EDUCAÇÃO e TECNOLOGIA

Revista do Instituto Politécnico da Guarda

"EDUCAÇÃO E TECNOLOGIA"
Revista do Instituto Politécnico da Guarda

DIRECTOR: João Bento Raimundo

REDACÇÃO: Rua Comandante Salvador do Nascimento
Telef. 21634 6300 GUARDA

PROPRIEDADE: Instituto Politécnico da Guarda

EXECUÇÃO GRÁFICA: Secção de Reprografia do IPG

Depósito Legal N.º 17.891/87

Nº 3 / Julho / 88

"E HOJE É JÁ OUTRO DIA"

E hoje é já outro dia. Certo. Real. Grande.

Caminhou-se da expectativa, da aposta e da incerteza para a realidade do conseguido.

O Instituto Politécnico da Guarda tomou uma maior dimensão. Ganhou o seu espaço próprio; arreigou-se no meio físico, social e intelectual; impôs-se como centro de saber, pensar e de fazer. O Instituto Politécnico da Guarda corresponde já às expectativas daqueles que o justificam - os estudantes. Por isso se tornou grande. Constroi-se hoje o amanhã que não tarda.

"*Educação e Tecnologia*" é hoje, no final de mais um ano lectivo – testemunho precioso de uma realidade pautada pela dinâmica que é também o apanágio desta Escola. E porque emerge do centro da vida do Instituto Politécnico da Guarda reflecte-a, naturalmente, também na sua autenticidade social e académica. Como espaço aberto, é dinâmica. Porque é dinâmica, é variada e polivalente. Pretendíamos que o fosse; sabemos que é. Estamos certos que continuará a sê-lo.

"E outra vez conquistemos a distância --
Do mar ou outra, mas que seja nossa"

(Fernando Pessoa)

João Bento Raimundo
Presidente da C.I. do I.P.G.

UTILIZAÇÃO DO ISAM EM BASIC/MS - DOS

Por: Álvaro Bento Leal: Prof. Coordenador do I.P.G.

Resumo: É apresentada a programação BASICA que permite realizar as operações típicas do método de acesso sequencial indexado, sob o sistema operativo MS - DOS.

O acesso directo ao registo de um ficheiro que possue determinado valor de um campo, a chave e, simultaneamente, a possibilidade de aceder aos registos pela sequência de ordenação das chaves, isto é, o método de acesso sequencial indexado (ISAM), é extremamente útil na programação de aplicações de processamento de dados.

A não inclusão deste método de acesso nas linguagens mais utilizadas nos PC, obriga a uma programação especifica de forma a colmatar a lacuna. O que aqui irá ser apresentado é o sistema que o autor tem utilizado em diversas aplicações e que se tem revelado eficiente no tratamneto de ficheiros com um número de registos até à ordem dos 5000, ocupando áreas da ordem dos 400 KB.

Existe uma enorme variedade de formas de organizar os dados de molde a serem implantadas as funções características do método ISAM. Duma maneira geral, pode dizer-se que o ficheiro a ser tratado, ficheiro base, é armazenado em simultâneo com o ficheiro índice, ficheiro este que possue a informação sobre a localização no ficheiro base dos registos em correspondência com as chaves. As diferenças entre os ISAM residem fundamentalmente na estruturação do ficheiro índice e na solução do problema da adição de novos registos ao ficheiro base.

A tendência actual é a de dar ao ficheiro índice uma estrutura em árvore do tipo "B - tree", com a qual o problema da adição de novos registos é eficazmente resolvido. No caso presente adoptou-se uma estrutura do tipo de lista ordenada, usando uma lista auxiliar para armazenamento temporário das adições. Cada elemento da lista índice e lista auxiliar é formado por dois

campos: a chave e o índice que, é o número do registo do ficheiro base que possue a respectiva chave.

Se bem que a solução do problema das adições não seja tão eficaz como no caso das "B - tree", isto não é inconveniente de maior no caso de um ambiente de um só utilizador como acontece nos PC. A solução adoptada apresenta por outro lado vantagens resultantes da sua simplicidade: uma menor ocupação de área e uma mais fácil compreensão dos detalhes por parte do programador e daí um melhor controlo na utilização.

