EDUCAÇÃO e TECNOLOGIA

Revista do Instituto Politécnico da Guarda

"EDUCACÃO 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

COMPOSIÇÃO E IMPRESSÃO: Secção do Reprografia do IPG

N.º 1 / Julho 1987

ESPAÇO DE INFORMAÇÃO E REFLEXÃO

Tudo temos feito para que o Instituto Politécnico da Guarda assuma a sua verdadeira dimensão de pólo dinamizador no contexto sócio-educativo e cultural da região. Para tal, não contam as iniciativas isoladamente, mas enquanto vertentes daquela mesma dimensão.

A informação, encarada a vários níveis, assume importância primordial — no seio do próprio Instituto, retratando a realidade em que se insere, projectando nela a sua própria dinâmica.

Porque existe para servir, o Instituto Politécnico da Guarda quer servir da forma mais adequada — um compromisso entre a realidade que é, a que queremos ter e a que é possível, em função de condicionalismos que tantas vezes transcendem a própria vontade.

Temos igualmente a consciência de que, em matéria de educação e de saber, nunca haverá obra acabada, mas um contínuo fluir; diremos que a obra nasce e, através de múltiplas formas de transformação, cresce.

Para tal é necessário o esforço de muitos, preferencialmente de todos — os que estão verdadeiramente empenhados no progresso e na modernização da sociedade.

Vários são os graus de responsabilidade no processo.

Várias são as formas de influenciar as decisões.

Várias são as estratégias para que se conclua sobre o que deve ser feito e como.

Está criado o espaço aberto de informação, de reflexão, de troca de experiências. "Educação e Tecnologia" é mais uma obra, ou melhor, mais uma vertente da obra que se pretende seja o I.P.G. na sua globalidade.

Professores, alunos e comunidade têm nela o seu espaço. A capacidade para dialogar, a coragem para expressar opiniões, a humildade para ouvir críticas construtivas, a vontade, enfim, para apresentar o melhor, da melhor forma, que pode ser, tão só, o possível, farão de "Educação e Tecnologia" uma verdadeira "obra" de todos.

João Bento Raimundo

Presidente da C.I. do Instituto Politécnico da Guarda

UMA ESTRUTURA DE DADOS PARA ACESSO A REGISTOS COM ATRIBUTOS DE OCORRÊNCIAS MÚLTIPLAS.

Álvaro Bento Leal, Prof. Coordenador do I.P.G.

Resumo: Mostra-se que as estruturas encadeadas permitem a elaboração fácil e eficiente de um método de pesquisa de atributos em correspondência (n,n) com os resgitos de um ficheiro base.
São descritos os algoritmos que permitem uma fácil programação do método.

1. INTRODUÇÃO

Para uma melhor situação do problema, considera-se o caso da informatização de uma biblioteca.

A informacão sobre os livros encontra-se num ficheiro de acesso directo em que, cada registo agrupa os atributos (campos) do livro que, é identificado pela posição relativa que o registo ocupa no ficheiro.

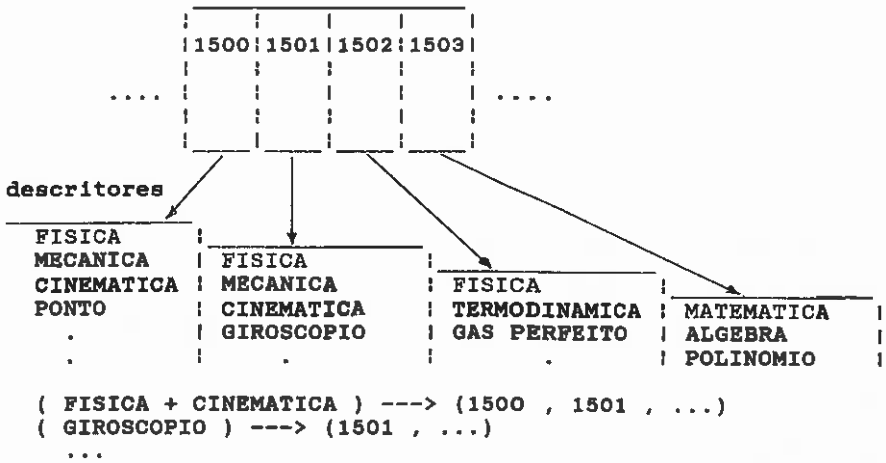
A pesquisa do valor de um campo para determinado valor do identificador do registo de um livro, por exemplo, a designação, o autor, etc, é imediata.

A pesquisa inversa, isto é, a determinação dos livros que possuem determinado valor de um atributo, por exemplo, os livros cujo editor é x, levanta alguns problemas.

