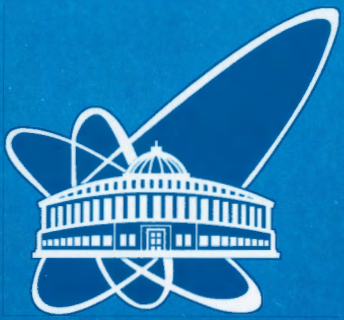

ОБЪЕДИНЕННЫЙ
ИНСТИТУТ
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ

Дубна

00-163

P11-2000-163

Э.Б.Душанов[1], М.Г.Емельяненко[2], Г.Ю.Коновалова[3]

О ФОРМАТАХ ПРЕДСТАВЛЕНИЯ ВЕЩЕСТВЕННЫХ ЧИСЕЛ И АЛГОРИТМЕ АВТОМАТИЧЕСКОГО ОПРЕДЕЛЕНИЯ КОНСТАНТ ВЕЩЕСТВЕННОЙ АРИФМЕТИКИ ЭВМ

Направлено в журнал «Программирование»

[1]Институт ядерной физики АН Республики Узбекистан
[2]Кафедра вычислительной математики и кибернетики, МГУ
[3]Международный университет природы, общества и человека «Дубна»

2000

Введение

При реализации многих алгоритмов линейной алгебры используются [1,2] значения констант вещественный арифметики ЭВМ, а также процедуры выделения мантиссы и порядка вещественного числа. Реально машинные константы зависят от архитектуры конкретной ЭВМ. Ознакомившись с соответствующей документацией по архитектуре, используемой ЭВМ, можно получить информацию об этих константах. Но это "знакомство" требует, как известно, значительных усилий как по поиску соответствующего тома описания технической и программной документации, так и "грамотного" его прочтения. Учитывая это обстоятельство, в данной работе предлагается простой алгоритм автоматического получения базовых констант машинной арифметики и, как следствие, поиска мантиссы и порядка вещественного числа в формате данной ЭВМ.

Как известно [1–3], любое вещественное число x в ЭВМ представляется в нормализованном (с плавающей точкой) виде:

$$x = \pm m \cdot \beta^e, \tag{1}$$

где $\beta(1 < \beta)$ — основание системы счисления данной ЭВМ, m — мантисса числа x. При этом $m = \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_t}{\beta^t}, \frac{1}{\beta} \le m < 1, 0 \le d_i \le \beta - 1, d_1 \ne 0, i = 1, 2, ..., t$, где t — число β-ичных разрядов, отведенных для хранения мантиссы; e — порядок числа $x(l \le e \le u)$, где $(l < 0, u > 0)$ — границы диапазона изменения порядка. Таким образом, нормализованное число x является конечным рациональным числом, приближающим заданное вещественное число $\bar{x} \in R$. При этом образом бесконечного множества вещественных чисел $\bar{x} \in R$ является рациональное число x и ошибка такого представления имеет вид

$$|x - \bar{x}| \le \begin{cases} \varepsilon_0, \text{ если } |\bar{x}| < \varepsilon_0, \\ |\bar{x}|\varepsilon_1, \text{ если } \varepsilon_0 \le |\bar{x}| \le \varepsilon_\infty, \end{cases} \tag{2}$$

где ε_∞ — максимальное число, представимое в данной ЭВМ (машинная бесконечность), ε_0 — минимальное число, представимое в данной ЭВМ (машинный нуль), ε_1 — относительная погрешность вычислений с нормализованными числами данной ЭВМ. Если $|\bar{x}| > \varepsilon_\infty$, то число $\bar{x}$ не приближается никакими машинными числами и в зависимости от принятой в данной серии ЭВМ концепции либо происходит остановка ("по переполнению"), либо $\bar{x}$ заменяется на ε_∞.

Множество чисел, представимых указанным выше (1)–(2) способом в любой ячейке памяти данной ЭВМ, не является бесконечным. В этом множестве, как известно [2], не более $2(\beta - 1)\beta^{t-1}(u - l + 1) + 1$ чисел. Таким образом, между любыми двумя соседними нормализованными числами x_1 и x_2 (1) в соответствии с аксиоматической теорией вещественных чисел [5] находится бесконечно много (континуум) вещественных чисел x.

Выполнение арифметических операций над числами с плавающей точкой приводит в свою очередь к появлению ошибок округления. Если обозначить $\otimes$ — любую из арифметических операций $(+, -, \cdot, /)$ над числами x, y вида (1), то для абсолютной погрешности результата имеет место неравенство

$$\Delta_{x \otimes y} \le \begin{cases} \varepsilon_0, \text{ если } |x \otimes y| < \varepsilon_0, \\ |x \otimes y|\varepsilon_1 + \varepsilon_0, \text{ если } \varepsilon_0 \le |x \otimes y| \le \varepsilon_\infty. \end{cases} \tag{3}$$

Можно существенно снизить [1,2,4] влияние накопления $\Delta_{x \otimes y}$ на конечный результат с помощью так называемой арифметики чисел с вынесенными порядками.

Ошибки округления (2)–(3) характеризуются, как видно, вещественными константами $\varepsilon_\infty, \varepsilon_0$ и ε_1. Эти машинные константы иногда заменяют [1,2,4] следующими четырьмя целыми константами: β, t, l и u. Значения конкретного из двух указанных наборов констант могут приводиться, как было отмечено выше, в руководствах данной вычислительной машины.

1. Форматы представления вещественных чисел в различных типах ЭВМ

В случае, когда любая константа из указанных наборов по какой-либо причине, отмеченной выше, не доступна, она может быть восстановлена пользователем. При этом существенную роль играет нормализованная форма (1) представления в ЭВМ вещественного числа. Представление чисел в виде (1) и выполнение операций над ними было реализовано [6] в 1954 г. в вычислительных машинах NORC и 704 фирмы IBM. С тех пор нормализованные числа в различных конфигурациях ЭВМ представляются по-разному в зависимости от конкретной реализации компилятора и архитектуры конкретной ЭВМ. На рис. 1 приведены форматы (*внутреннее "побитное" представление*) нормализованных вещественных чисел с двойной точностью для различных типов ЭВМ.

