Project: Interpreter Implementation

Members

이름	학번	기여	기여도
서한솔	20140285	자료구조(콜스택, 히스토리, 심볼테이블 등) 구현, 파서 구현, Control flow 인터프리팅 구현, 중간 발표	55%
김현수	20140168	Prompt 인터페이스 구현, AST 인터프리팅 구현, print/trace 로직 구현	40%
백대현	20183269	스캐너(Lexer.cpp) 구현과 중간발표 후 연락 부재	5%

Execution and Test

```
> .\bin\minic.exe .\example1.c
> .\bin\minic.exe .\example2.c
> .\bin\minic.exe .\example3.c
```

Data Structures

자료구조	설명	예시
Tokens	NUM, REAL, INC, DEC, EQUAL, ..., PRINTF 각각의 object	0,1,2,.. -> NUM
ParseTree	nonterminal이름, 라인번호, value, 부모노드, 형제노드, 자식노드 들을 저장하는 트리(AST)	
SymbolTable	변수이름, 타입, assigned여부, level, scope, 히스토리, 주소들을 저장하는 리스트	
TypeObject	Type을 recursive하게 정의하기 위해 type, baseType, arraySize, parameterType, returnType을 저장하는 트리	int **a[0];
HistoryTable	trace를 수행하기위해 각 심볼마다 정의되어 라인번호와 값을 해당 변수가 수정될 때마다 기록하는 리스트	
CallStack	현재 프로시져에서 다른 프로시져로 넘어가기 위해 현재 parseTree(AST)와 level을 저장하는 스택	

우리의 구현체는 라인번호 1개 당 1 개의 statement를 가진다고 가정하고 분기문이나 프로시져를 전환할 때마다 돌아올 라인번호가 아닌 statement 위치(parseTree pointer)를 저장한다.

Implementation

Next command

1. next 에 따라붙는 번호 만큼 statement를 실행시킨다.
2. statement를 실행시킬 때 declaration을 만나면 symbol table에 기록을 해준다.
3. assignment를 만나면 등호 왼쪽 변수의 주소를 계산하고 등호 오른쪽 변수의 값을 계산하여 해당 주소에 값을 저장해준다.
4. for, if를 만나면 condition 값에 따라 다음 계산할 parseTree를 선정한다.
5. function call을 만나면 evaluation을 중단하고 가장 마지막에 계산한 값들을 각 parseTree에 저장한 뒤 현재 parseTree와 level을 callstack에 저장하고 해당 function으로 이동한다.
6. 해당 function이 끝나면 callstack에서 pop하여 이어서 실행할 parseTree를 얻는다.
7. 현재 parseTree를 다시 실행하고 만약 child tree에 이미 계산한 값(function call직전까지 계산한 값)이 있다면 그 값을 리턴하여 evaluation을 수행한다.
8. printf를 만나면 첫번째 인수를 제외한 나머지를 evaluate하고 string literal인 첫번째 인수와 나머지 인수들을 그대로 c언어 함수인 printf에 넣는다.

> recursive function을 완전히 지원하기 위해서는 parseTree에 계산이 덜 끝난 값을, function call한 프로시져마다 기억하여 저장해야한다. 하지만 시간 상의 이유로 1단계 전의 프로시져 값만 저장할 수 있다.

Print command

1. 우리는 symbol table을 관리하고 있기 때문에 어느 순간이든 현재 변수의 값을 알 수 있다.
2. print의 인자로 들어온 string에 해당하는 symbolTable entry를 가져와서 값을 리턴해준다.

> 우리의 프로그램은 Array변수를 출력할 수는 있지만 Array변수의 element를 출력하지 못한다. 이 또한 시간 상의 이유로, 만약 print의 인자로 들어온 string을 parsing만 한다면 해결할 수 있다.

Trace command

1. 우리는 심볼마다 history를 관리하고 있기 때문에 어느 순간이든 현재까지 수정된 변수값들을 확인할 수 있다.
2. trace의 인자로 들어온 string에 해당하는 symbolTable entry를 찾고 해당 entry에 들어있는 histroy table을 탐색하여 값을 리턴해 준다.

> 마찬가지로 우리의 프로그램은 Array의 history는 출력하지만 Array element의 history를 출력하지 못한다. print와 마찬가지 방법으로 해결할 수 있다.