Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по рубежному контролю №1

по курсу «Анализ алгоритмов»

Студент ИУ7-51Б (Группа) _____ (Подпись, дата) Д. В. Шубенина (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Л. Л. Волкова (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Кластеризация данных является важным инструментом в области машинного обучения. Она позволяет группировать данные на основе их сходства и отделить их от остальных [1].

Целью данного рубежного контроля является параллелизация нечеткого алгоритма c-средних.

Для достижения поставленной цели необходимо решить следующие задачи:

1) описать понятие кластеризации;

2) описать нечеткий алгоритм кластеризации c-средних;

3) реализовать программу, выполняющую параллельную работу алгоритма с выводом информации.

1 Аналитическая часть

Кластерный анализ — это ряд математических методов интеллектуального анализа данных, предназначенных для разбиения множества исследуемых объектов на компактные группы, называемые кластерами. Под объектами кластерного анализа подразумеваются предметы исследования, нуждающиеся в кластеризации по некоторым признакам. Признаки объектов могут иметь как непрерывные, так и дискретные значения [2].

Метод с-средних — итеративный нечеткий алгоритм кластеризации. В данном методе кластеры являются нечеткими множествами, и каждый объект из выборки исходных данных относится одновременно ко всем кластерам с различной степенью принадлежности. Таким образом, матрица принадлежности объектов к кластерам содержит не бинарные, а вещественные значения, принадлежащие отрезку $[0; 1]$ [2].

Метод с-средних предполагает заполнение матрицы разбиения (принадлежности) $U = \{u_{ij}\}$ следующим образом [2]:

$$u_{ij} = \frac{1}{\sum_{k=1}^{p} \left(\frac{d^2(x_j, c_i)}{d^2(x_j, c_k)} \right)^{\frac{1}{w-1}}}, \tag{1.1}$$

где d — расстояние между объектом и центром кластера,
$C = \{c_i\}_{i=1}^{p}$ — множество центров кластеров,
$X = \{x_j\}_{j=1}^{n}$ — множество объектов,
w — показатель нечеткости, регулирующий точность разбиения.

Вывод

В данном разделе было рассмотрено понятие кластеризации. Также был описан нечеткий алгоритм с-средних.

2 Конструкторская часть

2.1 Требования к программному обеспечению

К программе предъявлен ряд требований:

1) наличие интерфейса для вывода результатов работы кластеризации;
2) наличие возможности ввода массива входных точек (x, y) из текстового файла;

2.2 Разработка алгоритмов

На рисунке 2.1 показана схема алгоритма главного потока нечеткого алгоритма c-средних.

