Министерство образования и науки РФ

Государственное бюджетное образовательное учреждение

высшего образования

«Тверской государственный университет»

Факультет «Прикладной математики и кибернетики»

Направление «02.03.02 Фундаментальная информатика
и информационные технологии»

**ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Вариант 17

Выполнил:

студент 26 группы

Баранов К.С.

Проверила: Мальцева Л.Ф.

Тверь, 2017

**Задание:**

 Даны два натуральных числа $n$ и $m$. Проверить, являются ли они числами Мерсена. Простое число называется числом Мерсена, если оно может быть представлено в виде $2^{p}-1$, где $p$ – простое число.

**Листинг программы на языке mini-Basic:**

1 rem Программа проверки, является ли число числом Мерсена

2 rem Входная переменная - a

3 rem Выходные переменные - r1 и r2

10 let a = 10

20 let n1 = a

30 gosub 1000

40 let r1 = p1

50 let a = 31

60 let n1 = a

70 gosub 1000

80 let r2 = p1

90 goto 5000

1000 rem Подпрограмма проверки числа Мерсена

1001 rem Входные данные - n1

1002 rem Выходные данные - p1

1010 let n2 = n1

1020 gosub 2000

1030 if p2 = 0 goto 1200

1040 let n1 = n1 + 1

1050 let n4 = n1

1060 gosub 4000

1070 let n2 = p4

1080 gosub 2000

1090 if p2 = 0 goto 1200

1100 let p1 = 1

1110 return

1200 let p1 = 0

1210 return

2000 rem Проверка числа на простоту

2001 rem Входные данные - n2

2002 rem Выходные данные - p2

2010 for i2 = 2 to n2 - 1 step 1

2020 let n3 = n2

2030 let m3 = i2

2040 gosub 3000

2050 if p3 = 0 goto 2100

2060 next i2

2070 let p2 = 1

2080 return

2100 let p2 = 0

2110 return

3000 rem Подпрограмма вычисления остатка от деления

3001 rem Входные параметры - n3, m3

3002 rem Выходной параметр - p3

3010 let a3 = n3 / m3

3020 let b3 = m3 \* a3

3030 let p3 = n3 - b3

3040 return

4000 rem Подпрограмма вычисления степени двойки

4001 rem Входной параметр - n4

4002 rem Выходной параметр - p4

4010 let p4 = 0

4015 if n4 > 1 goto 4020

4016 let p4 = 0

4017 return

4020 let n3 = n4

4030 let m3 = 2

4040 gosub 3000

4050 if p3 = 1 goto 4016

4055 let p4 = p4 + 1

4060 let n4 = n4 / 2

4070 if n4 = 1 goto 4080

4075 goto 4020

4080 return

5000 end

**Результат работы лексического блока**

Созданные лексемы:

 Лексема Значение Комментарий

 LABEL 2 1

 REM 0 REM

 LABEL 3 2

 REM 0 REM

 LABEL 4 3

 REM 0 REM

 LABEL 11 10

 LET 0 LET a=

 OPERAND 286 10.000000

 LABEL 21 20

 LET 65 LET n1=

 OPERAND 0 a

 LABEL 31 30

 GOSUB 1 GOSUB

 LABEL 41 40

 LET 69 LET r1=

 OPERAND 67 p1

 LABEL 51 50

 LET 0 LET a=

 OPERAND 287 31.000000

 LABEL 61 60

 LET 65 LET n1=

 OPERAND 0 a

 LABEL 71 70

 GOSUB 1 GOSUB

 LABEL 81 80

 LET 95 LET r2=

 OPERAND 67 p1

 LABEL 91 90

 GOTO 101 GOTO

 LABEL 1 1000

 REM 0 REM

 LABEL 102 1001

 REM 0 REM

 LABEL 103 1002

 REM 0 REM

 LABEL 104 1010

 LET 91 LET n2=

 OPERAND 65 n1

 LABEL 105 1020

 GOSUB 106 GOSUB

 LABEL 107 1030

 IF 0 IF

 OPERAND 93 p2

 OTN\_OP 1 OTN\_OP

 OPERAND 288 0.000000

 GOTO 108 GOTO

 LABEL 109 1040

 LET 65 LET n1=

 OPERAND 65 n1

 AR\_OP 1 AR\_OP

 OPERAND 289 1.000000

 LABEL 110 1050

 LET 143 LET n4=

 OPERAND 65 n1

 LABEL 111 1060

 GOSUB 112 GOSUB

 LABEL 113 1070

 LET 91 LET n2=

 OPERAND 145 p4

 LABEL 114 1080

 GOSUB 106 GOSUB

 LABEL 115 1090

 IF 0 IF

 OPERAND 93 p2

 OTN\_OP 1 OTN\_OP

 OPERAND 288 0.000000

 GOTO 108 GOTO

 LABEL 116 1100

 LET 67 LET p1=

 OPERAND 289 1.000000

 LABEL 117 1110

 RETURN 0 RETURN

 LABEL 108 1200

 LET 67 LET p1=

 OPERAND 288 0.000000

 LABEL 118 1210

 RETURN 0 RETURN

 LABEL 106 2000

 REM 0 REM

 LABEL 119 2001

 REM 0 REM

 LABEL 120 2002

 REM 0 REM

 LABEL 121 2010

 FOR 86 FOR

 OPERAND 290 2.000000

 TO 0 TO

 OPERAND 91 n2

 AR\_OP 2 AR\_OP

 OPERAND 289 1.000000

 STEP 0 STEP

 OPERAND 289 1.000000

 LABEL 122 2020

 LET 117 LET n3=

 OPERAND 91 n2

 LABEL 123 2030

 LET 116 LET m3=

 OPERAND 86 i2

 LABEL 124 2040

 GOSUB 125 GOSUB

 LABEL 126 2050

 IF 0 IF

 OPERAND 119 p3

 OTN\_OP 1 OTN\_OP

 OPERAND 288 0.000000

 GOTO 127 GOTO

 LABEL 128 2060

 NEXT 86 NEXT

 LABEL 129 2070

 LET 93 LET p2=

 OPERAND 289 1.000000

 LABEL 130 2080

 RETURN 0 RETURN

 LABEL 127 2100

 LET 93 LET p2=

 OPERAND 288 0.000000

 LABEL 131 2110

 RETURN 0 RETURN

 LABEL 125 3000

 REM 0 REM

 LABEL 132 3001

 REM 0 REM

 LABEL 133 3002

 REM 0 REM

 LABEL 134 3010

 LET 104 LET a3=

 OPERAND 117 n3

 AR\_OP 4 AR\_OP

 OPERAND 116 m3

 LABEL 135 3020

 LET 105 LET b3=

 OPERAND 116 m3

 AR\_OP 3 AR\_OP

 OPERAND 104 a3

 LABEL 136 3030

 LET 119 LET p3=

 OPERAND 117 n3

 AR\_OP 2 AR\_OP

 OPERAND 105 b3

 LABEL 137 3040

 RETURN 0 RETURN

 LABEL 112 4000

 REM 0 REM

 LABEL 138 4001

 REM 0 REM

 LABEL 139 4002

 REM 0 REM

 LABEL 140 4010

 LET 145 LET p4=

 OPERAND 288 0.000000

 LABEL 16 4015

 IF 0 IF

 OPERAND 143 n4

 OTN\_OP 3 OTN\_OP

 OPERAND 289 1.000000

 GOTO 141 GOTO

 LABEL 17 4016

 LET 145 LET p4=

 OPERAND 288 0.000000

 LABEL 18 4017

 RETURN 0 RETURN

 LABEL 141 4020

 LET 117 LET n3=

 OPERAND 143 n4

 LABEL 142 4030

 LET 116 LET m3=

 OPERAND 290 2.000000

 LABEL 143 4040

 GOSUB 125 GOSUB

 LABEL 144 4050

 IF 0 IF

 OPERAND 119 p3

 OTN\_OP 1 OTN\_OP

 OPERAND 289 1.000000

 GOTO 17 GOTO

 LABEL 56 4055

 LET 145 LET p4=

 OPERAND 145 p4

 AR\_OP 1 AR\_OP

 OPERAND 289 1.000000

 LABEL 145 4060

 LET 143 LET n4=

 OPERAND 143 n4

 AR\_OP 4 AR\_OP

 OPERAND 290 2.000000

 LABEL 146 4070

 IF 0 IF

 OPERAND 143 n4

 OTN\_OP 1 OTN\_OP

 OPERAND 289 1.000000

 GOTO 147 GOTO

 LABEL 76 4075

 GOTO 141 GOTO

 LABEL 147 4080

 RETURN 0 RETURN

 LABEL 101 5000

 END 0 END

 END\_FILE 0

**Результат работы синтаксического анализатора**

Таблица атомов:

 Название Атрибуты

--------------------------------------------

 НОМСТРОК 2

 НОМСТРОК 3

 НОМСТРОК 4

 НОМСТРОК 11

 ПРИСВ 0 286

 НОМСТРОК 21

 ПРИСВ 65 0

 НОМСТРОК 31

 ХРАН ПЕРЕХОД 1

 НОМСТРОК 41

 ПРИСВ 69 67

 НОМСТРОК 51

 ПРИСВ 0 287

 НОМСТРОК 61

 ПРИСВ 65 0

 НОМСТРОК 71

 ХРАН ПЕРЕХОД 1

 НОМСТРОК 81

 ПРИСВ 95 67

 НОМСТРОК 91

 ПЕРЕХОД 101

 НОМСТРОК 1

 НОМСТРОК 102

 НОМСТРОК 103

 НОМСТРОК 104

 ПРИСВ 91 65

 НОМСТРОК 105

 ХРАН ПЕРЕХОД 106

 НОМСТРОК 107

 УСЛ ПЕРЕХОД 93 288 1 108

 НОМСТРОК 109

 СЛОЖ 65 289 291

 ПРИСВ 65 291

 НОМСТРОК 110

 ПРИСВ 143 65

 НОМСТРОК 111

 ХРАН ПЕРЕХОД 112

 НОМСТРОК 113

 ПРИСВ 91 145

 НОМСТРОК 114

 ХРАН ПЕРЕХОД 106

 НОМСТРОК 115

 УСЛ ПЕРЕХОД 93 288 1 108

 НОМСТРОК 116

 ПРИСВ 67 289

 НОМСТРОК 117

 ВОЗВ ПЕРЕХОД

 НОМСТРОК 108

 ПРИСВ 67 288

 НОМСТРОК 118

 ВОЗВ ПЕРЕХОД

 НОМСТРОК 106

 НОМСТРОК 119

 НОМСТРОК 120

 НОМСТРОК 121

 ПРИСВ 86 290

 ВЫЧИТ 91 289 294

 ХРАНЕНИЕ 294 292

 ХРАНЕНИЕ 289 293

 МЕТКА 148

 ПРОВЕРКА 86 292 293 149

 НОМСТРОК 122

 ПРИСВ 117 91

 НОМСТРОК 123

 ПРИСВ 116 86

 НОМСТРОК 124

 ХРАН ПЕРЕХОД 125

 НОМСТРОК 126

 УСЛ ПЕРЕХОД 119 288 1 127

 НОМСТРОК 128

 УВЕЛ 86 293

 ПЕРЕХОД 148

 МЕТКА 149

 НОМСТРОК 129

 ПРИСВ 93 289

 НОМСТРОК 130

 ВОЗВ ПЕРЕХОД

 НОМСТРОК 127

 ПРИСВ 93 288

 НОМСТРОК 131

 ВОЗВ ПЕРЕХОД

 НОМСТРОК 125

 НОМСТРОК 132

 НОМСТРОК 133

 НОМСТРОК 134

 ДЕЛЕН 117 116 295

 ПРИСВ 104 295

 НОМСТРОК 135

 УМНОЖ 116 104 296

 ПРИСВ 105 296

 НОМСТРОК 136

 ВЫЧИТ 117 105 297

 ПРИСВ 119 297

 НОМСТРОК 137

 ВОЗВ ПЕРЕХОД

 НОМСТРОК 112

 НОМСТРОК 138

 НОМСТРОК 139

 НОМСТРОК 140

 ПРИСВ 145 288

 НОМСТРОК 16

 УСЛ ПЕРЕХОД 143 289 3 141

 НОМСТРОК 17

 ПРИСВ 145 288

 НОМСТРОК 18

 ВОЗВ ПЕРЕХОД

 НОМСТРОК 141

 ПРИСВ 117 143

 НОМСТРОК 142

