УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3

по предмету

Компиляторные технологии

Вариант 12

Выполнил:

Крутько А. А.

Проверила:

Шостак Е. В.

Группа 251004

Минск 2023

Задание:

Разработать программное средство, проверяющее исходный код программы на соответствие грамматике (Rust).

Лексический анализатор:

%option main

%%

[ ] {

printf("0\n");

}

fn {

printf("1\n");

}

main {

printf("2\n");

}

[{] {

printf("3\n");

}

[}] {

printf("4\n");

}

[(] {

printf("5\n");

}

[)] {

printf("6\n");

}

println! {

printf("7\n");

}

\".\*\" {

printf("8\n");

}

\n|\r\n {

printf("9\n");

}

; {

printf("10\n");

}

[0-9]{1,10} {

printf("11\n");

}

if {

printf("12\n");

}

else {

printf("13\n");

}

loop {

printf("14\n");

}

%%

**Грамматика языка:**

Main –> checkMainStart takeOperator\* }

checkMainStart –> Fn Main ( ) {

takeOperator –> printOperator+ | operatorIf | operatorLoop

printOperator –> Println!(“(.)\*”);

operatorIf –> oper\_If1 | oper\_If2 | oper\_If3 | oper\_If4 | oper\_If5 | oper\_If6

oper\_If1 –> if ( int ) takeOperator

oper\_If2 –> if ( int ) takeOperator else takeOperator

oper\_If3 –> if ( int ) { takeOperator\* }

oper\_If4 –> if ( int ) { takeOperator\* } else takeOperator

oper\_If5 –> if ( int ) takeOperator else { takeOperator\* }

oper\_If6 –> if ( int ) { takeOperator\* } else { takeOperator\* }

operatorLoop –> oper\_Loop1 | operLoop2

oper\_Loop1 –> loop takeOperator

oper\_Loop2 –> loop { takeOperator\* }

**Код программы:**

#include <fstream>  
#include <vector>  
#include <string>  
#include <iostream>  
  
typedef enum TTypes {  
 SPACE = 0,  
 FN,  
 MAIN,  
 LEFTFIGURE,  
 RIGHTFIGURE,  
 LEFTBRACKET,  
 RIGHTBRACKET,  
 PRINTLN,  
 STRINGCONST,  
 EOL,  
 SEMICOLON, //;  
 INT,  
 IF,  
 ELSE,  
 LOOP,  
 ERROR  
} Types;  
  
bool printOperator();  
bool operatorIf\_1();  
bool operatorIf\_2();  
bool operatorIf\_3();  
bool operatorIf\_4();  
bool operatorIf\_5();  
bool operatorIf\_6();  
bool operatorIf();  
bool operatorLoop();  
bool operatorLoop\_1();  
bool operatorLoop\_2();  
bool takeOperator();  
bool StartLoopForOperator();  
  
std::vector<Types> terms;  
int next = 0;  
  
bool checkTerm(const Types expected) {  
 return (expected == terms[next++]);  
}  
  
bool peekTerm(const Types expected) {  
 return (expected == terms[next]);  
}  
  
bool checkMainStart() {  
 bool result = true;  
 if (!checkTerm(FN)) result = false;  
 if (!checkTerm(MAIN)) result = false;  
 if (!checkTerm(LEFTBRACKET)) result = false;  
 if (!checkTerm(RIGHTBRACKET)) result = false;  
 if (!checkTerm(LEFTFIGURE)) result = false;  
 return result;  
}  
  
bool funcMainBlock() {  
 if (!checkMainStart()) return false;  
 bool result = StartLoopForOperator();  
 if (!checkTerm(RIGHTFIGURE)) result = false;  
 if (next != terms.size()) result = false;  
 return result;  
}  
  
bool printOperator() {  
 bool result = true;  
 if (!checkTerm(PRINTLN)) result = false;  
 if (!checkTerm(LEFTBRACKET)) result = false;  
 if (!checkTerm(STRINGCONST)) result = false;  
 if (!checkTerm(RIGHTBRACKET)) result = false;  
 if (!checkTerm(SEMICOLON)) result = false;  
 return result;  
}  
  
bool operatorLoop() {  
 int save = next;  
 if (operatorLoop\_1()) return true;  
 next = save;  
 if (operatorLoop\_2()) return true;  
  