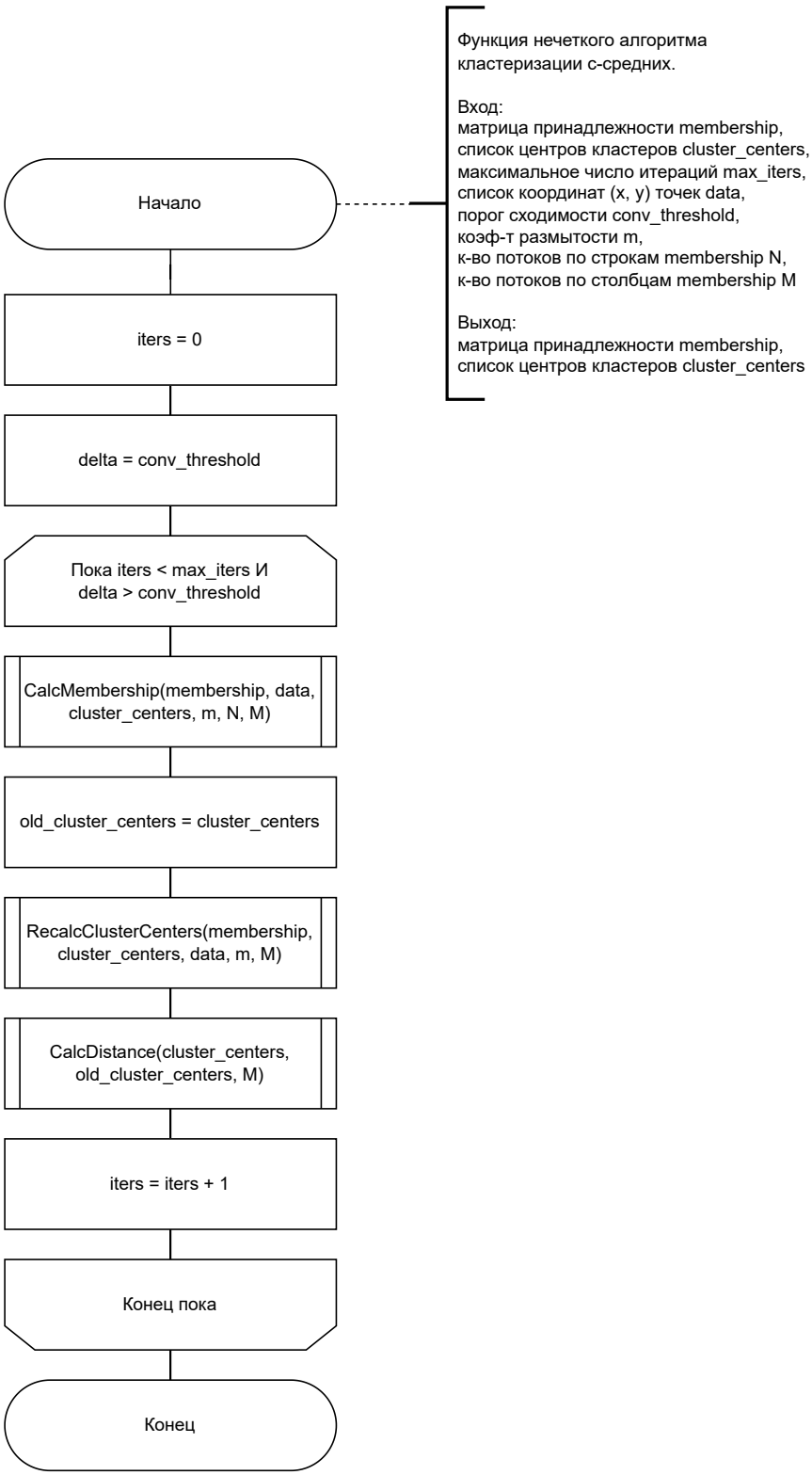

Рисунок 2.1 – Алгоритм главного потока нечеткого алгоритма с-средних

На рисунке 2.2 представлена схема алгоритма заполнения матрицы принадлежности.

