По теореме двойственности

$$\sum_{j\in M_\nu} \bar{x}_j c_j + \sum_{j\in M_\nu} \sum_{k\in \Gamma_j \setminus M_\nu} \bar{x}_{jk}\bar{u}_k = \min \sum_{j\in M_\nu} \Big(b_j + \sum_{i\in \Gamma_j^{-1}\setminus M_\nu} \bar{x}_{ij} \Big) u_j \qquad (6)$$

при ограничениях задачи $D(M_\nu)$.

Подставляя в (4), вместо левой части (6), правую, мы получаем

$$\sum_{j\in M_\nu} \Big(b_j + \sum_{i\in \Gamma_j^{-1}\setminus M_\nu} \bar{x}_{ij} \Big) \bar{u}_j > \min \sum_{j\in M_\nu} \Big(b_j + \sum_{i\in \Gamma_j^{-1}\setminus M_\nu} \bar{x}_{ij} \Big) u_j. \qquad (7)$$

Рассмотрим теперь любую игру, в которой M_ν входит в состав второго игрока. Ограничение $\bar{U}$ является оптимальной стратегией второго игрока в этой игре. При этом ограничение стратегии на множестве M_ν не зависит от ее ограничений на других множествах, входящих в состав второго игрока, и входит в функцию выигрыша аддитивно. Поэтому

$$\sum_{j\in M_\nu} \Big(b_j + \sum_{i\in \Gamma_j^{-1}\setminus M_\nu} \bar{x}_{ij} \Big) \bar{u}_j = \min \sum_{j\in M_\nu} \Big(b_j + \sum_{i\in \Gamma_j^{-1}\setminus M_\nu} \bar{x}_{ij} \Big) \bar{u}_j.$$

Полученное равенство противоречит (7), что завершает доказательство.

ЛИТЕРАТУРА

1. Д. Гейл. Теория линейных экономических моделей, М., Изд-во иностр. лит., 1963.
2. К. Берж. Теория графов и ее применения, М., Изд-во иностр. лит., 1962.

Поступила в редакцию
2 X 1966

ЭКОНОМИКА 1968
И МАТЕМАТИЧЕСКИЕ МЕТОДЫ том IV, вып. 4

ОРГАНИЗАЦИЯ БИБЛИОТЕКИ ПРОГРАММ И НЕКОТОРЫЕ ВОПРОСЫ РАСПРЕДЕЛЕНИЯ ПАМЯТИ КОМПИЛИРУЮЩЕЙ СИСТЕМОЙ

У. А. БОРМАН, О. К. ДАУГАВЕТ, Д. В. ИГОЛКИНА, И. В. КЛОКАЧЕВ, В. К. ЛИНИС, Л. А. РУХОВЕЦ, С. А. ХОЗИОСКИЙ

(*Ленинград, Рига*)

Метод автоматизации программирования, известный под названием метода библиотечных программ, заключается в следующем: по некоторой информации о схеме программы автоматическая система «собирает» рабочую программу из готовых частей — библиотечных программ, автоматически распределяя память, связывая библиотечные программы и устанавливая их в памяти машины путем соответствующей переработки адресов массивов программ. Исходя из этого, можно сформулировать следующую задачу: разработать систему формального описания программ, содержащую минимум информации, необходимой для автоматических систем, использующих библиотеку программ. Система (правила) формального описания программ, с одной стороны, определяет входной язык автоматических систем, использующих библиотеку программ, а с другой стороны, определяет сами библиотечные программы, т. е. формальные ограничения на структуру программ.

При разработке принципов организации библиотеки мы исходили из следующих основных положений:

библиотека должна накладывать минимальные ограничения на структуру библиотечной программы;

библиотека должна накладывать минимальные ограничения на структуру связей между библиотечными программами внутри рабочей программы;

библиотека должна допускать наличие в ней программ, использующих другие библиотечные программы, как угодно связанные между собой. При этом библиотека должна быть такова, чтобы при использовании библиотечной программы было не обязательно знать, из каких библиотечных программ состоит данная (рекурсивный принцип);

структура информации о библиотечной программе (о задаче) должна быть такова, чтобы для получения правильной рабочей программы необходимо было задавать лишь определенный минимум информации. Для получения более экономной (относительно использования памяти машины) программы нужно задавать дополнительную информацию;

принципы организации библиотеки должны допускать возможность совершенствования структуры библиотеки; это должно достигаться возможностью введения новых типов информации.

При разработке библиотеки мы предполагали, в основном, возможность использования ее автоматическими системами компилирующего типа.

Статья состоит из трех разделов. В первом разделе описывается структура библиотечных программ, т. е. разрабатывается система формального описания программ, определяющая библиотеку; во втором — приводится

возможный вариант распределения памяти автоматической системой; третий содержит формальное описание информации о программе. Следует отметить, что ни первый раздел, ни третий не представляют собой, каждый в отдельности, полного описания библиотеки.

В работе существенно использованы основные идеи метода библиотечных программ, изложенного в [1].

1. СТРУКТУРА БИБЛИОТЕЧНЫХ ПРОГРАММ

Библиотечные программы. Библиотечная программа рассматривается как совокупность массивов $x_1, x_2, \ldots, x_n$, причем каждый массив должен храниться в подряд идущих ячейках памяти. К этим массивам относятся как массивы команд программы, так и величины, над которыми работает алгоритм программы, т. е. аргументы, результаты, промежуточные (рабочие) величины и т. п. Каждый массив характеризуется длиной, т. е. количеством ячеек, необходимых для его размещения в памяти. Длины массивов могут зависеть от некоторых параметров. Начала массивов в программах обозначаются буквами, которые после распределения памяти заменяются на числа — конкретные адреса ячеек памяти. Буквами же предписывается обозначать выходы из программ, т. е. адреса ячеек массивов других (не определенных заранее) программ, на которые передается управление и в которые засылается возврат. Выходы программы будем также интерпретировать как массивы данной программы, имеющие нулевую длину [1, стр. 165]. Во всех дальнейших описаниях информация о таких массивах и работа с ними ничем не отличаются от обычных массивов.

