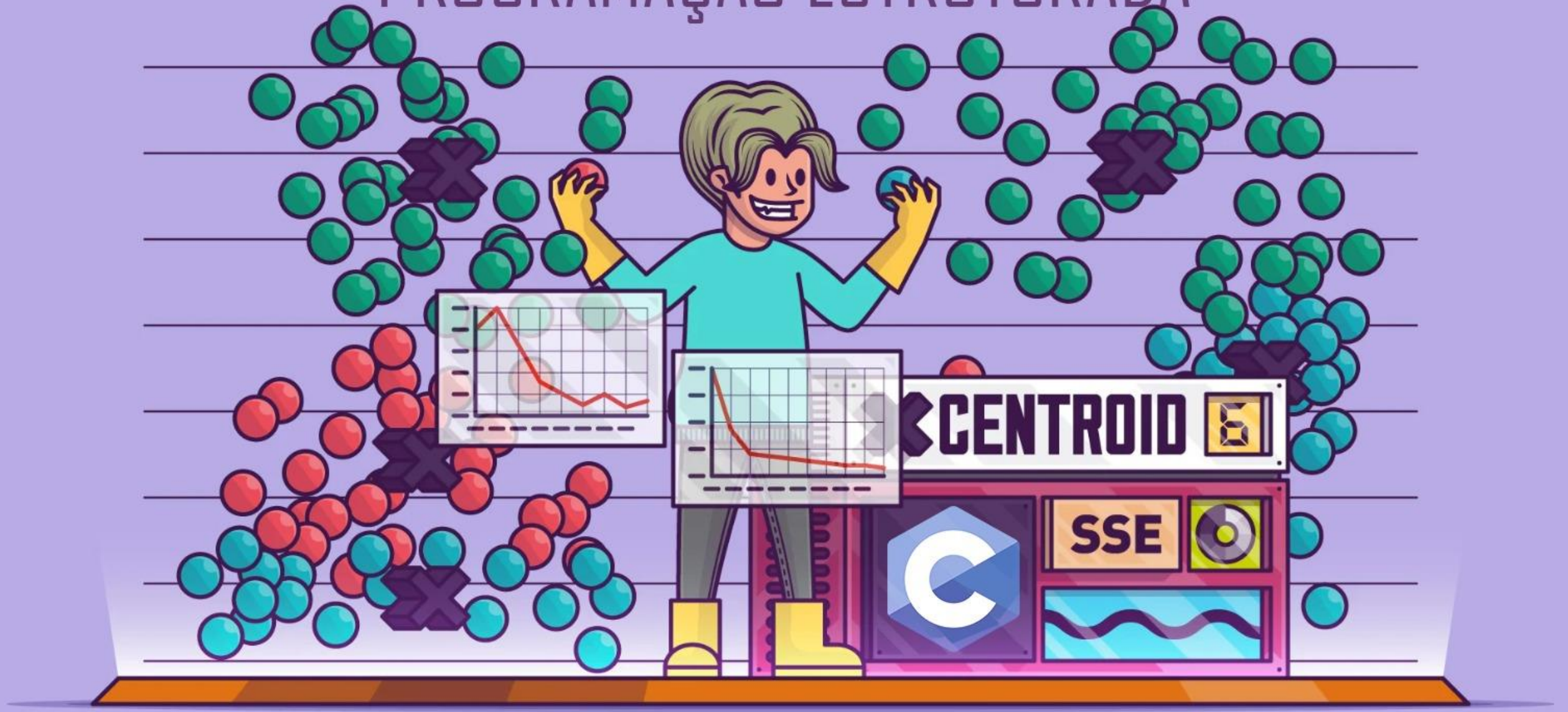
PROGRAMAÇÃO ESTRUTURADA



VITOR MENEGHETTI UGULINO DE ARAUJO

Universidade Federal da Paraíba - UFPB

Ciência de Dados e Inteligência Artificial

Programação Estruturada

Implementação do algoritmo K-MEANS na linguagem C

PROFESSOR:

VITOR MENEGHETTI UGULINO DE ARAUJO

Equipe:

DAVI NASIASENE AMORIM (20220056987)

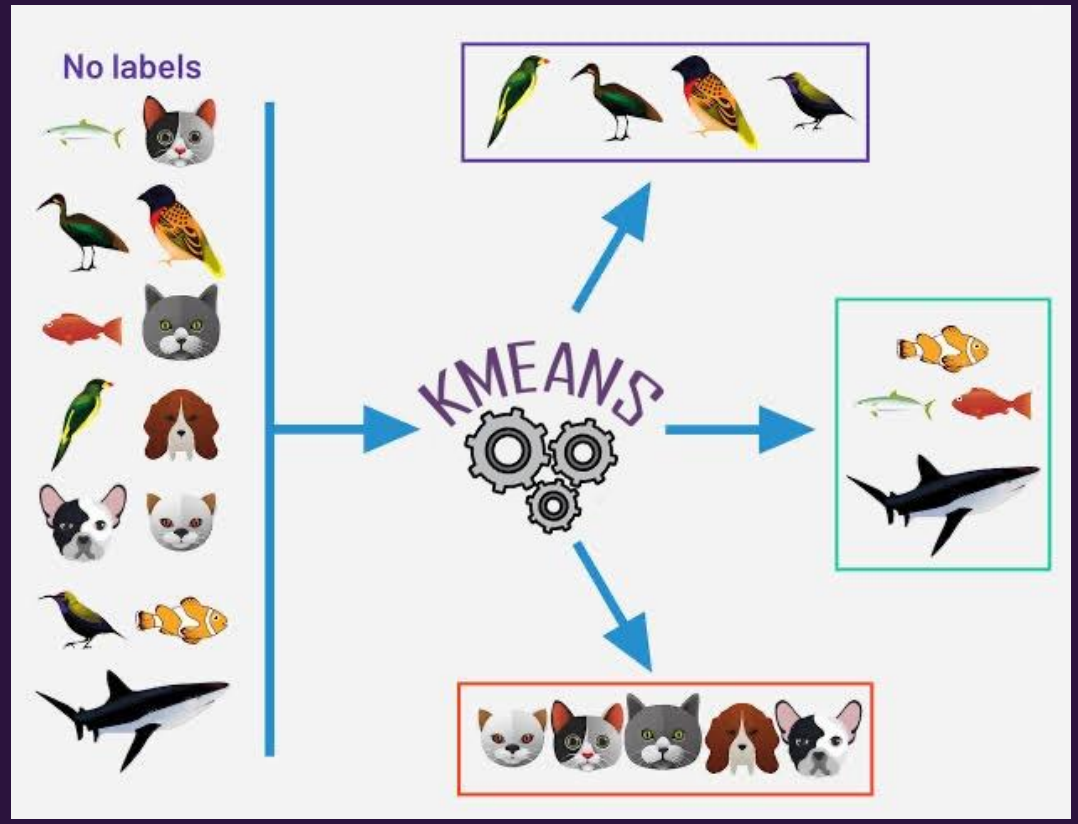
FRANCELINO TEOTONIO JUNIOR (20190035175)

