TECNOLOGIA SETÚBAL
ESCOLA SUPERIOR POLITÉCNICO SETÚBAL

ATAD 2023/24

Algoritmos e Tipos Abstratos de Dados

Enunciado de Projeto de Época ESPECIAL

Análise de Dados sobre os Jogos Olímpicos

(Processamento Estatístico e Visualização de Dados)

Regras

a) O Projeto deverá ser elaborado **individualmente.**

b) Este projeto tem uma entrega final, dia 17 de setembro às 10:00. A entrega deve ser submetida no Moodle. Submissões são penalizadas com 0,5 valores em cada bloco de 30min de atraso.

c) **As discussões do projeto são obrigatórias**. As datas das discussões serão publicadas após a entrega final dos trabalhos.

- A nota do projeto vale 90% e existe um Quiz individual no início da discussão, que perfaz os restantes 10% da classificação do projeto.
- **A nota do Projeto será atribuída após a discussão, sendo a classificação do projeto ponderada pela qualidade da discussão**. As discussões poderão ser orais e/ou com suporte escrito.
- A não comparência à discussão resulta na classificação 0 (zero) para o respetivo aluno.

d) **Só serão considerados para avaliação projetos funcionais**, i.e., deverão compilar, nas condições abaixo indicadas, e executar.

- **É obrigatório que o projeto compile em GCC – GNU Compiler Collection.**

e) **A apresentação de relatórios ou implementações plagiadas leva à imediata atribuição de nota zero a todos os trabalhos envolvidos, quer tenham sido o original ou a cópia**.

f) No rosto do relatório e nos ficheiros de implementação **deverá constar o número e nome do estudante**.

g) Qualquer situação omissa neste enunciado e/ou nas regras de avaliação publicadas no SI é decidida pelo RUC.

1. Descrição do Projeto

Pretende-se desenvolver um programa em C para extrair/apresentar informação útil de um ficheiro com dados sobre as várias edições dos **jogos olímpicos**.

O programa consiste num interpretador de comandos que o utilizador usa para obter diversos tipos de informação, principalmente informação estatística.

1.1 Representação e armazenamento de dados (em memória)

Para este programa será necessário representar informação sobre **Atletas**, sobre **Medalhas** conquistadas pelos atletas e sobre os **Locais** (Países) de acolhimento do evento.

É obrigatória a manutenção em memória da informação importada utilizando os tipos de dados e ADTs (coleções) definidos seguidamente:

Para os Atletas:

- **Tipo de dados:**

  Cada registo de um atleta corresponde a informação sobre um atleta que tenha participado em pelo menos uma edição dos Jogos Olímpicos.

  Essa informação é guardada **obrigatoriamente** com o tipo de dados `Athlete` (ver Figura 1), definido num módulo apropriado.

- **ADT:** a coleção destes itens numa instância do ADT List, sendo **`ListElem`** do tipo `Athlete`.

Para as Medalhas conquistadas:

- **Tipo de dados:**

  A informação sobre as medalhas conquistadas é representada, **obrigatoriamente**, pelo tipo de dados `Medal` (ver Figura 1), definido num módulo apropriado.

- **Array/Estrutura de dados:** o conjunto destes itens num *array* (alocado dinamicamente, i.e., depende do tamanho do ficheiro de entrada) do tipo apropriado para guardar essa informação.

  - Recomenda-se que seja definido um tipo de dados adicional para conter este *array* alocado dinamicamente e o seu tamanho (e.g., similar à estrutura de dados *array list*).

Para os locais de acolhimento:

- **Tipo de dados:**

  Para poder guardar informação sobre o local de uma determinada edição dos Jogos Olímpicos, deverá definir um tipo de dados denominado `Host`. Este tipo de dados deverá permitir armazenar a informação encontrada no ficheiro de dados **`hosts.csv`** - é da responsabilidade do grupo de trabalho a sua definição.

- **ADT:** a coleção destes itens numa instância do ADT Map, sendo **`ValueElem`** do tipo `Host` e o **`KeyElem`** de um tipo apropriado que permita guardar uma *string* (o id dos jogos - campo `game_slug`).

```
#define MAX_ID_LENGTH 50
```

```c
#define MAX_NAME_LENGTH 100
#define MAX_GAME_LENGTH 50

typedef struct athlete {
    char athleteID[MAX_ID_LENGTH]; // Identificador único do atleta
    char athleteName[MAX_NAME_LENGTH]; // Nome do atleta
    int gamesParticipations; // Número de jogos em que participou
    int yearFirstParticipation; // O ano em que participou pela primeira vez
    int athleteBirth;// Ano de nascimento
} Athlete;

#define MAX_DISC_LENGTH 50
#define MAX_GAME_LENGTH 50
#define MAX_EVENT_LENGTH 100
#define MAX_ID_A_LENGTH 50
#define MAX_COUNTRY_LENGTH 50
#define MAX_GENDER 20

typedef struct medal {
    char discipline[MAX_DISC_LENGTH]; // Modalidade
    char game[MAX_GAME_LENGTH]; // Nome da edição dos jogos olímpicos (ex. beijing-2022)
    char eventTitle[MAX_EVENT_LENGTH]; // Título da prova (ex. Women's Ski cross)
    char gender[MAX_GENDER]; // Género da prova
    char medalType; // G - GOLD, S - SILVER, B - BRONZE
    char participantType; // A - Athlete, G - GameTeam
    char athleteID[MAX_ID_A_LENGTH]; // Identificador de cada atleta
    char country[MAX_COUNTRY_LENGTH]; // País pelo qual o atleta competiu
} Medal;
```

Figura 1 - Definição de tipos de dados.

1.2 Dados de entrada

São disponibilizados 3 ficheiros de dados:

- `athletes.csv` - Ficheiro de dados sobre os atletas
- `medals.csv` - Ficheiro de dados que contém informação sobre as medalhas conquistadas
- `hosts.csv` - Ficheiro de dados que as edições dos jogos e o país de acolhimento

Todos os ficheiros encontram-se em formato CSV; a primeira linha dos ficheiros é uma linha com os cabeçalhos e não contém dados.

1.2.1 Formato da informação

A informação está em formato CSV (comma separated values), sendo o separador o caractere '`;`':

Formato de linha em athletes.csv:
`<athlete_id> ; <athlete_full_name> ; <games_participations> ; <first_game> ; <athlete_year_birth>`

Formato de linha em medals.csv:
`<discipline_title> ; <slug_game> ; <event_title> ; <event_gender> ; <medal_type> ; <participant_type> ; <participant_title> ; <athlete_id> ; <country_name> ; <country_3_letter_code>`

Formato de linha em hosts.csv:
`<game_slug> ; <game_end_date> ; <game_start_date> ; <game_location> ; <game_name> ; <game_season> ; <game_year>`

1.2.2 Garantias e conversões necessárias

Pode assumir-se que não existem ficheiros "mal-formados", i.e., o formato CSV é respeitado.

Contudo, podem existir linhas com campos em branco, onde deverão ser feitos os seguintes ajustes na importação de informação:

- **Ficheiro `athletes.csv`**
  - Se o campo `<athlete_year_birth>` estiver vazio, o valor do ano de nascimento na estrutura de dados deve ficar como 0 (zero).
- **Ficheiro `medals.csv`**
  - Se o campo `<athlete_id>` estiver vazio e o campo `<participant_type>` for igual a "Athlete", o valor do id do atleta na estrutura de dados deve ficar como "MISSING" (em vez de vazio).

1.3 Restrições e casos omissos

❌ Não é permitido:

- alterar o conteúdo dos ficheiros de entrada (*.csv);
- alterar as estruturas apresentadas na Figura 1;
- alterar as interfaces lecionadas dos ADT, nomeadamente os ficheiros `list.h` e `map.h`;

❌ Não deverá assumir que:

- a informação nos ficheiros de entrada segue uma ordenação particular.
- o número de linhas de um ficheiro é fixo.

✔ Na implementação dos comandos descritos neste enunciado podem definir/utilizar outros tipos de dados auxiliares que se achem úteis para a resolução dos problemas.

✔ Têm liberdade total para tomar decisões nos casos omissos neste enunciado.

1.4 Comandos

Há exatamente **13 comandos que o programa deve implementar**, que serão apresentados de seguida; **3 comandos** para carregamento de dados, **8 comandos** para mostrar resultados de cálculos sobre os dados, **1 comando** para sair da aplicação e **1 comando** para limpeza dos dados em memória.

Os comandos têm o seguinte grau de dificuldade previsto: BAIXA, MÉDIA e ALTA.

Os comandos são também agrupados em categorias:

A. Funcionalidades de importação de dados;
B. Funcionalidades que requerem processamento da informação de atletas;
C. Funcionalidades que requerem processamento da informação de medalhas;
D. Funcionalidades que requerem processamento simultâneo de atletas, medalhas e/ou locais.

Notas:

- Cada comando é representado por uma palavra que pode ser escrita pelo utilizador em maiúsculas ou em minúsculas, não importa.
- Sempre que um comando necessitar de algum input, e.g., nome do país, este deve ser solicitado ao utilizador.
- Sempre que um comando necessitar de informação que não está carregada, o comando deve indicar que informação está em falta, i.e., **"No athlete data available"** e/ou **"No medal data available"**, **"No game edition data available"**.

Deverão seguir as indicações e exemplos dados sobre a forma como os resultados devem ser apresentados no ecrã. Nos casos omissos deverão escolher a representação mais clara e "*user friendly*".

A. Os *comandos base* são os seguintes:

✓ **HELP**

Mostra a lista de comandos do programa (os que não estiverem implementados deve indicar que não estão implementados).

Relembrar: O programa deve ter um interpretador de comandos e os comandos devem respeitar os nomes indicados no enunciado. Os comandos podem ser escritos em maiúsculas ou minúsculas.
Não devem usar qualquer comando de "clear screen" para se poder ver tudo o que é impresso.

✓ **LOAD_A**

Abre o ficheiro "`athletes.csv`" e carrega-o em memória (ver 1.2), mostrando o número de atletas importados, e.g., **"<M> atlethes records imported"**.

Se o ficheiro não puder ser aberto, escreve "**File not found**" e a coleção respetiva fica vazia.

✓ **LOAD_M**

Abre o ficheiro "`medals.csv`" e carrega-o em memória (ver Secção 1.2), mostrando o número de linhas lidas e o número de dados de medalhas importados, e.g., **<M> medals records imported"**.

Se o ficheiro não puder ser aberto, escreve "**File not found**" e a coleção fica vazia.

✓ **LOAD_H**

Abre o ficheiro "`hosts.csv`" e carrega-o em memória (ver 1.2), mostrando o número de edições de jogos importadas, e.g., **`"<N> hosts records imported"`**.

Se o ficheiro não puder ser aberto, escreve "**`File not found"`** e a coleção respetiva fica vazia.

✓ **CLEAR**

Limpa a informação atualmente em memória. Deverá indicar o número de registos que foram descartados de cada tipo, e.g., **`"Records deleted from Athletes (<N1>) | Medals (<N2>) | Hosts (<N3>)"`**

✓ **QUIT**

Sai do programa, libertando toda a memória alocada[1], e.g., para as coleções.

B. Os comandos de indicadores simples para Atletas (os cálculos requeridos só precisam de processar informação da coleção de Atletas)

As listagens devem ser mostradas paginadas de 20 em 20 linhas, com a possibilidade de passar para próxima página ou terminar a listagem. Antes de apresentar a listagem deve apresentar o número de registos encontrados.
O valor de 20 deve ser uma constante para poder se facilmente modificado no código se necessário.

Deverá implementar obrigatoriamente uma função de paginação com a assinatura:
`void paginate(PtList athletes);`
Esta função é responsável por implementar a funcionalidade de paginação dos registos contidos no parâmetro `atlethes`, incluindo a apresentação do número de registos encontrados.

✓ **SHOW_ALL**

Mostra os dados de todos os atletas – uma listagem paginada e ordenada alfabeticamente (A-Z) por nome de atleta.
Deve seguir o exemplo do resultado nos anexos apresentados no final do documento.

✓ **SHOW_PARTICIPATIONS**

Mostra os dados dos Atletas **que participaram nos jogos olímpicos, pelo menos x vezes**, com "x" solicitado ao utilizador – uma listagem paginada e ordenada alfabeticamente (A-Z) por nome de atleta.

Caso não existam resultados neste critério, escreve "**`No athletes found with at least <x> participations`**".

✓ **SHOW_FIRST**

Mostra os dados dos Atletas cuja sua primeira participação nos jogos olímpicos **foi num determinado ano**, solicitado ao utilizador – uma listagem paginada e ordenada alfabeticamente (A-Z) por nome de atleta.

Caso o ano que inseriu não tenha dados disponíveis na coleção, escreve "**`No athletes whose first participation was at <year>`**".

[1] Deverá verificar que o *Valgrind* acusa uma correta gestão da memória dinâmica.

C. Os comandos de indicadores para estatísticas sobre as medalhas ganhas e/ou locais de acolhimento

✓ **SHOW_HOST**

Mostra os dados relativos a um local de acolhimento (host), sendo solicitado ao utilizador o nome da edição (`game_slug`).
Deverá apresentar a seguinte informação:

- Cidade de acolhimento (extraído de `game_name`);
- Ano;
- País de acolhimento;
- Número de dias do evento.

Caso não existam resultados neste critério, escreve "`No edition found`".

✓ **DISCIPLINE_STATISTICS**

Mostra, para uma edição dos jogos olímpicos solicitada ao utilizador:

- O número de diferentes modalidades (`disciplines`) que estiveram presentes nessa edição;
- Para cada disciplina indicar:
  - Qual o país que ganhou mais medalhas;
  - Ratio entre o número de mulheres/número total de atletas.

Nota: Para garantir que não existem modalidades repetidas, **deverá recorrer a uma coleção de suporte do tipo ADT Set** (implementado segundo a especificação em 1.5).

D. Os comandos de indicadores complexos (os cálculos requeridos precisam dos dados da coleção de atletas, das medalhas e/ou dos locais de acolhimento)

✓ **ATHLETE_INFO**

Mostra para um determinado atleta, cujo ID (`athleteId`) é solicitado ao utilizador, a seguinte informação:

- País pelo qual participou;
- Número de participações;
- Ano de Nascimento;
- Edições em que ganhou medalhas;
- Para cada edição, indicar as medalhas ganhas e as respetivas modalidades.

Caso o atleta não exista, escreve "`Athlete <ID> did not found`";
Caso o atleta não tenha ganho medalhas, escreve "`Athlete <ID> did not win any medals`".

✓ **TOPN**

Mostra a lista dos **N atletas** com mais medalhas conquistadas num intervalo de anos e **por tipo de jogo** (Winter/Summer).
Deve seguir o exemplo do resultado nos anexos apresentados no final do documento (o exemplo tem também informação útil sobre cálculos auxiliares)

Deve solicitar ao utilizador:

- O valor **N**;
- O intervalo dos anos a considerar: **o ano de início** e o **ano de fim**;
- O **tipo de Jogos** ("Winter" ou "Summer").

Dica: Deve primeiro ver que edições ocorreram no intervalo de anos dados para o tipo de jogo indicado. Depois pode calcular as estatísticas (indicadas na próxima página) para os atletas que tiveram medalhas nessas edições (usando a estrutura de dados das medalhas).

Deve mostrar os seguintes dados (💡 aconselha-se a criação de uma estrutura auxiliar com estes dados):

- *Id do atleta*
- *País pelo qual o atleta participou.*
  *Caso o atleta tenha participado por mais que um país coloque o primeiro país pelo qual participou e um * no nome do país. Caso isto aconteça no final da tabela deve indicar:*
  `This athlete changed countries during this period`
  Não necessita de indicar os outros países.
- *Número total de Medalha ganhas pelo atleta;*
- *Média de Medalhas ganhas por Edição de Jogo (considere o número total de edições que cumpre os critérios de filtragem indicados);*
- *Média de Medalhas por dia de Jogos (considere a soma dos dias de cada edição que cumpre os critérios de filtragem indicados).*
  Deve ignorar as horas e usar só a data contando o dia de início e de fim

A lista deve também estar ordenada por ordem decrescente por medalhas ganhas: em caso de empate considere o nome do atleta por ordem alfabética.
Caso o intervalo de tempo que inseriu não tenha dados disponíveis, escreve "`No data found for the requested period`".

✓ MEDALS_WON

Mostra informação estatística do número de medalhas conquistadas por **um país em 5 edições sucessivas** nos géneros: *Men*, *Women* e *Mixed*.

Deve ser solicitado ao utilizador:

- O nome do país em análise (deve dar erro se o país não existir);
- O **tipo de Jogos** ("Winter" ou "Summer");
- O ano da primeira das 5 edições a considerar.

Detalhes:

💡 É apresentado um exemplo de *output* na secção Anexo.

Deverá apresentar os dados relativos a **5 edições sucessivas** dos jogos, considerando o tipo de Jogos. Exemplo: para 1924 e tipo Winter, deverá mostrar os dados de 1924, 1928, 1932, 1936 e 1940.
💡 Note que não é garantido ao longo da história que o "espaçamento" seja sempre de quatro anos.

Para cada uma das edições deverá apresentar **a seguinte informação:**

- Nome da edição;
- Ano da Edição;
- Designação da categoria (M - Men, W – Women, X - Mixed);
- Número total de medalhas de ouro conquistadas;
- Número total de medalhas de prata conquistadas;
- Número total de medalhas de bronze conquistadas.

Os resultados devem ser apresentando seguindo o seguinte layout:

```
EDITION         | YEAR | CATEGORY | G   | S   | B
----------------|------|----------|-----|-----|---
Chamonix 1924   | 1924 | M        | 01  | 01  | 01
Chamonix 1924   | 1924 | W        | 00  | 01  | 00
Chamonix 1924   | 1924 | X        | 00  | 00  | 00
St. Moritz 1928 | 1928 | M        | 02  | 02  | 01
....
```

1.5 Implementação de Tipo Abstrato de Dados (ADT Set)

Deverá desenvolver o ADT Set. Um Set (conjunto) caracteriza-se por ser uma coleção com elementos não repetidos; a sua especificação está apresentada na Figura 2.

Set is a collection of elements of a specific type. The elements in a set are distinct, meaning no duplicate elements are allowed.

Supported operations:

- ***Create**: Create a new empty set.*
- ***Add**: Adds an element to the set if it's not already present.*
- ***Remove**: Removes an element from the set if it exists.*
- ***Contains**: Checks if a given element is present in the set.*
- ***Size**: Returns the number of elements in the set.*
- ***Subset**: Checks if a set is a subset of another set.*
- ***Empty**: Checks if the set is empty.*
- ***Clear**: Removes all elements from the set, making it empty.*
- ***Values**: Retrieve an array with all the elements of the set*
- ***Print**: Prints the contents of the set to the console*
- ***Destroy**: frees resources associated with a set*

Figura 2 - Especificação do ADT Set.

💡 Deverá seguir a metodologia de desenvolvimento de ADTs lecionada na UC.

É expectável que utilize este ADT na resolução de uma funcionalidade; o enunciado sugere o comando **DISCIPLINE_STATISTICS**, mas pode encontrar, alternativamente, outro uso.

1.6 Git Classroom e repositório template

Todos os projetos deverão ser obrigatoriamente versionados através do Git Classroom.

O link do *assignment* e procedimento necessário de aceitação encontra-se no Moodle, junto com a este enunciado.

Após o procedimento, o grupo de trabalho ficará com um repositório Git privado apenas acessível pelos membros do grupo e pelos docentes da UC.

2. Relatório e Documentação

2.1 Documentação

Todo o código deve ser documentado utilizando a **documentação *Doxygen***.

A mesma deve ser gerada para formato HTML e entregue a respetiva pasta "html" junto com o projeto.

⚠ A documentação *doxygen* é requerida na entrega.

2.2 Relatório

No relatório deverão constar as seguintes secções (para além de capa com identificação dos alunos e índice). Pode utilizar o *template* de relatório fornecido no Moodle.

a) **Introdução**: Qual o propósito do trabalho e qual a metodologia de desenvolvimento adotada;

b) **Divisão de trabalho** – Descrição da participação de cada elemento:
   - Para cada comando a percentagem de participação de cada elemento;
   - Percentagem de participação de cada elemento para o projeto como um todo.

c) **ADTs Utilizados** - Descrição breve dos ADTs utilizados, qual a implementação escolhida e porquê (comparação de eficiências para o problema de aplicação). Deverá dar uma especial atenção ao ADT Set implementado, e neste caso detalhar as opções de implementação que efetuaram;

d) **Algoritmos** - Escolha de 3 funcionalidades do tipo B, C e/ou D, onde apresentam o algoritmo implementado em pseudo-código ou linguagem natural;

e) **Complexidades Algorítmicas** - Para cada comando implementado (exceto LOAD_X, CLEAR e QUIT) fornecer:
   - A complexidade algorítmica da respetiva implementação, tendo em conta as complexidades algorítmicas das funções dos ADTs utilizadas (dependem da implementação escolhida).

f) **Limitações** - Quais os comandos que apresentam problemas ou não foram implementados;

g) **Conclusões** - Análise crítica do trabalho desenvolvido, dificuldades sentidas, o que foi aprendido e que competências foram desenvolvidas.

3. Data de Entrega e *Deliverables* – 17 de Setembro, 14H

Na data-limite de entrega, deverá:

- Ter a última versão do projeto atualizada no repositório *GitHub Classroom* respetivo.
- Submeter no Moodle um ZIP:
  - Com nome: `<sigla_docente>_<nome_leader>_final_ATAD2324.zip`
  - Com o seguinte conteúdo: Projeto VS Code (cópia do repositório) + Relatório (secção 2) em PDF + pasta "html" da geração *doxygen*.

4. Critérios de Avaliação

A avaliação será ponderada de acordo com os seguintes princípios:

- **Estruturação:** o programa deve estar estruturado de uma forma modular e procedimental;
- **Correção:** o programa deve executar as funcionalidades, tal como pedido.
- **Legibilidade e documentação:** o código deve ser escrito, formatado e comentado de acordo com o standard de programação definido para a disciplina.
- **Desempenho:** Os algoritmos implementados devem ter em conta a complexidade do mesmo, valorizando-se a implementação de algoritmos com menor complexidade. A gestão da memória deverá ser feita corretamente, garantindo que a mesma é libertada quando não está a ser utilizada. Utilização da ferramenta *Valgrind*, para validar a correta gestão de memória.

São aplicadas as cotações apresentadas na Tabela 1 e penalizações apresentadas na Tabela 2.

Descrição	**Cotação (valores)**
Menu + HELP	0,5
QUIT + CLEAR + LOAD_A + LOAD_M + LOAD_H	4
SHOW_ALL	1
SHOW_FIRST	1
SHOW_PARTICIPATIONS	1
SHOW_HOST	1
DISCIPLINES_STATISTICS	1,5
ATHLETE_INFO	1
TOPN	1,5
MEDALS_WON	1,5
Implementação e utilização do ADT Set	2
Documentação e Relatório	3

Descrição	Cotação (valores)
Menu + HELP	0,5
QUIT + CLEAR + LOAD_A + LOAD_M + LOAD_H	4
SHOW_ALL	1
SHOW_FIRST	1
SHOW_PARTICIPATIONS	1
SHOW_HOST	1
DISCIPLINES_STATISTICS	1,5
ATHLETE_INFO	1
TOPN	1,5
MEDALS_WON	1,5
Implementação e utilização do ADT Set	2
Versionamento Git	1
TOTAL	**20**

Tabela 1 - Tabela de cotações.

A seguinte tabela contém penalizações a aplicar:

Descrição	Penalização (valores)
Uso de variáveis globais	até 2
Não utilização de modularidade	até 2
Incorreta gestão de memória dinâmica	até 2
Warnings de compilação	até 2
Uso de funções de bibliotecas que implementem algoritmos solicitados, e.g., `qsort` para ordenação.	até 3
Erros de compilação	Anulado
Não utilização dos ADTs obrigatórios	Anulado

Tabela 2 - Tabela de penalizações.

(fim de enunciado)

ANEXO

O formato obtido no desenvolvimento pode diferir do apresentado nesta secção, desde que contenha os mesmos elementos informativos.

Exemplo do comando SHOW_ALL

- Primeira página dos atletas ordenados por ordem alfabética. O primeiro atleta tem um espaço no nome (“ DENI DENI”) que alfabeticamente é menor que o “A”, seguem-se nomes iniciados com caracteres como as aspas e os pontos e só depois os iniciados por “A”.
  NOTA: A ordenação no Excel pode ser diferente por poder ignorar as aspas.

```
75900 ATHLETES FOUND
```

ATHLETE ID	FULL NAME	PARTICIPATIONS	FIRST PARTICIPATION	BIRTH YEAR
deni	DENI DENI	2	2012	1989
aleksandar-anton-strain	"Aleksandar ""Anton"" STRAIN"	1	1948	0
brand-bram-evers	"Brand ""Bram"" EVERS"	1	1908	0
frantisek-barry-stejskal	"Franti�ek ""Barry"" STEJSKAL"	1	1920	0
deni-x0003	. DENI	1	2020	1989
priyanka	. PRIYANKA	1	2020	1996
rahul	. RAHUL	1	2020	1996
a-baser-wasiqi	A Baser WASIQI	1	1996	1975
a-j-hurt	A J HURT	1	2022	2000
a-aziz-hassan-jaloof	A-Aziz Hassan JALOOF	2	1992	1973
a-darnis	A. DARNIS	1	1900	0
a-germaine-golding	A. Germaine GOLDING	1	1924	1887
a-linger-andreas-linger	A. Linger ANDREAS LINGER	4	2002	1981
a-turnovsky	A. TURNOVSKY	1	1924	0
a-j-miller	A.J. MILLER	1	1924	1899
abudureheman	ABUDUREHEMAN ABUDUREHEMAN	1	2000	1978
anuj-kumar-na	ANUJ KUMAR NA ANUJ KUMAR NA	1	2004	1980
aynutdin	AYNUTDIN AYNUTDIN	1	1980	1955
aadijatmiko-finarsih	Aadijatmiko FINARSIH	2	1992	1972
aage-avaldorff-meyer	Aage Avaldorff MEYER	2	1928	1904

```
Lines from 1 to 20


SHOWALL PAGINATED
1. Next 20
2. Return
```

Exemplo do comando TOPN

- Parâmetros:
  - N = 10
  - startYear = 2000
  - endYear = 2020
  - gameType = Winter

Athlete ID	Country	Total medals	Avg medals by edition	Avg medals by day
marit-bjoergen	Norway	12	2.00	0.12
ireen-wust	Netherlands	10	1.67	0.10
ole-einar-bjorndalen	Norway	8	1.33	0.08
arianna-fontana	Italy	7	1.17	0.07
charlotte-kalla	Sweden	7	1.17	0.07
martina-sablikova	Czech Republic	7	1.17	0.07
anastasiya-kuzmina-1	Slovakia	6	1.00	0.06
janica-kostel	Croatia	6	1.00	0.06
martin-fourcade	France	6	1.00	0.06
victor-an	Republic of Korea *	6	1.00	0.06

*** This athlete changed countries during this period**

- Informação extra de cálculos intermédios:

6 edições de inverno entre 2000 e 2020:
salt-lake-city-2002, turin-2006, vancouver-2010, sochi-2014, pyeongchang-2018, beijing-2022

Somatório de dias de todas as edições: 103 (17 dias para 5 edições e 18 dias para pyeongchang-2018).
Exemplo para pyeongchang-2018:
Data de fim e de início: 2018-02-25 e 2018-02-08
Dias: 8, 9, 10, 11, 12 ,13 ,14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 (18 dias)
Nota: Atenção aos casos em que muda de mês, devem considerar o número de dias do mês correspondente (no caso de fevereiro depende de ser ano bissexto).

Exemplo do comando MEDALS_WON

- Parâmetros:
  - country = United States of America
  - gameType = Winter
  - startingYear = 1924

Edition	Year	Category	G	S	B
Chamonix 1924	1924	M	01	01	01
Chamonix 1924	1924	W	00	01	00
Chamonix 1924	1924	X	00	00	00
St. Moritz 1928	1928	M	02	02	01
St. Moritz 1928	1928	W	00	00	01
St. Moritz 1928	1928	X	00	00	00
Lake Placid 1932	1932	M	07	03	02
Lake Placid 1932	1932	W	00	00	01
Lake Placid 1932	1932	X	00	02	00
Garmisch-Partenkirchen 1936	1936	M	02	00	04
Garmisch-Partenkirchen 1936	1936	W	00	00	00
Garmisch-Partenkirchen 1936	1936	X	00	00	00
St. Moritz 1948	1948	M	02	03	03
St. Moritz 1948	1948	W	01	01	00
St. Moritz 1948	1948	X	00	00	00