ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
1973 год Выпуск 54

УДК 681.3.06:51

ЯЗЫК КОНСТРУКТИВНЫХ ОПРЕДЕЛЕНИЙ

П.А. Анишев, В.И. Кашин, Б.А. Сидристый

При решении задач с применением ЭВМ работа программиста может быть разделена на следующие этапы (рис. I):

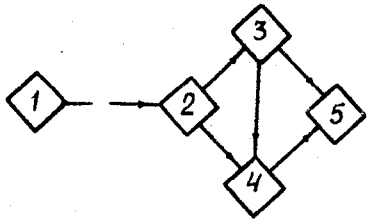

Рис. I

I. Формулировка задачи и выделение объектов задачи. Под объектами здесь понимаются исходные данные и получаемые результаты, их свойства и взаимосвязи.

2. Кодирование объектов задачи, т.е. сопоставление реальным объектам формальных объектов языка программирования. Сюда же относится решение вопросов представления информации в памяти ЭВМ, распределение памяти (если это не делается автоматически) и т.п.

3. Получение системы определений результативных и промежуточных объектов через исходные. Результаты этого этапа могут записываться на формальном языке или просто на русском языке.

4. Составление алгоритма решения задачи и написание программы на выбранном алгоритмическом языке.

5. Реализация программы на ЭВМ.

В отдельных случаях некоторые этапы могут быть пропущены. Например, путь I-2-4-5 возможен, если задача довольно простая и требуется только написать программу на каком-либо алгоритмическом языке. Нам кажется, что особого внимания заслуживает пункт 3 и путь I-2-3-5, так как логически завершенная система определений обычно достаточно однозначно задает алгоритм решения задачи и составление программы решения уже не требует затрат творческого труда и, в принципе, может быть получено автоматически. В этом случае исходной информацией для составления программы решения задачи на ЭВМ в машинных кодах служит описание системы определений на каком-либо входном языке.

В данной работе для такого описания предлагается язык конструктивных определений (КОП-язык). Исходная, промежуточная и результирующая информации задачи в этом языке представляются в виде совокупности отношений и функций. Основными конструкциями КОП-языка являются определения, при построении которых используется язык функционального исчисления первого порядка [2], а также логические и функциональные связки и некоторые операторы, описанные ниже.

Описание языка

I. Алфавит.

I.I. Большие и малые буквы латинского и русского алфавитов, а также символы - (дефис) и ~ (безразличность).

I.2. Цифры 0|I|...|9.

I.3. Логические связки: $\wedge$ (конъюнкция), $\vee$ (дизъюнкция), $\neg$ (отрицание), $\Longleftrightarrow$ (эквивалентность), $\Longrightarrow$ (импликация), $\oplus$ (антиэквивалентность или сумма по $mod\,2$).

I.4. Кванторы: $\forall$ - квантор общности, $\exists$ - квантор существования.

I.5. Функциональные связки: +|-|x|÷|/ (соответственно: сумма, разность, произведение, деление нацело, деление).

I.6. Символы стандартных предикатов и функций: =|≠|≥|≤|>|<|⊘|⊙|$\wedge$|пр i,...,j (где i,...,j - различные натуральные числа).

I.7. Специальные символы: =|по|итерация по|если|до|иначе|где|min|max|сумма|произведение|$card$|первый в|номер|в.

I.8. Технические символы (разделители): .|,|:|;
Для большей наглядности используются квадратные и фигурные скобки, которые, вообще говоря, не входят в алфавит.

2. Основное множество.

Обозначим через $\mathcal{D}$ множество действительных чисел, через $\mathcal{N}$ вполне упорядоченное множество имен, элементы которого предназначены для идентификации объектов. Тогда $\mathcal{A} = \mathcal{N} \cup \mathcal{D}$ есть основное множество.

Порядок любого конечного подмножества $\mathcal{M} \subset \mathcal{N}$ индуцируется естественным порядком в множестве $\mathcal{N}$. Если $\mathcal{M}_i \subset \mathcal{N}$ $(i=1,\dots,n)$ - конечные множества и $D = \mathcal{M}_1 \times \dots \times \mathcal{M}_n$, то порядок элементов множества D устанавливается каким-либо из обычных способов (например, при помощи канторовской нумерации). Порядок множества $A \subset D$ индуцируется по порядку в множестве D.

