МЕХАНІКА, МАТЕМАТИКА

Є. С. ШОХ

(Науковий керівник О. М. Костовський)

ДЕЯКІ ОСОБЛИВОСТІ СКЛАДАННЯ ПРОГРАМИ РОЗВ'ЯЗКУ ЗАДАЧ, У ЯКИХ ЧИСЛОВА ІНФОРМАЦІЯ ЗАДАЄТЬСЯ З ДОПОМОГОЮ ТАБЛИЦЬ, НА ЕЛЕКТРОННИХ ШВИДКОДІЮЧИХ ЦИФРОВИХ ОБЧИСЛЮВАЛЬНИХ МАШИНАХ

Програмування розв'язування задач з використанням таблиць, в яких ті чи інші величини, що входять в задачу, задані дискретно, має деякі специфічні труднощі, характерні для задач даного типу. Справа в тому, що з допомогою операцій, які може виконувати сучасна швидкодіюча електронна цифрова вичислювальна машина, буває досить важко вести такі математичні операції, як, наприклад, інтерполювання і ряд інших операцій, що зустрічаються в задачах даного типу. Але з допомогою деяких штучних прийомів в більшості випадків можна розв'язати поставлену задачу. І хоча іноді це приводить до збільшення часу роботи машини, розв'язування таких задач на електронних обчислювальних машинах має великий практичний інтерес.

Головною умовою, якій повинна задовольняти така машина, являється достатня ємкість оперативної пам'яті (внутрішнього накоплювача), бо якщо весь числовий матеріал не буде поміщений в оперативній пам'яті (внаслідок його малої ємкості), значно знизиться ефективність машини. Тому основною задачею програміста, що складає програму розв'язування подібної задачі, буде найбільш раціональне використання оперативної пам'яті, ємкість якої поки що у всіх електронних обчислювальних машинах є дуже невеликою.

Наприклад, у машини «Стрела» ємкість внутрішнього накоплювача (оперативної пам'яті) складає 2047 чисел, у машини «М-2» — 512 чисел.

Кожна машина має і другий запам'ятовуючий пристрій — так званий зовнішній накоплювач,— ємкість якого в залежності від типу машини коливається в межах від 5000 до 400000 чисел і більше, але цей пристрій, в порівнянні з оперативною пам'яттю, діє дуже повільно, чим значно гальмує роботу ма-

шини. Тому завжди треба вводити мінімальну кількість числового матеріалу, одержуючи, по можливості, необхідні числові дані з вже раніше введеного матеріалу за допомогою арифметичних і логічних операцій. Другим фактором, що приводить до тих самих результатів, є те, що пристрій, який вводить числову інформацію в машину, також працює дуже повільно, і тому швидше одержати потрібне число в самій машині, за допомогою певних операцій над числами, що вже є в машині, ніж вводити його.

Для ілюстрації всіх цих особливостей, а також деяких загальних прийомів програмування, що зустрічаються тільки в такому класі задач, розглянемо приклад програмування трохи складної, але дуже повчальної задачі.

Ця задача є практичним інженерним розрахунком.

Вона зводиться до таких дій:

параметр t приймає значення:

0; 0,1; 0,2;...; 0,9; 1,0; 1,2; 1,4;...; 4,8; 5,0; 5,5; 6,0;...; 29,5; 30,0,

тобто при $0 \leqslant t \leqslant 1$, t змінюється через 0,1,

при $1 \leqslant t \leqslant 5$, t змінюється через 0,2,

при $5 \leqslant t \leqslant 30$, t змінюється через 0,5.

Функція $s(t)$ задається таким чином.

$$f(t) = \begin{cases} \operatorname{tg}(1{,}457 - 0{,}0327 \cdot t) & \text{при } t \leqslant 22{,}5 \\ \text{const} & \text{при } t \geqslant 22{,}5 \end{cases}$$

Функція $h(t)$ задається так:

$$h(t) = h[f(t)] = 1081 \cdot [0{,}1143 \cdot f(t)]^{0{,}8}.$$

І нарешті дане рівняння:

$$p \cdot R \frac{J_1(pR)}{J_0(pR)} = h(t) \cdot R,$$

де $R = 0{,}0185 J_1(pR)$ і $J_0(pR)$ — функції Бесселя.

