**Студент: Моисеенков И. П.**

**Группа: М8О-208Б-19**

**Номер по списку: 21**

**«СИСТЕМЫ ПРОГРАММИРОВАНИЯ»**

**Курсовая работа 2021.**

**Часть 1.**

**Для заданного в Лабораторной №8 диалекта языка МИКРОЛИСП разработайте синтаксически управляемый транслятор на язык С++, применяя методику из Лабораторной №9, Правила TranslationRules21.rtf и TextLayout.txt .**

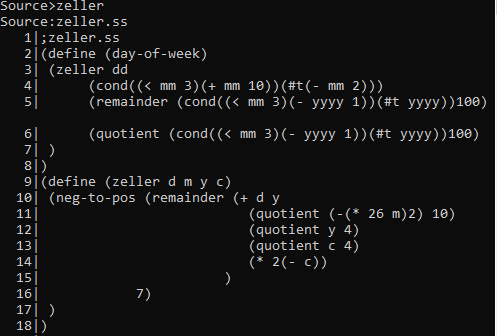
**Работоспособность транслятора проверьте на трех контрольных задачах из Лабораторной №8.**

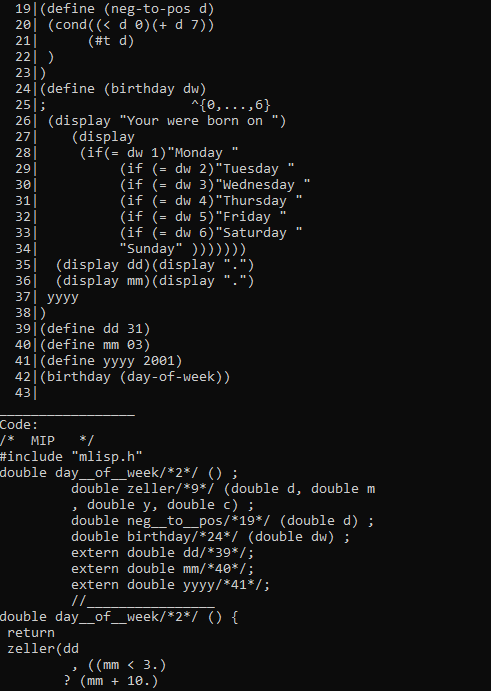
**Шаблон файла code-gen.cpp создайте с помощью приложения Make-code-gen.cpp .**

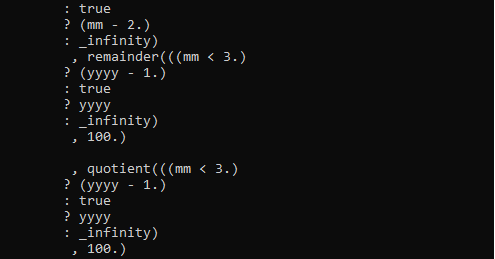
**Вариант грамматики: j21**

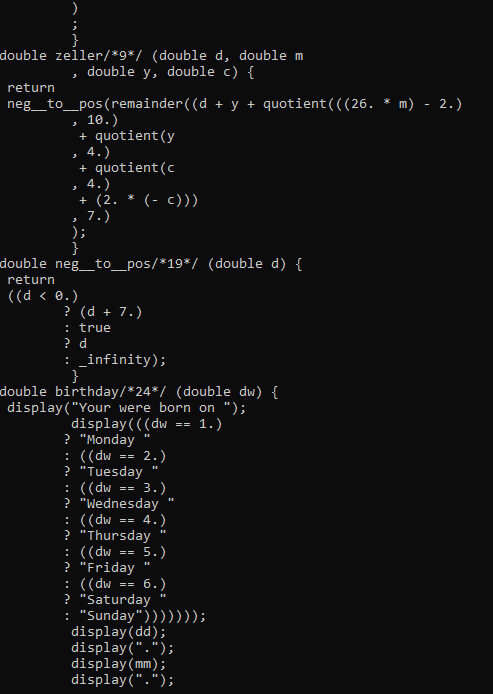
**Контрольная задача №1 – zeller.**

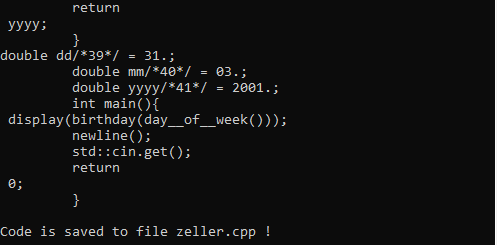
**Полный скриншот трансляции без трассировки.**

