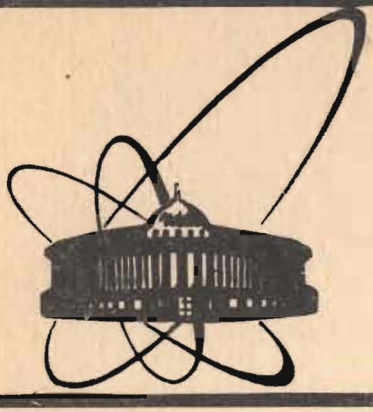


сообщения
объединенного
института
ядерных
исследований
дубна

Р11-88-720

В.Шуберт, Э.В.Шарапова

ОПЕРАЦИИ ДЛЯ ОБРАБОТКИ ПОСЛЕДОВАТЕЛЬНОСТИ СИМВОЛОВ В ФОРТРАНООРИЕНТИРОВАННЫХ ПРОГРАММАХ

1988

I. Введение

Описывается система программ, которая обеспечивает обработку последовательностей символов (ПС) в фортраноориентированных программах. Система программ написана на диалекте Фортрана СДФ [1,2] и может быть поставлена на всех ЭВМ, которые поддерживают фортраноориентированную технологию программирования (Ф-технологию) [3,7].

Работа была выполнена в ЛВТА ОИЯИ.

I.I. Краткие сведения об Ф-технологии

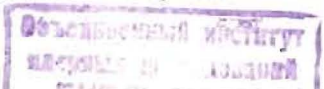
Ф-технология [1-3,7] является системой методических, лингвистических и программных средств для создания познаваемого, переносимого и пластичного программного обеспечения.

Ф-технология была разработана в ЛВТА ОИЯИ и поддерживается на базовых ЭВМ центрального вычислительного комплекса ОИЯИ (БЭСМ-6, CDC-6500 и ЕС ЭВМ). Средства Ф-технологии поддерживают создание, отладку, использование и техническое обслуживание программного обеспечения. К средствам лингвистической поддержки Ф-технологии относятся определенные диалекты базовых языков (Фортран, Паскаль), языки для проектирования программного обеспечения (Ф-псевдокод) и для представления текстовых структур.

Средством программной поддержки Ф-технологии являются прекомпиляторы, которые переводят диалекты базовых языков в базовые языки, программы для автоматического преобразования текстов к виду программной документации и др.

I.2. Обработка последовательности символов (ПС) в СДФ

Язык СДФ является диалектом Фортрана. Он позволяет программировать в соответствии с требованиями структурного программирования, что связано с разделением текста на дерево пошагового уточнения. В СДФ, однако, отсутствует возможность непосредственной обработки последовательности символов, если базовым языком является Фортран IУ. В Фортран IУ мы имеем дело с единичными символами, как с индексированными элементами массива. Между тем, в процессе обработки информации, поддерживаемом ЭВМ, в интерфейсе между человеком и ЭВМ в качестве носителя информации естественно выступают последовательности символов. Оказывается полезным рассматривать последовательность символов как



некоторую автономную структурную единицу и иметь возможность различных действий над этими структурами. Такие возможности обеспечиваются разработанной системой программ, которая реализует тип данных CHAR соответственно семантике языка PL/1 [4], что обеспечивает создание, изменение и уничтожение ПС с любой длиной. Из набора функций PL/1 для обработки ПС было реализовано представительное подмножество.

2. Цель обработки последовательности символов (ПС)

В процессе обработки информации, поддерживаемом ЭВМ, ПС, как носители информации, возникают в интерфейсе между человеком и ЭВМ. Для того чтобы ЭВМ могла обрабатывать ПС, они должны быть преобразованы в нужную форму. И обратно, при выводе ПС из ЭВМ, внутреннее представление информации должно быть заменено соответствующим внешним представлением. Этой цели служат операции ввода-вывода в языках программирования, а также ряд операций для обработки ПС, описываемых ниже.

2.1. Операции для обработки ПС

Типичными операциями для обработки ПС являются:

- вычисление длины ПС (LENGTH)
- вычисление частного ПС (SUBSTR)
- связывание частных ПС (JOIN)
- определение позиции частного ПС (INDEX)
- сравнение двух ПС (CHCOMP)
  а) по длине,
  б) по внутреннему представлению.