A eficiência do ISAM é a eficiência do acesso aos dados no ficheiro índice e neste aspecto o primeiro factor é a utilização tanto quanto possível da memória interna. Nos casos de maior interesse prático, ficheiros com alguns milhares de registos, a área total do ficheiro índice raramente ultrapassa os 100 KB, o que, nos PC, é perfeitamente comportável pela memória interna. Daí que se tenha adoptado a solução de armazenar integralmente o ficheiro índice em memória, obviamente mantendo sempre uma cópia em memória externa que é actualizada no final de cada processamento.

Quando se afirmou que nos PC a área disponível era suficiente para comportar todo o ficheiro índice, tal não significa que os dados sejam armazenados na área do programa, porque as linguagens impõe muitas vezes limites mais estreitos para a dimensão da área utilizável. Há que recorrer então à utilização do conceito de disco virtual, o que implica a utilização de uma versão 3. * ou superior do MS - DOS. Em termos práticos o problema fica resolvido definindo no ficheiro CONFIG . SYS um "device" com área suficiente e incluindo no AUTOEXEC. BAT uma instrução COPY que copie o ficheiro índice armazenado externamente para o disco virtual.

O 2º factor de eficiência no tratamento do ficheiro índice é o algoritimo de pesquisa usado, este algoritimo é dependente da estrutura de dados adoptada, no caso presente de uma lista ordenada a solução mais eficiente é a da pesquisa binária, pelo que foi a usada.

Quanto à solução do problema das adições adoptou-se a utilização de uma lista não ordenada onde as novas chaves e respectivos índices vão ocupando posições sucessivas pela ordem de entrada. Esta lista ocupa, por encosto, os últimos registos do ficheiro índice e pode conter um número máximo de 200 elementos, valor este que pode ser reconfigurado se houver necessidade. Quando o número de adições atinge o valor máximo ou, e sempre, no final da execução de cada programa, a lista auxiliar é ordenada em memória e é criado novo ficheiro índice por intercalação ("merge") do ficheiro inicial com a lista que contem as adições. O algoritmo usado na ordenação é o "quicksort" com recorrência ao algoritmo de inserção sempre que o número de

elementos dos subconjuntos que surgem no desenvolvimento for menor ou igual a 16.

O texto das subrotinas que formam o sistema é apresentado no Apêndice em linguagem BASICA.

Dado o caracter não local das subrotinas BASICA, as mesmas são dependentes do programa que as utiliza, pelo que haverá necessidade de algumas adaptações em cada caso particular.

A subrotina 20000 tem que ser adaptada, assim como as subrotinas 21000 e 21100, das restantes instruções somente as assinaladas com o comentário ' nnn, necessitarão de uma eventual adaptação. Para uma melhor compreensão do texto das subrotinas imaginem-se, na forma como estão apresentadas, aplicadas ao caso de um ficheiro designado por STOCKS com registos formados pelos seguintes campos:

```
 1 -  7 : SCOD$    - Código do produto
 8 - 37 : SNOME$   - Designação do produto
38 - 38 : SIVA$    - Código IVA
39 - 42 : SMIN     - Stock mínimo
43 - 46 : SMAX     - Stock máximo
47 - 50 : SQEX     - Quantidade existente
51 - 54 : SPRV     - Preço unitário
55 - 58 : SACM     - Consumo acumulado
```

O ficheiro índice com o nome INDSTO na versão armazenada em disco e D: VRTSTIN na versão em disco virtual, possue registos formados pelo campo chave, isto é, o valor de SCOD$ e o campo índice, isto é, o número do registo de STOCKS onde reside o registo com a correspondente chave.

Os programas que utilizam as subrotinas devem actuar somente nas variáveis que definem os registos de STOCK, isto é, as iniciadas pela letra S e na variável KM$, onde será colocado o valor de uma chave que se pretende pesquisar. A variável TESTE é devolvida pelas subrotinas com um valor que dá a indicação de condições especiais encontradas na execução. As restantes variáveis, com letra inicial Z, são controladas pelas subrotinas e não deverão ser, em princípio, alteradas pelo programa principal.

As subrotinas que realizam o conjunto de funções que formam o ISAM são as seguintes:

- *Definição dos parâmetros* (subortina 20000)
  Não executa propriamente qualquer função de acesso e pode estar englobada no início do programa principal. A finalidade é somente definir valores específicos de cada aplicação, antes da chamada de qualquer das restantes subrotinas.

- *Abertura dos ficheiros e inicialização* (subrotinas 20200)
  Deve ser executada no início do programa e eventual-

mente após a subrotina de fecho para utilizar de novo o ISAM.

