*Л.И. Брусиловский, Ю.А. Михайлов*

# РАЗРАБОТКА МОБИЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ЭВМ ЛИНИЙ PDP-11 И VAX

## 1. Общие положения

Наличие разнородных архитектур ЭВМ, разнообразных операционных систем даже для ЭВМ одной архитектуры и многочисленных языков программирования даже в рамках одной операционной системы породили острейшую проблему разработки мобильного программного обеспечения (ПО), т.е. такого, которое при минимальных затратах на модификацию (либо вообще без таковой) могло бы эксплуатироваться на ЭВМ с различной архитектурой, либо, по крайней мере, на одной и той же ЭВМ, но в различных операционных системах.

В настоящее время существуют два основных метода решения данной проблемы: разработка мобильных операционных систем (ОС) и создание универсальных языков программирования, имеющих средства выхода на "нижний" (аппаратный) уровень.

В первом случае программное обеспечение разрабатывается в рамках стандартизованных ОС, например, UNIX, CP/M, MS-DOS и т.д. При таком подходе предполагается, что разрабатываемые программы будут переносимы за счет идентичной реализации системных вызовов в этих ОС на ЭВМ различной архитектуры. В то же время здесь имеются по меньшей мере два препятствия для разработки мобильного ПО.

Во-первых, разрабатываемое программное обеспечение, как правило, будет зависеть от разрядности процессора и объема адресуемой физической памяти: ПО, разрабатываемое для 32-разрядных машин, скорее всего не будет работать на 16-разрядных даже в рамках одной и той же ОС (например, UNIX). Попытка использовать ПО для 16-разрядных машин на 32-разрядных приведет к резкому снижению эффективности использования аппаратуры 32-разрядной ЭВМ. Во-вторых, программы, написанные в рамках унифицированных ОС, как правило, не могут использовать характерные аппаратные достоинства конкретного типа процессора, в силу чего либо сужается круг задач, решаемых с помощью унифицированных ОС, либо возникает целое семейство подобных операционных систем, учитывающих специфику конкретных аппаратных решений, однако, за счет определенной потери мобильности. Например, существует целое семейство UNIX-подобных ОС: AIX, ULTRIX, PHOENIX, XENIX, VENIX, OS/9 и т.д.

Второе направление разработки мобильного программного обеспечения представлено универсальными языками программирования высокого уровня, одинаково реализуемыми во всех операционных системах и имеющими идентичный программный интерфейс для связи со всеми операционными системами. В первую очередь это языки ADA, Modula-2, C и мобильная расширенная версия языка Pascal - Pascal-2.

Мобильность программ на языке ADA [1,2] обусловлена жестко регламентированными требованиями к реализации и составу системы программирования. Не допускаются подмножества и расширения языка ADA относительно стандартного. Стандарт языка ADA не только специфицирует программный интерфейс с "нижним" уровнем (пакеты), но и состав системы программирования (СП) (отладчики, редакторы, библиотекари и т.д.). Программист, разрабатывающий прикладные программы на языке ADA, практически "не видит" подстилающей ОС и не испытывает необходимости в обращении к утилитам данной ОС.

Несмотря на очевидные достоинства языка ADA, разработка ПО на этом языке сдерживается рядом технических причин: компиляторы с языка ADA реализованы не для всех ЭВМ, они громоздки, не являются оптимизирующими и т.д. [3]. Но со временем язык ADA, безусловно, займет доминирующее положение в области разработки мобильного программного обеспечения.

По сравнению с языком ADA язык программирования Modula-2 [4] проще. Хотя он и включает в себя аналоги основных конструкций языка ADA, тем не менее имеет и ряд существенных недостатков, а именно:

- единичная точность плавающей арифметики (хотя в языке и не специфицируется точность представления);
- отсутствие встроенных средств работы со строковыми данными;
- возможность передачи в качестве параметров с изменяемыми верхними границами только одномерных массивов;
- отсутствие механизма работы с массивами, границы индексов у которых являются арифметическими выражениями;
- невозможность задания структурированных констант (например, массива констант).