Будем называть *информацией о программе* ту совокупность формальных описаний массивов программы, которая нужна автоматической системе для распределения памяти и изготовления рабочей программы; совокупность сведений о библиотечной программе, необходимую для смыслового использования, будем называть *инструкцией к программе*. В информациях о библиотечных программах массивы не разделяются по их смысловому назначению (закодированная программа, аргументы, результаты и т. п.). Среди массивов программы будем выделять некоторые массивы, которые так и назовем *выделенными* (к ним относятся главным образом массивы команд программы, а также могут относиться некоторые рабочие массивы, массивы констант и др.). Первоначальное содержание выделенного массива в закодированном виде хранится в информации о программе. При кодировании выделенного массива адреса ячеек, соответствующие адресам данного выделенного массива и других массивов, кодируются в виде $a + n$, где буква a соответствует началу массива, а n — число.

Массивы, адреса которых в явном виде присутствуют в адресах команд выделенного массива, назовем *относящимися к данному выделенному массиву*. Каждый массив может относиться лишь к одному выделенному массиву. Выделенный массив не относится к другому выделенному массиву. Выделенный массив вместе с относящимися к нему массивами назовем *элементарной программой*. Элементарные программы являются теми «кирпичами», из которых строятся библиотечные программы. Программа составляется так, что для ее массивов допускается установка в памяти машины на произвольное место.

Иногда выделенный массив содержит значительную формирующую часть, нужную лишь для настройки программы в соответствии со значениями некоторых параметров. Если эту формирующую часть оформить как самостоятельную элементарную программу, то она получает название *головка*. Если параметры заданы при работе автоматической системы, то в некоторых случаях появляется возможность предварительного «проигры-

вания» головки автоматической системой. В этом случае элементарная программа (головка) не будет включена в рабочую программу.

Библиотечная программа, состоящая из одной или нескольких элементарных, называется *простой*. Для построения библиотечной программы из элементарных следует указать связи между массивами элементарных программ. Под связью понимается указание о том, что два массива следует совместить своими началами или концами, или указание о совмещении начала одного массива с концом другого, или указание о совмещении Δ-й ячейки одного массива с началом или концом другого массива (причем Δ может зависеть от параметров). Связи могут определять совмещение результатов одной элементарной программы с аргументами другой, передачи управления, совмещение массивов рабочих ячеек и т. п.

Библиотечная программа может быть составлена из других библиотечных программ. Такая программа называется *сложной*. Сложная библиотечная программа может иметь или не иметь свои элементарные программы. Для построения сложной библиотечной программы из библиотечных программ и, может быть, своих элементарных программ следует указать связи между массивами этих программ. Для того чтобы при использовании сложной библиотечной программы не нужно было знать, какие библиотечные программы в нее входят, некоторые массивы и параметры входящих библиотечных программ определяются как массивы данной библиотечной программы.

Если библиотечная программа зависит от параметров (т. е. от параметров зависят длины входящих в нее массивов, Δ в связях массивов, или программа имеет головку), то информация о библиотечной программе содержит сведения об этих параметрах, а если значения некоторых или всех параметров закреплены, то содержит и эти значения.

Библиотечная программа может содержать некоторые дополнительные указания, позволяющие автоматической системе более экономно распределить память: указание о возможности совмещать данный массив программы с любыми массивами других программ (признак σ), о ненужности некоторых массивов при некоторых конкретных связях, о возможности выписывать в память элементарную программу один раз при использовании ее в нескольких местах программы (признак q и некоторые другие).

Рабочая программа составляется из библиотечных программ и специально «допрограммированных» частей, оформленных как элементарные программы. Для составления рабочей программы задается *информация о задаче*, которая по структуре совпадает с информацией о библиотечной программе.

Общая структура информации о программе. Каждая библиотечная программа имеет свой библиотечный номер. Информация о библиотечной программе составляется из информаций

$$И_0^k,\ И_1^k,\ И_2,\ И_3,\ И_4,\ И_5.$$

Информация $И_0^k$ содержит k-й выделенный массив библиотечной программы в закодированном виде.

В информации $И_1^k$ содержится информация о всех массивах, относящихся к k-му выделенному. Для каждого массива составляется шкала. Шкала для данного массива (для данной буквы) указывает, в каких командах и адресах выделенного массива содержится данная буква. Для одного массива (для одной буквы) шкал может быть несколько — для отдельных участков выделенного массива. Таким образом, информации $И_0^k$, N_1^k определяют элементарную программу. Элементарная программа не является библиотечной. Номер элементарной программы слагается из номера биб-

лиотечной программы, составной частью которой она является, и порядкового номера выделенного массива.

Вспомогательная информация $И_2$ служит для определения длин массивов, зависящих от параметров.

Информация $И_3$ определяет связи между массивами и состоит из строк, каждая из которых определяет связь между двумя массивами различных элементарных или библиотечных программ. Простая библиотечная программа состоит только из элементарных программ, у нее не может отсутствовать информация $И_0^k$. Сложная библиотечная программа может не иметь своих элементарных программ. В этом случае информации $И_0^k$, $И_1^k$, $И_2$ отсутствуют.

В информации $И_4$ некоторые (нужные по смыслу для использования в информациях других библиотечных программ или в информации о задаче) массивы и параметры входящих библиотечных программ определяются как массивы данной библиотечной программы. Информация $И_4$ может содержать также некоторые дополнительные сведения о массивах, например, о запрещении наложения некоторых массивов, о массивах, не нужных при данных связях, и др.