3. Основные понятия.

3.1. Предметные переменные и константы.

3.1.1. Идентификатором предметной переменной служит малая или большая буква русского или латинского алфавитов. Различаются вещественные и именные предметные переменные. Областью значений именной предметной переменной является множество $\mathcal{N}$. Областью значений вещественной предметной переменной является множество $\mathcal{D}$.

3.1.2. Предметной константой служит произвольный элемент множества $\mathcal{A}$. Именная предметная константа, т.е. элемент множества $\mathcal{N}$, представляет собой число, состоящее из перенумерованных частей одинаковой длины ℓ. Например, именная константа 65400 состоит из следующих частей длины $\ell = 2$: I-я часть - 00, 2-я часть - 54, 3-я часть - 06, 4-я (5-я, 6-я,...) часть - 00. Длина ℓ части константы фиксируется сразу для всех имен - элементов множества $\mathcal{N}$.

3.2. Пропозициональные константы: истина (И), ложь (Л), ~ (безразличное значение).

3.3. Предикатные переменные и константы.

3.3.1. Идентификатором предикатной переменной служит последовательность букв, цифр, пробелов и дефисов, начинающаяся с буквы. Например: R, $R1$, ЗАВ-СТЬ. Значением предикатной переменной является предикатная константа.

3.3.2. Предикатная константа - это конкретный предикат. Предикаты определяются на основном множестве $\mathcal{A}$. Областью значений предиката является множество {И, Л, ~}. Стандартные предикаты: $= | \neq | \geqslant | \leqslant | > | <$ - это известные бинарные отношения. Стан-

дартные предикаты $\textcircled{>} | \textcircled{<} | \underline{N}$ определяются следующим образом (знаки =, +, - в пунктах а) и б) относятся к метаязыку):

а) $a \textcircled{>} b = \underline{И}$ $(a \textcircled{<} b = \underline{И})$, если $a = b+1$ $(a = b-1)$ и $a, b \in \mathcal{D}$, или $a, b \in \mathcal{N}$; $a \textcircled{>} b = \underline{Л}$ $(a \textcircled{<} b = \underline{Л})$ в противном случае;

б) $\underline{N}(a) = \underline{И}$, если $a \in \mathcal{N}$, и $\underline{N}(a) = \underline{Л}$ - в противном случае.

3.4. Функциональные переменные и константы.

3.4.1. Идентификатор и значение функциональной переменной определяются так же, как и для предикатной переменной.

3.4.2. Функции (или, что то же самое, функциональные константы) определяются на основном множестве и могут быть двух типов: вещественные и именные. Областью значений вещественной функции является множество $\mathcal{D} \cup \{\sim\}$, областью значений именной функции - множество $\mathcal{N} \cup \{\sim\}$.

Стандартная функция • (свертка частей) определяется следующим образом. Функция $\bullet\, i_1, \dots, i_s$ (a) от именной переменной a имеет своим значением именную константу $a_{i_s}, \dots, a_{i_1}$, если a_{i_j} $(s \geq j \geq 1)$ есть i_j-я часть значения переменной a (см. 3.1.2). Например, если длина части равна 2, то имеем:

a	$\bullet\, 3\ (a)$	$\bullet\, 1,3\ (a)$	$\bullet\, 3,2,2\ (a)$
5400	0	0	545400
65400	6	600	545406
365002	36	3602	505036
...	...	...	...

если a - вещественная переменная, то $\bullet\, i_1, \dots, i_s = \sim$.

3.5. Кванторы.

3.5.1. Квантор $\forall$. Значением выражения

$$\forall x_i : R(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)$$

(где R - n - арный предикат) является $(n-1)$-арный предикат

$$Q = Q(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n),$$

причем

$$Q = \begin{cases} \underline{И}, & \text{если для каждого } a \text{ из области значений переменной } x_i \\ & \text{предикат } R(x_1, \dots, x_{i-1}, a, x_{i+1}, \dots, x_n) = \underline{И}; \\ \sim, & \text{если хотя бы для одного } a \text{ из области значений переменной } x_i, \text{ предикат } R(x_1, \dots, x_{i-1}, a, x_{i+1}, \dots, x_n) = \sim; \\ \underline{Л} & \text{- в остальных случаях.} \end{cases}$$

3.5.2. Квантор $\exists$ является двойственным квантору $\forall$.