****

****

****

****

****

**Распечатка файла zeller.cpp .**

***/\*  MIP   \*/***

**#include "mlisp.h"**

**double day\_\_of\_\_week*/\*2\*/* () ;**

**double zeller*/\*9\*/* (double d, double m**

**, double y, double c) ;**

**double neg\_\_to\_\_pos*/\*19\*/* (double d) ;**

**double birthday*/\*24\*/* (double dw) ;**

**extern double dd*/\*39\*/*;**

**extern double mm*/\*40\*/*;**

**extern double yyyy*/\*41\*/*;**

***//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**double day\_\_of\_\_week*/\*2\*/* () {**

**return**

**zeller(dd**

**, ((mm < 3.)**

**? (mm + 10.)**

**: true**

**? (mm - 2.)**

**: \_infinity)**

**, remainder(((mm < 3.)**

**? (yyyy - 1.)**

**: true**

**? yyyy**

**: \_infinity)**

**, 100.)**

**, quotient(((mm < 3.)**

**? (yyyy - 1.)**

**: true**

**? yyyy**

**: \_infinity)**

**, 100.)**

**)**

**;**

**}**

**double zeller*/\*9\*/* (double d, double m**

**, double y, double c) {**

**return**

**neg\_\_to\_\_pos(remainder((d + y + quotient(((26. \* m) - 2.)**

**, 10.)**

**+ quotient(y**

**, 4.)**

**+ quotient(c**

**, 4.)**

**+ (2. \* (- c)))**

**, 7.)**

**);**

**}**

**double neg\_\_to\_\_pos*/\*19\*/* (double d) {**

**return**

**((d < 0.)**

**? (d + 7.)**

**: true**

**? d**

**: \_infinity);**

**}**

**double birthday*/\*24\*/* (double dw) {**

**display("Your were born on ");**

**display(((dw == 1.)**

**? "Monday "**

**: ((dw == 2.)**

**? "Tuesday "**

**: ((dw == 3.)**

**? "Wednesday "**

**: ((dw == 4.)**

**? "Thursday "**

**: ((dw == 5.)**

**? "Friday "**

**: ((dw == 6.)**

**? "Saturday "**

**: "Sunday")))))));**

**display(dd);**

**display(".");**

**display(mm);**

**display(".");**

**return**

**yyyy;**

**}**

**double dd*/\*39\*/* = 31.;**

**double mm*/\*40\*/* = 03.;**

**double yyyy*/\*41\*/* = 2001.;**

**int main(){**

**display(birthday(day\_\_of\_\_week()));**

**newline();**

**std::cin.get();**

**return**

**0;**

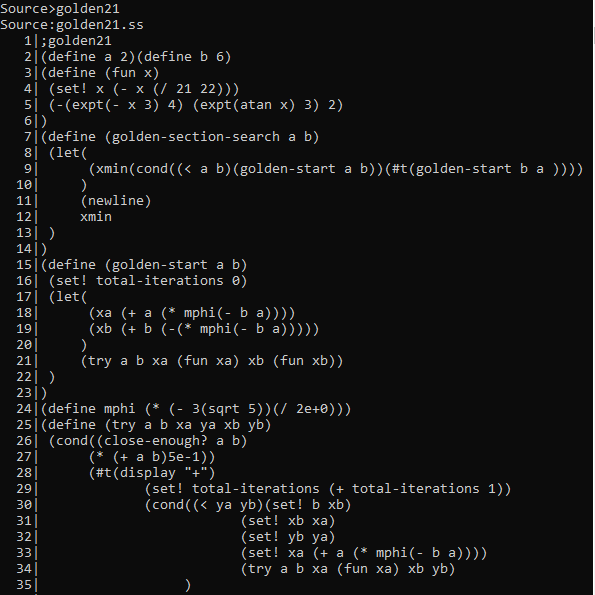
**}**

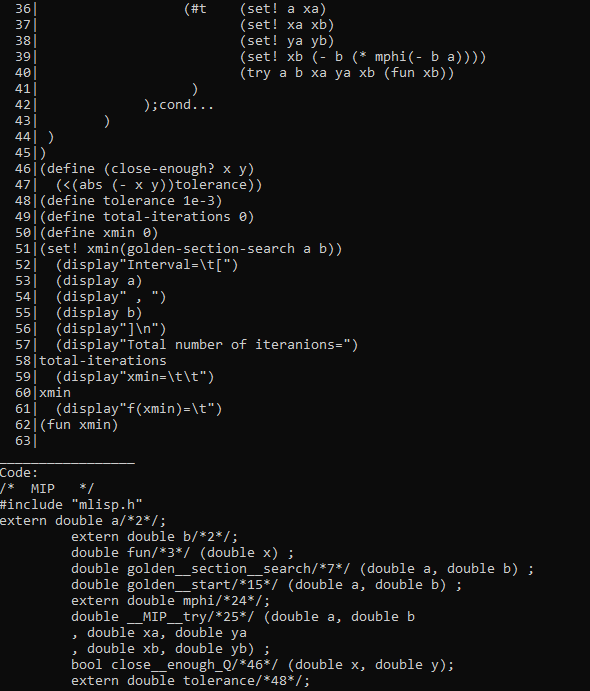
**Скриншот запуска задачи на С++.**

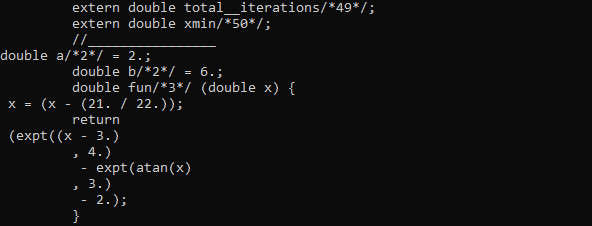
****

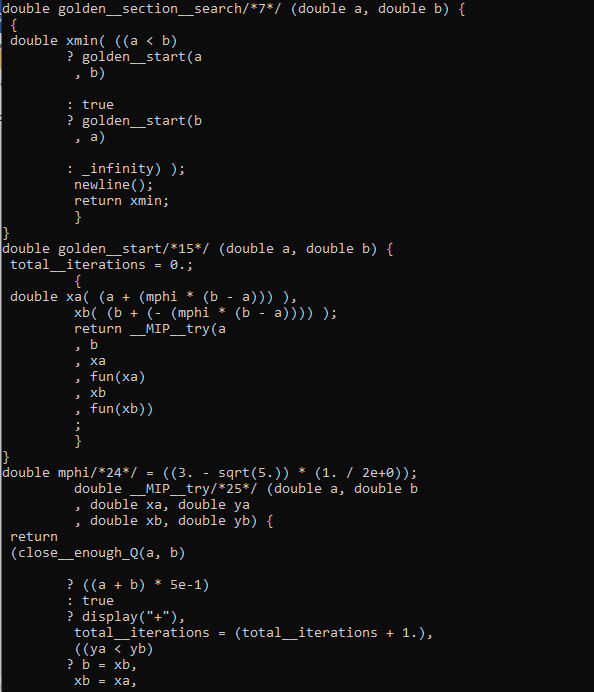
**Контрольная задача №2 – golden21.**

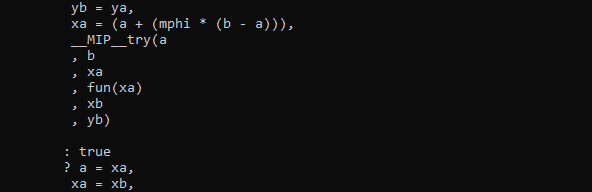
**Полный скриншот трансляции без трассировки.**