GUILHERME BARBOZA DE SOUSA (20220007418)

IGÓ FERREIRA MELO SILVA (20220155214)

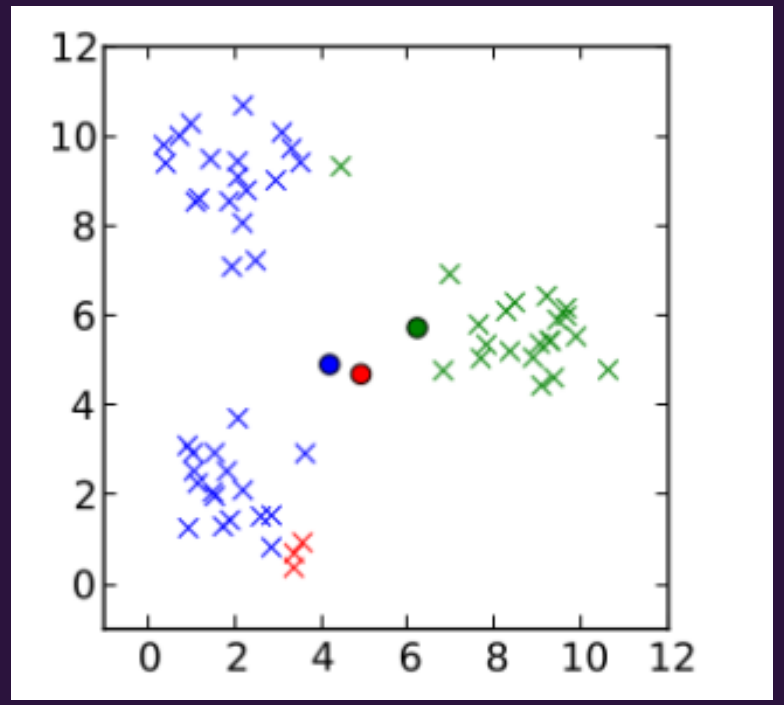
Introduzindo K-Means

O algoritmo K-MEANS é um método de aprendizado não supervisionado usado para agrupar dados em clusters. Ele busca dividir um conjunto de dados em grupos (clusters) com base em suas características ou similaridades.



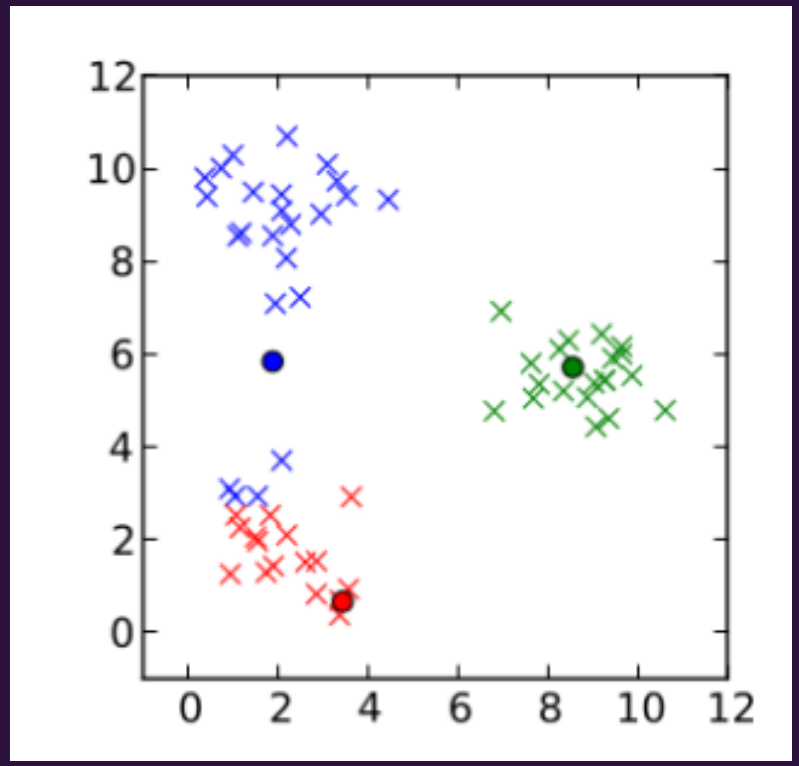
O algoritmo K-MEANS funciona da seguinte maneira:

- Inicialização: O algoritmo seleciona aleatoriamente K pontos como "centroides" iniciais. Esses centroides representam os pontos centrais de cada cluster.
- Associação: Cada ponto de dados é atribuído ao centroide mais próximo com base em uma medida de distância, geralmente a distância euclidiana.