O caso em que um livro possui um e um só valor do atributo é resolvido, na maioria dos casos, pela utilização directa dos procedimentos incluidos no sistema operativo: ordenação, criação de indices, etc.

Atenda-se, no entanto, ao seguinte caso:

Cada livro é classificado por um conjunto de descritores (palavras-chave) que identificam o conteúdo do livro. Pretende-se armazenar esta informação de forma que, posteriormente, seja possível identificar os livros que possuem em comum um ou mais desses descritores.



Uma pesquisa deste tipo é suportada por muitos dos sistemas de base de dados. No armazenamento clássico em ficheiros separados, em muitos casos o único existente, o problema levanta algumas dificuldades.

A introdução dos descritores no registo do livro respectivo se bem que, cumpra o princípio da coerência da informacão, é ineficiente sob vários aspectos:

— a duplicação do mesmo descritor em vários registos ocasiona um mau aproveitamento do espaço em disco;

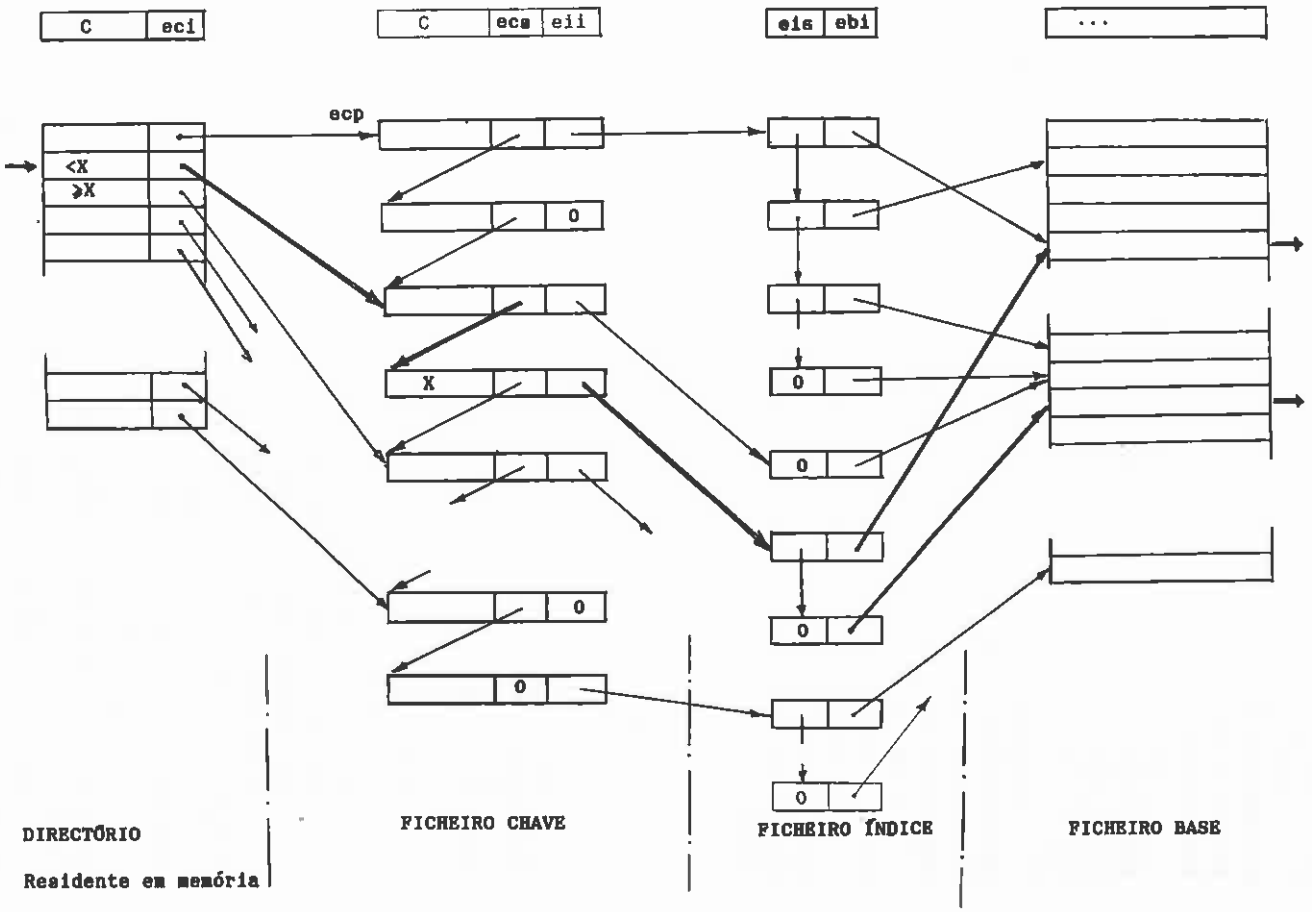
— dado que, o número de descritores é variável de livro para livro, somente a utlização de registos de comprimento variável permitiria que a maioria dos registos não contivesse espaço inútil, os registos de comprimento variável não são suportados por muitos sistemas operativos e quando o são, em muitos casos, o acesso indexado não é permitido a tais registos;

