Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Практична робота № 6

з курсу: «Розробка програмного забезпечення мобільних пристроїв»

**Виконав:**  
студент 4-го курсу,  
групи ТВ-з11

Роговський Назар Тарасович

Посилання на GitHub репозиторій: <https://github.com/dragonfir2016/PW6_TB-311_Rohovskyi_Nazar_Tarasovych>

**Перевірив:**

Недашківський О.Л.

Київ 2024/2025

Практична робота № 6

**Завдання:**

Завдання 1

Створіть мобільний калькулятор для розрахунку електричних навантажень об’єктів з використанням методу впорядкованих діаграм.

Цехова мережа складається з трьох типових цехів які під’єднується до трьох різних розподільчих шин (ШР1-ШР3) та кількох крупних електроприймачів (ЕП). Для спрощення приймемо що склад, номенклатура і характеристики ЕП всіх трьох цехів однакові. На основі складу ЕП та їх характеристик необхідно розрахувати силове навантаження цехової мережі.

**Хід виконання:**

Для виконання розразунків були використані наступні формули:  
Розрахунковий струм ЕП1:

A mathematical equation with black text

Description automatically generated  
Груповий коефіцієнт використання:

A mathematical equation with black letters

Description automatically generated

Ефективну кількість ЕП:

A mathematical equation with black letters

Description automatically generated

Розрахункове активне навантаження:



Розрахункове реактивне навантаження:

A black text on a white background

Description automatically generated

Повна потужність:

A square root of a square

Description automatically generated

Розрахунковий груповий струм ШР1:

A mathematical equation with black letters

Description automatically generated

Коефіцієнти використання цеху в цілому:

A mathematical equation with black letters

Description automatically generated

Ефективну кількість ЕП цеху в цілому:

A mathematical equation with numbers and symbols

Description automatically generated

Розрахункове активне навантаження на шинах 0,38 кВ ТП: A black text on a white background

Description automatically generated

Розрахункове реактивне навантаження на шинах 0,38 кВ ТП:

A black letter on a white background

Description automatically generated

Повна потужність на шинах 0,38 кВ ТП:

A square root of a function

Description automatically generated

Розрахунковий груповий струм на шинах 0,38 кВ ТП:

A mathematical equation with black text

Description automatically generated

Також для розрахунків будуть використовуватись дані з Таблиці 6.6:

A screenshot of a computer

Description automatically generated

Завдання 1

Код реалізації завдання 1 знаходиться у файлі MainActivity.kt у класі MainActivity.kt. Спочатку визначається дата клас RowData, для збереження даних про кожен елемент у таблиці:

data class RowData(  
 val name: String,  
 var cosPhi: Double,  
 var efficiency: Double,  
 var voltage: Double,  
 var quantity: Int,  
 var power: Double,  
 var usageFactor: Double,  
 var tgPhi: Double  
)

Після йього ініціалізуються початкові дані і найменування для елементів у таблиці:

val inputData = *mutableListOf*(  
 RowData("Шліфувальний верстат (1-4)", 0.9, 0.92, 380.0, 4, 20.0, 0.15, 1.33),  
 RowData("Свердлильний верстат (5-6)", 0.9, 0.92, 380.0, 2, 14.0, 0.12, 3.36),  
 RowData("Фугувальний верстат (9-12)", 0.9, 0.92, 380.0, 4, 42.0, 0.15, 1.33),  
 RowData("Циркулярна пила (13)", 0.9, 0.92, 380.0, 1, 36.0, 0.3, 1.52),  
 RowData("Прес (16)", 0.9, 0.92, 380.0, 1, 20.0, 0.5, 0.75),  
 RowData("Полірувальний верстат (24)", 0.9, 0.92, 380.0, 1, 40.0, 0.2, 1.0),  
 RowData("Фрезерний верстат (26-27)", 0.9, 0.92, 380.0, 2, 32.0, 0.2, 1.0),  
 RowData("Вентилятор (36)", 0.9, 0.92, 380.0, 1, 20.0, 0.65, 0.75)  
)  
  
val headerRow = TableRow(this)  
headerRow.addView(createHeaderCell("Найменування"))  
headerRow.addView(createHeaderCell("cosφ"))  
headerRow.addView(createHeaderCell("η"))  
headerRow.addView(createHeaderCell("U, В"))  
headerRow.addView(createHeaderCell("n"))  
headerRow.addView(createHeaderCell("Pn, кВт"))  
headerRow.addView(createHeaderCell("k\_e"))  
headerRow.addView(createHeaderCell("tgφ"))  
tableInput.addView(headerRow)