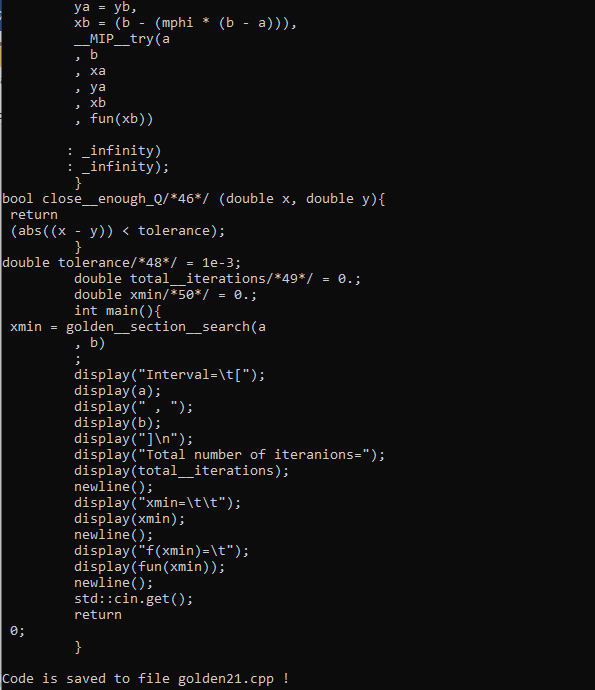
****

****

****

****

****

****

**Распечатка файла golden21.cpp .**

***/\*  MIP   \*/***

**#include "mlisp.h"**

**extern double a*/\*2\*/*;**

**extern double b*/\*2\*/*;**

**double fun*/\*3\*/* (double x) ;**

**double golden\_\_section\_\_search*/\*7\*/* (double a, double b) ;**

**double golden\_\_start*/\*15\*/* (double a, double b) ;**

**extern double mphi*/\*24\*/*;**

**double \_\_MIP\_\_try*/\*25\*/* (double a, double b**

**, double xa, double ya**

**, double xb, double yb) ;**

**bool close\_\_enough\_Q*/\*46\*/* (double x, double y);**

**extern double tolerance*/\*48\*/*;**

**extern double total\_\_iterations*/\*49\*/*;**

**extern double xmin*/\*50\*/*;**

***//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**double a*/\*2\*/* = 2.;**

**double b*/\*2\*/* = 6.;**

**double fun*/\*3\*/* (double x) {**

**x = (x - (21. / 22.));**

**return**

**(expt((x - 3.)**

**, 4.)**

**- expt(atan(x)**

**, 3.)**

**- 2.);**

**}**

**double golden\_\_section\_\_search*/\*7\*/* (double a, double b) {**

**{**

**double xmin( ((a < b)**

**? golden\_\_start(a**

**, b)**

**: true**

**? golden\_\_start(b**

**, a)**

**: \_infinity) );**

**newline();**

**return xmin;**

**}**

**}**

**double golden\_\_start*/\*15\*/* (double a, double b) {**

**total\_\_iterations = 0.;**

**{**

**double xa( (a + (mphi \* (b - a))) ),**

**xb( (b + (- (mphi \* (b - a)))) );**

**return \_\_MIP\_\_try(a**

**, b**

**, xa**

**, fun(xa)**

**, xb**

**, fun(xb))**

**;**

**}**

**}**

**double mphi*/\*24\*/* = ((3. - sqrt(5.)) \* (1. / 2e+0));**

**double \_\_MIP\_\_try*/\*25\*/* (double a, double