— sobrepondo-se aos inconvenientes apontados, o acesso aos descritores para a consulta, nos moldes indicados, torna-se difícil, somente à custa de um processamento global do ficheiro seria possível obter a informação.

No que se segue mostra-se que a utilização de ficheiros com estrutura encadeada, permite resolver eficientemente o problema, não sendo obrigatória a recorrência a sistemas gerais de gestão de base de dados.

2. ESTRUTURA DE DADOS

Na página seguinte está representada esquematicamente a estrutura geral de dados que é adoptada.


C
eci
C
ecs
eii
eis
ebi
...
ecp
<X
≥X
0
X
0
0
0
0
0
DIRECTÓRIO
Residente em memória
FICHEIRO CHAVE
FICHEIRO ÍNDICE
FICHEIRO BASE

As chaves, referidas como descritores em 1., são agrupadas no ficheiro chave, com estrutura ordenada encadeada simples.

Cada registo chave contém o valor C da chave, um número ecs que indica o registo que imediatamente o segue na ordenação ascendente por chave e, um número eii que referencia o primeiro registo no ficheiro índice que contém a informacão sobre um registo base associado à chave C.

Com o objectivo de reduzir o tempo de pesquisa, é reservada uma área de memória interna para o armazenamento do directório. O directório consiste numa lista criada em cada processamento e, formada pelas chaves e endereços do ficheiro chave de um conjunto de registos distanciados de determinado intervalo, por exemplo: 1, 5, 10, ...ou 1, 7, 14, ..., etc.

O ficheiro índice é formado por sequências encadeadas de registos contendo dois endereços. O primeiro eis representa o registo índice que se segue na cadeia, é zero se for o último de cada cadeia. O segundo endereço ebi é o número de um registo base relacionado com chave à qual o índice está subordinado.

Como se verifica pelo esquema apresentado a pesquisa chave - (registo base, ...) é de realização simples.

O método será eficiente se a manutenção dos ficheiros, chave e índice, puder ser realizada, pelos programas que introduzem ou alteram as relações chave-registo base, sem haver nececssidade de recorrer à reorganização global dos ficheiros. Ver-se-á no que se segue que tal é possivel.

3. FORMATAÇÃO

Embora não essencial na implementação do método, há interesse em formatar os ficheiros chave e índice, antes da sua utilização.

O 1.º registo físico dos ficheiros é utilizado para o armazenamento dos valores dos parâmetros de controle.

Assim o 1.º registo do ficheiro chave é formado por:
(ecp, ecl, ncr)
ecp - endereço do registo com menor chave.
ecl - endereço do 1.º registo livre.
ncr - número total de registos ocupados.

O ficheiro índice terá no 1.º registo o valor eil que representa o 1.º registo livre nesse ficheiro.

Na formatação dos ficheiros, os registos 2,3, ... são escritos com o endereço do registo que lhes segue, isto é, com estrutura encadeada. O último contém o endereço nulo.

Sempre que houver adição de um registo lógico ao ficheiro, a informação irá ocupar o 1.º registo livre, passando a figurar como 1.º registo livre o que era referenciado pelo que foi utilizado.

Quando se tratar de uma eliminação o registo físico libertado passa a figurar com o 1.º livre, pelo que é encadeado com o que tinha essa função anteriormente. Desta forma todo o espaço de armazenamento é aproveitado.

```
ALGORITMO  1: Formatação dos ficheiros

Dados: 'CHAVE'   {identificação do fich. chave}
       'INDICE'  {identificação do fich. índice}
       ntc       {num. total de registos no fich. chave}
       nti       {num. total de registos no fich. índice}

Início: Abrir 'CHAVE': directo;
        ecp:=0 , ecl:=2 , ncr:=0 ,
        Escrever 'CHAVE' reg.num. 1: ecp, ecl, ncr;
        Para i=2,3,...ntc-1 fazer:
         ! j:=i+1 ,
         ! Escrever 'CHAVE' reg.num. i: j;
         !_ ;
        j:=0 ,
        Escrever 'CHAVE' reg.num. ntc: j;
        Fechar: 'CHAVE';
        Abrir 'INDICE': directo;
        eil:=2 ,
        Escrever 'INDICE' reg.num. 1: eil;
        Para i=2,3,...nti-1 fazer:
         ! j:=i+1 ,
         ! Escrever 'INDICE' reg.num. i: j;
         !_ ;
        j:=0 ,
        Escrever 'INDICE' reg.num. nti: j;
        Fechar: 'INDICE';
        fim ;

Resultado: {'CHAVE' e "INDICE' formatados}
```

