**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ**

**И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

(СПбГУТ)

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ (**ИТПИ**)

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ (ПИ И ВТ)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа №2**

по дисциплине «Разработка приложений искусственного интеллекта в киберфизических системах»

Тема «Решение кубического уравнения комбинированным методом хорд и касательных»

Выполнил: Хайдаршин К.Р.

Группа ИКПИ-23

Принял: Ерофеев С.А.

Подпись\_\_\_\_\_\_\_\_\_\_\_

г.Санкт-Петербург

2024 г.

Оглавление

[**Постановка задачи** 3](#_Toc180666861)

[**Алгоритм решения** 3](#_Toc180666862)

[**Проверка работоспособности программы** 5](#_Toc180666863)

[**Код программы** 7](#_Toc180666864)

[**Вывод** 10](#_Toc180666865)

**Постановка задачи**

Написать программу на языке PROLOG, которая решает кубическое уравнение вида *ax3+bx2+cx+d=0* комбинированным методом хорд и касательных.

**Алгоритм решения**

Комбинированный метод хорд и касательных сочетает в себе принципы метода хорд и метода касательных и позволяет решать нелинейные уравнения с заданной точностью.

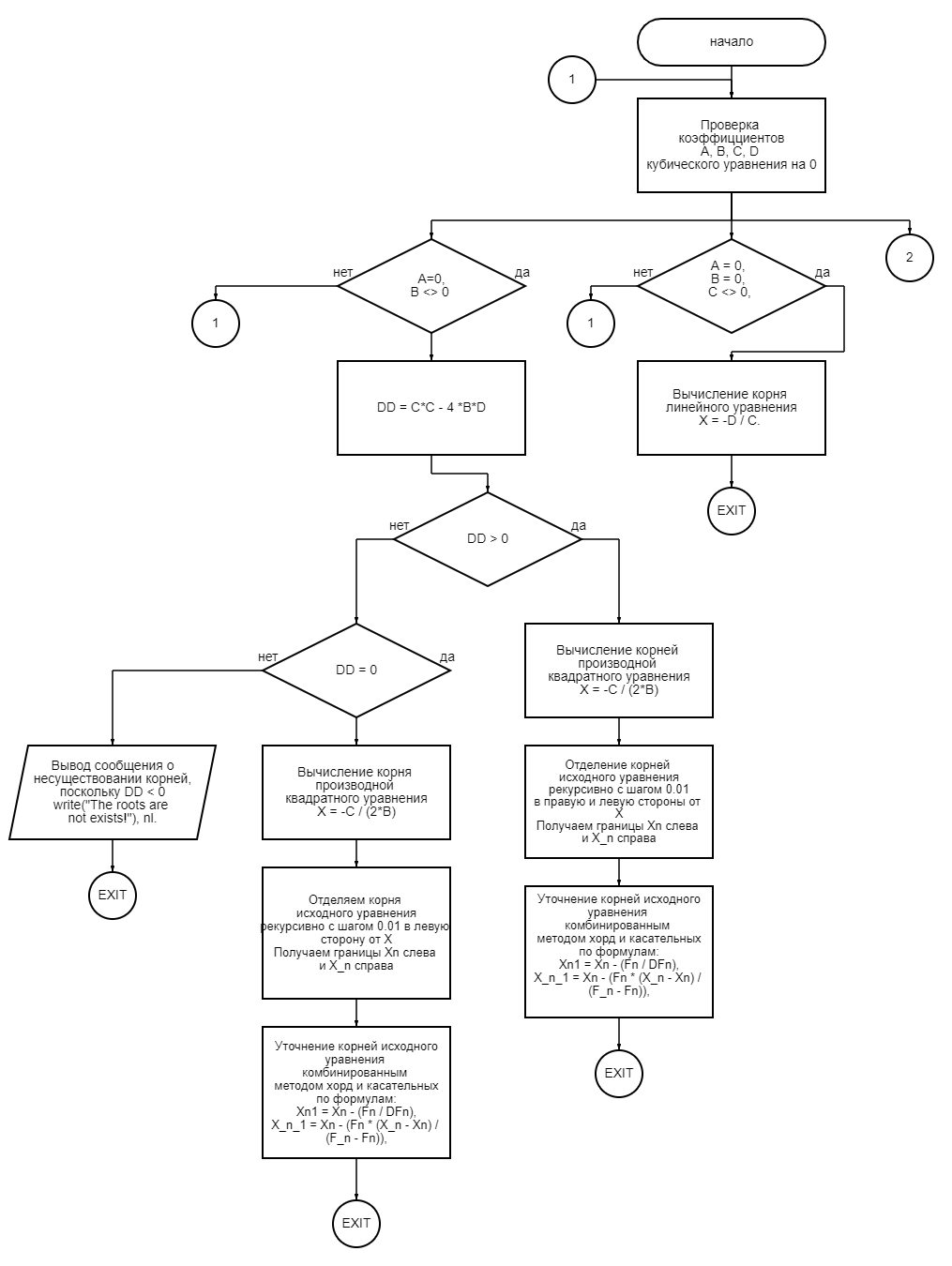
Приближение к искомому корню происходит одновременно с двух сторон отрезка, на котором отделён корень уравнения. Совместное использование методов позволяет на каждой итерации находить приближённые значения с недостатком и с избытком, что ускоряет процесс сходимости.