Além de realizar a abertura dos ficheiros, inicializa os apontadores de leitura sequencial por ordem ascendente e descendente, respectivamente, no início e no fim do ficheiro de índice.

- *Acesso directo por chave* (subrotina 20400)

Recebe na variável KM$ o valor da chave a pesquisar. Por pesquisa binária averigua da existência, em D:VRTSTIN, do registo com essa chave.

No caso de KM$ aí não residir, procura na lista das chaves adicionadas durante o processamento, por pesquisa sequencial.

Se for encontrado o registo com chave KM$, o registo base respectivo é acedido e os campos são transferidos do "buffer" para a área de dados, subrotina 21000, a variável TESTE é devolvida com o valor 0 e os apontadores de leitura sequencial ficam localizados no registo encontrado.

No caso da pesquisa não ser bem sucedida a variável TESTE é devolvida com um valor ≠0 e o acesso sequencial ascendente que se seguir fornece o registo com chave imediatamente inferior ou, no caso do acesso descendente, o registo com chave imediatamente superior.

- *Reescrita de registo no ficheiro base* (subrotina 21200)

Quando houver necessidade de alterar o conteúdo de campos, excepto a chave, num registo do ficheiro base, deverá primeiramente aceder-se, directa ou sequencialmente, ao registo e, após a alteração dos valores dos campos lidos, rescrever o registo usando esta subrotina.

A subrotina 21100 usada nesta operação tem por finalidade tranferir os dados da área de programa para o "buffer".

- *Eliminação de registos* (subrotina 21300)

Um registo considera-se eliminado quando o índice associado à respectiva chave é feito igual a 0.

No ficheiro base o registo permanecerá, simplesmente deixa de haver a possibilidade de ser acedido.

Por conveniência de uma eventual recuperação ou reorganização dos registos no ficheiro base, há interesse em marcar os registos eliminados com um código convencional. É o que é feito pela instrução 21330, onde ao campo SIVA$ é atribuído o valor "D".

- *Leitura sequencial por ordem ascendente* (subrotina 21500)

Faz avançar o apontador ZJJS% uma posição e lê o correspondente registo base.

A variável TESTE é devolvida com o valor 1, se o fim do ficheiro índice for atingido e, com o valor 0, nos restantes casos.

Notar que os registos adicionados no mesmo processamento não são acedidos sequencialmente. Se houver necessidade desse acesso é necessário proceder ao fecho seguido de nova abertura dos ficheiros.

- *Leitura sequencial por ordem descendente* (subrotina 21700)

Em tudo análogo ao que foi dito no caso anterior, excepto no que se refere à ordem das chaves.

- *Adição de registos* (subrotina 21900)

Faz avançar o apontador, ZKADD% uma posição e introduz no registo ZNTIN% + ZKADD% de D: VRTSTIN os valores KM$ e ZKRMX% que é o número do registo do ficheiro base disponível para receber a informação a introduzir.

ZKRMX% avança uma posição por cada adição.

O ficheiro base é escrito com o valor dos campos definidos no programa de chamada.

Quando o número de adições atinge 200 é invocada a subrotina de fecho e reorganização seguida de nova abertura dos ficheiros.

- *Fecho e reorganização de índices* (subrotina 22100)

Tem como finalidade reorganizar o ficheiro de índice, copiar o novo índice para memória externa e fechar os ficheiros.

A reorganização, quando necessária, é iniciada pela ordenação da área de adições (subrotina 23000). As adições ordenadas são intercaladas com o ficheiro de índice existente, criando-se em disco um novo ficheiro de índice. Nesta fase os registos índice eliminados são excluídos. Seguidamente o novo ficheiro de índice é copiado para o disco virtual, ficando desta forma garantida a permanência em disco virtual de versão mais actualizada do ficheiro índice para os processamentos futuros.

As regras a respeitar de forma e não deparar com situações anormais na utilização, podem resumir-se no seguinte:

- Garantir a existência do ficheiro índice em disco virtual no início de cada sessão de trabalho. Como foi dito,o mais prático, é incluir a cópia do ficheiro em disco para disco virtual no AUTOEXEC. BAT.
- Iniciar o processamento pela subrotina 20000, seguida da chamada da subrotina 20200.