4. FUNÇÕES BÁSICAS

A utilização da estrutura de dados, como foi definida em 2., pode ser feita pelos programas de exploração com a ajuda de um conjunto de funções, na forma de subprogramas, que se indica a seguir.

Nas descrições dos algoritmos que se seguem pressupõe-se que os parâmetros de controle e lista directório são reconhecidos globalmente, isto é, são dados externos.

4.1. INICIALIZAÇÃO

A utilização das restantes funções só é possível após a execução da inicialização, que tem como objectivo: a abertura dos ficheiros chave e índice, (a

abertura do ficheiro base está a cargo do utilizador) a criação do directório e o estabelecimento dos parâmetros de controle.

A dimensão da área ocupada pelo directório é definida pelo utilizador, no programa de chamada, de seguinte forma:

DC (i) (i =1,2,...nmd) — agrupamento ("array") com elementos de definição idêntica às chaves.

de (i) (i =1,2,...nmd) — agrupamento com elementos com a mesma definição que os endereços do ficheiro chave.

nmd — dimensão dos agrupamentos, estabelecida de acordo com a disponibilidade de memória, mas tendo presente que, quanto maior for, maior será a eficiência no processamento.

```
ALGORITMO  2: Inicialização

Dados: nmd                          {dimensão da área de directório}
       DC(i) (i=1,2,...nmd)  {área do directório p/ chaves}
       ed(i) (i=1,2,...nmd)  {área do directório p/ endereços}
Início: Abrir 'CHAVE': directo; Se erro então: teste:=1 , fim ;
                                              senão: ;
        Ler 'CHAVE' reg.num. 1: ecp, ecl, ncr;
        Se ncr>0
          então: Se ecp=0 então: teste:=-1 , fim ;
                          senão: pad:=int(ncr/nmd) ;
                                                 {divisão inteira}
                                 Se pad=0 então: pad:=1 ;
                                          senão: ;
                 j:=ecp , ntd:=0 ,
                 Para i=1,2,... enquanto j>0 fazer:
                  | ntd:=ntd+1 , ed(ntd):=j ,
                  | Ler 'CHAVE' reg.num. j: DC(ntd), j, a;
                  | Se pad>1
                  |   então: Para t=1,2,...pad-1 fazer:
                  |                | Ler 'CHAVE' reg.num. j: A, j, a;
                  |                |_ ;
                  |   senão: ;
                  |_ ;
          senão: ntd:=0 ;
        Abrir 'INDICE': directo; Se erro então: teste:=2 , fim ;
                                              senão: ;
        Ler 'INDICE' reg.num. 1: eil ;
        teste:=0 , fim ;
Resultado: ntd, DC(i), ed(i) (i=1,2,...ntd), ecp, ecl, eil, teste
```

O valor teste=0 indica que as operações foram bem sucedidas.

4.2. ADIÇÃO AO FICHEIRO CHAVE

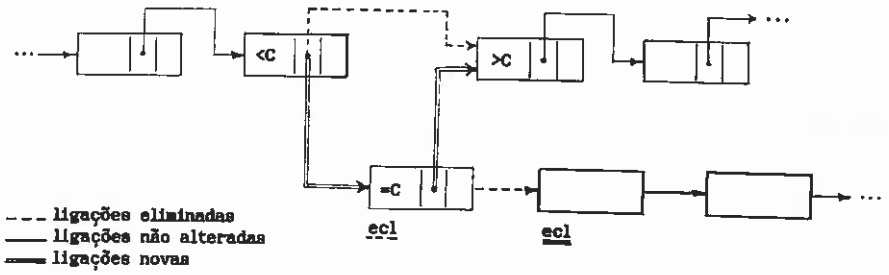
Esta função tem como finalidade permitir a actualização da informação suportada pela estrutura de dados.

A adição dos índices, em correspondência com determinada chave, só será realizada após a chave ser adicionada ao ficheiro chave.