С другой стороны, хотя СП Modula-2 имеются практически для всех классов ЭВМ, включая 8-разрядные, язык Modula-2, в отличие от языка ADA, не стандартизован и соответствующая библиотека программных интерфейсов не только не унифицирована, но и не специфицирована [5,6]. Тем не менее, язык Modula-2 является хорошей альтернативой языку ADA во многих приложениях. Разработка мобильного ПО на языке Modula-2 в настоящее время сдерживается, главным образом, отсутствием стандарта как на язык, так и на систему программирования в целом [7].

Язык программирования С [8,9] занимает особое место в ряду языков для разработки мобильного программного обеспечения. В языке специфицирован лишь минимум средств для написания программ, в силу чего он достаточно прост, и реализация компилятора с языка С не представляет проблем. В языке С определены лишь стандартные типы данных и правила построения из них агрегатов (массивов и структур), а также операции над ними. Все остальные средства реализуются посредством вызова функций, оформленных в библиотеки. В качестве стандарта обычно берется состав СП С для операционной системы UNIX. Идентичность имен функций, правила их вызова и реализуемых имен действий как раз и обеспечивает мобильность ПО, разрабатываемого на языке С. С этой точки зрения решение проблемы разработки мобильного ПО на языке С не отличается от ее решения на любом другом языке программирования: необходимо иметь соответствующие компиляторы во всех ОС, для которых предназначено разрабатываемое ПО, и унифицированные библиотеки программных интерфейсов со средствами "нижнего" уровня.

Отношение языка С и его вариантов (например, С++) к языкам типа Pascal (Pascal-2, Modula-2, ADA) подобно отношению ассемблера к языку FORTRAN. При наличии адекватных систем программирования на целевой ЭВМ нет такой задачи, которую можно было бы реализовать на языке типа Pascal, и нельзя было бы реализовать на языке С, и наоборот. Однако соображения технологии программирования (программотехники) представляют языки семейства Pascal более привлекательными. Например, при программировании на языке С гораздо легче написать ненадежное программное обеспечение с большим числом скрытых ошибок. Это обусловлено тем, что в языке С отсутствуют средства полного семантического контроля в пределах раздельно компилируемых единиц, а также средства поддержки коллективной разработки больших программных продуктов: пакеты или модули, имеющиеся в языках ADA и Modula-2. Синтаксис языка С настолько своеобразен, а программы на С настолько трудночитаемы, что проблематично разобраться не только в "чужой", но и модифицировать собственную программу. Идеология программирования на языке С нацеливает программиста скорее на "трюкачество", чем на разработку надежного мобильного ПО. Ряд дополнительных серьезных недостатков языка С отмечен в [10,11].

Для ЭВМ класса PDP-11, VAX и IBM PC в настоящее время имеется СП Pascal-2 [12-14], обеспечивающая стандарт ISO языка [15] и являющаяся мобильной между всеми операционными системами вышеуказанных ЭВМ [13]. Эта система программирования включает оптимизирующий компилятор, допускающий ряд расширений языка относительно стандарта (например, раздельную компиляцию модулей, средства "нижнего" уровня), символьный отладчик (уровня операторов исходного текста) и ряд утилит, включая пакет поддержки программирования на ассемблере. Входной язык СП Pascal-2 по своим возможностям практически адекватен возможностям языка Modula-2, за исключением средств мультипрограммирования. Входной язык Pascal-2 одинаков во всех операционных системах ЭВМ класса PDP-11, VAX и IBM PC.

Для обеспечения мобильности ПО, разработанного на языке Pascal-2, можно создать библиотеку программных интерфейсов, обращение к которой следует унифицировать для операционных систем RT-11, TSX-Plus, SHARE Plus, RSX-11M, RSTS/E, P/OS, VMS и MS-DOS. В качестве основы для построения такой библиотеки можно взять приложения к составу и функциям библиотеки для Modula-2 [5,6]. С другой стороны, аналогичные средства введены в СП Turbo-Pascal [16] в качестве расширения языка Pascal. Наличие большого числа программ, разработанных в рамках СП Turbo-Pascal делает целесообразным создание библиотеки модулей для СП Pascal-2,

реализующих эти расширения стандарта языка, что может облегчить перенос программ между СП Pascal-2 и СП Turbo-Pascal. Рассматриваемый ниже подход к разработке примитивов "нижнего" уровня можно использовать и для разработки аналогичного программного интерфейса для языка Modula-2.