Величини $p \cdot R \frac{J_1(pR)}{J_0(pR)}$, або, що те саме $h(t) \cdot R$, задані у вигляді таблиці з відповідними значеннями $p \cdot R$. Завдання полягає в тому, щоб для кожного значення $h(t)R$ при $0 \leqslant t \leqslant 30$ знайти п'ять відповідних значень $p \cdot R$ за таблицею.

Програмування вичислення $h(t) \cdot R$ проводиться за звичайними принципами і не являє особливого інтересу (на блок-схемі — блоки I-IV). Але дальші дії є в основному типовими для даного класу задач і заслуговують більш близького розгляду.

Позначимо вичислене для даного t значення $h(t) \cdot R$ через y', а відповідне йому значення $p \cdot R$ за таблицею через x'.

Тоді дальші вичислення повинні проводитись за формулою:

$$x' = x_2 - \frac{(x_2 - x_1)(y_2 - y')}{y_2 - y_1},$$

де y_2 — найближче за таблицею значення $p \cdot R \frac{J_1(p \cdot R)}{J_0(p \cdot R)} = h(t) \cdot R$ більше y', а y_1 — найближче менше y', тобто:

$$y_1 \leqslant y' \leqslant y_2,$$

$x_2 = pR$ відповідає y_2,

$x_1 = pR$ відповідає y_1.

В даній задачі $x_2 - x_1 = 0{,}05$, тобто таблиця складена з шагом аргумента 0,05. Це значно спрощує справу, оскільки не є необхідним вводити всі x, а досить ввести тільки перше значення x з даної групи таблиці.

V блок контролює, яке з табличних y стає більше y', і одночасно вичисляє $y_2 - y'$. VI блок виконує умовний перехід знову до V блоку, якщо взяте y менше y', або до VIII. VIII блок являє найбільший інтерес з усієї програми. В ньому використані елементи автоматичного програмування, тобто програма складена так, що машина сама собі віддає наказ, що робити далі.

Припустимо, що ми ведемо інтерполювання в першій групі таблиці. Значення y розташовані в ячейках з номерами від 1201 до 1213 (у вісімковій системі числення).

Нехай y, що стоїть в ячейці 1213, більше y', тобто $y' < 22{,}357 = y_2$. Як вичислити $y_2 - y_1$?

Для людини це очевидно — треба взяти ячейку 1212, подивитись, яке там стоїть число, і відняти його від y_2:

$$y_2 - y_1 = 22{,}357 - 14{,}911 = 7{,}446.$$

Але машина не може поступити аналогічно. Тут і приходять на допомогу прийоми автоматичного програмування. $y_2 - y'$ ми вичисляли командою, що стоїть в ячейці 0220. Очевидно, що в нашому випадку в ячейці 0220 залишиться стояти команда, перший адрес якої 1213 (VII блок при кожному поверненні із VI блока в V збільшував перший адрес команди, що стоїть в ячейці 0220, на одиницю). Тоді за допомогою операції 11 (логічне множення) ми виділяємо цей адрес. Одержимо число:

1213 0000 0000 0 00.

Припустимо, що ми хочемо мати значення $y_2 - y_1$ в ячейці 0513. Тоді ми запасаємося таким числом:

0000 0000 0513 0 03 (в ячейці 1012).

Тепер з допомогою операції 13 (логічне додавання), ми з цих двох чисел формуємо таке: 1213 0000 0513 0 03 (α).

