**Министерство науки и высшего образования Российской Федерации**Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

Институт прикладной математики, физики и информатики.

Кафедра функционального анализа и его приложений.

**ЛАБОРАТОРНАЯ РАБОТА №1**

по дисциплине  
«Численные методы»  
на тему:  
«Алгоритм Евклида. Устранение кратности корней многочлена с помощью алгоритма Евклида»

Выполнил:

ст. гр. МКН-120

Минаев И.Ю.

Проверил:

Корнеева О.А.

Владимир, 2023 г.

**Постановка задачи**

1. Реализовать алгоритм Евклида, для нахождения НОД (наибольший общий делитель) двух многочленов.
2. Реализовать метод устранения кратности корней с помощью алгоритма Евклида.

**Теоретическая часть**

Поиск НОД.

Алгоритм Евклида позволяет найти НОД (наибольший общий делитель) двух многочленов.

Применение:

1. Поиск НОД.
2. Устранение кратности корней.
3. Для декодирования БЧХ-кодов.
4. Нахождение обратных элементов в конечных полях.

Теорема: для любых двух многочленов от 1-го переменного P(x) и Q(x) существуют такие многочлены c(x) и r(x) : P(x) = Q(x) \* c(x) + r(x).

Обозначим P(x) = rk-1(x), Q(x) = r0(x).

rm-2(x) = rm-1(x)\*cm(x)+rm(x).

…

rk-2(x) = rk-1(x)\*ck(x)+rk(x) (1)

rk-1(x) = rk(x)\*ck+1(x)+0 (2)

rk(x) = НОД(P(x), Q(x))

Доказательство: из (2) следует, что rk-1(x) делится без остатка на rk(x). Из (1) следует, что rk-2(x) делится без остатка на rk(x). Следовательно, P(x) и Q(x) делится без остатка на rk(x).

Устранение кратности корней.

Найдём многочлен Q(x), который имеет те же корни, что и P(x), но не кратные Q(x) = qmxm+qm-1xm-1+…+q0=0.

Перезапишем P(x): Pn(x) = pn(x-x1)k1(x-x2)k2…(x-xn)k­n ki – кратность

P’n(x) = A(x-x1)k1-1(x-x2)k2-1…(x-xm)k­m-1\*R(x)

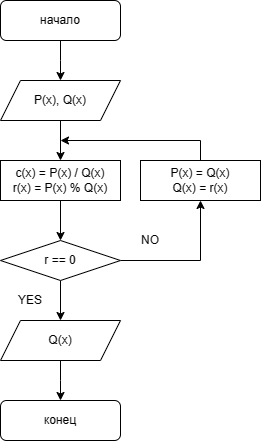
R(x) – многочлен среди которого нет корней исходного многочлена.

НОД(Pn(x), P’n(x)) = (x-x1)k1-1(x-x2)k2-1…(x-xm)k­m-1

**Практическая часть**

1. Реализовать алгоритм Евклида, для нахождения НОД (наибольший общий делитель) двух многочленов.

Блок-схема:



Код программы:

// Перегрузка оператора деления для многочлена / многочлен

Polynomial operator/(const Polynomial& other) const {

vector<double> dividend(coefficients);

vector<double> quotient((dividend.size() - 1) - (other.coefficients.size() - 1) + 1, 0);

while (dividend.size() >= other.coefficients.size()) {

double factor = dividend.back() / other.coefficients.back();

quotient[(dividend.size() - 1) - (other.coefficients.size() - 1)] = factor;

for (size\_t i = 0; i < other.coefficients.size(); ++i) {

dividend[dividend.size() - i - 1] -= factor \* other.coefficients[other.coefficients.size() - i - 1];

}

while (!dividend.empty() && dividend.back() == 0) {

dividend.pop\_back();

}

}

Polynomial res(quotient);

res.roundCoefficients();

return res;

};

// Перегрузка оператора остатка для многочлена / многочлен

Polynomial operator%(const Polynomial& other) const {

vector<double> dividend(coefficients);

while (dividend.size() >= other.coefficients.size()) {

double factor = dividend.back() / other.coefficients.back();

for (size\_t i = 0; i < other.coefficients.size(); ++i) {

dividend[dividend.size() - i - 1] -= factor \* other.coefficients[other.coefficients.size() - i - 1];

}

while (!dividend.empty() && dividend.back() == 0) {

dividend.pop\_back();

}

}

Polynomial res(dividend);

res.roundCoefficients();

res.removeLeadingZeros();

return res;

};

// Нахождение НОД двух многочленов

Polynomial gcd(const Polynomial& other) const {

Polynomial dividend = \*this;

Polynomial divisor = other;

Polynomial quotient = dividend / divisor;

Polynomial remainder = dividend % divisor;

while (!remainder.coefficients.empty()) {

dividend = divisor;

divisor = remainder;

quotient = dividend / divisor;

remainder = dividend % divisor;

}

return divisor;

};

int main() {

setlocale(LC\_ALL, "Russian");

Polynomial p({ -3, 1, 1, 0, 1 });

Polynomial q({ -3, 1, 1, 1 });

cout << "P: " << p << endl;

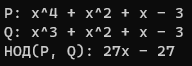
cout << "Q: " << q << endl;

cout << "НОД(P, Q): " << p.gcd(q) << endl;

return 0;

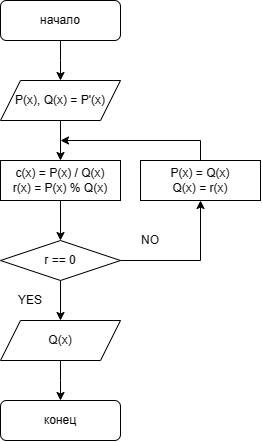
}

Вывод программы:



1. Реализовать метод устранения кратности корней с помощью алгоритма Евклида.

Блок-схема:



Код программы:

// Взятие производной от многочлена

Polynomial derivative() const {

vector<double> result(coefficients.size() - 1, 0);

for (size\_t i = 1; i < coefficients.size(); ++i) {

result[i - 1] = coefficients[i] \* i;

}

Polynomial res(result);

res.roundCoefficients();

res.removeLeadingZeros();

return res;

};

// Устранение кратности корней многочлена

Polynomial removeMultiplicity() const {

Polynomial dividend = \*this;

Polynomial divisor = this->derivative();

Polynomial quotient = dividend / divisor;

Polynomial remainder = dividend % divisor;

while (!remainder.coefficients.empty()) {

dividend = divisor;

divisor = remainder;

quotient = dividend / divisor;

remainder = dividend % divisor;

}

return divisor;

};

int main() {

setlocale(LC\_ALL, "Russian");

Polynomial p({ 0, 1, 0, -2, 0, 1 });

cout << "p: " << p << endl;

cout << "p’': " << p.derivative() << endl;

cout << "Многочлен без кратных корней: " << p.removeMultiplicity() << endl;

return 0;

}

Вывод программы:



**Вывод**

В ходе данной лабораторной работы были разобраны и реализованы на C++:

1. Алгоритм Евклида, для нахождения НОД двух многочленов.
2. Метод для получения производной многочлена.
3. Метод устранения кратности корней с помощью алгоритма Евклида.