Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Основы программирования»

Отчет по лабораторной работе №4
«Численные методы решения нелинейных уравнений»

Выполнил:
студент группы ИУ5-15Б
Трифонов Дмитрий
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Папшев И.С.
Подпись и дата:

Москва, 2021 г.

Постановка задачи

1. Разработайте программу для вычисления корней уравнения

$$x - k*\cos(x) = 0$$

при k=1 простой итерацией, половинным делением и методом Ньютона с погрешностью eps < 0.000001 и eps < 0.00000001. Для каждого из трех методов определить (и вывести на экран) количество шагов алгоритма, использованных для получения результата.

2. Выполнить п.1 для k=5 и объяснить результаты.

Программа должна обеспечивать возможность многократного выполнения пунктов 1 и 2 для различных значений параметров:

- начального приближения *x0*,
- коэффициента *k,*
- требуемой точности вычислений *eps*.

Для реализации этого требования используйте функцию «Меню» (Пример использования меню в программе приведён в п 7.2.1 Методических указаний).

3. Разработайте «универсальную» функцию для вычисления корней уравнений вида *y(x)=0* методом половинного деления, используя в качестве ***параметра указатель на функцию***.

Примените эту функцию для нахождения всех корней уравнения *y=x-5*cos(x)* и *y=x-10*cos(x),* добавив следующие параметры:

eps - требуемая точность вычислений

xl и xr - левая и правая границы интервала, содержащего ровно один корень.

Разработка алгоритма

Краткое словесное описание алгоритма

В данной лабораторной работе для поиска корней используется метод половинного деления.

Этот метод является наиболее подходящим для вычисления корней уравнений вида *f(x)=0* с помощью компьютера.

Для использования этого метода нужно задать границы интервала [xl, xr] на оси абсцисс, содержащего *ровно один корень*, и требуемую точность вычислений.

Суть метода заключается в следующем. Выбирают X на середине интервала [xl, xr] и определяют f(X). Если |f(X)| < eps, то середина интервала считается корнем уравнения, иначе корень ищется на том интервале из двух полученных, для которого значения функции на концах имеют разные знаки. Действия повторяются до тех пор, пока интервал >eps. Перед входом в цикл необходимо проверить, не являются ли границы интервала корнями уравнения f(X)=0.

Поиск же интервалов, на которых находится ровно один корень, производится интервальным методом. (с определённым шагом расширяем границы текущего интервала, пока f(начала) имеет такой же знак, что и f(конца)).

Входные данные

- double k (коэффициент из условия)
- double eps (точность вычислений)
- double start, finish (границы интервала, на котором будет производится поиск корней)
- double step (шаг, с которым будут отбираться интервалы для поиска корней методом половинного деления)

Выходные данные

После выполнения алгоритма на экран выводятся:

- Начальные данные для поиска корней
- Найденные корни и шаг, на котором их получили
  Корни и шаг, на котором их получили, записываются в std::pair <int, double> (далее root). Получившиеся пары записываются в std::vector <root> и через него выводятся на экран

