2023학년도 2학기 중간과제물(온라인 제출용)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **교과목명** | : | 프로그래밍언어론 |
|  | **학번** | : | 202234-153799 |
|  | **성명** | : | 한승환 |
|  | **연락처** | : | 010-2862-0200 |

EMB000023580f65 ※ A4용지 편집 사용

1. 1번문제
   1. 메모리에 적재된 프로그램의 기계어 명령어 하나를 컴퓨터의 CPU가 수행하는 과정

컴퓨터를 구성하는 장치에는 많은 것들이 있지만, 그 중 데이터를 처리하는 장치인 CPU와 명령어들의 집합을 저장하고 있는 각종 저장장치들이 BUS라는 데이터 전송로로 의해 연결되어 있다. 이렇게 메모리에 적재되어 있는 기계어 명령어는 CPU로 전송되고, CPU로 읽어 들인 명령어는 해석하여 실행된다. 이러한 처리 주기를 인출-해석-실행 주기로 불리며, 이 주기를 반복하여 각각의 명령어를 수행한다.

* 1. 고급 프로그래밍 언어로 작성된 프로그램을 컴퓨터의 CPU가 수행하기 위해 필요한 프로그래밍 언어 구현 방법 두가지

1. 인터프리터
2. 컴파일러
   1. 컴파일러를 이용한 프로그래밍 언어 구현 방법

프로그래머가 작성한 프로그램을 프로그램 실행과 동시에 한 줄씩 해석하여 명령어를 수행하는 인터프리터 방식과는 다르게, 컴파일러 방식은 고급언어로 작성된 소스코드 파일을 CPU가 이해하고 수행할 수 있도록 미리 프로그램 전체를 기계어로 번역한 뒤 그것을 목적 코드 파일로 생성한다. 이렇게 기계어로 미리 번역된 목적 코드 파일은 그대로 CPU에 의해 실행되며 프로그램 실행 중간에 번역될 필요가 없다. 이에 따라 소스코드 없이도 CPU가 프로그램을 목적 코드에 적재된 기계어 명령의 순서에 따라 절차적으로 수행할 수 있다. 그렇기 때문에 컴파일러 방식은 프로그램을 인터프리터 방식으로 했을 때 매 수행마다 해석을 해야 하는 비용을 줄일 수 있으므로, 프로그램을 더 효율적으로 실행할 수 있다. 또한 회사의 입장에서는 소스코드를 공개하지 않고 프로그램을 판매할 수 있기 때문에 기술보호의 차원에서 더 합리적인 프로그래밍 구현 방법의 선택지가 될 수 있다.

1. 2번문제
   1. BNF로 표현된 문장을 EBNF로 합쳐 표현

BNF 표현:

<do~while문> : : = do { <문장들> } while <논리식> ;

<문장들> : : = <문장> <문장들> | <문장>

<논리식> : : = True | False

EBNF 표현:

<do~while문> : : = do { <문장> { < 문장 > } } while ( True | False ) ;

EBNF 표현에 대한 설명 : 우선 BNF 표현에서 <문장들>이라는 표현은 <문장>이 1번이상 반복되어 <문장들>을 정의 되도록 하고 있다. 즉, 1번이상 <문장>을 사용하여 do 뒤에오는 중괄호 안에서 <문장들>을 작성할 수 있다. 이를 EBNF 표현으로 나타내면 <문장> { <문장> }으로 나타낼 수 있는데, 이렇게 되면 <문장>을 한 번 사용하고 중괄호 안의 <문장>은 메타 기호 ‘{ }’가 의미하듯 0번이상 반복하여 사용할 수 있다. 따라서 <문장>을 한번 이상 원하는 만큼 반복하여 사용할 수 있도록 유도하고 있다. 그리고 BNF 표현에서 <논리식>은 ‘|’라는 메타 기호에 의해서 True 또는 False를 <논리식>으로 정의하고 있는데, 이것은 괄호 안쪽 ‘|’의 양쪽의 어느 하나를 선택하여 사용할 수 있게 하는 EBNF의 ‘( )’를 사용하여 EBNF 표현으로 대체 할 수 있다. ‘( )’안의 ‘|’가 양쪽으로 분리하는 True 혹은 False 중 하나를 사용하여 while 뒤에 BNF의 <논리식>을 EBNF 표현인 (True | False)로 대체할 수 있다.

* 1. 2.1의 EBNF로 표현된 문장을 구문도표로 작성

구문도표

A diagram of a diagram

Description automatically generated

설명 : 2.1의 EBNF구문도표에서는 do, {, }, while, True, ;의 단말 기호와 ‘문장’이라는 비단말 기호가 있다. 먼저 구문도표의 화살표를 따라가보면 do~while문의 정의에서 ‘do’라는 단말 기호가 먼저 사용되고 그 다음 다른 단말 기호인 중괄호 ‘{‘가 사용된다. 그리고 ‘문장’이라는 비단말 기호가 나타나는데 오른쪽으로 같은 ‘문장’이라는 비단말 기호가 연속으로 사용되고, 이것은 화살표를 이용해 되돌아 가도록 연결되기 때문에, 반복하여 ‘문장’을 사용할 수 있는 것을 알 수 있다. 그리고 중괄호 ‘}’가 단말기호로써 다시 사용되며 그 다음은 ‘while’역시 단말 기호로 사용된다. ‘while’ 다음으로는 화살표가 두 갈래로 나뉘어서 다른 단말 기호 ‘True’와 ‘False’로 향하는데 둘 중 하나를 선택하여 표현을 이어 나간다. 그리고 마지막 단말 기호인 ‘;’를 나타내어 do~while문의 구문 정의를 끝마친다.