- Não efectuar adições de novos registos sem previamente, pela subrotina 20400, averiguar a não existência dos mesmos, sob pena de introduzir chaves em duplicado.
- No caso da alteração ou eliminação de registos, seguir sempre a ordem: leitura (directa ou sequencial) com resultado TESTE= 0, alteração dos campos do registo e só depois a reescrita ou a eliminação, sem a introdução de quaisquer outras operações ISAM entre as que foram indicadas.

Um cancelamento anormal do programa terá como resultado a perda dos registos introduzidos durante o processamento interrompido.

A recuperação do ficheiro de índice é possível por leitura directa do ficheiro base com introdução progressiva dos registos lidos, usando o ISAM, num novo ficheiro base e correspondente ficheiro índice.

APÊNDICE

```
20000 '
20010 ' Definiçao de parâmetros. Dependente de cada aplicaçao especifica
20020 '
20030 DIM ZK1K$(200),ZK1A%(200),ZE%(50),ZD%(50)       ' Agrupamentos de trabalho
20040 DIM ZVB$(8),ZLB%(8)             ' Variáveis p/definição do buffer de ZBASE$
20050 ZBASE$="STOCKS"                   ' Nome do ficheiro base
20060 ZINDE$="INDSTO"                   ' Nome do ficheiro de índice
20070 ZVIRT$="D:VRTSTIN"                ' Nome do ficheiro de índice virtual
20080 ZLB%(1)=7: ZLB%(2)=30: ZLB%(3)=1      ' ZLB%() definem a compartimentaçao
20090 ZLB%(4)=4: ZLB%(5)=4: ZLB%(6)=4       ' do buffer de ZBASE$
20100 ZLB%(7)=4: ZLB%(8)=4
20110 ZLKST%=7                          ' Comprimento da chave
20120 ZF1%=1                            ' Número do ficheiro ZBASE$
20130 ZF2%=2                            ' Número do ficheiro ZVIRT$
20140 ZF3%=3                            ' Número do ficheiro auxiliar
20150 RETURN
20200 '
20210 ' Abertura dos ficheiros e inicializaçao
20220 '
20230 ZLREC%=0
20240 FOR ZI1%=1 TO 8: ZLREC%=ZLREC%+ZLB%(ZI1%): NEXT ZI1%                ' nnn
20250 OPEN "R",ZF1%,ZBASE$,ZLREC%
20260 FIELD ZF1%,ZLB%(1) AS ZVB$(1),ZLB%(2) AS ZVB$(2),ZLB%(3) AS ZVB$(3),ZLB%(4
) AS ZVB$(4),ZLB%(5) AS ZVB$(5),ZLB%(6) AS ZVB$(6),ZLB%(7) AS ZVB$(7),ZLB%(8) AS
 ZVB$(8)                                                                ' nnn
20270 OPEN "R",ZF2%,ZVIRT$,ZLKST%+2
20280 FIELD ZF2%,ZLKST% AS ZMAT$,2 AS ZRG$
20290 ZKADD%=0                              ' ZKADD% num. chaves adicionadas
20300 ZKDEL%=0                              ' ZKDEL% num. chaves eliminadas
20310 ZKRMX%=LOF(ZF1%)/ZLREC%               ' ZKRMX% num. registos em ZBASE$
20320 ZNTIN%=LOF(ZF2%)/(ZLKST%+2)           ' ZNTIN% num. registos em ZVIRT$
20330 ZJJP%=0                           ' ZJJP% último registo de ZVIRT$ acedido
20340 ZJJR%=0                           ' ZJJR% último registo de ZBASE$ acedido
