UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSOS DE CIÊNCIA E ENGENHARIA DA COMPUTAÇÃO

DISCIPLINA DE CLASSIFICAÇÃO E PESQUISA DE DADOS

PROF. LEANDRO KRUG WIVES

JÚLIA MOMBACH DA SILVA (00281023)

JULIANA RODRIGUES DE VARGAS (00337553)

TRABALHO FINAL – ETAPA III

PORTO ALEGRE, ABRIL DE 2023

O presente trabalho visa desenvolver os conceitos aprendidos durante as aulas da disciplina de Classificação e Pesquisa de Dados da Universidade Federal do Rio Grande do Sul. Para tal, foi produzido um programa que gera frases aleatórias através de modelos pré prontos, inserindo palavras do dicionário em partes determinadas. O algoritmo foi desenvolvido na linguagem C++ em função de sua rapidez para execução, entretanto, a limpeza dos dados foi feita em Python para que algumas funcionalidades da linguagem fossem aproveitadas.

A base de dados utilizada foi o Dicionário Priberam da Língua Portuguesa*, que possuí palavras classificadas por classe, gênero e grau. Utilizamos esses parâmetros para definir a posição em que cada palavra poderia ser posta nos nossos modelos de frases.

Justificativa? O programa tem fins recreativos

*Dicionário Priberam da Língua Portuguesa , edição em português do Brasil para Kindle, junho 2011

LIMPEZA DE DADOS

O dicionário foi baixado em formato pdf no site PDFDRIVE e foi transformado em um arquivo txt a partir do site Convertio. Então, foram definidos diversos critérios e, utilizando a linguagem Python, foi criado um arquivo CSV, com cada linha representando uma palavra nova e cada coluna com um respectivo parâmetro. No fim da limpeza ficamos com 94.411 palavras.

	A	B	C	D	E
2285	afetividade	s.	f.		1. Faculdade afetiva, qualidade do que é afetivo. 2[Psicologia] Função geral, sob a qual se colocam os fenômenos afetivos
2286	afetivo	adj.			1. Em que há afeto. 2. Que mostra afeto ou afeição. = AFETUOS3. Relativo aos afetos ▸ Etimologia: latim affectivus, -a, -u
2287	afeto	s.	m.		1. Impulso do ânimo, sua manifestação. 2. Sentimento, paixão3. Amizade, amor, simpatia. • adj. 4. Dedicado, afeiçoado. 5. Incumbidoentregue
2288	afetuosamente	adv.			De modo afetuoso ▸ Etimologia: afetuoso + -ment
2289	afetuosidade	s.	f.		1. Qualidade de pessoa afetuosa, sentimento de afeiçãprofunda. 2. Afetividade
2290	afetuoso	adj.			1. Que demonstra afeto. 2. Meigo, carinhoso, afável. 3[Música] Brando e ternamente. • [Brasil] Plural: afetuosos |ó|. • [Portugal] Pluralafetuosos |ó| ▸ Etimologia: italiano affectuos
2291	afiação	s.	f.		1. Ato ou efeito de afiar. 2. Amoladura. 3. Aguçamento
2292	afiadeira	s.	f.		O mesmo que apara-lápis
2293	afiado	adj.			Que tem fio cortante, aguçado.afiador |ô| adj. s. m. 1. Que ou o que afia. 2. Amolador. 3. Instrumento pardar fio. 4. O mesmo que apara-lápis
2294	afia-lápis	s.	m.	pl.,si.	O mesmo que apara-lápis
2295	afiambrado	adj.			1. Diz-se do indivíduo muito apurado no trajar. 2. Carnpreparada à maneira de presunto
2296	afiambrar	v.,tr.			1. Preparar carne à maneira de fiambre. • v. pron. 2Esmerar-se no trajar. 3. Apropriar-se▸ Etimologia: a- + fiambre + -a
2297	afiançado	adj.			1. Abonado, acreditado, fiado. 2. [Jurídico, Jurisprudência] Qupresta fiança
2298	afiançador	adj.,s.	m.		Que ou o que afiança
2299	afiançar	v.,tr.			1. Prestar fiança por. 2. [Figurado] Assegurar. 3. Abonar. 4[Popular] Agarrar. • v. pron. 5. Prestar fiança
2300	afiançável	adj.	m.,f.		1. Que pode ser afiançado. 2. Que se pode afiançar. Antônimo geral: INAFIANÇÁVE ▸ Etimologia: afiançar + -ável
2301	afiançavelmente	adv.			De modo afiançável ▸ Etimologia: afiançável + -ment
2302	afiar	v.,tr.			1. Dar fio a. 2. Aguçar. 3. [Figurado] Apurar (os dentes), afiar o.dentes. 4. Preparar-se para comer bem
2303	aficar	v.,intr.,pron.			[Antigo] Insistir, teimar
2304	aficionado	adj.			1. Entendido em arte que não exerce. 2. [TauromaquiaApaixonado por touradas, amador ▸ Etimologia: espanhol aficionad
2305	afidalgado	adj.			Que tem ares ou maneiras de fidalgo
2306	afidalgamento	s.	m.		Ato de afidalgar ou de se afidalgar
2307	afidalgar	v.,tr.			1. Tornar fidalgo. • v. pron. 2. Adquirir modos de fidalgo, darse ares de fidalgo
2308	afídios	s.	m.	pl.	Nome científico dos pulgões
2309	afidívoro	s.	m.		Artrópode que se alimenta de pulgões
2310	afifar	v.,tr.,intr.			[Informal] O mesmo que afinfar ▸ Etimologia: alteração de afinfa
2311	afiguração	s.	f.		1. Ato ou efeito de afigurar. 2. Aparência, fantasia
2312	afigurar	v.,tr.			1. Dar figura a. • v. pron. 2. Mostrar-se na figura de. 3Imaginar, parecer.afigurativo adj. 1. Que encerra figura ou parábola, que afigura. 2. Figurado
2313	afilado	adj.			1. Aferido. 2. Adelgaçado. 3. Pontiagudo