Информация $И_5$ о параметрах описана ниже в разделе «О задании параметров».

Построение библиотечных программ. Для включения в библиотеку простой программы составляется информация о программе согласно формальному описанию. Библиотека строится таким образом, чтобы программа, использующая другие библиотечные программы, могла быть также включена в библиотеку.

Сначала рассмотрим процесс создания библиотечной программы Q только из других библиотечных программ. По правилам формального описания составляется информация $И_3$ о связях между массивами используемых программ, как того требует алгоритм программы Q. Пусть, например, программа Q составляется из двух библиотечных программ с номерами M и N. Пусть массивы аргументов программы M — $1M$ и $2M$, результатов — $3M$, а у программы $N3N$ — массив аргументов, $1N$ и $2N$ — массивы результатов; $4M$, $4N$, $5N$ — массивы рабочих ячеек этих программ, $5M$ — выход программы M, $6N$ — выход программы N. Входами являются третья команда массива $0M$ и первая команда массива $0N$. Алгоритм состоит в том, что по результатам программы M программа N выдает свои результаты. Связи будут установлены следующие:

$$\alpha\, 0N\, \beta\, 5M$$

$$\alpha\, 3M\, \beta\, 3N$$

$$\alpha\, 4M\, \beta\, 5N$$

(α и β — признаки, определяющие, совмещается ли начало или конец одного массива с началом или концом другого). После того как составлена информация $И_3$, вообще говоря, информация о программе Q составлена. Для того чтобы Q стала библиотечной программой, нужно определить ее массивы и параметры (т. е. массивы и параметры программ, входящих в программу Q, которые по смыслу необходимы для ее использования, нужно назвать массивами и параметрами программы Q). Для этого надо в информации $И_4$ определить массивы данной программы, так как при дальнейшем использовании программы Q мы можем иметь дело лишь с ее массивами и параметрами. Информация $И_4$ для нашего примера будет иметь вид $0M$ (для входов программы Q), $1M$, $2M$ (аргументы Q), $1N$, $2N$ (результаты Q), $6N$ (выход Q). Тем самым им присвоены номера массивов программы Q: $1Q$, $2Q$, ..., $6Q$. Затем надо написать «шапку» информации о програм-

ме, где будет стоять номер Q. Таким образом, информация о программе Q не будет содержать $И_0$, $И_1$, $И_2$, а лишь $И_3$, $И_4$ и, возможно, $И_5$.

Если программа Q составляется из библиотечных программ и программы, не входящей в библиотеку, то эта последняя оформляется как элементарная, т. е. для нее составляются $И_0$, $И_1$ и $И_2$. Дальнейший процесс аналогичен описанному, с той разницей, что первыми массивами программы Q считаются массивы элементарной программы. Элементарных программ может быть и несколько.

Программа Q в свою очередь может включаться в сложные библиотечные программы так же, как и простые программы, т. е. процесс построения библиотеки рекурсивен.

Если при составлении сложной библиотечной программы Q в информации о программе Q приходится использовать одну библиотечную программу N несколько раз, то к номеру программы N при различных ее использованиях присоединяется *номер упоминания*: N_r, N_s, ..., и в дальнейшем программы N_r, N_s, ... считаются различными программами. При распределении памяти различные упоминания программы могут быть и совмещены (см. подраздел «Использование одной библиотечной программы несколько раз»). Если программа N употребляется в различных библиотечных программах, используемых в задаче, то ее различные упоминания не различаются номерами в соответствующих информациях, но автоматическая система различает их.

Информация о задаче. Информация о задаче задается для составления рабочей программы. Рабочая программа составляется из библиотечных программ и специально «допрограммированных» частей, оформленных как элементарные программы.

Информация о задаче по структуре есть то же, что информация о программе, с некоторой дополнительной информацией, определенной в разделе 3. Так же как в информации о программе, любые типы информаций могут отсутствовать.

При составлении информации о задаче сведения о библиотечной программе, используемой в задаче, берутся только из инструкции к данной библиотечной программе. Форма записи информации о задаче фактически является входным языком для автоматических систем, использующих библиотеку.

Единая форма записи информации о задаче и информации о программе представляется удобной, так как форма записи информации о программе также является входным языком для автоматических систем, использующих библиотеку.

О задании параметров. Параметры задаются для библиотечных программ в информации $И_5$ — для каждого упоминания данной программы в отдельности.

Параметры бывают трех типов:

1) параметры для вычисления длин массивов. Они нужны для распределения памяти. Поэтому задаются такие значения параметров, при которых длина оказывается максимальной;

2) параметры для вычисления Δ в связях. Зависимость от параметров удобна, например, для задания различных вырезок из массивов;

3) параметры программы (для работы головок; головки описаны ниже).

Структуру задания параметров, разбор информации о параметрах рассмотрим на примере параметров первого типа. Пусть N — простая библиотечная программа, длины некоторых массивов которой зависят от параметров $p_1, p_2, \ldots, p_n$. В информации о программе N содержится программа для вычисления длин всех этих массивов. Программа составлена стандартным образом: содержит сведения о числе параметров, предполагает задание па-

раметров в порядке p_1, p_2, ... в стандартных ячейках, предполагает, что должны быть заданы все параметры (в противном случае программа не работает); длины выдаются в стандартных ячейках в порядке возрастания номеров массивов. Порядок и вид параметров, которых требует программа, описаны в инструкции, и именно так параметры задаются в информации $И_5$ программы, использующей данную. В информации о простой программе информация $И_5$, естественно, отсутствует, так что информация о параметрах не задается.