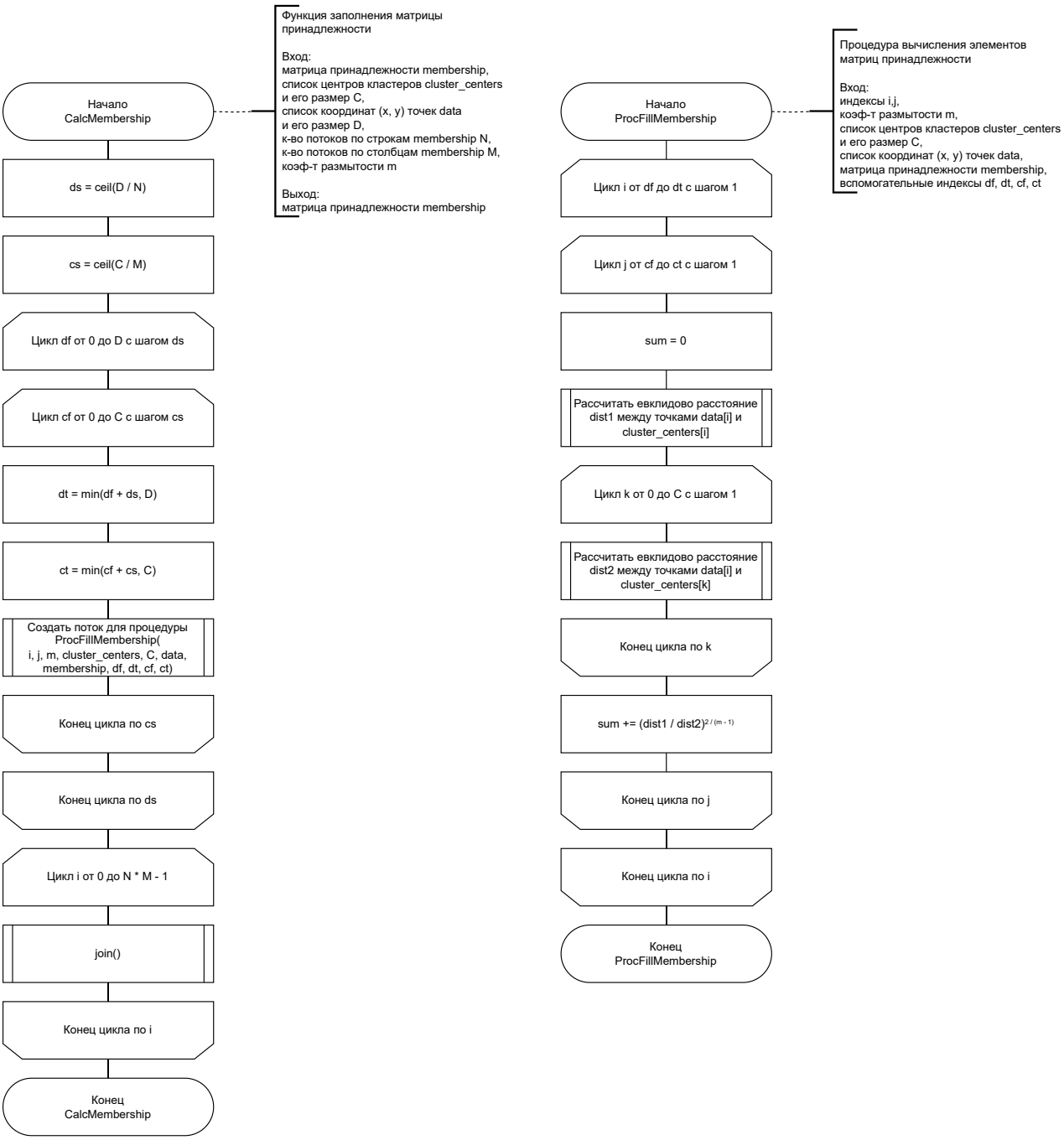

Рисунок 2.2 – Алгоритм заполнения матрицы принадлежности

На рисунке 2.3 показана схема алгоритма сдвига центров кластеров.

