УДК 004.43+004.422.636

ИНТЕРВАЛЫ ИНДЕКСОВ В ЛЯПАСЕ[1]

А. А. Песняк, Д. А. Стефанцов

Национальный исследовательский Томский государственный университет, г. Томск, Россия

Разработаны новые типы данных для работы с частями комплекса. Добавлены обозначения для этих типов данных, операции над ними, расположение этих объектов в памяти программы. Приведены примеры использования реализации абстрактных типов данных в языке ЛЯПАС.

Ключевые слова: *язык программирования ЛЯПАС, операции над комплексами, абстрактные типы данных.*

DOI 10.17223/20710410/39/9

INDEX INTERVALS IN LYAPAS

A. A. Pesnyak, D. A. Stefantsov

National Research Tomsk State University, Tomsk, Russia

E-mail: a.pesnyak92@mail.ru, d.a.stephantsov@gmail.com

The new data type called index interval is proposed for LYaPAS along with the operations defined on members of that type. It enables isolated work on parts of members of another LYaPAS data type, namely the contiguous sequence of homogeneous elements called complex. The syntax for describing the members of the new data type is introduced along with that for the operations on its members. As the implementation detail, the memory layout is discussed. Finally, implementations of some well-known algorithms using index intervals in LYaPAS are given, such as Karatsuba multiplication, quick sort, and the lookup operation in a binary search tree.

Keywords: *LYaPAS programming language, operations on complexes, abstract data type.*

Введение

В языке ЛЯПАС используются только такие типы данных, как комплексы и переменные [1]. В существующей реализации языка ЛЯПАС проблематичным является процесс задания поддиапазонов комплекса. В некоторых типах задач оптимальной является возможность выделения только части комплекса и последующей работы с ней. Например, такими задачами являются сортировка Хоара, умножение чисел методом Карацубы. Вместо того чтобы подавать весь комплекс на вход функции, достаточно отдать только его часть, необходимую для сортировки или умножения на данной итерации. Эта проблема особенно остро возникает при реализации абстрактных типов данных, использующих указатели. Необходимость их использования связана с применением абстрактных типов данных, таких, как списки и деревья, в множестве существующих программ и форматов хранения данных, например в файловой системе

[1]Работа поддержана грантом РФФИ, проект № 17-01-00354.

ext4. Чтение данных с томов ext4 является одной из задач на первых этапах развития ОС ЛЯПАС.

Альтернативой указателям являются индексы элементов комплекса, хранимые в элементах этого комплекса, что не отменяет некоторые проблемы. Так, например, проблемой безопасности является явная манипуляция индексами комплекса. Имея доступ на чтение и запись к любой ячейке комплекса, программист имеет возможность изменять индексы, тем самым получая доступ к другим участкам комплекса или даже меняя логику работы программы. Невозможно ограничить доступ к части комплекса и нецелесообразно использовать эту технику в символьных комплексах, так как размер их элементов составляет один байт, а следовательно, значение индекса не может быть больше 255, что слишком мало для практических задач.

В данной работе рассматривается создание новых типов данных языка ЛЯПАС — *интервалов* и *комплексов ссылок*. Описаны операции над ними, а также представление этих типов данных в памяти прикладной программы. Приведены примеры применения интервалов и комплексов ссылок.

1. Интервалы в ЛЯПАСе

1.1. Определения, правила, обозначения

Комплекс — структура данных в виде набора равных по размеру элементов, расположенных в памяти непосредственно друг за другом. Комплекс имеет потенциальный и текущий размеры, которые называются *ёмкостью* и *мощностью* соответственно. Обращение к элементу за пределами комплекса приводит к аварийному завершению программы.

Интервал — множество элементов логического или символьного комплекса, ограниченное двумя элементами комплекса — началом и концом интервала. У интервала, как и у комплекса, есть такие параметры, как ёмкость и мощность.

Таблица идентификаторов — служебная таблица, в которой хранится информация об интервалах заданного комплекса: индекс начала интервала, индекс конца интервала, мощность и ёмкость интервала, комплекс ссылок интервала.

Идентификатор (*ссылка*) — четырёхбайтовое значение, описывающее адрес некоторой строки таблицы идентификаторов.

Комплекс ссылок — комплекс, элементами которого являются идентификаторы.

Каждому интервалу соответствует свой комплекс ссылок, элементы этого комплекса являются идентификаторами интервалов, на которые ссылается данный интервал. К элементам комплекса ссылок запрещается применение операций присваивания и считывания.

Во всех приведённых ниже наименованиях символы i, j, k около комплексов или интервалов являются константами и составляют часть имени комплекса (интервала).

Для введённых определений используются следующие обозначения:

N_j — интервал с номером j;

W_j — мощность интервала N_j;

Y_j — ёмкость интервала N_j;

M_j — комплекс ссылок интервала N_j;

T_j — мощность комплекса ссылок M_j;

V_j — ёмкость комплекса ссылок M_j.

1.2. Операции над интервалами и комплексами ссылок

Создание интервала

`@+(x,y,Li/Nj)`

В результате выполнения данной операции в логическом комплексе L_i из элементов, начиная с элемента с индексом x и заканчивая элементом с индексом $y-1$, будет создан интервал N_j мощности и ёмкости $y-x$. В таблицу идентификаторов добавится соответствующая запись о созданном интервале.

Сохранение идентификатора

`@↓(i,Nj/Mk)`

Операция записывает адрес, соответствующий интервалу N_j в таблице идентификаторов, в элемент по индексу i комплекса ссылок M_k.

Восстановление по идентификатору

`@+(i,Mj/Nk)`

Операция восстанавливает сведения об интервале на основании идентификатора, находящегося в элементе с индексом i комплекса M_j, и ставит им в соответствие псевдоним N_k.

Обнуление идентфикатора

`OMi.j`

В результате выполнения данной команды в комплексе ссылок M_i в элементе с индексом j будет записано `null`-значение.

Переход по идентификатору

`↑(id?Mj.i)q`

Команда выполняет проверку элемента с индексом i комплекса ссылок M_j на соответствие идентификатору: если в таблице идентификаторов по этому адресу есть запись, то выполняется переход на параграф q.

Переход по пустому идентификатору

`↑(null?Mj.i)q`

Операция выполняет проверку элемента с индексом i комплекса ссылок M_j: если этот элемент равен `null` или в таблице идентификаторов по этому адресу отсутствует запись, то выполняется переход на параграф q.

Присваивание

`@+(Nj/Nk)`

В результате выполнения данной операции псевдониму N_k будет соответствовать интервал, соответствующий псевдониму N_j.

2. Использование и реализация

2.1. Примеры решения задач с помощью интервалов

Поддиапазоны

Интервалы позволяют выделять нужные части комплекса для последующей обработки. Это может потребоваться в реализации сортировки Хоара (листинг 1).

```
1  Hoare(N1/N1)
2       Oi  W1-1⇒j>1⇒r  N1r⇒x
```

```
3  §1  ↑(i>j)5
4  §2  ↑(N1i⩾x)3  Δ i  →2
5  §3  ↑(N1j⩽x)4  ▽ j  →3
6  §4  ↑(i⩾j)10  ⇔(N1ij)  →1
7  §5  ↑(i⩾W1)6  @+(i,W1,N1/N2)  *Hoare(N2/N2)
8  §6  j↪10  @+(0,j,N1/N2)  *Hoare(N2/N2)
9  §10  **
```

Листинг 1. Подпрограмма сортировки Хоара

На рис. 1 и в листинге 2 приведён пример реализации умножения больших чисел методом Карацубы [2]. Пусть N_{11}, N_{22} — множители одного размера ($W_{11} = W_{22}$), интервал N_{33} выделен на всей длине логического комплекса L_3 необходимого размера При мощности комплекса меньше 50 умножение выполняется с помощью стандартной операции языка ЛЯПАС-Т [3], так как для таких чисел умножение методом Карацубы работает медленнее. Комплекс L_4 является временным.

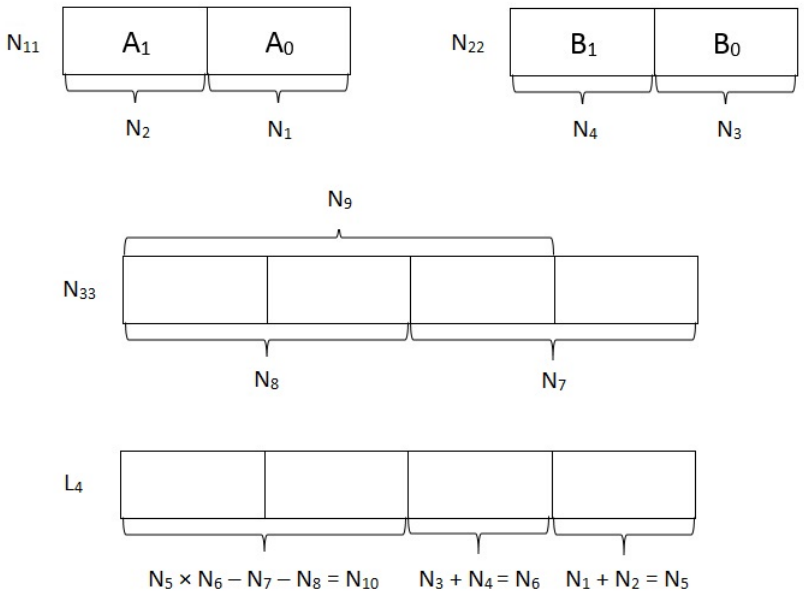

Рис. 1. Расположение интервалов в комплексах для перемножения Карацубы

```
1  Karatsuba(N11,N22/N33)
2      W11>1⇒n<1⇒t<1⇒f  ↑(n<25)1
3      @+(0,n,N11/N1)  @+(n,Q1,N11/N2)
4      @+(0,n,N22/N3)  @+(n,Q2,N22/N4)
5      @+(0,t,N33/N7)  @+(t,f,N33/N8)  @+(n,f,N33/N9)
6      f+4⇒s  @+L4(s)  ***  tmp
7      n+1⇒a<1⇒b
8      @+(0,a,L4/N5)  @+(a,b,L4/N6)  @+(b,s,L4/N10)
9      N1+N2⇒N5  N3+N4⇒N6
10     *Karatsuba(N1,N3/N7)
11     *Karatsuba(N2,N4/N8)
12     *Karatsuba(N5,N6/N10)
```

```
13        N10-N7-N8+N9⇒N9
14        @-L4  →2
15  §1    N11*N22⇒N33
16  §2    **
```

Листинг 2. Подпрограмма умножения Карацубы

Списки

С помощью интервалов можно реализовать списки. В качестве примера выбрано суммирование элементов списка, каждый из которых состоит из некоторого числового значения и ссылки на следующий элемент (рис. 2, листинг 3).

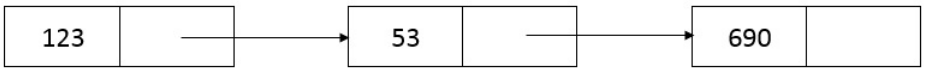

Рис. 2. Пример односвязного списка

```
1  sum_list(N1/s)
2       0s
3  §1   s+N1.0⇒s
4       ↑(null?M1.0)2
5       @+(0,M1/N1)  →1
6  §2  **
```

Листинг 3. Подпрограмма суммирования элементов списка

В данной реализации комплекс ссылок и интервал состоят из одного элемента — ссылки на следующий элемент списка и числового значения соответственно.

Деревья

Интервалы можно использовать для создания такой иерархической структуры данных, как деревья. Приведём пример программы поиска значения в двоичном дереве поиска; пример дерева представлен на рис. 3, его представление в комплексе — на рис. 4.

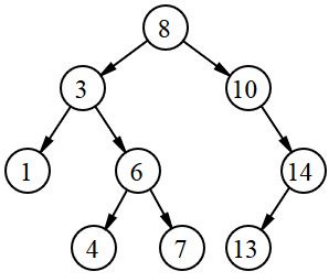

Рис. 3. Пример двоичного дерева поиска

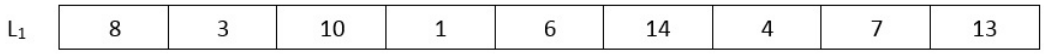

Рис. 4. Расположение дерева в комплексе

Каждому интервалу соответствует комплекс ссылок, содержащий два идентификатора — ссылки на левое и правое поддеревья. Связь комплексов между собой приведена на рис. 5, код программы — в листинге 4.

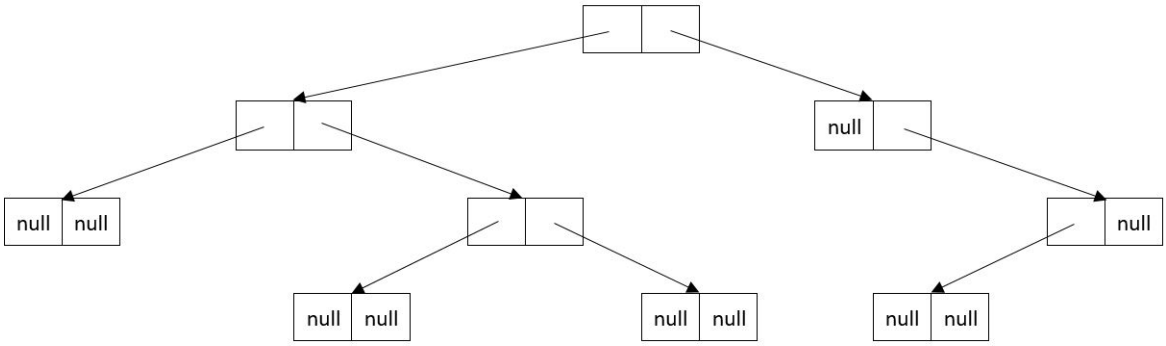

Рис. 5. Связь комплексов ссылок для двоичного дерева поиска

Входными параметрами подпрограммы являются ключ и интервал, соответствующий корню дерева. В случае успешного нахождения ключа в дереве поиска в нулевой элемент комплекса ссылок M_2, соответствующего выходному интервалу N_2, будет помещён идентификатор интервала, содержащего ключ; в противном случае — `null`-значение.

```
 1  search_BSTree(k,N1/N2)
 2      OM2.0
 3  §1  ↑(k<N1.0)2
 4      ↑(k>N1.0)3
 5      ↑(k=N1.0)4
 6  §2  ↑(null?M1.0)5
 7      @+(0,M1/N1) →1
 8  §3  ↑(null?M1.1)5
 9      @+(1,M1/N1) →1
10  §4  @↓(0,N1/M2)
11  §5  **
```

Листинг 4. Подпрограмма поиска ключа в бинарном дереве поиска

2.2. Представление в памяти

Комплексу, на котором определены интервалы, соответствует единственная таблица идентификаторов, имеющая формат табл. 1. Здесь и далее $a_1, a_2, a_3 \ldots$ — некоторые адреса таблицы интервалов при выполнении прикладной программы.

Таблица 1

Адрес	Начало интервала	Конец интервала	Мощность	Ёмкость	Комплекс ссылок
a_1	$i_{1,1}$	$i_{1,2}$	W_1	Y_1	M_1
a_2	$i_{2,1}$	$i_{2,2}$	W_2	Y_2	M_2

Подпрограмме листинга 4 соответствует следующая таблица интервалов (табл. 2):

Таблица 2

Адрес	Начало интервала	Конец интервала	Мощность	Ёмкость	Комплекс ссылок
a_1	0	1	1	1	M_1
a_2	1	2	1	1	M_2
a_3	2	3	1	1	M_3
a_4	3	4	1	1	M_4
a_5	4	5	1	1	M_5
a_6	5	6	1	1	M_6
a_7	6	7	1	1	M_7
a_8	7	8	1	1	M_8
a_9	8	9	1	1	M_9

Взаимосвязь комплексов ссылок для данного примера представлена на рис. 6.