Пусть Q — сложная библиотечная программа, использующая другие библиотечные программы M_r и N_s и, возможно, имеющая свои элементарные программы. Если при этом нужно задать значения параметров (всех или некоторых из них) программ M_r и N_s, то они задаются в информации $И_5$ программы Q. Информация $И_5$ записывается в виде строк:

$$M_r$$
$$\varepsilon q_1, \varepsilon q_2, \ldots, \varepsilon q_m,$$
$$N_s$$
$$\varepsilon p_1, \varepsilon p_2, \ldots, \varepsilon p_n.$$

Число элементов каждой строки должно быть равно числу параметров соответствующей программы, независимо от того, сколько параметров задано. Порядок параметров определяется соответствующей инструкцией. Заданные и незаданные параметры различаются признаком ε. Если все параметры какой-нибудь программы не надо задавать, то соответствующая строка в $И_5$ может отсутствовать. Для ненужных параметров (т. е. относящихся только к ненужным массивам) ставится признак заданности, и параметрам придается нулевое значение. Незаданные параметры автоматически становятся параметрами программы вместе с параметрами ее собственных массивов. Порядок параметров программы Q определяется следующим образом. Первыми перечисляются параметры собственных массивов (в том порядке, в каком требует программа вычисления длин массивов), затем следуют незаданные параметры других программ в порядке их упоминания сначала в информации о связях в $И_3$, а затем — в информации $И_4$.

В информации о задаче $И_5$ должны быть заданы все параметры всех используемых программ.

Разбирая все информации о параметрах $И_5$, можно получить все значения параметров для вычисления длин массивов по соответствующим программам, заслать их в ячейки аргументов программ, затем можно проработать эти программы и получить длины массивов, необходимые для распределения памяти.

Аналогично определяются и задаются параметры для вычисления Δ связей и параметры программы, аналогичны и правила составления соответствующей программы.

Для параметров программы (т. е. для головок) отличие состоит в следующем:

1) головка составлена по другим правилам, а именно как элементарная программа с некоторыми ограничениями (см. подраздел «О головках»);

2) компилятор не всегда «прорабатывает» головку, а лишь тогда, когда она стирается;

3) параметры для головки могут не только задаваться, но и вычисляться в каких-либо программах, для чего вводится специальный признак в информации $И_5$;

4) параметры для головок бывают двух типов: абсолютные и начала массивов, для чего введена специальная отметка. Начало массива обозначается его номером.

О головках. Если формирующая часть программы оформлена как головка, т. е. как элементарная программа с некоторыми дополнительными ограничениями, изложенными ниже, то появляется возможность предварительного «проигрывания» головки автоматической системой и стирания ее, т. е. под головку не отводится места в памяти. Элементарная программа может иметь только одну головку. Головка не может иметь головки.

На головку как на элементарную программу накладываются следующие ограничения: признак q головки должен быть равен 1 (см. подраздел «Использование одной библиотечной программы несколько раз»); массив ее аргументов — параметры программы — должен быть ее первым массивом (только один массив аргументов); обращение к головке не должно формироваться; головка не должна портить своих исходных данных; ее массивы не должны иметь фиксаций; головка устроена как подпрограмма. Если элементарная программа составлена как головка, то в информации $И_0{}^k$ ставится специальный признак g головки.

При разборе информации автоматическая система засылает заданные значения параметров в массив аргументов головки. Если оказывается возможным головку стереть (например, в случае, когда все параметры заданы), то автоматическая система «проигрывает» и стирает ее. Стирание головки заключается в стирании всех строк связей, в которых участвуют массивы головки, т. е. головка стирается как элементарная программа. Если головка стирается, то стираются и обращения к ней из массивов других библиотечных программ.

2. РАСПРЕДЕЛЕНИЕ ПАМЯТИ АВТОМАТИЧЕСКОЙ СИСТЕМОЙ

Совмещение массивов при распределении памяти. Для возможности экономного распределения памяти у каждого массива вводится специальный признак σ в информации о массиве (в информации $И_0$). Значение $\sigma = 1$ означает разрешение совмещения данного массива с любым другим. Это разрешение не распространяется на массивы, упомянутые в одной строке информации о запрещении наложения (в информации $И_4$). Информация о запрещении наложения вводится специально для того, чтобы два или несколько массивов, могущие быть совмещенными с любыми другими массивами, но не друг с другом, могли быть снабжены признаком $\sigma = 1$. Значение $\sigma = 0$ у массива в информации $И_0$ означает запрещение наложения этого массива на любой другой массив той же элементарной программы, за исключением, разумеется, наложения массивов, осуществляемого связями.

Рассмотрим некоторый массив A некоторой элементарной программы библиотечной программы N, который снабжен признаком $\sigma = 0$ в информации $И_0$. В информации $И_4$ библиотечной программы N, в которую входит элементарная программа, содержащая массив A, массиву A может быть присвоен признак $\sigma = 1$. Это означает, что массив A может быть совмещен с любым массивом, кроме массивов элементарной программы, в которую он входит. Как и выше, это разрешение совмещения не распространяется на массивы, упомянутые в одной строке информации о запрещении наложения. Если в информации $И_4$ программы N массиву A не присвоен признак $\sigma = 1$, то массив A не может быть совмещен ни с какими массивами элементарных и библиотечных программ, входящих в данную библиотечную программу N. Пусть данная библиотечная программа N входит в некоторую другую программу Q и наш массив A программы N определен в информации $И_4$ программы Q как массив программы Q. В информации $И_4$ программы Q массиву A может быть приписан признак $\sigma = 1$. Если при этом в информации $И_4$ программы N массиву A не был присвоен

признак $\sigma = 1$, то это означает, что массив A может быть совмещен с любым массивом любой программы, не входящей в данное упоминание библиотечной программы N, и не может быть совмещен ни с одним массивом данного упоминания программы N (см. подраздел «Использование одной библиотечной программы несколько раз»). Если же в программе Q нет присвоения массиву A признака $\sigma = 1$, то массив A не может быть совмещен ни с одним массивом элементарных и библиотечных программ, входящих в программу Q. Если массив A определен в информации $И_4$ программы Q как ее массив, то при вхождении программы Q в еще более «крупную» программу в последней этому массиву может быть присвоен признак $\sigma = 1$, действие которого распространяется на новом уровне, т. е. вне данного упоминания программы Q, и т. д.