Характерным при обработке ПС является создание и уничтожение ПС. Отсюда следует, что память для ПС должна быть динамически управляемой. На языке PL/I существуют для этого некоторые языковые средства (ALLOCATE,FREE,EMPTY).

Базисными средствами для динамического управления ПС являются:

- инициализация памяти для ПС (INITIA)
- определение пустого ПС (NEW)
- преобразование массива символов в ПС (CHWRIT)
- перезапись ПС из массива символов (REWRIT)
- уничтожение ПС (DELETE)
- преобразование ПС в массив символов (CHREAD)
- вычисление длины свободной памяти (FREEM)
- вычисление длины рабочей памяти. (LENWRK)

Кроме того, имеются более сложные операции:

- уничтожение частного ПС (IMPAND)
- вставка частной ПС в ПС (INSERT)
- перепись части ПС в ПС (OVRWRT)

Подробное описание операций дано в главе 3.4. Операции для обработки последовательности символов можно использовать вызовом SUBROUTINE с параметрами или FUNCTION в каждой фортраноориентированной программе. Некоторые другие полезные возможности обработки ПС, такие как

- преобразование массива символов в представление других типов данных и обратно,
- преобразование массива символов реализованы в самой операционной системе ЭВМ.

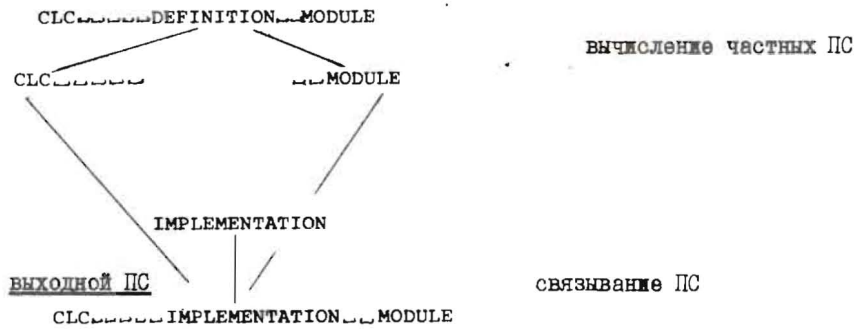
2.2. Примеры обработки ПС

2.2.1. Примеры ПС

MOSKVA	длина ПС=6
A+B-C*3	длина ПС=7
CLC␣␣␣␣␣␣DEFINITION␣␣MODULE	длина ПС=26

2.2.2. Создание и связывание ПС

Входной ПС



2.2.3. Поиск и сравнение ПС

входной ПС	искомый ПС	Исчисляемая позиция
MOSKVA	KV	4
сравнение		**результат сравнения**
MOSKVA	KV	больше ("MOSKVA" длиннее, чем "KV").
A+B-C*3	A-B-C*3	меньше (внутреннее представление символа "-" больше, чем символа "+", в коде ДКОИ).

2.2.4. Конструирование ПС

- уничтожение частного ПС

входной ПС: операция:

A+B-C*3 уничтожение с 3-го до 5-го знака

выходной ПС:

A+*3

- вставка ПС

входной ПС: операция:

A+*3 вставка ПС "A+B"за 2-ой позицей

выходной ПС:

A+A+B*3

Программная реализация примеров приведена в 4.

3. Проектирование системы программ

3.I. Выбранная архитектура

Система программ, реализующая возможности обработки ПС, имеет трехуровневую структуру. Каждый уровень представляет собой коробку данных, т.е. соединяет в себе функции и данные. Уровни находятся друг к другу в отношении обобщения, подобно концепции уровней операционной системы MULTICS /5/. Функции каждого уровня реализуются посредством функций более нижнего уровня и программного кода. С данными каждого уровня работают только функции соответствующего уровня.

Рассмотрим подробнее уровни, соответствующие функции и внутренние данные.

