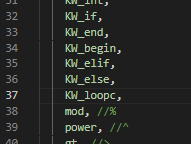
سپس بلوک‌های elif به ترتیب پردازش می‌شوند و هرکدام از آنها از یک بلوک then به بلوک ادغامی منتقل می‌شوند.

در نهایت، با ایجاد بلوک ادغامی و اتصال به آن، فرآیند تولید کد برای دستور شرطی به پایان می‌رسد.



* پیاده سازی حلقه:

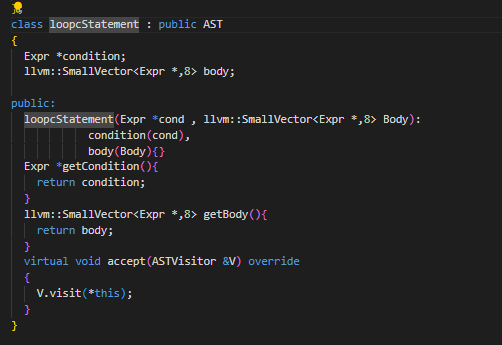
مشابه بخش if توکن های مربوطه را به Lexer اضافه می‌کنیم





در فایل AST.h کلاس loopcStatement را اضافه می‌کنیم برای ایجاد درخت پارس loop

که یک فیلد condition و body دارد که قسمت شرط حلقه و بدنه آن را ذخیره می‌کند.

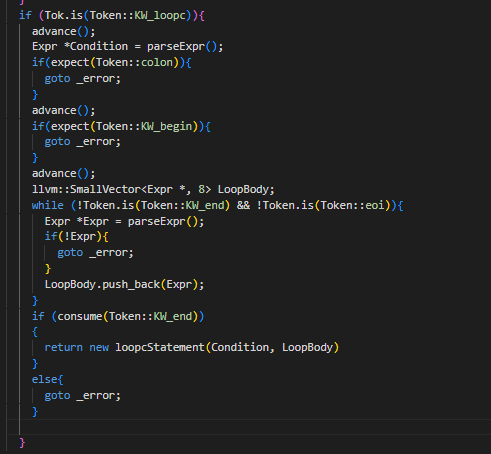


در فایل Parser.cpp و در parseCalc() یک شرط برای بررسی توکن مربوط به loop اضافه می‌کنیم

در صورتی که توکن loopc دیده شد وارد آن میشود سپس دراین شرط ابتدا عبارت روبه روی توکن حلقه گرفته میشود و بعد درصورتی که colon رو نبیند ارور می‌دهد در مرحله بعدی باید توکن begin را ببیند در غیر این صورت ارور میدهد

برای بدنه حلقه هم از یک وکتور استفاده شده که تعداد تکرار مورد نیاز حلقه را نگه میدارد و هر بار حلقه را تکرار میکند

در آخر هر دور با دیدن توکن end متوجه پایان حلقه می‌شود و بعد از آن یه نمونه جدید از loopcStatement ایجاد می‌کند



در فایل CodeGen,cpp یک تابع visit() اضافه میکنیم در این تابع سه BasicBlock برای شرط، بدنه و پایان ایجاد میکنیم نقطه ورود را loopCondBlock قرار می‌دهیم

مقدار loopConditionResult را برابر مقایسه صفر با loopCondition قرار داده می‌شود (با استفاده از CreateICmpNE) در صورتی که این دو برابر نباشد مقدار true میشود و به loopBodyBlock ، branch زده می‌شود در غیر این صورت اگر false شود به loopEndBlock میرود

