Relatório do projeto final da UC de programação – Análise de dados relativos à pandemia covid-19

Projeto realizado por:

-Eduardo Filipe Braz Barata (nº 99930)

-Nuno Filipe Trigo Fernandes (nº100695)

Alunos do 1º ano de MEEC – IST

No âmbito na UC de programação 2020/2021

1. Introdução

O presente relatório pretende descrever, de forma sucinta, a estrutura de dados escolhida para representação das informações recolhidas dos ficheiros, a forma como esta foi alocada dinamicamente, como a podemos aceder e como funciona e, por fim, uma descrição das características que decidimos acrescentar ao projeto.

2. Estrutura de dados:

```
typedef struct countries{
    char *country_name;
    char country_code[4];
    char continent[8];
    int population;
    struct countries *next;
    struct country_info *head;
} country;
```

Figura 1 – Estrutura de dados principal. Nesta estrutura serão guardados os dados considerados fixos, isto é, os que são independentes da semana considerada. Especialmente, para o nome do país, o espaço de memória para este será alocado dinamicamente. É também reservado um espaço para o endereço do nó do próximo país e para o primeiro dado guardado na lista secundária (abordado em mais detalhe no ponto 4).

```
typedef struct country_info{
    char indicator[6];
    int weekly_count;
    char year_week[7];
    double rate_14_day;
    int cumulative_count;
    struct country_info *nexti;
}countryinfo;
```

Figura 2 – Estrutura de dados secundária. Nesta estrutura serão guardados os dados considerados variáveis, ou seja, aqueles que dependem e variam de semana para semana. Nesta estrutura, reserva-se, também, um espaço para o endereço do próximo nó de dados variáveis (abordado em mais detalhe no ponto 4).

3. Alocação dinâmica das estruturas

Estas estruturas, são, por sua vez, alocadas dinamicamente de forma a poderem ser usadas como nós na lista que irá conter a informação dos ficheiros. Assim, aquando da leitura da linha de um ficheiro, é alocado, à partida, um bloco de memória para a estrutura *country_info* já que todas as linhas vão conter informação variáveis novas e relevantes a serem guardadas em memória. Já em relação à estrutura *country* verifica-se, primeiramente, se a linha que estamos a ler contém um país que já se encontra na nossa lista principal de forma a garantir que os dados são guardados da forma mais eficiente e que não se criam nós desnecessários.

4. Organização das listas

As listas encontram-se organizadas de uma forma bastante intuitiva, isto é:

- Existe uma lista principal formada por nós do tipo *country*, nós esses que contêm a informação descrita no ponto 2. Estes têm uma ligação simples uns com os outros, isto é, um nó aponta exclusivamente para o nó seguinte;
- Desta lista principal surge, para cada nó, uma lista secundária formada por nós do tipo *country_info*. À semelhança da lista principal estes estão simplesmente ligados.

No anexo (I), poderá ver-se um exemplo gráfico desta lista. Descrever-se-á, agora, a forma de aceder a cada um dos elementos da lista.

5. Como aceder à lista

Para se perceber melhor o funcionamento da lista deverá consultar-se o anexo (I) que contém um exemplo do tipo de lista que foi utilizada para este programa.

De uma forma sucinta, cria-se uma variável para conter o valor da posição do primeiro elemento da lista principal, isto é, da informação fixa do país. Este elemento, por sua vez, aponta para dois locais: para o elemento seguinte, ou seja, para um país diferente e para a sua sublista que contém os dados variáveis de cada semana. Cada elemento desta sublista apontará, à semelhança da lista principal para o elemento seguinte. No final de cada lista, estes apontadores irão apontar para o valor *NULL*.