20350 ZJJS%=0                           ' ZJJS% controla acesso sequen. ascende.
20360 ZJJT%=ZNTIN%+1                    ' ZJJT% controla acesso sequen. descend.
20370 RETURN
20400 '
20410 ' Acesso por chave
```

```
20420 '
20430 TESTE=1                                   ' TESTE indicador de condição
20440 ZJJS%=0: ZJJT%=0
20450 IF ZNTIN%=0 THEN 20720
20460 GET ZF2%,1                                ' Análise do registo com menor chave
20470 IF ZMAT$>KM$ THEN 20720
20480 ZA1%=1
20490 IF ZMAT$<KM$ THEN 20600
20500 ZJJR%=CVI(ZRG$)
20510 IF ZJJR%>0 THEN 20550
20520 TESTE=2                                   ' O registo com chave KM$ está anulado
20530 ZJJP%=0: ZJJR%=0: ZJJS%=ZA1%: ZJJT%=ZA1%
20540 RETURN
20550 TESTE=0                                   ' O registo tem chave = KM$
20560 ZJJP%=ZA1%: ZJJS%=ZA1%: ZJJT%=ZA1%
20570 GET ZF1%,ZJJR%
20580 GOSUB 21000                               ' Transferência buffer->data
20590 RETURN
20600 GET ZF2%,ZNTIN%                           ' Análise do registo com maior chave
20610 ZA1%=ZNTIN%
20620 IF ZMAT$<KM$ THEN 20710
20630 IF ZMAT$=KM$ THEN 20500
20640 ZI1%=1: ZM1%=ZNTIN%                       ' Inicio da pesquisa binária
20650 WHILE (ZM1%-ZI1%)>1
20660 ZA1%=INT((ZI1%+ZM1%)/2)
20670 GET ZF2%,ZA1%
20680 IF ZMAT$=KM$ THEN 20500
20690 IF ZMAT$>KM$ THEN ZM1%=ZA1% ELSE ZI1%=ZA1%
20700 WEND
20710 ZJJS%=ZA1%: ZJJT%=ZA1%+1
20720 IF ZKADD%=0 THEN RETURN
20730 FOR ZI1%=ZNTIN%+1 TO ZNTIN%+ZKADD%
20740 GET ZF2%,ZI1%
20750 IF ZMAT$=KM$ THEN 20790
20760 NEXT ZI1%
20770 ZJJR%=0: ZJJS%=0
20780 RETURN
20790 ZJJR%=CVI(ZRG$)
20800 IF ZJJR%>0 THEN 20840
20810 TESTE=2
20820 ZJJP%=0: ZJJR%=0
20830 RETURN
20840 ZJJP%=ZI1%
20850 GET ZF1%,ZJJR%
20860 GOSUB 21000
20870 TESTE=0
20880 RETURN
21000 '
21010 ' Transferência dos dados do buffer para as variáveis do programa
21020 '
21030 SCOD$=ZVB$(1): SNOME$=ZVB$(2): SIVA$=ZVB$(3)                    ' nnn
21040 SMIN=CVS(ZVB$(4)): SMAX=CVS(ZVB$(5))                            ' nnn
21050 SQEX=CVS(ZVB$(6)): SPRV=CVS(ZVB$(7)): SACM=CVS(ZVB$(8))         ' nnn
21060 RETURN
21100 '
21110 ' Transferência da variáveis do programa para o buffer
21120 '
21130 LSET ZVB$(1)=SCOD$: LSET ZVB$(2)=SNOME$: LSET ZVB$(3)=SIVA$     ' nnn
21140 LSET ZVB$(4)=MKS$(SMIN): LSET ZVB$(5)=MKS$(SMAX)                ' nnn
21150 LSET ZVB$(6)=MKS$(SQEX): LSET ZVB$(7)=MKS$(SPRV)                ' nnn
21160 LSET ZVB$(8)=MKS$(SACM)                                         ' nnn
21170 RETURN
21200 '
21210 ' Reescrita do ficheiro ZBASE$
21220 '
```

```
21230 GOSUB 21100                       ' Transferência das variáveis p/ buffer
21240 PUT ZF1%,ZJJR%
21250 RETURN
21300 '
21310 ' Eliminação de registo
21320 '