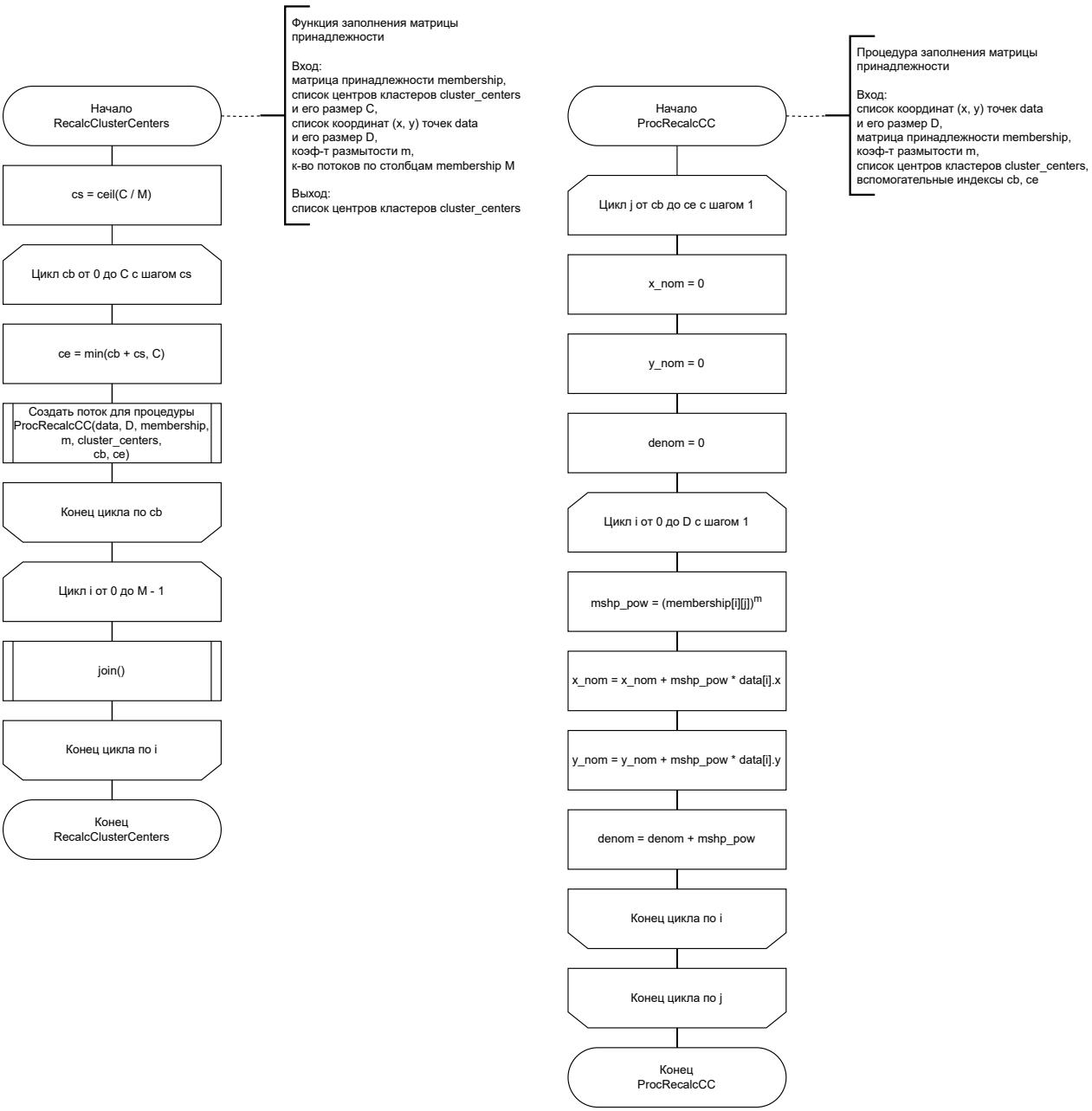

Рисунок 2.3 – Алгоритм сдвига центров кластеров

На рисунке 2.4 показана схема алгоритма расчета изменения позиции центров кластеров.