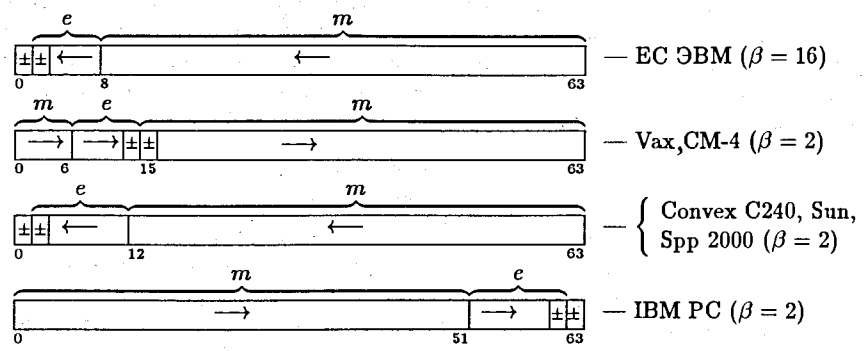

Рис. 1. Форматы представления вещественных чисел с двойной точностью различных типов ЭВМ. Стрелкой (⟶) указаны направления возрастания старших двоичных разрядов (битов) мантиссы и порядка, ± — знак мантиссы и порядка соответственно

В форматах ЭВМ различных (в том числе указанных) типов порядок e и мантисса m нормализованного числа x представляются по-разному. Это различие, как видно из приведенных выше форматов (рис. 1), а также приводимых ниже (рис. 2) двоичных представлений чисел, обусловлено следующим. В различных типах ЭВМ для хранения мантиссы m и порядка e, а также их знаков отводятся *разные* двоичные разряды ячеек памяти. При этом для двоичного представления указанных величин и их знаков используются *различные* коды. Существуют, как известно [7,8], *прямой, обратный* и *дополнительный* коды. Для упрощения иллюстрации различий прямого, обратного и дополнительного кодов рассмотрим представления в этих кодах целых чисел и вещественных чисел с фиксированной точкой.

Как известно [7,8], любое вещественное число x с *фиксированной* точкой представляется в ЭВМ в виде

$$x = \pm(d_{n-1}\beta^{n-1} + \cdots + d_0\beta^0 + d_{-1}\beta^{-1} + \cdots + d_{-\nu}\beta^{-\nu}) \equiv \pm d_{n-1}\dots d_0, d_{-1}\dots d_{-\nu}, \quad (4)$$

где $0 \le d_i \le \beta - 1$. n и ν число β–ичных разрядов, отведенных для целой части и дробной части числа x соответственно. Для представления знака ($\pm$) числа x отводится $k (k \ge 1)$ двоичных разрядов. Таким образом, в формате этого представления фиксирован β-ичный разряд содержащий d_0. Если x — целое число, то $\nu = 0$ и d_0 будет младшей значащей цифрой числа x. В случае x — вещественного $\nu \ne 0$ и d_0 будет младшей значащей цифрой целой части числа x.

Пусть x — целое или вещественное с фиксированной точкой число вида (4). *Прямым кодом* числа x называется целое $(k+n+\nu)$-разрядное число $[x]^k_{np}$, определяемое [7] формулой (4) вида

$$[x]^k_{np} = \begin{cases} \underbrace{00\dots0}_{k} d_{n-1}\dots d_1 d_0 d_{-1}\dots d_{-\nu} \text{ при } x \ge 0, \\ \underbrace{11\dots1}_{k} d_{n-1}\dots d_1 d_0 d_{-1}\dots d_{-\nu} \text{ при } x \le 0. \end{cases} \quad (5)$$

Обратным кодом числа x называется целое $(k+n+\nu)$-разрядное число $[x]^k_{обр}$, определяемое соответственно формулой

$$[x]^k_{обр} = \begin{cases} \underbrace{00\dots0}_{k} d_{n-1}\dots d_1 d_0 d_{-1}\dots d_{-\nu} \text{ при } x \ge 0, \\ \underbrace{11\dots1}_{k} \bar{d}_{n-1}\dots \bar{d}_1 \bar{d}_0 \bar{d}_{-1}\dots \bar{d}_{-\nu} \text{ при } x \le 0, \end{cases} \quad (6)$$

где $\bar{d}_i = (\beta - 1) - d_i (i = n-1, \dots, 1, 0, -1, \dots, -\nu)$. *Дополнительным кодом* числа x называется целое $(k+n+\nu)$-разрядное число $[x]^k_{don}$, определяемое соответственно формулой

$$[x]^k_{don} = \begin{cases} \underbrace{00\dots0}_{k} d_{n-1}\dots d_1 d_0 d_{-1}\dots d_{-\nu} \text{ при } x \ge 0, \\ \underbrace{11\dots1}_{k} \bar{d}_{n-1}\dots \bar{d}_1 \bar{d}_0 \bar{d}_{-1}\dots \bar{d}_{-\nu} \boxed{+} \underbrace{0\dots01}_{k+n+\nu} \text{ при } x \le 0, \end{cases} \quad (7)$$

где $\boxed{+}$ — операция арифметического сложения (начиная с младшего правого разряда) без переноса единицы из старшего разряда $n-1$ в n (первый знаковый разряд).

Старшие k разрядов всех кодов называют знаковыми, а остальные разряды — цифровыми. На практике чаще всего применяются коды с $k = 1$ и в обозначениях кодов $[x]^k$ верхний значок k опускается.

Как видно из приведенных формул для неотрицательных ($x \ge 0$) целых и вещественных (с фиксированной точкой) чисел, при одинаковом значении k их прямые, обратные и дополнительные коды совпадают. Для отрицательных чисел все три кода между собой отличаются. Пусть имеются, например, целые и вещественные (с фиксированной точкой) числа: 1 и 1., а также (-1) и $(-1.)$. Тогда их двоичные представления (при $k = 1$) в указанных выше (5)–(7) кодах для любых ЭВМ принимают вид, приведенный ниже в таблице 1. При этом в таблице 1 возрастание порядковых номеров двоичных разрядов (битов) происходит справа налево, а также знаковым разрядом числа является n-й старший (левый) разряд.

Таблица 1

	1(1.)	−1(−1.)
Прямой код	0000 ... 0001	1000 ... 0001
Обратный код	0000 ... 0001	1111 ... 1110
Дополнительный код	0000 ... 0001	1111 ... 1111

В случае же нормализованного числа x его порядок e и мантисса m могут представляться (как целые числа) в ЭВМ в любом из указанных кодов [6–8]. Например, в ЕС ЭВМ разряды 1-7 служат для представления порядка (характеристики) e числа x. При этом в этих разрядах порядок e представляется в дополнительном коде и $k = 1, n = 6$ (см. рис. $1_{1)}$). В старшем разряде будет 1, если $e \geq 0$, 0 — если $e < 0$. Мантисса представляется в прямом коде. В VAX, IBM PC, Convex, Spp и Sun ЭВМ мантиссы представляются в прямом коде, а порядки — в дополнительном коде. Значения величин k и n в Vax ЭВМ $k = 1, n = 7$ (см. рис. $1_{2)}$), а в Convex, Sun, Spp 2000 и IBM PC — $k = 1, n = 10$ (см. рис. $1_{3),4)}$).