Como podemos ver na imagem, a primeira coluna possui a palavra, a segunda a classe, a terceira o gênero, a quarta o grau e a quinta o significado.

CONSTRUÇÃO DOS ARQUIVOS BINÁRIOS

Há os seguintes arquivos binários:

1 - dictionary.bin
Esse arquivo contém as structs correspondentes a cada palavra do dicionário de entrada. As structs são construídas contendo ID, palavra, classe, gênero, número, significado, e um boolean para saber se a palavra está deletada ou não.

O preenchimento deste arquivo binário é feito pela função fillfFile, que recebe os seguintes parâmetros:

```
int fillFile(string arq_data_is, string arq_binary_is);
```

```
struct Word {
    int ID;
    string palavra;
    string classe;
    string genero;
    string numero;
    string significado;
    bool deleted;
};
```

2. adjetivos.bin, preposicoes.bin, substantivos.bin e verbos.bin
Esses são os arquivos invertidos usados como índices para o trabalho, e indexam as palavras. São arquivos binários que contém a struct Entry.

```
struct Entry {
    WordKey entryWord;
    streampos pos;
    int ID;
};
```

Nota se que a struct contém o elemento WordKey. Essa é uma classe construída para ser usada no lugar de strings, pois é sempre do mesmo tamanho e facilita as operações sobre os arquivos binários.

As função utilizada para o preenchimento deste arquivo é a seguinte:

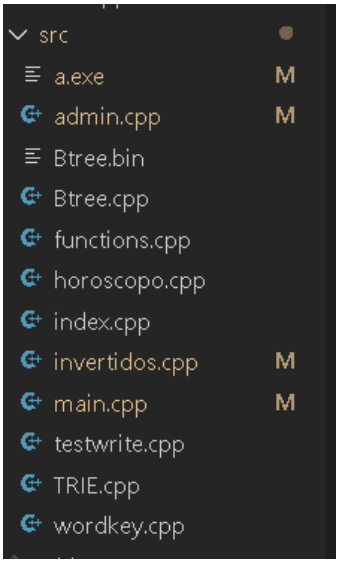
```
int generateInverted(string classe, string nomeArq);
```

3. int.bin, NumEntrys.bin

Estes são arquivos binários que servem para permanência de dados, mantendo sempre atualizado quantos registros há em cada arquivo, o que facilita pesquisa e inserção.