A função devolve um valor (teste=0) se a adição foi bem sucedida, (teste=1) indica a existência de uma chave igual no ficheiro, pelo que a operação não é realizada. Se não existir espaço livre no ficheiro é devolvido teste=-1.

```
ALGORITMO  3: Introdução de nova chave

Dados: C     (chave a adicionar ao fich. chave)

Início: Se ecl=0 então: teste:=-1 , fim ; senão: ;
        Se ecp=0 então: Ler 'CHAVE' reg.num. ecl: s;
                        z:=0 ,
                        Escrever 'CHAVE' reg.num. ecl: C, z, z;
                        ecp:=ecl , ecl:=s , ncr:=ncr+1 ,
                        ntd:=1 , DC(1):=C , ed(1):=ecp ,
                        fim ;
                senão: ;
        Se C<=DC(1) então: u:=ecp , d:=1 ;
                    senão: Para d=1,2,...ntd enquanto DC(d)<C fazer:
                               |_ u:=ed(d) ;
        Se C=DC(d) então: teste:=1 , fim ; senão: ;
        z:=0 , a:=u ,
        Para i=1,2,... fazer:
         | Ler 'CHAVE' reg.num. a: W, u, b;
         | Se W=C então: teste:=1 , fim ; senão: ;
         | Se (W>C) ou (u=0)
         |    então: Ler 'CHAVE' reg.num. ecl: s;
         |            Escrever 'CHAVE' reg.num. ecl: C, u, z;
         |            Se a=ecp
         |              então: ecp:=ecl , DC(1):=C , ed(1):=ecp ;
         |              senão: Escrever 'CHAVE' reg.num. a: WA, ecl, ai; ;
         |            ecl:=s , teste:=0 , ncr:=ncr+1 , fim ;
         |    senão: a:=u , WA:=W , ai:=b ;
         |_ ;

Resultado: teste
```

O algoritmo baseia-se na conhecida técnica de adição de um novo elemento a uma lista encadeada com ordeancão ascendente, como é representada na figura seguinte:



As funções seguintes pressupõem que a 1.ª entrada do directório corresponde à menor chave no ficheiro chave, pelo que, para garantir a coerência, sem reorganizar todo o directório, tem, no caso de a chave adicionada passar a ser a menor do ficheiro, substituir a 1.ª entrada pela informacão relativa à chave que foi adicionada.

Todas as funções são insensíveis ao facto de o intervalo entre entradas sucessivas do directório deixar de ser um número constante após a operação descrita. Pode, inclusive, acontecer que alguns desses intervalos se tornem nulos, como resultado da função de eliminação de chaves que, é tratada a seguir.

4.3. ELIMINAÇÃO DE CHAVES

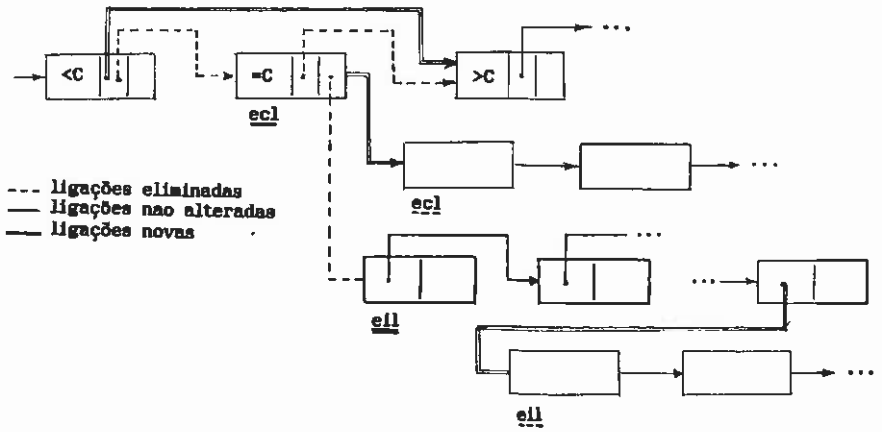
Pode haver necessidade de eliminar uma chave, porque contém um valor errado ou, porque o nome da chave está erradamente representado e necessita de ser corrigido. Neste último caso há necessidade de previamente guardar todos os índices subordinados pára, após a eliminação do nome errado introduzir o valor correcto e seguidamente os índices salvaguardados. Notar que a eliminação duma chave implica a anulação de todos os índices subordinados.

