Задание 4

Выполнил **Сергей Цветков**

Группа **PY_gr2**

In [174]:

```
import pandas as pd
import numpy as np
%pylab inline
plt.style.use('seaborn-dark')
import warnings
warnings.filterwarnings("ignore") # отключение варнингов
pd.set_option('display.max_columns', None) # pd.options.display.max_columns = None
# pd.set_option('display.max_rows', None) # не прятать столбцы при выводе дата-фреймов
import matplotlib.pyplot as plt
import matplotlib as mpl
plt.rc('font', size=14)
# увеличим дефолтный размер графиков
from pylab import rcParams
rcParams['figure.figsize'] = 10, 8
```

```
Populating the interactive namespace from numpy and matplotlib
```

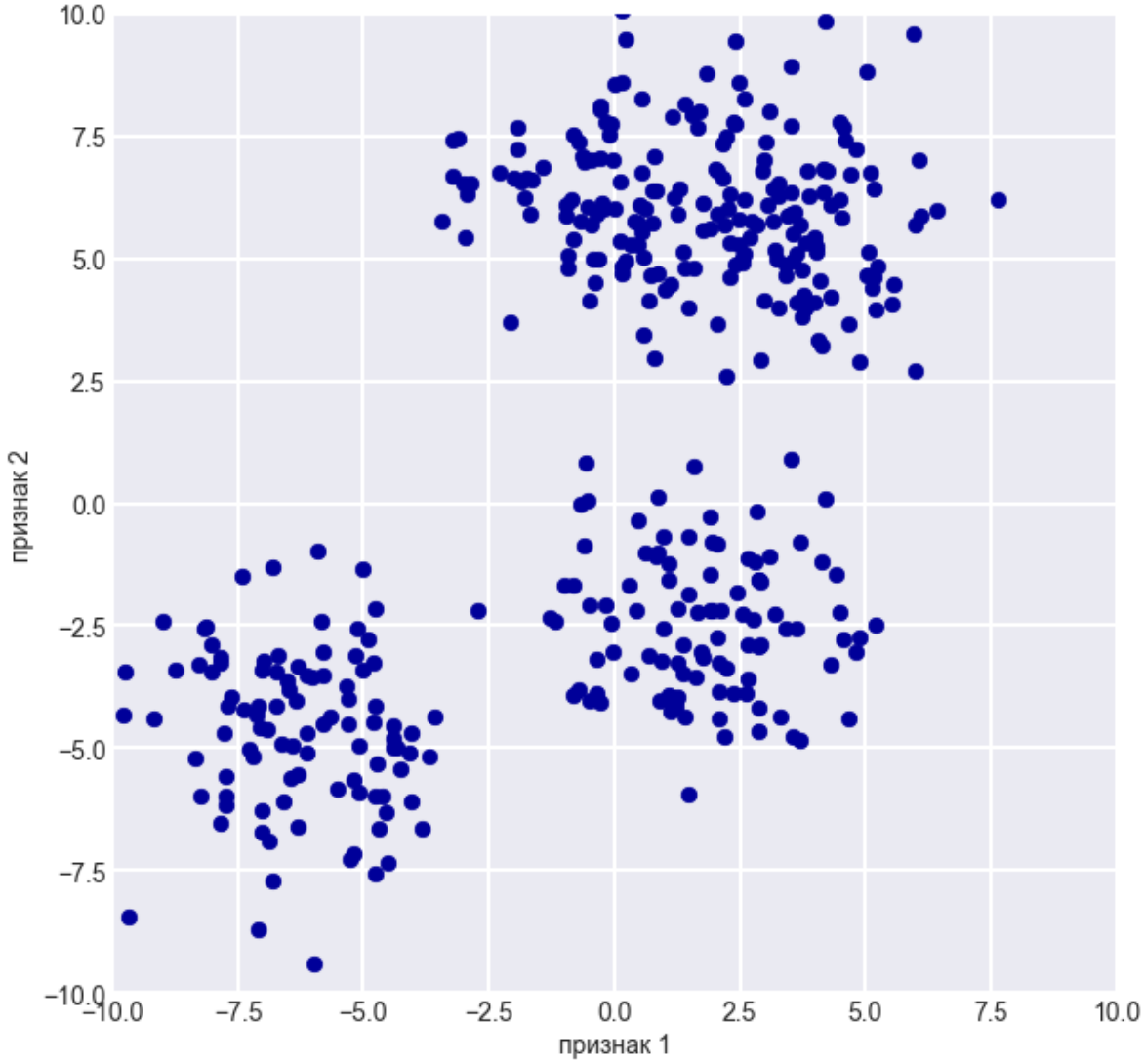
Встроенные датасеты (кучки)

In [175]:

```
# Генерация набора точек (кучек) с гауссовым распределением
from sklearn.datasets import make_blobs
seed = 100 # Разброс
n_data = 400 # Количество точек
n_clusters = 4 # Количество кластеров
n_centers = 4 # Количество центров скопления точек

# n_features = сколько столбцов или объектов будет иметь сгенерированные наборы данных

blobs, blob_labels = make_blobs(n_samples=n_data, n_features=2,
                                centers=n_centers, random_state=seed,
                                cluster_std=2)

blobs *= 0.75
blobs += 1.1

# Рисуем палитру
plt.figure(figsize=(10, 10))
# Рисуем зависимость
plt.scatter(blobs[:, 0], blobs[:, 1], s=70,
            color='#000099',  linewidth=0.8)

plt.xlabel('признак 1')
plt.ylabel('признак 2')

plt.grid(lw=2)
plt.xlim([-10, 10])
plt.ylim([-10, 10])
# c=clusters_blob1
```