ESTRUTURA DO PROGRAMA:

O programa se organiza nos seguintes diretórios:

- data: que contém os dados utilizados para o funcionamento do trabalho, como dictionary.bin e NumEntrys.bin, e também um arquivo de boas frases, que pode ser usado para futuras implementações;
- invertidos: que contém os arquivos binários invertidos;
- include: que contém o arquivo de header principal do programa, bib.h, e o arquivo de header da classe WordKey criada;
- listagem: contém os txt que estão disponíveis para o usuário;
- src: contém os arquivos .cpp do trabalho
- 

  As funções presentes em cada arquivo .cpp estão separadas no arquivo principal de header do trabalho.

  Estas foram as bibliotecas utilizadas:

```cpp
#include <iostream>
#include <fstream>
#include <cstring>
#include <locale>
#include <vector>
#include <string>
#include <sstream>
#include <unistd.h>
#include <algorithm>

#include "wordkey.h"

using namespace std;
```

ALGORITMOS DESENVOLVIDOS

A seguir, faremos uma descrição apropriada dos principais algoritmos envolvidos no funcionamento do trabalho, assim como as principais dificuldades encontradas em sua implementação.

1. **BTree**

    Embora a BTree não apareça diretamente no trabalho, boa parte do trabalho foi dedicada à sua implementação. Entretanto, houveram dificuldades que nos fizeram optar pelos arquivos invertidos. A implementação conteria as seguintes structs:

```cpp
struct Key {
    int ID;
    WordKey word;
    streampos address;
};

//node declaration
struct Node {
    Key keys[t];
    streampos children[t+1];
    bool isLeaf;
    int numChildren;
};
```

E as seguintes funções:

```
//FUNCTIONS BTREE
Node init();
streampos putInArq(Node node);
Node readInArq (streampos pos);
int swapStruct(streampos pos, Node node);
void traverse(streampos p);
void sort(Key *keys, int numKeys);
Key split_child(streampos x, int i);
void insert(Key key);
```

O objetivo para o gerenciamento da árvore em memória secundária era: manter os nodos e os respectivos endereços de seus filhos (variável streampos para cursor em arquivo) em memória secundária, trazendo apenas os nodos que estivessem sendo operados para a memória principal, e em seguida atualizando-os no arquivo com a função swapStruct. Para que os nodos tivesse sempre o mesmo tamanho, também foi necessário a utilização acima citada da classe Wordkey.

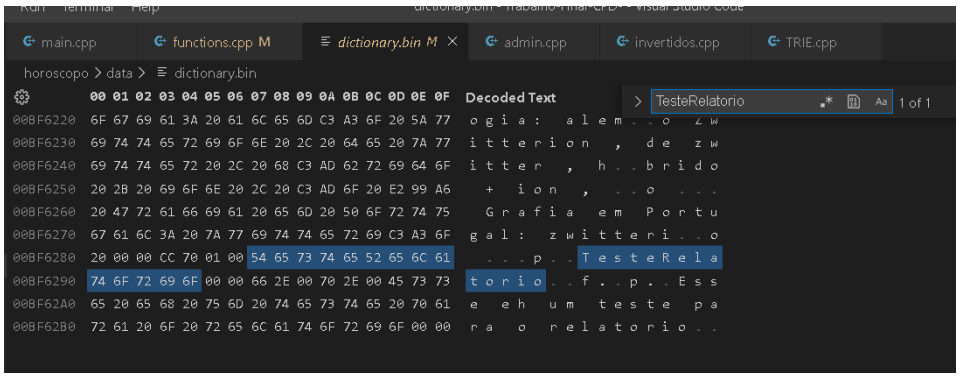
Entretanto, encontramos dificuldade para implementar split_child quando havia necessidade de também realizar o split na raíz, e por isso esta árvore não foi utilizada.

