



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»
на тему: «Кластеризация нечетким алгоритмом с-средних»

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	3
1 Аналитическая часть	4
2 Конструкторская часть	5
2.1 Требования к программному обеспечению	5
2.2 Разработка алгоритмов	5
Вывод	9
3 Технологическая часть	10
3.1 Средства реализации	10
3.2 Реализация алгоритмов	10
Вывод	14
4 Исследовательская часть	15
4.1 Демонстрация работы программы	15
Вывод	15
ЗАКЛЮЧЕНИЕ	16
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	17

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 с-средних.

Для достижения поставленной цели необходимо решить следующие задачи:

1. описать понятие кластеризации;
2. описать нечеткий алгоритм кластеризации с-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]:

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

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 функции главного потока нечеткого алгоритма с-средних.

На рисунке 2.2 представлена схема функции заполнения матрицы принадлежности.

На рисунке 2.3 показана схема функции сдвига центров кластеров.

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

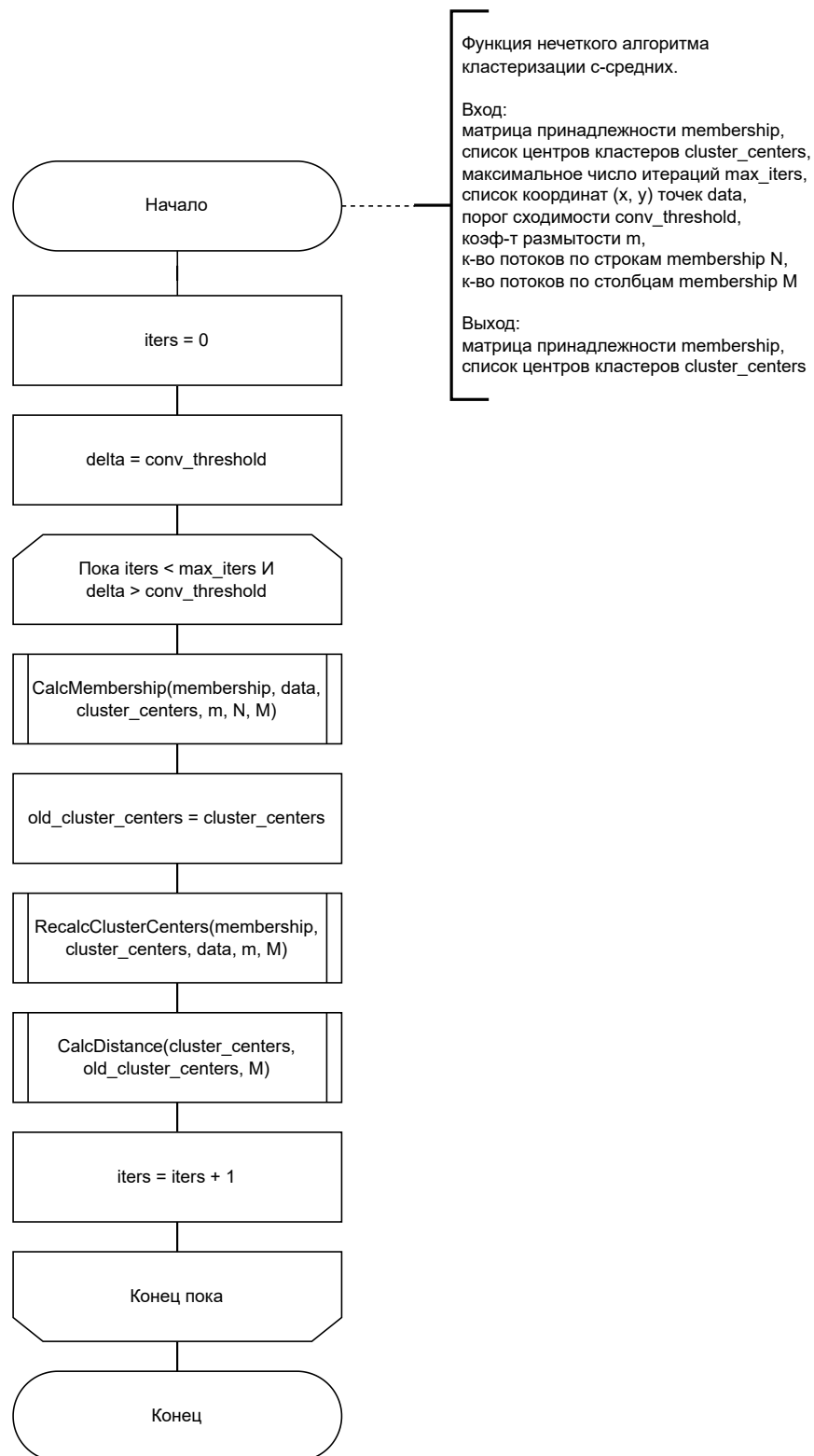


Рисунок 2.1 – Функция главного потока нечеткого алгоритма с-средних

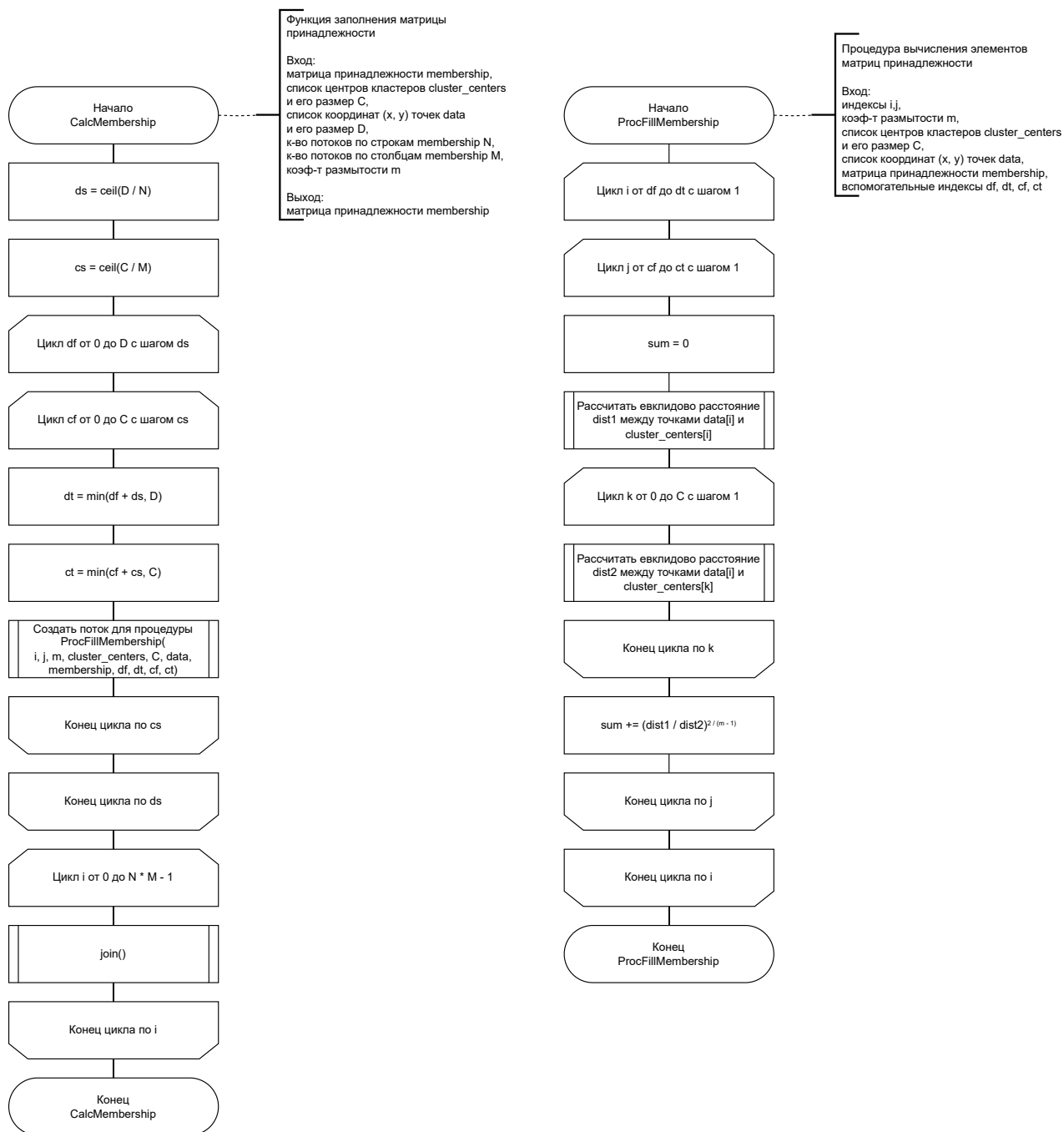


Рисунок 2.2 – Функции заполнения матрицы принадлежности

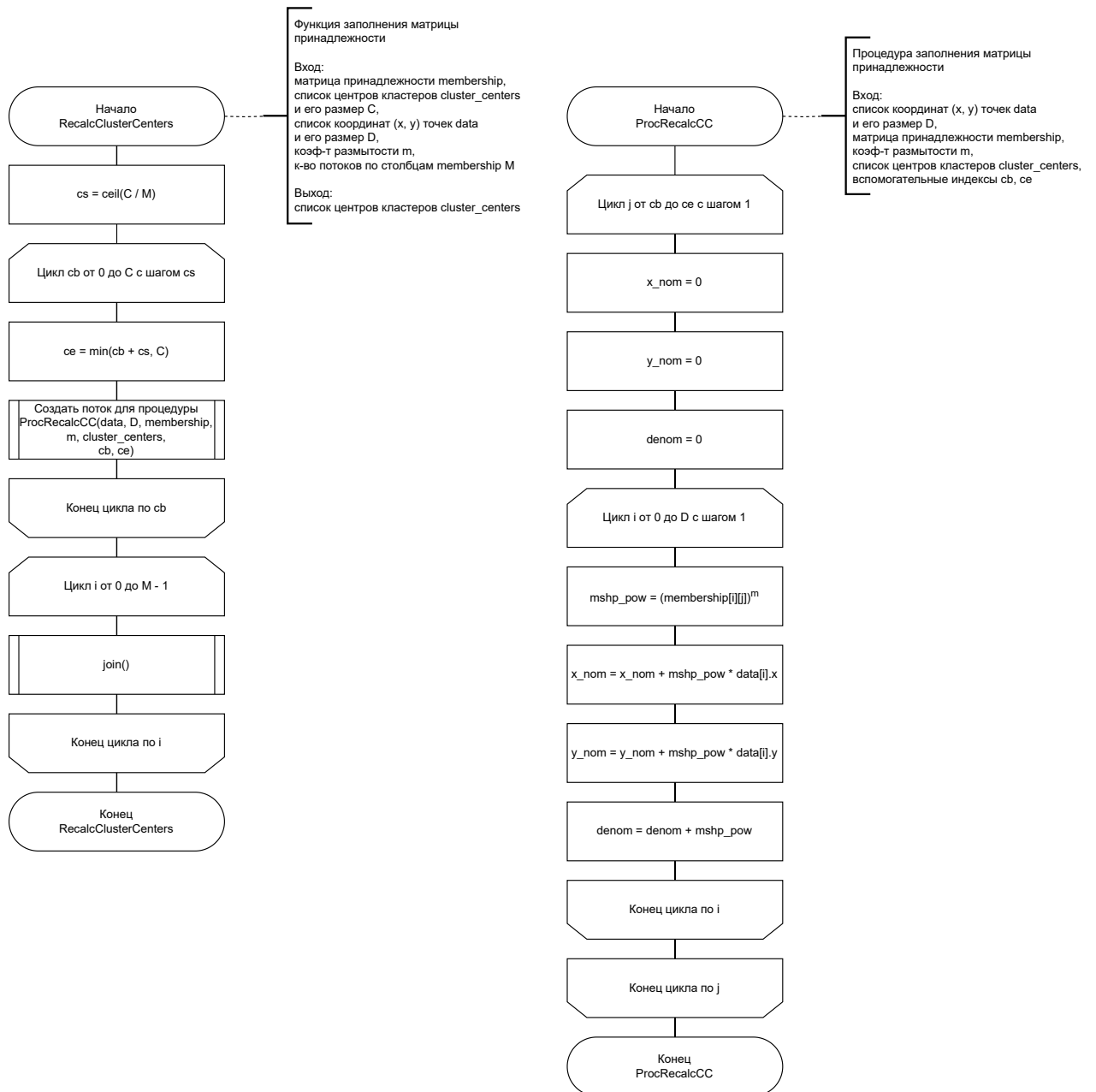