b**

**, double xa, double ya**

**, double xb, double yb) {**

**return**

**(close\_\_enough\_Q(a, b)**

**? ((a + b) \* 5e-1)**

**: true**

**? display("+"),**

**total\_\_iterations = (total\_\_iterations + 1.),**

**((ya < yb)**

**? b = xb,**

**xb = xa,**

**yb = ya,**

**xa = (a + (mphi \* (b - a))),**

**\_\_MIP\_\_try(a**

**, b**

**, xa**

**, fun(xa)**

**, xb**

**, yb)**

**: true**

**? a = xa,**

**xa = xb,**

**ya = yb,**

**xb = (b - (mphi \* (b - a))),**

**\_\_MIP\_\_try(a**

**, b**

**, xa**

**, ya**

**, xb**

**, fun(xb))**

**: \_infinity)**

**: \_infinity);**

**}**

**bool close\_\_enough\_Q*/\*46\*/* (double x, double y){**

**return**

**(abs((x - y)) < tolerance);**

**}**

**double tolerance*/\*48\*/* = 1e-3;**

**double total\_\_iterations*/\*49\*/* = 0.;**

**double xmin*/\*50\*/* = 0.;**

**int main(){**

**xmin = golden\_\_section\_\_search(a**

**, b)**

**;**

**display("Interval=\t[");**

**display(a);**

**display(" , ");**

**display(b);**

**display("]\n");**

**display("Total number of iteranions=");**

**display(total\_\_iterations);**

**newline();**

**display("xmin=\t\t");**

**display(xmin);**

**newline();**

**display("f(xmin)=\t");**

**display(fun(xmin));**

**newline();**

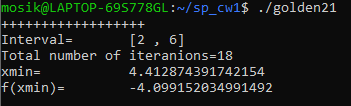
**std::cin.get();**

**return**

**0;**

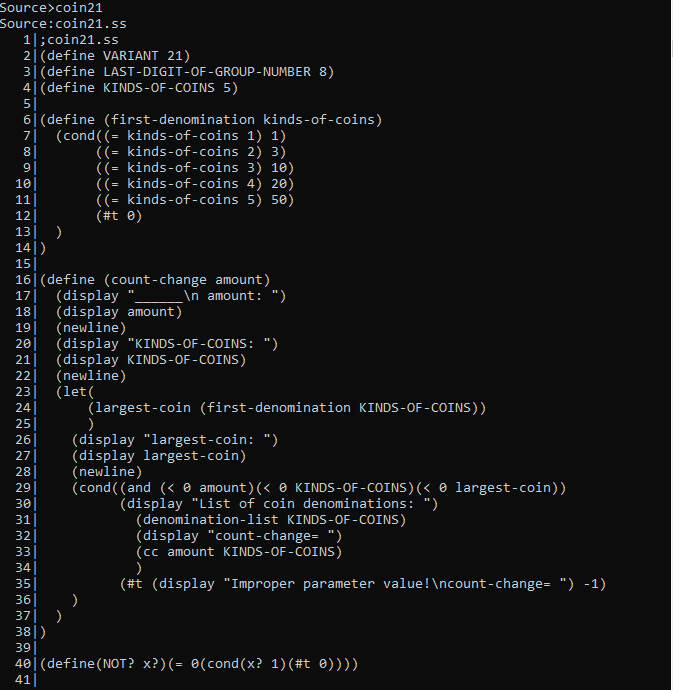
**}**

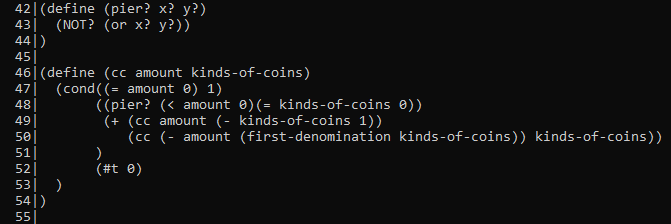
**Скриншот запуска задачи на С++.**

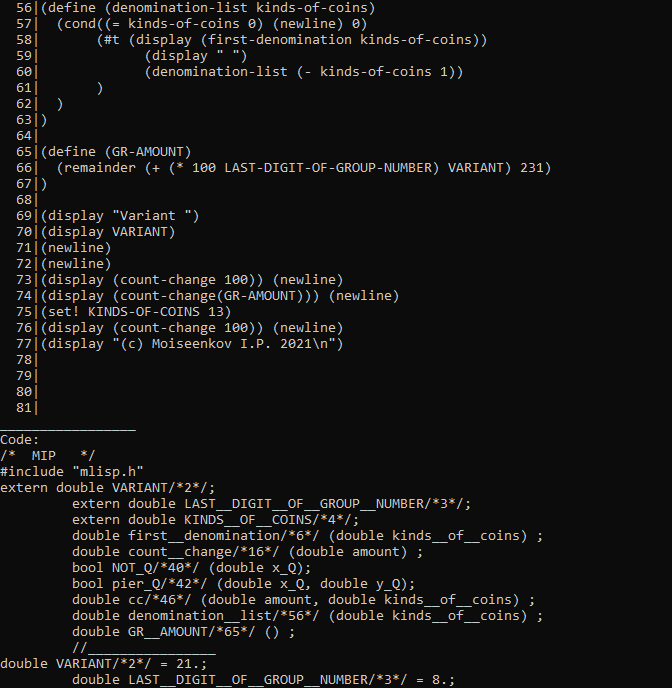
****

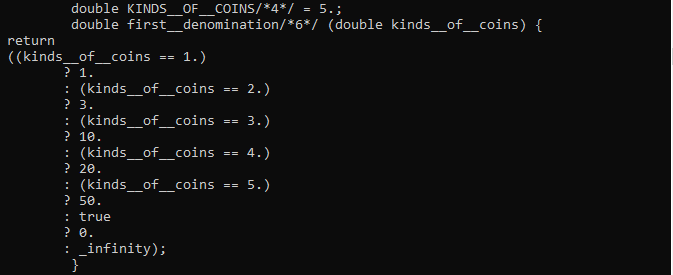
**Контрольная задача №3 – coin21.**

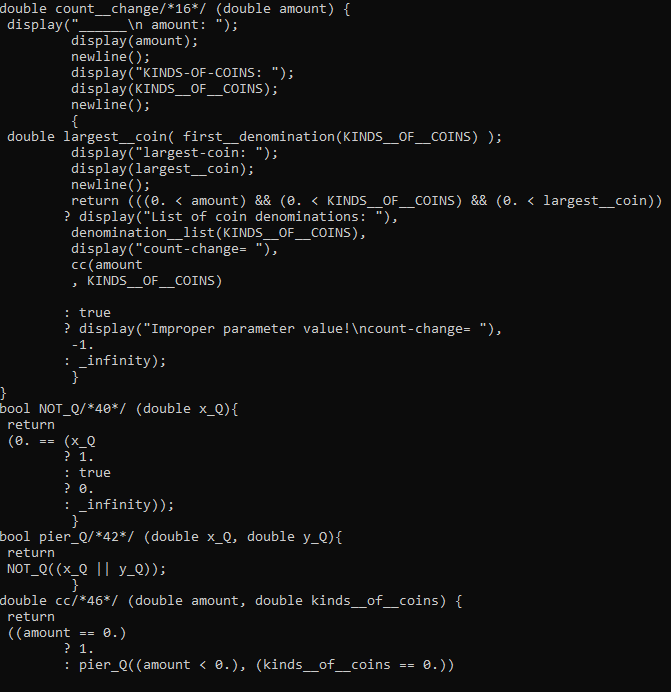
**Полный скриншот трансляции без трассировки.**