Ниже (рис. 2) приведены примеры нормализованного представления некоторых из машинных констант: $\varepsilon_0 = \frac{1}{\beta} \cdot \beta^l$ — "машинный" нуль, $\varepsilon_\infty = (1 - \frac{1}{\beta^t}) \cdot \beta^u$ — "машинная" бесконечность, $\varepsilon_1 = \frac{1}{\beta} \cdot \beta^{2-t}$ — относительная погрешность, $1 = \frac{1}{\beta} \cdot \beta^1$ — единица и $\frac{1}{\beta} = \frac{1}{\beta} \cdot \beta^0$ — число обратное к основанию системы счисления. При этом на рис. 2 показано как реально указанные числа представляются в двоичном коде с учетом приведенных выше (рис. 1) форматов в ячейках памяти ЭВМ указанных типов.

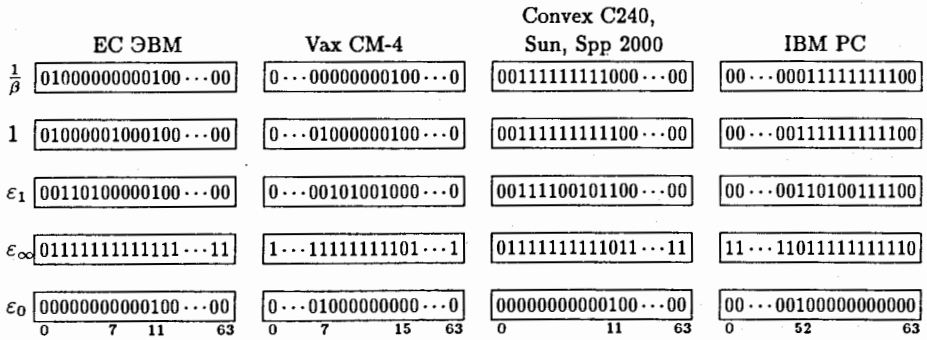

Рис. 2. Представление нормализованных чисел $\frac{1}{\beta}, 1, \varepsilon_1, \varepsilon_\infty$ и ε_0 в форматах различных ЭВМ

Информация о форматах представления чисел в ЭВМ важна, в частности, при вычислениях с использованием масштабирования (нормировки) чисел. Эта процедура реально может приводить к более устойчивым вычислениям на ЭВМ. Известны [1] два способа масштабирования чисел. Пусть, например, арифметические вычисления на ЭВМ производятся над множеством $\{x_i\}$ чисел x_i. При *первом способе нормировки,* прежде чем эти вычисления осуществить, числа x_i представляются в виде $x_i = a \cdot (\frac{1}{a} x_i) = a\tilde{x}_i$ с вещественным скаляром a. При этом [1] a может быть выбран, например, в виде $a = \sqrt{\sum_i x_i^2}$, либо в виде $a = \max_i |x_i|$. При *втором способе нормировки* числа x_i представляются в виде $x_i = \beta^\xi(\beta^{e_i - \xi} m_i) = \beta^\xi \hat{x}_i$, где e_i, m_i — β–ичный порядок и мантисса числа x_i, ξ — целое число, β — основание системы счисления. При этом ξ может быть выбран, например, в виде $\xi = \max_i e_i$, либо в виде среднего арифметического значения порядков $e_1, e_2, \ldots$. После выполнения указанных нормировок реальная работа ЭВМ осуществляется с числами $\tilde{x}_i$ либо $\hat{x}_i$.

Наиболее выгодный, с точки зрения накопления минимальных ошибок из-за округлений в результате выполнения арифметических операций на ЭВМ, способ масштабирования — второй [1] из указанных выше способов. При таком способе нормировки в среднем обеспечивается наиболее эффективный режим выполнения арифметических операций над мантиссами чисел.

Рассмотрим, например, следующую совместную невырожденную систему линейних алгебраических уравнений $AX = Y$:

$$\begin{bmatrix} 1/4 & (1-2^{-55})2^{-127} \\ 1/4 & (1-2^{-56})2^{-127} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

точным решением которой является вектор $X = (4, 0)$. Однако при решении этой системы, например, на VAX ЭВМ (с параметрами $\beta = 2, t = 56, l = -127, u = 127$) матрица системы "становится" вырожденной. В самом деле, при решении системы с помощью программ, например, `DEQN,DBEQN` из библиотеки `CERNLIB` [9] (метод Гаусса) и `PSOL` из пакета `LINA` [1] (SVD-разложение) выдаются сообщения о вырожденности и паталогически плохой обусловленности системы и решение не находится. Программа `SLAY` из библиотеки `LIBJINR` [10] (метод регуляризации Тихопова) вычисляет вектор–решение $X^\alpha = (4 + 10^{-16}, 0)$ с $\alpha \approx 10^{-9}$, если задается относительная ошибка правой части $\delta y = \varepsilon_1$. Решение указанной системы методом критических компонент [11,12] с использованием программ `LinsysccmSolver, Lin3dsysccmSolver` из библиотеки `LIBJINR` дает результаты $X = (4, 0)$ и $X = (4, 0)$ соответственно.

Если применить масштабирование по столбцам с $D_2 = \mathrm{diag}(1/4, (1-2^{-56})2^{-127})$ и $\tilde{D}_2 = \mathrm{diag}(2^{-1}, 2^{-127})$, где D_2 получена первым, а $\tilde{D}_2$ — вторым способом нормировки соответственно, то получим решения $\tilde{X} = (4 + 10^{-17}, 0)$ и $\bar{X} = (4, 0)$. Отсюда видно, что второй способ нормировки приводит к более точному результату. Отметим, наконец, что решение совместной линейной алгебраической системы $AX = Y$ порядка m может быть также найдено с использованием масштабирования обоих типов: $D_1^{-1} A D_2 Z = D_1^{-1} Y$, где $X = D_2 Z$. При этом

$$\begin{cases} D_1 = \mathrm{diag}(a_1, \ldots, a_m), \\ D_2 = \mathrm{diag}(b_1, \ldots, b_m) \end{cases} \text{ или } \begin{cases} D_1 = \mathrm{diag}(\beta^{e_1}, \ldots, \beta^{e_m}), \\ D_2 = \mathrm{diag}(\beta^{h_1}, \ldots, \beta^{h_m}), \end{cases}$$

где диагональные матрицы D_1, D_2 получены первым и вторым способом масштабирования соответственно. Если $D_2 = E$ (либо $D_1 = E$), где E — единичная матрица, то масштабирование называют строчным (столбцевым).

Форматы представления чисел используются также для получения констант вещественной арифметики ЭВМ.

2. Алгоритм получения констант вещественной арифметики ЭВМ

Ниже приводятся алгоритмы автоматического получения констант $\varepsilon_\infty, \varepsilon_0$ и ε_1 вещественной арифметики ЭВМ, а также целых констант β, t, l и u на основе анализа форматов представления нормализованных чисел в памяти ЭВМ.

Как известно [6], в современных ЭВМ нормализованные числа представляются в форматах из 8 байтов (64 двоичных разрядов), а целые числа — в форматах из 4 байтов (32 разрядов).