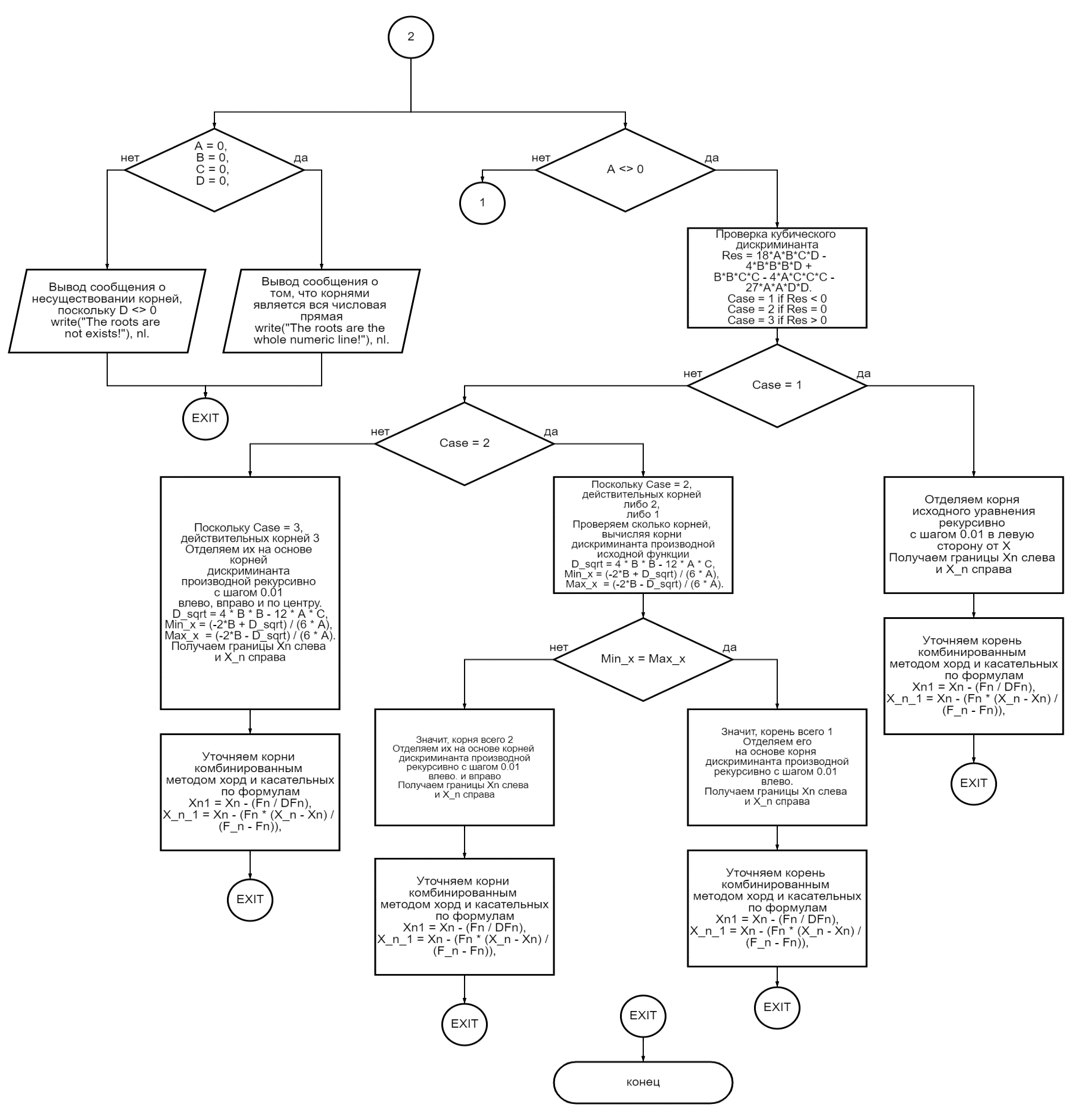
Алгоритм решения кратко представлен на рисунке 1.

