МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

АЛГОРИТМЫ АЛГЕБРЫ И ТЕОРИИ ЧИСЕЛ

ЛАБОРАТОРНАЯ РАБОТА №15

студента 4 курса 431 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Никитина Арсения Владимировича

Проверил
доцент ______________ А. С. Гераськин

Саратов 2022

СОДЕРЖАНИЕ

1 Задание лабораторной работы

Вычисление значений и корней полиномов.

2 Теоретическая часть

В данной работе будет рассмотрен алгоритм нахождения корней и значений полиномов с помощью схемы Горнера.

Теорема Безу, несмотря на внешнюю простоту и очевидность, является одной из фундаментальных теорем теории многочленов. В этой теореме алгебраические свойства многочленов (которые позволяют работать с многочленами как с целыми числами) связываются с их функциональными свойствами (которые позволяют рассматривать многочлены как функции).

Теорема Безу

Остаток от деления многочлена $F(x)$ на линейный двучлен $x - a$ равен значению многочлена в точке a, т.е. числу $F(a)$.

Доказательство

Поделим с остатком многочлен $P(x)$ на двучлен $x - a$:

$P(x) = (x - a)Q(x) + R(x)$, где $R(x)$ — остаток. Так как $\deg R(x) < \deg(x - a) = 1$, то $R(x)$ — многочлен степени не выше 0, то есть константа, обозначим её за r. Подставляя $x = a$, поскольку $(a - a)Q(a) = 0$, имеем $P(a) = R(x) = r$.

Следствия

1. Число a является корнем многочлена $p(x)$ тогда и только тогда, когда $p(x)$ делится без остатка на двучлен $x - a$ (отсюда, в частности, следует, что множество корней многочлена $P(x)$ тождественно множеству корней соответствующего уравнения $P(x) = 0$).
2. Свободный член многочлена делится на любой целый корень многочлена с целыми коэффициентами (если старший коэффициент равен 1, то все рациональные корни являются и целыми).
3. Пусть a — целый корень приведённого многочлена $A(x)$ с целыми коэффициентами. Тогда для любого целого k число $A(k)$ кратно $a - k$.

Схема Горнера — алгоритм деления многочленов, записанный для частного случая, когда частное равно двучлену $x - a$.

Схема Горнера

Пусть $P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$ — делимое, $Q(x) = b_{n-1} x^{n-1} + b_{n-2} x^{n-2} + \ldots + b_0$ — частное (его степень, очевидно, будет на 1 меньше), r —

остаток (так как деление осуществляется на многочлен 1-ой степени, то степень остатка будет на 1 меньш, то есть нулевая, значит, остаток — константа).

По определению деления с остатком $P(x) = Q(x) \cdot (x - a) + r$.

После подстановки выражений многочленов получим:

$a_n x^n + a_{n-1} x^{n-1} + a_0 = \left(b_{n-1} x^{n-1} + b_{n-2} x^{n-2} + \ldots b_0\right) \cdot (x - a) + r.$

Раскроем скобки и приравняем коэффициенты при одинаковых степенях, после чего легко выразить коэффициенты частного через коэффициенты делимого и делителя.

Коэффициенты при одинаковых степенях	Выражение коэффициентов b_k через коэффициенты a_k
$a_n = b_{n-1}$	$b_{n-1} = a_n$
$a_{n-1} = b_{n-1} - ab_{n-1}$	$b_{n-2} = ab_{n-1} + a_{n-1}$
$\cdots$	$\cdots$
$a_k = b_{k-1} - ab_k$	$b_{k-1} = a_k + ab_k$
$\cdots$	$\cdots$
$a_0 = r - ab_0$	$r = ab_0 + a_0$

Затем вычисления сводятся в следующую таблицу:

a	a_n	$\cdots$	$\cdots$	a_k	$\cdots$	a_0
	$b_{n-1} = a_n$	$\cdots$	b_k	$b_{k-1} = a_k + ab_k$	$\cdots$	$r = ab_0 + a_0$

В таблице выделены те клетки, содержимое которых участвует в вычислениях на очередном шаге.

В алгоритме нахождения значений полинома в точке в с помощью результата работы алгоритма нахождения разложения полинома по теореме Горнера находится требуемое значение.