Out[175]:

```
(-10, 10)
```



In [176]:

```
make_blobs
```

Out[176]:

```
<function sklearn.datasets._samples_generator.make_blobs(n_samples=100, n_fe
atures=2, centers=None, cluster_std=1.0, center_box=(-10.0, 10.0), shuffle=T
rue, random_state=None)>
```

In [177]:

```
blobs
```

```
       [ 3.41004885e-01, -3.46635777e+00],
       [ 3.98162142e+00,  4.09020479e+00],
       [ 5.17592658e+00,  4.61766028e+00],
       [-5.10303614e-01, -2.06263303e+00],
       [ 2.21266278e+00,  7.50202598e+00],
       [-8.33496473e+00, -5.19369962e+00],
       [ 4.51315134e+00,  5.83533437e+00],
       [-5.80000728e+00, -4.50290700e+00],
       [ 4.28376543e-01, -2.19650080e+00],
       [ 1.55194830e-01,  4.67377275e+00],
       [ 3.86532821e+00,  6.29051212e+00],
       [ 4.54627546e+00,  7.68456311e+00],
       [-4.39616332e+00, -4.79026598e+00],
       [-2.01910328e+00,  6.62921895e+00],
       [ 2.87675412e+00, -4.18494773e+00],
       [ 2.37282186e+00,  7.79098633e+00],
       [-4.81401633e+00, -3.26746850e+00],
       [ 4.00055447e+00,  5.13381647e+00],
       [ 2.10784106e+00, -2.20413640e+00],
```

In [178]:

```
blob_labels
```

Out[178]:

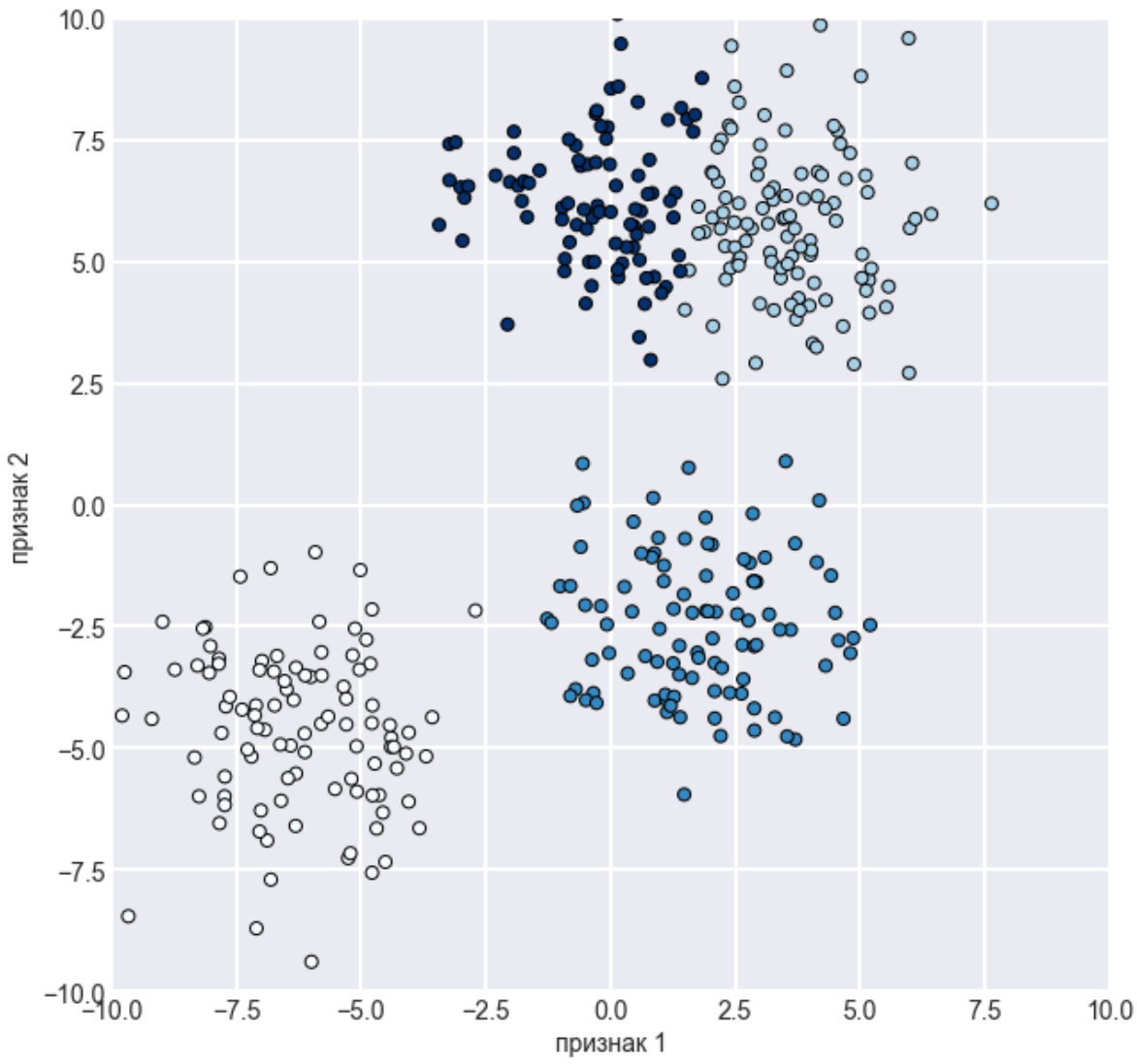
```
array([2, 1, 1, 3, 0, 2, 2, 1, 3, 2, 0, 3, 0, 1, 0, 2, 0, 3, 3, 0, 3, 2,