A função descrita a seguir, devolve (teste =0), se a chave foi efectivamente eliminada e (teste =1) se o valor C da chave dada não existir no ficheiro.

```
ALGORITMO  4: Eliminação de chave existente

Dados: C      {chave a eliminar no fich. chave}

Início: Se (ecp=0) ou (C<DC(1)) então: teste:=1 , fim ; senão; ;
        Se C=DC(1)
          então: Ler 'CHAVE' reg.num, ecp: W, a, ii;
                 Se a>0
                   então: Ler 'CHAVE' reg.num, a: W, b, s;
                          Para i=1,2,...ntd enquanto C=DC(i) fazer:
                           !_ DC(i):=W , ed(i):=a ;
                          ;
                   senão: ntd:=0 ;
```

```
                Escrever 'CHAVE' reg.num. ecp: ecl;
                ecl:=ecp , ecp:=a , ncr:=ncr-1 ;
          senão: Para d=1,2,...ntd enquanto DC(d)<C fazer:
                 |_ u:=ed(d) ;
                t:=0 ,
                Para i=1,2,... enquanto t=0 fazer:
                 | Ler 'CHAVE' reg.num. u: W, s, ii;
                 | Se W>C então: teste:=1 , fim ; senão: ;
                 | Se W=C então: t:=1 ;
                 |        senão: Se s=0
                 |                  então: teste:=1 , fim ;
                 |                  senão: a:=u , u:=s , WA:=W , ai:=ii ;
                 |_ ;
                Escrever 'CHAVE' reg.num. a: WA, s, ai;
                Para i=d+1,d+2,...ntd enquanto DC(i)=C fazer:
                 |_ DC(i):=WA , ed(i):=a ;
                Escrever 'CHAVE' reg.num. u: ecl;
                ecl:=u , ncr:=ncr-1 ;
          Para i=1,2,... enquanto ii>0 fazer:
           | Ler 'INDICE' reg.num. ii: s;
           | Escrever 'INDICE' reg.num. ii: eil ;
           |_ eil:=ii , ii:=s ;
          teste:=0 , fim ;

Resultado: teste
```

O algoritmo é baseado no esquema lógico seguinte:



Quanto ao directório, para manter a coerência nas operações que, eventualmente, sucedam a esta, há que atender ao seguinte:

Quando a chave é idêntica a uma entrada do directório esta é substituída pela precedente no directório, criando-se assim uma duplicação de entradas que não acarreta problemas, a não ser o facto de, como é indicado no alogaritmo 4, ter de ser feita a averiguação das entradas que seguem a que foi substituida.

No caso da chave eliminada ser a menor do ficheiro há que actualizar a 1.ª entrada do directório com a informação relativa à nova menor chave.

4.4. ADIÇÃO DE ÍNDICES

Esta função destina-se a permitir a associação de registos base a uma chave dada.

Os dados serão a chave C e um agrupamento eb(j) (j =1,2,...ni), de valores dos endereços dos registos do ficheiro base que se pretendem pôr em correspondência com a chave C.

Na saída é devolvido o valor (teste =0), se C existe no ficheiro ou, (teste =1) no caso contrário.

Se (teste =0), é devolvido um agrupamento ti (j) (j =1,2,...ni), cujos valores significam: se (ti(j) =0), o endereço eb (j) foi adicionado, se (ti (j) =1), o endereço eb (i) já existia previamente como subordinado da chave C.

ALGORITMO 5: Introdução de novos índices

```
Dados: C              {chave a qual os índices ficam subordinados}
       ni             {num. de índices a introduzir}
       eb(i) (i=1,2,...ni)  {endereços base p/ cada índice}


Início: Se eil=0 então: teste:=-1 , fim ; senão: ;
        Se (ecp=0) ou (C<DC(1)) então: teste:=1 , fim ; senão: ;
        Para d=1,2,...ntd enquanto C<=DC(d) fazer:
         |_ u:=ed(d) ;
        Para j=1,2,...ni fazer:
         |_ ti(j):=1 ;
        Para i=1,2,... fazer:
         | Ler 'CHAVE' reg.num. u: W, a, ii;
         | Se C=W
         |  então: Se ii=0 então: nr:=0 ;
         |                 senão: s:=ii , nr:=0 ,
         |                        Para j=1,2,... enquanto s>0 fazer:
         |                         | Ler 'INDICE' reg.num. s: q, b;
         |                         | nr:=nr+1 , r(nr):=b ,
         |                         |_ iu:=s , s:=q ;
         |                        ;
         |          Para j=1,2,...ni fazer:
         |           | v:=0 ,
         |           | Para p=j-1,j-2,...1 enquanto v=0 fazer:
         |           |  | Se eb(p)=eb(j) então: v:=1 ; senão: ;
         |           |  |_ ;
         |           | Para p=1,2,...nr enquanto v=0 fazer:
         |           |  | Se r(p)=eb(j) então: v:=1 ; senão: ;
         |           |  |_ ;
         |           | Se v=1
         |           |   então: ti(j):=1 ;
         |           |   senão: Ler 'INDICE' reg.num. eil: s;
         |           |          Escrever 'INDICE' reg.num. eil: ii, eb(j);
         |           |          ii:=eil , eil:=s , ti(j):=0 ,
         |           |          Se (eil=0) e (j<ni)
         |           |            então: teste:=-(j+1) , fim ; senão: ;
         |           |_ ;
         |          Escrever 'CHAVE' reg.num. u: W, a, ii;
         |          teste:=0 , fim ;
         |  senão: Se (C>W) ou (a=0) então: teste:=1 , fim ;
         |                                  senão: u:=a ;
         |          ;
         |_ ;


Resultado: teste , ti(j) (j=1,2,...ni)
```