4. Элементарные конструкции.

Элементарными конструкциями являются термы, формулы, операторы и операторные выражения. Определение терма и формулы см. [I].

4.I. Термы. Значения термов вычисляются по таблице I .

4.2. Формулы. Значения формул вычисляются по таблицам 2 и 3.

Принятое здесь расширение общепринятых арифметических и булевских операций состоит в том, что результат принимает значение $\{\sim\}$, если хотя бы один из операндов имеет значение $\{\sim\}$.

Таблица I

A	B	+	-	/	x	÷
C_1	C_2	C_1+C_2	C_1-C_2	C_1/C_2	$C_1 \times C_2$	$C_1 \div C_2$
C_1	~	~	~	~	~	~
~	C_2	~	~	~	~	~
~	~	~	~	~	~	~

Таблица 2

F	$\neg$F
И	Л
Л	И
~	~

Таблица 3

F	G	$\wedge$	$\vee$	$\Rightarrow$	$\oplus$	$\Leftrightarrow$
И	И	И	И	И	Л	И
И	Л	Л	И	Л	И	Л
И	~	~	~	~	~	~
Л	И	Л	И	И	И	Л
Л	Л	Л	Л	И	Л	И
Л	~	~	~	~	~	~
~	И	~	~	~	~	~
~	Л	~	~	~	~	~
~	~	~	~	~	~	~

4.3. Операторы.

Операторы в КОП-языке определены на предикатах и функциях и в качестве значений имеют функции. Будем обозначать через $\hat{x}$ предметную переменную x , от которой некоторая функция не зависит.

4.3.I. Операторы $\underline{min}$, $\underline{max}$, $\underline{\text{сумма}}$, $\underline{\text{произведение}}$.

Пусть $F(x_1,\dots,x_n)$-некоторая вещественная функция. $R(x_1,\dots,x_n)$- предикат, и $s \le n$.

Оператор $\underline{min}\, F(x_1,\dots,x_n)$ $\underline{\text{по}}$ $x_{i_1},\dots,x_{i_s} : R(x_{i_1},\dots,x_{i_s})$ в качестве значения имеет функцию $G(x_1,\dots,\hat{x}_{i_1},\dots,\hat{x}_{i_s},\dots,x_n)$, причем минимум вычисляется по тем значениям предметных переменных $x_{i_1},\dots,x_{i_s}$, для которых предикат R истинен, и для тех наборов значений предметных переменных $x_1,\dots,x_n$, для которых значение функции F не равно $\sim$. Если предикат R истинен для всех наборов значений переменных $x_1,\dots,x_n$, то оператор min можно записать так:

$$\underline{min}\, F(x_1,\dots,x_n) \text{ по } x_{i_1},\dots,x_{i_s}.$$

Операторы $\underline{max}$, $\underline{\text{сумма}}$, $\underline{\text{произведение}}$, определяются аналогично.

4.3.2. Оператор

$$\underline{card}\ R(x_1,\dots,x_n) \text{ по } x_{i_1},\dots,x_{i_s}.$$

(R - некоторый предикат, $s \le n$) в качестве значения имеет функцию

$$F(x_1,\dots,\hat{x}_{i_1},\dots,\hat{x}_{i_s},\dots,x_n),$$

которая для каждого набора

$$a_1,\dots,a_j,\dots,a_n \quad (j \ne i_k \text{ для } k=1,\dots,s)$$

значений своих переменных равна количеству n -ок

$$(a_1,\dots,x_{i_1},\dots,a_j,\dots,x_{i_s},\dots,a_n),$$

для которых предикат R истинен. В частности,

$$\underline{card}\ R(x_1,\dots,x_n) \text{ по } x_1,\dots,x_n$$

имеет значение нуль-арной функции - мощности множества n -ок ($a_1,\dots,a_n$), для которых $R(a_1,\dots,a_n) = \underline{\text{И}}$.

4.3.3. Оператор

$$\underline{\text{номер}}\ x_{i_1},\dots,x_{i_s}\ \underline{\text{в}}\ R(x_1,\dots,x_n).$$

В этом операторе предметные переменные $x_{i_1},\dots,x_{i_s}$ могут быть только именными, а остальные предметные переменные предиката R , если они есть, любые. Мы определим частный вид этого оператора: $\underline{\text{номер}}\ x\ \underline{\text{в}}\ R(x,y)$.