В таблице 1 приведено описание предикатов, используемых в программе.

|  |  |
| --- | --- |
| Предикат | Описание |
| f(x, x, x, x, x, x) | Основная функция |
| df(x, x, x, x, x) | Первая производная основной функции |
| second\_df(x, x, x, x) | Вторая производная основной функции |
| root\_df\_sq(x, x, x) | Корни уравнения при условии равенства нулю коэффициента a |
| f\_discr\_sq(x, x, x, x) | Дискриминант основного уравнения при условии равенства нулю коэффициента a |
| root\_f(x, x, x) | Корень основного уравнения при условии его линейности |
| discr(x, x, x, x) | Дискриминант первой производной основной функции |
| cubic\_discr(x, x, x, x, x) | Кубический дискриминант |
| square\_root(x, x) | Квадратный корень |
| roots\_df(x, x, x, x, x) | Корни первой производной |
| min\_max\_root\_df(x, x, x, x, x) | Минимальный и максимальный корни первой производной |
| plus\_one(x, x) | Прибавление 0.01 |
| minus\_one(x, x) | Убавление 0.01 |
| mul(x, x, x) | Умножение |
| check\_cubic\_discr(x, x, x, x, x) | Проверка знака кубического дискриминанта для выявления одного из 3 случаев |
| check\_sign\_interval\_left(x, x, x, x, x, x, x) | Отделение левого корня |
| check\_sign\_interval\_right(x, x, x, x, x, x, x) | Отделение центрального и правого корней |
| newton\_chord\_method(x, x, x, x, x, x, x, x) | Комбинированный метод хорд и касательных |
| method\_loop(x, x, x, x, x, x) | Рекурсия комбинированного метода хорд и касательных |
| main(x, x, x, x) | Управляющий предикат |

*Таблица 1 - Предикаты*

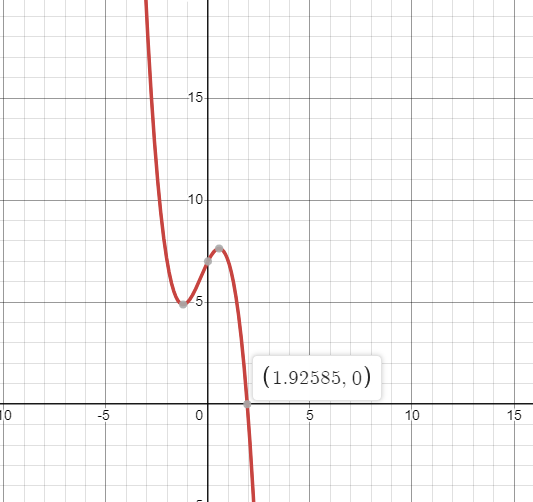
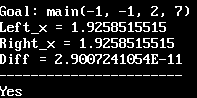