Рисунок 2.3 – Функции сдвига центров кластеров

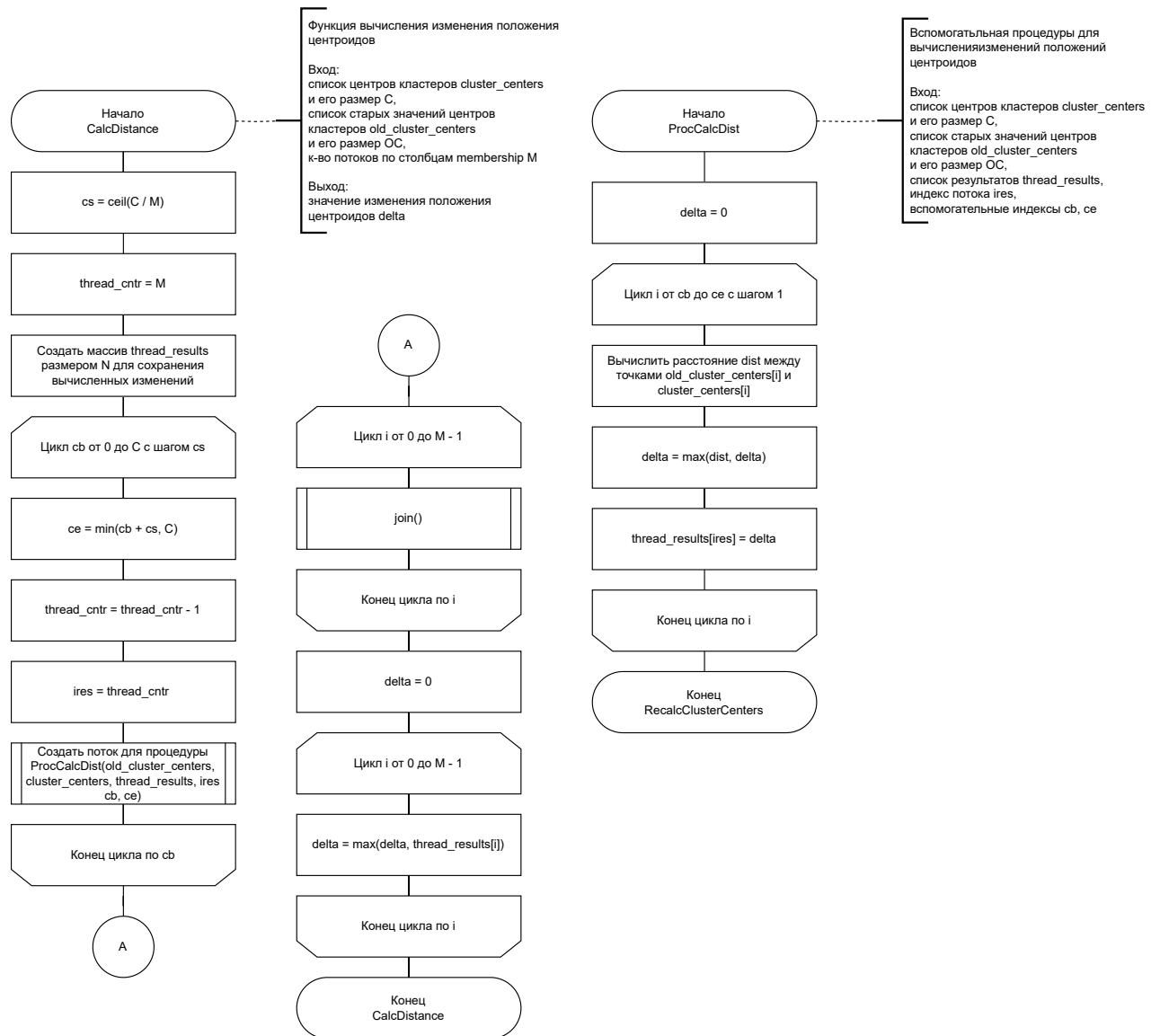


Рисунок 2.4 – Функции расчета изменения позиции центров кластеров

Вывод

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

3 Технологическая часть

3.1 Средства реализации

Для реализации 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 основной функции нечеткого алгоритма с-средних.

На листинге 3.2 представлена реализация функции расчета элементов матрицы принадлежности.

На листинге 3.3 показана реализация функции перерасчета положений центров кластеров.

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

Листинг 3.1 – Реализация основной функции нечеткого алгоритма с-средних

```
1 void c_means_parallel(  
2     membership_t &membership, point_vec_t &cluster_centers,  
3     const point_vec_t &data,  
4     double m, double conv_threshold, int max_iters)  
5 {  
6     int iters = 0;  
7     double delta = conv_threshold + 1.0;  
8     while (iters < max_iters && delta > conv_threshold)  
9     {  
10         std::vector<std::thread> threads;  
11         mt_calc_membership(membership, data, cluster_centers, m,  
12             2, 2);  
13  
14         auto old_cluster_centers = cluster_centers;  
15         mt_recalc_clustercenters(membership, cluster_centers,  
16             data, m, 2);  
17         delta = mt_calc_distance(cluster_centers,  
18             old_cluster_centers, 2);  
19         ++iters;  
20     }  
21 }
```