2. Inserção em dictionary.bin

A inserção no arquivo binário é feita levando em consideração o número de IDs (que está salvo no binário numEntrys.bin) e atualizando-se esse valor. A inserção é realizada no final do arquivo.

```
streampos insertWordFinal(Word word)
```

```
Qual palavra você deseja incluir? Preencha as informações:
Palavra: TesteRelatorio

Genero: f.

Numero: p.

Significado: Esse eh um teste para o relatorio

Arquivo inserido com sucesso!Digite 1 para voltar para Home. Outro valor para sair.
5
```



Aqui é possível perceber que uma inserção é feita com sucesso no final de dictionary.bin

3. Inserção nos arquivos invertidos

Além disso, quando uma palavra é inserida em dictionary.bin, seu registro é criado e inserido no arquivo invertido respectivo. Para isto, é feita uma busca binária que devolve a posição para o registro ser inserido. Depois disso, é chamada uma função que desloca todos os registros abaixo dessa posição, e escreve um novo registro nela.

```
streampos binarySearchPos(string nomeArq, string targetWord)
```

```
void writeEntryPosition(string filename, Entry newEntry, streampos pos)
```

4. Listagem em ordem crescente e decrescente

As palavras nos arquivos de índice podem ser listas para o usuário, com a geração de um arquivo .txt com o nome que ele escolher. Essa listagem pode ser feita tanto de forma crescente como de forma decrescente. Exemplos dessa listagem estão no diretório *listagem*.

Abaixo, segue um trecho do algoritmo para a listagem em ordem decrescente, que acessa o indíce pelo final e percorre subtraindo sizeof(Entry)

```
        // Obtenha o tamanho do arquivo em bytes.

        file.seekg(0, std::ios::end);

        std::streampos size = file.tellg();
```

```
          // Calcule o número de registros no arquivo.

          std::size_t num_records = size / sizeof(Entry);

           // Posicione o ponteiro de leitura/gravação no final do
arquivo.

            file.seekg(-static_cast<std::streamoff>(sizeof(Entry)),
std::ios::end);

          // Leia os registros do final para o início.

          for (std::size_t i = num_records; i > 0; --i) {
            Entry entry;

             std::streampos                posic                =
static_cast<std::streampos>(i-1) * sizeof(Entry);

            file.seekg(posic);

                        file.read(reinterpret_cast<char*>(&entry),
sizeof(Entry));

            string palavra;

             palavra = entry.entryWord.toString();

             arquivo << palavra << endl;

          }
```

5. Criação de chave aleatória para a frase

Para a aleatorização das palavras criadas, foi feito um algoritmo que gera uma chave aleatória que corresponde a um endereço dentro de um arquivo invertido. Essa função leva em consideração o tamanho de cada arquivo invertido, que estão sempre atualizados em int.bin. A aleatorização segue a seguinte expressão:

```
    aleatorio = rand();
```

```
    random = abs((fator*aleatorio) % tam-1);

    random = random*sizeof(Entry);
```

Em seguida, esse endereço é buscado nos arquivos invertidos.

6. **Busca nos arquivos invertidos**

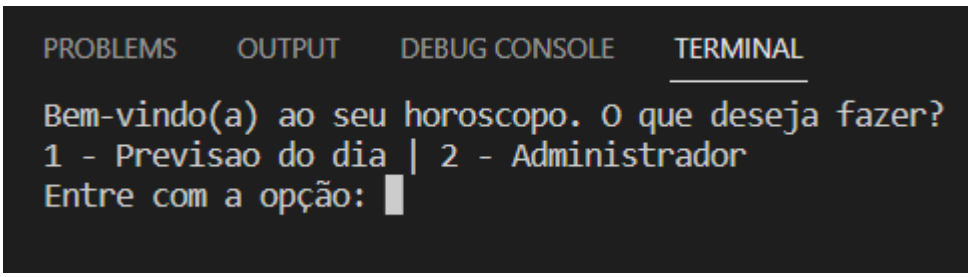
A busca nos arquivos invertidos é feita de acordo com a seguinte função:

```
Entry findInverted(streampos pos, string nomeArq){

    Entry auxEntry;

    ifstream invertedRead(nomeArq, ios::binary);

    invertedRead.seekg(pos);

    invertedRead.read((char*)&auxEntry, sizeof(Entry));

    return auxEntry;

}
```