В основе предлагаемого алгоритма лежит *поразрядное исследование формата* представления нормализованных чисел в ЭВМ. Пусть в формате, например, из 64 двоичных разрядов представлено нормализованное число $\gamma_1 = (1.)$. Копируем этот

формат в другой формат τ такой же длины. Прежде чем описать дальнейшие шаги алгоритма отметим следующее. Если данная ЭВМ работает при основании $\beta = 2$, то известно [6], что содержимое старшего бита мантиссы может храниться и вне формата. Это обеспечивает увеличение длины мантиссы на один разряд. Поэтому при $\beta = 2$ в разрядах формата, отведенных для хранения мантиссы, в случае $\gamma_1 = (1.)$ будут все нули.

При $\beta > 2$ (и при $\beta = 2$, если единица мантиссы $\gamma_1 = (1.)$ хранится в формате) в разрядах, отведенных для хранения мантиссы числа $\gamma_1 = (1.)$, будут нули и лишь одна единица в старшем β-ичном разряде мантиссы. Эта единица является машинным представлением числа $1/\beta$. Для хранения любой значащей цифры от 0 до $\beta - 1$ (при основании β) будет отводиться n_β – двоичных разрядов формата. Следовательно мантисса числа $\gamma_1 = (1.)$ будет храниться в виде единицы в младшем из n_β – двоичных разрядов мантиссы формата τ.

Пусть мантисса m_1 и порядок e_1 числа $\gamma_1 = (1.)$ хранятся в прямом коде*) в формате τ в виде двух единиц в соответствующих разрядах. Все остальные разряды формата τ будут при этом нулевыми. Обозначим номера двоичных разрядов формата τ, в которых хранятся единицы соответственно μ_i и μ_j (см. рис. 3).

Рис. 3

При этом дальнейшее описание алгоритма не зависит от того является ли $i > j$ или наоборот. Прибавим (1.)–единицу к числу γ_1, т.е. получим в формате τ число $\gamma_2 = (\gamma_1 + 1.)$. Тогда, если основание β было равно 2, то в формате τ, в котором хранится теперь число $\gamma_2 = 2.$, единица мантиссы останется в прежнем двоичном разряде μ_i, а единица порядка переместится в следующий старший разряд μ_{j+1}. При этом в разряде μ_j будет 0 (см. рис. 4).

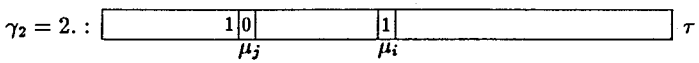

Рис. 4

Это обусловлено тем, что при любом основании β в n_β-двоичных разрядах будет представима единицами $(\beta - 1)$ — максимальная значащая цифра. Поэтому число β — основание системы счисления будет уже двузначным и оно представляется в формате единицей в разрядах мантиссы и двойкой в разрядах порядка. Следовательно, если произошел указанный сдвиг единицы в разрядах порядка, то основание $\beta = 2$ найдено. Если указанный сдвиг порядка не произошел, то $\beta > 2$. Но при этом единица мантиссы перемещается в следующий разряд μ_{i+1} (см. рис. 5).

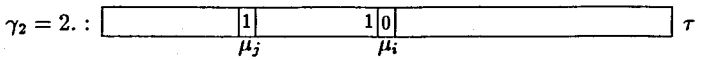

Рис. 5

В разряде μ_i будет 0, а единица порядка останется в прежнем двоичном разряде μ_j. Далее прибавляем (1.)–единицу к числу $\gamma_2 = (2.)$, т.е. получим в формате τ число $\gamma_3 = (3.)$. Тогда, если основание β было равно 3, то в формате τ, в котором хранится

*)Рассуждения, подобные приводимым ниже, имеют место и в случае любого другого кода представления нормализованных чисел. Отметим также, что при любом основании β-*базисными* (значащими) цифрами являются $0, 1, 2, \ldots, \beta - 1$.

теперь "двузначное" число $\gamma_3 = 3.$, единица мантиссы возвращается в разряд μ_i, разряд μ_{i+1} будет содержать 0, а единица порядка переместится в следующий старший разряд μ_{j+1} (см. рис. 6).

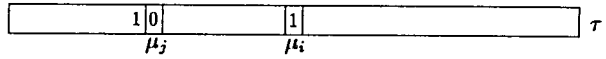

Рис. 6

Если такой сдвиг порядка не произошел, то $\beta > 3$ и алгоритм поиска β продолжается. Итак, пусть $\beta > 3$. Тогда в соответствии с выше описанной процедурой заполнения двоичных n_β-разрядов работа алгоритма поиска β прекращается как только происходит перемещение единицы порядка из разряда μ_j в двоичный разряд μ_{j+1}. При этом в n_β-двоичных разрядах мантиссы должно было бы храниться (в двоичном коде) число β, которое, как отмечено уже выше, является двузначным при основании β. Поэтому, как и при основаниях $\beta = 2; 3$, в младший из n_β-двоичных разрядов переместится 1, а остальные $(n_\beta - 1)$-старших из этих двоичных разрядов будут нулевыми. Таким образом, будет найдено число n_β-двоичных разрядов, в которых хранится в двоичном коде $(\beta - 1)$-максимальная из значащих цифр $0, 1, \ldots, \beta - 1$ при основании β в виде

$$\beta - 1 = (\mu_{i+n_\beta-1} \equiv 1) \cdot 2^{n_\beta - 1} + (\mu_{i+n_\beta-2}) \cdot 2^{n_\beta - 2} + \cdots + (\mu_{i+1}) \cdot 2^1 + (\mu_i) \cdot 2^0,$$

где $(\mu_{i+n_\beta-1}), (\mu_{i+n_\beta-2}), (\mu_{i+1}), (\mu_i)$ — содержимое (1 или 0) указанных двоичных разрядов $\mu_{i+n_\beta-1}, \mu_{i+n_\beta-2}, \mu_{i+1}, \mu_i$. При этом $\gamma_{k-1} = \beta - 1$ и $\gamma_k = \beta$ хранятся в формате τ в виде (см. рис. 7).

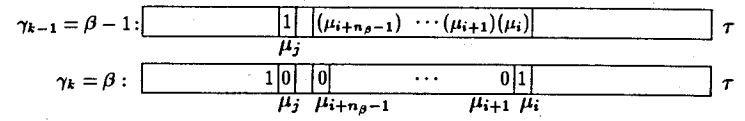

Рис. 7

Итак, основание β и n_β-двоичных разрядов, в которых хранятся значащие цифры $0, 1, 2, \ldots, \beta - 1$, найдены.

В дальнейшем группу из n_β-двоичных разрядов будем называть одним β-ичным разрядом. Найдем теперь константу t — число β-ичных разрядов в мантиссе нормализованного числа. Пусть в формате τ представлен нуль $(\gamma_0 = 0.)$. При этом все двоичные разряды формата (в том числе β-ичные разряды мантиссы) будут нулевыми. Прибавим к числу γ_0 число $s_1 = (\beta - 1)/\beta$. Тогда получим в формате τ число $\gamma_1 = \gamma_0 + s_1$. При этом в n_β-двоичных разрядах $\mu_{i+n_\beta-1}, \mu_{i+n_\beta-2}, \ldots, \mu_{i+1}, \mu_i$ будет храниться число $\beta - 1$ с учетом того, что β — основание системы счисления, а все остальные разряды формата τ будут нулевыми (см. рис. $8_{1),2)}$).