**

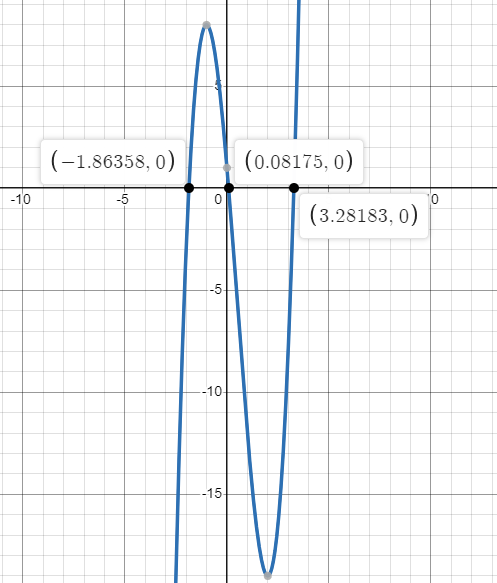
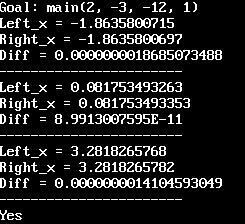
*Рисунок 1 - Алгоритм решения задачи*

**Проверка работоспособности программы**

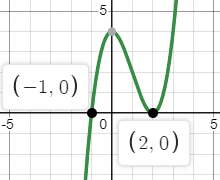
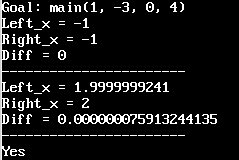
На рисунках 1-9 показаны примеры работы программы с различными входными данными.



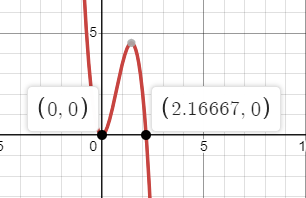
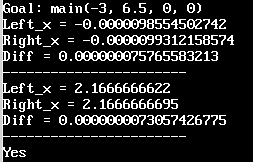
*Рис. 1. Уравнение y=-x3-x2+2x+7*



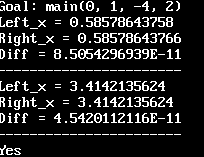
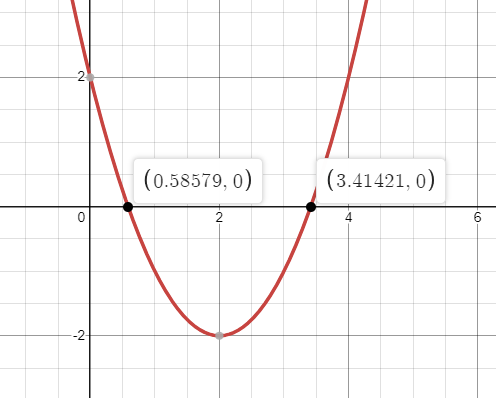
*Рис. 2. Уравнение 2x3-3x2-12x+1=0*



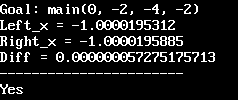
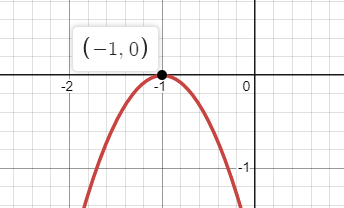
*Рис. 3. Уравнение x3-3x2+4=0*



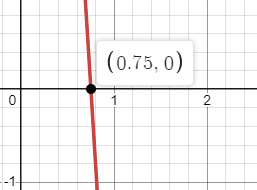
*Рис. 4. Уравнение -3x3+6.5x2=0*

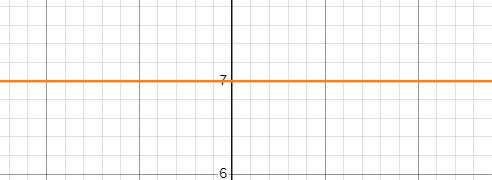
*Рис. 5. Уравнение x2-4x+2=0*

** 

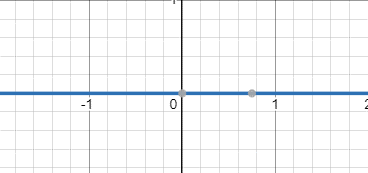
*Рис. 6. Уравнение -2x2-4x-2=0*

** 

*Рис. 7. Уравнение -16x+12=0*

** 

*Рис. 8. Уравнение 7=0*

** 

*Рис. 9. Уравнение 0=0*

**Код программы**

CODE=9000

HEAP=9000

NOWARNINGS

DOMAINS

x = **real**

PREDICATES

f(x, x, x, x, x, x)

df(x, x, x, x, x)

second\_df(x, x, x, x)

root\_df\_sq(x, x, x)

f\_discr\_sq(x, x, x, x)