Кратко упомянем такую проблему как перепрограммирование, т.е. перепись программ с одного языка на другой по мере появления новых универсальных языков программирования. Частично данная проблема может решаться введением в новые языки встроенных средств связи с другими языками программирования. Так, в языке ADA - это прагма INTERFACE, в ряде реализаций языка C - это служебные слова FORTRAN и ASM, в СП Pascal-2 - директива NONPASCAL. Следует отметить, что аналогичного механизма в языке Modula-2 нет, и это может послужить серьезной причиной отказа от него при разработке прикладного ПО, например, в случае необходимости использовать существующие пакеты, допускающие вызовы через некоторые входные точки и написанные на языке, отличном от Modula-2.

## 2. Реализация программного интерфейса

Расширение языка Pascal в СП Turbo-Pascal можно разбить на три группы, реализующие основные функции программного интерфейса с "нижним" уровнем:

- средства управления памятью;
- средства работы со строками;
- средства управления видеотерминалом.

Две первые группы легко реализуются в рамках самой СП Pascal-2 и являются не зависящими от подстилающей операционной системы. Расширения третьей группы зависят от операционной системы, под управлением которой выполняется программа на Pascal-2. Поэтому, они должны быть реализованы отдельно для каждой операционной системы. Рассматриваемые ниже реализации модулей интерфейса ориентированы на версию СП Pascal-2 v2.1. Там, где отличие между версиями 2.1 и 2.0 принципиальны, будет сделана соответствующая оговорка.

### 2.1. РАБОТА С ТЕРМИНАЛОМ

Средства управления видеотерминалом в СП Turbo-Pascal включает:

- получение одиночного символа с клавиатуры;
- позиционирование курсора;
- управление экраном.

Ввод одиночного символа с клавиатуры без эхосопровождения и буферизации в СП Turbo-Pascal выполняется с помощью оператора read специального вида:

```
read (KBD, CH);
```

Аналогичная возможность в операционных системах RT-11 и TSX-Plus реализуется с помощью следующего ассемблерного кода с использованием средств пакета Pasmac [14]:

```
; function GETC: char; external;
; Get immediately one character from keyboard
; (RT-11/TSX-Plus version)

        .TITLE  GETC
        FUNC    GETC,CH.CHAR
        SAVE    <R0>
        BEGIN
        .MCALL  .TTYIN
        BIS     #5000.,@#44
        .TTYIN  CH(SP)
        BIC     #127777.,@#44
        ENDPR
        .END
```

В ОС RSX-11M искомая функция может быть просто реализована на самом языке Pascal:

```
function GETC: char; external;
function GETC;
(* Get immediately one character from keyboard
  (RSX-11M version) *)
var
    C: char;
begin
    read(C);
    GETC:=C
end;
```

Однако при этом терминал должен быть открыт в специальном режиме (см. модуль CRTMOU). Для версии v2.0 этот модуль реализуется на ассемблере следующим образом:

```
; Get immediately one character from keyboard
; (RSX-11M version)

        .TITLE  GETC
CH:     .BLKW   1
        FUNC    GETC,C,CHAR
        BEGIN
        .MCALL  QIOW$S
        QIOW$S  #IO.RAL!TF.RNE,#5,#5,,,,<#CH,#1>
        MOVB    CH,C(SP)
        ENDPR
        .END
```

В ОС RSTS/E этот модуль также реализуется на самом Pascal-2:

```
function GETC: char; external;
function GETC;
(* Get immediately one character from keyboard
  (RSTS/E version) *)
var
    C: char;
begin
    EMT(255); EMT(18); (* отменить эхосопровождение *)
    EMT(255); EMT(20); (* посимвольный ввод *)
    read(C);
    GETC:=C
    EMT(255); EMT(16); (* восстановить эхосопровождение *)
end;
```

Ниже все модули, зависящие от конкретной ОС будут приводиться на ассемблере для ОС TSX-Plus и RSX-11M. В ОС RSTS/E все модули могут быть реализованы средствами самого языка Pascal-2 с использованием системных вызовов [17].