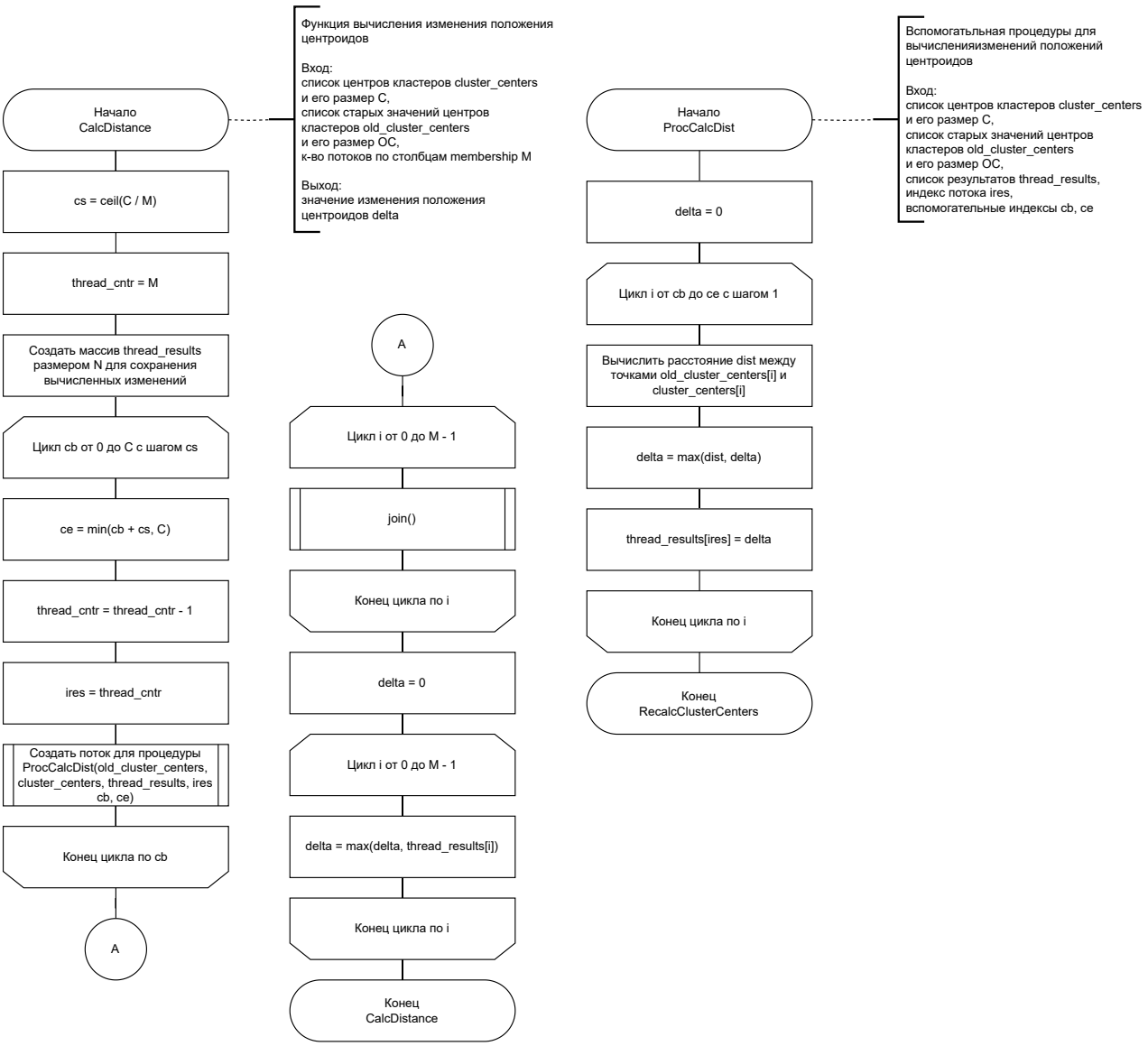

Рисунок 2.4 – Алгоритм расчета изменения позиции центров кластеров

Вывод

В данном разделе выдвинуты требования к программному обеспечению и представлена схема нечеткого алгоритма c-средних.

3 Технологическая часть

3.1 Средства реализации

Для реализация нечеткого алгоритма с-средних был выбран язык *C++* [3], так как данный язык был использован в лабораторной работе №4.

3.2 Реализация алгоритмов

На листинге 3.1 показана реализация алгоритма главного потока нечеткого алгоритма с-средних.

Листинг 3.1 – Реализация алгоритма главного потока нечеткого алгоритма с-средних

```
void c_means_parallel(
    membership_t &membership, point_vec_t &cluster_centers,
    const point_vec_t &data,
    double m, double conv_threshold, int max_iters)
{
    int iters = 0;
    double delta = conv_threshold + 1.0;
    while (iters < max_iters && delta > conv_threshold)
    {
        std::vector<std::thread> threads;
        mt_calc_membership(membership, data, cluster_centers, m,
           2, 2);

        auto old_cluster_centers = cluster_centers;
        mt_recalc_clustercenters(membership, cluster_centers,
           data, m, 2);
        delta = mt_calc_distance(cluster_centers,
           old_cluster_centers, 2);
        ++iters;
    }
}
```

На листинге 3.2 представлена реализация алгоритма расчета элементов матрицы принадлежности.