21330 SIVA$="D"                                                    ' nnn
21340 GOSUB 21200                       ' Reescreve ZBASE$
21350 LSET ZRG$=MKI$(0)
21360 PUT ZF2%,ZJJP%
21370 ZKDEL%=ZKDEL%+1
21380 RETURN
21500 '
21510 ' Leitura sequencial por ordem ascendente
21520 '
21530 TESTE=1
21540 ZJJS%=ZJJS%+1
21550 IF ZJJS%>ZNTIN% THEN ZJJS%=0      ' Se o fim de ficheiro é atingido
21560 IF ZJJS%=0 THEN RETURN            ' TESTE=1 e posiciona-se no início
21570 GET ZF2%,ZJJS%
21580 ZA1%=CVI(ZRG$)
21590 IF ZA1%=0 THEN 21540              ' Testa registos eliminados
21600 ZJJP%=ZJJS%: ZJJR%=ZA1%: ZJJT%=ZJJS%
21610 GET ZF1%,ZJJR%
21620 GOSUB 21000
21630 TESTE=0
21640 RETURN
21700 '
21710 ' Leitura sequencial por ordem descendente
21720 '
21730 TESTE=1
21740 ZJJT%=ZJJT%-1
21750 IF ZJJT%<=0 THEN ZJJT%=ZNTIN%+1  ' Se é atingido o início do ficheiro
21760 IF ZJJT%>ZNTIN% THEN RETURN       ' TESTE=1 e posiciona-se no fim
21770 GET ZF2%,ZJJT%
21780 ZA1%=CVI(ZRG$)
21790 IF ZA1%=0 THEN 21740              ' Ignoram-se os registos eliminados
21800 ZJJP%=ZJJT%: ZJJR%=ZA1%: ZJJS%=ZJJT%
21810 GET ZF1%,ZJJR%
21820 GOSUB 21000
21830 TESTE=0
21840 RETURN
21900 '
21910 ' Adição de registos
21920 '
21930 IF ZKADD%<200 THEN 21960
21940 GOSUB 22100                       ' Se o número de adições =200, há que
21950 GOSUB 20200                       ' reorganizar o ficheiro de índices
21960 ZKRMX%=ZKRMX%+1
21970 ZKADD%=ZKADD%+1
21980 LSET ZMAT$=KM$: LSET ZRG$=MKI$(ZKRMX%)
21990 ZJJP%=ZNTIN%+ZKADD%
22000 PUT ZF2%,ZJJP%
22010 ZJJR%=ZKRMX%
22020 GOSUB 21100
22030 PUT ZF1%,ZJJR%
22040 RETURN
22100 '
22110 ' Fecho com reorganização dos índices se necessário
22120 '
22130 IF ZKADD%<>0 OR ZKDEL%<>0 THEN 22160
22140 CLOSE ZF1%,ZF2%                   ' Não é necessário reorganizar
22150 RETURN
22160 ZA1%=0: ZHIG$=STRING$(ZLKST%,CHR$(255))
22170 IF ZKADD%=0 THEN 22250
```

```
22180 FOR ZI1%=ZNTIN%+1 TO ZNTIN%+ZKADD%          ' Os registos de ZVIRT$ adicio-
22190 GET ZF2%,ZI1%                               ' nados são lidos para os vec-
22200 IF CVI(ZRG$)=0 THEN 22230                   ' tores auxiliares
22210 ZA1%=ZA1%+1
22220 ZK1K$(ZA1%)=ZMAT$: ZK1A%(ZA1%)=CVI(ZRG$)
22230 NEXT ZI1%
22240 IF ZA1%>1 THEN GOSUB 23000                  ' Ordenaçao
22250 OPEN "R",ZF3%,"76UC4569",ZLKST%+2           ' Ficheiro auxiliar
22260 FIELD ZF3%,ZLKST% AS ZBK$,2 AS ZBR$
22270 ZI1%=1: ZI2%=1: ZKADD%=ZA1%: ZA1%=0         ' Inicio da intercalação
22280 IF ZI1%>ZNTIN% THEN 22340
22290 GET ZF2%,ZI1%
22300 ZRG1%=CVI(ZRG$)
22310 IF ZRG1%>0 THEN 22330
22320 ZI1%=ZI1%+1: GOTO 22280                     ' Ignora registos eliminados
22330 ZKK1$=ZMAT$: GOTO 22350
22340 ZKK1$=ZHIG$
22350 IF ZI2%>ZKADD% THEN 22380
