

ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

ассистент

должность, уч. степень, звание

подпись, дата

Н. И. Чулочникова

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 7

ИНТЕРАКТИВНАЯ КАРТА. ОПТИМИЗАЦИЯ ПЕРЕБОРА МАССИВА

по курсу:

КРОССПЛАТФОРМЕННОЕ ПРОГРАММИРОВАНИЕ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №

4326

подпись, дата

Г. С. Томчук

инициалы, фамилия

Санкт-Петербург 2025

1 Цель работы

Цель работы: создание мобильного приложения на Kotlin для ОС Android с использованием API интерактивной карты и практикой оптимизации перебора массива.

2 Задание

Вариант работы: № 17 (Москва).

1. «Интерактивная карта». Разработайте приложение с интерактивной картой, используя Google Maps API или Яндекс.Карты API. Добавьте возможность маркировки мест на карте, а также отображение информации о выбранном месте. Карта разрабатывается для города в соответствии городом по варианту.
2. Оптимизируйте перебор массива используя встроенные средства языка программирования Kotlin и протестируйте на массивах от 100000 элементов. Выведите в Activity исходный, итоговый и промежуточные массивы.

3 Листинг программы

Листинг основного кода приложения представлен в Приложении А.

4 Результаты работы программы

На рисунках 1–6 изображены результаты тестирования программы.



ИНТЕРАКТИВНАЯ КАРТА (МОСКВА)

ОБРАБОТКА МАССИВА (150K)