В этой работе не исследуются варианты распределения памяти на предмет выбора оптимального распределения памяти. Покажем, что задача распределения памяти между массивами библиотечных программ сводится к случаю распределения памяти между массивами, рассматриваемому В. В. Мартынюком [2].

Пусть имеется некоторая информация о задаче, в которой в явном виде используются несколько библиотечных программ. Каждая из этих библиотечных программ в свою очередь может включать некоторые другие библиотечные программы и т. д. Можно структуру всей программы представить в виде ориентированного графа, вершинами которого являются различные библиотечные программы, а стрелки соответствуют «вложенности» программ, т. е. если программа N использует программу M, то вершины N и M соединены ребром, направленным стрелкой к M. При этом если одна библиотечная программа используется несколько раз в данной библиотечной программе или в нескольких библиотечных программах, то будем все ее различные упоминания рассматривать как разные программы и изображать разными вершинами. Назовем этот граф *графом структуры программы*. Очевидно, граф структуры программы представляет собой дерево, основанием которого является программа задачи, а все ветви оканчиваются элементарными программами. Будем называть программу M *подчиненной* программе N, если в графе структуры существует путь от вершины N к вершине M, и неподчиненной — в противном случае.

Покажем, как может быть построена матрица T несовместимости элементов массивов (см. [2]). Будем считать, что выписаны в любом порядке все строки запрещения наложения массивов и все строки информаций о связях массивов (информации $И_3$) в терминах массивов элементарных программ. Массивы, принадлежащие элементарным программам, изображаемым различными вершинами, считаются *различными*. Все массивы сгруппируем по принципу связанности. Будем называть два массива A и B связанными, если существует строка информации $И_3$, которая определяет связь массивов A и B, или если существует третий массив C, связанный с каждым из массивов A и B. В одну группу связанных массивов отнесем все массивы, попарно связанные между собой. Очевидно, все строки информаций о связях также разбиваются на соответствующие группы. Отдельные группы составляют массивы, которые не участвуют в связях.

Значения элементов матрицы T определим следующим образом. Если два элемента i и j каких-то массивов окажутся несовместимыми, то соответствующим двум элементам t_{ij} и t_{ji} (в силу симметричности) матрицы T будем присваивать значение 1, в противном случае — 0. Заполним сначала все элементы матрицы T единицами. Затем рассмотрим все информации $И_0$ и $И_4$ всех входящих библиотечных программ. Если в информации $И_0$ массиву X присвоен признак $\sigma = 1$, то элементам t_{ij} (и t_{ji}) матрицы T при-

сваивается значение 0, если i-я строка (столбец) соответствует элементу массива X, а j-й столбец (строка) соответствует элементу массива, который 1) не является массивом X, 2) не упомянут в одной строке запрещения наложения массивов с массивом X, 3) не связан с массивом X.

Если в информации $И_4$ некоторой библиотечной программы некоторому массиву X, описанному как массив программы M*, входящей в N, присвоен признак $\sigma = 1$, то элементам t_{ij} (и соответственно t_{ji}) присваивается значение 0, если i-я строка (столбец) соответствует элементу массива X, а j-й столбец (строка) соответствует элементу массива, который 1) не является массивом X, 2) не упомянут в одной строке запрещения наложения массивов с массивом X, 3) не связан с массивом X, 4) не является массивом программы M или программы, подчиненной программе M.

Во всех остальных случаях значения элементов матрицы T сохраняются прежними. После просмотра всех информаций $И_0$ и $И_4$ всех входящих библиотечных программ матрица T представляет собой матрицу несовместимости элементов всех массивов, но без учета связей. Совместим теперь все массивы одной группы связанности между собой в соответствии с информациями о связях (заметим, что все параметры, определяющие Δ в связях между массивами и длины массивов в информации о задаче, определены; см. подраздел «О задании параметров»), преобразуя одновременно должным образом матрицу T. Эти совмещения и преобразования матрицы осуществляются следующим образом. Выбирается из выделенной группы связанности любая строка о связи двух массивов; обозначим их X и Y. В матрице T последовательно логически складываются строки, соответствующие совмещаемым связью элементам массивов X и Y. То же делается со столбцами матрицы T. Затем, но это просто для удобства, строки (и столбцы соответственно) переставляются так, чтобы подряд идущим элементам массива соответствовали в матрице T подряд идущие строки и столбцы. Наложенные соответственно связям массивы X и Y представим как один массив Z. Во всех строках информаций о связях рассматриваемой группы связанности, в которые входят массив X или массив Y, заменим массив X или Y на массив Z, изменив должным образом Δ в связях. Строка о связи массивов X и Y вычеркивается. Затем выбирается любая строка о связях двух массивов и проводятся совершенно аналогичные преобразования матрицы T и строк о связях. Процесс продолжается до тех пор, пока все строки информации о связях данной группы связанности не будут вычеркнуты.

Вместо нескольких массивов, образующих одну группу связанности, в результате описанного процесса получается один массив x_1 некоторой длины k_1. Применяя описанные преобразования к остальным группам связанности, получим m массивов x_j с длинами k_j, для которых одновременно получена в окончательном виде матрица T. Очевидно, для любой пары массивов x_i и x_j из матрицы T могут быть получены множества s_{ij} (и s_{ji} соответственно), рассматриваемые В. В. Мартынюком. Таким образом, получена полная информация для применения алгоритмов распределения памяти между массивами по В. В. Мартынюку.