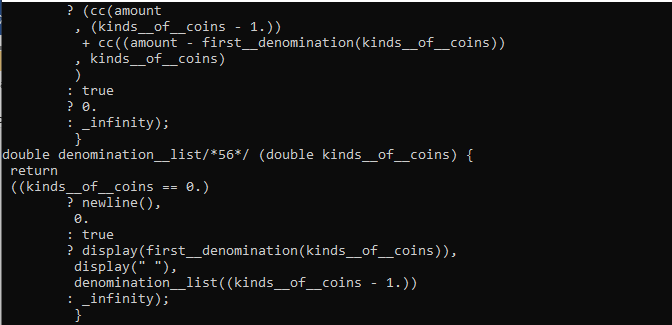
****

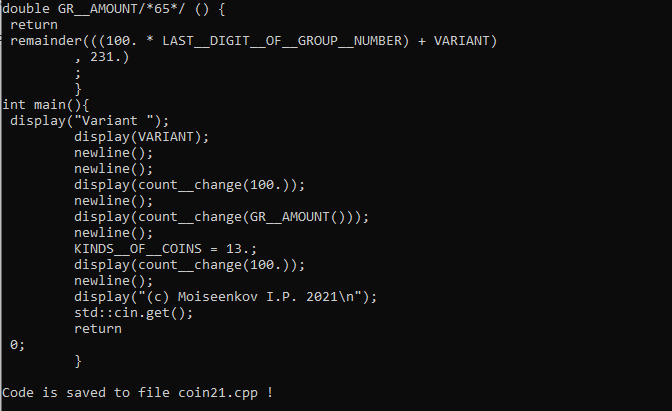
****

****

****

****

****

****

**Распечатка файла coin21.cpp .**

***/\*  MIP   \*/***

**#include "mlisp.h"**

**extern double VARIANT*/\*2\*/*;**

**extern double LAST\_\_DIGIT\_\_OF\_\_GROUP\_\_NUMBER*/\*3\*/*;**

**extern double KINDS\_\_OF\_\_COINS*/\*4\*/*;**

**double first\_\_denomination*/\*6\*/* (double kinds\_\_of\_\_coins) ;**

**double count\_\_change*/\*16\*/* (double amount) ;**

**bool NOT\_Q*/\*40\*/* (double x\_Q);**

**bool pier\_Q*/\*42\*/* (double x\_Q, double y\_Q);**

**double cc*/\*46\*/* (double amount, double kinds\_\_of\_\_coins) ;**

**double denomination\_\_list*/\*56\*/* (double kinds\_\_of\_\_coins) ;**

**double GR\_\_AMOUNT*/\*65\*/* () ;**

***//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**double VARIANT*/\*2\*/* = 21.;**

**double LAST\_\_DIGIT\_\_OF\_\_GROUP\_\_NUMBER*/\*3\*/* = 8.;**

**double KINDS\_\_OF\_\_COINS*/\*4\*/* = 5.;**

**double first\_\_denomination*/\*6\*/* (double kinds\_\_of\_\_coins) {**

**return**

**((kinds\_\_of\_\_coins == 1.)**

**? 1.**

**: (kinds\_\_of\_\_coins == 2.)**

**? 3.**

**: (kinds\_\_of\_\_coins == 3.)**

**? 10.**

**: (kinds\_\_of\_\_coins == 4.)**

**? 20.**

**: (kinds\_\_of\_\_coins == 5.)**

**? 50.**

**: true**

**? 0.**

**: \_infinity);**

**}**

**double count\_\_change*/\*16\*/* (double amount) {**

**display("\_\_\_\_\_\_\n amount: ");**

**display(amount);**

**newline();**

**display("KINDS-OF-COINS: ");**

**display(KINDS\_\_OF\_\_COINS);**

**newline();**

**{**

**double largest\_\_coin( first\_\_denomination(KINDS\_\_OF\_\_COINS) );**

**display("largest-coin: ");**

**display(largest\_\_coin);**

**newline();**

**return (((0. < amount) && (0. < KINDS\_\_OF\_\_COINS) && (0. < largest\_\_coin))**

**? display("List of coin denominations: "),**

**denomination\_\_list(KINDS\_\_OF\_\_COINS),**

**display("count-change= "),**

**cc(amount**

**, KINDS\_\_OF\_\_COINS)**

**: true**

**? display("Improper parameter value!\ncount-change= "),**

**-1.**

**: \_infinity);**

**}**

**}**

**bool NOT\_Q*/\*40\*/* (double x\_Q){**

**return**

**(0. == (x\_Q**

**? 1.**

**: true**

**? 0.**

**: \_infinity));**

**}**

**bool pier\_Q*/\*42\*/* (double x\_Q, double y\_Q){**

**return**

**NOT\_Q((x\_Q || y\_Q));**

**}**

**double cc*/\*46\*/* (double amount, double kinds\_\_of\_\_coins) {**

**return**

**((amount == 0.)**

**? 1.**

**: pier\_Q((amount < 0.), (kinds\_\_of\_\_coins == 0.))**

**? (cc(amount**

**, (kinds\_\_of\_\_coins - 1.))**

**+ cc((amount - first\_\_denomination(kinds\_\_of\_\_coins))**

**, kinds\_\_of\_\_coins)**

**)**

**: true**

**? 0.**

**: \_infinity);**

**}**

**double denomination\_\_list*/\*56\*/* (double kinds\_\_of\_\_coins) {**

**return**

**((kinds\_\_of\_\_coins == 0.)**

**? newline(),**

**0.**

**: true**

**? display(first\_\_denomination(kinds\_\_of\_\_coins)),**

**display(" "),**

**denomination\_\_list((kinds\_\_of\_\_coins - 1.))**

**: \_infinity);**

**}**

**double GR\_\_AMOUNT*/\*65\*/* () {**

**return**

**remainder(((100. \* LAST\_\_DIGIT\_\_OF\_\_GROUP\_\_NUMBER) + VARIANT)**

**, 231.)**

**;**

**}**

**int main(){**

**display("Variant ");**

**display(VARIANT);**

**newline();**

**newline();**

**display(count\_\_change(100.));**

**newline();**

**display(count\_\_change(GR\_\_AMOUNT()));**

**newline();**

**KINDS\_\_OF\_\_COINS = 13.;**

**display(count\_\_change(100.));**

**newline();**

**display("(c) Moiseenkov I.P. 2021\n");**

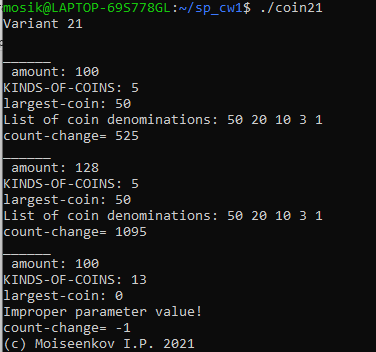
**std::cin.get();**

**return**

**0;**

**}**

**Скриншот запуска задачи на С++.**

****

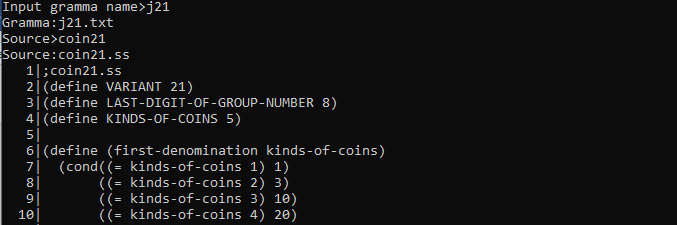
**Распечатка файла code-gen.cpp.**

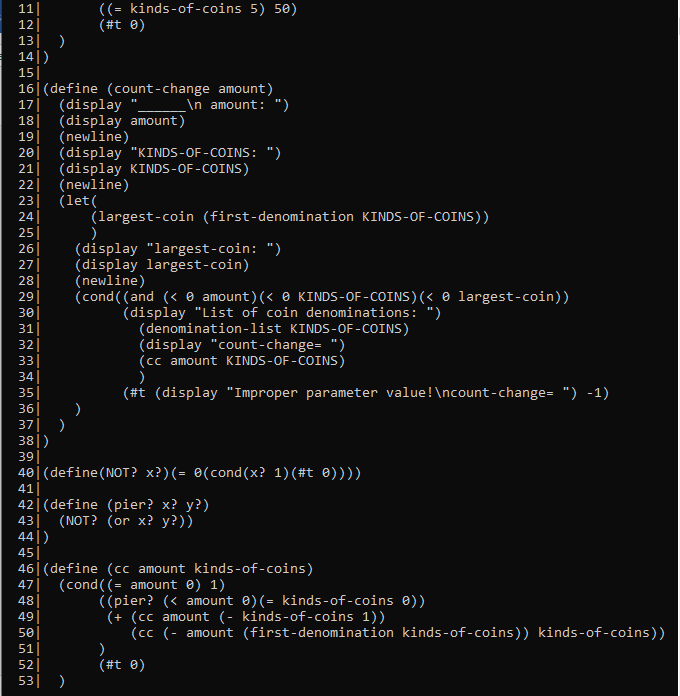
**/\* $j21 \*/  
#include "code-gen.h"  
using namespace std;  
void tCG::init(){declarations.clear();  
 Authentication = "MIP";  
}  
int tCG::p01(){ // S -> PROG  
 string header ="/\* " + Authentication +" \*/\n";  
 header += "#include \"mlisp.h\"\n";  
 header += declarations;  
 header += "//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n";  
 S1->obj = header + S1->obj;  
 return 0;  
}  
int tCG::p02(){ // PROG -> CALCS  
 S1->obj = "int main(){\n " + S1->obj + "std::cin.get();\n\t return\n 0;\n\t }\n";  
 return 0;  
}  
int tCG::p03(){ // PROG -> DEFS  
 S1->obj += "int main(){\n "  