С точки зрения эффективности использования клавиатуры видеотерминала важным является модуль определения наличия символа для ввода в буфере ввода (в Turbo-Pascal это функция KEYPRESSED). С ее помощью можно определять, например, была ли нажата клавиша <ESC> или какая-либо из функциональных клавиш (нажатие функциональной клавиши приводит к передаче в буфер терминала символа <ESC> и еще одного или нескольких символов - так называемой ESC-последовательности). Реализация функции KEYPRESSED для ОС TSX-Plus может иметь вид:

```
; function KEYPRESSED: boolean: external;
BLKARG: .BYTE   0,123
        .TITLE  KEYPRESSED
        FUNC    KEYPRESSED,YES,BOOLEAN
        SAVE    <R0>
        BEGIN
        CLRB    YES(SP)
        MOV     #BLKARG,R0
        EMT     375
        BCS     10
        INCB    YES(SP)
10:     ENDPR
        .END
```

Аналогично для ОС RSX-11M:

```
        .TITLE  KEYPRESSED
        .MCALL  QIOW$S,TTSYM$
BLKARG: .BYTE   TC.TBF,0
        FUNC    KEYPRESSED,YES,SCALAR
        BEGIN
        TTSYM$
        QIOW$S  #SF.GMC,#5,#5,,,,<#BLKARG,#2>
        CLRB    YES(SP)
        TSTB    BLKARG+1
        BEQ     10
        INCB    YES(SP)
10:     ENDPR
        .END
```

В ОС TSX-Plus и RSX-11M для правильной интерактивной работы необходимо установить характеристики терминала. Например, в ОС TSX-Plus режим ввода одиночных символов можно установить из монитора, запустив программу командой RUN/S, либо из программы соответствующим запросом. Кроме того, при нажатии клавиши <CR> в программу обычно передаются два символа: <CR> и <LF>. Добавление символа <LF> можно подавить соответствующей установкой режима терминала. Кроме того, в составе ЭВМ линии PDP-11 и VAX могут работать терминалы различного типа, поэтому в модуле инициализации работы с терминалом желательно динамически (т.е. при запуске программы) определить тип терминала, с которым предполагается работа. Поскольку эта информация важна в дальнейшем, тип терминала можно запомнить в программе для повторного использования.

Аналогично системе Turbo-Pascal можно определить процедуры CRTINIT и CRTEXIT инициализации и завершения работы с терминалом. В ОС TSX-Plus здесь можно было бы ввести установку режима ввода одиночных символов, подавление символа <LF> после нажатия <CR> и определение типа используемого терминала. В ОС RSTS/E здесь можно было бы выполнять установку таких функций терминала, как LC INPUT, LC OUTPUT, ESC, ESCSEQ и т.д.

В ОС TSX-Plus установление спецрежимов терминала можно выполнить с помощью нижеприведенной процедуры SETTERM:

```
(*nomain*)
(*own*)
type
    TERMTYPE = (UNKNOWN, VT52, VT100, HAZELTINE, ADM3A,
                LA36, LA120, DIABLO_AND_QUME, RESERVED, VT200);
var
    ITEMTYP  : TERMTYPE (* переменная, в которой хранится тип
                           используемого терминала *)
```

```
function TERMTYP: TERMTYPE; external;
function TTYPE: TERMTYPE; external;
function TTYPE;
(* вернуть тип используемого терминала *)
begin
    TTYPE:=ITERMTYP
end;
procedure CRTINIT; external;
procedure SETTERM(FUNC, ARG:char); external;
procedure CRTINIT;
begin
(* установить режим посимвольной активации *)
    SETTERM('S',chr(0));
(* подавить символ <LF> после символа <CR> *)
    SETTERM('D',chr(13));
(* определить и запомнить тип используемого терминала *)
    ITERMTYP:=TERMTYP
end;
procedure CRTEXIT; external;
procedure CRTEXIT;
begin SETTERM('T',chr(0));
end;
```

Процедура SETTERM на ассемблере может иметь вид:

```
; procedure SETTERM(FUNC, ARG:char); external;
BLKARG: .BYTE   0,152
        .BLKW   2
        .TITLE  SETTERM
        PROC    SETTERM
        PARAM   FCODE,CHAR
        PARAM   ARGVAL,CHAR
        SAVE    <R0>
        BEGIN
        MOVB    FCODE(SP),BLKARG+2
        MOVB    ARGVAL(SP),BLKARG+4
        MOV     #BLKARG,R0
        EMT     375
        ENDPR
        .END
```

