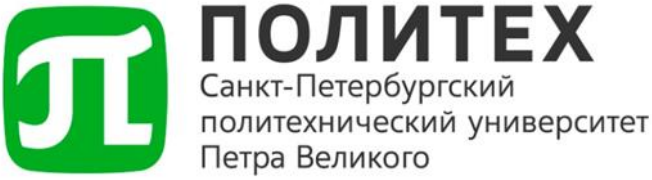
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий

Высшая школа программной инженерии



К У Р С О В А Я Р А Б О Т А

Арифметическое “if”: <логическое выражение>? <выражение>: <выражение>

по дисциплине «Теория автоматов и формальных языков»

Студент гр. *3530202/80002* Е.А. Козлова

Руководитель

Ст. преподаватель Ю.Г. Карпов

Санкт-Петербург

2020 г

Содержание

Задание.

К имеющемуся функционалу языка Милан добавить арифметическое “if”: <логическое выражение>? <выражение>: <выражение>

Описание изменений в языке.

⟨program⟩ ::= ‘begin’ ⟨statementList⟩ ‘end’

⟨statementList⟩ ::= ⟨statement⟩ ‘;’ ⟨statementList⟩| ϵ

⟨statement⟩ ::= ⟨ident⟩ ‘:=’ ⟨expression⟩

| ‘if’ ⟨relation⟩ ‘then’ ⟨statementList⟩ [‘else’ ⟨statementList⟩] ‘fi’

| ‘while’ ⟨relation⟩ ‘do’ ⟨statementList⟩ ‘od’

| ‘write’ ‘(’ ⟨expression⟩ ‘)’

| ‘if’ ⟨relation⟩ ‘?’ ⟨statement⟩ ‘:’ ⟨statement⟩

⟨expression⟩ ::= ⟨term⟩ {⟨addop⟩ ⟨term⟩}

⟨term⟩ ::= ⟨factor ⟩ {⟨mulop⟩ ⟨factor ⟩}

⟨factor ⟩ ::= ⟨ident⟩ |⟨number⟩ | ‘(’ ⟨expression⟩ ‘)’

⟨relation⟩ ::= ⟨expression⟩ ⟨cmp⟩ ⟨expression⟩

⟨addop⟩ ::= ‘+’ | ‘-’

⟨multop⟩ ::= ‘*’ | ‘/’

⟨cmp⟩ ::= ‘=’ | ‘!=’ | ‘<’ | ‘<=’ | ‘>’ | ‘>=’

⟨ident⟩ ::= ⟨letter ⟩ {⟨letter ⟩ |⟨digit⟩}

⟨letter ⟩ ::= ‘a’ | ‘b’ | ‘c’ |. . .| ‘z’ | ‘A’ | ‘B’ | ‘C’ |. . .| ‘Z’

⟨digit⟩ ::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’

Арифметическое “if”:
-условие,
-первое выражение, которое выполняется, если условие истинное,
-второе выражение, которое выполняется, если условие ложное.

Описание изменений в компиляторе.

scanner.cpp

```
void Scanner::nextToken()
{
	skipSpace();
	//если комментарий
	while(ch_ == '/') {
		…
	}
	//если символ конца файла
	if(input_.eof()) {
		…
	}
	//если число
	if(isdigit(ch_)) {
		…
	}
	else if(isIdentifierStart(ch_)) {
		…
	}
	//Символ не является буквой, цифрой, "/" или признаком конца файла
	else {
		switch(ch_) {
			…
			//Если встречаем ":", то дальше смотрим наличие символа "=". Если
находим, то считаем что нашли лексему присваивания
			//Иначе - просто ":".
			case ':':
				nextChar();
				if(ch_ == '=') {
					token_ = T_ASSIGN;
					nextChar();

				}
				else {
					token_ = T_COLON;
				}
				break;
			…
			case '?':
				token_ = T_QMARK;
				nextChar();
				break;
			…
		}
	}
```