Использование одной библиотечной программы несколько раз. Как было сказано выше, при составлении сложной библиотечной программы Q одна и та же библиотечная программа N может быть использована в ней несколько раз. Номера библиотечной программы N, используемые в информации о программе Q, имеют вид N_k, где k — номер упоминания. Для того чтобы различить одинаковые упоминания одной и той же программы в ин-

* Программа M может быть как другой библиотечной программой, так и элементарной программой программы N.

формациях различных программ, нет необходимости в дополнительной информации. Автоматическая система может присоединить к номеру программы N_k номер той библиотечной программы Q, которая использует программу N_k. Автоматическая система разбирает все информации, сводит их к информациям об элементарных программах с учетом различных упоминаний.

Выясним, можно ли одну и ту же элементарную программу, упомянутую дважды, использовать, выписав однократно.

Библиотека предусматривает возможность задания специального признака q возможности совмещения разных упоминаний одной и той же программы. Признак q относится к элементарной программе и ставится в информации $И_0$. Он может принимать два значения — 0 и 1. При этом $q = 0$ означает запрещение совмещать разные упоминания данной элементарной программы, $q = 1$ означает, что возможность совмещения разных упоминаний для каждой конкретной задачи может быть установлена по информации о задаче. Под совмещением элементарных программ понимается отождествление всех соответствующих массивов.

Каждая составленная библиотечная программа имеет признак $q = 0$, независимо от значений признака q у входящих в нее программ. Но при конкретном использовании библиотечной программы в другой программе признаку q может быть присвоено значение 1 в информации $И_4$ программы, использующей данную. Присвоение признаку q упоминания N_k библиотечной программы N значения 1 означает, что возможность совмещения элементарных программ, подчиненных N_k, с упоминаниями соответствующих элементарных программ, не подчиненных N_k, может быть установлена по информации о задаче.

В подразделе «Совмещение массивов при распределении памяти» рассматривается вопрос о распределении памяти без учета возможности совмещений различных упоминаний одной элементарной программы. Рассмотрим теперь задачу распределения памяти с учетом возможности таких совмещений.

Образуем пары упоминаний элементарных программ, подлежащие исследованию на возможность совмещения. Два упоминания M_r и M_s одной элементарной программы M образуют пару, если 1) в информации $И_0^{(k)}$ стоит признак $q = 1$ или 2) стоит признак $q = 0$, но признакам q программ M_r и M_s (или двух программ, которые подчиняют соответственно M_r и M_s в графе структуры программы) присвоены значения 1 в программах, не подчиненных друг другу.

Пусть на предмет исследования возможности совмещения выделено n пар элементарных программ. Среди этих пар могут быть, в частности, такие, совмещение которых противоречит связям или строкам запрещения наложения массивов. Возможность совмещения элементарных программ одной пары может зависеть от того, предполагается ли совмещать программы каких-то других пар. Поэтому вопрос о совмещении программ, образующих пары, надо решать в совокупности.

Рассмотрим n-мерный вектор совмещения пар $A = \{A_i\}_{i=1}^{n}$, компоненты которого могут принимать два значения — 0 и 1. $A_i = 1$ означает, что программы i-й пары должны быть совмещены, $A_i = 0$ — что программы i-й пары не совмещаются. Вектор A, рассматриваемый как n-значное двоичное число, может принимать 2^n значений. В соответствии с каждым значением вектора A можно преобразовать матрицу несовместимости T и попытаться распределить память. Однако не все значения вектора A будут допустимыми: некоторые совмещения программ, входящих в пары, могут противоречить связям или несовмещению других пар. Предлагается алгоритм перебора непротиворечивых значений вектора A, начиная с за-

ведомо непротиворечивого значения 00, . . . , 0. Для каждого непротиворечивого значения из матрицы T порядка m получается матрица $T1$ порядка $m1 \leqslant m$ совмещением элементов массивов тех элементарных программ, которые должны быть совмещены. По матрице $T1$ производится распределение памяти. Если распределение памяти возможно, то на этом процесс заканчивается. Если невозможно — выбираем следующее непротиворечивое значение вектора A.

Алгоритм распределения памяти с учетом совмещения программ. Здесь приводится описание одного алгоритма перебора непротиворечивых значений вектора A на языке АЛГОЛ. В записи алгоритма используется русский алфавит. В алгоритме используются три процедуры, не описанные на языке АЛГОЛ. Заголовки их следующие:

1. *procedure* подготовка матрицы несовместимости (T, $T1$, m, $m1$, A);
array, T, $T1$, A; *integer* m, $m1$.

Эта процедура по вектору A и матрице несовместимости T порядка m строит матрицу $T1$ и вычисляет ее порядок $m1$. Метод построения $T1$ аналогичен методу преобразования при совмещении массивов одной группы связанности (см. подраздел «Совмещение массивов при распределении памяти»).

2. *procedure* распределение памяти ($T1$, $m1$, $b1$);
array $T1$; *integer* $m1$; *Boolean* $b1$;

Эта процедура реализует алгоритм распределения памяти В. В. Мартынюка по матрице несовместимости $T1$ порядка $m1$. Если распределение памяти оказалось возможным, переменной $b1$ присваивается значение *true*, в противном случае — *false*.

3. *procedure* исследование вектора A (A, $A1$, $b2$);
array A, $A1$; *Boolean* $b2$.