І для кожного значення створюється відповідна змінювана клітинка:

inputData.*forEach* **{** rd **->** val tr = TableRow(this)  
 tr.addView(createReadOnlyCell(rd.name, 200))  
 tr.addView(createEditableCell(rd.cosPhi))  
 tr.addView(createEditableCell(rd.efficiency))  
 tr.addView(createEditableCell(rd.voltage))  
 tr.addView(createEditableCell(rd.quantity))  
 tr.addView(createEditableCell(rd.power))  
 tr.addView(createEditableCell(rd.usageFactor))  
 tr.addView(createEditableCell(rd.tgPhi))  
 tableInput.addView(tr)  
**}**

Після цього якщо користувач натискає кнопку “Розрахувати”, запускається функція що проводить всі розрахунки згідно з формулами. Зчитуються дані введені користувачем:

val updatedData = *mutableListOf*<RowData>()  
var idx = 1  
inputData.*forEach* **{** old **->** val row = tableInput.getChildAt(idx) as TableRow  
 val cosVal = (row.getChildAt(1) as EditText).*text*.toString().*toDoubleOrNull*() ?: 0.9  
 val etaVal = (row.getChildAt(2) as EditText).*text*.toString().*toDoubleOrNull*() ?: 0.92  
 val uVal = (row.getChildAt(3) as EditText).*text*.toString().*toDoubleOrNull*() ?: 380.0  
 val nVal = (row.getChildAt(4) as EditText).*text*.toString().*toIntOrNull*() ?: 1  
 val pnVal = (row.getChildAt(5) as EditText).*text*.toString().*toDoubleOrNull*() ?: 1.0  
 val keVal = (row.getChildAt(6) as EditText).*text*.toString().*toDoubleOrNull*() ?: 0.1  
 val tgVal = (row.getChildAt(7) as EditText).*text*.toString().*toDoubleOrNull*() ?: 1.0  
 updatedData.add(  
 old.copy(  
 cosPhi = cosVal,  
 efficiency = etaVal,  
 voltage = uVal,  
 quantity = nVal,  
 power = pnVal,  
 usageFactor = keVal,  
 tgPhi = tgVal  
 )  
 )  
 idx++  
**}**

Ініціалізуються назви для колонок у таблиці з результатами:

val header1 = TableRow(this)  
header1.addView(createHeaderCell("Найменування", 190))  
header1.addView(createHeaderCell("n·Pn"))  
header1.addView(createHeaderCell("n·Pn·kₑ"))  
header1.addView(createHeaderCell("n·Pn·kₑ·tgφ"))  
header1.addView(createHeaderCell("n (Pn)²"))  
header1.addView(createHeaderCell("Ip"))

Після цього для кожеого елементу в таблиці розраховуються його показники:

for (ep in updatedData) {  
 val nPn = ep.quantity \* ep.power  
 val nPnKe = nPn \* ep.usageFactor  
 val nPnKeTg = nPnKe \* ep.tgPhi  
 val nPn2 = ep.quantity \* ep.power.*pow*(2)  
 val ip = if (ep.voltage > 0 && ep.cosPhi > 0 && ep.efficiency > 0) {  
 nPn / (*sqrt*(3.0) \* ep.voltage \* ep.cosPhi \* ep.efficiency)  
 } else 0.0  
 sumNpn += nPn  
 sumNpnKe += nPnKe  
 sumNpnKeTg += nPnKeTg  
 sumNpn2 += nPn2  
 val trow = TableRow(this)  
 trow.addView(createTextView(ep.name, 190))  
 trow.addView(createTextView(formatResult(nPn)))  
 trow.addView(createTextView(formatResult(nPnKe)))  
 trow.addView(createTextView(formatResult(nPnKeTg)))  
 trow.addView(createTextView(formatResult(nPn2)))  
 trow.addView(createTextView(formatResult(ip)))  
 tableOutput1.addView(trow)  
}

Також розраховуються показники для ШР1, ШР2, ШР3 і цеху в цілому:

val groupKb = if (sumNpn > 1e-9) sumNpnKe / sumNpn else 0.0  
val groupNe = if (sumNpn2 > 1e-9) sumNpn.*pow*(2) / sumNpn2 else 0.0  
val groupKp = kpGroup  
val groupPp = groupKp \* groupKb \* sumNpn  
val groupQp = sumNpnKeTg  
val groupSp = *sqrt*(groupPp.*pow*(2) + groupQp.*pow*(2))  
val groupIp = if (uGroupKv > 1e-9) groupPp / uGroupKv else 0.0

