프로그램 실행 시각화에 의한 프로그램 이해도 향상

허정수, 하상호
순천향 대학교 정보기술공학부
e-mail: majya@sch.ac.kr, hsh@sch.ac.kr

Enhancing Program Understanding by Program Execution Visualization

Jung-su Hur, Sangho Ha
School of Information Technology Engineering,
Soonchunhyang University

요 약

오늘날 컴퓨터와 네트워킹의 향상된 기술을 이용하여 학습하는 e-learning이 제공되며 앞으로 수요는 늘어날 것으로 예상된다. e-learning이 성공하기 위해서는 사용자에게 개인화된 학습 제공이 중요하며 개인화된 학습을 제공하기 위한 e-learning이 연구되고 있다. 논문에서는 프로그래밍 학습을 위한 e-learning을 고려한다. 프로그래밍의 이해를 높이려는 연구는 계속되어 왔으나 프로그램의 부분적인 이해를 높이는 연구만이 이루어지고 있다. 논문에서는 프로그램 실행의 시각화를 통해 프로그램의 전체적인 실행 과정에 대한 이해를 높여 주는 시스템을 개발한다.

1. 서론

오늘날 지식 기반 사회에서는 사람들이 효과적인 방식으로 새로운 지식과 기술을 배우는 것이 필요하다. 컴퓨터와 네트워킹의 향상된 기술을 이용하여 유연하고, 개인적이고, 이동성 있는 학습을 지원하는 e-learning[1]이 연구되어 왔다. 오늘날 수많은 학교, 회사, 기관 등에서 e-learning 형태의 학습을 지원 및 제공하고 있으며, 그 수요는 더욱 폭발할 것으로 예상된다. e-learning이 성공적이기 위해서는 학습자와의 상호작용이 중요하며, 이러한 상호작용은 개인화되어야 한다. 즉, 학습자의 수준, 학습 패턴 등 학습자 문맥에 따른 학습이 진행될 수 있게 하는 것이 필요하다. 최근에는 이러한 개인화된 학습을 제공하기 위한 e-learning 기술이 연구[2,3]되고 있다.

논문에서는 컴퓨터 프로그래밍 학습을 위한 e-learning을 고려한다. 그 동안 프로그램의 이해를 높이려는 많은 연구가 진행되었다. 초기에는 소스 코드의 제어 흐름을 순서도(flow chart) 형태로 보여주는 연구[4,5], 소스 코드를 CSD(Control Structure Diagram)로 시각화하여 보여주는 연구[6]가 진행되었다. 이러한 시각화는 프로그램에 대한 정적인 뷰이며 실행 과정에 대해서는 고려하지 않는다. 최근에는 알고리즘 애니메이션(algorithm animation)과 프로그램 시각화(program visualization)에 대한 연구가 진행되고 있다. 이러한 연구는 프로그램 실행 과정에 대한 시각화를 통해서 프로그램 이해를 향상시키는 것이 그 목적이다. 알고리즘 애니메이션은 그 대상이 주로 자료구조이며, [7,8,9]에서와 같이 자료 구조(data structure)에 대한 동적인 뷰를 다이어그램 형태로 시각화하여 보여준다. 프로그램 시각화에 대한 연구로, [10]에서는 C 프로그램 실행과정에서, 함수들간의 호출 관계를 다이어그램 형태로 보여주며, [11]에서는 Java 프로그램 실행시에 참조되는 객체와 속성을 시각화하여 보여준다.

위의 연구들은 자료 구조, 함수 호출 관계, 참조 객체 등과 같이 관심 있는 알고리즘이나 프로그램 일부분에 대한 시각화이며, 제어 구조에 대해서는 고려하지 않는다. 제어 구조는 [4,5]에서처럼 정적 뷰만 고려되었고 프로그램 실행에 따른 동적인 뷰는 고려되지 않았다. 그러나 프로그램 실행 중 제어 흐름이 어떻게 진행되는지에 대한 시각화는 프로그래밍 초보자를 대상으로 하는 프로그래밍 학습

e-learning을 위해서 필요하다.