O algoritmo K-MEANS funciona da seguinte maneira:

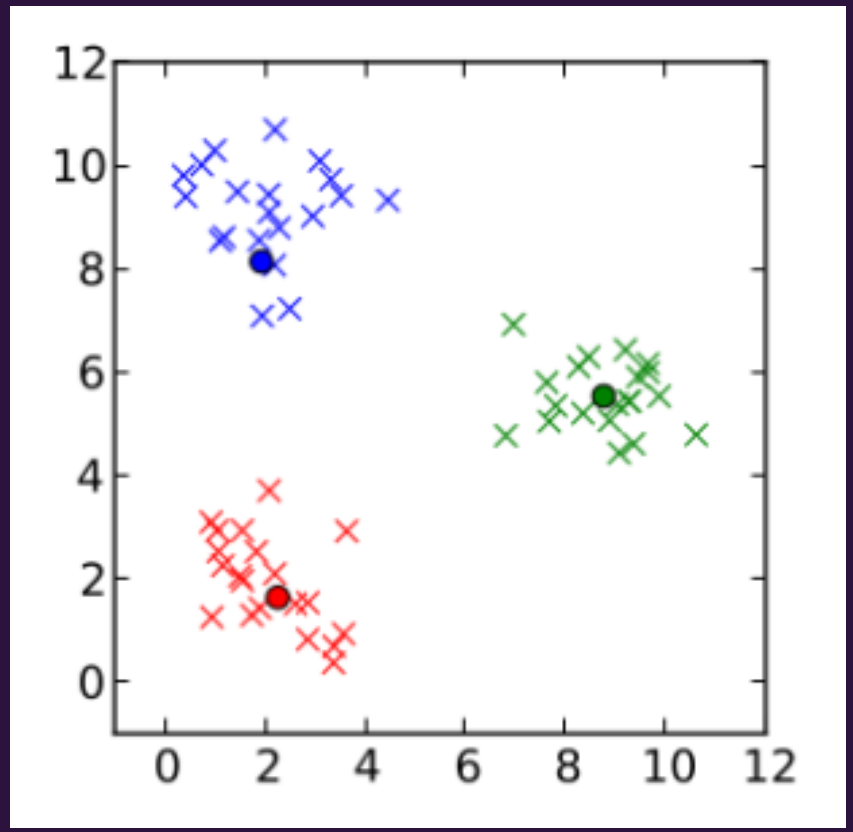
- Atualização: Os centroides dos clusters são atualizados recalculando-se as médias dos pontos atribuídos a cada cluster. Isso envolve mover o centroide para o centro dos pontos que estão atualmente associados a ele.
- Repetição: Os passos 2 e 3 são repetidos até que ocorra uma condição de parada. A condição de parada pode ser um número máximo de iterações, quando não ocorrem mais alterações significativas nos centroides ou quando os clusters convergem para uma configuração estável.



O algoritmo K-MEANS funciona da seguinte maneira:

O resultado final do algoritmo K-MEANS é um conjunto de K clusters, cada um contendo pontos de dados semelhantes. Os centroides finais podem ser usados para classificar novos pontos de dados em clusters existentes.

O K-MEANS é amplamente utilizado em várias áreas, como análise de dados, mineração de dados, reconhecimento de padrões e aprendizado de máquina. Ele pode ajudar a identificar padrões, segmentar clientes, agrupar documentos e muito mais.

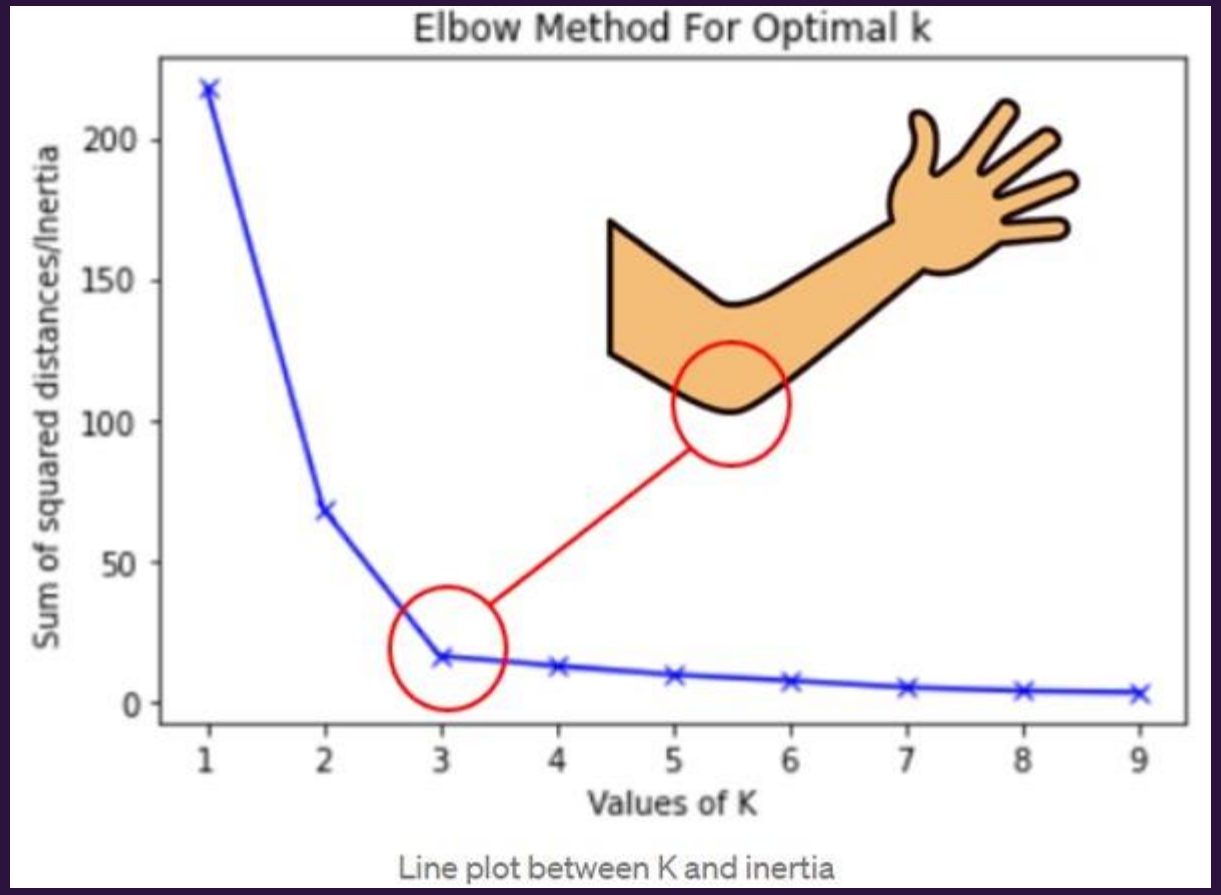


O algoritmo K-MEANS funciona da seguinte maneira:

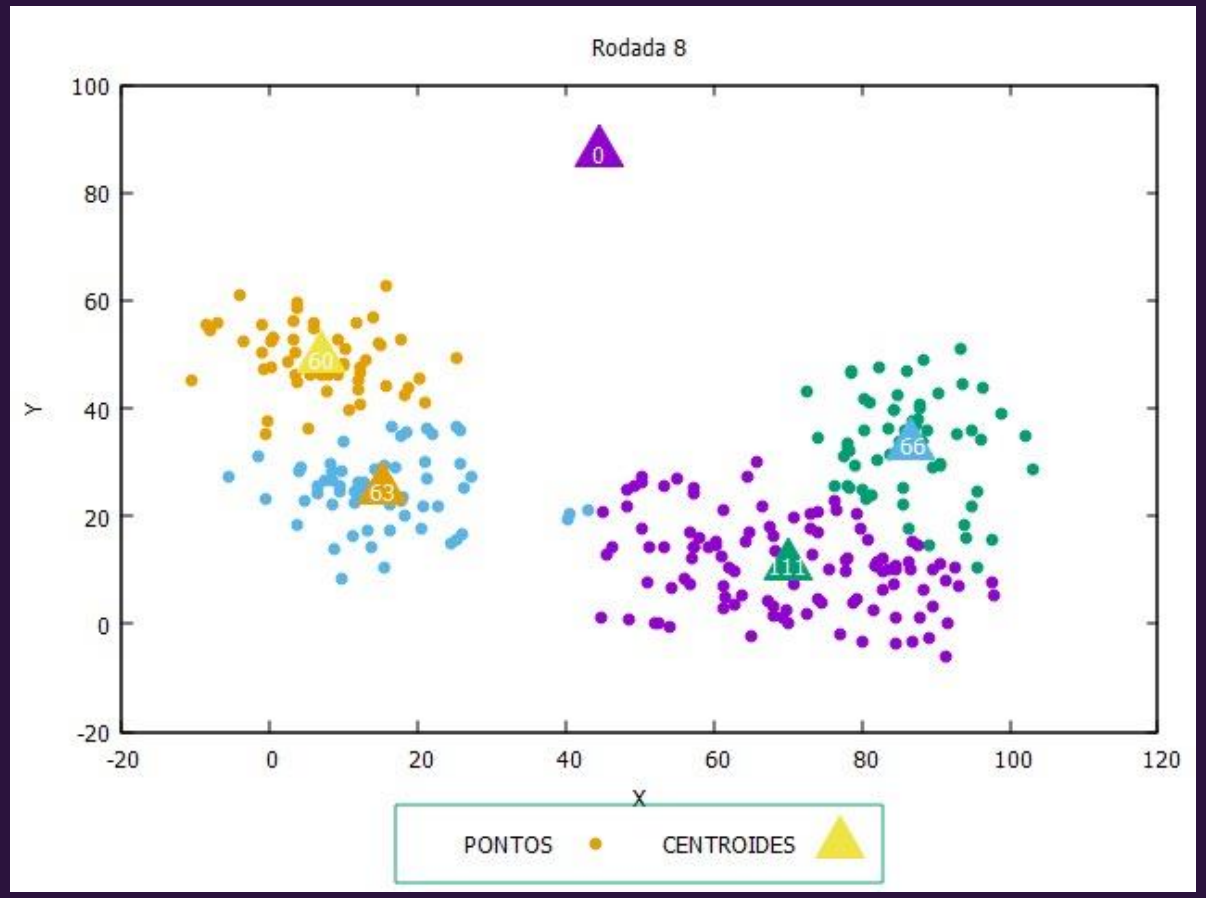
É importante lembrar que o K-MEANS requer que o número de clusters K seja especificado antecipadamente e que pode haver sensibilidade à inicialização dos centroides.

O algoritmo K-MEANS funciona da seguinte maneira:

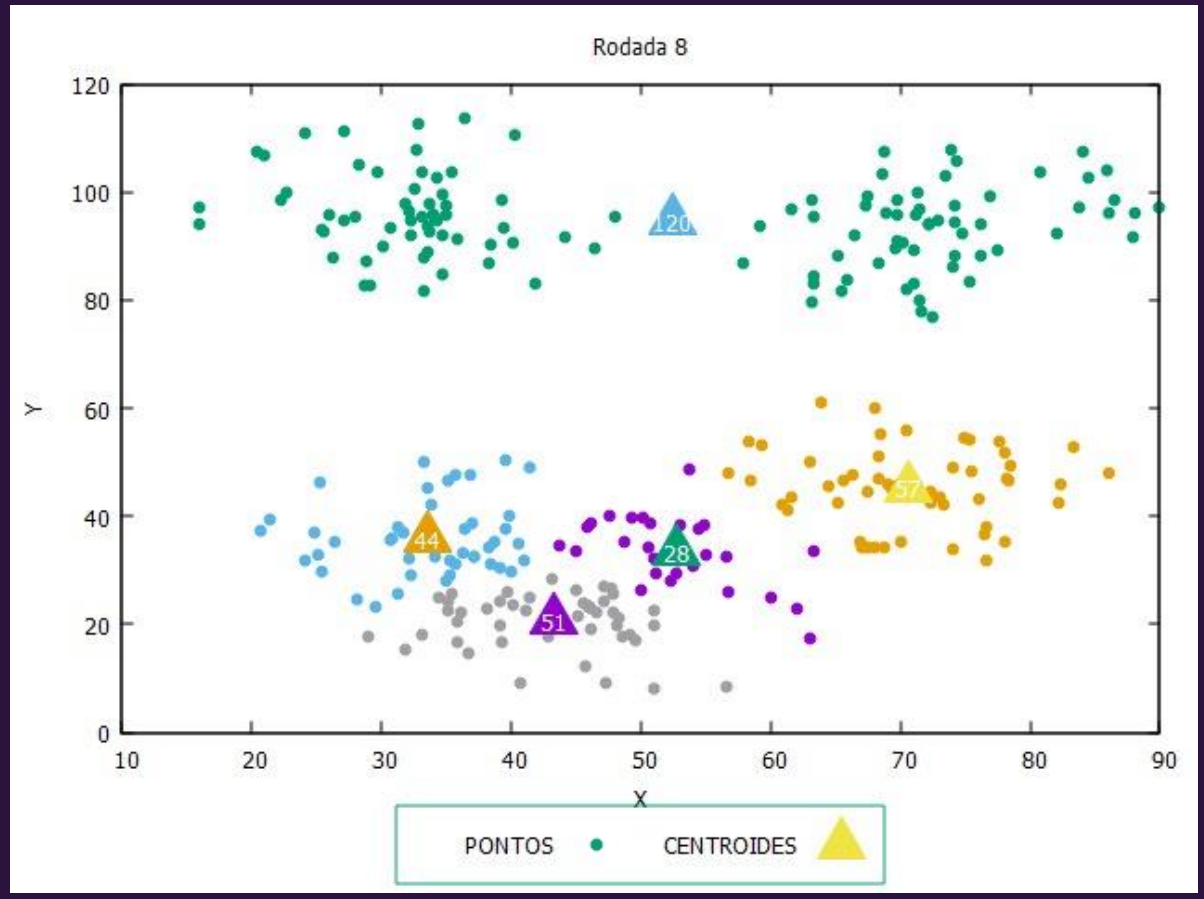
Algumas soluções são executar o algoritmo várias vezes com diferentes inicializações aleatórias e escolher o resultado com a menor soma das distâncias dos pontos aos centroides, usar métodos de inicialização mais sofisticados como o K-MEANS++, ou usar o método do cotovelo com a soma dos erros quadráticos.