Рисунок 1 — Главный экран

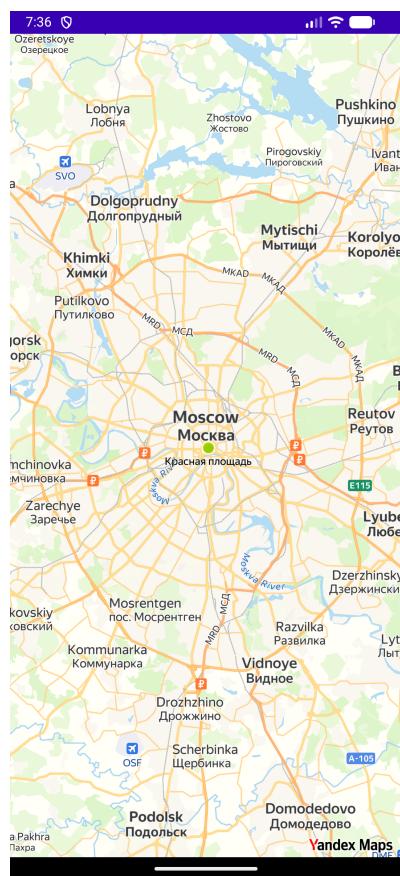


Рисунок 2 — Экран с интерактивной картой

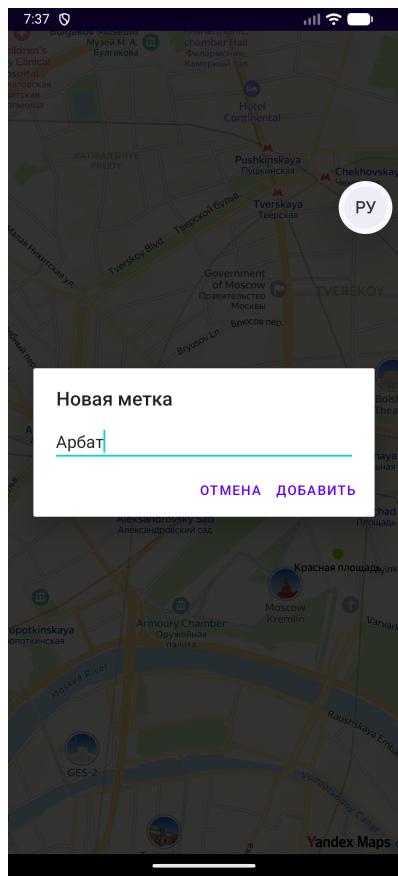


Рисунок 3 — Диалог при добавлении новой метки

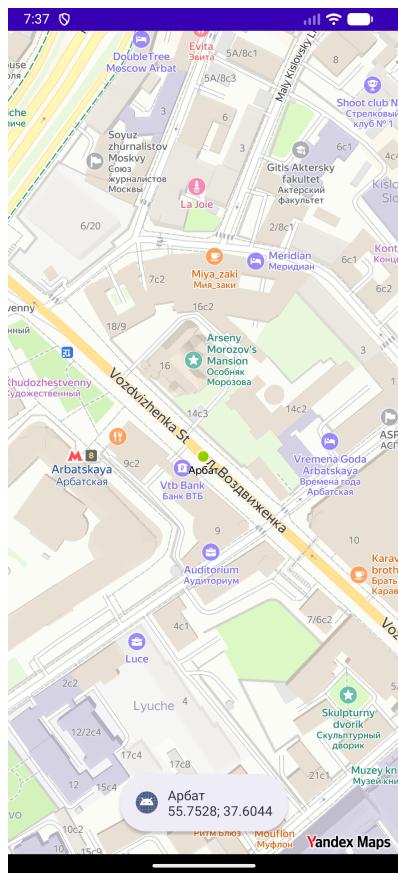


Рисунок 4 — Новая метка на карте

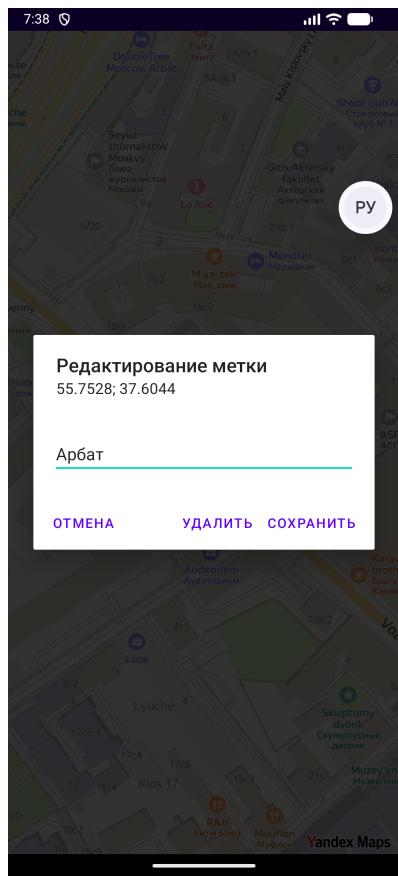


Рисунок 5 — Диалог редактирования метки

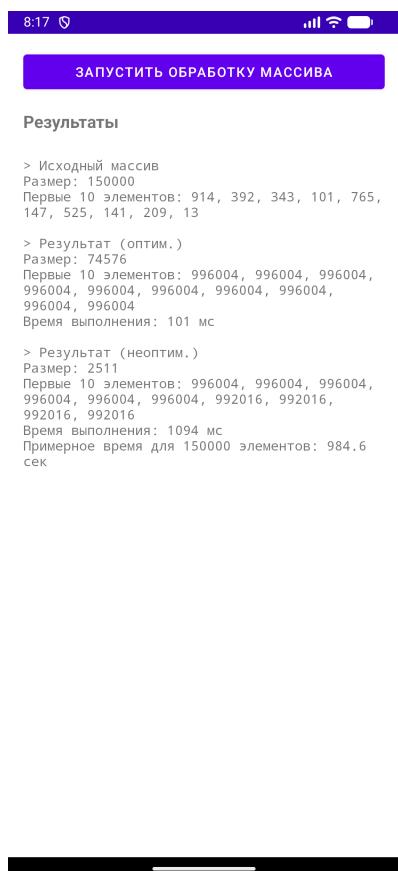


Рисунок 6 — Экран с результатами обработки массива двумя способами

5 Выводы

В ходе выполнения лабораторной работы было разработано Android-приложение с интерактивной картой Москвы с использованием SDK Яндекс.Карт (MapKit). Освоены основные принципы интеграции картографических сервисов в мобильные приложения, а также работа с картой в режиме реального времени.

