1. Arquitectura dos módulos C

* Modulos e a sua API

filehandling

main

hashing

funcoes\_processamento

tabAnos

util

estruturas

main:

* Modulo principal que gere todo o programa;
  + utiliza o modulo filehandling e o modulo hashing;

filehandling:

* Modulo que abre o lista.txt e retira a informação necessária dele, cria os E.txt, D.txt e G.txt. Envia ainda as informações relevantes dos artigos aceites para o modulo tabAnos;
  + utiliza o modulo funções de processamento, o modulo tabAnos , o modulo estruturas e o modulo util;

int readJ(char\* fileName, int \*revRej);

(abre e processa os j-\*.txt devolve quantos leu e por referencia quantos rejeitou)

int readC(char\* fileName, int \*confRej);

(abre e processa os c-\*.txt devolve quantos leu e por referencia quantos rejeitou)

int read\_list(char \* basePath, const char \*fileName);

(abre e processa o lista.txt)

void logRejected(const char \*fileName, int nRejected);

(cria o E.txt)

void logStats(int confsAceites, int revsAceites, int confsRej, int revsRej);

(cria o D.txt)

void logIntervalo(int anoInicial, int anoFinal, int nArtigos);

(cria a parte dos intervalos no G.csv)

void logAutoresArtigo(int nAutores, int nArtigos);

(cria a parte dos Autores-Artigo no G.csv)

void logAnAutArt(int ano, int nAutores, int nArtigos);

(cria a parte dos Ano - Autores - Artigos no G.csv)

void logAnosAutoresPercentagem(int ano, int nAutores, double percentagem);

(cria a parte dos Ano - Autores - Percentagem no G.csv)

void logIntervaloTIT();

(cria o titulo da função logIntervalo)

void logAutoresArtigoTIT();

(cria o titulo da função logAutoresArtigo)

void logAnAutArtTIT();

(cria o titulo da função logAnAutArt)

void logAnosAutoresPercentagemTIT();

(cria o titulo da função logAnosAutoresPercentagem)

int readIntervalos(char \*fileName);

(lê o datas3.txt e processa a partir dos intervalos que leu)

int gestorFilehandling (char \*basepath, const char \*filename);

(controla a ordem da execução de todos os processos no filehandling)

int initG();

(inicia G.csv (para não estar a ser aberto sempre que é utilizado))

void closeG();

(fecha o G.csv)

hashing:

* Modulo que gere a rede de Autores e possui todas as funções que podem aceder à tabela de hash;
  + utiliza o modulo util e o modulo estruturas;

int processaAutores(const char\* str);

(parte uma lista de autores e insere-os na HashTable)

int qntsHT();

(conta quantos autores estão na HashTable);

void q4Top10Artigos();

(responde à pergunta 4)

void q3maxArtigosComum();

(responde à pergunta 3)

int q2qntsArtigosComum(char\* autor, char\* coautor);

(responde à pergunta 2)

int q1qntsAutoresEscreveu(char\* autor);

(responde à pergunta 1)

funcoes\_processamento:

* Modulo que recebe um artigo e o parte de forma a todas as informações necessárias serem postas numa estrutura temporária e ainda decide se aceita ou rejeita um determinado artigo;
  + utiliza o modulo estruturas, o modulo hashing;

Artigo \*retira\_dados\_REV(const char\* dados, int len);

(parte o artigo de revista nos seus campos, ie, autor, ano, pagina inicial, pagina final... e devolve uma struct com os dados necessários)

Artigo \*retira\_dados\_CONF(const char\* dados, int len);

(parte o artigo de conferencia nos seus campos, ie, autor, ano, pagina inicial, pagina final... e devolve uma struct com os dados necessários)

char\* strLower(const char\* str);

(converte uma string para só minusculas);

tabAnos:

* Modulo que contém as informações sobre os artigos (ano, número de autores e número de artigos) e possui as funções de cálculo sobre essas informações que depois são enviadas para o filehandling;
  + utiliza o modulo estruturas,o modulo util e o filehandling;

void \*initTA();

(inicializa a estrutura da tabela de autores dinamica e estatica)

void \*insertTA(TabAutores \*tabA, int ano, int nautores);

(insere um elemento na tabela de autores)

int intervalosAnos(int anoInicial, int anoFinal);

(calcula o número de artigos de um intervalo de anos -ex 2001-2003 14141)

int nAutoresArtigo();

(calcula o numero de artigos por determinado numero de autores -ex 1 20303)

int anoAutoresArtigo();

(calcula o numero de autores e artigos de um determinado ano -ex. 2010 3 131)

int criaG();

(cria o G.csv com todos os elementos)

estruturas:

* Modulo que contém quase todas as estruturas utilizadas ao longo programa;
  + não utiliza nenhum modulo;

util:

* Modulo que possui funções auxilares a vários modulos;
  + não utiliza nenhum modulo;

void clear\_str(char\* array, int length);

(povoa uma string com o caracter '/0')

void freeStatus(Status\* toFree);

8

7

6

5

4

3

2

1

(apaga da memoria um determinado Status)

figura 1

3

4

5

6

7

8

1

2

char\* nome;

char\* key

int maxComum

HashTableEntry \*seg

Autor \*coautores

char\* nome

int artigosComum

Autor \*seg

1) HashTE \*HashTable [CAP]

3) Autor

2) hashTableEntry

4)

2

1

char\* trim(char\* str);

(retira os caracteres ' ' do inicio de uma String)

int numeroAutores(char\* listaAut);

(calcula o numero de autores de uma String);

* Estruturas fundamentais
  + HashTable (modulo hashing)

A HashTable apresentada na **figura 1** é a estrutura utilizada para guardar o nome dos autores e os autores com quem trabalharam. É uma estrutura que apesar de ter uma dimensão base torna-se muito dinamica devido as inserções de coautores e pela forma como são tratadas as colisões.