논문에서는 제어구조에 대한 프로그램 시각화를 통해서 프로그램의 이해를 향상시키는 시스템을 개발한다. 개발된 시스템은 프로그래밍 e-learning 도구로 사용될 수 있을 것으로 본다. 제어구조 시각화 방법으로 순서도를 사용한다. 즉, 프로그램이 실행되는 과정을 순서도를 통해서 단계적으로 시각화하여 보여준다. 따라서 프로그램 실행과 함께 순서도는 동적으로 생성되고, 커지고 줄어든다. 또한 프로그램 변수의 상태 변화를 순서도와 함께 시각화하여 보여준다.

논문에서는 C 프로그램에 대한 e-learning을 고려한다. 시스템은 컴파일러, 인터프리터, 시각화의 크게 3가지 모듈로 구성된다. 컴파일러는 사용자가 입력한 프로그램을 중간코드로 번역하는데, 이 중간코드는 시각화 정보를 포함하도록 정의된다. 인터프리터가 중간 코드를 해석하면서, 실행된 중간 코드와 실행 결과를 시각화 모듈에 전달한다. 이 모듈은 전달받은 정보를 토대로 실행된 중간코드를 적절하게 시각화시킨다. 시스템은 Java로 개발되며, Java를 이용함으로써 어떠한 시스템에서도 이용 가능한 이점을 갖는다.

2. 시스템 설계

시스템의 전체 구조

C언어 해석기는 사용자에게 사용자 프로그램 입력, 실행하기, 실행 결과 보기, 실행 과정 보기라는 네 가지 기능을 사용자 인터페이스로 제공한다. 프로그램 입력은 사용자가 C언어 프로그램을 입력할 수 있는 기능을 제공한다. 실행 기능은 입력한 프로그램을 실행하는 기능이다. 실행은 컴파일과 중간코드 실행이라는 두 개의 기능을 포함하고 있다. 컴파일은 중간코드와 EAST 생성하는 기능으로 확장되며 중간 코드는 인터프리터를 이용하여 실행하고 EAST는 시각화기를 통해 사용자에게 보이게 된다. 중간코드 실행을 컴파일에서부터 생성된 중간코드를 이용한다. 명령어 실행은 실행 번호를 포함하고 있다. 컴파일 기능으로부터 생성된 EAST와 중간코드의 줄번호를 이용해서 시각화한다. 시각화 하는 화면은 실행과정 기능을 통해 사용자에게 보이게 된다. 그림1은 전체적인 시스템의 구조를 나타내고 있다. 시스템 설계를 위한 UML의 Use Case 다이어그램은 그림 2와 같다.

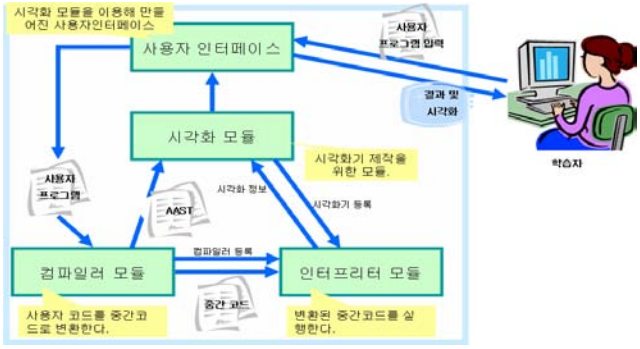

그림 1. 전체 시스템 구조도

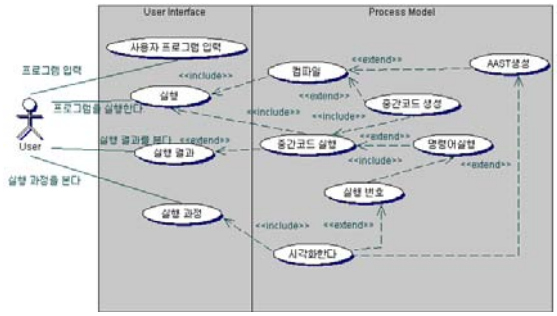

그림 2. Use Case 다이어그램

컴파일러

컴파일러는 사용자 프로그램을 받아 인터프리터가 실행할 수 있는 중간코드를 생성하며 사용자 프로그램의 구문에 맞춰 시각화 정보를 담고 있는 트리 구조인 EAST를 생성한다. C언어 BNF문을 이용하여 어휘 분석기인 JFlex[4]와 구문 분석기인 CUP[5]를 이용하여 제작된다. C언어 구문별로 알맞게 중간 코드를 배치하고 EAST 노드를 만들어서 조합하는 역할을 한다. 시스템에서는 인터프리터에서의 컴파일러 기본 명세를 이용하여 제작되며 인터프리터에 등록되어 인터프리터의 관리를 받는다.

