Комитет по образованию Правительства Санкт-Петербурга

**САНКТ-ПЕТЕРБУРГСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОНЫХ ТЕХНОЛОГИЙ**

**Отчет по практической работе № 2**

**МДК 01.03 Разработка мобильных приложений**

**Тема: Разработка многопоточного приложения**

Выполнил

студент Группы 493

сидоров антон дмитриевич

Проверила Фомин А. В.

Оценка \_\_\_\_\_\_\_\_

Санкт-Петербург 2022

**Содержание**

[1. Цели работы 3](#_Toc116595675)

[2. Макеты экранов приложения и их описание 3](#_Toc116595676)

[3. Программный код 4](#_Toc116595677)

[4. Демонстрация работы приложения 4](#_Toc116595678)

[5. Вывод 10](#_Toc116595679)

[Приложение 11](#_Toc116595680)

# Цели работы

Разработать приложение для обработки изображения несколькими потоками.

# Макеты экранов приложения и их описание

* 1. Примечания по макетам

Окна в работающем приложения могут незначительно отличаться от их макетов.

* 1. Сам макет

В приложении присутствует единственное окно, макет которого представлен на рисунке 1.

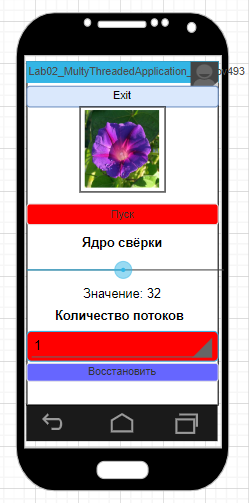


Рисунок 1 – Макет начального окна

На этом окне задаётся:

* Значение ядра свёртки (ползунок)
* Количество потоков для обработки изображения (Выпадающий список)

Под логотипом имеется кнопка *Exit*, производящая выход из приложения.

Также на этом окне есть 2 кнопки:

* *Пуск* – Запуск обработки изображения
* *Восстановить* – Возвращение изначального

# Программный код

Программный код представлен в приложении 1.

# Демонстрация работы приложения

Данное приложение предназначено для обмена сообщениями между двумя устройствами, поэтому будут использоваться 2 телефона:

1. *BQ-5731L\_08* с операционной системой *Android 9* и IP-адресом *192.168.0.9*.
2. *Redmi 4A* версии *MIUI Global 10.2.3 Стабильная 10.2.3.0 (NCCMIXM)* с операционной системой *Android 7.1.2 N2G47H*.
   1. Демонстрация работы приложения
      1. Телефон 1

*Ядро свёртки* – размерность квадратной матрицы (например ядро свёртки, равное 5 – матрица 5x5).

Изображение, обработанное матрицей с ядром 3 и одним потоком, показано на рисунке 2.

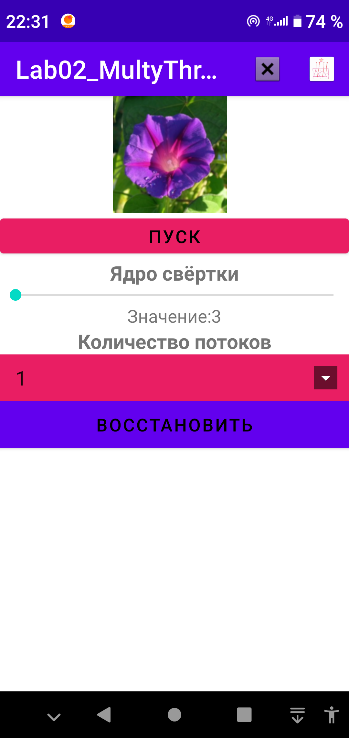


Рисунок 2 – Работа приложения.

Изображение, обработанное матрицей с ядром 10 и одним потоком, показано на рисунке 3.

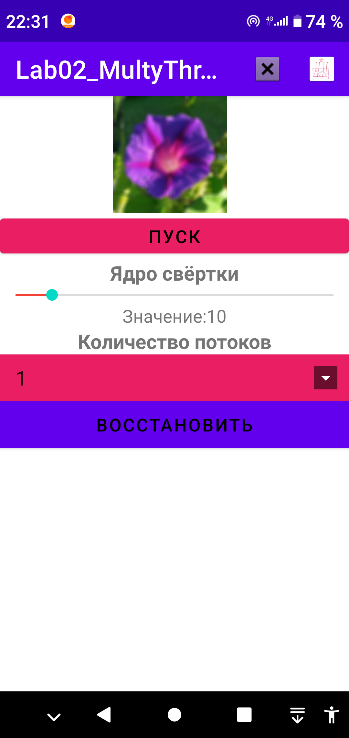


Рисунок 3 – Работа приложения.

