Asad Ranavaya

CSCI-GA-2110-1-001

Prof. Goldberg

24 October 2021

Homework 1

1. Regex Expressions
   1. ([a-z])\*(([A-Z]([a-z])\*[0-9]([a-z])\*)|([0-9]([a-z])\*[A-Z]([a-z])\*))\*
   2. (-|ε)[0-9]\*.[0-9]\*E(-|ε)[0-9][0-9]\*
   3. [a-zA-Z]([a-zA-Z0-9](\_| ε))\*
2. Context Free Grammar

PROG -> FUNS

FUNS -> FUN | FUN FUNS

FUN -> fun DEFS

DEFS -> DEF | DEF “|” DEFS

DEF -> name PARS “=” EXPRS

PARS -> CONST | CONST PARS | ɛ

EXPRS -> EXP | EXP OPERAND EXP | LET

EXP -> CONST | EXP “(”EXPRS”)” | EXP EXPRS

LET = let VALFUNS in EXPRS end

VALFUNS -> VALFUN | VALFUN VALFUNS

VALFUN -> val DEF | FUN

OPERAND -> + | - | \* | /

CONST -> name | number

Parse Tree:

Image at bottom of homework.

1. Scoping
   1. In static scoping, a variable always refers to the inner-most scope where it was defined. However, in dynamic scoping, a variable refers to the calling functions scope to find a definition. In other words, a variable is defined in the order of the call stack in dynamic scoping.
   2. In this short c++ code, static scoping yields the value of x to be 3, whereas if it were dynamically scoped x would be 5.

#include <iostream>

int x = 3;

int a() {

return x;

}

int b() {

int x = 5;

return a();

}

int main()

{

std::cout << "Value of x is: " << b() <<std::endl;

}

* 1. One of the primary advantages of static scoping is that it is much easier for a programmer to read and understand what the output of a result would be. In dynamic scoping the programmer has to understand the call structure of the code to accurately predict what the value of a variable and function would be.

1. Call stack for the program defined is:

B

Calls C

Int x =20

C

Calls F with parameter x =20

E

Prints z = 20 and x = 17

F

Calls E with parameter y = 20

A

Calls B with parameter E

Int x = 17

Dynamic Link

Static Link

D | CP | EP

* 1. The program prints z = 20 and x = 17.
  2. If the program were dynamically scoped it would print z = 20 and x = 20.

1. Parameter passing
   1. Pass by value would print: 1,2,3,4,5,6,7,8,9,10
   2. Pass by reference would print: 1,45,3,4,5,6,7,8,9,10
   3. Pass by value result would print: 1,15,3,4,5,6,7,8,9,10
   4. Pass by name would print: 1,2,3,4,5,6,7,8,45,10
2. Ada Program that prints 1-100 in order with pairs being consecutively printed:
3. with Text\_Io;
4. use Text\_Io;
5. procedure objmain is
6. I : Integer;
7. *--Directive Space--*
8. package Int\_Io is new integer\_io(integer);
9. use Int\_Io;
11. procedure PrintProc is
12. task Task1;
13. task Task2;
14. *--Task Bodies--*
15. task body Task1 is
16. begin
17. Put(I);
18. *--Put(" Task 1");*
19. New\_Line;
20. end Task1;
21. task body Task2 is
22. begin
23. Put(I+1);
24. *--Put(" Task 2");*
25. New\_Line;
26. end Task2;
27. *--End Task bodies--*
28. begin
29. null;
30. end PrintProc;
31. *--End Directive Space--*
32. begin
33. I := 1;
34. for J in 1..50 loop
35. PrintProc;
36. I := I+2;
37. end loop;
38. end objmain;

A picture containing shape

Description automatically generated A picture containing chart

Description automatically generatedA picture containing diagram

Description automatically generated

Proof of concurrency above.

Parse Tree:

Diagram

Description automatically generated

Second part of parse Tree:

A picture containing flock

Description automatically generated