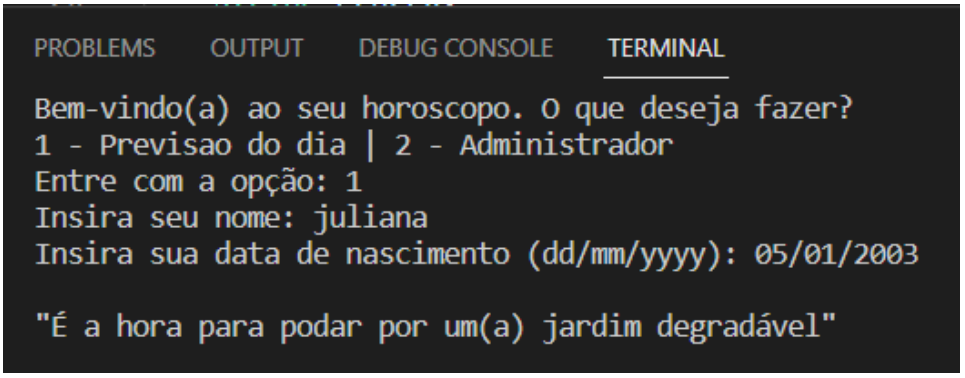
Ela pode ser utilizada para qualquer um dos arquivos, pois recebe o nome do arquivo a ser realizada a busca como parâmetro.

FUNCIONALIDADES DO PROGRAMA:

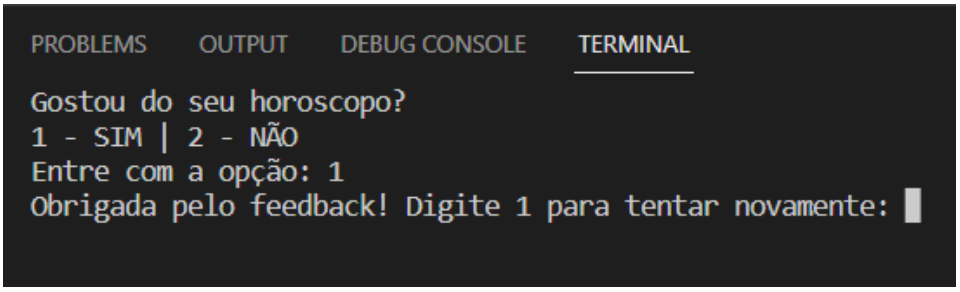
Página inicial:



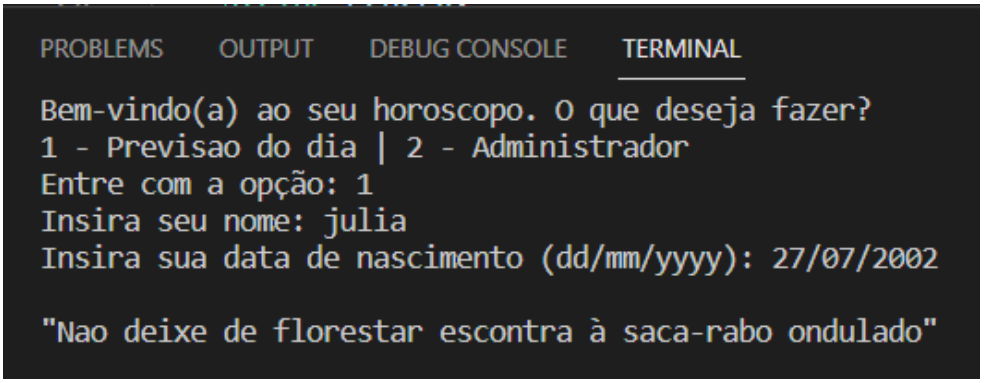
Caso a opção escolhida seja 1, é solicitado nome e data de nascimento do usuário. A partir disso é gerada uma frase aleatória.



