폰트, 텍스트, 그래픽, 그래픽 디자인이(가) 표시된 사진

자동 생성된 설명

Internal Document

시계, 폰트, 그래픽, 상징이(가) 표시된 사진

자동 생성된 설명

* 과목명: 프로그래밍언어론
* 담당교수님: 김진성 교수님
* 학과: 소프트웨어학부
* 보고서작성자: 20195815 이석진

20220702 서정민

# 작성한 소스코드에 대한 설명

(어느 부분에서 무엇을 작업하는지(각 함수들의 역할 설명)정도 요약) – 작성후 지우기

1. LexicalAnalyzer.h

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

charClass: 현재 문자 유형을 나타내는 변수

lexeme: 현재 어휘 분석 중인 문자열을 저장할 배열

nextChar: 다음에 분석할 문자를 저장

lexLen: 현재 lexeme의 길이

token: 현재 lexeme의 토큰 값

nextToken: 다음에 읽을 어휘 항목의 토큰 값

in\_fp: 입력 파일에 대한 파일 포인터

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

//문자 유형 상수

LETTER, DIGIT, UNKNOWN 세가지 문자 유형 상수를 정의하여 문자 판별에 사용

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

//토큰 코드 상수 – 각 토큰 유형을 구분

INT\_LIT: 정수 리터럴

IDENT: 식별자

ASSIGN\_OP: 할당 연산자 // 배정으로 고치는게 나아보이네요

ADD\_OP, SUB\_OP, MULT\_OP, DIV\_OP: 사칙연산 연산자

LEFT\_PAREN, RIGHT\_PAREN: 괄호

SEMI\_COLON: 세미콜론

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

// 함수 선언

addChar(): lexeme에 현재 문자를 추가함

getChar(): 다음 문자를 읽고 charClass를 업데이트

getNonBlank(): 공백을 건너뜀

lexical(): 어휘 분석의 핵심 함수로, 다음 어휘 항목을 식별하고 토큰을 반환

lookup(char ch): 주어진 문자를 검사하고 해당 토큰을 반환

2. SyntaxAnalyzer.h

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

OK, WARNING, ERROR 등의 상수는 구문 분석 결과를 나타냄

parseStatus는 구문 분석 과정에서의 상태를 추적하는 변수 // 삭제됨

idCount, constCount, opCount는 각각 식별자, 상수, 연산자의 개수를 추적하는 변수

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명

SymbolTable 구조체는 변수 이름, 값, 정의 여부를 저장하며, 심볼 테이블에서 사용

// 이전 SymbolTable은 이제 Symbol 구조체로 바뀌고, 현재 SymbolTable은 Symbol의 배열로 바뀌었습니다. 또한, SymbolTable 구조체의 메서드가 추가되었습니다.

// 모든 char 배열 string으로 변경되었습니다.

symTable 전역 배열과 symTableSize 변수는 전체 심볼 테이블을 관리

// SymbolTable 구조체 하나로 변경

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

// 심볼 테이블 함수

// SymbolTable 구조체 내부의 메서드로 변경되었고, 마찬가지로 전부 string으로 변경되었습니다.

createSymbol: 새로운 심볼 생성

findSymbol: 이름 기반으로 심볼을 검색

updateSymbolTable: 특정 변수에 대해 값을 업데이트

printSymbolTable: 현재 심볼 테이블을 출력

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

// 마찬가지로 전부 string으로 변경되었습니다.

// 이제 value는 int형으로, 해당 트리 노드에서의 값을 나타내고, Boolean 타입 isDefined, int status(위에서 Define 된 ERROR, OK, WARNING 값 저장), string message(status에 따라 출력되는 메시지) 변수가 추가되었습니다.

ParseTreeNode 구조체는 구문 분석 트리의 노드를 나타내며, 각 노드는 토큰, 값, 자식 및 형제 노드에 대한 포인터를 포함

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

// 파싱 트리 함수

createNode: 새로운 파싱 트리 노드를 생성

addChild: 부모 노드에 자식 노드를 추가 (자식이 이미 있으면 자식의 형제 노드로 등록)

freeTree: 트리의 루트부터 메모리를 해제

// 구문 분석 함수 - 주어진 입력을 기반으로 구문 구조를 파싱하고 트리를 형성하는 함수

program, statements, statement: 프로그램 구조와 문(statement) 수준에서 파싱을 담당

expression, term, factor, term\_tail, factor\_tail: 표현식과 그 하위 구성 요소를 파싱

// 각 tail 함수의 argument로서 int leftValue가 추가되었습니다. 이를 통해 term이나 factor 의 값을 leftValue로 전달하고, 연산자가 있을 경우 계산 한 뒤의 값을 node에 추가하여 return, 없을 경우 leftValue를 그대로 return 합니다.

evaluateParseTree: 파싱 트리의 값을 평가, 왼쪽 값(leftValue)을 초기 값으로 사용하여 계산을 수행 // 삭제됨

3. SyntaxAnalyzer.cpp

program 함수부터 아래 recursive하게 내려가며 (top-down) 끝에 도달하면 값과 오류 발생 여부를 검증합니다.

오류의 경우 가장 먼저 발생한 오류(leftmost 이므로 가장 왼쪽 아래 노드의 오류)를 먼저 출력하며, status가 OK인 경우에는 WARNING, ERROR로 덧씌어지고, OK, WARNING인 경우에는 ERROR로 덧씌어집니다.

4. LexicalAnalyzer.cpp

5. main.cpp