Пілся цього всі розраховані показники повертаються користувачу у 2 таблицях:

val header2 = TableRow(this)  
 header2.addView(createHeaderCell("Показник", 340))  
 header2.addView(createHeaderCell("Значення"))  
 tableOutput2.addView(header2)  
  
 val groupItems = *listOf*(  
 "Груповий коефіцієнт використання для ШР1=ШР2=ШР3" *to* formatResult(groupKb),  
 "Ефективна кількість ЕП для ШР1=ШР2=ШР3" *to* formatResult(groupNe),  
 "Розрахунковий коефіцієнт активної потужності для ШР1=ШР2=ШР3" *to* formatResult(groupKp),  
 "Розрахункове активне навантаження для ШР1=ШР2=ШР3" *to* formatResult(groupPp),  
 "Розрахункове реактивне навантаження для ШР1=ШР2=ШР3" *to* formatResult(groupQp),  
 "Повна потужність для ШР1=ШР2=ШР3" *to* formatResult(groupSp),  
 "Розрахунковий груповий струм для ШР1=ШР2=ШР3" *to* formatResult(groupIp)  
 )  
 groupItems.*forEach* **{** (label, valstr) **->** val row = TableRow(this)  
 row.addView(createTextView(label, 340))  
 row.addView(createTextView(valstr))  
 tableOutput2.addView(row)  
 **}** val workshopItems = *listOf*(  
 "Коефіцієнти використання цеху в цілому" *to* formatResult(workshopKb),  
 "Ефективна кількість ЕП цеху в цілому" *to* formatResult(workshopNe),  
 "Розрахунковий коефіцієнт активної потужності цеху в цілому" *to* formatResult(workshopKp),  
 "Розрахункове активне навантаження на шинах 0,38 кВ ТП" *to* formatResult(workshopPp),  
 "Розрахункове реактивне навантаження на шинах 0,38 кВ ТП" *to* formatResult(workshopQp),  
 "Повна потужність на шинах 0,38 кВ ТП" *to* formatResult(workshopSp),  
 "Розрахунковий груповий струм на шинах 0,38 кВ ТП" *to* formatResult(workshopIp)  
 )  
 workshopItems.*forEach* **{** (label, valstr) **->** val row = TableRow(this)  
 row.addView(createTextView(label, 340))  
 row.addView(createTextView(valstr))  
 tableOutput2.addView(row)  
 **}  
 }**}

Також у коді ініціалізуються функції для полегшення створення UI:

private fun createHeaderCell(value: String, width: Int = TableRow.LayoutParams.*WRAP\_CONTENT*): TextView {  
 val tv = TextView(this)  
 tv.*text* = value  
 tv.setPadding(8,8,8,8)  
 tv.*textSize* = 14f  
 tv.*maxWidth* = width  
 tv.setBackgroundColor(*resources*.getColor(R.color.*green*))  
 tv.setTextColor(*resources*.getColor(android.R.color.*white*))  
 return tv  
}  
  
private fun createReadOnlyCell(value: String, width: Int = TableRow.LayoutParams.*WRAP\_CONTENT*): TextView {  
 val tv = TextView(this)  
 tv.*text* = value  
 tv.setPadding(8,8,8,8)  
 tv.*isSingleLine* = true  
 tv.*maxWidth* = width  
 return tv  
}  
  
private fun createEditableCell(value: Any): EditText {  
 val et = EditText(this)  
 et.setText(value.toString())  
 et.setPadding(8,8,8,8)  
 et.*backgroundTintList* = *resources*.getColorStateList(R.color.*green*)  
 return et  
}

Сам зовнішній вигляд програми налаштовується у файлі “activity\_main.xml”, який відповідає за UI для Завдання 1 і виглядає наступним чином:

A screenshot of a computer screen

Description automatically generated

При запуску програми користувачу показується таблиця з вже презаповненими даними згідно Таблиці 6.6:

A screenshot of a computer

Description automatically generated

При запуску контрольного прикладу виходять наступні результати:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Та при запуску варіанту 7 з таблиці з варіантами наступні результати:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Висновок**

В результаті виконання практичної роботи №6 було створено мобільний калькулятор для розрахунку електричних навантажень об’єктів з використанням методу впорядкованих діаграм. Було реалізовано арифметичні операції для визначення активного, реактивного й повного навантаження, а також динамічне додавання та зчитування даних у таблиці. Я вдосконалив навички програмування мовою Kotlin та розуміння створення та використання функцій і класів. Завдяки цьому було забезпечено коректне опрацювання введених користувачем значень та автоматичне відображення результатів розрахунків.