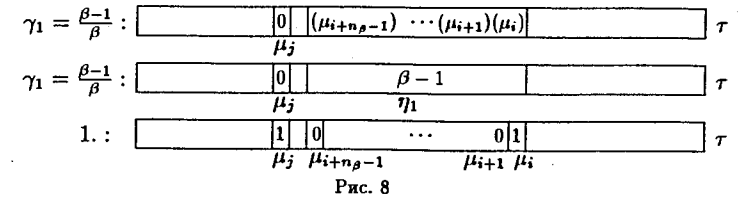

Рис. 8

Обозначим этот β-ичный разряд в виде η_1 (рис. 8_2)). Прибавим теперь к числу γ_1

число $s_2 = (\beta-1)/\beta^2 = s_1/\beta$. Тогда в формате τ должно будет храниться число

$$\gamma_2 = \gamma_1 + s_2 = 0.(\beta-1)(\beta-1).$$

Если $t = 1$, то в двоичных разрядах μ_i и μ_j формата τ будут единицы, а все остальные разряды формата будут нулевыми (см. рис. 8_3)). Это обусловлено следующим обстоятельством. Если для хранения мантиссы при основании β было отведено только n_β-двоичных разрядов, то число $\gamma_2 = 0.(\beta-1)(\beta-1)$ уже не может быть представлено в этих n_β-двоичных разрядах. Поэтому оно будет *округлено до единицы,* которая и представляется в формате τ в виде рис. 8_3). Если же для хранения мантиссы отведено более двух β-ичных разрядов (т.е. $t \geq 2$), то число γ_2 будет храниться в формате τ в виде, представленном на рис. 9.

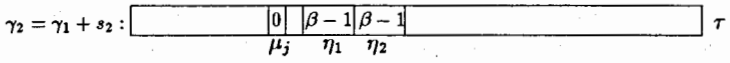

Рис. 9

Продолжая этот процесс при $t > 2$, получаем числа $\gamma_t = \gamma_{t-1} + (\beta-1)/\beta^t$. При этом, если для хранения мантиссы числа отводилось t β-ичных разрядов, то в итоге число γ_t будет храниться в формате τ в виде, представленном на рис. 10.

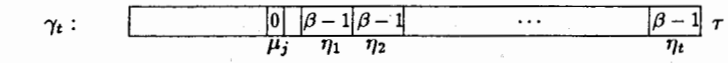

Рис. 10

Если теперь к числу γ_t добавить число $(\beta-1)/\beta^{t+1}$, то число γ_{t+1}

$$\gamma_{t+1} = \gamma_t + (\beta-1)/\beta^{t+1} = 0.\underbrace{\overbrace{(\beta-1)(\beta-1)\cdots(\beta-1)}^{t}(\beta-1)}_{t+1}$$

должно будет храниться также в формате τ в виде, представленном на рис. 10, что невозможно по вышеуказанной причине при $t = 1; 2; \ldots$. В результате произойдет округление числа γ_{t+1} до единицы и смена формата (рис. 10) на формат рис. 8_3). Таким образом, смена форматов τ (рис. 8, при $t = 1$, рис. 9, при $t = 2$, и рис. 10, при любом t), в которых хранятся числа $\gamma_1, \gamma_2, \ldots, \gamma_t$, на формат рис. 8_3), свидетельствует о том, что число t β-ичных разрядов найдено.

Найдем теперь n_e-двоичных разрядов, отведенных для хранения порядка нормализованного числа. Пусть в формате τ представлено число $\gamma_0 = 1/\beta$. При этом в разряде μ_i будет единица мантиссы, все остальные разряды формата будут нулевыми (рис. 11).

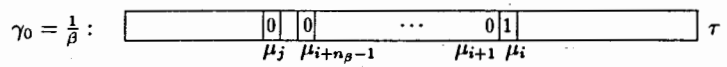

Рис. 11

Умножим число γ_0 на число $q = 1/\beta$. Тогда в формате τ должно будет храниться $\gamma_1 = 1/\beta^2$. Если при этом для хранения порядка отводился один двоичный разряд с учетом знака, то $l = 0$ найдено. Если же $|l| > 0$, то в формате τ будет храниться, с учетом основания β, число $\gamma_1 = 1/\beta^2 = 0,01$, что представлено на рис. 12.

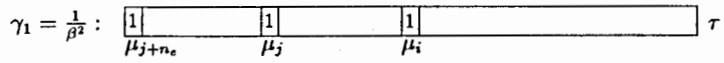

Рис. 12

Таким образом, в результате в порядковом разряде μ_j появится (-1), знак которой хранится в разряде*) μ_{j+n_e}, а разряды мантиссы не изменяются. Итак, если $l = -1$, то l найдено. Пусть теперь $l < -1$. Тогда снова умножим число γ_1 на число $q = 1/\beta$. В результате получим с учетом основания β число $\gamma_2 = 1/\beta^3 = 0,001$, которое должно будет храниться в формате τ в виде, изображенном па рис. 13.

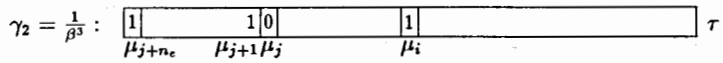

Рис. 13

В формате τ (рис. 13) может также храниться число $\gamma_3 = 1/\beta^4 = 0,0001$. При этом формат принимает вид, представленный на рис. 14.

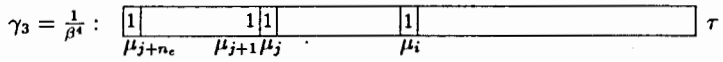

Рис. 14

Таким образом, если для хранения порядка l отводилось три двоичных разряда ($n_e = 2$) с учетом знакового разряда, то $l = -3$ найдено. Продолжая этот процесс, найдем число l, которое будет храниться в n_e-двоичных разрядах формата τ в виде, представленном на рис. 15.

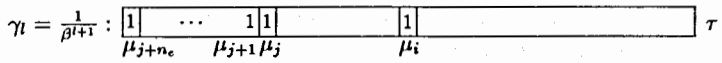

Рис. 15

Итак, найдено число $\varepsilon_0 = \frac{1}{\beta}\beta^l$ — минимальное из представимых в формате данной ЭВМ. Как уже отмечено выше, все вещественные числа $x \in R$, удовлетворяющие условию $|x| < \varepsilon_0$, будут машинными нулями (см. рис. 16). Для современных ЭВМ величина $\varepsilon_0 (\varepsilon_0 < \beta^{-1.5t})$ обычно находится в пределах $10^{-38} - 10^{-308}$ [3,6].