 ПРИСВ 116 290

 НОМСТРОК 143

 ХРАН ПЕРЕХОД 125

 НОМСТРОК 144

 УСЛ ПЕРЕХОД 119 289 1 17

 НОМСТРОК 56

 СЛОЖ 145 289 298

 ПРИСВ 145 298

 НОМСТРОК 145

 ДЕЛЕН 143 290 299

 ПРИСВ 143 299

 НОМСТРОК 146

 УСЛ ПЕРЕХОД 143 289 1 147

 НОМСТРОК 76

 ПЕРЕХОД 141

 НОМСТРОК 147

 ВОЗВ ПЕРЕХОД

 НОМСТРОК 101

 ОКОНЧАНИЕ

**Результат работы генератора кода**

.486 ; create 32 bit code

.model flat, stdcall ; 32 bit memory model

option casemap :none ; case sensitive

include \masm32\include\masm32rt.inc

.data

va dd 0

vn1 dd 0

vp1 dd 0

vr1 dd 0

vi2 dd 0

vn2 dd 0

vp2 dd 0

vr2 dd 0

va3 dd 0

vb3 dd 0

vm3 dd 0

vn3 dd 0

vp3 dd 0

vn4 dd 0

vp4 dd 0

v286 dd 10

v287 dd 31

v288 dd 0

v289 dd 1

v290 dd 2

v291 dd 0

v292 dd 0

v293 dd 0

v294 dd 0

v295 dd 0

v296 dd 0

v297 dd 0

v298 dd 0

v299 dd 0

hout dd ?

ns dd ?

.code

MainProc proc

l2:

l3:

l4:

l11:

 mov eax, v286

 mov va, eax

l21:

 mov eax, va

 mov vn1, eax

l31:

 call l1

l41:

 mov eax, vp1

 mov vr1, eax

l51:

 mov eax, v287

 mov va, eax

l61:

 mov eax, va

 mov vn1, eax

l71:

 call l1

l81:

 mov eax, vp1

 mov vr2, eax

l91:

 jmp l101

l1:

l102:

l103:

l104:

 mov eax, vn1

 mov vn2, eax

l105:

 call l106

l107:

 mov eax, vp2

 cmp eax, v288

 je l108

l109:

 mov eax, vn1

 add eax, v289

 mov v291, eax

 mov eax, v291

 mov vn1, eax

l110:

 mov eax, vn1

 mov vn4, eax

l111:

 call l112

l113:

 mov eax, vp4

 mov vn2, eax

l114:

 call l106

l115:

 mov eax, vp2

 cmp eax, v288

 je l108

l116:

 mov eax, v289

 mov vp1, eax

l117:

 ret

l108:

 mov eax, v288

 mov vp1, eax

l118:

 ret

l106:

l119:

l120:

l121:

 mov eax, v290

 mov vi2, eax

 mov eax, vn2

 sub eax, v289

 mov v294, eax

 mov eax, v294

 mov v292, eax

 mov eax, v289

 mov v293, eax

l148:

 mov eax, vi2

 cmp eax, v292

 jg l149

l122:

 mov eax, vn2

 mov vn3, eax

l123:

 mov eax, vi2

 mov vm3, eax

l124:

 call l125

l126:

 mov eax, vp3

 cmp eax, v288

 je l127

l128:

 mov eax, vi2

 add eax, v293

 mov vi2, eax

 jmp l148

l149:

l129:

 mov eax, v289

 mov vp2, eax

l130:

 ret

l127:

 mov eax, v288

 mov vp2, eax

l131:

 ret

l125:

l132:

l133:

l134:

 mov eax, vn3

 cdq

 div vm3

 mov v295, eax

 mov eax, v295

 mov va3, eax

l135:

 mov eax, vm3

 mul va3

 mov v296, eax

 mov eax, v296

 mov vb3, eax

l136:

 mov eax, vn3

 sub eax, vb3

 mov v297, eax

 mov eax, v297

 mov vp3, eax

l137:

 ret

l112:

l138:

l139:

l140:

 mov eax, v288

 mov vp4, eax

l16:

 mov eax, vn4

 cmp eax, v289

 jg l141

l17:

 mov eax, v288

 mov vp4, eax

l18:

 ret

l141:

 mov eax, vn4

 mov vn3, eax

l142:

 mov eax, v290

 mov vm3, eax

l143:

 call l125

l144:

 mov eax, vp3

 cmp eax, v289

 je l17

l56:

 mov eax, vp4

 add eax, v289

 mov v298, eax

 mov eax, v298

 mov vp4, eax

l145:

 mov eax, vn4

 cdq

 div v290

 mov v299, eax

 mov eax, v299

 mov vn4, eax

l146:

 mov eax, vn4

 cmp eax, v289

 je l147

l76:

 jmp l141

l147:

 ret

l101:

invoke GetStdHandle, STD\_OUTPUT\_HANDLE

mov hout, eax

invoke WriteConsoleA, hout, str$(vr1), sizeof vr1, offset ns, NULL

invoke WriteConsoleA, hout, str$(vr2), sizeof vr2, offset ns, NULL

invoke ExitProcess, 0

MainProc endp

end MainProc

**Результат счёта**

Первым числом было задано 10, вторым – 31.



**Листинг программы генератора кода**

//--Code\_Generator.h

#pragma once

#include "Syntax\_Block.h"

class Code\_Generator : public Syntax\_Block {

public:

 Code\_Generator(Syntax\_Block&);

 ~Code\_Generator();

 friend std::ostream& operator << (std::ostream&, Code\_Generator&);

protected:

 std::vector<std::string> errors;

private:

 int exp\_count; // Счетчик выполнений возведения в степень

 void printMainInfo(std::ostream&);

 void printVar(std::ostream&, int);

 void descVar (std::ostream&);

 void okonchanie(atom\_cell&, std::ostream&);

 void nomstr(atom\_cell&, std::ostream&);

 void prisv (atom\_cell&, std::ostream&);

 void perehod(atom\_cell&, std::ostream&);

 void hran\_perehod(atom\_cell&, std::ostream&);

 void vozv\_perehod(atom\_cell&, std::ostream&);

 void usl\_perehod(atom\_cell&, std::ostream&);

 void hranenie(atom\_cell&, std::ostream&);

 void metka(atom\_cell&, std::ostream&);

 void proverka(atom\_cell&, std::ostream&);

 void uvel(atom\_cell&, std::ostream&);

 void sloj(atom\_cell&, std::ostream&);

 void vichit(atom\_cell&, std::ostream&);

 void umnoj(atom\_cell&, std::ostream&);

 void delen(atom\_cell&, std::ostream&);

 void exp(atom\_cell&, std::ostream&);

 void plus(atom\_cell&, std::ostream&);

 void minus(atom\_cell&, std::ostream&);

};

//--Code\_Generator.cpp

#include "stdafx.h"

#include "Code\_Generator.h"

using namespace std;

