# Выбор целевой платформы

Целевая платформа: Unix-подобная ОС с компилятором GCC. Язык программирования будет транслироваться в язык C.

# Разработка языка программирования

Концепт:

Русский язык – один из наиболее распространённых языков в мире. Людей, которые владеют языком много, среди них есть программисты. Поэтому rus++ — это упрощенный язык программирования, который ориентирован на русскоязычных пользователей. Он будет иметь синтаксис, базирующийся на русском языке, что делает его доступным для начинающих программистов и детей.

**Описание команд:**

1. **Объявить** – команда для объявления переменной. Используется как: объявление a или объявление a 5.
2. **Прировнять** – команда для приравнивания переменной. Используется как: прировнять a 5 или прировнять a b.
3. **Прибавить** – команда для прибавления значения к переменной. Используется как: прибавить a 5 или прибавить a b.
4. **Умножить** – команда для умножения переменной. Используется как: умножить a 5 или умножить a b.
5. **Разделить** – команда для деления переменной. Используется как: разделить a 5 или разделить a b.
6. **Вычесть** – команда для вычитания значения из переменной. Используется как: вычесть a 5 или вычесть a b.
7. **Цикл** – команда для выполнения цикла. Используется как: цикл a > 7 или цикл b < a
8. **Конец** – команда для обозначения конца цикла или оператора «если».
9. **Показать** – выводит переменную на экран. Используется как: показать a
10. **Если** – выводит переменную на экран. Используется как: если a > 5 или если a < b

# Ручная трансляция под целевую платформу

Для примера ручной трансляции языка была написана следующая программа:

объявить b

прировнять b 8

объявить q 10

показать b

прибавить b 7

вычесть b q

объявить i 0

цикл i < 3

умножить b q

разделить q 2

прибавить i 1

конец

если b > q

прибавить q b

конец

показать b

показать q

Ниже представлен код, транслированный на язык С:

#include <stdio.h>

int main(){

int b;

b = 8;

int q = 10;

printf("%d", b); // 8

b += 7; // b = 15

b -= q; // b = 5

int i = 0;

while (i < 3)

{

b \*= q;

q /= 2;

i += 1;

} // b = 500, q = 1, i = 3

if (b > q)

q += b; // q = 501

printf("%d", b); // 8500

printf("%d", q); // 8500501

return 0;

}

# Разработка лексического анализатора

Для разработанного языка был написан лексический анализатор:

%{

#include <stdlib.h>

#include <string.h>

#include "flexer.h"

%}

letter [а-яa-z<>]

digit [0-9]

%%

объявить { return(DECLARE); }

прировнять { return(EQUAL); }

прибавить { return(APPEND); }

вычесть { return(SUBTRACT); }

умножить { return(MULTIPLY); }

разделить { return(DIVIDE); }

цикл { return(LOOP); }

конец { return(END); }

показать { return(PRINT); }

вернуть { return(RETURN); }

если { return(IF); }

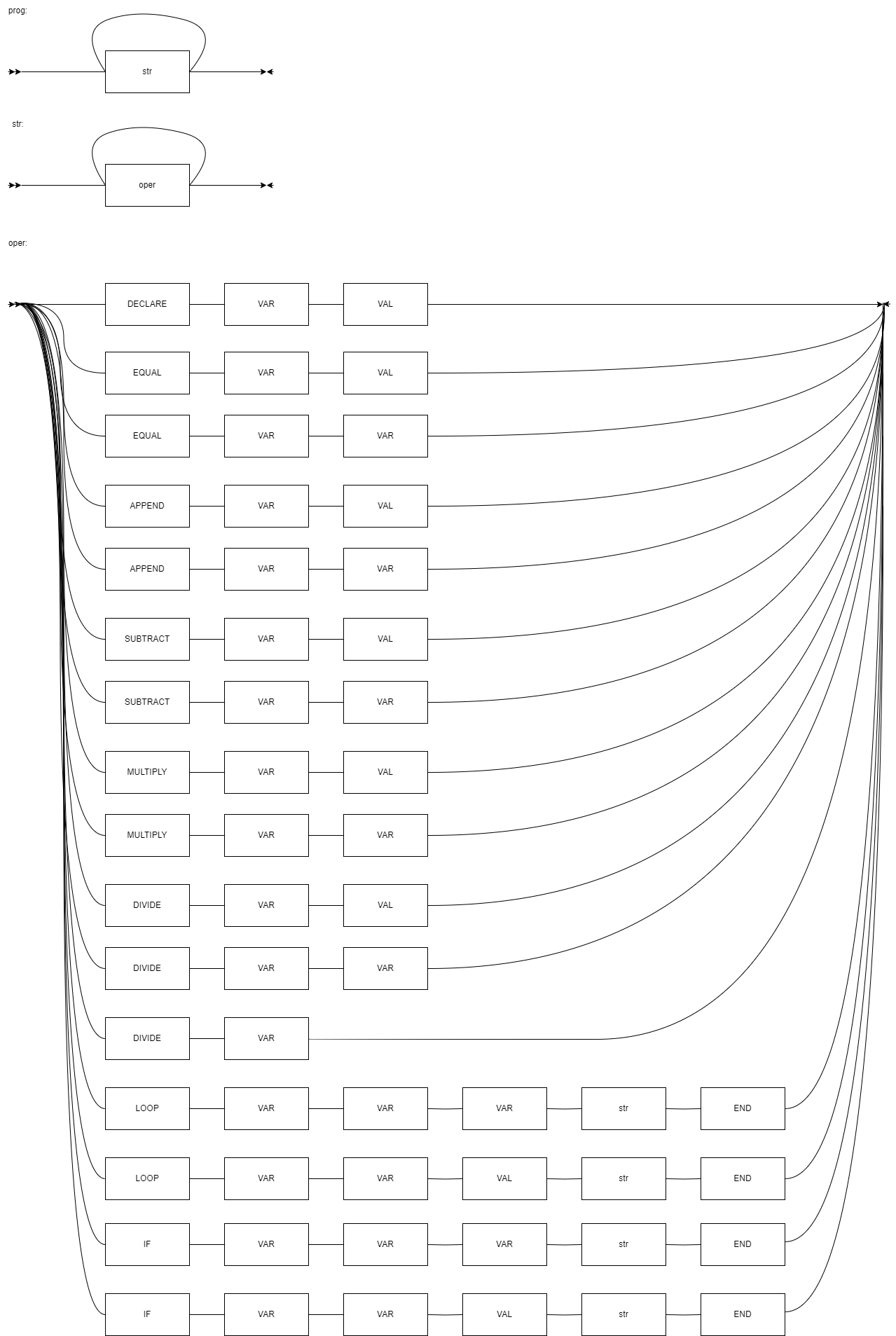
{letter}+ { strcpy( yylval.var, yytext ) ; return( VAR ); }