6. Verificação de *input*

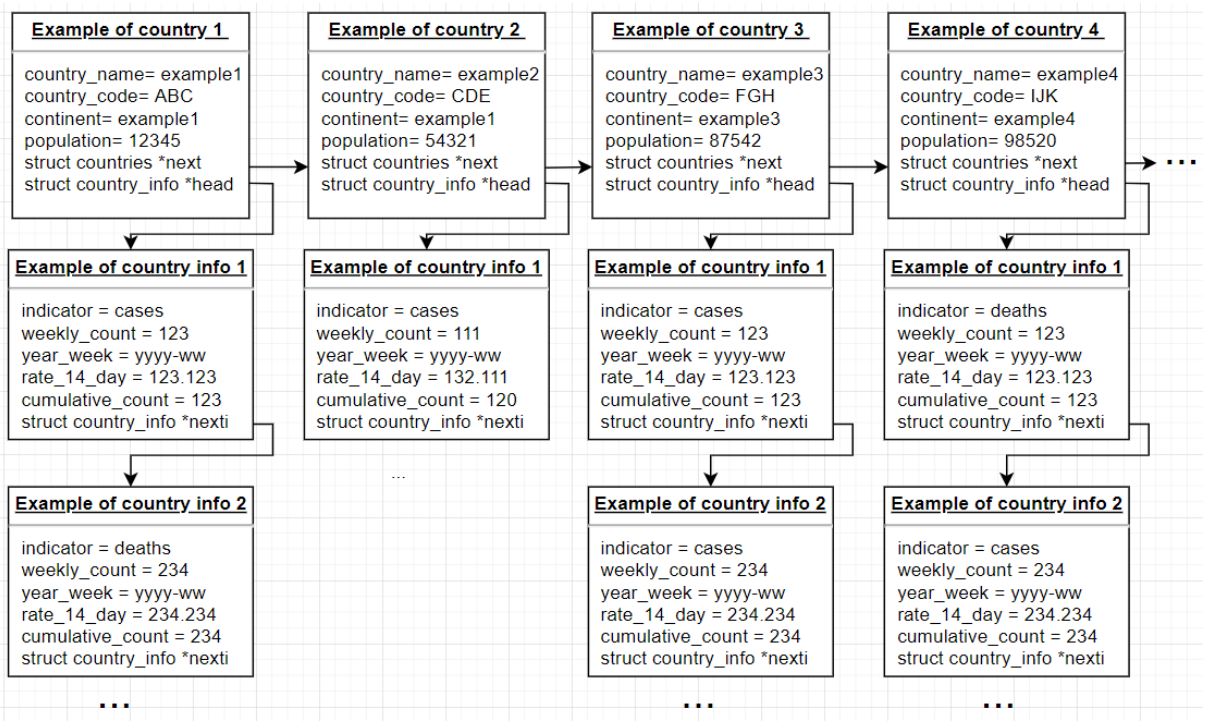
Neste programa, o nosso grupo, procurou validar todos os *inputs* do utilizador de uma forma muito minuciosa de modo a que fosse muito difícil ou quase impossível serem provocados erros no programa que pudessem originar perdas de memória.

7. Impressão de dados

Por motivos práticos de verificar o output gerado pelo programa, para além das funcionalidades pedidas de impressão para ficheiros do tipo .csv ou do tipo .dat, o nosso programa realiza também uma impressão para o terminal de forma a facilitar a análise do resultado do tratamento de dados.

8. Anexos

I.


Example of country 1
country_name= example1
country_code= ABC
continent= example1
population= 12345
struct countries *next
struct country_info *head
Example of country 2
country_name= example2
country_code= CDE
continent= example1
population= 54321
struct countries *next
struct country_info *head
Example of country 3
country_name= example3
country_code= FGH
continent= example3
population= 87542
struct countries *next
struct country_info *head
Example of country 4
country_name= example4
country_code= IJK
continent= example4
population= 98520
struct countries *next
struct country_info *head
Example of country info 1
indicator = cases
weekly_count = 123
year_week = yyyy-ww
rate_14_day = 123.123
cumulative_count = 123
struct country_info *nexti
Example of country info 1
indicator = cases
weekly_count = 111
year_week = yyyy-ww
rate_14_day = 132.111
cumulative_count = 120
struct country_info *nexti
Example of country info 1
indicator = cases
weekly_count = 123
year_week = yyyy-ww
rate_14_day = 123.123
cumulative_count = 123
struct country_info *nexti
Example of country info 1
indicator = deaths
weekly_count = 123
year_week = yyyy-ww
rate_14_day = 123.123
cumulative_count = 123
struct country_info *nexti
Example of country info 2
indicator = deaths
weekly_count = 234
year_week = yyyy-ww
rate_14_day = 234.234
cumulative_count = 234
struct country_info *nexti
Example of country info 2
indicator = cases
weekly_count = 234
year_week = yyyy-ww
rate_14_day = 234.234
cumulative_count = 234
struct country_info *nexti
Example of country info 2
indicator = cases
weekly_count = 234
year_week = yyyy-ww
rate_14_day = 234.234
cumulative_count = 234
struct country_info *nexti