В нас ще немає другого адреса. Діємо так: здвигаємо число 1213 0000 0000 0 00 вправо на 12 розрядів (двійкових) за допомогою операції 14. Одержимо: 0000 1213 0000 0 00. За допомогою операції 15 зменшуємо другий адрес на одиницю:

0000 1212 0000 0 00,

а тепер знову за допомогою 13-ої операції з (α) і одержаного числа формуємо команду:

$$\begin{array}{r} \downarrow 0000\ 1212\ 0000\ 0\ 00 \downarrow \\ 1213\ 0000\ 0513\ 0\ 03 \\ \hline 1213\ 1212\ 0513\ 0\ 03 \end{array}$$

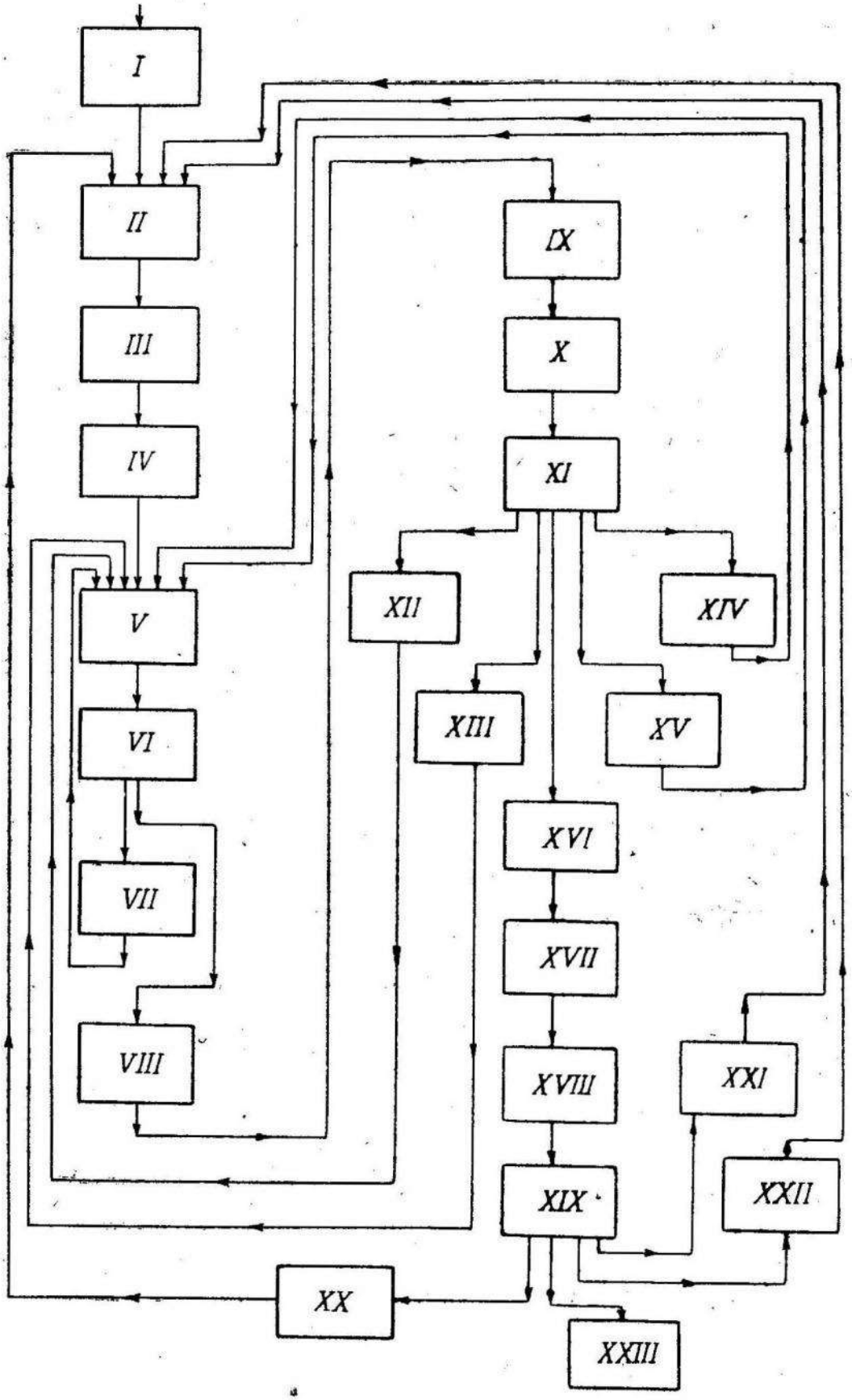
Маємо команду: із числа, що знаходиться в ячейці 1213, відняти число, що знаходиться в ячейці 1212, і результат направити в ячейку 0513. Тобто якраз те, що нам потрібно. Таким чином, не знаючи навіть, де зупиниться «пошук» в таблиці, програміст може наперед знати за допомогою відповідних прийомів, де він одержить той чи інший потрібний йому результат.