       3, 2, 0, 1, 3, 3, 2, 1, 0, 1, 2, 3, 0, 1, 0, 2, 3, 2, 0, 0, 0, 3,
       0, 3, 3, 1, 1, 1, 0, 0, 2, 3, 3, 0, 2, 1, 1, 1, 1, 2, 1, 1, 3, 2,
       2, 3, 2, 3, 2, 1, 0, 0, 0, 1, 3, 0, 1, 1, 2, 0, 0, 2, 0, 2, 3, 0,
       1, 0, 1, 2, 2, 1, 1, 2, 3, 3, 3, 3, 1, 2, 1, 0, 0, 3, 3, 1, 2, 1,
       0, 0, 2, 2, 2, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 3, 1, 0, 1, 0,
       0, 2, 0, 0, 1, 0, 3, 3, 3, 2, 1, 3, 3, 2, 2, 0, 3, 2, 3, 3, 1, 2,
       3, 0, 0, 2, 1, 2, 2, 3, 1, 3, 3, 3, 2, 3, 3, 2, 0, 2, 2, 1, 2, 2,
       3, 0, 3, 1, 3, 2, 0, 1, 2, 1, 0, 0, 3, 3, 3, 2, 3, 1, 0, 2, 0, 3,
       3, 2, 3, 1, 0, 2, 0, 3, 3, 3, 1, 2, 1, 0, 0, 1, 0, 1, 0, 2, 1, 0,
       1, 0, 1, 2, 3, 3, 2, 3, 2, 0, 1, 1, 2, 3, 1, 1, 1, 2, 0, 1, 3, 1,
       3, 1, 0, 1, 3, 1, 2, 2, 3, 0, 1, 3, 0, 0, 2, 3, 2, 2, 0, 3, 1, 0,
       2, 0, 0, 0, 0, 0, 1, 2, 2, 2, 0, 3, 0, 2, 0, 0, 3, 3, 3, 2, 3, 1,
       0, 0, 0, 1, 0, 2, 2, 3, 1, 2, 3, 0, 1, 3, 0, 2, 1, 1, 3, 1, 3, 2,
       2, 2, 0, 3, 1, 1, 0, 2, 0, 2, 3, 1, 0, 1, 2, 2, 3, 2, 0, 1, 1, 2,
       2, 3, 3, 0, 0, 3, 0, 1, 2, 1, 1, 2, 1, 0, 3, 3, 1, 2, 3, 3, 3, 2,
       0, 3, 1, 2, 0, 1, 0, 3, 1, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2, 3, 1, 1,
       3, 3, 0, 1, 0, 1, 1, 2, 2, 2, 0, 3, 1, 1, 0, 3, 2, 3, 3, 3, 1, 3,
       3, 0, 2, 2])
```