Листинг 3.2 – Реализация алгоритма расчета элементов матрицы принадлежности

```
static void mt_calc_membership(
    membership_t &membership,
```

```
3       const point_vec_t &data, point_vec_t &cluster_centers,
            double m,
4       int rows_n_thread, int cols_n_thread)
5   {
6       std::vector<std::thread> threads;
7
8       int data_step = ceil(float(data.size()) /
            float(rows_n_thread));
9       int c_step = ceil(float(cluster_centers.size()) /
            float(cols_n_thread));
10
11      for (size_t data_from = 0; data_from < data.size();
            data_from += data_step)
12      {
13          for (size_t c_from = 0; c_from < cluster_centers.size();
                c_from += c_step)
14          {
15              int data_to = std::min<int>(data_from + data_step,
                    data.size());
16              int center_to = std::min<int>(c_from + c_step,
                    cluster_centers.size());
17              threads.emplace_back(
18                  [ &, data_from, data_to, c_from, center_to ]
19                  {
20                      for (int i = data_from; i < data_to; ++i)
21                      {
22                          for (int j = c_from; j < center_to; ++j)
23                          {
24                              double sum = 0.0;
25                              double dist1 = sqrt(
26                                  pow(data[i][0] -
                                        cluster_centers[j][0], 2) +
27                                  pow(data[i][1] -
                                        cluster_centers[j][1], 2)
28                              );
29                              for (size_t k = 0; k <
                                    cluster_centers.size(); ++k)
30                              {
31                                  double dist2 = sqrt(
32                                      pow(data[i][0] -
                                            cluster_centers[k][0], 2)
```

```
                                        +
                                    pow(data[i][1] -
                                        cluster_centers[k][1], 2)
                                );
                                sum += pow(dist1 / dist2, 2.0 /
                                    (m - 1.0));
                            }
                            membership[i][j] = 1.0 / sum;
                        }
                    }
                });
        }
    }
    for (auto &thr : threads)
        thr.join();
}
```

На листинге 3.3 показана реализация алгоритма перерасчета положений центров кластеров.

Листинг 3.3 – Реализация алгоритма перерасчета положений центров кластеров

```
static void mt_recalc_clustercenters(
    membership_t &membership, point_vec_t &cluster_centers,
    const point_vec_t &data,
    double m, int n_threads)
{
    std::vector<std::thread> threads;
    int cltr_step = ceil(float(cluster_centers.size()) /
        float(n_threads));
    for (size_t cltr_beg = 0; cltr_beg < cluster_centers.size();
        cltr_beg += cltr_step)
    {
        int cltr_end = std::min<int>(cltr_beg + cltr_step,
            cluster_centers.size());
        threads.emplace_back(
            [ &, cltr_beg, cltr_end ]
            {
                for (int j = cltr_beg; j < cltr_end; ++j)
                {
                    double x_nom = 0.0, y_nom = 0.0, denom = 0.0;
                    for (size_t i = 0; i < data.size(); ++i)
```

```
18                      {
19                          double membership_pow_m =
                               pow(membership[i][j], m);
20                          x_nom += membership_pow_m * data[i][0];
21                          y_nom += membership_pow_m * data[i][1];
22                          denom += membership_pow_m;
23                      }
24                      cluster_centers[j] = { x_nom / denom, y_nom
                           / denom };
25                  }
26              }
27          );
28      }
29      for (auto &thr : threads)
30          thr.join();
31  }
```

На листинге 3.4 представлена реализация алгоритма расчета смещений положений кластеров относительно их старой позиции.