Дальші обчислення ідуть за загальними правилами і прийомами програмування і тому не являють великого інтересу. Деякі ускладнення вносить те, що в таблиці є п'ять груп: для кожного $h(t)\cdot R$ потрібно знайти п'ять відповідних $p\cdot R$, але це досягається засиланням в 0220 ячейку відповідної команди замість тієї, що стояла там раніше, всі вони знаходяться у внутрішній пам'яті в ячейках 1005, 1006, 1007, 1010.

Якщо, наприклад, замість команди, що стоїть в ячейці 0220, спочатку заслати команду з 1007 ячейки, то вичислення почнуться не з ячейки 1201, а з ячейки 1261, тобто в четвертій групі таблиці; процес вичислення буде абсолютно той же. Тому всі вичислення, що стосуються одного t, включаючи і вивід результатів, запрограмовані циклічно і займають ячейки з 0201 по 0264. Вичислення нового t, відновлення «зіпсованих» в процесі розрахунків команд і контроль кінця обчислень здійснюються командами з 0265 по 0307 (вся числова інформація, що вводиться в машину, за виключенням чисел, з якими ведуться розрахунки, представляються в вісімковій системі числення). Цікаво відмітити ще трохи незвичайне застосування операції 04 (віднімання модулів чисел), яка використана в команді 0240. Справа в тому, що тут операція 04 використана для послідовного одержання сигнала ω то рівного одиниці, то рівного нулю. Це наочно видно з прикладу: в ячейці β вмістимо нуль, а в ячейці 7400 стоїть 1.

Команда: β 7400 β0 04.

Блок-схема програми розв'язування задачі



Послідовність дії:

$$|0| - |1| = -1 \quad -1 \to (\beta)\ \omega = 1$$
$$|-1| - |1| = 0 \quad 0 \to (\beta)\ \omega = 0$$
$$|0| - |1| = -1 \quad -1 \to (\beta)\ \omega = 1$$
$$|-1| - |1| = 0 \quad 0 \to (\beta)\ \omega = 0$$

і т. д. скільки потрібно.

Кілька слів про відладку програми (перевірка правильності складання програми).

Після того, як буде введений весь числовий матеріал і програма, вводять додатково кілька команд, які змінюють програму так, що вичислення проводяться тільки для $t = 0$ і $t = 0{,}1$. Ці ж вичислення проводяться від руки. Якщо результати однакові, програма правильна.

Програмування подібних задач на машинах з трьохадресною системою кодування операцій набагато легше, ніж на машинах з одноадресною системою. Це пояснюється тим, що на одноадресних машинах прийоми автоматичного програмування складніші, ніж на трьохадресних. Однак і там, не дивлячись на великі труднощі програмування, розв'язування подібних задач, що потребують великого об'єму арифметичних і логічних операцій, являють величезну практичну вигоду. Досить сказати, що розібрана задача розв'язувалась на машині «Стрела» всього п'ять хвилин, не рахуючи вводу, який тягнувся три хвилини. Всього на розв'язування задачі витрачено вісім хвилин машинного часу, в той час, як обчислювачу, що робить одну арифметичну операцію за дві секунди, тобто з допомогою механічної лічильної машини (клавішної), на розв'язування цієї задачі було б потрібно біля місяця безперервної роботи, або біля трьох місяців при восьмигодинному робочому дні. І все ж точність вичислень була б набагато нижча, ніж при машинному способі розв'язування (див. рисунок).