 "display(\"No calculations!\");\n\t newline();\n\t "  
 "std::cin.get();\n\t return\n 0;\n\t }\n";  
 return 0;  
}  
int tCG::p04(){ // PROG -> DEFS CALCS  
 S1->obj += "int main(){\n " + S2->obj + "std::cin.get();\n\t return\n 0;\n\t }\n";  
 return 0;  
}  
int tCG::p05(){ // E -> $id  
 S1->obj = decor(S1->name);  
 return 0;  
}  
int tCG::p06(){ // E -> $int  
 S1->obj = S1->name + ".";  
 return 0;  
}  
int tCG::p07(){ // E -> $dec  
 S1->obj = S1->name;  
 return 0;  
}  
int tCG::p08(){ // E -> AREX  
 return 0;  
}  
int tCG::p09(){ // E -> COND  
 return 0;  
}  
int tCG::p10(){ // E -> CPROC  
 return 0;  
}  
int tCG::p11(){ // CPROC -> HCPROC )  
 S1->obj += ")";  
 if (S1->count >= 2) {  
 S1->obj += "\n\t ";  
 }  
 return 0;  
}  
int tCG::p12(){ // HCPROC -> ( $id  
 S1->obj = decor(S2->name) + "(";  
 return 0;  
}  
int tCG::p13(){ // HCPROC -> HCPROC E  
 if (S1->count) {  
 S1->obj += "\n\t , ";  
 }  
 S1->obj += S2->obj;  
 ++S1->count;  
 return 0;  
}  
int tCG::p14(){ // AREX -> HAREX E )  
 if (S1->name == "/" && S1->count == 0) {  
 S1->obj = "(1. " + S1->obj + " " + S2->obj + ")";  
 }  
 else {  
 S1->obj = "(" + S1->obj + " " + S2->obj + ")";  
 }  
 return 0;  
}  
int tCG::p15(){ // HAREX -> ( AROP  
 S1->obj = S2->obj;  
 S1->name = S2->name;  
 return 0;  
}  
int tCG::p16(){ // HAREX -> HAREX E  
 if (S1->count == 0)  
 S1->obj = S2->obj + " " + S1->name;  
 else  
 S1->obj = S1->obj + " " + S2->obj + " " + S1->name;  
 ++S1->count;  
 return 0;  
}  
int tCG::p17(){ // AROP -> +  
 S1->obj = S1->name;  
 return 0;  
}  
int tCG::p18(){ // AROP -> -  
 S1->obj = S1->name;  
 return 0;  
}  
int tCG::p19(){ // AROP -> \*  
 S1->obj = S1->name;  
 return 0;  
}  
int tCG::p20(){ // AROP -> /  
 S1->obj = S1->name;  
 return 0;  
}  
int tCG::p21(){ // COND -> ( cond BRANCHES )  
 S1->obj = "(" + S3->obj + "\_infinity)";  
 return 0;  
}  
int tCG::p22(){ // BRANCHES -> CLAUS  
 return 0;  
}  
int tCG::p23(){ // BRANCHES -> CLAUS BRANCHES  
 S1->obj += S2->obj;  
 return 0;  
}  
int tCG::p24(){ // CLAUS -> ( BOOL CLAUSB )  
 S1->obj += S2->obj + "\n\t? " + S3->obj + "\n\t: ";  
 return 0;  
}  
int tCG::p25(){ // CLAUSB -> E  
 return 0;  
}  
int tCG::p26(){ // CLAUSB -> INTER CLAUSB  
 S1->obj += ",\n\t " + S2->obj;  
 ++S1->count;  
 return 0;  
}  
int tCG::p27(){ // STR -> $str  
 S1->obj = S1->name;  
 return 0;  
}  
int tCG::p28(){ // STR -> SIF  
 return 0;  
}  
int tCG::p29(){ // SIF -> ( if BOOL STR STR )  
 S1->obj = "(" + S3->obj + "\n\t? " + S4->obj + "\n\t: " + S5->obj + ")";  
 return 0;  
}  
int tCG::p30(){ // BOOL -> $bool  
 S1->obj = (S1->name == "#t" ? "true" : "false");  
 return 0;  
}  
int tCG::p31(){ // BOOL -> $idq  
 S1->obj = decor(S1->name);  
 return 0;  
}  
int tCG::p32(){ // BOOL -> REL  
 return 0;  
}  
int tCG::p33(){ // BOOL -> OR  
 return 0;  
}  
int tCG::p34(){ // BOOL -> AND  
 return 0;  
}  
int tCG::p35(){ // BOOL -> CPRED  
 return 0;  
}  
int tCG::p36(){ // CPRED -> HCPRED )  
 S1->obj += ")";  
 if (S1->count >= 2) {  
 S1->obj += "\n\t ";  
 }  
 return 0;  
}  
int tCG::p37(){ // HCPRED -> ( $idq  
 S1->obj = decor(S2->name) + "(";  
 return 0;  
}  
int tCG::p38(){ // HCPRED -> HCPRED ARG  
 if (S1->count) {  
 S1->obj += S1->count % 2 ? ", " : "\n\t , ";  
 }  
 S1->obj += S2->obj;  
 ++S1->count;  
 return 0;  
}  
int tCG::p39(){ // ARG -> E  
 return 0;  
}  
int tCG::p40(){ // ARG -> BOOL  
 return 0;  
}  
int tCG::p41(){ // REL -> ( = E E )  
 S1->obj = "(" + S3->obj + " == " + S4->obj + ")";  
 return 0;  
}  
int tCG::p42(){ // REL -> ( < E E )  
 S1->obj = "(" + S3->obj + " < " + S4->obj + ")";  
 return 0;  
}  
int tCG::p43(){ // OR -> HOR BOOL )  
 S1->obj = "(" + S1->obj + S2->obj + ")";  
 return 0;  
}  
int tCG::p44(){ // HOR -> ( or  
 return 0;  
}  
int tCG::p45(){ // HOR -> HOR BOOL  
 S1->obj += S2->obj + " || ";  
 return 0;  
}  
int tCG::p46(){ // AND -> HAND BOOL )  
 S1->obj = "(" + S1->obj + S2->obj + ")";  
 return 0;  
}  
int tCG::p47(){ // HAND -> ( and  
 return 0;  
}  
int tCG::p48(){ // HAND -> HAND BOOL  
 S1->obj += S2->obj + " && ";  
 return 0;  
}  
int tCG::p49(){ // SET -> HSET E )  
 S1->obj += S2->obj;  
 return 0;  
}  
int tCG::p50(){ // HSET -> ( set! $id  
 S1->obj = decor(S3->name) + " = ";  
 return 0;  
}  
int tCG::p51(){ // DISPSET -> ( display E )  
 S1->obj = "display(" + S3->obj + ")";  
 return 0;  
}  