 return false;  
}  
  
bool operatorLoop\_2() {  
 return checkTerm(LOOP) &&  
 checkTerm(LEFTFIGURE) &&  
 StartLoopForOperator() &&  
 checkTerm(RIGHTFIGURE);  
}  
  
bool operatorLoop\_1() {  
 return checkTerm(LOOP) &&  
 takeOperator();  
}  
  
  
  
  
bool takeOperator() {  
 int save = next;  
 if (printOperator()) return true;  
 next = save;  
 if (operatorIf()) return true;  
 next = save;  
 if (operatorLoop()) return true;  
 return false;  
}  
  
bool StartLoopForOperator() {  
 bool result = true;  
 while (!peekTerm(RIGHTFIGURE) && result) {  
 result = takeOperator();  
 }  
 return result;  
}  
  
  
  
  
  
  
bool operatorIf\_1() {  
 return checkTerm(IF) &&  
 checkTerm(LEFTBRACKET) &&  
 checkTerm(INT) &&  
 checkTerm(RIGHTBRACKET) &&  
 takeOperator() &&  
 !peekTerm(ELSE);  
}  
  
bool operatorIf\_2() {  
 return checkTerm(IF) &&  
 checkTerm(LEFTBRACKET) &&  
 checkTerm(INT) &&  
 checkTerm(RIGHTBRACKET) &&  
 takeOperator() &&  
 checkTerm(ELSE) &&  
 takeOperator();  
}  
  
bool operatorIf\_3() {  
 return checkTerm(IF) &&  
 checkTerm(LEFTBRACKET) &&  
 checkTerm(INT) &&  
 checkTerm(RIGHTBRACKET) &&  
 checkTerm(LEFTFIGURE) &&  
 StartLoopForOperator() &&  
 checkTerm(RIGHTFIGURE) &&  
 !peekTerm(ELSE);  
}  
  
bool operatorIf\_4() {  
 return checkTerm(IF) &&  
 checkTerm(LEFTBRACKET) &&  
 checkTerm(INT) &&  
 checkTerm(RIGHTBRACKET) &&  
 checkTerm(LEFTFIGURE) &&  
 StartLoopForOperator() &&  
 checkTerm(RIGHTFIGURE) &&  
 checkTerm(ELSE) &&  
 takeOperator();  
}  
  
bool operatorIf\_5() {  
 return checkTerm(IF) &&  
 checkTerm(LEFTBRACKET) &&  
 checkTerm(INT) &&  
 checkTerm(RIGHTBRACKET) &&  
 takeOperator() &&  
 checkTerm(ELSE) &&  
 checkTerm(LEFTFIGURE) &&  
 StartLoopForOperator() &&  
 checkTerm(RIGHTFIGURE);  
}  
  
bool operatorIf\_6() {  
 return checkTerm(IF) &&  
 checkTerm(LEFTBRACKET) &&  
 checkTerm(INT) &&  
 checkTerm(RIGHTBRACKET) &&  
 checkTerm(LEFTFIGURE) &&  
 StartLoopForOperator() &&  
 checkTerm(RIGHTFIGURE) &&  
 checkTerm(ELSE) &&  
 checkTerm(LEFTFIGURE) &&  
 StartLoopForOperator() &&  
 checkTerm(RIGHTFIGURE);  
}  
  
bool operatorIf() {  
 int save = next;  
 if (operatorIf\_1()) return true;  
 next = save;  
 if (operatorIf\_2()) return true;  
 next = save;  
 if (operatorIf\_3()) return true;  
 next = save;  
 if (operatorIf\_4()) return true;  
 next = save;  
 if (operatorIf\_5()) return true;  
 next = save;  
 if (operatorIf\_6()) return true;  
 return false;  
}  
  
std::vector<Types> GetTerms(std::ifstream &in) {  
 std::vector<Types> temp;  
 std::string line;  
 getline(in, line);  
 while (!line.empty())  
 {  
 auto type = (Types) stoi(line);  
 if (type != SPACE && type != EOL)  
 temp.push\_back(type);  
 getline(in, line);  
 }  
 return temp;  
}  
  
int main() {  
 std::string path;  
 std::cout << "Enter your path to the file\n";  
 getline(std::cin, path);  
 std::ifstream in(path);  
  
 if (in.is\_open()) {  
 next = 0;  
 terms = GetTerms(in);  
 in.close();  
  
 printf(funcMainBlock() ? "SUCCESS\n" : "ERROR\n");  
  
 return 0;  
 }  
}