root\_f(x, x, x)

discr(x, x, x, x)

cubic\_discr(x, x, x, x, x)

square\_root(x, x)

roots\_df(x, x, x, x, x)

min\_max\_root\_df(x, x, x, x, x)

plus\_one(x, x)

minus\_one(x, x)

mul(x, x, x)

check\_cubic\_discr(x, x, x, x, x)

check\_sign\_interval\_left(x, x, x, x, x, x, x)

check\_sign\_interval\_right(x, x, x, x, x, x, x)

newton\_chord\_method(x, x, x, x, x, x, x, x)

method\_loop(x, x, x, x, x, x)

main(x, x, x, x)

CLAUSES

f(X, F, A, B, C, D) :- F = A\*X\*X\*X + B\*X\*X + C\*X + D.

df(X, DF, A, B, C) :- DF = 3\*A\*X\*X + 2\*B\*X + C.

second\_df(X, A, B, SDF) :- SDF = 6\*A\*X + 2\*B.

root\_df\_sq(X, B, C) :- X = -C / (2\*B).

f\_discr\_sq(DD, B, C, D) :- DD = **round**(C \* C - 4 \* B \* D).

root\_f(X, C, D) :- X = -D / C.

discr(D, A, B, C) :- D = **round**(4 \* B \* B - 12 \* A \* C).

cubic\_discr(A, B, C, D, Res) :- Res = 18\*A\*B\*C\*D -

4\*B\*B\*B\*D + B\*B\*C\*C - 4\*A\*C\*C\*C -

27\*A\*A\*D\*D.

square\_root(Var, X) :- X = **sqrt**(Var).

roots\_df(X1, X2, A, B, C) :-

discr(D, A, B, C),

square\_root(D, D\_sqrt),

X1 = (-2\*B + D\_sqrt) / (6 \* A),

X2 = (-2\*B - D\_sqrt) / (6 \* A).

min\_max\_root\_df(Min\_x, Max\_x, A, B, C) :-

roots\_df(X1, X2, A, B, C),

X1 < X2,

Min\_x = X1,

Max\_x = X2.

min\_max\_root\_df(Min\_x, Max\_x, A, B, C) :-

roots\_df(X1, X2, A, B, C),

X1 = X2,

Min\_x = X1,

Max\_x = X1.

min\_max\_root\_df(Min\_x, Max\_x, A, B, C) :-

roots\_df(X1, X2, A, B, C),

X1 > X2,

Min\_x = X2,

Max\_x = X1.

plus\_one(Var, Res) :- Res = Var + 0.01.

minus\_one(Var, Res) :- Res = Var - 0.01.

mul(X, Y, Res) :- Res = X \* Y.

check\_cubic\_discr(A, B, C, D, Case) :-

cubic\_discr(A, B, C, D, Res),

Res > 0,

Case = 3.

check\_cubic\_discr(A, B, C, D, Case) :-

cubic\_discr(A, B, C, D, Res),

Res = 0,

Case = 2.

check\_cubic\_discr(A, B, C, D, Case) :-

cubic\_discr(A, B, C, D, Res),

Res < 0,

Case = 1.

check\_sign\_interval\_left(Start, End, Out, A, B, C, D) :- %Lp-Lp

f(Start, F\_start, A, B, C, D),

f(End, F\_end, A, B, C, D),

F\_start \* F\_end <= 0,

Out = Start.

check\_sign\_interval\_left(Start, End, Out, A, B, C, D) :-

f(Start, F\_start, A, B, C, D),

f(End, F\_end, A, B, C, D),

F\_start \* F\_end > 0,

New\_start = Start - 0.01,

check\_sign\_interval\_left(New\_start, End, Out, A, B, C, D).

check\_sign\_interval\_right(Start, End, Out, A, B, C, D) :-

f(Start, F\_start, A, B, C, D),

f(End, F\_end, A, B, C, D),

F\_start \* F\_end <= 0,

Out = End.

check\_sign\_interval\_right(Start, End, Out, A, B, C, D) :-

f(Start, F\_start, A, B, C, D),

f(End, F\_end, A, B, C, D),

F\_start \* F\_end > 0,

New\_end = End + 0.01,

check\_sign\_interval\_right(Start, New\_end, Out, A, B, C, D).

