СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
ДУБНА

10 - 7587

Д.Д.Арнаудов

ОРГАНИЗАЦИЯ МУЛЬТИСПИСОЧНЫХ УЗЛОВЫХ СТРУКТУР И ИХ ПРОГРАММНАЯ РЕАЛИЗАЦИЯ НА КОБОЛе

1973

ЛАБОРАТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИНИ И АВТОМАТИЗАЦИИ

10 - 7587

Д.Д.Арнаудов

ОРГАНИЗАЦИЯ МУЛЬТИСПИСОЧНЫХ УЗЛОВЫХ СТРУКТУР И ИХ ПРОГРАММНАЯ РЕАЛИЗАЦИЯ НА КОБОЛе

Арнаудов Д.Д. 10 - 7587

Организация мультисписковых узловых структур и их программная реализация на КОБОЛе

В работе рассмотрены четыре основных вида списочных структур, организация и программирование цепных списков.

Сообщение Объединенного института ядерных исследований
Дубна, 1973

В работе /1/ отмечено, что существуют четыре основные вида списочных структур, рассмотрены организация и программирование алгоритмов цепных списков.

Особенно важное значение при реализации ИПС (информационно-поисковых систем) имеет способ построения мультисписочных узловых структур. Мы постараемся показать, как можно использовать его при организации информации в памяти вычислительной машины.

Предположим, что в ИПС объекты (наименования объектов) с некоторыми общими свойствами (признаками) объединены в список, причем каждый объект может входить не в один, а в несколько списков. При узловом способе все списочные слова, которые представляют один и тот же объект в разных списках, располагаются в памяти машины не зависимо друг от друга, как это имеет место при построении цепных списков, а подряд /2/, образуя так называемый узел. Ясно, что каждый узел является точкой пересечения списков, которые связывают его с узлами других объектов, входящих в те же самые списки, что и данный объект. Так как каждый объект в общем случае может входить в разное количество списков, то число адресов связи, соответственно числу списочных слов в каждом узле, разное. Необходимо отметить, что в узле можно хранить как информацию о данном объекте, так и только адрес, указывающий на место в памяти машины, где хранится нужная информация об объекте.

Рассмотрим один из возможных примеров реализации узлового метода. Мультисписочная структура в данном примере состоит из заголовков списков, число которых равняется числу списков, и из конкретных узлов. На рис. I показана структура одного узла и заголовок некоторого списка.

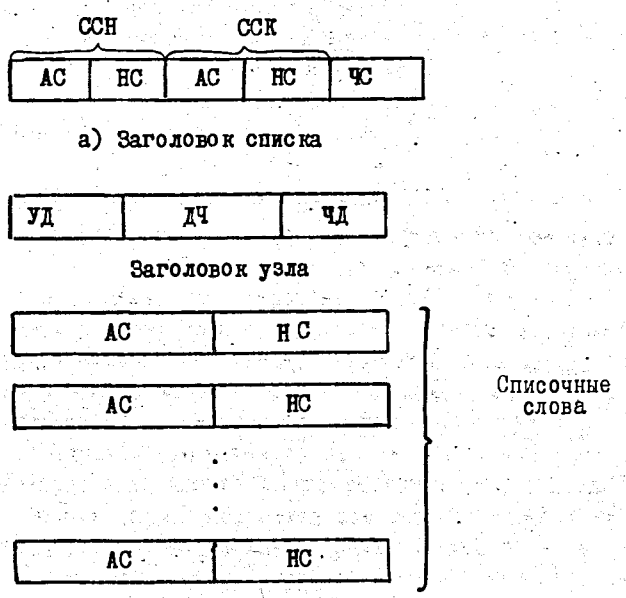

Рис.I.

Как видно на рисунке,формат заголовка списка состоит из начального спискового слова (ССН), конечного списочного слова (ССК) и числа членов списка (ЧС).

Списочное слово начальное позволяет найти узел, относящийся к первому объекту,записанному в данный список, и место списочного слова в узле, принадлежащем списку. Списочное слово начальное состоит из АС и НС, где:

а) АС - адрес связи, указывающий на начало цепного списка объектов, которые имеют данный признак;

б) НС - номер слова, который указывает на место спискового слова в первом узле, входящем в данный список.

ССК выполняет те же функции ,что и списочное слово начальное, но для последнего узла в списке. ССК используется для включения новых сведений в списочную структуру.

Ассоциативный узел состоит из заголовка узла (см. рис.Iб) и списочных слов. Заголовок узла может иметь три величины - УД,ДЧ,ЧД,

а) УД - указатель документа (объекта), является кодом документа ,его порядковым номером или адресом записи объекта;

б) ДЧ - дополнительная информация об объекте;

в) ЧД - число слов в узле.

Каждое списочное слово состоит из :

а) АС - адреса связи;

б) НС - номера слова.

Эти величины имеют то же значение, но уже в пределах данного узла.