А процедура определения типа терминала TERMTYP:

```
; function TTYPE: TERMTYPE; external;
BLKARG: .BYTE   0,137
        .TITLE  TTYPE
        FUNC    TTYPE,TTY,SCALAR
        SAVE    <R0>
        BEGIN
        MOV     #BLKARG,R0
        EMT     375
        MOVB    R0,TTY(SP)
        ENDPR
        .END
```

Результат, возвращаемый функцией TERMTYP, имеет следующий смысл:

| | |
|---|---|
| 0 | UNKNOWN (неизвестный тип терминала) |
| 1 | VT52 (15И3-000-013, ВТА-2000-3, ВТА-2000-15) |
| 2 | VT100 (ВТА-2000-10М) |
| 3 | HAZELTINE |
| 4 | ADM3A |

```
5       LA36
6       LA120
7       DIABLO, QUME
8       PESERVED
9       VT200 (MC 7105)
```

Функция определения типа терминала в ОС RSX-11M будет иметь вид (для обеспечения мобильности полагаем, что тип TERMTYP тот же, что и для ОС TSX-Plus):

```
; function TERMTYP: TERMTYPE; external;
        .TITLE  TERMTYP
        .MCALL  QIOW$S,TTSYM$
BLKARG: .BYTE   TC.TTP,0
        FUNC    TERMTYP,TTY,SCALAR
        SAVE    <R0>
        BEGIN
        TTSYM$
        QIOW$S  #SF.GMC,#5,#5,,,,<#BLKARG,#2>
        CLR     R0
        MOVB    BLKARG+1,R0
        BEQ     2$
        CMPB    #11,R0
        BNE     1$
        MOVB    #1,R0
        BR      2$
1$:     CMPB    #15,R0
        BNE     3$
        MOVB    #2,R0
        BR      2$
3$:     CLRB    R0
2$:     MOVB    R0,TTY(SP)
        ENDPR
        .END
```

Тогда в ОС RSX-11M процедура CRTMOU может иметь аналогичный вид, за исключением того, что процедура CRTINIT должна выглядеть следующим образом:

```
procedure CRTINIT; external;
procedure CRTINIT;
begin
    reset(INPUT,'TI:/RAL/NOECHO/BUFF:1');
    rewrite(OUTPUT,'TI:/BUFF:1');
    ITERMTYP:=TERMTYP
end;
```

А для версии v2.0 в ОС RSX-11M:

```
procedure CRTINIT; external;
procedure CRTINIT;
begin
    rewrite(OUTPUT,'TI:/BUFF:1');
    ITERMTYP:=TERMTYP
end;
```

В СП Turbo-Pascal позиционирование курсора реализуется с помощью предопределенной процедуры GOTOXY (COL,ROW), где COL и ROW соответственно номера столбца и строки, в которые нужно переместить курсор. В Pascal-2 эту процедуру можно реализовать на самом Pascal:

```
(*nomain*)
type
    TERMTYPE = (UNKNOWN, VT52, VT100, HAZELTINE, ADM3A,
                LA36, LA120, DIABLO_AND_QUME, RESERVED, VT200);
function ttype: TERMTYPE; external;
procedure GOTOXY(COL,ROW:integer); external;
procedure GOTOXY;
const
    ESC=155;
begin
    case TTYPE of
        VT52    : write(chr(ESC),'Y',
                        chr(ROW+31),
                        chr(COL+31));
        VT100   : write(chr(ESC),'[',
                        chr((ROW div 10)+48),
                        chr((ROW mod 10)+48),';',
                        chr((COL div 10)+48),
                        chr((COL mod 10)+48),'H');
        ........................................
    end
end;
```

Процедуры управления экраном в СП Turbo-Pascal: CLRSCR (очистка экрана), CLREOL (очистка от курсора до конца строки), CLREOS (очистка от курсора до конца экрана), LOWVIDEO (снижение яркости), NORMVIDEO (нормальная яркость) можно реализовать следующим образом:

```
const
    ESC=155;
procedure CLRSCR;  (* очистка экрана *)
begin
    case TTYPE of
        VT52    : write(chr(ESC),'E');
        VT100   : write(chr(ESC),'[2J');
        ........................................
    end
end;

procedure CLREOL; (* очистка от курсора  до  конца  строки *)
begin
    case TTYPE of
        VT52    : write(chr(ESC),'K');
        VT100   : write(chr(ESC),'[K');
        ........................................
    end
end;
procedure CLREOS; (* очистка от курсора до конца экрана *)
begin
    case TTYPE of
        VT52    : write(chr(ESC),'J');
        VT100   : write(chr(ESC),'[J');
        ........................................
    end
end;
.................
```

Полный перечень управляющих кодов можно получить в соответствующем руководстве по аппаратному обеспечению терминалов.