A lógica é a habitual na inserção de novos elementos a uma lista, mas note-se que, neste caso, não existe obrigatoriedade de ordenação pelo conteúdo pelo que, o novo elemento pode ser encadeado directamente ao 1.º da lista.

4.5. ELIMINAÇÃO DE INDÍCES

São seguidas as mesmas convenções de 4.4 com a lógica adequada.

O significado dos elementos: ti(j) é o mesmo que anteriormente: ti (j) =0 indica sucesso na eliminacão de eb (j) e ti (j) =1 que significa que o indíce eb (j) não existia.

```
ALGORITMO  6: Eliminação de índices

Dados: C                {chave a qual os índices estão subordinados}
       ni               {num. de índices a eliminar}
       eb(i) (i=1,2,...ni)  {endereços base dos índices a eliminar}
Início: Se (ecp=0) ou (C<DC(1)) então: teste:=1 , fim ; senão: ;
        Para d=1,2,...ntd enquanto C<=DC(d) fazer:
         |_ u:=ed(d) ;
        Para j=1,2,...ni fazer:
         |_ ti(j):=1 ;
        Para i=1,2,... fazer:
         | Ler 'CHAVE' reg.num. u: W, a, ii;
         | Se C=W
         |   então: Se ii:=0 então: teste:=0 , fim ; senão: ;
         |           t:=ii ,
         |           Para p=1,2,... enquanto t>0 fazer:
         |            | Ler 'INDICE' reg.num. t: s, b;
         |            | Para j=1,2,...ni fazer:
         |            |  | Se eb(j)=b
         |            |  |   então: Escrever 'INDICE' reg.num. t: eii;
         |            |  |          eii:=t ,
         |            |  |          Se t=ii
         |            |  |            então: ii:=s ;
         |            |  |            senão:
         |            |  |             Escrever 'INDICE' reg.num. q: s, bb;
         |            |  |                  ;
         |            |  |          ti(j):=0 ;
         |            |  |   senão: ;
         |            |  |_ ;
         |            |_ q:=t , bb:=b , t:=s ;
         |           Escrever 'CHAVE' reg.num. u: W, a, ii;
         |           teste:=0 , fim ;
         |   senão: Se (C>W) ou (a=0) então: teste:=1 , fim ;
         |                                   senão: u:=a ;
         |_ ;

Resultado: teste , ti(j) (j=1,2,...ni)
```

4.6 PESQUISA

A pesquisa pode estabelecer-se como resposta ao seguinte inquérito: "Quais os registos base que estão simultaneamente (intersecção) subordinados a um conjunto de chaves dadas?"

Dada a lista K (i) (i =1,2,...nk) a função devolve uma lista eb(j) (j =1,2,...nb) de endereços base simultaneamente subordinados a todas as chaves K (i) (i =1,2,...nk).

É devolvida igualmente uma lista tt (i) (i =1,2,...nk) cujos elementos têm o seguinte significado: (tt(i) =-1) indica que a chave K (i) não existe no ficheiro, (tt(i) ≥ 0) representa o número total de registos base que estão associados à chave K(i).