Изображение, обработанное матрицей с ядром 10 и 8 потоками, показано на рисунке 4.

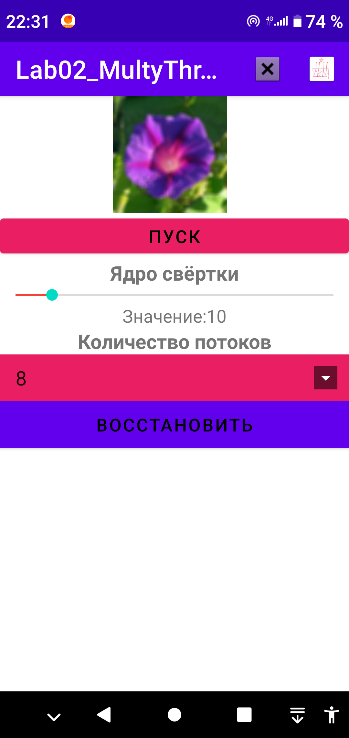


Рисунок 4 – Работа приложения.

Изображение, обработанное матрицей с ядром 16 и 8 потоками, показано на рисунке 5.

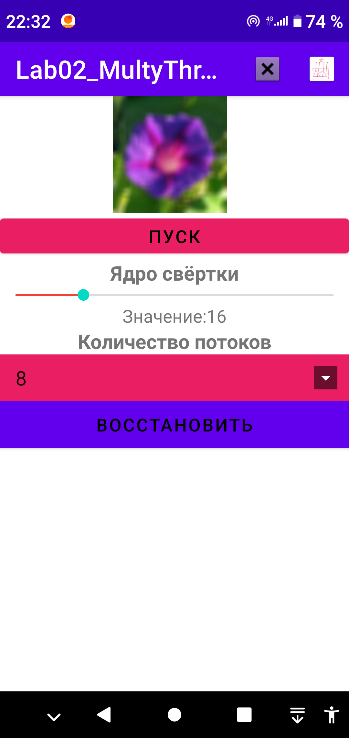


Рисунок 5 – Работа приложения.

Исходное/Восстановленное изображение показано на рисунке 6.

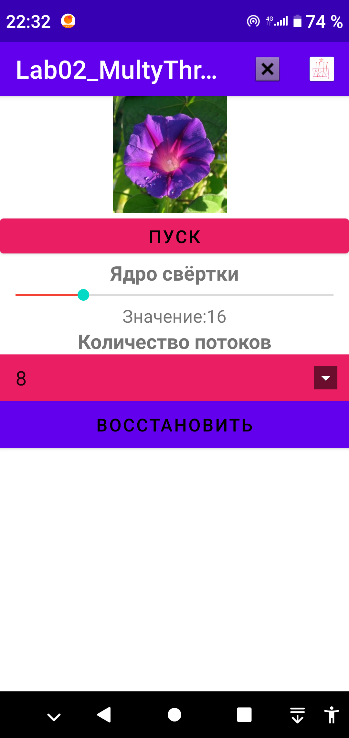


Рисунок 6 – Работа приложения.

* + 1. Телефон 2

Исходное изображение показано на рисунке 7.

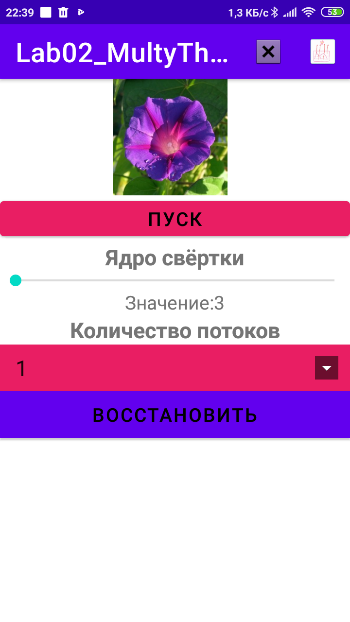


Рисунок 7 – Работа приложения.

Изображение, обработанное матрицей с ядром 12 и 5 потоками, показано на рисунке 8.