A HashTable é a estrura base (ponto 1) com uma capacidade (CAP) pré definida e que não muda quer sejam inseridos mais ou menos autores. Cada posição da HashTable é composta por uma estrutura como a do ponto 2, estrutura esta denominada de HashTableEntry (daqui para a frente denominada com HashTE), esta possui um campo nome (com o nome do Autor que ocupa essa posição), um campo key (usada para calcular a o hashcode para essa autor), um Autor (parte dinamica que será falada de seguida), um maxComum que guardará o valor maximo entre o autor e o coautor com quem escreveu mais Artigos e uma HashTE \*seg (caso haja colisões, também será abordado mais a frente). Um autor (ponto 3) possui um nome (nome de um coautor), o artigosComum (numero de artigos que esse coautor e o autor possuem em comum) e por fim um Autor \*seg onde ficarão guardados os restantes coautores. Por fim, referindo o ponto 4, podem existir colisões, ie, ser calculada pela hash uma posição que já se encontrava ocupada e neste caso utiliza-se o HashTableEntry \*seg, que criará uma lista ligada com as colisões.

* + - Top10Entry (modulo estruturas mas com ligação directa e unica ao modulo hashing)

Esta estrutura (**figura 2**) possui apenas dois campos, um inteiro nCoautores com o numero de coautores e um nome (nome de um autor). Esta estrutura é utilizada apenas como apoio para calcular o top10 (questão 4, fase 2b).

int nCoautores

char\* nome

figura 2

* + Tabela de Autores (modulo tabAnos)

A tabela de Autores (**figura 3**) possui uma estrutura para guardar os elementos necessários para gerar estatisticas denominada de TabAutores. A TabAutores é composta por duas partes, uma parte estática denominada de NautSTA (ponto 1) que possui uma matriz de dimensões 114x10 e uma parte dinámica (ponto 2) chamada NautDIN, esta estrutura dinamica é populada por todos os elementos que o ano seja menor do que 1900 ou maior que 2013 e ainda se o número de autores for superior a 10. Dentro de cada NautDIN existe um ano (ano do artigo), um nAutores (numero de autores que o escreveram), um nArtigos (numero de artigos que foram escritos nesse ano e com esse numero de autores) e ainda um nAutDIN \*seg para compor com mais anos e/ou com mais numero de autores. Esta estrutura é preenchida com uma função bastante complexa de forma a que a estrutura dinamica fique ordenada quer no numero de autores quer nos anos, facilitando e tornando mais rapido o seu acesso a nivel futuro.

int ano

int nAutores

int nArtigos

nAutDIN \*seg

1) NautSTA

2) NautDIN

figura 3

...

* + Artigo (estrutura temporaria para mover a informação entre o modulo funcoes de processamento e o modulo tabAnos)

char \* autores

int numAutores;

int ano;

figura 4

A estrutra Artigo (**figura 4**) é apenas temporária e serve para transportar o que as funções de processamento obtiveram a partir de cada Artigo. Esta estrutura possui um campo autores (campo este que deveria ter sido retirado mas devido a esquecimento manteve-se, este campo é desnecessário pois a propria função que recolhe a informação de um artigo insere os autores na HashTable), um campo numAutores (o número de autores que escreveram esse artigo) e ainda um campo ano (ano do artigo).

* Descrição do comportamento da estrutura de dados

As estruturas de dados descritas em cima como dinamicas variam conforme o numero de Artigos que são processados. A hashTable tem um tamanho fixo de posições mas quantos mais autores maior a probabilidade de haver colisões e portanto com cada colisão o seu tamanho aumenta. Com o maior número de autores também a estrutura Autor (ligada à hashTable) aumenta pois haverá maior número de coautores.

No caso da tabela de Autores a parte dinamica (a lista ligada) aumenta ou diminui conforme existam mais ou menos autores para além dos 10. Pois se for menor do que 10 este vai para a parte estática não havendo qualquer aumento de ocupação de memória.

* Dimensionamentos base

A hashTable começa com 150001 posições. Porquê este valor? Este valor foi definido com base no teste1fase2, pois o número de autores diferentes encontrava-se muito perto dos 150 mil e 150001 é o primeiro número primo a seguir ao valor referido. É certo que com menor número de posições também funciona mas haverá maior numero de colisões.

A matrix tabAnos com dimensões 114x10 possui este mesmo tamanho porque 95% dos artigos têm 10 ou menos artigos portanto é aceitavel ter apenas 10 posições para o número de autores, já o 114 é o segmento de 1900 a 2014 porque parece-me improvável que haja artigos para além desses anos, quer para menos ou para mais (mais chega a ser impossivel, enquanto). Mas caso os haja também a parte dinamica assegura que não serão excluidos.

O MAXINT com 10 e o TAM\_BASE com 1024 são valores que achei serem ideais, o primeiro porque o MAXINT é usado para o maximo em guardar anos e nenhum ano usa 10. O TAM\_BASE porque nenhum artigo possui mais do que 1024 caracteres.