I-й блок (команди 0201 — 0203) — підготовка.
II-й блок (команди 0204 — 0206) — вичислення $-1{,}457 + 0{,}0327 \cdot t + \frac{\pi}{2}$
III-й блок (команди 0207 — 0212) — вичислення $f(t)$.
IV-й блок (команди 0213 — 0217) — вичислення y'.
V-й блок (команда 0220) — вичислення $y_2 - y'$.
VI-й блок (команда 0221) — умовний перехід.
VII-й блок (команди 0222 — 0224) — переадресація. Вичислення x_2.
VIII-й блок (команди 0225 — 0232) — формування команди 0232, вичислення $y_2 - y_1$.
IX-й блок (команди 0233 — 0236) — вичислення x'.
X-й блок (команда 0237) — переадресація.
XI-й блок (команди 0240 — 0241) — умовний перехід.
XII-й блок (команди 0242 — 0244) — переадресація.
XIII-й блок (команди 0245 — 0250) — переадресація.
XIV-й блок (команди 0251 — 0254) — переадресація.
XV-й блок (команди 0255 — 0260) — переадресація.
XVI-й блок (команди 0261 — 0264) — переведення з двійкової в десяткову систему і вивід результатів.

XVII-й блок (команди 0265 — 0270) — відновлення.
XVIII-й блок (команда 0271) — підрахунок циклів.
XIX-й блок (команда 0272) — умовний перехід.
XX-й блок (команди 0273 — 0274) — вичислення нового t.
XXI-й блок (команди 0275 — 0301) — переадресація.
XXII-й блок (команди 0302 — 0306) — переадресація.
XXIII-й блок (команда 0307) — останов.

Програма розв'язування задачі

```
0201 0507 0000 0507 0 05
0202 0501 0000 0501 0 05
0203 1333 0004 0502 0 45
0204 1326 0507 0510 0 05
0205 1327 0510 0510 0 03
0206 7474 0510 0511 0 03
0207 0510 0000 0510 0 67
0210 0511 0000 0511 0 67
0211 0511 0000 0511 0 62
0212 0510 0511 0510 0 05
0213 1332 0510 0510 0 05
0214 0510 0000 0510 0 66
0215 1330 0510 0510 0 05
0216 0510 0000 0510 0 64
0217 1331 0510 0510 0 05
0220 1201 0510 0511 0 03
0221 0225 0222 0000 0 20
0222 0220 7423 0220 0 02
0223 0502 1340 0502 0 01
0224 0220 0220 0000 0 20
0225 0220 7446 0512 0 11
0226 1012 0512 0521 0 13
0227 0512 7547 0512 0 14
0230 0512 7424 0512 0 15
0231 0521 0512 0232 0 13
0232 0000 0000 0000 0 00
0233 0511 1340 0511 0 05
0234 0513 0000 0513 0 62
0235 0511 0513 0513 0 05
0236 0502 0513 0514 0 03
0237 0236 7430 0236 0 02
0240 0501 7400 0501 0 04
0241 0245 0242 0000 0 20
0242 1005 0000 0220 0 13
0243 0223 7430 0223 0 02
0244 0220 0220 0000 0 20
0245 1006 0000 0220 0 13
0246 0223 7430 0223 0 02
0247 0241 1004 0241 0 02
0250 0220 0220 0000 0 20
0251 1007 0000 0220 0 13
0252 0223 7430 0223 0 02
0253 0241 1003 0241 0 02
0254 0220 0220 0000 0 20
0255 1015 0000 0220 0 13
0256 0223 7430 0223 0 02
0257 0241 1002 0241 0 02
0260 0220 0220 0000 0 20
0261 0510 0000 0510 0 70
0262 0514 0004 0514 0 70
0263 0510 0000 0000 0 44
0264 0514 0004 0000 0 44
0265 1010 0000 0220 0 13
0266 1011 0000 0223 0 13
0267 1013 0000 0241 0 13
0270 1014 0000 0236 0 13
0271 1016 7400 1016 0 03
0272 0273 0275 0000 0 20
0273 0507 1323 0507 0 01
0274 0202 0202 0000 0 20
0275 0273 7424 0273 0 02
0276 0507 1324 0507 0 01
0277 0271 7430 0271 0 02
0300 0272 1001 0272 0 02
0301 0202 0202 0000 0 20
0302 0273 7424 0273 0 02
0303 0507 1325 0507 0 01
0304 0271 7430 0271 0 02
0305 0272 1001 0272 0 02
0306 0202 0202 0000 0 20
0307 0000 0000 0000 0 40
```