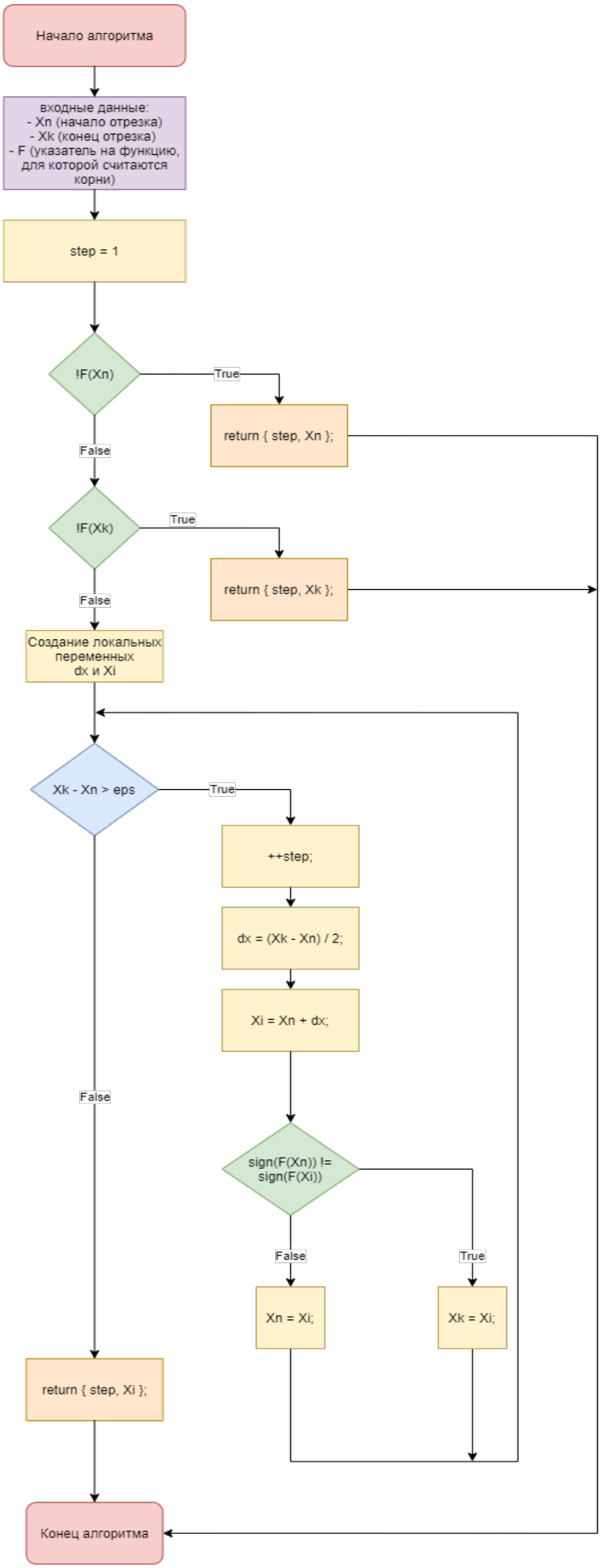
Промежуточные данные

Вектор корней уравнения получается из функции vector <root> solveFunc(double (*F) (double)), которая как параметр принимает функцию y(x), для которой и считаются корни y(x) = 0. Для поиска интервалов с одним корнем используется функция bool getNextInterval(double& Xn, double& Xk, double (*F) (double)) (в случае нахождения интервала с одним корнем возвращает 1, иначе 0).
Сами же корни методом половинного деления находит функция root calculateRoot(double Xn, double Xk, double (*F) (double)). Она использует переменные:

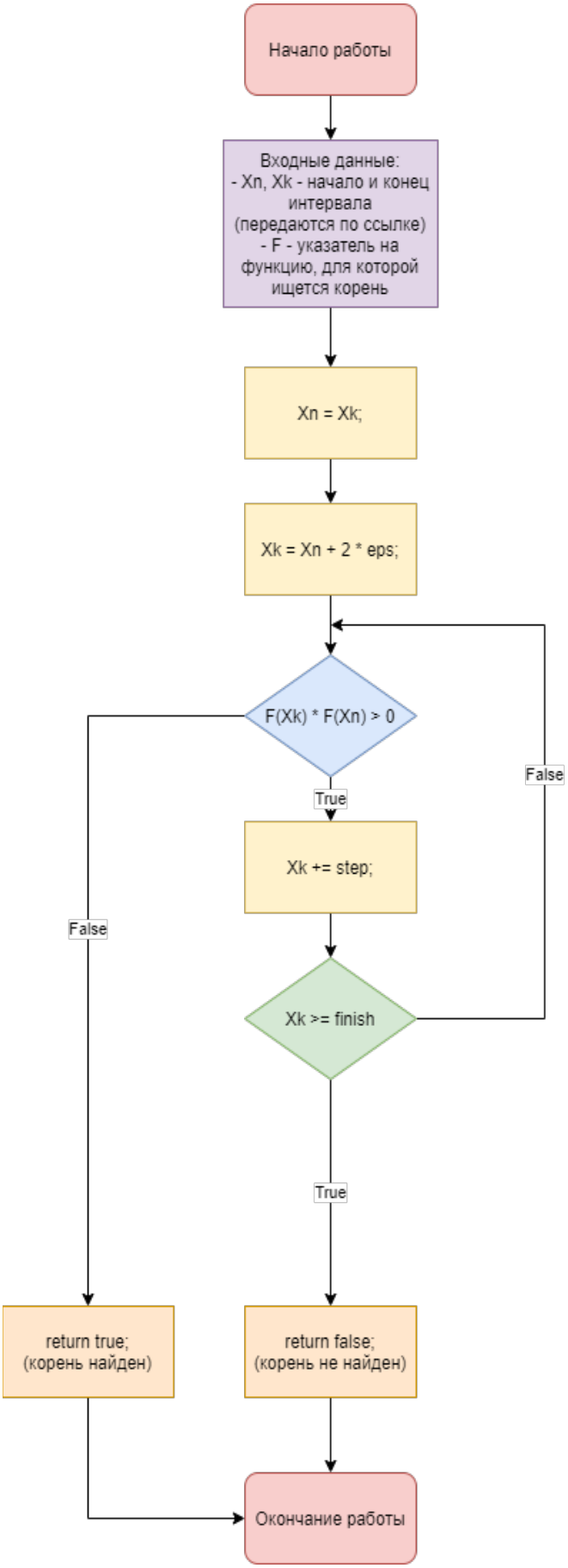
- int step (шаг, на котором было выполнено последнее вычисление)
- double dx, Xi (переменные для самого алгоритма половинного деления).