УРОВЕНЬ	ФУНКЦИЯ (см.2.I)	ВНУТРЕННИЕ ДАННЫЕ
I. Управление ПС	INITIA NEW CHWRIT REWRIT DELETE CHREAD FREEM LENGTH LENWRK	- управляющая таблица для ПС (POINT) - длина управляющей таблицы для ПС (LPOINT) - массив строк символов (CHAR) - максимальная длина строк символов (CHRBLK)
2. Анализ ПС	SUBSTR JOIN INDEX CHCOMP	- рабочий массив для ПС (WORK) - длина рабочего массива для ПС (LWORK)
3. Синтез ПС	IMPAND INSERT OVRWRT	

Номера уровней увеличиваются с уровнем обобщения. Программы каждого уровня структурированы на модули. Модули переводятся отдельно и связаны с редактором связи.

3.2. Спецификация модулей

Спецификация модулей следует из определения модульного интерфейса. Форма написания соответствует построению DEFINITION MODULE в MODULA2 /6/. DEFINITION MODULE - это определение внешнего вида модуля (наименование). IMPORT - определяет, какие программные объекты модуль требует для своей работы из внешнего мира. EXPORT - определяет, какие программные объекты модуль содержит и можно использовать в других модулях. Дополнительно пишется высказывание "содержит следующие частные модули" - PARTS .

Ниже приводятся примеры уровней декомпозирования:

- целый модуль

```
DEFINITION MODULE STR
IMPORT;
EXPORT INITIA,NEW,CHWRIT,REWRIT,DELETE,CHREAD,FREEM,LENGTH,
       LENWRK,SUBSTR,JOIN,INDEX,CHCOMP,IMPAND,
       INSERT,OVRWRT;
PARTS  STR2,STR4,STR5;
```

- управление ПС

```
DEFINITION  MODULE STR2
IMPORT;
EXPORT INITIA, NEW,CHWRIT,REWRIT,DELETE,CHREAD,FREEM,
       LENGTH,LENWRK;
PARTS STR21,STR22;
```

- управление памятью

```
DEFINITION MODULE STR21
IMPORT POINT, LPOINT, CHRBLK, CHAR;
EXPORT INITIA, NEW, DELETE, LENGTH, FREEM,
       POINT, LPOINT, CHRBLK, CHAR;
PARTS;
```

- чтение и запись ПС

```
DEFINITION MODULE STR22
IMPORT NEW, FREEM, POINT,LPOINT,CHRBLK, CHAR;
EXPORT CHWRIT, REWRIT, CHREAD,POINT,LPOINT,CHRBLK,CHAR;
PARTS;
```

- анализ ПС

```
DEFINITION MODULE STR4
FROM STR2 IMPORT LENGTH, FREEM,CHWRIT,CHREAD;
EXPORT SUBSTR, JOIN, INDEX, CHCOMP, LENWRK;
PARTS STR43,STR44;
```

- анализ ПС (часть I)

```
DEFINITION MODULE STR43
IMPORT LENGTH,FREEM,CHWRIT,CHREAD,LWORK,WORK;
EXPORT SUBSTR,JOIN,LWORK,WORK;
PARTS;
```

- анализ ПС (часть 2)

```
DEFINITION MODULE STR44
IMPORT LENGTH,CHREAD,LWORK,WORK;
EXPORT INDEX,CHCOMP,LENWRK,LWORK,WORK;
PARTS;
DEFINITION MODULE STR5
FROM STR2 IMPORT DELETE,LENGTH,FREEM;
FROM STR4 IMPORT SUBSTR,JOIN;
EXPORT IMPAND,INSERT,OVRWRT;
PARTS;
```

3.3. Проектирование уровня внутренних данных

Описывание уровня внутренних данных ведется соответственно MODULA2 и разделено в определении типов и декларации переменных и констант.

- определение типов:

управляющая таблица для ПС

```
TYPE CONTROLTABLE = ARRAY[1...LPOINT,1,...3]
                    OF INTEGER
```

массив строк символов

```
TYPE CHARACTERSTORAGE = ARRAY[1...LPOINT,1...CHRBLK]
                        OF LOGICAL * 1
```

рабочий массив для ПС

```
TYPE WORKAREA = ARRAY[1...LWOOK] OF LOGICAL * 1
```

Замечание: данные базовые типы (INTEGER,LOGICAL * 1) ВХОДЯТ В состав FORTRANa .