## 2.2. ОБРАБОТКА СТРОК

Второй группой примитивов "нижнего" уровня является обработка строк. СП Turbo-Pascal имеет ряд предопределенных процедур и функций для работы со строками. Аналогичные процедуры и функции имеются и в пакете **STRING.PAS**, вхо-

дящем в состав СП Pascal-2. В Turbo-Pascal и в пакете STRING.PAS СП Pascal-2 версии не ниже v2.1 длина ограничена 255 символами. Это обусловлено тем, что строка рассматривается как массив символов размерностью [0..255], причем нулевой элемент массива используется для хранения текущей длины строки. Такое определение строки отличается от принятого в стандарте языка (нижняя граница массива равна 1), поэтому для присвоения строке некоторого значения при использовании пакета STRING.PAS необходимо обращаться к соответствующей процедуре. Другим серьезным недостатком такого подхода является ограничение на длину обрабатываемых строк в 255 символов, так как в ряде приложений это ограничение может оказаться критическим.

Кроме того, реализация Pascal-2 версии v2.1 так называемых согласованных массивов-параметров не допускает их использование в качестве фактических параметров, что резко сужает область применения механизма согласованных массивов для манипулирования со строками.

Универсальным решением будет являться подход, используемый при обработке строк с помощью подпрограмм и функций системной библиотеки SYSLIB: конец строки помечается специальным символом chr(0). Тогда длина строки равна количеству символов, встретившихся в строке до появления символа chr(0).

В принципе можно было бы воспользоваться этими модулями библиотеки SYSLIB, тем более, что наличие директивы nonpascal в языке Pascal-2 позволяет вызывать модули, написанные на других языках (например, на FORTRAN'е или ассемблере), однако фактические параметры непаскальных модулей должны передаваться по ссылке (с описателем var), что может опять создать проблемы, связанные с передачей в качестве параметров строк различной длины. Поэтому целесообразно реализовать модули на самом Pascal-2 и поместить их в библиотеку.

Для вызова этих процедур и функций можно использовать следующий метод. В качестве параметра-строки необходимо указать переменную типа integer, передаваемую по значению (без var), а при вызове соответствующей подпрограммы в качестве фактического параметра задавать адрес строки, приведенный к типу integer, что можно легко осуществить с помощью встроенных функций loophole и ref. Это позволяет передавать в подпрограммы строки различного размера. Системные процедуры и функции требуют, чтобы конец строки указывался нулевым байтом chr(0).

Например, если L - строка, описанная как

L: packet array [1..81] of char;

то ее "обнуление" выполняется просто, как L[1]:=chr(0), вычисление ее текущей длины выполняется как

```
.........
function LENGTH(L:integer): integer; nonpascal;
.........
begin
.........
    J:=LENGTH(loophole(integer,ref(L)));
.........
end;
```

а сама функция LENGTH может иметь вид:

```
(*nomain*)
type
    STRING = packet array [1..81] of char;
(* верхняя граница не существенна *)
function LENGTH(var S: STRING): integer; external;
function LENGTH;
var
    I : integer;
begin
    I:=0;
    while S[I+1]<>chr(0) do I:=I+1;
    LENGTH:=I
end;
```

Аналогично программируются другие обращения к подпрограммам и функциям обработки строк.

## 2.3. УПРАВЛЕНИЕ ПАМЯТЬЮ

Третья группа примитивов "нижнего" уровня - динамическое управление памятью.