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A = 0,

f\_discr\_sq(DD, B, C, D),

DD > 0,

df(Xn, DF, A, B, C),

DF <> 0,

f(Xn, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

df(Xn, DFn, A, B, C),

second\_df(Xn, A, B, SDF),

mul(DFn, SDF, Res),

Res < 0,

Xn1 = Xn - (Fn / DFn),

X\_n\_1 = Xn - (Fn \* (X\_n - Xn) / (F\_n - Fn)),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A = 0,

f\_discr\_sq(DD, B, C, D),

DD > 0,

df(Xn, DF, A, B, C),

DF <> 0,

f(Xn, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

df(Xn, DFn, A, B, C),

second\_df(Xn, A, B, SDF),

mul(DFn, SDF, Res),

Res > 0,

Xn1 = Xn - (Fn \* (X\_n - Xn) / (F\_n - Fn)),

X\_n\_1 = Xn - (Fn / DFn),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A = 0,

f\_discr\_sq(DD, B, C, D),

DD = 0,

df(Xn, DF, A, B, C),

DF = 0,

Xn\_new = Xn - 0.02,

f(Xn\_new, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

df(Xn\_new, DFn, A, B, C),

second\_df(Xn\_new, A, B, SDF),

mul(DFn, SDF, Res),

Res < 0,

Xn1 = Xn\_new - (Fn / DFn),

X\_n\_1 = Xn\_new - (Fn \* (X\_n - Xn\_new) / (F\_n - Fn)),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A = 0,

f\_discr\_sq(DD, B, C, D),

DD = 0,

df(Xn, DF, A, B, C),

DF = 0,

Xn\_new = Xn - 0.02,

f(Xn\_new, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

df(Xn\_new, DFn, A, B, C),

second\_df(Xn\_new, A, B, SDF),

mul(DFn, SDF, Res),

Res > 0,

Xn1 = Xn\_new - (Fn \* (X\_n - Xn\_new) / (F\_n - Fn)),

X\_n\_1 = Xn\_new - (Fn / DFn),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A = 0,

f\_discr\_sq(DD, B, C, D),

DD = 0,

df(Xn, DF, A, B, C),

DF <> 0,

f(Xn, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

df(Xn, DFn, A, B, C),

second\_df(Xn, A, B, SDF),

mul(DFn, SDF, Res),

Res < 0,

Xn1 = Xn - (Fn / DFn),

X\_n\_1 = Xn - (Fn \* (X\_n - Xn) / (F\_n - Fn)),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A = 0,

f\_discr\_sq(DD, B, C, D),

DD = 0,

df(Xn, DF, A, B, C),

DF <> 0,

f(Xn, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

df(Xn, DFn, A, B, C),

second\_df(Xn, A, B, SDF),

mul(DFn, SDF, Res),

Res > 0,

Xn1 = Xn - (Fn \* (X\_n - Xn) / (F\_n - Fn)),

X\_n\_1 = Xn - (Fn / DFn),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A <> 0,

df(Xn, DFn, A, B, C),

DFn = 0,

Xn\_new = Xn - 0.02,

df(Xn\_new, DFn\_new, A, B, C),

f(Xn\_new, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

second\_df(Xn\_new, A, B, SDF),

mul(DFn\_new, SDF, Res),

Res < 0,

Xn1 = Xn\_new - (Fn / DFn\_new),

X\_n\_1 = Xn\_new - (Fn \* (X\_n - Xn\_new) / (F\_n - Fn)),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A <> 0,

df(Xn, DFn, A, B, C),

DFn = 0,

Xn\_new = Xn - 0.02,

df(Xn\_new, DFn\_new, A, B, C),

f(Xn\_new, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

second\_df(Xn\_new, A, B, SDF),

mul(DFn\_new, SDF, Res),

Res > 0,

df(X\_n, DF\_n, A, B, C),

Xn1 = Xn\_new - (Fn \* (X\_n - Xn\_new) / (F\_n - Fn)),

X\_n\_1 = X\_n - (F\_n / DF\_n),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A <> 0,

df(Xn, DFn, A, B, C),

DFn <> 0,

f(Xn, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

second\_df(Xn, A, B, SDF),

mul(DFn, SDF, Res),

Res < 0,

Xn1 = Xn - (Fn / DFn),

X\_n\_1 = Xn - (Fn \* (X\_n - Xn) / (F\_n - Fn)),

Left\_x = Xn1,

Right\_x = X\_n\_1.