In [179]:

```
from sklearn.cluster import KMeans

clusters = KMeans(n_clusters=4, random_state=0, init='random').fit_predict(blobs)



plt.figure(figsize=(10, 10))

plt.scatter(blobs[:, 0], blobs[:, 1],
            c=clusters, s=50, cmap='Blues',
            edgecolors='black', linewidth=1.0)

plt.xlabel('признак 1')
plt.ylabel('признак 2')

plt.grid(lw=2)
plt.xlim([-10, 10])
plt.ylim([-10, 10])
```

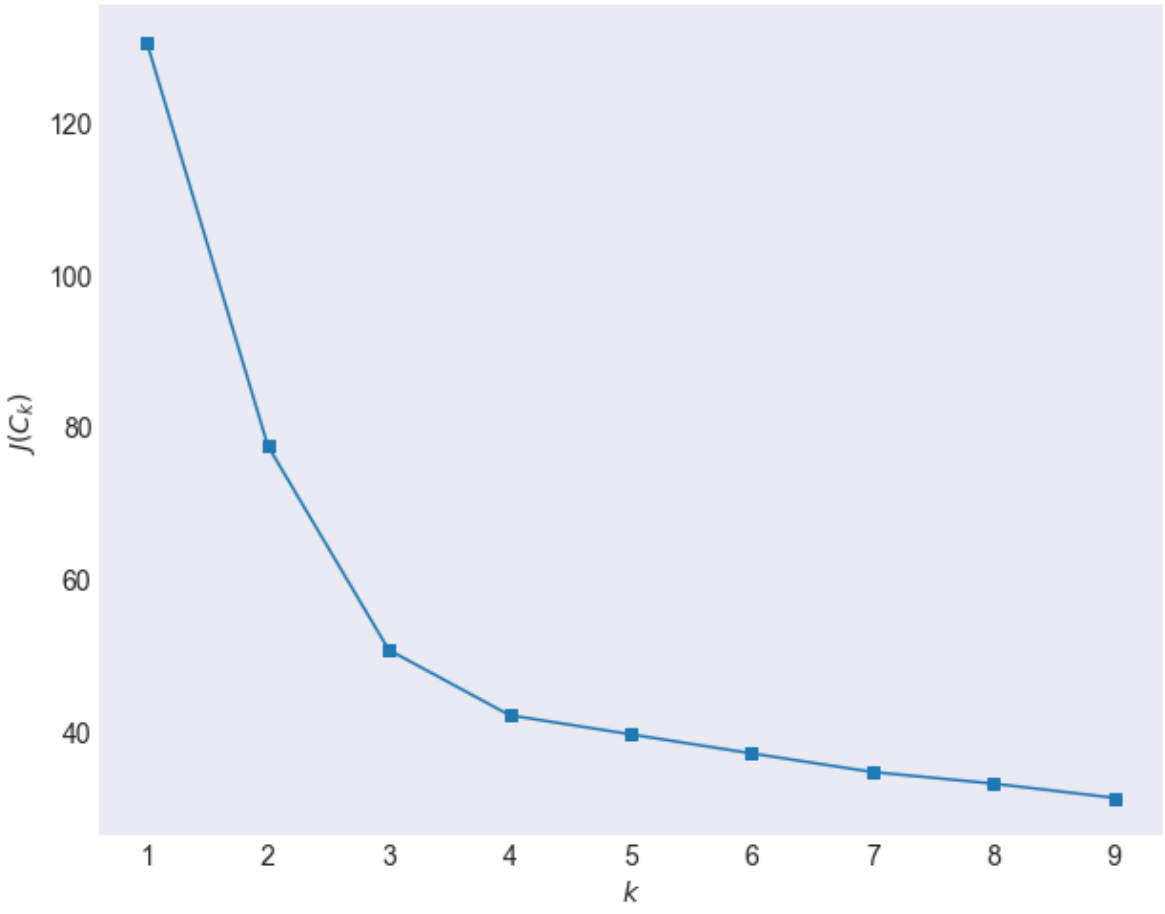
Out[179]:

```
(-10, 10)
```



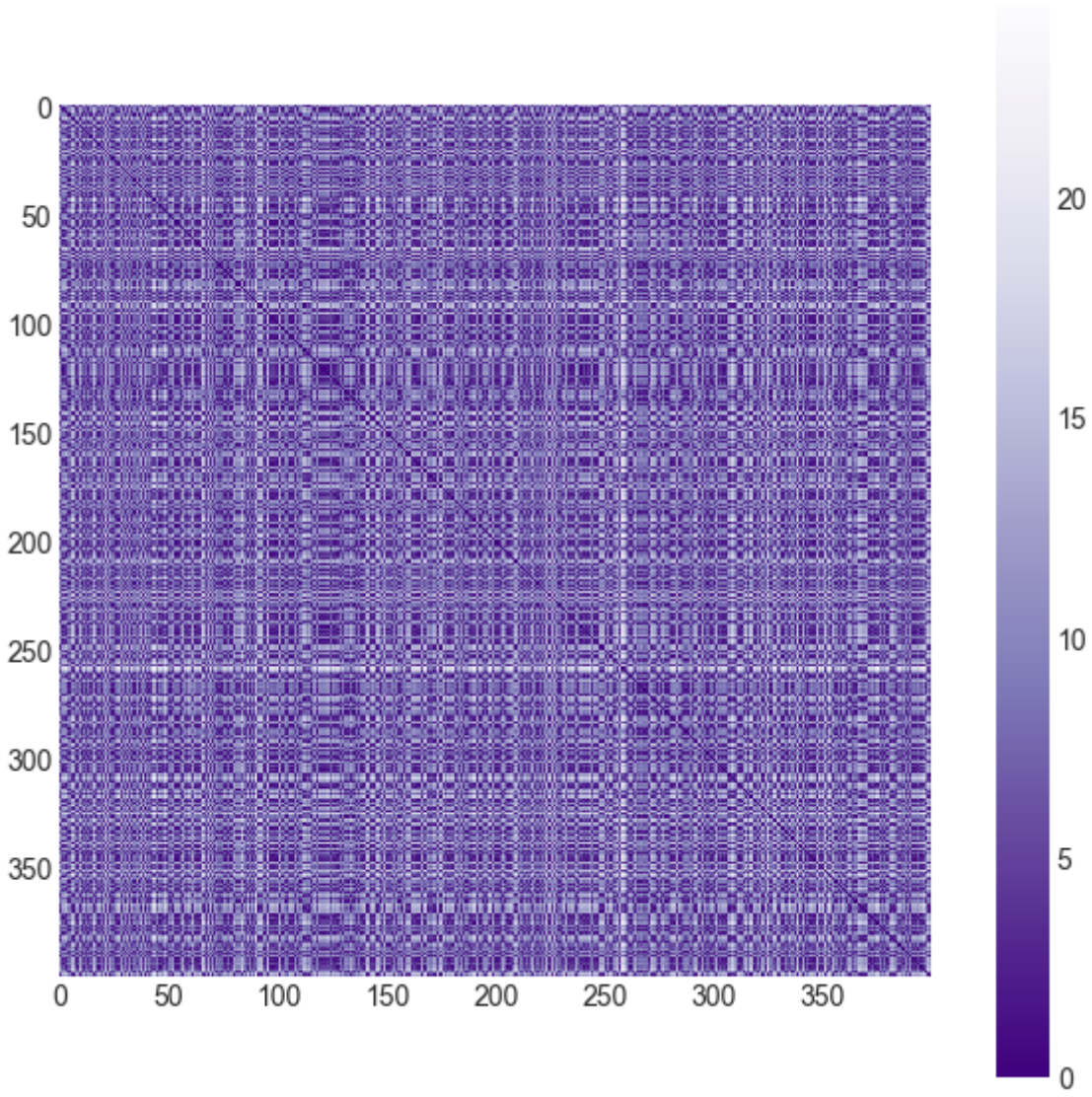
In [180]:

```
# Строим диаграмму, отображающую коэфициент эффективности для получения оптимального количч
from sklearn.cluster import KMeans

inertia = []
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, random_state=1).fit(blobs)
    inertia.append(np.sqrt(kmeans.inertia_))

plt.plot(range(1, 10), inertia, marker='s');
plt.xlabel('$k$')
plt.ylabel('$J(C_k)$');
```