- декларация переменных

```
VAR POINT:CONTROLTABLE,
    CHAR:CHARACTERSTORAGE,
WORK:WORKAREA;
```

- декларация констант

```
CONST LPOINT = 500 - максимальное число ПС,
      CHRBLK = 160 - размер блока для ПС,
      LWORK  = 65536 - максимальное число символов рабочего массива.
```

Каждой строке в CONTROLTABLE соответствует одна строка в CHARACTERSTORAGE (равный индекс). Одна строка в CONTROLTABLE предназначена для хранения следующей информации:

I позиция: тип соответственной строки в CHARACTERSTORAGE

- 0 свободно
- I занято
- 2 временно занято
- 3 постоянно занято
- 4 постоянно занято (разрешается только чтение)

2 позиция: индекс строки продолжения

- 0 – если продолжения не существует

3 позиция: число хранимых символов в строке.

3.4. Проектирование функциональных возможностей

3.4.I. Операции и их параметры

Ниже приводятся операции и их параметры. Дается пояснение семантики параметров.

INITIA(RC) – инициализация памяти для ПС

RC – код выполнения операции (Кво)

всегда 0.

NEW(TYPE,POINT,RC) – определение пустого ПС

TYPE	тип ПС
2	временное хранение
3	постоянное хранение
4	постоянное хранение констант.
POINT	численный идентификатор ПС (ЧИПС)
RC	Кво
0	без ошибок
2	память недостаточна
I6	тип ПС не разрешается.

DELETE(POINT,RC) – уничтожение ПС

POINT	ЧИПС
RC	Кво
0	без ошибок
4	ЧИПС идентифицирует частный ПС
8	ЧИПС не идентифицирует ПС.

LENGTH(POINT,TYPE,LEN,RC) – вычисление длины ПС

POINT	ЧИПС
TYPE	тип ПС
LEN	длина ПС
RC	Кво
0	без ошибок
8	ЧИПС не идентифицирует ПС.

FREEM(LEN,RC) – вычисление длины свободной памяти

LEN длина

RC Кво

всегда 0.

CHWRIT(TYPE,STRING,LEN,POINT,RC) – преобразование массива символов в ПС

TYPE тип ПС

STRING массив символов

LEN длина массива

POINT ЧИПС

RC Кво

см. NEW

REWRIT(POINT,STRING,LEN,RC) – перезапись ПС из массива символов

POINT ЧИПС

STRING массив символов

LEN длина массива

RC Кво

0 без ошибок

2 память недостаточна

8 ЧИПС не идентифицирует ПС

CHREAD(POINT,TYPE,STRING,LEN,RC) – преобразование ПС в массив символов

POINT ЧИПС

TYPE тип ПС

STRING массив символов

LEN число символов

RC Кво

см. DELETE

SUBSTR(POINT1,POS,LEN,POINT2,RC) – вычисление частного ПС.

POINT1 ЧИПС, входной

POS начальная позиция во входном ПС

LEN длина частного ПС

POINT2 ЧИПС, выходной

RC Кво

см.CHREAD,CHWRIT

JOIN(POINT1,POINT2,POINT3,RC) – связывание частных ПС

POINT1 ЧИПС,1.входной

POINT2 ЧИПС,2.входной

POINT3 ЧИПС, выходной

RC Кво

СМ.CHREAD,CHWRIT

INDEX(POINT1,POS1,POINT2,LEN2,POS2,RC) – определение позиции частного ПС

POINT1 ЧИПС, входной

POS1 начальная позиция для поиска

POINT2 ЧИПС, ПС сравнения

LEN длина ПС сравнения

POS2 найденная позиция

RC Кво

см. CHREAD

CHCOMP(POINT1,POINT2,C) – сравнение двух ПС

POINT1 ЧИПС, 1ПС

POINT2 ЧИПС, 2ПС

C код сравнения

0 равно

1 меньше

2 больше

8 ЧИПС не идентифицирует ПС.