A inicialização quando aleatória resulta em um centroide se mantendo isolado.



A inicialização resultou em uma má distribuição dos pontos.



Recursos empregados

- Estruturas condicionais
- Estruturas de repetição
- Vetores
- Ponteiros
- Funções
- Structs
- Manipulação de Arquivos

BIBLIOTECAS, ARQUIVOS E CONSTANTES

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "funcoes.h"

#define MAX_ITERATIONS 30
#define NUM_POINTS 300
#define NUM_CLUSTERS 5
#define ESPALHAMENTO 19
#define ARQUIVO_PONTOS "pontos.csv"
#define ARQUIVO_CENTROIDES "centroides.csv"
```

```c
typedef struct {
    int id;
    double x;
    double y;
    int cluster;
} Point;

typedef struct {
    int id;
    double x;
    double y;
    int count;
    double sumx;
    double sumy;
} Centroid;
```

STRUCTS

ESTRUTURAÇÃO DO MAIN

```c
int main() {

    // Definindo arrays de pontos e centroides
    Point points[NUM_POINTS];
    Centroid centroids[NUM_CLUSTERS];
    // Semente para geração de pontos e centroides
    srand(time(NULL));
    //Gera pontos
    criaPontos();
    //Gera centroides
    criaCentroides();
    // Plota o gráfico na condição inicial
    plotagem(0);
    // Pausa para visualização
    pausa();
```

ESTRUTURAÇÃO DO MAIN

```
    for (int iteration = 0; iteration < MAX_ITERATIONS;
iteration++) {
        printf("\nRodada %d - ", iteration+1);

        // Carrega dados dos arquivos csv
        carregaDados(points, centroids);

        // Associa os pontos aos centroides mais próximos
        associaPontosaosCentroides(points, centroids);
```

```
        // Associa os pontos aos centroides mais próximos
        associaPontosaosCentroides(points, centroids);

        // Atualiza a posição dos centroides
        atualizaPosicaoCentroides(points, centroids);

        // Atualiza o arquivo dos pontos
        atualizaPontos (points);

        // Atualiza o arquivo dos centroides
        atualizaCentroides (centroids);

        // Plota o gráfico
        plotagem(iteration+1);