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D) :-

A <> 0,

df(Xn, DFn, A, B, C),

DFn <> 0,

df(Xn, DFn, A, B, C),

f(Xn, Fn, A, B, C, D),

f(X\_n, F\_n, A, B, C, D),

second\_df(Xn, A, B, SDF),

mul(DFn, SDF, Res),

Res > 0,

df(X\_n, DF\_n, A, B, C),

Xn1 = Xn - (Fn \* (X\_n - Xn) / (F\_n - Fn)),

X\_n\_1 = X\_n - (F\_n / DF\_n),

Left\_x = Xn1,

Right\_x = X\_n\_1.

method\_loop(Xn, X\_n, A, B, C, D) :-

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D),

Check = **abs**(Right\_x - Left\_x),

Check <= 0.0000001,

**write**("Left\_x = ", Left\_x), **nl**,

**write**("Right\_x = ", Right\_x), **nl**,

**write**("Diff = ", Check), **nl**,

**write**("-----------------------"), **nl**.

method\_loop(Xn, X\_n, A, B, C, D) :-

newton\_chord\_method(Xn, X\_n, Left\_x, Right\_x, A, B, C, D),

Check = **abs**(Right\_x - Left\_x),

Check > 0.0000001,

method\_loop(Left\_x, Right\_x, A, B, C, D).

main(A, B, C, D) :-

A = 0,

B <> 0,

f\_discr\_sq(DD, B, C, D),

DD > 0,

root\_df\_sq(X, B, C),

check\_sign\_interval\_left(X, X, L\_start, A, B, C, D),

check\_sign\_interval\_right(X, X, R\_end, A, B, C, D),

L\_end = L\_start + 0.01,

R\_start = R\_end - 0.01,

method\_loop(L\_start, L\_end, A, B, C, D),

method\_loop(R\_start, R\_end, A, B, C, D).

main(A, B, C, D) :-

A = 0,

B <> 0,

f\_discr\_sq(DD, B, C, D),

DD = 0,

root\_df\_sq(X, B, C),

check\_sign\_interval\_left(X, X, L\_start, A, B, C, D),

L\_end = L\_start + 0.01,

method\_loop(L\_start, L\_end, A, B, C, D).

main(A, B, C, D) :-

A = 0,

B <> 0,

f\_discr\_sq(DD, B, C, D),

DD < 0,

**write**("The roots are not exists!"), **nl**.

main(A, B, C, D) :-

A = 0,

B = 0,

C <> 0,

root\_f(X, C, D),

**write**("The root is ", X), **nl**.

main(A, B, C, D) :-

A = 0,

B = 0,

C = 0,

D <> 0,

**write**("The roots are not exists!"), **nl**.

main(A, B, C, D) :-

A = 0,

B = 0,

C = 0,

D = 0,

**write**("The roots are the whole numeric line!"), **nl**.

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 1,

discr(DD, A, B, C),

DD < 0,

**write**("The method is not suitable for solving this equation!"), **nl**.

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 1,

discr(DD, A, B, C),

DD >= 0,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

Min\_x <> Max\_x,

f(Min\_x, F\_, A, B, C, D),

plus\_one(Min\_x, Df\_x),

df(Df\_x, DF\_, A, B, C),

mul(F\_, DF\_, Res),

Res > 0,

check\_sign\_interval\_right(Max\_x, Max\_x, R\_end, A, B, C, D),

R\_start = R\_end - 0.01,

method\_loop(R\_start, R\_end, A, B, C, D).

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 1,

discr(DD, A, B, C),

DD >= 0,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

Min\_x <> Max\_x,

f(Max\_x, F\_, A, B, C, D),

plus\_one(Max\_x, Df\_x),

df(Df\_x, DF\_, A, B, C),

mul(F\_, DF\_, Res),

Res > 0,

check\_sign\_interval\_left(Min\_x, Min\_x, L\_start, A, B, C, D),

L\_end = L\_start + 0.01,

method\_loop(L\_start, L\_end, A, B, C, D).