중간 코드

중간 코드는 번호와 명령어, 피연산자로 이루어진다. 번호는 사용자 프로그램의 열 번호로 이루어지며 앞으로 설명할 EAST를 이용해 시각화할 시에 실행 위치를 구분하기 위해 사용되며 각 명령어는 실행 구문에 따른 고유의 번호를 갖게 된다. 명령어는 중간코드 내의 부프로그램을 위한 PROC, RET, CALL이 있으며 인터프리터 내에서 시스템 스택에 데이터의 이동을 담당하는 PUSH, POP, MOV가 있다. 비교연산을 위한 CLT, CGT, CLE, CGE, CNE, CEQ가 있으며, 실행 위치 변경을 위한 라벨설정인 콜론과 JF, JT, JMP가 있다. 산술연산을 위한 ADD, SUB, MULT, DIV, REMN과 비트연산을 위한 AND, OR, XOR, NOT를 제공하고 변수의 메모리 주소값을 알기 위한 ADDR을 제공한다. 표1은 전체 명령어표이다.

명령어			
PROC	CLE	JT	REMN
PUSH	CGE	JMP	AND
POP	CLT	ADD	OR
:	CGT	SUB	XOR
MOV	CEQ	MULT	NOT
RET	CNE	DIV	ADDR
CALL	JF		

표 1. 명령어 집합과 명령어 구문

EAST

Extended Abstract Syntax Tree의 약자로써 C 언어로 된 프로그램을 이용할 수 있는 정보의 단위를 계층적인 트리로 나타낼 수 있는 자료구조이다. 컴파일러의 구문 분석에 의해 생성된 EAST는 시각화기를 이용해 사용자에게 시각적인 정보로 표현된다. EAST의 Node는 노드 형태, 번호, 내용, 자식노드로 구성된다. 노드 형태는 C언어 구문에 기초한 C언어 구문에 대한 형태를 나타내는 상수로 이루어져 있다. 노드의 형태에 따라 자식노드의 개수와 시각적인 표현 방법을 알 수 있다. 노드의 번호는 중간 코드의 번호와 같은 번호를 이용하며 구문을 구분하는데 사용된다. 내용은 하위 자식노드를 조합하여 사용자 프로그램의 블록 단위의 내용을 스트링으로 저장된다. 자식노드는 EAST Node를 갖게 된다.

인터프리터

컴파일러로부터 사용자 코드에 대해서 생성된 중간코드를 실행하는 모듈이다. 인터프리터는 본 논문의 시스템의 중추이며 컴파일러와 시각화기를 등록함으로써 두 모듈을 관리한다. 중간 코드를 줄 단위로 실행하며 실행되는 정보를 시각화기에 보내서 시각화를 유도한다. 시각화기에 보내는 정보는 열 번호, 명령어, 피연산자가 있다. 정의된 컴파일러와 시각화기 인터페이스를 이용하여 컴파일러와 시각화기를 제작하면 다른 언어나 다른 시각화 시스템도 적용할 수 있다.

시각화기

컴파일러로부터 EAST를 받아 저장하며 명령어를 실행할 때마다 인터프리터로부터 오는 정보를 이용해 사용자에게 시각화하여 보여준다. EAST를 관리하고 EAST 노드의 형태에 따라 알맞은 시각화 방법을 이용해 사용자에게 보여주며 현재 실행되는 중간 코드 명령어를 사용자에게는 사용자 프로그램에서의 실행 위치로 보여준다. 시각화기 모듈은 컴파일러 모듈과 같이 인터프리터에 등록되며 관리를 받게 된다.

3. 적용

설계된 시스템을 이용한 시나리오에 의한 적용을 보면 그림 3과 같다.