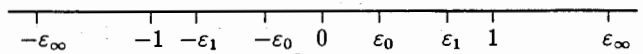

Рис. 16

Найдем, наконец, число ε_∞ — максимальное из представимых в формате данной ЭВМ, отличное от машинной бесконечности (см. рис. 16). При этом, как отмечено выше, все вещественные числа $x \in R$ вида $|x| > \varepsilon_\infty$ будут машинными бесконечностями. В соответствии с определением ε_0 в формате с плавающей точкой число ε_∞ определяют [3] в виде $\tilde{\varepsilon}_\infty \approx \varepsilon_0^{-1}$. Итак, пусть $\tilde{\varepsilon}_\infty \approx \varepsilon_0^{-1}$ хранится в формате τ. Тогда, если в формате τ может храниться $\beta\tilde{\varepsilon}_\infty$, то $\varepsilon_\infty = \beta\tilde{\varepsilon}_\infty$. Все числа $|x| > \varepsilon_\infty$ считаются машинными бесконечностями. При этом, если при вычислениях получается $|x| > \varepsilon_\infty$, то в зависимости от принятой в данной ЭВМ концепции (принцип умолчания), либо $|x|$ полагают равным ε_∞, либо вычислительный процесс останавливается. Итак, найдено число ε_∞ — максимальное из представимых в формате данной ЭВМ с плавающей точкой.

*) Двоичный разряд μ_{j+n_e} – знаковый разряд порядка может быть в любом месте формата. Для наглядности нами выбран крайне левый двоичный разряд формата.

Таким образом, в соответствии с представленным выше алгоритмом могут быть автоматически найдены единственные базисные константы $(\beta, t, l, u$ и $\varepsilon_\infty, \varepsilon_0, \varepsilon_1)$ вещественной арифметики с плавающей точкой данного компьютера.

3. Описание программ получения констант вещественной арифметики ЭВМ

В этом параграфе приводится описание программ: INIT — получения констант вещественной арифметики ЭВМ, а также получения параметров настройки функций GTEXP — выделения мантиссы и порядка вещественного числа и CHEXP — восстановления вещественного числа по его мантиссе и порядку. Приводятся также некоторые результаты работы программы INIT, а также тексты программ INIT, GTEXP, CHEXP. Программы написаны на Фортране-77 [13] и на основе алгоритмов получения констант вещественной арифметики ЭВМ, изложенных в параграфе 2.

Описание программы INIT

Структура: SUBROUTINE

Внешние подпрограммы и функции: INITP — подпрограмма печати вычисленных машинных констант

Общие блоки: COMMON /RCONST/ OVFLO,UNDFLO,EPS,EPSMIN
COMMON /ICONST/ RADIX,MAXEXP,MINEXP,MANTSZ
COMMON /ICHGT/ INO,ILT,MNE1,MNE2,INB,LWT,KB,BINV

Обращение: CALL INIT, если печать вычисленных костант не требуется;
CALL INITP, если константы печатаются

Входные данные: нет (т.к. подпрограмма определяет базовые костанты данной ЭВМ)

Выходные данные: вещественные константы $\varepsilon_\infty, \varepsilon_0, \varepsilon_1, \varepsilon_2 = \varepsilon_1/\beta$ и целые константы β, u, l, t. Выдаются в блоках RCONST и ICONST в указанных последовательностях соответственно. В блоке ICHGT содержатся следующие выделенные параметры настройки функции GTEXP, CHEXP:

- BINV – (real*8) вещественное число равное $1/\beta$;
- KB – (integer) номер половины формата, в котором хранится младший разряд порядка нормализованного числа;
- LWT – (integer) представление целого порядка $e = 0$ в разрядах формата, отведенных для порядка;
- INO – (integer) количество β-ичных разрядов, отведенных для порядка с учетом его знака (в формате для хранения вещественного числа);
- INB – (integer) количество β-ичных разрядов, отведенных для мантиссы с учетом ее знака (в формате для хранения вещественного числа);
- MNE1 – (integer) признак кода $(e > 0)$–положительного порядка. При этом MNE1 $= [e]^k_{код} - e$ (см. параграф 1);
- MNE2 – (integer) признак кода $(e < 0)$–отрицательного порядка. При этом MNE2 $= [e]^k_{код} - e$ (см. параграф 1);

Метод: алгоритм получения констант вещественной арифметики ЭВМ, изложенный в параграфе 2.

Печать: подпрограмма INITP вычисляет и печатает константы вещественной арифметики ЭВМ.

Функции GTEXP, CHEXP выделяют мантиссу и порядок и, соответственно, восстанавливают по ним вещественное число. Если при этом содержимое блока ICHGT уже сформировано, то функции GTEXP, CHEXP работают без предварительного обращения к INIT. В противном случае, перед обращением к функциям, сначала вызывается INIT.

Описание программы GTEXP

Структура: DOUBLE PRECISION FUNCTION

Внешние подпрограммы и функции: нет

Общие блоки: COMMON /ICHGT/ INO,ILT,MNE1,MNE2,INB,LWT,KB,BINV

Обращение: RM=GTEXP(R,E)

Входные данные: R – (real*8) вещественное число, порядок и мантисса которого выделяется; блок ICHGT содержит настроечные параметры, определенные в INIT (или заранее известные)

Выходные данные: RM – (real*8) содержит выделенную мантиссу числа R; E – (integer) содержит выделенный порядок числа R

Метод: логические операции по выделению мантиссы и порядка с учетом информации блока ICHGT.

Описание программы CHEXP

Структура: DOUBLE PRECISION FUNCTION

Внешние подпрограммы и функции: нет

Общие блоки: COMMON /ICONST/ RADIX,MAXEXP,MINEXP,MANTSZ
COMMON /ICHGT/ INO,ILT,MNE1,MNE2,INB,LWT,KB,BINV

Обращение: R=CHEXP(RM,E)

Входные данные: RM – (real*8) мантисса числа R; E – (integer) порядок числа R; блоки ICONST и ICHGT содержат элементы, определенные в INIT (или заранее известные)

Выходные данные: R – (real*8) содержит вещественного числа β^{E}·RM=R

Метод: логические операции по восстановлению вещественного число по мантиссе и порядку с учетом информации блоков ICONST, ICHGT.