Code\_Generator::Code\_Generator(Syntax\_Block &sb) :

 Syntax\_Block(sb), exp\_count(0), for\_count(0) {}

Code\_Generator::~Code\_Generator() {}

void Code\_Generator::printMainInfo(std::ostream &os) {

 os << ".486 ; create 32 bit code" << endl <<

 ".model flat, stdcall ; 32 bit memory model" << endl <<

 "option casemap :none ; case sensitive" << endl << endl <<

 "include \\masm32\\include\\masm32rt.inc" << endl;

}

inline void Code\_Generator::printVar(std::ostream &os, int a) {

 os << 'v';

 if (a < 286) os << getVar(a);

 else os << a;

}

void Code\_Generator::descVar(ostream &os) {

 os << ".data" << endl;

 for (int i = 0; i < 286; i++) {

 if (UTO[i] == -1) {

 os << 'v' << getVar(i) << '\t' << "dd" << '\t' <<

'0' << endl;

 }

 }

 for (int i = 286; i < UTO.size(); i++) {

 os << 'v' << i << '\t' << "dd" << '\t';

 if (UTO[i] > 0) os << UTO[i];

 else os << '0';

 os << endl;

 }

 os << "hout\tdd\t?" << endl <<

 "ns\tdd\t?" << endl;

}

void Code\_Generator::okonchanie(atom\_cell &ac, std::ostream &os) {

 os << "invoke GetStdHandle, STD\_OUTPUT\_HANDLE" << endl <<

 "mov hout, eax" << endl <<

 "invoke WriteConsoleA, hout, str$(vr1), sizeof vr1,

offset ns, NULL" << endl <<

 "invoke WriteConsoleA, hout, str$(vr2), sizeof vr2,

offset ns, NULL" << endl;

}

void Code\_Generator::nomstr(atom\_cell &ac, std::ostream &os) {

 os << 'l' << ac.params[0] << ": " << endl;

}

void Code\_Generator::prisv(atom\_cell &ac, ostream &os) {

 if (ac.params[0] >= 286) {

 errors.push\_back("Попытка присваивания к константе");

 return;

 }

 os << "\tmov eax, ";

 printVar(os, ac.params[1]);

 os << endl << "\tmov ";

 printVar(os, ac.params[0]);

 os << ", eax" << endl;

}

void Code\_Generator::perehod(atom\_cell &ac, ostream & os) {

 os << "\tjmp l" << ac.params[0] << endl;

}

void Code\_Generator::hran\_perehod(atom\_cell &ac, std::ostream &os) {

 os << "\tcall l" << ac.params[0] << endl;

}

void Code\_Generator::vozv\_perehod(atom\_cell &ac, ostream &os) {

 os << "\tret" << endl;

}

void Code\_Generator::usl\_perehod(atom\_cell &ac, ostream &os) {

 os << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tcmp eax, ";

 printVar(os, ac.params[1]);

 os << endl << '\t';

 switch (ac.params[2]) {

 case (1): os << "je "; break;

 case (3): os << "jg "; break;

 case (2): os << "jl "; break;

 case (5): os << "jge "; break;

 case (4): os << "jle "; break;

 case (6): os << "jne "; break;

 }

 os << 'l' << ac.params[3] << endl;

}

void Code\_Generator::hranenie(atom\_cell &ac, std::ostream &os) {

 os << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tmov ";

 printVar(os, ac.params[1]);

 os << ", eax" << endl;

}

void Code\_Generator::metka(atom\_cell &ac, std::ostream &os) {

 os << "l" << ac.params[0] << ':' << endl;

}

void Code\_Generator::proverka(atom\_cell &ac, std::ostream &os) {

 os << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tcmp eax, ";

 printVar(os, ac.params[1]);

 os << endl << "\tjg l" << ac.params[3] << endl;