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 1,

discr(DD, A, B, C),

DD >= 0,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

Min\_x = Max\_x,

f(Min\_x, F, A, B, C, D),

F > 0,

minus\_one(Min\_x, F\_left\_x),

plus\_one(Max\_x, F\_right\_x),

f(F\_left\_x, F\_left, A, B, C, D),

f(F\_right\_x, F\_right, A, B, C, D),

F\_left < F\_right,

check\_sign\_interval\_left(Min\_x, Min\_x, L\_start, A, B, C, D),

L\_end = L\_start + 0.01,

method\_loop(L\_start, L\_end, A, B, C, D).

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 1,

discr(DD, A, B, C),

DD >= 0,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

Min\_x = Max\_x,

f(Min\_x, F, A, B, C, D),

F > 0,

minus\_one(Min\_x, F\_left\_x),

plus\_one(Max\_x, F\_right\_x),

f(F\_left\_x, F\_left, A, B, C, D),

f(F\_right\_x, F\_right, A, B, C, D),

F\_left > F\_right,

check\_sign\_interval\_right(Max\_x, Max\_x, R\_end, A, B, C, D),

R\_start = R\_end - 0.01,

method\_loop(R\_start, R\_end, A, B, C, D).

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 1,

discr(DD, A, B, C),

DD >= 0,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

Min\_x = Max\_x,

f(Min\_x, F, A, B, C, D),

F < 0,

minus\_one(Min\_x, F\_left\_x),

plus\_one(Max\_x, F\_right\_x),

f(F\_left\_x, F\_left, A, B, C, D),

f(F\_right\_x, F\_right, A, B, C, D),

F\_left < F\_right,

check\_sign\_interval\_right(Max\_x, Max\_x, R\_end, A, B, C, D),

R\_start = R\_end - 0.01,

method\_loop(R\_start, R\_end, A, B, C, D).

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 1,

discr(DD, A, B, C),

DD >= 0,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

Min\_x = Max\_x,

f(Min\_x, F, A, B, C, D),

F < 0,

minus\_one(Min\_x, F\_left\_x),

plus\_one(Max\_x, F\_right\_x),

f(F\_left\_x, F\_left, A, B, C, D),

f(F\_right\_x, F\_right, A, B, C, D),

F\_left > F\_right,

check\_sign\_interval\_left(Min\_x, Min\_x, L\_start, A, B, C, D),

L\_end = L\_start + 0.01,

method\_loop(L\_start, L\_end, A, B, C, D).

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 2,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

Min\_x = Max\_x,

check\_sign\_interval\_left(Min\_x, Min\_x, L\_start, A, B, C, D),

L\_end = L\_start + 0.02,

method\_loop(L\_start, L\_end, A, B, C, D).

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 2,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

Min\_x <> Max\_x,

check\_sign\_interval\_left(Min\_x, Min\_x, L\_start, A, B, C, D),

L\_end = L\_start + 0.02,

method\_loop(L\_start, L\_end, A, B, C, D),

check\_sign\_interval\_right(Max\_x, Max\_x, R\_end, A, B, C, D),

R\_start = R\_end - 0.02,

method\_loop(R\_start, R\_end, A, B, C, D).

main(A, B, C, D) :-

A <> 0,

check\_cubic\_discr(A, B, C, D, Case),

Case = 3,

min\_max\_root\_df(Min\_x, Max\_x, A, B, C),

check\_sign\_interval\_left(Min\_x, Min\_x, L\_start, A, B, C, D),

L\_end = L\_start + 0.02,

method\_loop(L\_start, L\_end, A, B, C, D),

check\_sign\_interval\_right(Min\_x, Min\_x, C\_end, A, B, C, D),

C\_start = C\_end - 0.02,

method\_loop(C\_start, C\_end, A, B, C, D),

check\_sign\_interval\_right(Max\_x, Max\_x, R\_end, A, B, C, D),

R\_start = R\_end - 0.02,

method\_loop(R\_start, R\_end, A, B, C, D).

**Вывод**

Было необходимо разработать программу на языке PROLOG, которая решает кубическое уравнение вида *ax3+bx2+cx+d=0* комбинированным методом хорд и касательных. В ходе разработки были введены необходимые предикаты. Было проведено тестирование. Результаты тестирования приведены в отчёте. Поставленная задача выполнена.