int tCG::p52(){ // DISPSET -> ( display BOOL )  
 S1->obj = "display(" + S3->obj + ")";  
 return 0;  
}  
int tCG::p53(){ // DISPSET -> ( display STR )  
 S1->obj = "display(" + S3->obj + ")";  
 return 0;  
}  
int tCG::p54(){ // DISPSET -> ( newline )  
 S1->obj = "newline()";  
 return 0;  
}  
int tCG::p55(){ // DISPSET -> SET  
 return 0;  
}  
int tCG::p56(){ // INTER -> DISPSET  
 return 0;  
}  
int tCG::p57(){ // INTER -> E  
 return 0;  
}  
int tCG::p58(){ // CALCS -> CALC  
 return 0;  
}  
int tCG::p59(){ // CALCS -> CALCS CALC  
 S1->obj += S2->obj;  
 return 0;  
}  
int tCG::p60(){ // CALC -> E  
 S1->obj = "display(" + S1->obj + ");\n\t newline();\n\t ";  
 return 0;  
}  
int tCG::p61(){ // CALC -> BOOL  
 S1->obj = "display(" + S1->obj + ");\n\t newline();\n\t ";  
 return 0;  
}  
int tCG::p62(){ // CALC -> STR  
 S1->obj = "display(" + S1->obj + ");\n\t newline();\n\t ";  
 return 0;  
}  
int tCG::p63(){ // CALC -> DISPSET  
 S1->obj += ";\n\t ";  
 return 0;  
}  
int tCG::p64(){ // DEFS -> DEF  
 return 0;  
}  
int tCG::p65(){ // DEFS -> DEFS DEF  
 S1->obj += S2->obj;  
 return 0;  
}  
int tCG::p66(){ // DEF -> PRED  
 return 0;  
}  
int tCG::p67(){ // DEF -> VAR  
 return 0;  
}  
int tCG::p68(){ // DEF -> PROC  
 return 0;  
}  
int tCG::p69(){ // PRED -> HPRED BOOL )  
 S1->obj += "return\n " + S2->obj + ";\n\t }\n";  
 return 0;  
}  
int tCG::p70(){ // HPRED -> PDPAR )  
 S1->obj += ")";  
 declarations += S1->obj + ";\n\t ";  
 S1->obj += "{\n ";  
 return 0;  
}  
int tCG::p71(){ // PDPAR -> ( define ( $idq  
 S1->obj = "bool " + decor(S4->name) + "/\*" + S4->line + "\*/ (";  
 S1->count = 0;  
 return 0;  
}  
int tCG::p72(){ // PDPAR -> PDPAR $idq  
 if (S1->count) {  
 S1->obj += S1->count % 2 ? ", " : "\n\t , ";  
 }  
 S1->obj += "double " + decor(S2->name);  
 ++S1->count;  
 return 0;  
}  
int tCG::p73(){ // PDPAR -> PDPAR $id  
 if (S1->count) {  
 S1->obj += S1->count % 2 ? ", " : "\n\t , ";  
 }  
 S1->obj += "double " + decor(S2->name);  
 ++S1->count;  
 return 0;  
}  
int tCG::p74(){ // VAR -> VARDCL E )  
 declarations += "extern double " + S1->obj + "/\*" + S1->line + "\*/;\n\t ";  
 S1->obj = "double " + S1->obj + "/\*" + S1->line + "\*/ = " + S2->obj + ";\n\t ";  
 return 0;  
}  
int tCG::p75(){ // VARDCL -> ( define $id  
 S1->obj = decor(S3->name);  
 return 0;  
}  
int tCG::p76(){ // PROC -> HPROC BLOCK )  
 S1->obj += S2->obj + "}\n";  
 return 0;  
}  
int tCG::p77(){ // PROC -> HPROC E )  
 S1->obj += "return\n " + S2->obj + ";\n\t }\n";  
 return 0;  
}  
int tCG::p78(){ // HPROC -> PCPAR )  
 S1->obj += ") ";  
 declarations += S1->obj + ";\n\t ";  
 S1->obj += "{\n ";  
 return 0;  
}  
int tCG::p79(){ // HPROC -> HPROC INTER  
 S1->obj += S2->obj + ";\n\t ";  
 return 0;  
}  
int tCG::p80(){ // PCPAR -> ( define ( $id  
 S1->obj = "double " + decor(S4->name) + "/\*" + S4->line + "\*/ (";  
 S1->count = 0;  
 S1->name = S4->name;  
 return 0;  
}  
int tCG::p81(){ // PCPAR -> PCPAR $id  
 if (S1->count) {  
 S1->obj += S1->count % 2 ? ", " : "\n\t , ";  
 }  
 S1->obj += "double " + decor(S2->name);  
 ++S1->count;  
 return 0;  
}  
int tCG::p82(){ // BLOCK -> HBLOCK E )  
 S1->obj += "return " + S2->obj +";\n\t }\n";  
 return 0;  
}  
int tCG::p83(){ // HBLOCK -> BLVAR )  
 S1->obj += ";\n\t ";  
 return 0;  
}  
int tCG::p84(){ // HBLOCK -> HBLOCK INTER  
 S1->obj += S2->obj + ";\n\t ";  
 return 0;  
}  
int tCG::p85(){ // BLVAR -> ( let ( LOCDEF  
 S1->obj += "{\n double " + S4->obj;  
 return 0;  
}  
int tCG::p86(){ // BLVAR -> BLVAR LOCDEF  
 S1->obj += ",\n\t " + S2->obj;  
 return 0;  
}  
int tCG::p87(){ // LOCDEF -> ( $id E )  
 S1->obj += decor(S2->name) + "( " + S3->obj + " )";  
 return 0;  
}  
//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
int tCG::p88(){return 0;} int tCG::p89(){return 0;}   
int tCG::p90(){return 0;} int tCG::p91(){return 0;}   
int tCG::p92(){return 0;} int tCG::p93(){return 0;}   
int tCG::p94(){return 0;} int tCG::p95(){return 0;}   
int tCG::p96(){return 0;} int tCG::p97(){return 0;}   
int tCG::p98(){return 0;} int tCG::p99(){return 0;}   
int tCG::p100(){return 0;} int tCG::p101(){return 0;}   
int tCG::p102(){return 0;} int tCG::p103(){return 0;}   
int tCG::p104(){return 0;} int tCG::p105(){return 0;}   
int tCG::p106(){return 0;} int tCG::p107(){return 0;}   
int tCG::p108(){return 0;} int tCG::p109(){return 0;}   
int tCG::p110(){return 0;}**