사용자로부터 프로그램이 입력되면 컴파일러에 의해 중간코드와 EAST가 만들어진다. 시나리오에 사용되는 사용자 코드는 C언어 프로그램이며 max 함수를 이용해 두 개의 수를 비교해서 큰 수를 화면에 출력하는 프로그램이다. 이 프로그램과 컴파일러에 의해 변환된 중간코드는 그림 4와 같으며 컴파일러에 의해 생성된 EAST는 그림 5에 내용과 번호로 간략하게 나타내고 있다.

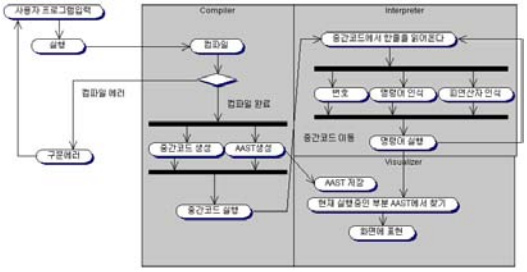

그림 3 . 실행 시나리오

```
int max( int a, int b ){
        if ( a > b )
                return a;
        return b;
}

void main(){
        printf( max( 5, 6 ) );
}
```

```
-1   CALL   main
0    PROC   max
10   POP    a
17   POP    b
33   PUSH   a
37   PUSH   b
35   CGT
28   JF     ENDIF_0
50   PUSH   a
43   RET
-1   :ENDIF_0
61   PUSH   b
54   RET
72   PROC   main
95   PUSH   6
93   PUSH   5
88   CALL   max
80   CALL   printf
-1   RET
```

그림 4. 사용자 프로그램과 중간 코드

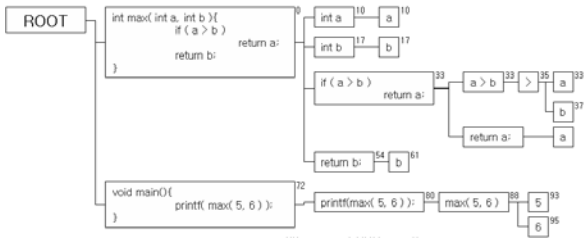

그림 5 . 사용자 프로그램의 EAST

중간 코드에서 -1은 시각화와는 관련이 없는 명령어들로 프로그램의 끝을 나타내는 RET와 라벨 명령어에는 -1이 들어가며 전체 프로그램의 시작인 CALL main명령어도 -1로 되어있다. 번호는 사용자 프로그램의 문자별 열 번호로 구성되어 있다. EAST는 컴파일러에 의해 사용자 프로그램이 파싱되고 중간 코드에서 명령어가 생성되지 않는 구문은 자식노드 중에 가장 작은 번호를 갖게 되어 있다.

컴파일러에 의해 중간코드와 EAST가 생성되면 컴파일러는 시각화기와 인터프리터에게 각각 EAST와 중간코드를 준다. 중간 코드를 받은 인터프리터는 중간코드의 첫줄부터 실행한다. “-1 CALL main”이 들어오면 부프로그램 중에 main을 찾아간다. 부프로그램 main을 찾으면 인터프리터는 main

의 번호인 72번을 시각화기에 보낸다. 시각화기는 EAST의 ROOT의 자식 노드 중에 72번을 찾아 화면에 그림 6과 같이 표현한다.

main을 실행하면서 95번 “PUSH 6″으로 스택에 6을 넣고 시각화기에 보낸다. 시각화기는 PUSH명령은 시각화하지 않기 때문에 그냥 넘어간다. 93번, 역시 PUSH명령이므로 시각화하지 않고 스택에 5를 저장한다. “88 CALL max”가 실행되면 인터프리터는 부프로그램 max을 찾아낸다. max의 번호인 0번을 시각화기에 보내주면 시각화기는 현재 표현하고 있는 노드에 벗어나는 번호이므로 현재 표현되고 있는 72번 노드를 시각화 스택에 저장한 뒤에 0번 노드를 화면에 표시할 준비를 한다. 사용자에게 보여주기 위해 시각화기는 시각화되는 화면을 다시 그리는데 시각화 스택에 있는 노드들을 밑바탕에 그리고 현재 표현되어야 하는 0번 노드를 최상위 화면에 그리고 매개변수인 10번과 17번을 오른쪽 상단에 표현한다. 그림 7과 같이 표현된다.