Блок -схемы

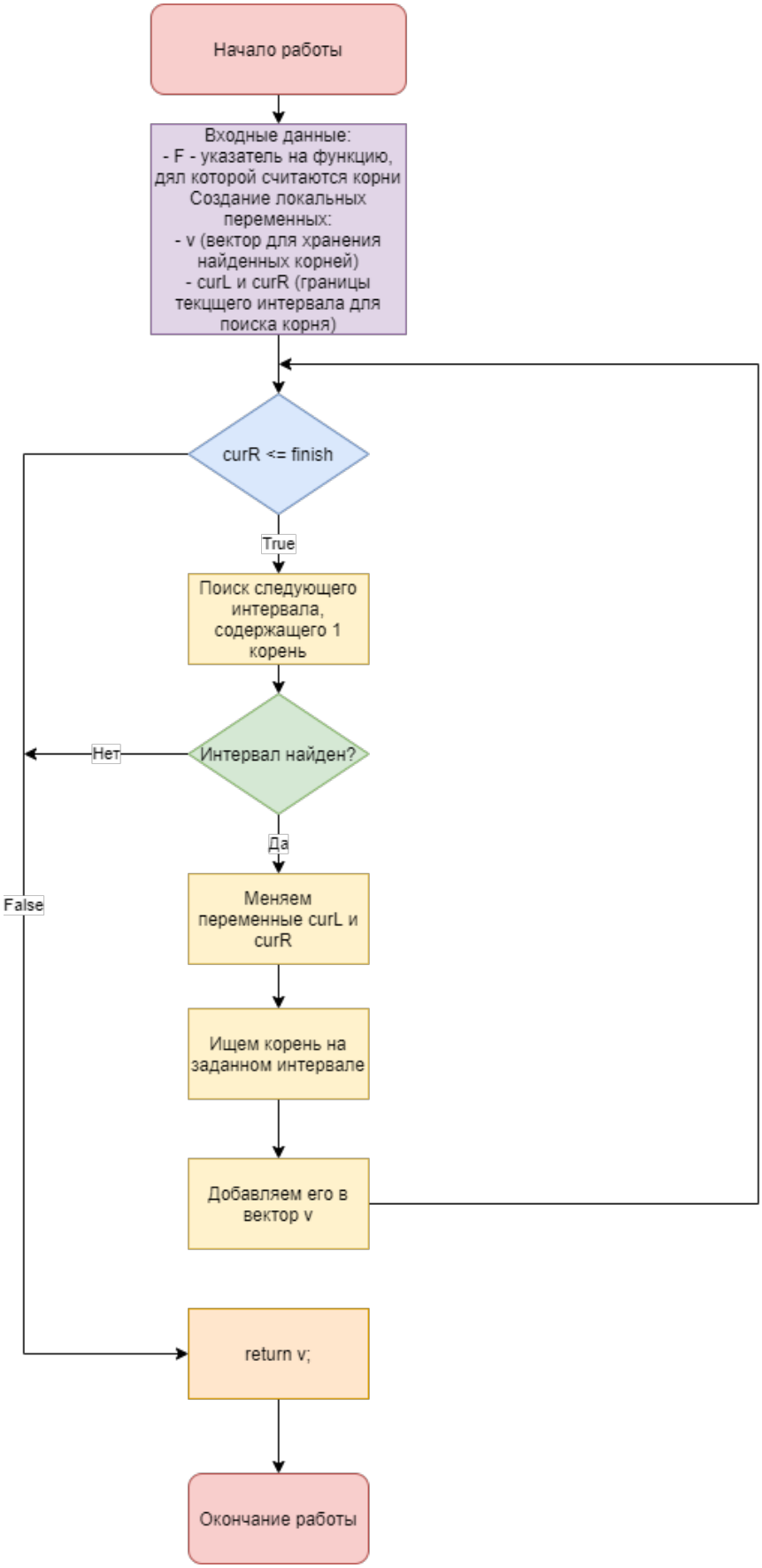
1. Нахождение корня на заданном интервале методом половинного деления