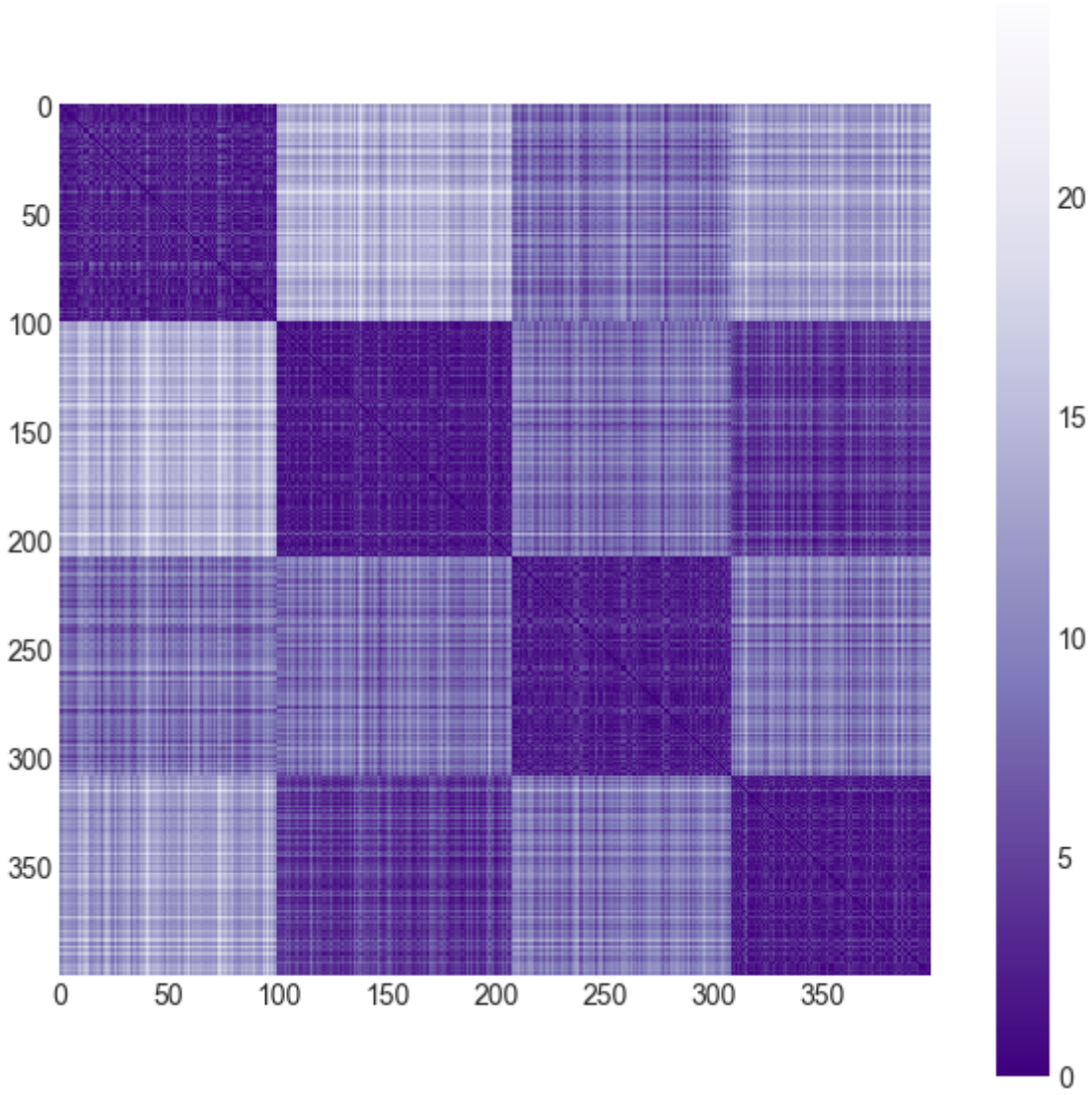
In [181]:

```
D = (blobs[:,0][:, np.newaxis] - blobs[:,0]) ** 2
D += (blobs[:,1][:, np.newaxis] - blobs[:,1]) ** 2
D = np.sqrt(D)

plt.figure(figsize=(10, 10))
plt.imshow(D, cmap='Purples_r')
plt.colorbar(orientation='vertical', pad=0.06);
```