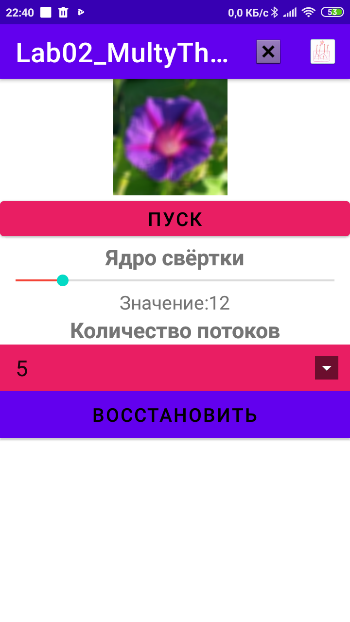


Рисунок 8 – Работа приложения.

* 1. Демонстрация Профайлера в Android Studio

Для более наглядной демонстрации было выбрано ядро свёртки 33, как показано на рисунке 9.

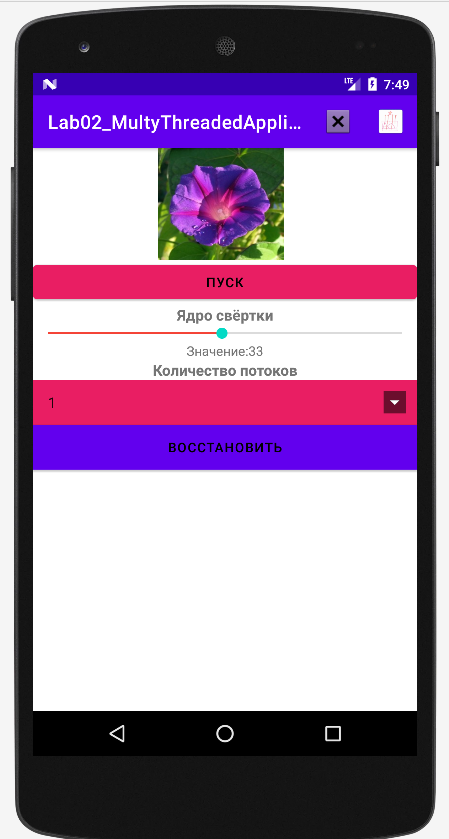


Рисунок 9 – Подготовка к тестированию

Выполнение такой обработки изображения одним потоком показано на рисунке 10.

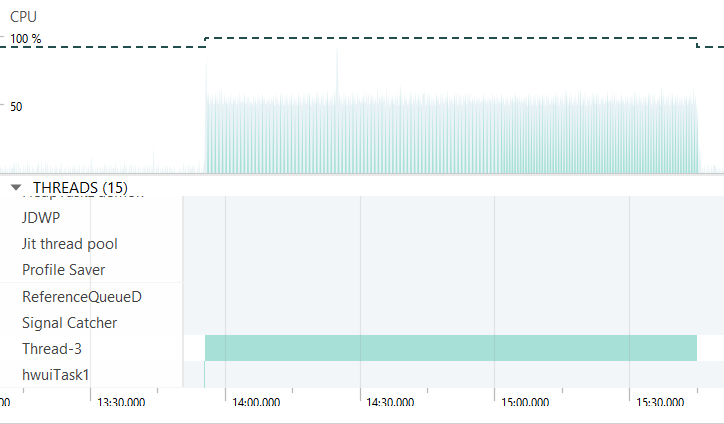


Рисунок 10 – Profiler

Выполнение такой обработки изображения пятью потоками показано на рисунке 11.

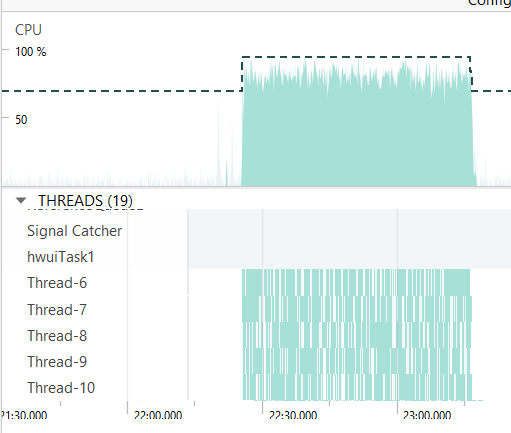


Рисунок 11 – Profiler

Выполнение такой обработки изображения восьмью потоками показано на рисунке 12.