22360 ZKK2$=ZK1K$(ZI2%): ZRG2%=ZK1A%(ZI2%)
22370 GOTO 22390
22380 ZKK2$=ZHIG$
22390 WHILE ZKK1$<>ZHIG$ OR ZKK2$<>ZHIG$          ' Intercalação
22400 IF ZKK1$>ZKK2$ THEN 22500
22410 LSET ZBK$=ZKK1$: LSET ZBR$=MKI$(ZRG1%)
22420 ZA1%=ZA1%+1: PUT ZF3%,ZA1%
22430 ZI1%=ZI1%+1
22440 IF ZI1%>ZNTIN% THEN 22490
22450 GET ZF2%,ZI1%
22460 ZRG1%=CVI(ZRG$)
22470 IF ZRG1%=0 THEN 22430
22480 ZKK1$=ZMAT$: GOTO 22560
22490 ZKK1$=ZHIG$: GOTO 22560
22500 LSET ZBK$=ZKK2$: LSET ZBR$=MKI$(ZRG2%)
22500 LSET ZBK$=ZKK2$: LSET ZBR$=MKI$(ZRG2%)
22510 ZA1%=ZA1%+1: PUT ZF3%,ZA1%
22520 ZI2%=ZI2%+1
22530 IF ZI2%>ZKADD% THEN 22550
22540 ZKK2$=ZK1K$(ZI2%): ZRG2%=ZK1A%(ZI2%): GOTO 22560
22550 ZKK2$=ZHIG$
22560 WEND
22570 CLOSE ZF1%,ZF2%,ZF3%
22580 KILL ZINDE$                                 ' O ficheiro ZINDE$ existente no
22590 NAME "76UC4569" AS ZINDE$                   ' inicio é substituido p/auxiliar
22600 GOSUB 22700                                 ' Cópia de ZINDE$ p/ ZVIRT$
22610 RETURN
22700 '
22710 ' Cópia de ZINDE$ para ZVIRT$
22720 '
22730 OPEN "R",ZF3%,ZINDE$,ZLKST%+2
22740 FIELD ZF3%,ZLKST% AS ZBK$,2 AS ZBR$
22750 KILL ZVIRT$                                 ' Elimina ZVIRT$ anterior
22760 OPEN "R",ZF2%,ZVIRT$,ZLKST%+2
22770 FIELD ZF2%,ZLKST% AS ZMAT$,2 AS ZRG$
22780 ZA1%=LOF(ZF3%)/(ZLKST%+2)
22790 FOR ZI1%=1 TO ZA1%
22800 GET ZF3%,ZI1%
22810 LSET ZMAT$=ZBK$: LSET ZRG$=ZBR$
22820 PUT ZF2%,ZI1%
22830 NEXT ZI1%
22840 CLOSE ZF2%,ZF3%
22850 RETURN
23000 '
23010 ' Ordenação. Algoritmo QUICKSORT
23020 '
23030 ZP%=1: ZE%(1)=1: ZD%(1)=ZA1%
```

```
23040 WHILE ZP%>0
23050 ZLI%=ZE%(ZP%): ZRI%=ZD%(ZP%): ZP%=ZP%-1
23060 IF ZRI%-ZLI%>=16 THEN 23090
23070 ZI1%=ZLI%: ZI2%=ZRI%: GOSUB 23300            ' Conjuntos c/num. <=16 são
23080 GOTO 23260                                   ' ordenados p/ inserção
23090 ZIM%=(ZLI%+ZRI%)/2
23100 ZC$=ZK1K$(ZIM%)
23110 ZI%=ZLI%-1: ZJ%=ZRI%+1
23120 ZI%=ZI%+1
23130 IF ZK1K$(ZI%)<ZC$ THEN 23120
23140 ZJ%=ZJ%-1
23150 IF ZK1K$(ZJ%)>ZC$ THEN 23140
23160 IF ZI%>ZJ% THEN 23200
23170 ZT$=ZK1K$(ZI%): ZK1K$(ZI%)=ZK1K$(ZJ%): ZK1K$(ZJ%)=ZT$
23180 ZY%=ZK1A%(ZI%): ZK1A%(ZI%)=ZK1A%(ZJ%): ZK1A%(ZJ%)=ZY%
23190 GOTO 23120
23200 ZP%=ZP%+1
23210 IF ZI%>=ZIM% THEN 23240
23220 ZE%(ZP%)=ZI%: ZD%(ZP%)=ZRI%: ZRI%=ZI%-1
23230 GOTO 23060
23240 ZE%(ZP%)=ZLI%: ZD%(ZP%)=ZI%-1: ZLI%=ZI%
23250 GOTO 23060
23260 WEND
23270 RETURN
23300 '
23310 ' Ordenação. Algoritmo da inserção
23320 '
23330 FOR ZI%=ZI1%+1 TO ZI2%
23340 ZC$=ZK1K$(ZI%): ZB%=ZK1A%(ZI%)
23350 ZJ%=ZI%-1
23360 WHILE ZJ%>0
23370 IF ZC$>ZK1K$(ZJ%) THEN 23410
23380 ZK1K$(ZJ%+1)=ZK1K$(ZJ%): ZK1A%(ZJ%+1)=ZK1A%(ZJ%)
23390 ZJ%=ZJ%-1
23400 WEND
23410 ZK1K$(ZJ%+1)=ZC$: ZK1A%(ZJ%+1)=ZB%
23420 NEXT ZI%
23430 RETURN
```