Обобщение не вызывает существенных трудностей.

Оператор

номер x в $R(x,y)$

имеет своим значением бинарную функцию $F(x,y)$, которая при каждом фиксированном наборе значений $\overline{x}$, $\overline{y}$ своих переменных равна порядковому номеру элемента $\overline{x}$ в множестве $\{a\}$ таких значений переменной x, что $R(a,y) = И$ в соответствии с введенным порядком в множестве $\{a\}$ (см. п. 2).

4.3.4. Оператор

первый в $R(x)$

(R - унарный предикат) имеет своим значением нуль-арную функцию, т.е. принимает такое значение a именной переменной x, что

$[$номер x в $R(x)]$ = I при $x = a$,

где a - первый, начальный по стандартному порядку элемент множества значений переменной x, для которых предикат R истинен.

4.3.5. Оператор

$F(x_1,\dots,x_n)$, где $x_{i_1},\dots,x_{i_s} : R(x_{i_1},\dots,x_{i_s})$.

Частный вид этого оператора:

$F(x_1,\dots,x_n)$, где $x_i : Q(x_i)$

имеет значением функцию

$G(x_1,\dots,x_{i-1},x_{i+1},\dots,x_n) = F(x_1,\dots,x_{i-1},a,x_{i+1},\dots,x_n)$,

где a есть значение оператора первый в $Q(x_i)$. Применяя такой оператор s раз, получим общий вид.

4.4. Операторные выражения.

4.4.I. Операторы, описанные в 4.3, являются операторными выражениями.

4.4.2. Если A -терм, зависящий от предметной переменной x, G - формула, в которую x входит свободно, то

$\min$ A по $x : G(x)$,
$\max$ A по $x : G(x)$,
сумма A по $x : G(x)$,
произведение A по $x : G(x)$,
card $G(x)$ по x,
номер x в $G(x)$,
A, где $x : G(x)$,
первый в $G(x)$ -

операторные выражения, не зависящие от переменной x.

4.4.3. других операторных выражений нет.

4.4.4. Значение операторного выражения вычисляется согласно пунктам 4.3.I - 4.3.5 после того, как уже вычислены значения входящих в него термов и формул.

5. О п р е д е л е н и е .

Определение - основная конструкция КОП-языка. Символом определения является символ $\doteq$. Программа, написанная на КОП-языке, состоит из последовательности определений, разделенных между собой точкой с запятой. В конце последнего определения ставится точка. Определение используется для присвоения предикатным и функциональным переменным конкретных значений. Имеются четыре типа определений.

5.I. Элементарное определение.

5.I.I. Если R-n -арная предикатная переменная, A_i -предметные переменные, $i = 1,\dots,n$, G -формула со свободными переменными A_i, то

$R(A_1,\dots,A_n) \doteq G(A_1,\dots,A_n)$

есть элементарное определение предикатной переменной R.

5.I.2. Если F-n -местная функциональная переменная, A_i - предметные переменные, $i = 1,\dots,n$, B - терм или операторное выражение, зависящее лишь от переменных A_i, то

$F(A_1,\dots,A_n) \doteq B(A_1,\dots,A_n)$

есть элементарное определение функциональной переменной F.

В пунктах 5.I.I и 5.I.2 символы R и F не должны входить в соответствующие правые части определений.

5.2. Определение по условию.

Если F- n -местная функциональная переменная, G_j - формула, C_j - терм или операторное выражение, зависящее от n предметных переменных ($j = 1,\dots,k$), то $F \doteq C_1$, если G_1 иначе C_2 ... иначе C_k, если G_k есть определение по условию и $F \doteq C_1$, если G_1 иначе C_2 ... иначе C_{k-1} если G_{k-1} иначе C_k есть определение по условию.

5.3. Определение по итерации.