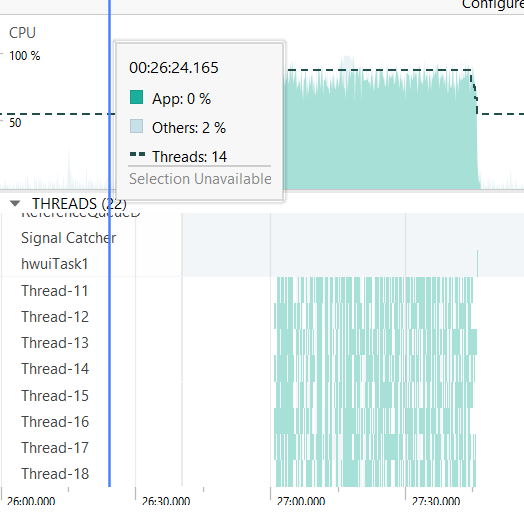


Рисунок 12 – Profiler

# Вывод

Освоено создание многопоточного приложения.

# Приложение

**Приложение 1. Программный код**

**ackage** com.example.lab02\_multythreadedapplication\_sidorov493;  
  
**import** androidx.appcompat.app.AlertDialog;  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.DialogInterface;  
**import** android.content.res.AssetManager;  
**import** android.graphics.Bitmap;  
**import** android.graphics.BitmapFactory;  
**import** android.graphics.Color;  
**import** android.graphics.Paint;  
**import** android.graphics.Picture;  
**import** android.graphics.drawable.BitmapDrawable;  
**import** android.graphics.drawable.Drawable;  
**import** android.os.Bundle;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.Button;  
**import** android.widget.ImageView;  
**import** android.widget.SeekBar;  
**import** android.widget.Spinner;  
**import** android.widget.TextView;  
  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.util.ArrayList;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 ImageView **image**;  
 SeekBar **ConvolutionKernel**;  
 TextView **size**;  
 Spinner **threads**;  
 Drawable **draw**;  
 Bitmap **bit**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **ConvolutionKernel** = findViewById(R.id.***SizeChange***);  
 **size** = findViewById(R.id.***Sizetext***);  
 **ConvolutionKernel**.setOnSeekBarChangeListener(**new** SeekBar.OnSeekBarChangeListener (){  
 @Override  
 **public void** onProgressChanged(SeekBar seekBar, **int** progress, **boolean** fromUser) {  
 **size**.setText(String.*valueOf*(progress+3));  
 }  
  
 @Override  
 **public void** onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 **public void** onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
  
  
 } );  
 **threads** = findViewById(R.id.***ThreadCount***);  
 ArrayList<String> count = **new** ArrayList<>();  
 **for**(**int** i = 1; i<=8; i++)  
 {  
 count.add(String.*valueOf*(i));  
 }  
 ArrayAdapter<String> adapter=**new** ArrayAdapter<>(**this**, android.R.layout.***simple\_list\_item\_1***, count);  
 **threads**.setAdapter(adapter);  
  
  
 **image** = findViewById(R.id.***ImagePanel***);  
 ImageView imageView = **image**;  
 String filename = **"Hypomeya.jpg"**;  
  
  
 AssetManager asset = getAssets();  
 InputStream stream = **null**;  
 **try** {  
 stream = asset.open(filename);  
 }  
 **catch** (IOException e){}  
  
 Bitmap bitmap = BitmapFactory.*decodeStream*(stream);  
 BitmapDrawable drawable = **new** BitmapDrawable(bitmap);  
  
 **try**(InputStream inputStream = getApplicationContext().getAssets().open(filename)){  
 **draw** = drawable;  
 **bit** = drawable.getBitmap();  
  
 imageView.setImageDrawable(**draw**);  
 imageView.setScaleType(ImageView.ScaleType.***FIT\_XY***);  
 }  
 **catch** (IOException e){  
 e.printStackTrace();  
 }  
 }  
  