Obviamente se algum (tt(i) ≤0) a lista eb é vazia, isto é, (nb =0).

```
ALGORITMO  7: Pesquisa dos registos base simultaneamente associados
              com um conjunto de chaves

Dados: nk                      {num. de chaves a pesquisar}
       K(i) (i=1,2,...nk)      {conjunto das chaves}

Início: nb:=0 ,
        Se ecp=0 então: fim ; senão: ;
        Para i=1,2,...nk fazer:
         | tt(i):=0 ,
         | Se K(i)<DC(1)
         |   então: tt(i):=-1 , nb:=0 ;
         |   senão: Para d=1,2,...ntd enquanto K(i)<=DC(d) fazer:
         |           |_ u:=ed(d) ;
         |          t:=0 ,
         |          Para p=1,2,... enquanto t=0 fazer:
         |           | Ler 'CHAVE' reg.num. u: W, a, ii;
         |           | Se K(i)=W
         |           |   então: nr:=0 ,
         |           |          Para j=1,2,... enquanto ii>0 fazer:
         |           |           | Ler 'INDICE' reg.num. ii: s, b;
         |           |           |_ nr:=nr+1 , r(nr):=b , ii:=s ;
         |           |          tt(i):=nr ,
         |           |          Se nr=0 então: nb:=0 ; senão: ;
         |           |          Se (nr>1) e (nb>0)
         |           |            então:                    {ordenação}
         |           |                   Para j=2,3,...nr fazer:
         |           |                    | tr:=r(j) ,
         |           |                    | Para d=j-1,j-2,...1
         |           |                    | |    enquanto r(d)>tr fazer:
         |           |                    | |_ r(d+1):=r(d) ;
         |           |                    |_ r(d):=tr ;
         |           |            senão: ;
         |           |          Se (i=1) e (nr>0)
```

```
|          |          então: Para j=1,2,...nr fazer:
|          |                  |_ eb(j):=r(j) ;
|          |                 nb:=nr ;
|          |          senão: ;
|          |        Se (i>1) e (nb>0)
|          |          então: jb:=1 ,     {actualização de eb}
|          |                 Para j=1,2,...nr
|          |                  |      enquanto jb<=nb fazer:
|          |                  | Se r(j)=eb(jb)
|          |                  |   então: jb=jb+1 ;
|          |                  |   senão: Se r(j)>eb(jb)
|          |                  |                então: eb(jb):=0 ,
|          |                  |                       jb:=jb+1 ;
|          |                  |                senão: ;
|          |                  |_ ;
|          |                 jb:=0 ,
|          |                 Para j=1,2,...nb fazer:
|          |                  | Se eb(j)>0
|          |                  |   então: jb:=jb+1 ,
|          |                  |          eb(jb):=eb(j) ;
|          |                  |   senão: ;
|          |                  |_ ;
|          |                 nb:=jb ;
|          |          senão: ;
|          |  senão: Se (K(i)>W) ou (a=0)
|          |          então: t:=1 ;
|          |          senão: u:=a ;
|          |_ ;
|_ ;
fim ;
```

Resultado: nb , eb(j) (j=1,2,...nb) , tt(i) (i=1,2,...nk)

A operação de ordenação, para a qual se utilizou o algoritmo da inserção, tem como finalidade permitir uma maior eficiência na operação de actualização da lista de saída que está ordenada por ordem crescente.

4.7. FECHO

Antes de terminar um processamento é indispensável executar esta função, cuja finalidade é armazenar para os processamentos seguintes os valores dos parâmetros de controle que eventualmente tenham sido alterados.

Se o processamento unicamente usou a função de inicialização e pesquisa a não execução desta função não acarreta problemas futuros. O mesmo já não acontece, em princípio, com as restantes funções.

Este problema está ligado com a recuperação de informação quando ocorre uma interrupção acidental do processamento.

O utilizador deve fazer um programa, utilizando as funções que foram descritas, que realize essa recuperação.

A lógica desse programa pode ser a seguinte:

— Identificar todos os registos chave ocupados e determinar qual não é referenciado por outro registo no ficheiro, encontra-se assim o parâmetro ecp;

```
ALGORITMO  8: Fecho

Início: Escrever 'CHAVE' reg.num. 1: ecp, ecl, ncr;
        Fechar: 'CHAVE';
        Escrever 'INDICE' reg.num. 1: eil;
        Fechar: 'INDICE';
        fim ;
```

— Realizar a mesma operação com os registos chave livres e obtendo-se ecl;

— Por contagem determinar ncr;

— Identificar os registos índice livres e verificar qual não é referenciado por outro registo, o endereço encontrado é eil.

No caso de o programa de recuperação não existir ter-se-á que fazer cópias de segurança dos ficheiros, assim como dos ficheiros de movimentos de actualização, mais frequentes.

5. CONCLUSÕES

Mostrou-se que:

— O método proposto pode ser facilmente implementado com um pequeno número de subprogramas.

— A programação pode realizar-se em qualquer das linguagens usuais por uma transcrição quase directa das descrições feitas.

— O trabalho inicial de programação para uma aplicação é quase integralmente aproveitado para outras aplicações de índole diferente.

— Todo o tratamento do ficheiro base é independente do tratamento da estrutura de dados criada, pelo que é possível instalar esta estrutura sobre ficheiros de aplicações já existentes. Só não é permitida a utilização de registos eliminados no ficheiro base para a introdução de informacão relativa a um identificador diferente do que anteriormente foi eliminado.