3 Практическая часть

3.1 Пример работы алгоритма

```
Вычислить значения и корни полинома - \enter
Выход из программы - 2
Введите значение:
Введите коэффициенты полинома, начиная с коэффициента при наибольшей степени:
1 2 -21 -20 71 114 45
Полином имеет вид:
x^6 + 2*x^5 + -21*x^4 + -20*x^3 + 71*x^2 + 114*x^1 + 45
Данный полином можно представить ввиде:
x^6 + 2*x^5 + -21*x^4 + -20*x^3 + 71*x^2 + 114*x^1 + 45 = (x + -1) ^ 3 * (x + 3) ^ 2 * (x + -5) ^ 1 * 1
Целочисленные значения корней: dict_keys([-1, 3, -5])
Вычислить значение полинома в точке - 1: 1
Введите точку, в которой требуется посчитать значение полинома: 12314321
Значение полинома в точке 12314321 равно 3487086744270854901685169787710009868288000

Вычислить значения и корни полинома - \enter
Выход из программы - 2
Введите значение: 2
Работа программы завершена
```

Рисунок 1

3.2 Код программы, реализующей рассмотренный алгоритм

```python
import sympy


def polinomial_view(coefs, flag=False):
    n = len(coefs) - 1
    a = ''
    mul = '*'
    if coefs:

        last = coefs[-1]
        coefs = coefs[:-1:]
        if coefs:
            for i, coef in enumerate(coefs):
                if coef:
                    print(f'{str(coef) + mul if coef != 1 else a}x^{n - i} +', end=' ')
        print(last, end='')
    else:
        print(coefs, end='')
    if not flag:
        print()
    return

```

```
def get_coefs():
    print('Введите коэффициенты полинома, начиная с коэффициента при' +
    ' наибольшей степени:')
    koef_integer = lambda x : int(x)
    coefs = map(koef_integer, input().split())
    return list(coefs)


def divide(number, coefs):
    new_coefs = [coefs[0]]
    cur = coefs[0]

    for i in coefs[1::]:
        cur = cur * number + i
        new_coefs.append(cur)
    if not new_coefs[-1]:
        return True, new_coefs[:-1:]
    else:
        return False, coefs


def horner_schema(p : list, dividers : dict):

    for i in sympy.divisors(p[-1]):
        result, new_coefs = divide(i, p)

        if result:
            p = new_coefs
            if i not in dividers.keys():
                dividers[i] = 1
            else:
                dividers[i] += 1

            return horner_schema(p, dividers)

        else:
            result1, new_coefs_other = divide(-i, p)
            if result1:
                p = new_coefs_other
                if -i not in dividers.keys():
```

```python
                    dividers[-i] = 1
                else:
                    dividers[-i] += 1

                return horner_schema(p, dividers)

    return p, dividers


def compute_value(poly : dict):

    x = int(input('Введите точку, в которой требуется посчитать значение полинома: '))
    result = 1

    for (key, value) in poly.items():
        result *= ((x + key) ** value)
    print(f'Значение полинома в точке {x} равно {result}')
    return


def left_bracket():
    print('(', end='')
    return


def main():

    while True:

        print('\nВычислить значения и корни полинома - \enter')

        print('Выход из программы - 2')

        try:
            value = int(input('Введите значение: '))

        except ValueError:
            value = 1

        if value == 1:
```

```
            p = get_coefs()

            poly, result = horner_schema(p.copy(), dict())
            print('Полином имеет вид: ')
            polinomial_view(p)

            if poly != [1]:
                print('Полином не является приводимым')
            else:

                print('Данный полином можно представить ввиде:')
                polinomial_view(p, True)
                print(' = ', end='')

                for (key, value) in result.items():
                    print(f'(x + {key}) ^ {value} * ', end='')
                print(1)

                print(f'Целочисленные значения корней: {result.keys()}')

                chosen_option = int(input('Вычислить значение полинома в точке - 1: '))
                if chosen_option == 1:
                    compute_value(result)

        elif value == 2:
            print('Работа программы завершена')
            return


if __name__ == "__main__":
    main()

# Пример:
# horner_schema([1, 2, -21, -20, 71, 114, 45], dict())
```