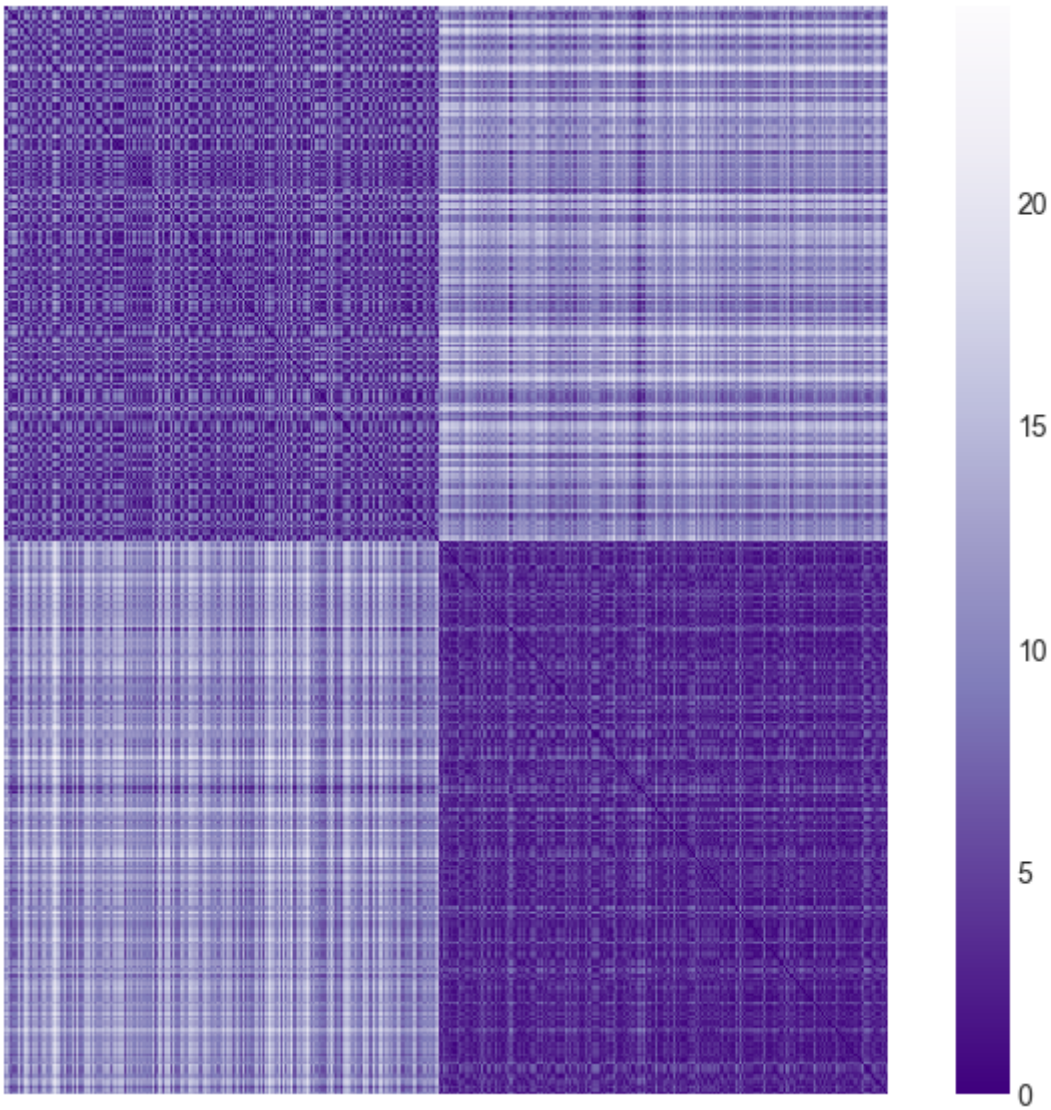
In [182]:

```
#clusters = KMeans(n_clusters=, random_state=0, init='random').fit_predict(blobs)

i  = np.argsort(clusters)
blobs2 = blobs[i, :]

D = (blobs2[:,0][:, np.newaxis] - blobs2[:,0]) ** 2
D += (blobs2[:,1][:, np.newaxis] - blobs2[:,1]) ** 2
D = np.sqrt(D)

plt.figure(figsize=(10, 10))
plt.imshow(D, cmap='Purples_r')
plt.colorbar(orientation='vertical', pad=0.06);
```



In [183]:

```
clusters = KMeans(n_clusters=2, random_state=0, init='random').fit_predict(blobs)

i  = np.argsort(clusters)
blobs2 = blobs[i, :]

D = (blobs2[:,0][:, np.newaxis] - blobs2[:,0]) ** 2
D += (blobs2[:,1][:, np.newaxis] - blobs2[:,1]) ** 2
D = np.sqrt(D)

plt.figure(figsize=(10, 10))
plt.imshow(D, cmap='Purples_r', aspect='auto')
plt.xticks([])
plt.yticks([])
plt.colorbar(orientation='vertical', pad=0.06);
```