{digit}+ { sscanf( yytext, "%d", &yylval.val ) ; return( VAL ); }

[ \n] {}

. { write(2,yytext,1); return( yytext[0] ); }

# Грамматика синтаксического разбора



# Разработка синтаксического анализатора

%{

#include <stdio.h>

char buf[2048];

%}

%union {

int val;

char var[2048];

}

%token <val> VAL

%token <var> VAR

%token DECLARE EQUAL APPEND SUBTRACT LOOP END MULTIPLY DIVIDE PRINT RETURN IF

%type <var> str

%type <var> oper

%%

prog: str {printf ("main() { %s }", $1);}

str: oper {

sprintf($$, "%s", $1);

}

| oper str {

sprintf($$, "%s\n%s", $1, $2);

};

oper: DECLARE VAR {

sprintf($$, "int %s;", $<var>2);

}

| DECLARE VAR VAL {

sprintf($$, "int %s=%d;", $<var>2, $<val>3);

}

| EQUAL VAR VAL {

sprintf($$, "%s=%d;", $<var>2, $<val>3);

}

| EQUAL VAR VAR {

sprintf($$, "%s=%s;", $<var>2, $<var>3);

}

| APPEND VAR VAL {

sprintf($$, "%s+=%d;", $<var>2, $<val>3);

}

| APPEND VAR VAR {

sprintf($$, "%s+=%s;", $<var>2, $<var>3);

}

| SUBTRACT VAR VAL {

sprintf($$, "%s-=%d;", $<var>2, $<val>3);

}

| SUBTRACT VAR VAR {

sprintf($$, "%s-=%s;", $<var>2, $<var>3);

}

| MULTIPLY VAR VAL {

sprintf($$, "%s\*=%d;", $<var>2, $<val>3);

}

| MULTIPLY VAR VAR {

sprintf($$, "%s\*=%s;", $<var>2, $<var>3);

}

| DIVIDE VAR VAL {

sprintf($$, "%s/=%d;", $<var>2, $<val>3);

}

| DIVIDE VAR VAR {

sprintf($$, "%s/=%s;", $<var>2, $<var>3);

}

| PRINT VAR {

sprintf($$, "printf(\"%s\", %s);", "%d", $<var>2);

}

| RETURN VAR {

sprintf($$, "return %s;", $<var>2);

}

| LOOP VAR VAR VAR str END {

sprintf($$, "while(%s %s %s) {%s}", $<var>2, $<var>3, $<var>4, $<var>5);

}

| LOOP VAR VAR VAL str END {

sprintf($$, "while(%s %s %d) {%s}", $<var>2, $<var>3, $<val>4, $<var>5);

}

| IF VAR VAR VAR str END {

sprintf($$, "if(%s %s %s) {%s}", $<var>2, $<var>3, $<var>4, $<var>5);

}

| IF VAR VAR VAL str END {

sprintf($$, "if(%s %s %d) {%s}", $<var>2, $<var>3, $<val>4, $<var>5);

}

;

%%

yyerror (s)

char \*s;

{

printf ("err:%s\n", s);

}

main()

{

yyparse();

}

# Компиляция компилятора

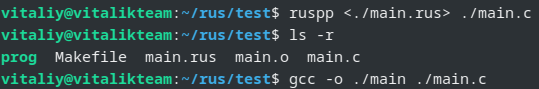
bison -d -o flexer.c flexer.y

flex -o lexer.c lexxr.y

gcc -o ruspp lexer.c flexer.c –lfl

sudo mv ruspp /usr/local/bin/

# Компилирование программы на своем языке



1. Запуск транслированной программы на языке C



1. MakeFile

TARGET = prog

cc = gcc

cflags = -Wall -g

all: $(TARGET)

$(TARGET): ./main.o

$(cc) $(cflags) -o $@ $^

main.c: ./main.rus

ruspp <./main.rus> ./main.c

main.o: ./main.c

$(cc) $(cflags) -c $< -o $@

clean:

rm -f \*.o