2. Нахождение следующего интервала, содержащего один корень



3. Функция нахождения всех корней



Текст программы


```cpp
#include <iostream>
#include <iomanip>
#include <conio.h>
#include <cmath>
#include <vector>

using namespace std;

double k = 5, step = 0.1, eps = 0.000001;
double start = -10, finish = 10;

typedef pair <int, double> root;

template <typename T> int sign(T val) {
    return (T(0) < val) - (val < T(0));
}
int getDoubleLength(double x) {
    return -log10(x);
}
double Function(double x) {
    return x - k * cos(x);
}

void getK() {
    cout << "Введите значение для коэффициента k" << endl;
    cout << "Текущее значение: " << k << endl;
    cin >> k;
}
void getEps() {
    cout << "Выберите точность вычислений(eps)" << endl;
    cout << "Текущее значение: " << eps << endl;
    cin >> eps;
    if (eps <= 0)
        eps = 0.1;
}
void getStep() {
    cout << "Введите шаг для поиска интервалов с одним корнем" << endl;
    cout << "Текущее значение: " << step << endl;
    cin >> step;
    if (step <= 0)
        step = 0.1;
}
void getBoarders() {
    cout << "Введите границы для поиска корней через пробел" << endl;
    cout << "Текущее границы: " << start << " - " << finish << endl;
    cin >> start >> finish;
    if (finish < start) {
        swap(start, finish);
    }
```

```
50  }
51
52  void printWrongCommand() {
53      cout << "Введена некорректная команда." << endl;
54      cout << "Для выхода нажмите 6." << endl;
55      cout << "(Для продолжения нужно было нажать любую клавишу)" << endl;
56      while (!_kbhit());
57  }
58
59  void setupOutput() {
60      system("chcp 1251 > nul");
61      cout << fixed << showpoint;
62  }
63
64  void printProgressBar(double start, double end) {
65      cout << '\r';
66      cout << "\x1b[0K";
67      cout << "Loading: |";
68      int pointsAmount = ((float)(100.0 * (start / end)) / 5.0);
69      for (int i = 0; i < pointsAmount; ++i) {
70          cout << "#";
71      }
72      for (int i = pointsAmount; i < 20; ++i)
73          cout << " ";
74      cout << '|';
75  }
76
77  root calculateRoot(double Xn, double Xk, double (*F) (double)) {
78      int step = 1;
79      if (!F(Xn))
80          return { step, Xn };
81      if (!F(Xk))
82          return { step, Xk };
83
84      double dx, Xi;
85      while (Xk - Xn > eps) {
86          ++step;
87          dx = (Xk - Xn) / 2;
88          Xi = Xn + dx;
89          if (sign(F(Xn)) != sign(F(Xi)))
90              Xk = Xi;
91          else
92              Xn = Xi;
93      }
94      return { step, Xi };
95  }
96  bool getNextInterval(double& Xn, double& Xk, double (*F) (double)) {
97      Xn = Xk;
98      Xk = Xn + 2 * eps;
```

```
 99      while (F(Xk) * F(Xn) > 0) {
100          Xk += step;
101          if (Xk >= finish)
102              return false;
103      }
104      return true;
105  }
106  vector <root> solveFunc(double (*F) (double)) {
107      vector <root> v;
108      double curL = 0, curR = start;
109      while (curR <= finish) {
110          if (!getNextInterval(curL, curR, F)) {
111              printProgressBar(finish, finish);
112              return v;
113          }
114          printProgressBar(curR, finish);
115          v.push_back(calculateRoot(curL, curR, F));
116      }
117      return v;
118  }
119  void printFunc(double (*F) (double)) {
120      cout << "Расчёт корней функции со следующими параметрами:" << endl;
121      cout << "k = " << k << endl;
122      cout << setprecision(getDoubleLength(eps)) << "eps = " << eps << endl;
123      cout << "Границы: " << start << " - " << finish << endl;
124      cout << "Шаг: " << setprecision(getDoubleLength(step)) << step << endl;
125
126      vector <root> funcRoots = solveFunc(F);
127      cout << endl;
128      if (!funcRoots.size()) {
129          cout << "Корни не найдены." << endl;
130          return;
131      }
132      for (auto& i : funcRoots) {
133          cout << setprecision(getDoubleLength(eps)) << i.second;
134          cout << "(Шаг: " << i.first << " )" << endl;
135      }
136      cout << "Итого корней: " << funcRoots.size() << endl;
137      cout << "Для продолжения нажмите любую клавишу" << endl;
138      while (!_kbhit());
139  }
140
141  short menu() {
142      cout << "============Menu============" << endl;
143      cout << "| 1 - Ввести значение k    |" << endl;
144      cout << "| 2 - Ввести значение eps  |" << endl;
145      cout << "| 3 - Ввести границы       |" << endl;
146      cout << "| 4 - Вычислить корни      |" << endl;
147      cout << "| 5 - Выбрать шаг          |" << endl;
```

```
148     cout << "| 6 - Выход                |" << endl;
149     cout << "===========================" << endl;
150     cout << "Выберите номер:" << endl;
151     int choice;
152     cin >> choice;
153     while (cin.fail()) {
154         cout << "Ошибка ввода. Повторите ввод\n";
155         cin.clear();
156         cin.ignore(10, '\n');
157         cin >> choice;
158     }
159     return choice;
160 }
161
162 int main() {
163     setupOutput();
164
165     while (true) {
166         short choice = menu();
167         system("cls");
168
169         switch (choice) {
170         case 1:
171             getK();
172             break;
173         case 2:
174             getEps();
175             break;
176         case 3:
177             getBoarders();
178             break;
179         case 4:
180             printFunc(Function);
181             break;
182         case 5:
183             getStep();
184             break;
185         case 6:
186             return 0;
187         default:
188             printWrongCommand();
189         }
190         system("cls");
191     }
192 }
```