В приложении реализована интерактивная работа с картой: добавление меток по одиночному нажатию, редактирование меток по долгому нажатию, удаление меток и отображение информации о выбранной метке. Это позволило изучить обработку различных пользовательских жестов, работу с объектами карты и передачу данных между элементами интерфейса.

Во второй части работы была проведена оптимизация перебора массивов с использованием встроенных средств языка Kotlin. Выполнено тестирование операций filter, map и sortedDescending на массиве размером 150.000 элементов с замером времени выполнения. Для сравнения были реализованы аналогичные операции с использованием обычных циклов и сортировки пузырьком.

Результаты тестирования показали, что стандартные функции коллекций Kotlin обладают значительно лучшей производительностью и читаемостью кода по сравнению с ручными переборами и неэффективными алгоритмами сортировки. Особенно заметно преимущество при работе с большими объёмами данных, где использование неоптимальных алгоритмов приводит к экспоненциальному увеличению времени выполнения.

В ходе лабораторной работы были закреплены навыки разработки Android-приложений с использованием сторонних SDK, работы с интерактивными картами, а также получен практический опыт анализа и оптимизации производительности кода на языке Kotlin.

ПРИЛОЖЕНИЕ А

Листинг 1 — MainActivity.kt

```
package com.grigorijtomczuk.mapuser

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.grigorijtomczuk.mapuser.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.btnMap.setOnClickListener {
            startActivity(Intent(this, MapActivity::class.java))
        }

        binding.btnArray.setOnClickListener {
            startActivity(Intent(this, ArrayActivity::class.java))
        }
    }
}
```

Листинг 2 — MapActivity.kt

```
package com.grigorijtomczuk.mapuser

import android.app.AlertDialog
import android.os.Bundle
import android.widget.EditText
import android.widget.LinearLayout
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.yandex.mapkit.Animation
import com.yandex.mapkit.MapKitFactory
import com.yandex.mapkit.geometry.Point
import com.yandex.mapkit.map.CameraPosition
import com.yandex.mapkit.map.InputListener
import com.yandex.mapkit.map.Map
import com.yandex.mapkit.map.MapObjectTapListener
import com.yandex.mapkit.map.PlacemarkMapObject
import com.yandex.mapkit.map.TextStyle
import com.yandex.mapkit.mapview.MapView
import com.yandex.runtime.image.ImageProvider

data class MarkerData(var title: String, val snippet: String)

class MapActivity : AppCompatActivity() {

    private lateinit var mapView: MapView
    private val markers = mutableListOf<PlacemarkMapObject>()

    private val inputListener = object : InputListener {
        override fun onMapTap(map: Map, point: Point) {
```

```

        showAddMarkerDialog(point)
    }

    override fun onMapLongTap(map: Map, point: Point) {
        val touchedMarker = findMarkerAt(point)
        if (touchedMarker != null) {
            val userData = touchedMarker.userData
            if (userData is MarkerData) {
                showEditMarkerDialog(touchedMarker, userData)
            }
        }
    }

private val placemarkTapListener = MapObjectTapListener { mapObject, point ->
    val userData = mapObject.userData
    if (userData is MarkerData) {
        Toast.makeText(
            this@MapActivity,
            "${userData.title}\n${userData.snippet}",
            Toast.LENGTH_SHORT
        ).show()
    }
    true
}

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_map)

    mapView = findViewById(R.id.mapview)

    // Move to Moscow
    mapView.map.move(
        CameraPosition(Point(55.7558, 37.6173), 10.0f, 0.0f, 0.0f),
        Animation(Animation.Type.SMOOTH, 1f),
        null
    )

    // Add initial marker (Red Square)
    val redSquarePoint = Point(55.7539, 37.6208)
    addMarker(redSquarePoint, "Красная площадь")

    mapView.map.addInputListener(inputListener)
}

private fun addMarker(point: Point, title: String) {
    val placemark = mapView.map.mapObjects.addPlacemark(point)
    val snippet = "${String.format("%.4f", point.latitude)}; ${
        String.format(
            "%.4f",
            point.longitude
        )
    }"

    // Using a standard system drawable for the marker
    placemark.setIcon(ImageProvider.fromResource(this,
        android.R.drawable.presence_online))
    placemark.userData = MarkerData(title, snippet)
    placemark.setText(title, TextStyle().apply { placement =
        TextStyle.Placement.BOTTOM })
    placemark.addTapListener(placemarkTapListener)
}