Применение узлового метода в ИПС хорошо описано в работах[3,4]. В настоящей работе мы остановимся на программной реализации узловых мультисписочных структур на КОБОЛе.Для работы в реальном масштабе времени с использованием терминальных устройств необходимо предусмотреть расположение узловых структур на диске.Только в этом случае можно легко осуществить произвольный доступ к нужной информации в данном узле.В работе[3] показано,что фактически каждый узел можно разделить на две части и организовать два вида массивов: массивы заголовков узлов и списочных слов. Это дает возможность построить однородные массивы, которые легко вводятся в ЭВМ с помощью входного языка системы КОБОЛ-СДС.

Можно ,конечно ,сделать и иначе: описать узел как некоторое групповое данное и составить массив из таких узлов. Но в этом случае затруднился бы процесс подготовки исходных данных для ввода в машину. Кроме того, что особенно важно, в этом случае трудно организовать узлы переменной длины. В нашем же случае можно менять длину узла, так как он фактически существует в двух отдельных массивах. Разделение узла на две составные части не накладывает никаких ограничений на использование узлового метода в ИПС[4]. Информационная сущность узла сохраняется,и в данном случае узел дает возможность в одной точке сосредоточить всю поисковую информацию о документе , находящемся в нескольких цепных списках.

Итак, ясно, что расположение и организация узловых структур на диске сводится к эффективной организации файлов с прямым доступом. При этом надо иметь в виду, что это файлы с достаточно большим количеством элементов. В данном случае, когда файл становится настолько большим, что невозможно или по крайней мере неэкономично просматривать каждый элемент файла для проверки его соответствия, появляется особенно важная проблема - проблема организации файла. Она порождает технические трудности при разработке системы информационного поиска, связанные с согласованием объема памяти, времени, необходимого для получения ответа и т.д. Поэтому при разработке организационной структуры файла попытаемся достичь приемлемый компромисс между:

I) временем, затрачиваемым на поиск элемента файла;

2) объемом памяти на диске, необходимым для размещения данного файла.

В КОБОЛе имеется возможность организовать файлы на диске с последовательным и произвольным доступом. Для организации файлов с произвольным доступом имеется аппарат, т.н. ACTUAL или SYMBOLIC KEY : ACTUAL KEY (фактический ключ) определяет поле данных, не являющееся частью входной-выходной записи. Оно содержит относительный номер дорожки (взятый относительно первой дорожки файла).

SYMBOLIC KEY (символический ключ), кроме этого, содержит и некоторое наименование.

Файлы с прямым доступом должны открываться как OUTPUT. Для внесения записи необходимо использовать соответствующий ключ и оператор WRITE . Значение ключа в данный момент определяет и место записи в файле. Необходимо отметить, что при этом всегда в секции FILE - CONTROL необходимо указывать предполагаемое число элементов файла, т.е. писать фразу FILE - LIMIT IS literal - 1 . Когда значение ключа превышает значение literal - 1 , срабатывает статья INVALID KEY . Это означает, что либо ключ указывает на несуществующую дорожку, либо запись с заданным ключом не будет найдена.

Покажем на примере, как можно организовать два файла с прямым доступом на диске.


```
IDENTIFICATION  DIVISION.
PROGRAM-ID. EXAMP - 2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE - COMPUTER. 6200.
OBJECT - COMPUTER. 6200.
SPECIAL - NAMES .
OUTPUT IS PRINT.
INPUT - OUTPUT SECTION.
FILE - CONTROL.
       SELECT OMPOD  ASSIGN TO DISKO1
       FILE LIMIT IS 19
       ACCESS MODE IS RANDOM
       ACTUAL KEY IS ACT - KEY.
       SELECT MZD ASSIGN TO DISKO2
       FILE LIMIT IS 7
       ACCESS MODE IS RANDOM
       ACTUAL KEY IS ART - KEY.
       SELECT  F-CARD  ASSIGN TO INPUT.
DATA DIVISION.
FILE SECTION.
FD OMPOD
       LABEL RECORDS OMITTED
       DATA RECORD IS OM-REC.
   01  OM-REC.
   02  KD PICTURE 9(4).
   02  DU PICTURE X(25).
FD MZD
       LABEL RECORDS OMITTED
       DATA RECORD IS MOZ-REC.
   01  MOZ-REC.
   02  KD PICTURE 9(4).
   02  ACH PICTURE 9(6).
   02  RDU PICTURE 9(5).
```

```
FD  F-CARD
    LABEL RECORDS OMITTED
    DATA RECORDS ARE CARD-REC MAC-REC.
 01 CARD-REC.
    02 KD PICTURE 9(4).
    02 DU PICTURE X(25).
 01 MAC-REC.
    02 KD PICTURE 9(4).
    02 ACH PICTURE 9(6).
    02 RDU PICTURE 9(5).
WORKING - STORAGE SECTION.
 77  N PICTURE 9(2) VALUE 0.
 01  ACT-KEY PICTURE 9(2) VALUE 0.
 01  ART-KEY PICTURE 9(2) VALUE 0.
PROCEDURE DIVISION.
    OPEN INPUT F-CARD OUTPUT OMPOD MZD.
    PERFORM NACH-1  THRU   NAM-2 VARYING
    N FROM 1 BY 1 UNTIL N=20.
    GO TO MARCH.
NACH-1
    READ F-CARD INTO OM-REC  AT END
    GO TO NAM-2 .
NAM-2.
    MOVE N TO ACT-KEY.
    WRITE OM-REC INVALID KEY
    PERFORM ERROR-ON.
ERROR-ON.
    DISPLAY  "OSHIBKA" UPON PRINT.
    GO TO  KON-2.
MARCH.
    PERFORM SLED-1 THRU MET-1
    VARYING N FROM 1 BY 1 UNTIL N=8.
    GO TO KON-2.
```



```
SLED-1.
    READ F-CARD INTO MOZ-REC AT
    END GO TO MET-1.
MET-1.
    MOVE N TO ART-KEY.
    WRITE MOZ-REC INVALID KEY
    PERFORM ERROR-ON.
KOW-2.
    CLOSE F-CARD OMPOD MZD.
    STOP RUN.
```