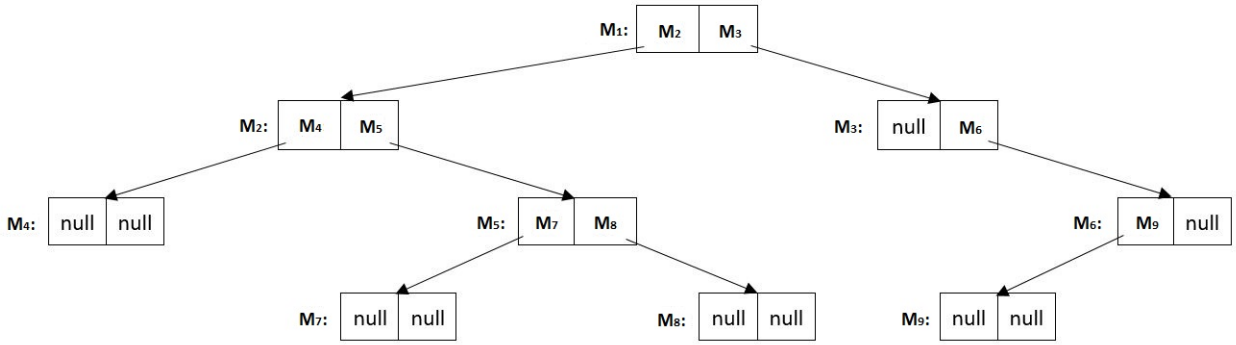

Рис. 6. Связь комплексов ссылок для двоичного дерева поиска

2.3. Дополнительные изменения в языке ЛЯПАС

В некоторых случаях программисту требуется работа со всем комплексом, как с интервалом. Для этого предложен механизм, позволяющий произвести данную операцию — интервал будет создан на всём комплексе и будет содержать все его элементы.

В целях удобства работы с интервалами предлагается ввести следующие дополнительные операции:

Распаковка в переменные

```
@+(i,Nk/x,y,z)
```

Операция производит запись элементов интервала N_k, начиная с элемента с индексом i, в переменные, указанные в качестве выходных параметров. Их количество произвольно и задаётся программистом путём перечисления переменных. В данном примере будут считаны три элемента из интервала N_k и последовательно записаны в переменные x, y, z.

Распаковка в аргументы подпрограммы

```
*func(x,y,@+(i,j,Lk/*)/a)
```

В данном примере созданный интервал над комплексом L_k будет распакован на отдельные входные переменные подпрограммы `func`.

Упаковка из переменных

```
@+(x,y,z/Nk)
```

В результате выполнения данной операции будет создан интервал N_k, содержащий значения всех входных параметров. Как и в операции распаковки в переменные, количество входных переменных произвольно.

Упаковка из выходных значений подпрограммы

```
*proc(x,y/@+(*/Ni))
```

Данная команда создаёт интервал N_i из выходных значений процедуры `proc`.

В целях стандартизации языковых конструкций предлагается заменить операцию создания комплекса `@+Li(n)` [4] на `@+(n/Li)`.

Заключение

В языке ЛЯПАС разработаны новые типы данных, позволяющие работать с частями комплекса — интервалы и комплексы ссылок. Описаны обозначения для этих типов данных, операции над ними, расположение этих объектов в памяти программы. Рассмотрены примеры использования интервалов для описания абстрактных типов данных.

ЛИТЕРАТУРА

1. *Торопов Н. Р.* Язык программирования ЛЯПАС // Прикладная дискретная математика. 2009. № 2. С. 9–25.
2. *Кнут Д.* Искусство программирования. Т. 2. Получисленные алгоритмы. 3-е изд. М.: Изд. дом «Вильямс», 2001. 832 с.
3. *Грибанов А. С., Сибирякова В. А.* Программная реализация операций над большими числами в языке ЛЯПАС-Т // Прикладная дискретная математика. Приложение. 2014. № 7. С. 146–148.
4. *Агибалов Г. П., Липский В. Б., Панкратова И. А.* О криптографическом расширении и его реализации для русского языка программирования // Прикладная дискретная математика. Приложение. 2013. № 3. С. 93–104.

REFERENCES

1. *Toropov N. R.* Yazik programmirovaniya LYaPAS [Programming language LYaPAS]. Prikladnaya Diskretnaya Matematika, 2009, no. 2, pp. 9–25. (in Russian)
2. *Knuth D. E.* The Art of Computer Programming, vol. 2. Seminumerical Algorithms, Third Ed. Reading, Massachusetts, Addison-Wesley, 1997.
3. *Gribanov A. S. and Sibiryakova V. A.* Programmnaya realizatsiya operatsiy nad bolshimi chislami v yazike LYaPAS-T [Software implementation of operations over large numbers in LYaPAS-T]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2014, no. 7, pp. 146–148. (in Russian)
4. *Agibalov G. P., Lipskiy V. B., and Pankratova I. A.* O kriptograficheskom rasshirenii i ego realizatsii dlya russkogo yazyka programmirovaniya [Cryptographic extension and its implementation for Russian programming language]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2013, no. 3, pp. 93–104. (in Russian)