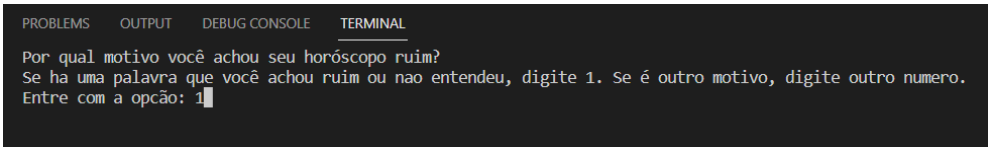
Então é solicitado o feedback do usuário. Caso a resposta seja positiva, o usuário retorna para a página inicial ou sai do programa.



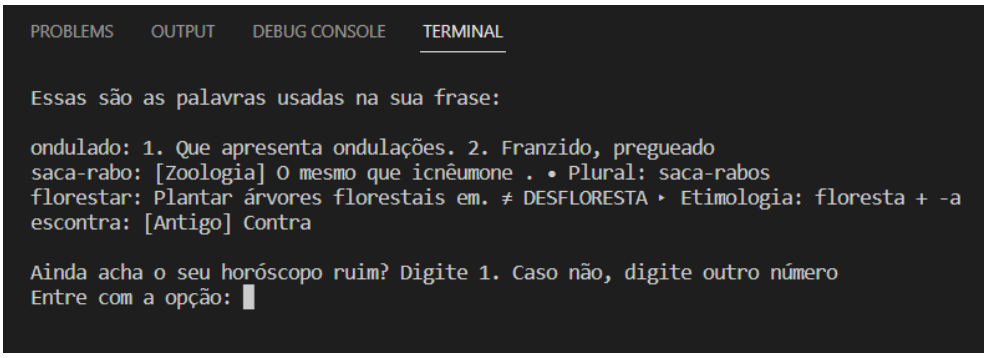
Vamos para outro caso: uma nova frase foi gerada para um novo usuário.



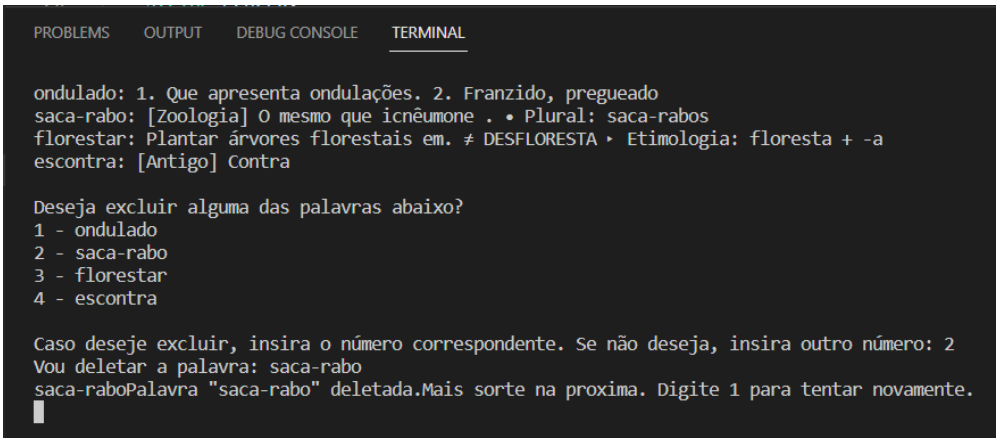
Aqui, o feedback foi negativo.



O usuário entrou com a opção 1, então foram listadas todas as palavras sorteadas junto com seus respectivos significados. Ou seja, *dada uma consulta no arquivo invertido por ID, foi buscada a informação no arquivo binário do significado das palavras.*



O usuário digitou 1, então é perguntado se deseja excluir uma das palavras do banco de dados. Então foi pedida a exclusão da palavra 2.