Розташування інформації в пам'яті машини:

ячейки 1201 — 1213 — перша група таблиці,
ячейки 1214 — 1235 — друга група таблиці,
ячейки 1236 — 1260 — третя група таблиці,
ячейки 1261 — 1302 — четверта група таблиці,
ячейки 1303 — 1322 — п'ята група таблиці.
1323 — 100 000 000 0 00 — 0,1
1324 — 200 000 000 0 00 — 0,2
1325 — 500 000 000 0 00 — 0,5
1326 — 327 000 000 1 01 — 0,0327

```
1327 — 145 700 000·0 01 — 1,457
1330 — 800 000 000 0 00 — 0,8
1331 — 199 985 000 0 02 — 19,9985
1332 — 114 300 000 0 00 — 0,1143
1333 — 180 000 000 0 01 — 1,8
1334 — 445 000 000 0 01 — 4,45
1335 — 740 000 000 0001 — 7,4
1336 — 104 500 000 0002 — 10,45
1337 — 135 500 000 0 02 — 13,55
1340 — 500 000·000 1 01 — 0,05
1001 — 0000 0005 0000 0 00
1002 — 0000 0010 0000 0 00
1003 — 0010 0000 0000 0 00
1004 — 0000 0007 0000 0 00
1005 — 1214 0510 0511 0 03
1006 — 1236 0510 0511 0 03
1007 — 1261 0510 0511 0 03
1010 — 1201 0510 0511 0 03
1011 — 0502 1340 0502 0 01
1012 — 0000 0000 0513 0 03
1013 — 0245 0242 0000 0 20
1014 — 0502 0513 0514 0 03
1015 — 13030510 0511 0 03
1016 — 2400 0000 0000 0 04 — 10
1017 — 2000 0000 0000 0 03 — 4
1020 — 2400 0000 0000 0 03 — 5
```

В ячейках, номери яких починаються з цифри 7, стоять спеціальні константи для логічних операцій.

ЛІТЕРАТУРА

1. Быстродействующая вычислительная машина М-2. Под редакцией И. С. Брука. Гос. издательство технико-теоретической литературы, М., 1957.
2. М. Уилкс, Д. Уиллер, С. Гилл. Составление программ для электронных счетных машин. ИЛ, М., 1953.
3. Р. К. Ричардс. Арифметические операции на цифровых вычислительных машинах. ИЛ, М., 1957.

Ю. І. КОЙФМАН

(Науковий керівник М. П. Шереметьєв)

ПРО ОДИН СПОСІБ РОЗВ'ЯЗУВАННЯ ЗАДАЧ ДЛЯ БЕЗМЕЖНОЇ ПЛАСТИНКИ З ОТВОРОМ, КРАЙ ЯКОГО ПІДКРІПЛЕНИЙ ТОНКИМ КІЛЬЦЕМ

Граничні умови для задач такого типу виведені М. П. Шереметьєвим в роботі [1], причому кільце прийняте за пружну лінію, яка працює на розтяг та згін.

Ці граничні умови мають такий вигляд:

$$\begin{aligned} &\Phi_1(t) + \overline{\Phi_1(t)} - e^{2i\alpha}\{t\,\overline{\Phi}_1'(t) + \Psi_1(t)\} = e^{i\alpha}(X_n - i\,Y_n); \\ &\varkappa\,\overline{\Phi_1(t)} - \Phi_1(t) + e^{2i\alpha}\{\overline{t}\,\Phi_1'(t) + \Psi_1(t)\} = 2\mu(\varepsilon_0 - i\Theta). \end{aligned} \tag{1}$$

§ 1. Відобразимо нашу область на площину (ζ) з круговим отвором за допомогою функції $Z = \omega(\zeta, m)$, де $\omega(\zeta, m)$ — будь-яка раціональна функція, що аналітично залежить від малого параметру m. Оскільки функції $\Phi_1 = \Phi[\omega(\zeta, m)]$, $\Psi_1 = \Psi[\omega(\zeta, m)]$ також залежать від цього параметру, то будемо шукати їх в такому вигляді: