IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Projeto de ATAD

Vacinação COVID-19
Processamento de dados e Previsão

Licenciatura de Engenharia Informática

Diogo Letras, Nº 200221114, Turma: 1
Luís Góias, Nº 200221134, Turma: 4
Pedro Cunha, Nº 200221086, Turma: 4
Tiago Vaz, Nº 200221089, Turma: 1

Docentes: Aníbal Ponte e Miguel López

Índice

a) ADTs utilizados

Os ADT utilizados neste projeto foram ADT List e ADT Map. O ADT List foi utilizado para conter dados relativos à vacinação e o ADT Map foi utilizado para conter dados relativos aos países.

Implementação	listAdd	listRemove	listGet	listSet
Array List	$O(n)$	$O(n)$	$O(1)$	$O(1)$

Fig.1 - Complexidades algorítmicas do ADT List.

Implementação	mapPut	mapRemove	mapGet	mapContains
Array List	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Fig.2 - Complexidades algorítmicas do ADT Map.

Foi escolhida a implementação em Array List para ambos os ADTs, pois é mais eficiente no caso do ADT List (funções listGet e listSet) e pelo facto de não ser possível obter uma melhor complexidade algorítmica no caso do ADT Map.

As funções utilizadas estão descritas no livro “Tipos Abstratos de Dados - Linguagem C”, do docente Bruno Silva.

b) Complexidade algorítmica dos comandos implementados

Comandos A (exceto CLEAR e QUIT):

- LOADC - Este comando, que carrega dados sobre países num mapa, tem complexidade *O(n)*, porque usa a função mapPut, uma quantidade estática de vezes.

- LOADV - Este comando, que carrega dados sobre vacinas numa lista, tem complexidade $O(n^2)$, pois a seguir a inserir todos os dados, estes são ordenados usando um algoritmo bubble sort.

Comandos B:

- SHOWALL - Este comando tem complexidade *O(n)* se for escolhida a opção para mostrar toda a informação (opção “ALL”) ou complexidade *O(1)* se for escolhida a opção para mostrar uma amostra (opção “SAMPLE”).

- SHOWC - Este comando que mostra os dados de vacinação disponíveis de um país solicitado ao utilizador, tem complexidade *O(n)*, pois contém um ciclo for dependente do tamanho de uma lista, enquanto que todas as outras iterações têm um tamanho fixo.

- LISTC - Este comando que mostra a lista de países com dados sobre a vacinação, tem complexidade *O(n)*, porque contém um ciclo for dependente do tamanho de uma lista, enquanto que todas as outras iterações têm um tamanho fixo.

- LISTV - Este comando que mostra a lista de vacinas existentes nos registos sobre a vacinação (nomes únicos), tem complexidade *O(n)*, porque contém um ciclo for que dependente do tamanho de uma lista, enquanto que todas as outras iterações têm um tamanho fixo.

- AVERAGE - Este comando está dividido em duas funções, uma que mostra a média global do número diário de vacinações (AVERAGEGLOBAL), que tem complexidade O(n), porque tem um ciclo dependente do tamanho de uma lista.
  A outra função (AVERAGEPERCOUNTRY), que calcula a média do número diário de vacinações em cada país, tem complexidade *O(n)*, pois contém um ciclo dependente do tamanho de uma lista.

- SHOWV - Este comando, que mostra para cada uma das vacinas conhecidas, a lista de países que a usaram na vacinação, tem complexidade $O(n^2)$, pois contém um ciclo que depende do tamanho de uma lista e de um mapa.

- TOPN - Este comando, que mostra de forma decrescente os dados de vacinação de N países, sendo o critério decrescente pelo atributo “peopleVaccinatedPerHundred”, tem complexidade $O(n^2)$. Foram criadas 3 funções para implementar este comando:

- calcTopN - Complexidade $O(n^2)$, depende do tamanho de duas listas diferentes.
- sortTopN - Complexidade $O(n^2)$, pelo facto de ser um algoritmo bubble sort.
- topN - Complexidade $O(n^2)$, pois implementa as funções acima descritas.

Comandos C:

- COUNTRY_S - Este comando que mostra os dados dos países ordenados por número de habitantes por ordem decrescente, tem complexidade $O(n^3)$, pois contém dois ciclos encadeados dependentes do tamanho do mapa, utilizando a função mapGet.

- REGIONS - Este comando, que mostra vários dados agregados por região mundial, tem complexidade $O(n^2)$, pois contém ciclos encadeados dependentes de duas listas diferentes, mas também porque dois métodos contidos nesta função também são de complexidade $O(n^2)$.

c) Pseudocódigo de 3 funcionalidades

SHOWALL (Algorithm)

Input: [*ptList – PtList pointer, *choice - char]

BEGIN

size <- 0

listSize(*ptList, &size)

IF strcmp(choice, "ALL") = 0 **THEN**

FOR i <- 0 **TO** (i < size) **DO**

ListElem elem

listGet(*ptList, i, &elem)

PRINT "Country: $elem.country"

PRINT "Date: $elem.date.day / $elem.date.month / $elem.date.year"

PRINT "Vaccines: "

FOR j <- 0 **TO** (j < 5) **DO**

IF strcmp(elem.vaccines[j], "") != 0 **THEN**

PRINT "$elem.vaccines[j] "

END IF

END FOR

PRINT "Daily Vaccination: $elem.dailyVaccination Daily Vaccinations Per Million: $elem.dailyVaccinationsPerMillion"

PRINT "People Fully Vaccinated: $elem.peopleFullyVaccinated People Fully Vaccinated Per Hundred: $elem.peopleFullyVaccinatedPerHundred People Vaccinated: $elem.peopleVaccinated People Vaccinated Per Hundred: $elem.peopleVaccinatedPerHundred"

PRINT "Number of Vaccines: $elem.number_vaccines Total Vaccination: $elem.totalVaccination Total Vaccinations Per Hundred: $elem.totalVaccinationsPerHundred"

```
            PRINT "-------------------------------------------------------------------------"

        END FOR

END IF

ELSE IF strcmp(choice, "SAMPLE" = 0)
FOR  i <- 0; TO (i <= 100) DO

        int randomSample = (rand() is divisible by size)

        ListElem elem

        listGet(*ptList, randomSample, &elem)

        PRINT "Country: $elem.country"

        PRINT "Date: $elem.date.day / $elem.date.month / $elem.date.year"

        PRINT "Vaccines: "

        FOR j <- 0 TO (j < 5) THEN
            IF strcmp(elem.vaccines[j], "") != 0 THEN
                PRINT "$elem.vaccines[j]"
            END IF
        END FOR

        PRINT "Daily Vaccination: $elem.dailyVaccination  Daily Vaccinations Per Million:
        $elem.dailyVaccinationsPerMillion"

        PRINT "People Fully Vaccinated: $elem.peopleFullyVaccinated  People Fully
        Vaccinated Per Hundred: $elem.peopleFullyVaccinatedPerHundred  People
        Vaccinated: $elem.peopleVaccinated  People Vaccinated Per Hundred:
        $elem.peopleVaccinatedPerHundred"

        PRINT "Number of Vaccines: $elem.number_vaccines  Total Vaccination:
        $elem.totalVaccination Total Vaccinations Per Hundred:
        $elem.totalVaccinationsPerHundred"

        PRINT "-------------------------------------------------------------------------"
END FOR

ELSE THEN

        PRINT "Please type the correct command"

END ELSE
```

SHOWC (Algorithm)

Input: [*ptList – PtList pointer, *country - char]

BEGIN

```
PtList countryChoice = listCreate();

getReportsOfCountry(*ptList, country, &countryChoice);

int sizeCountryList <- 0;
listSize(countryChoice, &sizeCountryList);
IF  sizeCountryList = 0 THEN
        PRINT " Vaccination data not available for $country"
END IF
ELSE THEN
        FOR i <- 0 TO (i <= sizeCountryList) DO
                ListElem elem;
                listGet(countryChoice, i, &elem);
                PRINT "Country: $elem.country";
                PRINT "Date: $elem.date.day/$elem.date.month/elem.date.yea"
                PRINT "Vaccines: "
                FOR j <- 0 TO ( j < 5;)
                        IF strcmp(elem.vaccines[j], "") != 0 THEN
                              PRINT "$elem.vaccines[j]"
                         END IF
                 END FOR
                PRINT "Daily Vaccination: $elem.dailyVaccination  Daily Vaccinations Per
        Million: $elem.dailyVaccinationsPerMillion"

                PRINT "People Fully Vaccinated: $ elem.peopleFullyVaccinated  People Fully
        Vaccinated Per Hundred: $elem.peopleFullyVaccinatedPerHundred  People
        Vaccinated: $elem.peopleVaccinated  People Vaccinated Per Hundred:
        $elem.peopleVaccinatedPerHundred"

               PRINT "Number of Vaccines: $elem.number_vaccines  Total Vaccination:
        $elem.totalVaccination Total Vaccinations Per Hundred:
        $elem.totalVaccinationsPerHundred"

                PRINT "-----------------------------------------------------------------------"
        END FOR
    END ELSE
listDestroy(&countryChoice);
END
```

AVERAGEGLOBAL (Algorithm)

Input: *ptList – PtList pointer

```
BEGIN
        size <- 0;
        count <- 0;
        avgGlobal <- 0;

        ListElem *ptElem = malloc(sizeof(ListElem));

        listSize(ptList, &size);

        FOR i = 0 TO (i < size) DO
                listGet(ptList, i, ptElem);

                IF ptElem->dailyVaccination >= 0 THEN
                        avgGlobal <- avgGlobal + ptElem->dailyVaccination;
                END IF

                count++;
         END FOR

        avgGlobal /= count;

        PRINT "Population Average of Daily Vaccinations = $avgGlobal"

        free(ptElem);
END
```

d) Limitações

O comando PREVISION não foi implementado devido à sua complexidade algorítmica, bugs e problemas de alocação de memória, por isso não conseguimos implementar com sucesso este comando.

O comando REGIONS foi implementado, porém não conseguimos verificar a sua veracidade.

e) Conclusões

Este projeto permitiu ao grupo aprimorar várias competências na programação.
Até agora, o grupo estava habituado a programar de forma mais direta, sem pensar muito na lógica geral do problema.
A capacidade de pegar num problema complexo e dividi-lo em problemas menores, foi sem dúvida, melhorada com o desenvolvimento deste projeto.
Também foi deduzido pelo grupo, que as complexidades algorítmicas são extremamente importantes, e deve-se optar sempre pela mais eficiente.
Sem dúvida, que a experiência obtida pelo desenvolvimento deste projeto, vai ser útil no futuro dos elementos do grupo, tanto em contexto académico como em contexto profissional.