Пусть имеется n -арный предикат (или функция) H и некоторый унарный предикат $P(N)$ от именной предметной переменной. Определение по итерации n -арного предиката (или функции) T описывается следующим образом:

$T \doteq$ **итерация по** N:
$T_0 \doteq H$;
. . . . . .
$T_{N+1} \doteq \gamma(T_N, H1, H2, \ldots, Ht)$;
до $P(N)$;

Здесь $H1, H2, \ldots, Ht$ -данные, или определенные ранее, или определенные вслед за T_0 предикаты и функции, символ γ означает, что T_{N+1} определяется при помощи одного из определений 5.I - 5.2. T_N $(N>0)$ определяются так же, как T_{N+1}. Результатом определения по итерации является предикат (или функция) $T = T_{\overline{N}}$ для такого минимального $\overline{N}$, что $P(\overline{N}) =$ И.

Определение по итерации допускает определение сразу нескольких предикатов и функций:

$T, \ldots, S \doteq$ **итерация по** N:
$T_0 \doteq H$;
. . . . . .
$S_0 \doteq U$;
. . . . . .
. . . . . .
$T_{N+1} \doteq \gamma(T_N, \ldots, S_N, H1, \ldots, Ht)$;
. . . . . .
$S_{N+1} \doteq \omega(S_N, \ldots, T_{N+1}, U1, \ldots, U_3)$;
до $P(N)$

5.4. Определение через алгоритмический язык.

Определение предикатов и функций $T, \ldots, S$ с помощью программы, написанной на каком-либо алгоритмическом языке, описывается следующим образом:

$T, \ldots, S \doteq$ АЛГОЛ $H, \ldots U$: < программа на АЛГОЛе >;

Здесь $H, \ldots, U$ - **входные массивы информации**, $T, \ldots, S$ - **выходные массивы информации в стандартной форме представления, т.е. в виде отношений и функций,** АЛГОЛ - **признак используемого языка (может быть** КОБОЛ, ФОРТРАН **и т.д.).**

В качестве примера описания на КОП-языке рассмотрим задачу нахождения критического пути в графе.

Каждой вершине графа (рис.2) сопоставляется число, называемое весом. Нужно определить вершины, лежащие на таком пути от начальной вершины A до конечной - B, суммарный вес которого является наибольшим по сравнению с другими путями от A до B.

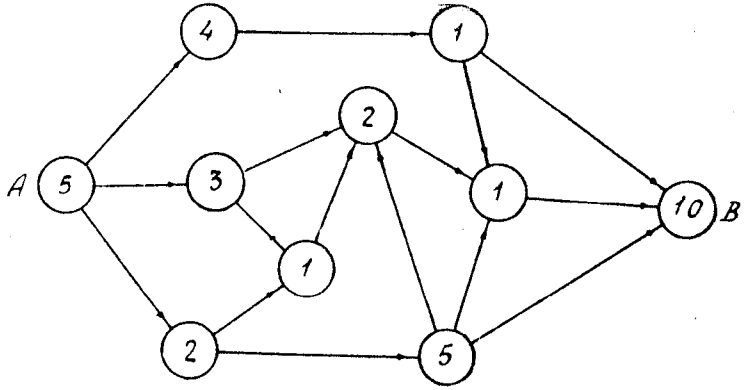

Рис. 2

Исходными данными являются предикат ГРАФ (α, β) и функция ВЕС (α):

ГРАФ (α, β) = И, если из вершины α есть дуга в вершину β и Л - в противном случае;

ВЕС (α) - числовая функция на вершинах графа.

При описании используются также следующие вспомогательные предикаты и функции:

КР ПУТЬ (α) = И, если вершина α лежит на критическом пути, и Л в противном случае.

ПУТЬ А (α) - функция на вершинах графа, определяющая максимальное расстояние данной вершины α от начальной вершины А.

ПУТЬ В (α) - функция на вершинах графа, определяющая максимальное расстояние от данной вершины α до конечной вершины В.

Описание на КОП-языке выглядит так.

