|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**Лабораторная работа № 2**

**«Синтаксические деревья»**

***по курсу «Конструирование компиляторов»***

Студент *Ионов Т.Р. 61Б*

Преподаватель *Коновалов А.B.*

*Москва, 2022 г.*

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc106224508)

[2 Индивидуальное задание 3](#_Toc106224509)

[3 Реализация 3](#_Toc106224510)

[4 Тестирование 4](#_Toc106224511)

[Вывод 4](#_Toc106224512)

# 1 Цель работы

Целью данной работы является изучение представления синтаксических деревьев в памяти компилятора и приобретение навыков преобразования синтаксических деревьев.

# 2 Индивидуальное задание

Подсчитать число присваиваний в процессе выполнения программы.

3 Реализация

Листинг 1 − функция countAssign.

func countAssign(file \*ast.File) {

ast.Inspect(file, func(node ast.Node) bool {

if file, ok := node.(\*ast.File); ok {

file.Decls = append([]ast.Decl{

&ast.GenDecl{

Doc: nil,

Tok: token.VAR,

Specs: []ast.Spec{

&ast.ValueSpec{

Names: []\*ast.Ident{

&ast.Ident{Name: "counter"},

},

Values: []ast.Expr{

&ast.BasicLit{Kind: token.INT, Value: "0"},

},

},

},

},

}, file.Decls...)

} else if blockStmt, block\_ok := node.(\*ast.BlockStmt); block\_ok {

block\_len := len(blockStmt.List)

for i := block\_len-1; i >= 0; i-- {

stmt := blockStmt.List[i]

if assign, assign\_ok := stmt.(\*ast.AssignStmt); assign\_ok {

if left, ok := assign.Lhs[0].(\*ast.Ident); ok && left.Name != "counter" {

block := append(blockStmt.List, nil)

copy(block[i+1:], block[i:])

block[i+1] = &ast.IncDecStmt{

X: &ast.Ident{ Name: "counter"},

TokPos: assign.End(),

Tok: token.INC,

}

blockStmt.List = block

}

}

}

} else if funcDecl, ok := node.(\*ast.FuncDecl); ok {

if funcDecl.Name.Name == "main" {

funcDecl.Body.List = append(funcDecl.Body.List, &ast.ExprStmt{

X: &ast.CallExpr{

Fun: &ast.SelectorExpr{

X: ast.NewIdent("fmt"),

Sel: ast.NewIdent("Printf"),

},

Args: []ast.Expr{

&ast.BasicLit{

Kind: token.IDENT,

Value: "counter",

},

},

},

})

}

}

return true

})

}

# 4 Тестирование

Вход:

func main() {

d := 10

for 1 == 1 {

d = 10

}

c := 10

}

func a() {

a := 10

}

Результат:

var counter = 0

func main() {

d := 10

counter++

for 1 == 1 {

d = 10

counter++

}

c := 10

counter++

fmt.Printf(counter)

}

func a() {

a := 10

counter++

}

# Вывод

В результате выполнения данной лабораторной работы были изучены представления синтаксических деревьев в памяти компилятора и приобретены навыков преобразования синтаксических деревьев.