```

```

        markers.add(placemark)
    }

private fun findMarkerAt(point: Point): PlacemarkMapObject? {
    // Convert map point to screen point
    val mapWindow = mapView.mapWindow ?: return null
    val touchScreenPoint = mapWindow.worldToScreen(point) ?: return null

    // Threshold in pixels (approx 50dp)
    val threshold = 50 * resources.displayMetrics.density

    var closestMarker: PlacemarkMapObject? = null
    var minDistance = Double.MAX_VALUE

    for (marker in markers) {
        val markerPoint = marker.geometry
        val markerScreenPoint = mapWindow.worldToScreen(markerPoint) ?: continue

        val dx = touchScreenPoint.x - markerScreenPoint.x
        val dy = touchScreenPoint.y - markerScreenPoint.y
        val distance = Math.hypot(dx.toDouble(), dy.toDouble())

        if (distance <= threshold && distance < minDistance) {
            minDistance = distance
            closestMarker = marker
        }
    }

    return closestMarker
}

private fun showAddMarkerDialog(point: Point) {
    val builder = AlertDialog.Builder(this)
    builder.setTitle("Новая метка")

    val input = EditText(this)
    input.hint = "Введите название"

    val container = LinearLayout(this)
    container.orientation = LinearLayout.VERTICAL
    val params = LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT
    )
    val margin = (20 * resources.displayMetrics.density).toInt()
    params.setMargins(margin, margin / 2, margin, 0)
    input.setLayoutParams(params)
    container.addView(input)

    builder.setView(container)

    builder.setPositiveButton("Добавить") { _, _ ->
        val title = input.text.toString().ifBlank { "Новая метка" }
        addMarker(point, title)
    }
    builder.setNegativeButton("Отмена") { dialog, _ -> dialog.cancel() }

    builder.show()
}

private fun showEditMarkerDialog(mapObject: PlacemarkMapObject, data: MarkerData)

```

```

{
    val builder = AlertDialog.Builder(this)
    builder.setTitle("Редактирование метки")
    builder.setMessage(data.snippet)

    val input = EditText(this)
    input.setText(data.title)

    val container = LinearLayout(this)
    container.orientation = LinearLayout.VERTICAL
    val params = LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT
    )
    val margin = (20 * resources.displayMetrics.density).toInt()
    params.setMargins(margin, margin / 2, margin, 0)
    input.setLayoutParams(params)
    container.addView(input)

    builder.setView(container)

    builder.setPositiveButton("Сохранить") { _, _ ->
        val newTitle = input.text.toString()
        if (newTitle.isNotBlank()) {
            data.title = newTitle
            mapObject.setText(newTitle)
            Toast.makeText(this, "Метка обновлена", Toast.LENGTH_SHORT).show()
        }
    }

    builder.setNegativeButton("Удалить") { _, _ ->
        mapView.map.mapObjects.remove(mapObject)
        markers.remove(mapObject)
        Toast.makeText(this, "Метка удалена", Toast.LENGTH_SHORT).show()
    }

    builder.setNeutralButton("Отмена") { dialog, _ -> dialog.cancel() }

    builder.show()
}

override fun onStop() {
    mapView.onStop()
    MapKitFactory.getInstance().onStop()
    super.onStop()
}

override fun onStart() {
    super.onStart()
    MapKitFactory.getInstance().onStart()
    mapView.onStart()
}
}