  
 **public void** Exit\_Click(View v)  
 {  
 AlertDialog.Builder bld = **new** AlertDialog.Builder(**this**);  
  
 bld.setPositiveButton(**"Нет"**,  
 **new** DialogInterface.OnClickListener()  
 {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 dialog.cancel(); *// Закрываем диалоговое окно* }  
 });  
 bld.setNegativeButton(**"Да"**, **new** DialogInterface.OnClickListener(){  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 finish(); *// Закрываем Activity* }  
 });  
 AlertDialog dlg = bld.create();  
 dlg.setTitle(**"Выход из приложения"**);  
 dlg.setMessage(**"Уважаемый пользователь \n"** +  
 **"Вы действительно хотите выйти из программы \n"** +  
 **"Вы, также, можете запустить программу снова \n"** +  
 **"С уважением и любовью, Создатель программы, Сидоров Антон Дмитриевич"**);  
 dlg.show();  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu)  
 {  
 getMenuInflater().inflate(R.menu.***menu***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item)  
 {  
 **int** id = item.getItemId();  
 **switch** (id)  
 {  
 **case** R.id.***Close***: {  
 View v = **new** Button(**this**);  
 Exit\_Click(v);  
 }  
 **break**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
 **public void** Run\_onClick(View v)  
 {  
 **int** count = **threads**.getSelectedItemPosition();  
 count++;  
  
 Runnable[] runs = **new** Runnable[count];  
 **int** size = **ConvolutionKernel**.getProgress() + 3;  
 **int**[][] convolution = **new int**[size][];  
 **for**(**int** i = 0; i < convolution.**length**; i++)  
 {  
 convolution[i] = **new int**[size];  
 **for**(**int** j = 0; j<convolution[i].**length**; j++)  
 {  
 convolution[i][j] = 1;  
 }  
 }  
  
 **int** centre = size/2 + 1;  
 **final int**[] line = {size - centre};  
 **int** ModLine = (size % 2 == 0)? centre -1 : line[0];  
 Bitmap result = Bitmap.*createBitmap*(**this**.**bit**.getWidth(), **this**.**bit**.getHeight(), Bitmap.Config.***ARGB\_8888***);  
 Bitmap bit = Bitmap.*createBitmap*(**this**.**bit**, 0, 0, **this**.**bit**.getWidth(), **this**.**bit**.getHeight());  
  
 **int** width = bit.getWidth();  
 **int** height = bit.getHeight();  
 **int** lineThread = width/count;  
 **int**[] h = **new int**[count];  
 **for**(**int** i = 0; i< count; i++)  
 {  
 **if**(i == 0)  
 h[i] = 0;  
 **else** h[i] = h[i-1] + lineThread;  
 }  
 **if**(count > 1)  
 {  
 **if**(width % count != 0)  
 {  
 **if**(lineThread \* count < width) {  
 lineThread++;  
 }  
 }  
 }  
  
 **for**(**int** i = 0; i < count; i++) {  
 **int** xh = h[i];  
 **int** finalLineThread = lineThread;  
 runs[i] = **new** Runnable() {  
 @Override  
 **public void** run() {  
 **int** w = width;  
 **int** w1 = w - 1;  
 **int** h = height;  
 **int** h1 = h - 1;  
 **for** (**int** x = xh; x < xh + finalLineThread; x++) {  
 **if** (x > w1)  
 **break**;  
 **int** x0 = x - ModLine;  
 **for** (**int** y = 0; y < h; y++) {  
 **if** (y > h1)  
 **break**;  
 **int** y0 = y - ModLine;  
 **int** red = 0, green = 0, blue = 0;  
 **for** (**int** k = 0; k < size; k++) {  
 **int** px = x0 + k;  
 **if** (px < 0)  
 px = 0;  
 **else if** (px > w1)  
 px = w1;  
 **for** (**int** l = 0; l < size; l++) {  
 **int** py = y0 + l;  
 **if** (py < 0)  
 py = 0;  
 **else if** (py > h1)  
 py = h1;  
 **int** color = bit.getPixel(px, py);  
 red += Color.*red*(color);  
 green += Color.*green*(color);  
 blue += Color.*blue*(color);  
 }  
 }  
 **int** size1 = size \* size;  
 red /= size1;  
 green /= size1;  
 blue /= size1;  
 **int** color = Color.*argb*(255, red, green, blue);  
 result.setPixel(x, y, color);  
  
 }  
 }  
 }  
 };  
 }  
  
  
 Thread[] run = **new** Thread[runs.**length**];  
 **for**(**int** j = 0; j < runs.**length**; j++)  
 {  
 run[j] = **new** Thread(runs[j]);  
 run[j].start();  
  
 }  
  
 **for**(**int** j = 0; j < run.**length**; j++)  
 {  
 **try** {  
 run[j].join();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **image**.setImageBitmap(result);  
 }  
  
 **public void** Restore (View v)  
 {  
 **image**.setImageBitmap(**bit**);  
 }  
  
}