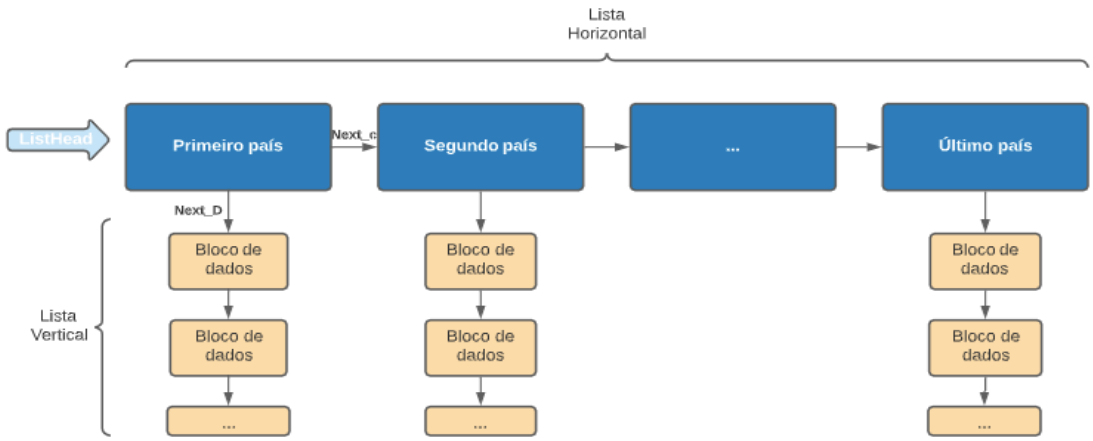
Projeto Final de Programação – Covid19

UC: Programação

Inês Cardoso Paiva 99961 & Maria Teresa Nogueira 100029

Criação da lista de listas ou "Nested list" – função create_country

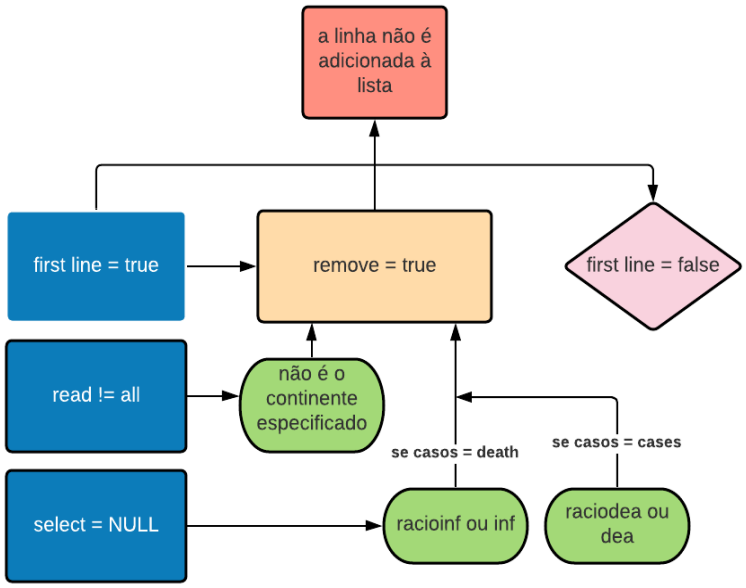
Considerando as restrições indicadas no enunciado do projeto, nomeadamente , *"(...) a lista principal deve ter apenas um nó para cada país presente na coluna 1."* Concluímos que a implementação mais eficiente seria a criação de uma lista de listas, assim cada nó da lista horizontal adquire os dados ditos fixos, relativos ao país em questão e a lista vertical, ligada a cada nó da lista horizontal adquire os dados variáveis para esse país:



Esta criação é auxiliada por 4 funções coadjuvantes:

Função remove:

A função remove é responsável pela remoção da primeira linha do ficheiro csv, cuja informação não demonstra qualquer utilidade à reprodução do ficheiro como também auxilia, se assim for necessário, a realização das funcionalidades quando os parâmetros de entrada -L e -D são, respetivamente -L != all e -D!=NULL, tal como é ilustrado no gráfico infra:



Funções insert_end_tail, check_country e find_data_tail :