Листинг 3.2 – Реализация функции расчета элементов матрицы принадлежности

```
1 static void mt_calc_membership(  
2     membership_t &membership,  
3     const point_vec_t &data, point_vec_t &cluster_centers,  
4     double m,  
5     int rows_n_thread, int cols_n_thread)  
6 {  
7     std::vector<std::thread> threads;  
8  
9     int data_step = ceil(float(data.size()) /  
10         float(rows_n_thread));  
11     int c_step = ceil(float(cluster_centers.size()) /  
12         float(cols_n_thread));  
13  
14     for (size_t data_from = 0; data_from < data.size();  
15         data_from += data_step)  
16     {  
17         for (size_t c_from = 0; c_from < cluster_centers.size();  
18             c_from += c_step)  
19         {  
20             int data_to = std::min<int>(data_from + data_step,  
21                 data.size());  
22             int center_to = std::min<int>(c_from + c_step,  
23                 cluster_centers.size());  
24             threads.emplace_back(  
25                 [ &, data_from, data_to, c_from, center_to ]  
26                 {  
27                     for (int i = data_from; i < data_to; ++i)  
28                     {  
29                         for (int j = c_from; j < center_to; ++j)  
30                         {  
31                             double sum = 0.0;  
32                             double dist1 = sqrt(  
33                                 pow(data[i][0] -  
34                                     cluster_centers[j][0], 2) +  
35                                 pow(data[i][1] -  
36                                     cluster_centers[j][1], 2)  
37                             );  
38                             for (size_t k = 0; k <  
39                                 cluster_centers.size(); ++k)
```

```

30         {
31             double dist2 = sqrt(
32                 pow(data[i][0] -
33                     cluster_centers[k][0], 2)
34                 +
35                 pow(data[i][1] -
36                     cluster_centers[k][1], 2)
37             );
38             sum += pow(dist1 / dist2, 2.0 /
39                 (m - 1.0));
40         }
41         membership[i][j] = 1.0 / sum;
42     }
43 }
44 for (auto &thr : threads)
45     thr.join();

```

Листинг 3.3 – Реализация функции перерасчета положений центров кластеров

```

1 static void mt_recalc_clustercenters(
2     membership_t &membership, point_vec_t &cluster_centers,
3     const point_vec_t &data,
4     double m, int n_threads)
5 {
6     std::vector<std::thread> threads;
7     int cltr_step = ceil(float(cluster_centers.size()) /
8         float(n_threads));
9     for (size_t cltr_beg = 0; cltr_beg < cluster_centers.size();
10         cltr_beg += cltr_step)
11     {
12         int cltr_end = std::min<int>(cltr_beg + cltr_step,
13             cluster_centers.size());
14         threads.emplace_back(
15             [ &, cltr_beg, cltr_end ]
16             {
17                 for (int j = cltr_beg; j < cltr_end; ++j)
18                 {
19                     double x_nom = 0.0, y_nom = 0.0, denom = 0.0;
20                     for (size_t i = 0; i < data.size(); ++i)

```

```

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

```

Листинг 3.4 – Реализация функции 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()) /
9                          float(n_threads));
10    for (size_t cltr_beg = 0; cltr_beg < cluster_centers.size();
11         cltr_beg += cltr_step)
12    {
13        int cltr_end = std::min<int>(cltr_beg + cltr_step,
14                                     cluster_centers.size());
15        int ires = --n_threads;
16        threads.emplace_back(
17            [ &, cltr_beg, cltr_end, ires ]
18            {
19                double delta = 0;
20                for (int i = cltr_beg; i < cltr_end; ++i)
21                {
22                    double distance = sqrt(
23                        pow(old_cluster_centers[i][0] -

```

```

21         cluster_centers[i][0], 2) +
        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 с-средних. Также была предоставлена реализация алгоритма.

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

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 с-средних;
3. реализована программа, выполняющая параллельную работу алгоритма с выводом информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Каримов К. Х., Е.А. В.* Теоретические основы кластеризации данных // . — Общество с ограниченной ответственностью "Научно-издательский центр «Вестник науки», 2023. — С. 242—247.
2. *Лосев Д. Г.* Разработка и сравнение параллельных реализаций итеративных алгоритмов кластеризации // Наука молодых - будущее России. — 2021.
3. C++ language. — URL: <https://en.cppreference.com/w/cpp/language>.