Пример: Выделить на IBM PC $(\beta = 2, t = 53, l = -1021, u = 1024)$ мантиссы и порядки вещественных чисел: $1 = \frac{1}{\beta} \cdot \beta^1$ — единицы, $\varepsilon_1 = \frac{1}{\beta} \cdot \beta^{2-t}$ — относительной погрешности, $\varepsilon_0 = \frac{1}{\beta} \cdot \beta^l$ — "машинного" нуля и $\varepsilon_\infty = (1 - \frac{1}{\beta^t}) \cdot \beta^u$ — "машинной" бесконечности, а также восстановить эти числа по их мантиссам и порядкам. Ниже приводятся текст программы решения этой задачи, а также таблицы 2 и 3 результатов работы программы.

```
PROGRAM TEST
IMPLICIT REAL*8 (A-H, O-Z)
INTEGER RADIX
COMMON /RCONST/ OVFLO,UNDFLO,EPS,EPSMIN
COMMON /ICONST/ RADIX,MAXEXP,MINEXP,MANTSZ
CALL INIT
D1=GTEXP(1DO,I)
DE=GTEXP(EPS,J)
DO=GTEXP(OVFLO,L)
```

```
      DU=GTEXP(UNDFLO,K)
      D=1D0
      PRINT1,D,D1,I,EPS,DE,J,UNDFLO,DU,K,OVFLO,DO,L,
    1 FORMAT(10X,'ЧИСЛО',10X,':',6X,'МАНТИССА',6X,': ПОРЯДОК',/,
     * 25('-'),
     * '+',20('-'),'+',8('-'),4(/,E24.15E3,' :',F19.16,' : ',I5),/)
      CALL SUB(D1,I,DE,J,DO,L,DU,K)
      STOP
      END
C
      SUBROUTINE SUB(D,I,DE,J,DO,L,DU,K)
      IMPLICIT REAL*8 (A-H, O-Z)
      D1=CHEXP(D,I)
      EPS=CHEXP(DE,J)
      OVFL=CHEXP(DO,L)
      UNDF=CHEXP(DU,K)
      PRINT1,D,I,D1,DE,J,EPS,DE,K,UNDF,DO,L,OVFL
    1 FORMAT(6X,'МАНТИССА',6X,': ПОРЯДОК:',10X,'ЧИСЛО',/,20('-'),
     * '+',8('-'),'+',25('-'),4(/,F19.16,' : ',I5,'  :',E24.15E3))
      RETURN
      END
```

Таблица 2 *выделенных мантисс и порядков вещественных чисел:* $1 = \frac{1}{\beta} \cdot \beta^1$, $\varepsilon_1 = \frac{1}{\beta} \cdot \beta^{2-t}, \varepsilon_0 = \frac{1}{\beta} \cdot \beta^l$ и $\varepsilon_\infty = (1 - \frac{1}{\beta^t}) \cdot \beta^u$

ЧИСЛО	МАНТИССА	ПОРЯДОК
.100000000000000E+001	.5000000000000000	1
.222044604925031E-015	.5000000000000000	-51
.222507385850720E-307	.5000000000000000	-1021
.179769313486232E+309	.9999999999999999	1024

Таблица 3 *восстановленных по мантиссам и порядкам вещественных чисел:* $\frac{1}{\beta} \cdot \beta^1 = 1, \frac{1}{\beta} \cdot \beta^{2-t} = \varepsilon_1, \frac{1}{\beta} \cdot \beta^l = \varepsilon_0$ и $(1 - \frac{1}{\beta^t}) \cdot \beta^u = \varepsilon_\infty$

МАНТИССА	ПОРЯДОК:	ЧИСЛО
.5000000000000000	1	.100000000000000E+001
.5000000000000000	-51	.222044604925031E-015
.5000000000000000	-1021	.222507385850720E-307
.9999999999999999	1024	.179769313486232E+309

Результаты работы программы INIT

Приведем таблицы 4 и 5 констант вещественной арифметики ЭВМ, полученные с помощью программы INIT, текст которой приводится ниже. При этом в таблицах вещественные числа $\varepsilon_\infty, \varepsilon_0, \varepsilon_1, \varepsilon_2$ представлены в десятичной системе счисления.

Таблица 4

Тип ЭВМ	β	l	u	t	$\varepsilon_2 = \varepsilon_1/\beta$
Convex 120, Sun, Spp 2000	2	-1021	1024	53	1.110223024625157E-16
Convex 240	2	-1023	1023	53	1.110223024625157E-16
Vax,CM-4	2	-127	127	56	1.387778780781416E-17
IBM PC	2	-1021	1024	53	1.110223024625157E-16
EC ЭВМ	16	-64	63	14	1.387778780781416E-17

Таблица 5

Тип ЭВМ	ε_0	ε_∞	ε_1
Convex 120, Sun, Spp 2000	2.225073858507201E-308	1.797693134862316E308	2.220446049250313E-16
Convex 240	5.562684646268003E-309	8.988465674311579E308	2.220446049250313E-16
Vax,CM-4	2.938735877055719E-039	1.701411834604692E038	2.775557561562891E-17
IBM PC	2.225073858507201E-308	1.797693134862316E308	2.220446049250313E-16
EC ЭВМ	5.397605346934028E-079	7.237005577332262E075	2.220446049250313E-16