A função insert_end_tail coloca um novo nó de um elemento da lista horizontal ligado ao anteriormente ligado, após a alocação de memória para este mesmo nó, no entanto, a realização desta alocação, bem como da função acima descrita está dependente da da função check_country que verifica se um dado país já foi previamente alocado e ligado à função horizontal, retorna o ponteiro para esse mesmo nó ou então um nó vazio se o país não existir, previne assim a criação de nós desnecessários. Se assim for a função create_country é redirecionada para a sua parte inferior onde irá procurar o último elemento colocado na lista vertical, através da função, **find_data_tail** (é importante referir que o ponteiro data_head é apenas atribuido ao primeiro nó de dados criado, ligando-se ao ponteiro Next_D da estrutura de dados fixos, é por isto utilizada a noção de data tail quando pretendemos realizar ligações entre nós da lista vertical)

Por fim a função create_country retorna o ponteiro *send (ponteiro para o primeiro nó criado, apenas guardado uma vez) que se tornará o ponteiro da "cabeça" da lista de listas.

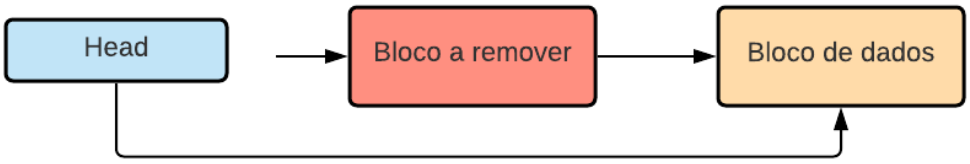
É importante referir ainda, embora evidente que a alocação é realizada em dois momentos da construção da lista no momento em que é criado um nó para o país e para um nó de dados variáveis.

Algorítmo de seleção:

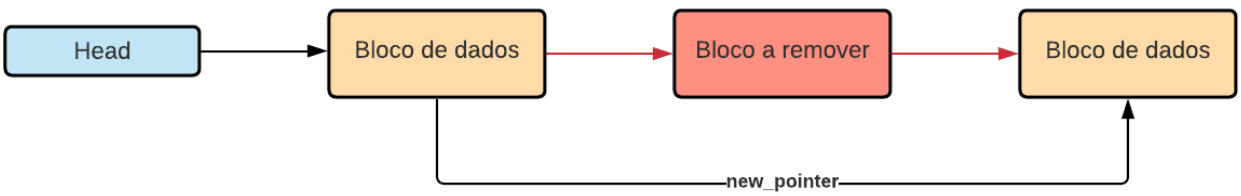
O algorítmo de seleção é facilmente ilustrado a partir dos seguintes gráficos:

Considerando por exemplo um parâmetro de entrada -D inf:

Se o número de infetados do segundo for maior que o primeiro o primeiro é libertado a partir da função free(bloco a remover)



Se o número de infetados do primeiro for maior que o segundo o segundo é libertado a partir da função free(bloco a remover)



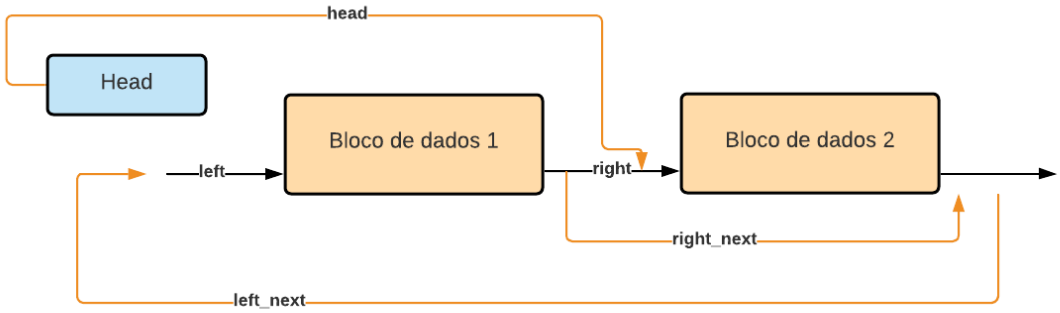
Posteriormente é feita a atualização do ponteiro, no primeiro caso basta garantir que o ponteiro current é igual ao novo listHead da lista vertical, no segundo o ponteiro é colocado no anterior, agora ligado ao terceiro, para que a sua atualização passe para esse mesmo terceiro após o loop de while. No caso dos valores serem iguais, é realizado um desempate com a data mais recente. (esta função é auxiliada por um "manager" que decide os parâmetros a comparar, segundo o

input do utilizador). No fim, é esperado que permaneça o valor mais elevado do parâmetro escolhido.