В версии Turbo-Pascal имеются еще две процедуры выделения и освобождения памяти из хипа: GETMEM и FREEMEM. В языке Pascal динамически может выделяться память только под объекты с фиксированными размерами. Во многих прикладных программах было бы удобно динамически выделять память под объекты, размер которых на этапе компиляции не известен, например, под массивы с переменными верхними границами. Все возможные значения верхней границы массивов можно было бы указать с помощью записи с вариантами, однако, это было бы слишком громоздким решением. Процедура GETMEM позволяет программисту управлять объемом выделяемой из хипа памяти. Процедура GETMEM имеет два параметра:

GETMEM (переменная, выражение);

где "переменная" - произвольная переменная-указание, а "выражение" - целое выражение, задающее требуемый размер динамически выделяемой памяти в байтах. Аналогично процедура

FREEMEM (переменная, выражение);

выполняет обратную функцию - возвращает в хип память, выделенную процедурой GETMEM. Значение параметра "выражение" процедуры FREEMEM должно быть в точности равно параметру "выражение" соответствующей процедуры GETMEM.

В версиях Pascal-2, начиная с v2.1, реализованы внешняя функция P$INEW и процедура P$DISPOSE, управляющие выделением и освобождением выделенной памяти из хипа. Здесь построение соответствующего аналога тривиально:

```
(*nomain*)
type
    POINTER = ^integer;
function P$INEW(SIZE: integer): pointer: external;
procedure P$DISPOSE(PTR: pointer; SIZE: integer); external;
procedure GETMEM(var PTR: pointer; SIZE: integer); external;
(* extends new procedure to support allocation of variable
   length aggregates *)
procedure GETMEM;
begin
    PTR:=P$INEW(SIZE)
end;
procedure FREEMEM(P: pointer; SIZE: integer); external;
(* complements GETMEM procedure to support freeing of variable
   length aggregates *)
procedure FREEMEM;
begin
    P$DISPOSE(P, SIZE)
end;
```

В версиях v2.0 реализация функции P%INEW и процедуры P%DISPOSE может выглядеть следующим образом:

```
; function P%INEW(SIZE: integer): pointer; external;
        FUNC    P%INEW,PTR,ADDRESS
        PARAM   SIZE,INTEGER
        BEGIN
        INC     SIZE(SP)
        BIC     #1,SIZE(SP)     ;выделить четное число байтов
        MOV     SIZE(SP),-(SP)
        CALL    %B70
        .GLOBL  %B70
        MOV     (SP)+,PTR(SP)
        ENDPR
        .END

; procedure P%DISPOSE(P: pointer; SIZE: integer); external;
        FUNC    P%DISPOSE
        PARAM   P,ADDRESS
        PARAM   SIZE,INTEGER
        BEGIN
        MOV     SIZE(SP),R0
        INC     R0
        BIC     #1,R0
        MOV     P(SP),-(SP)
        CALL    %B72
        .GLOBL  %B72
        ENDPR
        .END
```

Эти две процедуры выделяются системно независимыми, т.е. справедливы для ОС RT-11/TSX-Plus, RSX-11M, RSTS/E и могут использоваться в любой из указанных ОС при работе с СП Pascal-2.

Полезной может оказаться функция MEMAVAIL, возвращающая разность между вершинами стека и хипа, т.е. объем доступной динамической памяти в байтах. В Pascal-2 эту роль выполняет внешняя функция SPACE

```
; function MEMAVAIL: integer; external;
function SPACE: integer; external;
function MEMAVAIL: integer; external;
(* return value of memory available for allocation in bytes *)
function MEMAVAIL;
begin
    MEMAVAIL:=SPACE
end;
```

Использовать вышеприведенные процедуры можно следующим образом. При входе в процедуру, в которой требуется выделить память под объект, размер которого задается, например, с помощью какого-либо параметра, необходимо вызвать процедуру GETMEM, передав ей в качестве параметров локальную переменную, имеющую тип ссылки на такой объект, и фактический требуемый размер памяти в байтах под этот объект. Обращения к этому объекту после выделения памяти осуществляются с помощью переменной-ссылки. Перед выходом из процедуры выделенная динамически память освобождается с помощью процедуры FREEMEM. При необходимости размещения памяти под несколько подобных объектов следует вызвать процедуру GETMEM соответствующее число раз, а затем столько же раз и процедуру FREEMEM (желательно указать ссылки на объекты в порядке, обратном их порядку при вызовах процедуры

GETMEM). Другим вариантом может быть запоминание адреса вершины хипа перед выделением динамической памяти, а затем освобождение динамической памяти восстановлением старого значения вершины хипа с помощью процедуры RELEASE.