부프로그램 max를 실행하면 “10 POP a”가 실행이 된다. 인터프리터는 스택에서 값을 가져와 a에 저장을 하고 시각화기에 “10 POP a”를 넘겨주면 시각화기는 10번의 위치에 a의 값을 표시한다. “17 POP b” 명령도 인터프리터로 실행되고 시각화기에서 b의 값이 표시된다. 값이 표시된 화면은 그림 8과 같다.

이러한 행동을 반복하여 사용자에게 실행하는 과정을 보이고 사용자가 원하는 부분을 시각화하여 보인다.

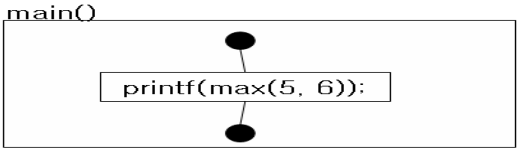

그림 6 . 시각적으로 표현된 EAST 노드

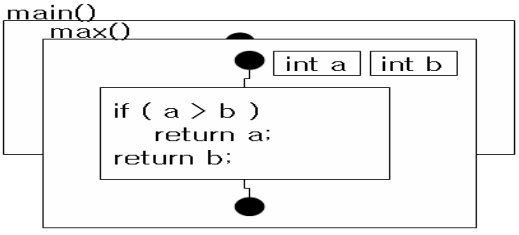

그림 7 . max함수의 실행 시각화 화면

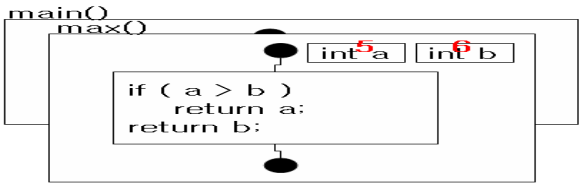

그림 8 . 매개변수에 입력 값이 표현된 화면

4. 결론

본 논문에서는 사용자 프로그램을 실행하면서 실행 과정을 시각적으로 보이기 위한 시스템을 제안, 설계하였으며 적용을 보인다. 전체 시스템을 인터프리터, 컴파일러, 시각화 모듈과 시각화 모듈에 의해 작성된 사용자 인터페이스로 구성된다. 컴파일러는 중간코드와 EAST를 사용자 프로그램으로부터 생성하며 인터프리터는 중간코드를 실행하고 시각화기는 EAST를 기초하여 사용자에게 실행되는 과정을 보인다. 전체 시스템은 시스템 구성과 적용을 간단하게 하기 위해 C언어의 구문을 일부분만 수용하고 있어 전체적인 C언어의 실행은 부족하다.

향후 과제로는 현재 구성된 시스템을 구현하는 것이다. 또한 구현을 마친 후에 확장하여 부족한 C언어 구문을 늘려 모든 C언어 구문을 받아들일 수 있는 시스템을 목표로 확장해 나아갈 것이다.

참고문헌

[1] Shrea, R.H. "E-learning today-As an industry shakes out, the survivors offer no-frills education for grown-ups: U.S. News & World Report, Oct. 28, 2002.
[2] Aleksander Binemann-Zdanowicz, "SiteLang::Edu -Towards a Context-Driven E-Learning Content Utilization Model", ACM Symposium on Applied Computing, 2004.
[3] "D. Zhang, et al., Can E-Learning Replace Classroom Learning?", CACM, Vol. 47, No.5, May 2004.
[4] L.H. Haibt, "A Program to Draw Multi-Level Flow Charts". In Proceedings of the Western Joint Computer Conference, 1959.
[5] D.E. Knuth, "Computer-DrawnFlowcharts", CACM No.6, 1963.
[6] Dr. James H.Cross II, "GRASP/Ada95: Visualization with Control Structure Diagrams", 1995
[7] DynaLab, http://www.cs.montana.edu/~dynalab/
[8] Swan, http://simon.cs.vt.edu/Swan/Swan.html
[9] JHAVE, http://csf11.acs.uwosh.edu/
[10] AiCall, http://www.aicall.de
[11] John Hamer "Alightweight Visualizer for Java", Third Program Visualization Workshop, 2004