Эта процедура проводит исследование, какие еще пары должны быть совмещены при совмещении пар в соответствии с вектором A, и производит соответствующую корректировку вектора A. Процедура выдает специальный сигнал, если при совмещениях, определяемых вектором A, возникает какое-либо противоречие. Делает все это процедура следующим образом. Переменной $b2$ присваивается значение *true*. Пусть l-я компонента вектора A содержит 1 и пусть l-я пара состоит из программ M_r и M_s. Совмещение элементарных программ M_r и M_s означает совмещение всех их соответствующих массивов: iM_r и iM_s. Если два соответствующих массива iM_r и iM_s попали в одну группу связанности, т. е. в один массив x_j (см. подраздел «Совмещение массивов при распределении памяти»), и не совпадают, то совмещение невозможно, переменной $b2$ присваивается значение *false*, продолжается совмещение других массивов программ M_r и M_s. Если массивы iM_r и iM_s попали в разные массивы x_l и x_k, то при совмещении iM_r и iM_s происходит полное или частичное совмещение массивов x_l и x_k. При этом могут совместиться своими началами какие-то два массива, принадлежащие элементарным программам другой пары. В этом случае к соответствующему разряду вектора A по правилу логического сложения добавляется единица.

Переменной $b2$ присваивается значение *false*, если при совмещении массивов x_l и x_k совмещаются элементы каких-то массивов, несовместимые в силу матрицы T (кроме совмещения элементов массивов, совмещаемых началами и принадлежащих элементарным программам другой пары, для которой $A1[j] = 0$). Аналогичный процесс производится со всеми единицами вектора A, как исходными, так и получившимися во время работы процедуры.

procedure перебор значений $A(T, m, A, n)$;
array T, A; *integer* m, n; *comment*.

Процедура «перебор значений A» имеет выход на нелокализованную метку «распределение памяти невозможно».

T — матрица несовместимости порядка m, A — вектор совмещения пар размера n;

```
begin  array T1 [1 : m; 1 : m];
       array A1 [1 : n];
       Boolean b1, b2; integer i, k, j;
       i: = 0, k: = 0;
       for j: = 1 step 1 until n do
       A1(j): = A(j): = 0;
l1:    подготовка матрицы несовместимости (T, T1, m, m1, A); распределение памяти (T1, m1, b1);
       if  b1 then go to end;
l2:    if  A1[i] ≠ 0 then go to l3;
       k: = k + 1; A[i]: = 1; go to l4;
l3:    i: = i + 1;
       if  i = n + 1 then go to l5 else go to l2;
l4:    исследование вектора A(A, A1, b2);
         for j: = 1 step 1 until n do
       if  A[j] = 1 then
           begin if A1[j] = 0 then A1[j]: = k end;
       if b2 then go to l1;
       for j: = 1 step 1 until n do
       if  A1[j] = k then A[j]: = 0; go to l3;
l5:    for j: = 1 step 1 until n do
       if  A1[j] = k − 1 then
           begin if A[j] = 0 then
           begin k: = k − 1;
       if  k = 0 then go to распределение памяти невозможно
           else go to l5
       end
           else go to l6
     end;
l6:    for j: = 1 step 1 until n do
       if  A1[j] = k then begin i: = j; go to l7 end;
l7:    for j: = 1 step 1 until n do
         begin if A1[j] ⩾ k then A1[j]: = A[j]: = 0;
       if  A1[j] = k − 1 then A[j]: = 0
       end;
         k: = k − 1; go to l2;
       end:
       end;
```

Стирание головок. При разборе информации о задаче для каждой элементарной программы, являющейся головкой, определяется, имеет ли она параметры, которые вычисляются, или все значения ее параметров заданы — через посредство информаций $И_5$.

Вопрос о возможности проигрывания головки компилирующей системой и стирания ее вместе со всеми ее массивами может быть решен только вместе с вопросом о совмещении программ. Предлагается следующий метод решения этого вопроса.

Если головка не входит в пары программ, подлежащие исследованию на возможность совмещения, то вопрос о ее стирании решается без исследования: головка стирается, если все ее параметры заданы, и не стирается, если хоть один из ее параметров вычисляется.

Все дальнейшее описание относится к случаю, когда головка входит в пары. Пусть имеется n головок, входящих в пары, все параметры которых заданы. Введем в рассмотрение вектор B порядка n. Обозначим через B_i его i-ю компоненту, относящуюся к i-й головке. $B_i = 1$ означает, что головка может быть стерта, а $B_i = 0$ — что головку стирать нельзя. Требуется определить значение B.

Предлагается следующий порядок перебора всех возможных вариантов стирания головок. Всем компонентам вектора B придаются последовательно всевозможные значения. Составляется соответствующий вектор A (см. подраздел «Использование одной библиотечной программы несколько раз») совмещения пар всех программ, включая нестертые головки. При установлении непротиворечивости значения вектора A (т. е. варианта совмещения пар программ; см. алгоритм в предыдущем разделе) к указанным ранее противоречиям должны быть добавлены еще два: 1) совмещаются головки, у которых не совпадают соответствующие заданные параметры; 2) совмещаются программы, у которых относящиеся к ним головки стерты, а их параметры не совпадают.

Если все значения вектора A исчерпаны, выбирается следующий вариант вектора B.

3. ИНФОРМАЦИЯ О ПРОГРАММЕ И ЗАДАЧЕ

В разделе 1 была описана структура библиотечных программ. При описании структуры обосновывалась необходимость или полезность введения тех или иных типов информации и показывался их смысл. Настоящий раздел содержит вариант формального описания информации о библиотечной программе. Как было сказано в разделе 1, для получения правильной программы необходимо задавать лишь определенный минимум информации, поэтому информация о конкретной библиотечной программе может не содержать некоторые из разделов информации, приводимых ниже. Заметим еще, что формальное описание информации приведено схематично, здесь опущены некоторые необходимые очевидные сведения, как, например, длины и контрольные суммы информаций, признаки типа информации, конца информации и т. п.

Информация о библиотечной программе. 1) Номер программы — Q.

2) $И_0^k$ $(k = 0, 1, \ldots)$ — информация о k-м выделенном массиве: g — признак головки; q — признак возможности совмещения программы kQ в разных упоминаниях; закодированный k-й выделенный массив.

Буквы в условных адресах кодируются числами. Допустимые значения для кодов букв определяются разрядностью адреса.

3) $И_1^k$ $(k = 0, 1 \ldots)$ — информация для установки k-го выделенного массива.

а) Информация о массивах: код буквы или признак отсутствия шкалы; длина массива или признак вычисления длины по программе; признак выделенного массива; σ — признак, определяющий, играет ли роль содержание массива до и после работы программы вне зависимости от ее входов и выходов.

б) Информация о шкалах: начало обрабатываемой части (по букве); длина обрабатываемой части (по букве).

в) Шкалы.

4) $И_2$ — информация о длинах массивов: программа вычисления длин массивов.

5) $И_3$ — информация о связях массивов.

а) Строки, определяющие связи, вида

$$\alpha\ iN_r\ \beta\ jM_s\ \Delta,$$

где iN_r — i-й массив r-го упоминания программы N; jM_s — j-й массив s-го упоминания программы M; α, β — признаки, определяющие, связываются начало или конец массива, перед которым стоит признак; Δ — относительный адрес начала (конца) массива iN_r относительно начала (конца) массива jM_s или признак вычисления Δ по программе.

б) Программа для вычисления Δ.

Замечание. Если в $И_3$ указан номер массива данной программы, то номер упоминания ее самой не указывается.

6) $И_4$ — дополнительная информация в случае сложной программы.

а) Информация об определении массивов данной программы, являющихся массивами других библиотечных программ. Информация содержит список номеров массивов других библиотечных программ, определяемых как массивы данной. Номера записываются в виде nK_s, где K — номер входящей библиотечной программы; s — номер упоминания программы K в информации $И_3$ программы Q; n — номер массива программы K. Перечисленные массивы могут быть снабжены признаком $\sigma = 1$.

б) Информация о присвоении признака $q = 1$. Информация содержит список номеров программ K_s, которым присваивается $q = 1$.

в) Информация о ненужных массивах. Информация содержит список массивов других библиотечных программ iM_s, не нужных при данных связях.

Замечание. Если ненужный массив — выделенный, то он не нужен вместе со своими массивами.

г) Информация о запрещении наложения массивов. Информация состоит из строк, содержащих список номеров массивов данной библиотечной программы $i, j, \ldots, l$, которые не могут при распределении памяти налагаться друг на друга вне зависимости от значения σ.

Замечание. Массивы библиотечной программы нумеруются последовательными натуральными числами, начиная с 0, следующим образом. Сначала нумеруются выделенные массивы; затем последовательно нумеруются массивы, относящиеся к выделенным, в порядке расположения их информации о массивах, при этом первыми нумеруются все массивы, относящиеся к нулевому выделенному массиву, затем к первому, и т. д. Далее нумеруются массивы, определенные в информации $И_4$ данной программы, в порядке перечисления номеров массивов.

7) $И_5$ — информация о параметрах. Информация состоит из строк вида

$$N_r$$
$$\varepsilon p_1,\ \varepsilon p_2,\ \ldots,\ \varepsilon p_{n_1}$$
$$\varepsilon q_1,\ \varepsilon q_2,\ \ldots,\ \varepsilon q_{n_2}$$
$$\varepsilon\delta\omega t_1,\ \varepsilon\delta\omega t_2,\ \ldots,\ \varepsilon\delta\omega t_{n_3},$$

где Nr — r-е упоминание программы N; p_i — параметры для вычисления длин; q_i — параметры для вычисления Δ; t_k — параметры программы (для головок); ε — признак того, определен ли параметр; δ — признак того, вычисляется ли параметр; ω — признак типа параметра.

Замечания. а) В информации о задаче все параметры должны быть определены.

б) Параметры программы в информации $И_5$ программы, использующей данную, задаются в следующем порядке. Для библиотечной программы, не опирающейся на другие программы, параметры задаются в той последовательности, какую требует соответствующая программа вычисления длин (или Δ) или головка. Для программ, опирающихся на другие библиотечные, сначала задаются параметры для собственных массивов данной программы, затем — незаданные параметры программ, на которые она опирается, в порядке их упоминания в связях, а затем в $И_4$.

в) Параметр считается определенным, если он либо вычисляется, либо задается его значение.

г) Параметры программы (для головок) могут быть двух типов: параметры, которые определяются абсолютными числовыми значениями, и параметры, которые являются началом массива. В первом случае задается число, во втором случае — номер массива.

Информация о задаче. Информация о задаче состоит из разделов:

1) Информация о рабочей программе типа информации о библиотечной программе;

2) Дополнительная информация для компилятора:

а) информация о том, в каком виде должна быть выдана программа (перфокарты, магнитная лента и т. д.);

б) информация о входе.

ЛИТЕРАТУРА

1. Г. И. Кожухин, Н. М. Нагорный, И. В. Поттосин. Принципы организации и использования библиотеки программ. В сб. Вычислительная математика, № 7. М., Изд-во АН СССР, 1961.
2. В. В. Мартынюк. Об экономном распределении памяти. Ж. вычисл. матем. и матем. физ., 1962, т. 2, № 3.
3. О. К. Даугавет, И. В. Клокачев, А. А. Руховец. Организация библиотеки программ ассоциации БЭСМ. Электронные вычислительные машины и решение инженерных задач на них. Материалы Всероссийского научно-технического совещания по механике и автоматике инженерных и управленческих работ в промышленности и строительстве. М., 1963.

Поступила в редакцию
18 X 1966