Данная программа дает возможность записать на диске два файла OMPOD с I9 элементами и MZD с 7 элементами. В случае необходимости, используя READ и давая разные значения ключа (ACT - KEY или ART - KEY) можно читать в оперативную память с диска соответствующий элемент данного файла.

Но предложенная организация файла очень неэкономична с точки зрения расхода памяти, т. к. каждый элемент файла имеет еще и индекс, занимающий некоторое место на дорожке диска. Это, конечно, дает возможность быстро, произвольным образом, обращаться к необходимому элементу.

Но при достаточно большом файле становится очень неэкономичным располагать его таким же образом на диске и каждый раз читать только один элемент файла в оперативную память. Гораздо экономнее (с точки зрения памяти и числа обращений к диску) читать определенное количество элементов массива с диска в оперативную память и там соответствующим образом уметь выделять необходимый элемент. Это можно достичь с помощью следующей структуры файла. Сам файл как бы состоит из отдельных зон (элементов файла), а внутри каждой зоны имеется необходимое число членов зоны. Тогда с помощью ключа можно читать в память целую зону, а потом искать нужный член зоны уже в оперативной памяти машины. Каждую зону можно описать как групповую величину и использовать фразу OCCURS для описаний однородных, идентично описываемых, подряд расположенных данных. Тогда, например, файл из I5 элементов можно организовать последовательностью трех зон, каждая из которых имеет по пять членов. Например, для нашего случая файл OMPOD можно описать следующим образом:

```
   SELECT OMPOD ASSIGN TO DISK01
   FILE LIMIT IS 3
FD  OMPOD
   LABEL RECORDS OMITTED
   DATA RECORD IS OM-REC .
01 OM-REC.
   02 KD PICTURE 9(4) OCCURS 5 TIMES.
   02 DU PICTURE X(25) OCCURS 5 TIMES.
```

Тогда имеем следующую картину (см. рис.2).

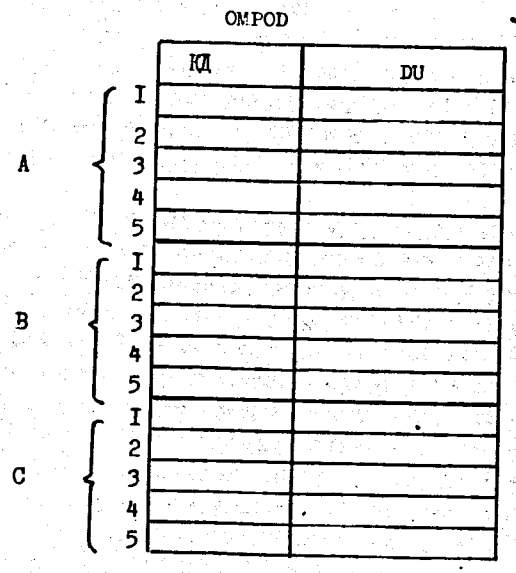

Рис. 2.

При помощи ключа можно записывать и читать зоны A, B, C в файле, а при помощи индексных скобок ( ) и соответствующих значений индекса (от I до 5) обращаться к соответствующему элементу в данной зоне.

Этот метод дает возможность путем экспериментальной проверки достичь приемлемый компромисс между временем, затрачиваемым на поиск элемента файла и объемом памяти для размещения данного файла. В этом случае возможна организация не только мультисписковых узловых структур на диске, но и построение любых информационных массивов произвольной длины.

ЛИТЕРАТУРА

I. Д.Д.Арнаудов. Сообщение ОИЯИ I0-7586, Дубна, 1973.

2. А.И.Китов. " Программирование экономических и управленческих задач". М., Сов. радио, 1971.

3. Д.Д.Арнаудов. "Применение системы автоматизации программирования АЛГЭМ-СТЭ для работы со списковыми величинами", сб. Цифровая вычислительная техника и программирование, вып.6, М., Сов.радио, 1971.

4. Д.Д.Арнаудов. "Об одном способе организации поискового массива в библиографических ИПС", сб. Цифровая вычислительная техника и программирование, вып.7, М., Сов. радио, 1972.

Рукопись поступила в издательский отдел
6 декабря 1973 года.