        // Pausa para visualização
        pausa();
    }
    return 0;
}
```

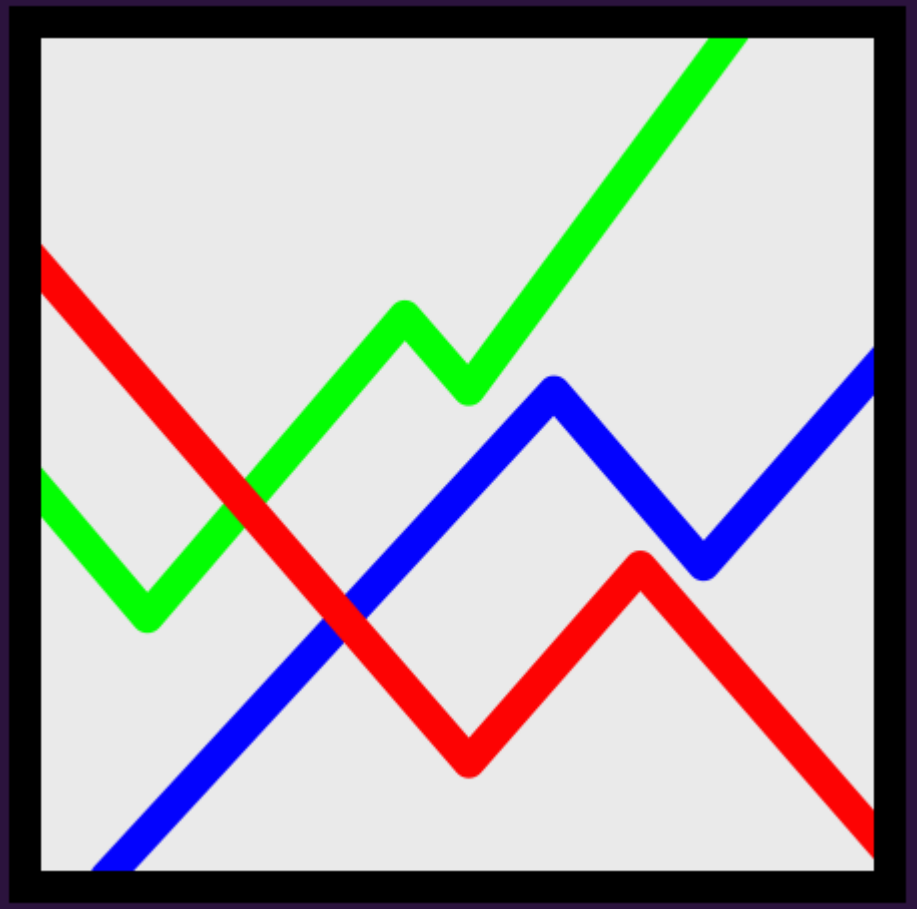
ESTRUTURAÇÃO DO MAIN

PLOTAGEM DO GRÁFICO

GNUPLOT

O Gnuplot é um programa de visualização de dados e criação de gráficos. A linguagem utilizada no Gnuplot é uma linguagem de script própria, projetada especificamente para criar gráficos e visualizações.

A linguagem do Gnuplot permite que você defina e manipule dados, aplique estilos aos gráficos, personalize os eixos, adicione legendas, títulos e rótulos, além de oferecer recursos avançados, como interpolação de superfície e ajuste de curvas.

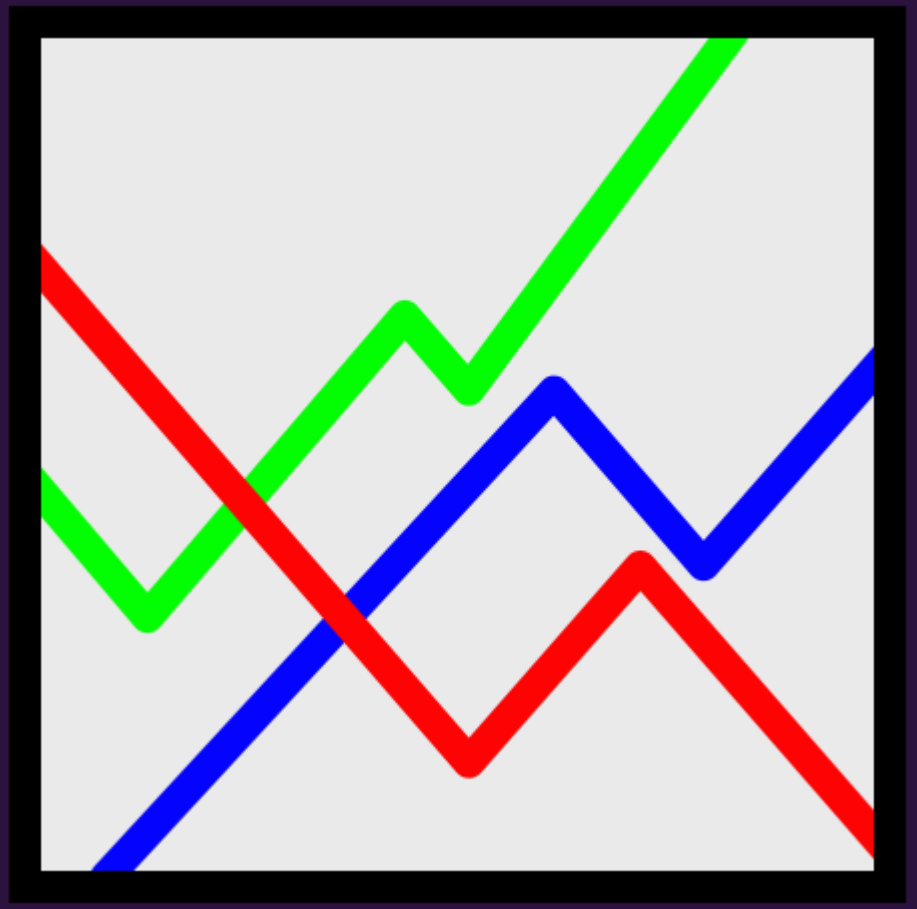


http://www.gnuplot.info

PLOTAGEM DO GRÁFICO
GNUPLOT

O Gnuplot é projetado para ser uma ferramenta de visualização independente e pode ser integrado a aplicativos e scripts em várias linguagens de programação, incluindo C.

Ao usar o Gnuplot com C, você pode automatizar a criação de gráficos, visualizar dados em tempo real ou incorporar gráficos em aplicativos.



http://www.gnuplot.info

FUNÇÃO RELATIVA À PLOTAGEM

```c
// Plota o gráfico
void plotagem(int iteration) {
    FILE *points = fopen(ARQUIVO_PONTOS, "r");
    FILE *centroids = fopen(ARQUIVO_CENTROIDES, "r");

    if (points == NULL || centroids == NULL) {
        printf("Erro ao abrir os arquivos.\n");
        return;}
    // Prepara o comando para chamar o gnuplot
    FILE *gnuplotPipe = popen("gnuplot -persist", "w");
    if (gnuplotPipe == NULL) {
        printf("Erro ao abrir o pipe do gnuplot.\n");
        return;}

    // Define parâmetros de plotagem do gnuplot
    fprintf(gnuplotPipe, "set title 'Rodada %d'\n", iteration);
    fprintf(gnuplotPipe, "set xlabel 'X'\n");
    fprintf(gnuplotPipe, "set ylabel 'Y'\n");
    fprintf(gnuplotPipe, "set key box linestyle 2 width 2 height 2 spacing 1
horizontal outside bottom center\n");
    fprintf(gnuplotPipe, "plot '-' using 1:2:3 with points lc variable pt 7 title
'PONTOS', \
                          '-' using 1:2:3 with points lc variable pt 9 ps 5
title 'CENTROIDES', \
                          '-' using 1:2:3:4 with labels textcolor rgb
'white' notitle\n");
```

FUNÇÃO RELATIVA À PLOTAGEM

```c
// Lê e plota os pontos
fprintf(gnuplotPipe, "# Pontos\n");
float x, y;
int id, clu, count;
while (fscanf(points, "%d,%f,%f,%d", &id, &x, &y, &clu) == 4){
    fprintf(gnuplotPipe, "%f %f %d\n", x, y, clu);
}
fprintf(gnuplotPipe, "e\n");

// Lê e plota os centroides
fprintf(gnuplotPipe, "# Centroides\n");
while (fscanf(centroids, "%d,%f,%f,%d", &id, &x, &y, &count) == 4) {
    fprintf(gnuplotPipe, "%f %f %d\n", x, y, id);
}
fprintf(gnuplotPipe, "e\n");