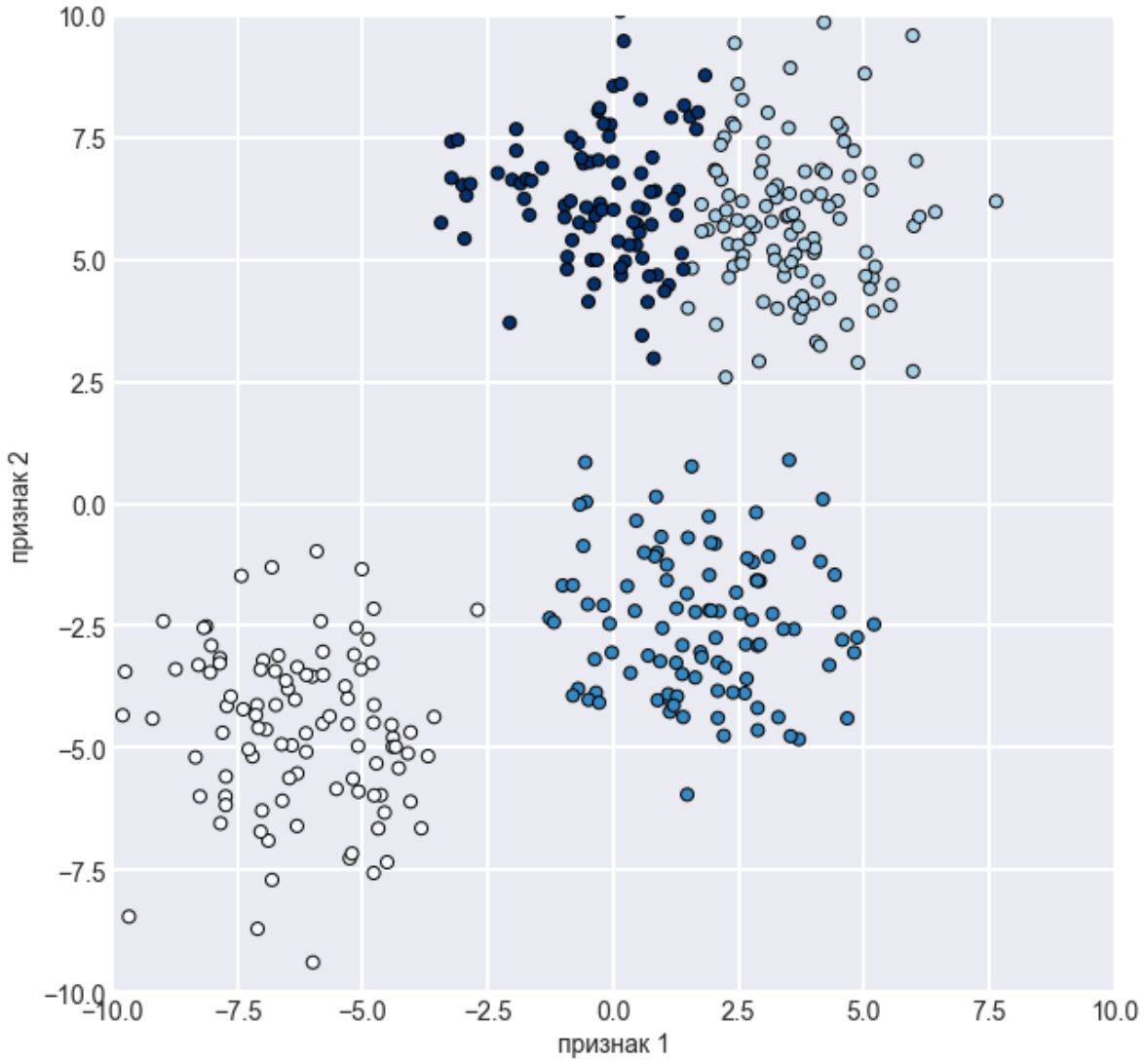


In [184]:

```
from sklearn.cluster import KMeans

clusters = KMeans(n_clusters=4, random_state=0, init='random').fit_predict(blobs)

plt.figure(figsize=(10, 10))

plt.scatter(blobs[:, 0], blobs[:, 1],
            c=clusters, s=50, cmap='Blues',
            edgecolors='black', linewidth=1.0)

plt.xlabel('признак 1')
plt.ylabel('признак 2')

plt.grid(lw=2)
plt.xlim([-10, 10])
plt.ylim([-10, 10])
```

Out[184]:

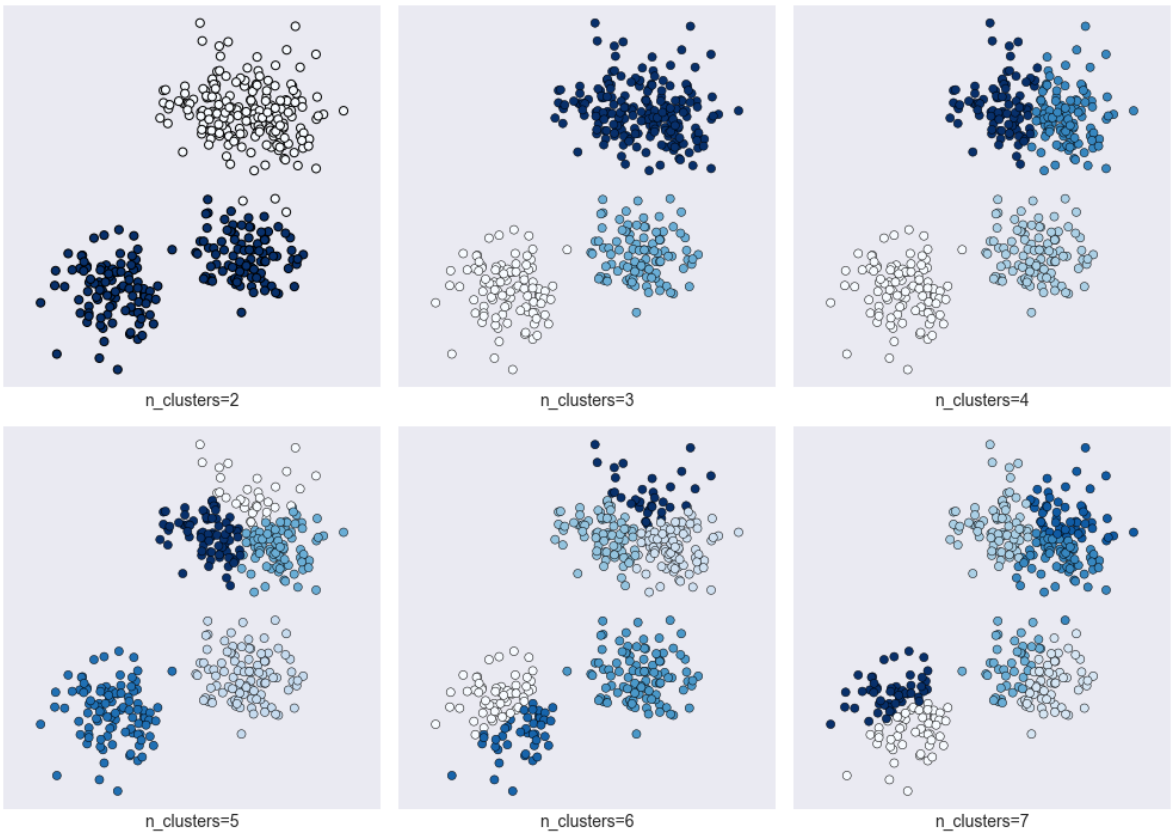
```
(-10, 10)
```



In [185]:

```
# Выводим диаграммы зависимостей от количества кластеров
import numpy as np
import matplotlib.pyplot as plt

clusters_blob1 = KMeans(n_clusters=2, random_state=1, init='random').fit_predict(blobs)
clusters_blob2 = KMeans(n_clusters=3, random_state=2, init='random').fit_predict(blobs)
clusters_blob3 = KMeans(n_clusters=4, random_state=3, init='random').fit_predict(blobs)
clusters_blob4 = KMeans(n_clusters=5, random_state=4, init='random').fit_predict(blobs)
clusters_blob5 = KMeans(n_clusters=6, random_state=5, init='random').fit_predict(blobs)
clusters_blob6 = KMeans(n_clusters=7, random_state=6, init='random').fit_predict(blobs)

# Generate data uniformly at random and run k-means
# uniform = np.random.rand(n_data, 2)
# clusters_uniform = KMeans(n_clusters=n_clusters, random_state=seed).fit_predict(uniform)


figure = plt.figure(figsize=(14, 10))

plt.subplot(231)
plt.scatter(blobs[:, 0], blobs[:, 1], c=clusters_blob1, s=50, cmap='Blues', edgecolors='bla
# plt.title("k-mean (1)", fontsize=14)
# plt.axis('off')
plt.axis('equal')
plt.xticks([])
plt.yticks([])
plt.xlabel('n_clusters=2')

plt.subplot(232)
plt.scatter(blobs[:, 0], blobs[:, 1], c=clusters_blob2, s=50, cmap='Blues', edgecolors='bla
# plt.title("k-mean (2)", fontsize=14)
#plt.axis('off')
plt.axis('equal')
plt.xticks([])
plt.yticks([])
plt.xlabel('n_clusters=3')

plt.subplot(233)
plt.scatter(blobs[:, 0], blobs[:, 1], c=clusters_blob3, s=50, cmap='Blues', edgecolors='bla
# plt.title("k-mean (2)", fontsize=14)
#plt.axis('off')
plt.axis('equal')
plt.xticks([])
plt.yticks([])
plt.xlabel('n_clusters=4')

plt.subplot(234)
plt.scatter(blobs[:, 0], blobs[:, 1], c=clusters_blob4, s=50, cmap='Blues', edgecolors='bla
# plt.title("k-mean (2)", fontsize=14)
#plt.axis('off')
plt.axis('equal')
plt.xticks([])
plt.yticks([])
plt.xlabel('n_clusters=5')

plt.subplot(235)
plt.scatter(blobs[:, 0], blobs[:, 1], c=clusters_blob5, s=50, cmap='Blues', edgecolors='bla
# plt.title("k-mean (2)", fontsize=14)
#plt.axis('off')
plt.axis('equal')
```

```
plt.xticks([])
plt.yticks([])
plt.xlabel('n_clusters=6')

plt.subplot(236)
plt.scatter(blobs[:, 0], blobs[:, 1], c=clusters_blob6, s=50, cmap='Blues', edgecolors='bla
# plt.title("k-mean (2)", fontsize=14)
#plt.axis('off')
plt.axis('equal')
plt.xticks([])
plt.yticks([])
plt.xlabel('n_clusters=7')


plt.tight_layout()
```



In [186]:

```
from scipy.cluster import hierarchy
from scipy.spatial.distance import pdist

X = np.zeros((150, 2))

np.random.seed(seed=42)
X[:50, 0] = np.random.normal(loc=0.0, scale=.3, size=50)
X[:50, 1] = np.random.normal(loc=0.0, scale=.3, size=50)

X[50:100, 0] = np.random.normal(loc=2.0, scale=.5, size=50)
X[50:100, 1] = np.random.normal(loc=-1.0, scale=.2, size=50)

X[100:150, 0] = np.random.normal(loc=-1.0, scale=.2, size=50)
X[100:150, 1] = np.random.normal(loc=2.0, scale=.5, size=50)

distance_mat = pdist(X) # pdist посчитает нам верхний треугольник матрицы попарных расстоян

Z = hierarchy.linkage(distance_mat, 'single') # linkage — реализация агломеративного алгори
plt.figure(figsize=(10, 5))
dn = hierarchy.dendrogram(Z, color_threshold=0.5)
```