}

void Code\_Generator::plus(atom\_cell &ac, std::ostream &os) {}

void Code\_Generator::minus(atom\_cell &ac, std::ostream &os) {

 os << "\tmov eax, 0";

 os << endl << "\tsub eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tmov ";

 printVar(os, ac.params[1]);

 os << ", eax" << endl;

}

ostream & operator<<(ostream &os, Code\_Generator &cg) {

 std::vector<Code\_Generator::atom\_cell>::iterator it;

 cg.printMainInfo(os);

 os << endl;

 cg.descVar(os);

 os << endl;

 os << ".code" << endl << "MainProc proc" << endl;

 for (it = cg.atom\_table.begin();

it != cg.atom\_table.end(); it++) {

 switch (it->name) {

 case ENDS: cg.okonchanie(\*it, os); break;

 case NOMSTR: cg.nomstr(\*it, os); break;

 case PRISV: cg.prisv(\*it, os); break;

 case PEREHOD: cg.perehod(\*it, os); break;

 case HRAN\_PEREHOD: cg.hran\_perehod(\*it, os); break;

 case VOZV\_PEREHOD: cg.vozv\_perehod(\*it, os); break;

 case USL\_PEREHOD: cg.usl\_perehod(\*it, os); break;

 case HRANENIE: cg.hranenie(\*it, os); break;

 case METKA: cg.metka(\*it, os); break;

 case PROVERKA: cg.proverka(\*it, os); break;

 case UVEL: cg.uvel(\*it, os); break;

 case SLOJ: cg.sloj(\*it, os); break;

 case VICHIT: cg.vichit(\*it, os); break;

 case UMNOJ: cg.umnoj(\*it, os); break;

 case DELEN: cg.delen(\*it, os); break;

 case EXP: cg.exp(\*it, os); break;

 case PLUS: cg.plus(\*it, os); break;

 case MINUS: cg.minus(\*it, os); break;

 }

 }

 os << "invoke ExitProcess, 0" << endl <<

 "MainProc endp" << endl << "end MainProc" << endl;

 return os;

}

void Code\_Generator::uvel(atom\_cell &ac, std::ostream &os) {

 os << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tadd eax, ";

 printVar(os, ac.params[1]);

 os << endl << "\t mov ";

 printVar(os, ac.params[0]);

 os << ", eax" << endl;

}

void Code\_Generator::sloj(atom\_cell &ac, ostream &os) {

 os << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tadd eax, ";

 printVar(os, ac.params[1]);

 os << endl << "\tmov ";

 printVar(os, ac.params[2]);

 os << ", eax" << endl;

}

void Code\_Generator::vichit(atom\_cell &ac, std::ostream &os) {

 os << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tsub eax, ";

 printVar(os, ac.params[1]);

 os << endl << "\tmov ";

 printVar(os, ac.params[2]);

 os << ", eax" << endl;

}

void Code\_Generator::umnoj(atom\_cell &ac, std::ostream &os) {

 os << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tmul ";

 printVar(os, ac.params[1]);

 os << endl << "\tmov ";

 printVar(os, ac.params[2]);

 os << ", eax" << endl;

}

void Code\_Generator::delen(atom\_cell &ac, std::ostream &os) {

 os << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tcdq";

 os << endl << "\tdiv ";

 printVar(os, ac.params[1]);

 os << endl << "\tmov ";

 printVar(os, ac.params[2]);

 os << ", eax" << endl;

}

void Code\_Generator::exp(atom\_cell &ac, std::ostream &os) {

 os << "\tmov edx, 0" << endl << "\tmov eax, ";

 printVar(os, ac.params[0]);

 os << endl << "\tmov cx, ";

 printVar(os, ac.params[1]);

 os << endl << "\tdec cx" << endl << 'e' <<

exp\_count << ':' << endl;

 os << "\tmul ";

 printVar(os, ac.params[0]);

 os << endl << "\tloop e" << exp\_count << endl;

 os << "\tmov ";

 printVar(os, ac.params[2]);

 os << ", eax" << endl;

 ++exp\_count;

}