Тексты программ INIT, GTEXP, CHEXP на Фортране-77

```
      SUBROUTINE INIT
      IMPLICIT REAL*8 (A-H, O-Z)
      INTEGER RADIX,J1(2),J2(2),J3(2),N1(2),N2(2),N3(2)
      COMMON /RCONST/ OVFLO,UNDFLO,EPS,EPSMIN
      COMMON /ICONST/ RADIX,MAXEXP,MINEXP,MANTSZ
      COMMON /ICHGT/ INO,ILT,MNE1,MNE2,INB,LWT,KB,BINV
      EQUIVALENCE (J1,R1),(J2,R2),(J3,R3)
      EQUIVALENCE (OVFLO,N1),(UNDFLO,N2),(EPS,N3)
      GOTO 1
      ENTRY INITP
      LPCONST=1
    1 R1=1D0
      RADIX=1
      KB=1
C     ЦИКЛ ПОИСКА (BETA) - ОСНОВАНИЯ СИСТЕМЫ СЧИСЛЕНИЯ
    2 RADIX=RADIX+1
      R2=RADIX
      R3=1D0/R2
      ILT=J2(1)-J1(1)
      INO=J2(2)-J1(2)
      IF(ILT.NE.(J1(1)-J3(1)).OR.INO.NE.(J1(2)-J3(2))) GOTO 2
C     ЦИКЛ ПОИСКА t - РАЗРЯДОВ, ОТВЕДЕННЫХ ДЛЯ ХРАНЕНИИ МАНТИССЫ m
      D=R3*(R2-R1)
      UNDFLO=D
      MANTSZ=0
      BINV=R3
      IF(INO.EQ.0) GOTO 3
      ILT=INO
      KB=2
```

```
    3   OVFLO=UNDFLO
        MANTSZ=MANTSZ+1
        D=D*R3
        UNDFLO=UNDFLO+D
        EPS=UNDFLO
        IF(EPS.LT.R1) GOTO 3
        EPSMIN=R1-OVFLO
        MINEXP=2-MANTSZ
        EPS=EPSMIN
        UNDFLO=EPS*R2
        MNE2=N2(KB)
        IF(MNE2.LT.N3(KB)) MNE2=-MNE2
        MNE2=MNE2/ILT-MINEXP
        I=IABS(N3(KB)-N2(KB))
C     ЦИКЛ ВЫЧИСЛЕНИЯ e_0 - МИНИМАЛЬНО ПРЕДСТАВИМОГО
C     В ЭВМ ЧИСЛА И ЕГО ПОРЯДКА
    4   UNDFLO=EPS
        MINEXP=MINEXP-1
        D=UNDFLO*R3
        EPS=D
        J=N3(KB)-N2(KB)
        IF(EPS.GT.ODO.AND.I.EQ.IABS(J)) GOTO 4
C     ЦИКЛ ВЫЧИСЛЕНИЯ u - ПОРЯДКА МАКСИМАЛЬНО ПРЕДСТАВИМОГО
C     В ЭВМ ЧИСЛА e_8
        EPS=R1
        D=R2
        I=1
    5   MAXEXP=2*I
        INO=N3(KB).OR.INO
        IF(MAXEXP.GE.-MINEXP) GOTO 6
        EPS=EPS*D
        D=D*D
        I=MAXEXP
        GOTO 5
    6   IF((MAXEXP+MINEXP).LE.1) MAXEXP=MAXEXP-1
C     ЦИКЛ ВЫЧИСЛЕНИЯ e_8 - МАКСИМАЛЬНО ПРЕДСТАВИМОГО В ЭВМ ЧИСЛА
        OVFLO=OVFLO*EPS
        DO 7 K=I,MAXEXP
        OVFLO=OVFLO*R2
    7   CONTINUE
C     БЛОК ВЫЧИСЛЕНИЯ ПАРАМЕТРОВ ВЫДЕЛЕНИЯ ПОРЯДКА И МАНТИССЫ ЧИСЛА
        EPS=R3*R3
        I=MINO(N3(KB),N2(KB),J3(KB))
        INO=(INO-I).OR.(N2(KB)-I).OR.(J3(KB)-I)
        INO=INO.OR.(N3(KB)-I).OR.(J1(KB)-I)
        INB=.NOT.INO
        LWT=J3(KB).AND.INO
        MNE1=LWT/ILT
```



```
        EPS=EPSMIN*R2
        IF(LPCONST.NE.1) GOTO 8
        PRINT*,'The constants of mashin's arithmetic ...'
        PRINT9, MAXEXP,OVFLO,MINEXP,UNDFLO,RADIX,EPS,MANTSZ,EPSMIN
    9   FORMAT(1X,'MAXEXP=',I8,'   OVFLO=',1PE24.15E3,/,
     *         1X,'MINEXP=',I8,'  UNDFLO=',1PE24.15E3,/,
     *         1X,' RADIX=',I8,'     EPS=',1PE24.15E3,/,
     *         1X,'MANTSZ=',I8,'  EPSMIN=',1PE24.15E3)
    8   RETURN
        END
C
        REAL FUNCTION GTEXP*8(R,E)
        INTEGER I(2),E
        REAL*8 R,Z,BINV
        COMMON /ICHGT/ INO,ILT,MNE1,MNE2,INB,LWT,KB,BINV
        EQUIVALENCE (Z,I)
        IF(R.EQ.ODO) GOTO 2
        Z=R
        J=MNE1
        E=(I(KB).AND.INO)/ILT
        IF(DABS(Z).GE.BINV) GOTO 3
        IF(MNE2.LT.0) E=-E
        J=MNE2
    3   E=E-J
        I(KB)=(I(KB).AND.INB).OR.LWT
        GTEXP=Z
    1   RETURN
    2   GTEXP=0
        E=0
        GOTO 1
        END
C
        REAL FUNCTION CHEXP*8(R,E)
        INTEGER I(2),L,J,E,RADIX
        REAL*8 R,Z,BINV
        COMMON /ICHGT/ INO,ILT,MNE1,MNE2,INB,LWT,KB,BINV
        COMMON /ICONST/ RADIX,MAXEXP,MINEXP,MANTSZ
        EQUIVALENCE (Z,I)
        IF(E.EQ.0.OR.R.EQ.ODO) GOTO 2
        Z=R
        K=MNE1
        J=(I(KB).AND.INO)/ILT
        IF(DABS(Z).GE.BINV) GOTO 4
        IF(MNE2.LT.0) J=-J
        K=MNE2
    4   L=J-K+E
        IF(L.GT.MAXEXP) L=MAXEXP
        K=MNE1
```

```
      IF(L.GT.0) GOTO 5
      IF(L.LT.MINEXP) GOTO 3
      K=IABS(MNE2)
      IF(MNE2.LT.0) L=-L
    5 I(KB)=(I(KB).AND.INB).OR.((L+K)*ILT)
      CHEXP=Z
    1 RETURN
    2 CHEXP=R
      GOTO 1
    3 CHEXP=0
      GOTO 1
      END
```

Благодарности

Авторы выражают благодарность д.ф.-м.н. профессору Г.А.Емельяненко за постановку задачи, плодотворные дискуссии и советы. Авторы признательны также к.ф.-м.н. А.П.Сапожникову за полезные обсуждения и замечания.

Литература

[1] Малышев А.Н. Введение в вычислительную линейную алгебру (с приложением алгоритмов на ФОРТРАНе). Новосибирск, "Наука", 1991.

[2] Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений. М., "Мир", 1980.

[3] Воеводин В.В. Вычислительные основы линейной алгебры. М., "Наука", 1977. вычислений. М., "Мир", 1980.

[4] Голуб Дж. Ван Лоун Ч. Матричные вычисления. М., "Мир", 1999.

[5] Фихтенгольц Г.М. Курс дифференциального и интегрального исчисления. Т. I. М., "Наука", 1969.

[6] Майерс Г. Архитектура современных ЭВМ. М., "Мир", 1985.

[7] Трифонов Н.П., Громыко В.И. Программирование на автокоде ЕС ЭВМ. М., "Наука", 1985.

[8] Криницкий Н.А., Миронов Г.А., Фролов Г.Д. Программирование и алгоритмические языки. М., "Наука", 1979.

[9] CERNLIB-CERN Program Library (Short Writeups). Application Software Group. Computing and Networks Division. CERN, Geneva, Switzerland (May, 1998).

[10] Федорова Р. Н., Широкова А. И. Библиотека программ на ФОРТРАНе. т. *VI – VII*. Описание программ. Дубна, 1990.

[11] Emel'yanenko G. A., Emelianenko M. G., Rakhmonov T. T., Dushanov E.B., Konovalova G.Yu. JINR Preprint, E11-98-302, Dubna, 1998.

[12] Новости ОИЯИ (JINR News). Информационный бюллетень Объединенного института ядерных исследований. 3, 1996, стр. 12.

[13] Белецки Я. Фортран 77. М., "Высшая школа", 1991.

Рукопись поступила в издательский отдел
17 июля 2000 года.