LENWRK(RC) – вычисление длины рабочей памяти

RC Кво

всегда 0.

IMPAND(POINT,POS,LEN,RC) – уничтожение частного ПС.

POINT ЧИПС

POS последняя, неизменяемая позиция

LEN число уничтоженных символов

RC Кво

см. CHREAD,CHWRIT

INSERT(POINT1,POS,POINT2,RC) – вставка частного ПС в ПС.

POINT1 ЧИПС, базовый ПС

POS последняя, неизменяемая позиция

POINT2 вставляемый ПС

RC Кво

см. CHREAD,CHWRIT

OVRWRT(POINT1,POS,POINT2,RC) – переписываниечасти ПС с ПС

POINT1 ЧИПС, базовый ПС
POS последняя, неизменяемая позиция
POINT2 переписывающий ПС
RC Кво
см. CHREAD,CHWRIT

3.4.2. Функции и их атрибуты.

Атрибутом функции является:

имя функции (атрибут I.параметра,..., атрибут m. параметра)

Атрибутом параметра является:

.свойство. может быть: входным (> ..)
входным и выходным (> .. >)
выходным (.. >)

.тип параметра

Примеры для типов данных – это "целое число" (I) и "массив данных" (S).

Функции уровня (кроме I-ого) базируются на функциях нижних уровней.

номер функции	атрибут функции	номера вызываемых функций
1.	INITIA (>I>)	
2.	NEW (>I,>I>,>I>)	
3.	DELETE (>I,>I>)	
4.	LENGTH (>I,>I>,>I>,>I>)	
5.	FREEM (>I>,>I>)	
6.	CHWRIT (>I,>S,>I>,>I>,>I>)	2,5
7.	REWRIT (>I,>S,>I>,>I>)	4
8.	CHREAD (>I,>I>,>S>,>I>,>I>)	
9.	SUBSTR (>I,>I,>I,>I>,>I>)	4,5,6,8
10.	JOIN (>I,>I,>I>,>I>)	4,5,6,8
11.	INDEX (>I,>I,>I,>I,>I>,>I>)	4,8
12.	CHCOMP (>I,>I,>I>)	4,8
13.	LENWRK (>I>)	
14.	IMPAND (>I>,>I,>I,>I>)	3,4,5,9,10
15.	INSERT (>I>,>I,>I,>I>)	3,4,5,9,10
16.	OVRWRT (>I>,>I,>I,>I>)	3,4,5,9,10

4. Примеры использования

Операции для обработки последовательности символов (ПС) можно использовать вызовом SUBROUTINE или FUNCTION(LENWRK) в каждой фортраноориентированной программе. С помощью примеров из части 2.2 будет показана возможность их использования

4.I. Определение исходных ПС

Предполагаем, что существуют следующие массивы символов:

имя	значение
VEC1	MOSKVA
VEC2	A+B-C*3
VEC3	CLC DEFINITION MODULE
VEC4	IMPLEMENTATION

из этих массивов символов мы сгенерируем ПС с именами S1,S2,S3 и S4

```
CALL CHWRIT(2,VEC1,6,S1,RC1)      S1⇒MOSKVA
CALL CHWRIT(2,VEC2,7,S2,RC2)      S2 ⇒ A+B-C*3
CALL CHWRIT(2,VEC3,26,S3,RC3)     S3⇒CLC DEFINITION MODULE
CALL CHWRIT(2,VEC4,14,S4,RC4)     S4⇒IMPLEMENTATION
```

Выбор типа "2" для хранения ПС был взят для примера. (Тип "3" или "4" также возможен).

S1...S4 являются переменными типа INTEGER ,предназначенными для хранения числовых идентификаторов последовательностей символов (ЧИПС), сгенерированных ПС.