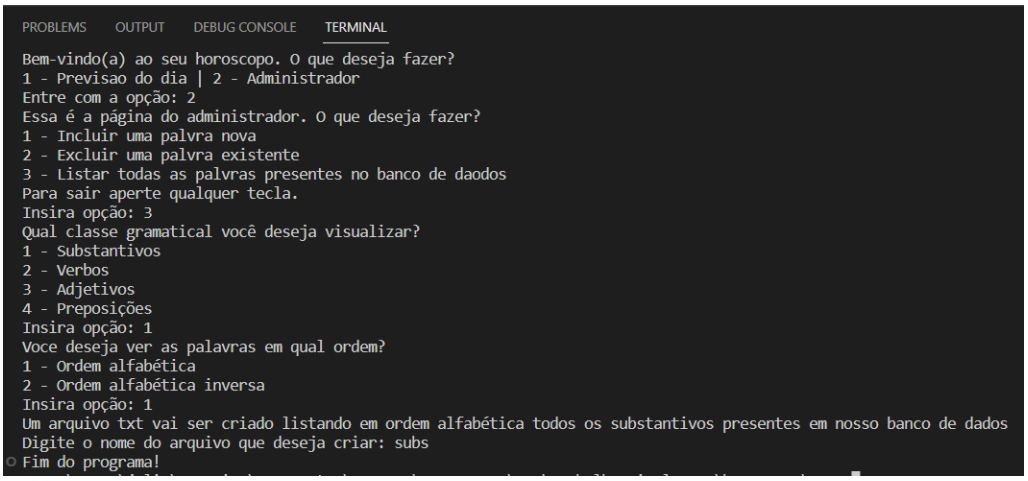


A exclusão consiste na alteração no boolean “excluded” da struct Word para verdadeiro, e na alteração do ID da palavra no arquivo invertido para -1;

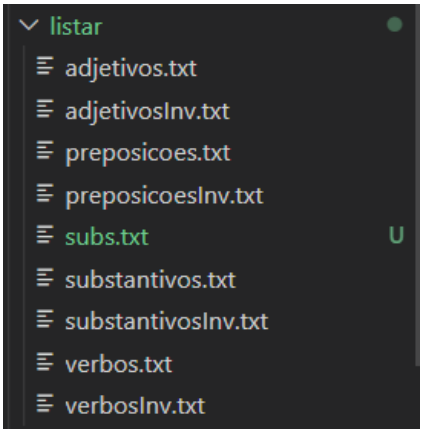
Página do administrador:

Outra opção presente na página inicial é a página do administrador. São dadas 3 opções: *Incluir* uma palavra nova, *excluir* uma palavra existente e *listar* todas as palavras

presentes no banco de dados. Nesse caso, o usuário solicitou a listagem das palavras da classe substantivo em ordem alfabética. O nome que o usuário solicitou para o arquivo foi “subs”.



O arquivo foi criado na pasta ‘listar’, presente no diretório com o nome escolhido pelo usuário.



CONSIDERAÇÕES FINAIS

Este foi um trabalho de difícil implementação. As principais dificuldades encontradas foram (1) Poucos materiais sobre implementação de árvores em memória em ++ disponíveis na internet e (2) A manipulação de arquivos e seus endereços, que exige atenção aos detalhes. Possíveis alterações futuras consistem em (1) implementação adequada da BTree e (2) Interface gráfica mais agradável que o terminal. Entretanto, foi de muita utilidade para compreender o gerenciamento de arquivos importantes para um programa em memória principal.

ELEMENTOS DE TERCEIROS E REFERÊNCIAS

Código pego como base para a implementação da BTree:

C Program to Implement B Tree. Disponível em: <https://www.tutorialspoint.com/cplusplus-program-to-implement-b-tree>. Acesso em: 11 abr. 2023.

Auxílio para a implementação da busca binária nos arquivos invertidos e na leitura e escrita com endereços:

ChatGPT. Disponível em: <https://chat.openai.com/chat/a57102a5-8666-4de9-9957-f880aa575c6b>. Acesso em: 11 abr. 2023.