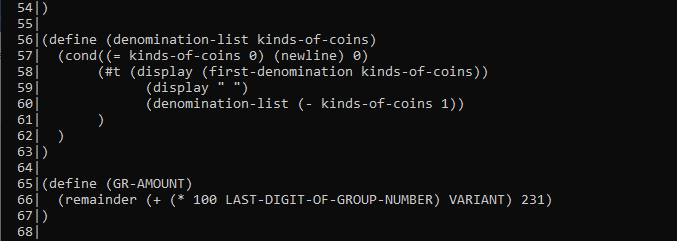
**ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ**

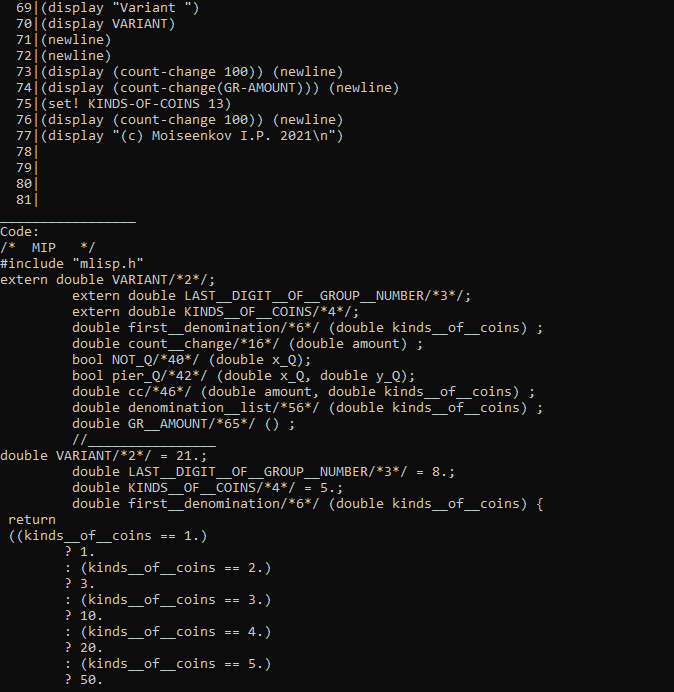
**После определения БУЛЕВСКОЙ функции поместить  
  комментарий с именем ее прототипа в Микролиспе.  
  Например, bool pier\_Q/\*33\*/(...){  
           ...  
          }//pier?  
Продукции могут получать имена ТОЛЬКО из атрибута name и НЕ могут извлекать их из кода на С++. Продемонстрировать на контрольной задаче coin21.**

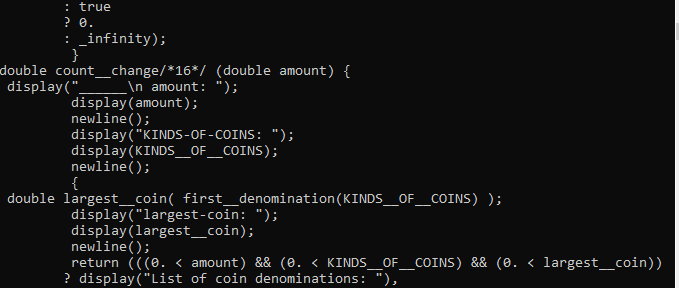
**Скриншот трансляции coin21**

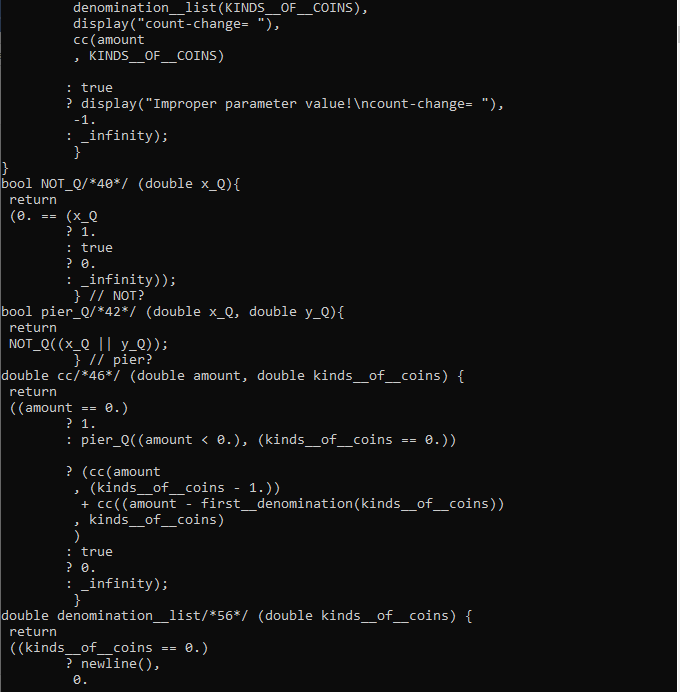
****

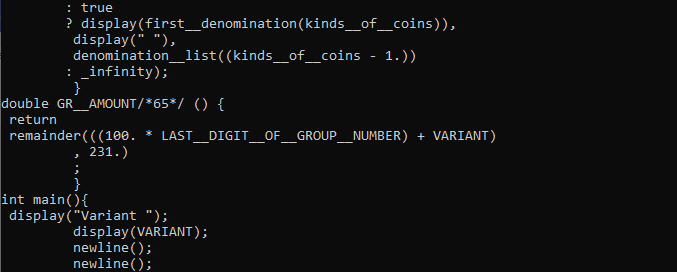
****

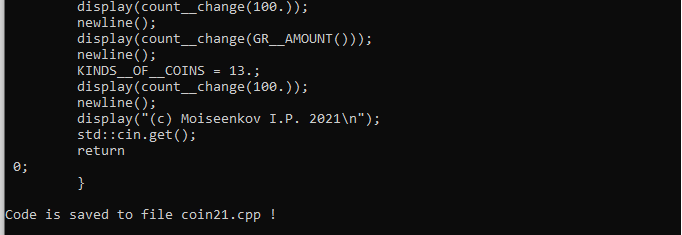
****

****

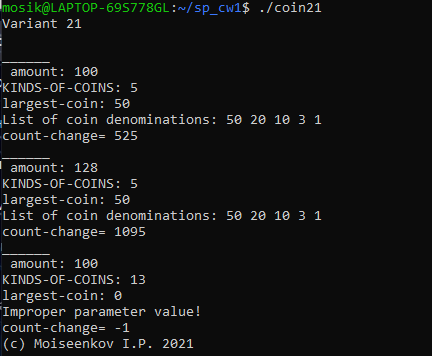
****

****

****

****

**Скриншот запуска на С++**

****

**Распечатка ИЗМЕНЕННЫХ продукций**

**int tCG::p69(){ // PRED -> HPRED BOOL )  
 S1->obj += "return\n " + S2->obj + ";\n\t } // " + S1->name + "\n";  
 return 0;  
}**

**int tCG::p71(){ // PDPAR -> ( define ( $idq  
 S1->obj = "bool " + decor(S4->name) + "/\*" + S4->line + "\*/ (";  
 S1->count = 0;  
 S1->name = S4->name;  
 return 0;  
}**