ПУТЬ А $(\alpha) \doteq$ итерация по N:
ПУТЬ $A_0(\alpha) \doteq$ ВЕС (α);
ПУТЬ $A_{N+1}(\alpha) \doteq$ ВЕС (α) если $\neg\exists\beta$: ГРАФ (β, α)
иначе
$\{$ВЕС (α) + [max ПУТЬ $A_N(\beta)$ по β: ГРАФ (β, α)]$\}$;
до
$\forall\alpha$: [ПУТЬ $A_N(\alpha)$ = ПУТЬ $A_{N+1}(\alpha)$];
ПУТЬ В $(\alpha) \doteq$ итерация по N:
ПУТЬ $B_0(\alpha) \doteq$ ВЕС (α);
ПУТЬ $B_{N+1}(\alpha) \doteq$ ВЕС (α), если $\neg\exists\beta$: ГРАФ (α, β)

иначе

{ВЕС (c) + [max ПУТЬ B_N (b) по b : ГРАФ (a, b)]} ;

до

$\forall a$: [ПУТЬ B_N (a) = ПУТЬ B_{N+1} (a)];

ПУТЬ (a) $\doteq$ ПУТЬ А (a) + ПУТЬ В (a) - ВЕС (a);

КР ПУТЬ (a) $\doteq$ $\forall b$:[ПУТЬ (a) $\geq$ ПУТЬ (b)].

В заключение отметим некоторые особенности КОП-языка и его использования.

На КОП-языке описывается система определений результативных объектов задачи через исходные объекты. При этом основным понятием, которое используется в таком описании, является понятие отношения (которому в языке сопоставляется понятие предиката) и функции. В некоторых алгоритмических языках есть возможность использования таких укрупненных объектов, выступающих в качестве операндов, над которыми производятся определенные действия, что позволяет получить довольно простые программы при решении сложных задач. Однако, поскольку в алгоритмических языках должна быть точно указана структура каждого типа операндов, приходится определять ряд операций над этими операндами, которые являются следствием не только логического смысла этих операндов, но и их структуры. Это усложняет программирование, так как в этом случае приходится постоянно следить не только за логическим смыслом преобразований, но и иметь в виду структуру операндов.

При использовании КОП-языка программист описывает только логические взаимосвязи между объектами задачи, не заботясь о том, как объекты языка представляются в вычислительной машине, и освобождается таким образом от большого объема дополнительной работы. Это становится возможным благодаря тому, что при переходе от описания на КОП-языке к описанию на алгоритмическом языке, объекты КОП-языка (имена, числа, отношения и функции) для каждого вычислительного устройства, на котором ведется программирование, можно представить в стандартной форме, и именно выбранная форма определит список операций над этими объектами. При этом, как показал опыт, например, при списочном представлении отношений и функций, таких операций достаточно мало (порядка I5). Эти операции могут быть оформлены в виде стандартных подпрограмм в языке машины или в каком-нибудь алгоритмическом языке.

Таким образом, имея в распоряжении всего лишь эти подпрограммы и компонуя их в порядке, который определяется описанием на КОП-языке, можно получить при сравнительно небольших затратах труда системы программ, решающие довольно сложные задачи.

Указанные операции компоновки стандартных подпрограмм могут быть автоматизированы, в результате будет реализован путь I-2-3-5 (рис.I), что позволит реализовать преимущества, упомянутые в начале статьи.

При представлении отношений и функций в виде матриц, выражениям КОП-языка могут быть сопоставлены параллельные поэлементные операции над матрицами.

n -арные предикаты и функции представляются n -мерными логическими и числовыми матрицами соответственно. Например, выражениям, образованным с помощью логических и функциональных связок

$$\wedge \mid \vee \mid \neg \mid \Leftrightarrow \mid \Rightarrow \mid \oplus \mid + \mid - \mid / \mid \div ,$$

сопоставляются соответствующие параллельные поэлементные операции над матрицами (конъюнкция дизъюнкция, отрицание и т.д. в соответствии с таблицами I-3).

Выражениям, образованным с помощью кванторов $\forall$, $\exists$, сопоставляются параллельные операции конъюнкции и дизъюнкции над элементами логических матриц по координате, соответствующей переменной, связанной квантором.

Таким образом, указанный способ представления объектов КОП-языка приводит к естественному распараллеливанию вычислений.

В принципе на КОП-языке может быть описана вся система программ. Однако в некоторых случаях для описания отдельных частей системы применение какого-либо алгоритмического языка может оказаться более выгодным.

Литература

I. МАЛЬЦЕВ А.И. Алгебраические системы. М., "Наука", I970.

2. ЧЁРЧ А. Введение в математическую логику. М., ИЛ, I960.

Поступила в ред.-изд.отд.
24 ноября I972 г.