parser.cpp

```
void Parser::statement()
{
	//⟨ident⟩ ‘:=’ ⟨expression⟩
	if(see(T_IDENTIFIER)) {
		…
	}
	// Если встретили IF, то затем должно следовать условие. На вершине стека лежит 1 или 0 в зависимости от выполнения условия.
	// Затем зарезервируем место для условного перехода JUMP_NO к блоку ELSE (переход в случае ложного условия). Адрес перехода
	// станет известным только после того, как будет сгенерирован код для блока THEN.

	//Если после условия следует символ «?» - значит это арифметическое if
	// На вершине стека лежит 1 или 0 в зависимости от выполнения условия.
	// Затем зарезервируем место для условного перехода в случае ложного условия). Адрес перехода станет известным только после того, как будет сгенерирован код для выражения, если условие неложное
	else if(match(T_IF))
	{
	relation();

	int jumpNoAddress = codegen_->reserve();

	if(match(T_THEN))
	{
		statementList();
		if(match(T_ELSE)) {
		//Если есть блок ELSE, то чтобы не выполнять его в случае выполнения THEN,
		  //зарезервируем место для команды JUMP в конец этого блока
		  int jumpAddress = codegen_->reserve();
		  //Заполним зарезервированное место после проверки условия инструкцией перехода в начало блока ELSE.
		  codegen_->emitAt(jumpNoAddress, JUMP_NO, codegen_->getCurrentAddress());
		  statementList();
		 //Заполним второй адрес инструкцией перехода в конец условного блока ELSE.
		  codegen_->emitAt(jumpAddress, JUMP, codegen_->getCurrentAddress());
		}
		else {
		//Если блок ELSE отсутствует, то в зарезервированный адрес после проверки условия будет записана
		//инструкция условного перехода в конец оператора IF...THEN
		  codegen_->emitAt(jumpNoAddress, JUMP_NO, codegen_->getCurrentAddress());
		}
		mustBe(T_FI);
	}
		else if(match(T_QMARK))
		{
		  statement();
		  codegen_->emitAt(jumpNoAddress,JUMP_NO, codegen_->getCurrentAddress()+1);
		  mustBe(T_COLON);
		  int jumpAddress = codegen_->reserve();
		  statement();
		  codegen_->emitAt(jumpAddress, JUMP, codegen_->getCurrentAddress());
		}
	}
	else if(match(T_WHILE)) {
		…
	}
}
```

Тест.

Тест1:

```
BEGIN
        i := 1;
        j := 5;
        IF i > j ? i:=111 : j:=1111;
        WRITE(i);
         WRITE(j)
END
```

Тест2:

```
BEGIN

        i := 1;

        j := 5;

        IF i < j ? i:=111 : j:=1111;

        WRITE(i);

         WRITE(j)

END
```

Результат работы компилятора.

Тест1:

```
0:      PUSH    1
1:      STORE   0
2:      PUSH    5
3:      STORE   1
4:      LOAD    0
5:      LOAD    1
6:      COMPARE 3
7:      JUMP_NO 11
8:      PUSH    111
9:      STORE   0
10:     JUMP    13
11:     PUSH    1111
12:     STORE   1
13:     LOAD    0
14:     PRINT
15:     LOAD    1
16:     PRINT
17:     STOP
```

Тест2:

```
0:      PUSH    1
1:      STORE   0
2:      PUSH    5
3:      STORE   1
4:      LOAD    0
5:      LOAD    1
6:      COMPARE 2
7:      JUMP_NO 11
8:      PUSH    111
9:      STORE   0
10:     JUMP    13
11:     PUSH    1111
```

```
12:     STORE   1
13:     LOAD    0
14:     PRINT
15:     LOAD    1
16:     PRINT
17:     STOP
```

Результат работы интерпретатора.

Тест1:

1
1111

Тест2:

111
5