Funções de ordenação:

A ordenação da lista é realizada através do algorítmo bubble sort dado em aula, cuja imagem infra ilustra o funcionamento.

Se ocorrer troca (bloco de dados 1 < bloco de dados 2) temos portanto:



Da mesma forma que as funções de seleção, a função de bubble sort é auxiliada por um manager que procura evidenciar os parâmetros a comparar. É ainda importante especificar que no caso de ordenações associadas a um indicador e a uma data fazem uso de uma função de search para encontrar o node vertical específico a comparar, no caso de não existirem os parâmetros retorna NULL que, admitindo que o node seguinte também o é não é realizada troca, se o primeiro não for NULL não será realizada a troca (seria o mesmo que node1>node2) no caso do inverso será realizada a troca (seria o mesmo que node1<node2).

Funções de restrição:

Por motivos de brevidade podemos afirmar que as funcionalidades dos algorítmos de restrição são em tudo semelhantes aos de seleção, com a adição da responsabilidade de remover dados da lista vertical totalmente como também o node da lista horizontal se em parâmetros de date ou dates não existirem as datas especificadas.

Funções de reprodução e leitura do ficheiro binário:

A função de escrita em binário é em tudo semelhante à de escrita em csv, os loops de while realizados são exatamente os mesmos, no entanto são escritos os dados através da função fwrite da seguinte forma:

```
for (current_c = listHead; current_c != NULL; current_c = current_c->nextC)
{
    fwrite(&current_c->country, sizeof(current_c->country), 1, fptr);
    fwrite(&current_c->country_ID, sizeof(current_c->country_ID), 1, fptr);
    fwrite(&current_c->continent, sizeof(current_c->continent), 1, fptr);
    fwrite(&current_c->population, sizeof(current_c->population), 1, fptr);

    for (current_d = current_c->nextD; current_d != NULL; current_d = current_d->next)
    {
        fwrite(&current_d->indicator, sizeof(current_d->indicator), 1, fptr);
        fwrite(&current_d->weekly_count, sizeof(current_d->weekly_count), 1, fptr);
        fwrite(&current_d->year_week, sizeof(current_d->year_week), 1, fptr);
        fwrite(&current_d->rate_14_day, sizeof(current_d->rate_14_day), 1, fptr);
        fwrite(&current_d->cumulative_count, sizeof(current_d->cumulative_count), 1, fptr);
        fwrite(&current_d->next, sizeof(current_d->next), 1, fptr);
    }
}
```

São, portanto, escritos todos os elementos, um a um de cada uma das estruturas (de dados variáveis e fixos) no ficheiro binário, bem como a adição do número de nós de países logo no início do ficheiro, de forma a facilitar a sua leitura.

A leitura por sua vez é realizada, tal como a de escrita elemento a elemento através da função fread, o número de nós de países é ainda passado para uma variável de forma a facilitar o termínio do loop da lista horizontal, a vertical, por sua vez, é terminada quando é realizada a leitura de um NULL para o ponteiro next:

```
while (end == false)
{
    if ((data_node = (data *)calloc(1, sizeof(data))) == NULL)
    {
        printf("Não foi possível alocar o bloco de memória\n");
        exit(EXIT_FAILURE);
    }

    fread(&data_node->indicator, sizeof(data_node->indicator), 1, file_in);
    fread(&data_node->weekly_count, sizeof(data_node->weekly_count), 1, file_in);
    fread(&data_node->year_week, sizeof(data_node->year_week), 1, file_in);
    fread(&data_node->rate_14_day, sizeof(data_node->rate_14_day), 1, file_in);
    fread(&data_node->cumulative_count, sizeof(data_node->cumulative_count), 1, file_in);
    fread(&data_node->next, sizeof(data_node->next), 1, file_in);

    if (data_node->next == NULL)
    {
        end = true;
    }

    if (first == true)
    {
        first = false;
        data_head = data_node;
        data_tail = data_node;

        new_country->nextD = data_head;
    }
```

A sua estrutura é extremamente semelhante à função create_country, de certa forma simplificada, (reconhece-se a ordem dos países e portanto ná há a necessidade de utilizar um encontra países nem um encontra blocos verticais respetivamente para a função horizontal e vertical) é apenas usada a função insert_end_tail para conectar os elementos da lista horizontal.