// Lê e plota os rótulos dos centroides
fprintf(gnuplotPipe, "# Rótulos dos Centroides\n");
rewind(centroids);  // Volta ao início do arquivo de centroides
while (fscanf(centroids, "%d,%f,%f,%d", &id, &x, &y, &count) == 4) {
    fprintf(gnuplotPipe, "%f %f %d %d\n", x, y, count, count);
}
fprintf(gnuplotPipe, "e\n");
```

```
    // Fecha os arquivos e o pipe do gnuplot
    fclose(points);
    fclose(centroids);
    pclose(gnuplotPipe);
}
```

FUNÇÃO RELATIVA À PLOTAGEM

```c
// Pausa para visualização
void pausa(){
    printf("Pressione ENTER para continuar...");
    while(getchar() != '\n');
}
```

PAUSA PARA VISUALIZAÇÃO

FUNÇÃO DE GERAÇÃO DE PONTOS

```c
// Gera pontos
void criaPontos() {
    FILE *pontos = fopen(ARQUIVO_PONTOS, "w");
    if (pontos == NULL) {
        printf("Erro ao abrir os arquivo.\n");
        return;
    }

    // Gera centros de agrupamentos
    Point centroids[NUM_CLUSTERS];
    for (int i = 0; i < NUM_CLUSTERS; i++) {
        centroids[i].id = i + 1;
        centroids[i].x = (float)(rand() % 100);  // Coordenadas de
referência - x entre 0 e 100
        centroids[i].y = (float)(rand() % 100);  // Coordenadas de
referência - y entre 0 e 100
    }
```

```
    // Gera pontos aleatóriamente
    Point points[NUM_POINTS];
    for (int i = 0; i < NUM_POINTS; i++) {
        int cluster = i % NUM_CLUSTERS;
        points[i].id = i;
        float radius = (float)(rand() % ESPALHAMENTO);  // Ajustar o
raio para controlar o espalhamento
        float angle = (float)(rand() % 360) * 3.1415 / 180.0;  // Ângulo
aleatório em radianos
        points[i].x = centroids[cluster].x + radius * cos(angle);
        points[i].y = centroids[cluster].y + radius * sin(angle);
        points[i].cluster = 0;  // Centroide associado inicializa em 0
    }

    // Salva pontos no CSV
    for (int i = 0; i < NUM_POINTS; i++) {
        fprintf(pontos, "%d,%.2f,%.2f,%d\n", points[i].id, points[i].x,
points[i].y, points[i].cluster);
    }

    fclose(pontos);
```

FUNÇÃO DE GERAÇÃO DE PONTOS

FUNÇÃO DE GERAÇÃO DOS CENTRÓIDES

```c
// Gera centroides
void criaCentroides() {
    FILE *centroides = fopen(ARQUIVO_CENTROIDES, "w");
    if (centroides == NULL) {
        printf("Erro ao abrir os arquivo.\n");
        return;
    }

    // Gera centroides aleatoriamente
    Centroid centroids[NUM_CLUSTERS];
    for (int i = 0; i < NUM_CLUSTERS; i++) {
        centroids[i].id = i + 1;
        centroids[i].x = (float)(rand() % 100); // Coordenadas x dos
centroides entre 0 e 100
        centroids[i].y = (float)(rand() % 100);  // Coordenadas y dos
centroides entre 0 e 100
        centroids[i].count = 0;
    }
```

FUNÇÃO DE GERAÇÃO DOS CENTRÓIDES

```
    // Salva centroides no CSV
    for (int i = 0; i < NUM_CLUSTERS; i++) {
        fprintf(centroides, "%d,%.2f,%.2f,%d\n", centroids[i].id,
centroids[i].x, centroids[i].y, centroids[i].count);
    }

    fclose(centroides);
}
```

CARREGANDO OS DADOS

```c
// Carrega dados dos arquivos csv
void carregaDados(Point points[], Centroid centroids[]) {
    // Carrega dados dos pontos
    FILE *pontos = fopen(ARQUIVO_PONTOS, "r");
    if (pontos == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }
    char line[100];
    for (int i = 0; i < NUM_POINTS; i++) {
        fgets(line, sizeof(line), pontos);
        sscanf(line, "%d,%lf,%lf,%d", &points[i].id, &points[i].x,
&points[i].y, &points[i].cluster);
    }
    fclose(pontos);
```

CARREGANDO OS DADOS

```
    // Carrega dados dos centroides
    FILE *centroides = fopen(ARQUIVO_CENTROIDES, "r");
    if (centroides == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }
    char line2[100];
    for (int i = 0; i < NUM_CLUSTERS; i++) {
        fgets(line2, sizeof(line2), centroides);
        sscanf(line2, "%d,%lf,%lf,%d", &centroids[i].id,
&centroids[i].x, &centroids[i].y, &centroids[i].count);
    }
    fclose(centroides);
}
```

ASSOCIANDO PONTOS AOS CENTROIDES

```
// Calcula a distância entre os pontos e os centroides
double distancia(Point points, Centroid centroids) {
    return sqrt(pow(points.x - centroids.x, 2) + pow(points.y -
centroids.y, 2));
}
```

```
// Associa os pontos aos centroides mais próximos
void associaPontosaosCentroides(Point points[], Centroid centroids[]) {

    int final = 0;
    for (int i = 0; i < NUM_POINTS; i++) {
        double minDistance = INFINITY;
        int closestCentroid = 0;
        for (int j = 0; j < NUM_CLUSTERS; j++) {
            double d = distancia(points[i], centroids[j]);
            if (d < minDistance) {
                minDistance = d;
                closestCentroid = j;
            }
        }
        if(points[i].cluster != closestCentroid){
            printf(".");
            final = 1;
        }
        points[i].cluster = closestCentroid; // Altera o centroide
associado ao ponto
    }
```

ASSOCIANDO PONTOS AOS CENTROIDES

```
    // Quando não houve qualquer alteração de associação
    if (final == 0){
        printf("Estabilidade atingida. ");
        pausa();
        exit(0); // Finaliza com sucesso
    }
}
```

ASSOCIANDO PONTOS AOS CENTROIDES

ATUALIZANDO POSIÇÃO DOS CENTROIDES