Например, пусть в процедуре A необходимо динамически выделить массив вещественных чисел, размер которого определяется параметром N:

```
type
    ARR = array [1..2] of real;
(* верхняя граница не существенна *)
    PTR = ^ARR;
procedure A(N: integer); external;
procedure GETMEM(var P:PTR; SIZE: integer); external;
procedure FREEMEM(var P:PTR; SIZE: integer); external;
procedure A;
var
    P : PTR;
begin
    GETMEM(P,4*N); (* длина числа типа real = 4 байта *)
.................
    P^[I]:=P^[I-1]+1.0; (* работа с динамическим массивом *)
.................
    FREEMEM(P,4*N)
end;
```

## 2.4. ОРГАНИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Авторы в данной статье не ставили перед собой цель реализовать примитивы "нижнего" уровня, адекватные имеющимся в языке Modula-2, например, процедуры NEWPROCESS и TRANSFER. Это станет предметом следующих работ, в которых помимо СП Turbo-Pascal будет рассмотрена СП MicroPower/Pascal фирмы DEC, также являющаяся мобильной для линии PDP-11 и VAX средством разработки автономных программируемых контроллеров на языке Pascal, включающем поддержку мультипрограммирования. Авторы утверждают, что подобные процедуры могут быть построены и для СП Pascal-2.

# Выводы

Использование СП Pascal-2 позволяет разрабатывать мобильное программное обеспечение для ЭВМ линии PDP-11, VAX и IBM PC, большая часть которого может быть написана на языке Pascal. Примитивы, зависящие от типа операционной системы или версии СП, можно локализовать в отдельные модули, которые и будут подлежать переделке при переносе программного обеспечения из одной среды (ОС или версии СП) в другую. Наличие в СП Pascal-2 средств раздельной компиляции модулей позволяет разрабатывать мобильные средства расширения стандарта языка.

Разработка ПО на едином языке высокого уровня, обладающем явными достоинствами с точки зрения программотехники, может существенно сократить усилия и время коллективов программистов, одновременно гарантируя высокую надежность создаваемого ПО.

## Л И Т Е Р А Т У Р А


1. П а й л Я. АДА - язык встроенных систем. М.: Финансы и статистика, 1984.

2. Язык программирования АДА. ГОСТ 27831-88 (ИСО 8652-87).

3. C o l e m a n D. ADA/Z // Hardcopy, 1985. Vol. 14, N 11.

4. Вирт Н. Программирование на языке Модула-2. М.: Мир, 1987.

5. Biagioni E., Hinrichs K., Heiser G., Muller C. A portable Operating System Interface and Utility Library // IEEE Software, 1986, Vol. 3, N 6.

6. Craig J.M., Martel J.M., Richan K.B. Design of a Modula-2 Standard Library // J. Pascal, ADA & Modula-2, 1985. Vol. 4, N 2.

7. Cornelius B.J. Problems with the Language Modula-2, Software // Pract. and Exper., 1988, Vol. 18, N 6.

8. Болски М.И. Язык программирования СИ: Справочник. М.: Радио и связь, 1988.

9. Керниган Б., Ритчи Д. Язык программирования СИ. Задачи по языку СИ. М.: Финансы и статистика, 1985.

10. Abrahams P.W. Some Sad Remarks about String Handling in C. // Sigplan Not, 1988, Vol. 13, N 10.

11. Pohl I., Edelson D. A to Z: C Language Shortcommings // Comput. Lang, 1988, Vol. 13, N 2.

12. Брусиловский Л.И., Михайлов Ю.А. Организация взаимодействия межъязыковых модулей (Pascal-2 - Fortran-4) в операционных системах RSX-11M и RT-11: // Компьютерная оптика: Сборник / МЦНТИ, М., 1989, Вып. 4.

13. Hume J.N.P., Holf R.C. VAX Pascal, Reston: Reston Publ. Co. 1983.

14. Pascal-2. Version 2.1 for RSX-11. Oregon MiniSoftware Inc., 1983.

15. Вирт Н., Йенсен К. Паскаль: Руководство для пользователя. М.: Финансы и статистика, 1989.

16. Wood S. Using Turbo-Pascal: Covers Version 3.0, Berkely: McBraw-Hill, 1986.

17. Михайлов Ю.А. Программирование системных вызовов на Pascal-2 в ОС РАФОС и ДОС КП // Управляющие системы и машины, 1989. N 5.