Анализ результатов

```
Расчёт корней функции со следующими параметрами:
k = 5.000000
eps = 0.000001
Границы: -20.000000 - 20.000000
Шаг: 0.1
Loading: |####################|
-3.837468(Шаг: 25 )
-1.977383(Шаг: 22 )
1.306440(Шаг: 23 )
Итого корней: 3
Для продолжения нажмите любую клавишу
```

$y = x - 5 * \cos(x)$

```
Расчёт корней функции со следующими параметрами:
k = 10.000000
eps = 0.000001
Границы: -20.000000 - 20.000000
Шаг: 0.1
Loading: |####################|
-10.099020(Шаг: 25 )
0.099020(Шаг: 25 )
Итого корней: 2
Для продолжения нажмите любую клавишу
```

$y = x^2 + 10 * x - 1$

```
Расчёт корней функции со следующими параметрами:
k = 10.000000
eps = 0.000001
Границы: -20.000000 - 20.000000
Шаг: 0.1
Loading: |####################|
-18.849556(Шаг: 22 )
-15.707963(Шаг: 23 )
-12.566371(Шаг: 23 )
-9.424778(Шаг: 23 )
-6.283185(Шаг: 23 )
-3.141593(Шаг: 23 )
-0.000000(Шаг: 23 )
3.141592(Шаг: 23 )
6.283185(Шаг: 23 )
9.424778(Шаг: 23 )
12.566370(Шаг: 23 )
15.707964(Шаг: 23 )
18.849555(Шаг: 23 )
Итого корней: 13
Для продолжения нажмите любую клавишу
```

$y = \sin(x)$

```
Расчёт корней функции со следующими параметрами:
k = 10.000000
eps = 0.000001
Границы: -20.000000 - 20.000000
Шаг: 0.1
Loading: |####################|
-9.678884(Шаг: 25 )
-8.966017(Шаг: 21 )
-4.271096(Шаг: 24 )
-1.746329(Шаг: 23 )
1.427551(Шаг: 23 )
5.267116(Шаг: 23 )
7.068891(Шаг: 22 )
Итого корней: 7
Для продолжения нажмите любую клавишу
```

$y = x - 10 * \cos(x)$