Листинг 3.4 – Реализация алгоритма расчета смещений положений кластеров относительно их старой позиции

```
1  static double mt_calc_distance(
2      point_vec_t &cluster_centers,
3      point_vec_t &old_cluster_centers,
4      int n_threads)
5  {
6      std::vector<std::thread> threads;
7      std::vector<double> thr_results(n_threads);
8      int cltr_step = ceil(float(cluster_centers.size()) /
          float(n_threads));
9      for (size_t cltr_beg = 0; cltr_beg < cluster_centers.size();
          cltr_beg += cltr_step)
10     {
11         int cltr_end = std::min<int>(cltr_beg + cltr_step,
              cluster_centers.size());
12         int ires = --n_threads;
13         threads.emplace_back(
14             [ &, cltr_beg, cltr_end, ires ]
15             {
16                 double delta = 0;
17                 for (int i = cltr_beg; i < cltr_end; ++i)
```

```
18                  {
19                      double distance = sqrt(
20                          pow(old_cluster_centers[i][0] -
                               cluster_centers[i][0], 2) +
21                          pow(old_cluster_centers[i][1] -
                               cluster_centers[i][1], 2)
22                      );
23                      if (distance > delta)
24                          delta = distance;
25                  }
26                  thr_results[ires] = delta;
27              }
28          );
29      }
30      for (auto &thr : threads)
31      {
32          thr.join();
33      }
34      double delta = 0;
35      for (auto &distance : thr_results)
36      {
37          if (distance > delta)
38              delta = distance;
39      }
40      return delta;
41  }
```

Вывод

В данном разделе были приведены сведения о средствах реализации нечеткого алгоритма c-средних. Также была предоставлена реализация разработанных алгоритмов.

4 Исследовательская часть

4.1 Демонстрация работы программы

На рисунке 4.1 приведен пример работы программы для случая, когда пользователь выбирает пункт 1 «Кластеризация методом с-средних» меню и указывает число кластеров 3.

```
                Меню
1. Кластеризация методом с-средних
2. Редактировать файл, содержащий множество точек.
0. Выход.

Выберите опцию (0-2): 1

Введите число кластеров (1-6): 3
Матрица принадлежности
 0.001  0.998  0.001
 0.000  1.000  0.000
 0.001  0.998  0.001
 0.017  0.000  0.983
 0.015  0.000  0.985
 0.998  0.000  0.002
Центроиды кластеров: [ [ 70.999 18.997 ] [ 4.000 2.334 ] [ 69.499 13.501 ] ]

                Меню
1. Кластеризация методом с-средних
2. Редактировать файл, содержащий множество точек.
0. Выход.

Выберите опцию (0-2): 0
```

Рисунок 4.1 – Демонстрация работы программы

4.2 Вывод

В данном разделе была приведена демонстрация работы программы.

ЗАКЛЮЧЕНИЕ

Цель, поставленная в начале работы, была достигнута. Были решены следующие задачи:

1) дано описание понятия кластеризации;

2) дано описание нечеткого алгоритма кластеризации c-средних;

3) реализована программа, выполняющая параллельную работу алгоритма с выводом информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Каримов К. Х.*, *Василий Е. А.* Теоретические основы кластеризации данных // Актуальные вопросы фундаментальных и прикладных научных исследований : Сборник научных статей по материалам II Международной научно-практической конференции, Уфа, 19 мая 2023 года. Том Часть 2. — Уфа : Общество с ограниченной ответственностью "Научно-издательский центр «Вестник науки»", 2023. — С. 242—247. — EDN XNJGQS.

2. *Лосев Д. Г.* Разработка и сравнение параллельных реализаций итеративных алгоритмов кластеризации // Наука молодых - будущее России : сборник научных статей 6-й Международной научной конференции перспективных разработок молодых ученых (9-10 декабря 2021 года), в 5 томах. Т. 4. — Курск : Юго-Зап. гос. ун-т, 2021. — С. 71—74.

3. C++ language. — [Электронный ресурс]. — Режим доступа: `https://en.cppreference.com/w/cpp/language` (дата обращения: 12.12.2023).