```

Листинг 3 — ArrayActivity.kt

```

package com.grigorijtomczuk.mapuser

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.grigorijtomczuk.mapuser.databinding.ActivityArrayBinding
import kotlinx.coroutines.CoroutineScope

```

```

import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import kotlin.math.pow
import kotlin.random.Random
import kotlin.system.measureTimeMillis

class ArrayActivity : AppCompatActivity() {

    private lateinit var binding: ActivityArrayBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityArrayBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.btnProcessArray.setOnClickListener {
            processArray()
        }
    }

    private fun processArray() {
        binding.tvResults.text = "Вычисление..."
        binding.btnProcessArray.isEnabled = false

        CoroutineScope(Dispatchers.Default).launch {
            val sb = StringBuilder()
            val size = 150_000

            // Генерация массива
            val originalArray = IntArray(size) { Random.nextInt(0, 1000) }

            appendInfo(sb, "Исходный массив", originalArray)

            var sortedListOptimized: List<Int> = emptyList()
            val timeOptimized = measureTimeMillis {
                val filteredList = originalArray.filter { it % 2 == 0 }
                val mappedList = filteredList.map { it * it }
                sortedListOptimized = mappedList.sortedDescending() // O(N * log N)
            }

            appendInfo(sb, "Результат (оптим.)", sortedListOptimized)
            sb.append("Время выполнения: $timeOptimized мс\n")

            // Используем уменьшенный массив, иначе Bubble Sort на 150к зависнет
            val unoptimizedSize = 5_000

            val smallArray = originalArray.copyOfRange(0, unoptimizedSize)
            var sortedListUnoptimized = ArrayList<Int>()

            val timeUnoptimized = measureTimeMillis {
                val filtered = ArrayList<Int>()
                for (i in smallArray) {
                    if (i % 2 == 0) filtered.add(i)
                }

                val mapped = ArrayList<Int>()
                for (i in filtered) {
                    mapped.add(i * i)
                }
            }

            // bubble sort O(N^2)
        }
    }
}

```

```

        for (i in 0 until mapped.size - 1) {
            for (j in 0 until mapped.size - i - 1) {
                if (mapped[j] < mapped[j + 1]) {
                    val temp = mapped[j]
                    mapped[j] = mapped[j + 1]
                    mapped[j + 1] = temp
                }
            }
        }
        sortedListUnoptimized = mapped
    }

    appendInfo(sb, "Результат (неоптим.)", sortedListUnoptimized)
    sb.append("Время выполнения: $timeUnoptimized мс\n")

    if (unoptimizedSize < size) {
        val projectedTime =
            (timeUnoptimized * (size / unoptimizedSize).toDouble().pow(2))
        sb.append("Примерное время для $size элементов: ${projectedTime / 1000} сек\n")
    }

    withContext(Dispatchers.Main) {
        binding.tvResults.text = sb.toString()
        binding.btnAdd.isEnabled = true
    }
}

private fun appendInfo(sb: StringBuilder, label: String, list: List<Int>) {
    sb.append("\n> $label\n")
    sb.append("Размер: ${list.size}\n")
    sb.append("Первые 10 элементов: ${list.take(10).joinToString(", ")}\n")
}

private fun appendInfo(sb: StringBuilder, label: String, array: IntArray) {
    sb.append("\n> $label\n")
    sb.append("Размер: ${array.size}\n")
    sb.append("Первые 10 элементов: ${array.take(10).joinToString(", ")}\n")
}
}

```