```
// Atualiza a posição dos centroides
void atualizaPosicaoCentroides(Point points[], Centroid centroids[]) {
    // Zera as posições dos centroides
    for (int k = 0; k < NUM_CLUSTERS; k++) {
        centroids[k].sumx = 0;
        centroids[k].sumy = 0;
        centroids[k].count = 0;}
    // Soma as posições dos pontos associados a cada centroide
    for (int i = 0; i < NUM_POINTS; i++) {
        int cluster = points[i].cluster;
        centroids[cluster].sumx = centroids[cluster].sumx + points[i].x;
        centroids[cluster].sumy = centroids[cluster].sumy + points[i].y;
        centroids[cluster].count++;}
    // Calcula a média das posições dos pontos associados a cada
centroide
    for (int j = 0; j < NUM_CLUSTERS; j++) {
        if (centroids[j].count > 0) {
            centroids[j].x = centroids[j].sumx / centroids[j].count;
            centroids[j].y = centroids[j].sumy / centroids[j].count;
        }
    }
}
```

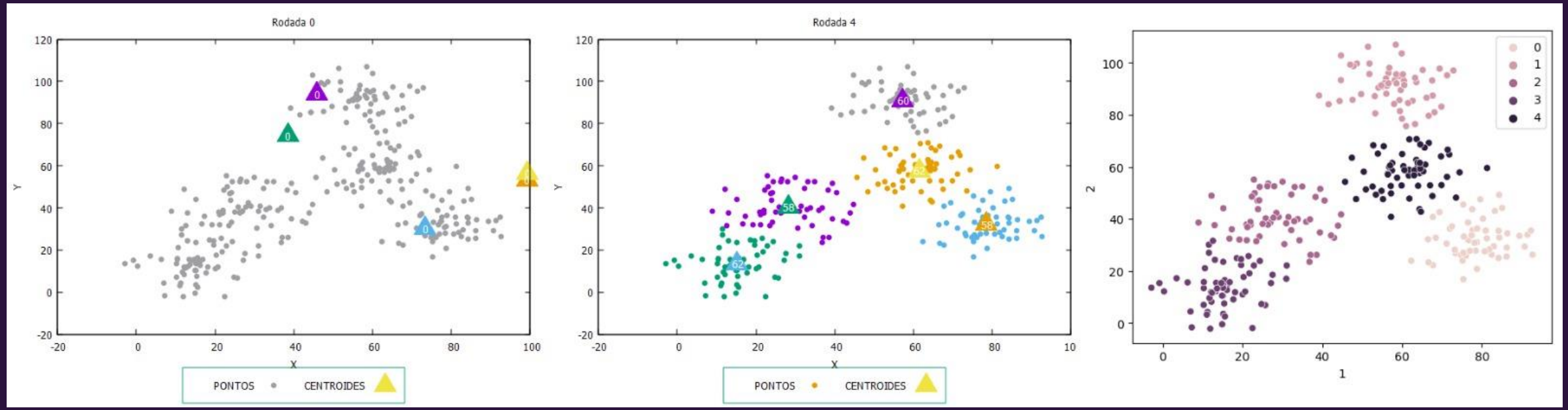
```c
// Atualiza o arquivo dos pontos
void atualizaPontos (Point points[]){
    FILE *pontos = fopen(ARQUIVO_PONTOS, "w");
    if (pontos == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);}
    for (int i = 0; i < NUM_POINTS; i++) {
        fprintf(pontos, "%d,%lf,%lf,%d\n", points[i].id, points[i].x,
points[i].y, points[i].cluster);
    }
    fclose(pontos);
}
```

ATUALIZA ARQUIVO DOS PONTOS

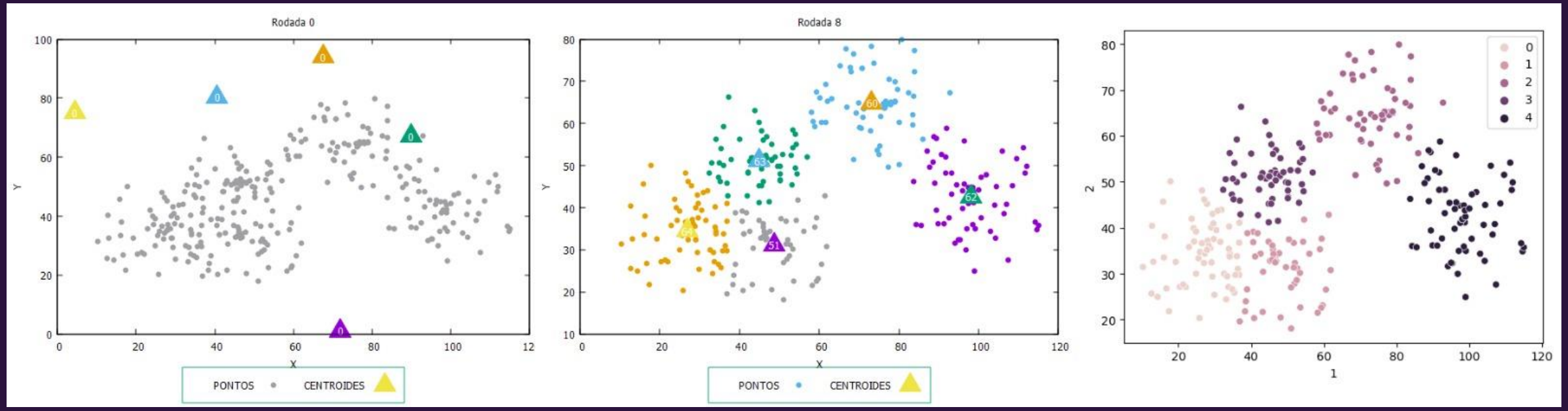
```c
// Atualiza o arquivo dos centroides
void atualizaCentroides (Centroid centroids[]){
    FILE *centroides = fopen(ARQUIVO_CENTROIDES, "w");
    if (centroides == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }
    for (int j = 0; j < NUM_CLUSTERS; j++) {
        fprintf(centroides, "%d,%lf,%lf,%d\n", centroids[j].id,
centroids[j].x, centroids[j].y, centroids[j].count);
    }
    fclose(centroides);
}
```

ATUALIZA ARQUIVO DAS CENTROIDES

COMPARANDO COM O KMEANS DO PYTHON



COMPARANDO COM O KMEANS DO PYTHON



OBRIGADO!