4.2. Создание и связывание ПС

```
CALL SUBSTR(S3,1,8,S5,RC5)          S5⇒CLC
CALL SUBSTR(S3,19,8,S6,RC6)         S6 ⇒ MODULE
CALL JOIN(S5,S4,S7,RC7)             S7⇒CLC      IMPLEMENTATION
CALL DELETE(S5,RC5)                 S5 ⇒
CALL JOIN(S7,S6,S8,RC8)             S8⇒CLC      IMPLEMENTATION
CALL DELETE(S6,RC6)                                    MODULE
CALL DELETE(S7,RC7)                 S6 ⇒
CALL CHREAD(S8,T8,VECTOR,L8,RC8)    S7⇒
```

Здесь

имя	значение
T8	2
VECTOR	CLC IMPLEMENTATION MODULE
L8	30

```
CALL DELETE(S8,RC8)
```

Операции DELETE в последовательности вызовов можно опускать. Они служат для экономии памяти.

4.3. Поиск и сравнение ПС

–поиск

```
CALL SUBSTR(S1,4,2,S5,RC5)          S5⇒KV
CALL INDEX(S1,1,S5,2,P5,RC5)
```

значение P5:4
т.е."KV" начинается на позиции 4 в ПС S1 .

- сравнение

а)
```
CALL CHCOMP(S1,S5,C5)
CALL DELETE(S5,RC5)          значение  C5 : 2 (больше)
```

в)
```
CALL SUBSTR(S2,1,1,S5,RC5)        S5⇒A
CALL SUBSTR(S2,4,1,S6,RC6)        S6⇒ -
CALL SUBSTR(S2,3,5,S7,RC7)        S7⇒ B-C*3
CALL JOIN(S5,S6,S8,RC8)           S8⇒A -
CALL JOIN(S8,S7,S9,RC9)           S9⇒A-B-C*3
CALL CHCOMP(S2,S9,C9)
CALL DELETE(S5,RC5)          значение  C9:1 (меньше)
```

4.4. Конструирование ПС

- уничтожение частного ПС

```
CALL SUBSTR(S2,1,3,S5,RC5)       S5⇒A+B
CALL IMPAND(S2,2,3,RC2)          S2⇒A+*3
CALL CHREAD(S2,T2,VECTOR,L2,RC2)
                          здесь      имя        значение
                                      T2            2
                                   VECTOR         A+*3
                                      L2            4
```

- вставка ПС

```
CALL INSERT(S2,2,S5,RC2)         S2⇒A+A+B*3
CALL CHREAD(S2,T2,VECTOR,L2,RC2)
                          здесь      имя        значение
                                      T2            2
                                   VECTOR       A+A+B*3
                                      L2            7
```

5. Заключение

Операции для обработки ПС принимались как базовые функции в программах, расширяющих инструментальный комплекс Ф-технологии. Показано, что применение операций для обработки ПС полезно, если массивы или текстовые файлы не строго форматированы.

Преимущество разработанных операций для ПС, по сравнению с теми же операциями в Фортран-77, состоит в большей гибкости. Возможно динамически генерировать и уничтожать ПС практически любой длины.

В настоящее время операции для обработки ПС реализуют функции управления базами данных в разрабатываемой системе для объектно - ориентированного программирования. Кроме того, разработанный инструмент был использован для контроля соответствия модульной спецификации реализации модуля в разработанных системах программ.

ЛИТЕРАТУРА

I. Корнейчук А.А. ОИЯИ, II-80-382, Дубна, I980.
2. Железнова К.М., Корнейчук А.А., Шарапова Э.В. ОИЯИ, 5-83-226, Дубна, I983.
3. Железнова К.М., Корнейчук А.А., Шарапова Э.В. ОИЯИ, II-85-805, Дубна, I985.
4. Фролов Г.Д., Олюнин В.Ю. Практический курс программирования на языке PL/1 . М.: Наука, I983.
5. Organick E.I.: The Multics System: An examination of its structure MIT Press, Cambridge, Mass, 1972.
6. Вирт Н. Программирование на языке МОДУЛА-2. М.:Мир, I987.
7. Корнейчук А.А., Шарапова Э.В. Фортраноориентированная технология программирования.В сб.:2Всесоюзная конференция "Технология программирования", тезисы докладов, ч.I, Киев, ИК АН УССР, I986, с.II7-II8.
8. Meyer B. Object - oriented design, First European software Engineering Conference, Strasbourgh , 8-11, September, 1987.

Рукопись поступила в издательский отдел
30 сентября I988 года.