Filtr przedrostkowy (PK2 Projekt)

Wygenerowano przez Doxygen 1.8.17

Rozdział 1

Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury BufferedFileReader

`#include <BufferedFileReader.h>`

Atrybuty publiczne

- FILE * **file**

  Uchwyt do pliku.
- char **buf** [**BFR_BUF_SIZE**]

  Bufor do wczytywania danych.
- size_t **readCount**

  Ilość dostępnych znaków w buforze.

3.1.1 Opis szczegółowy

Buforowany wczytywacz (reader) danych z pliku. Wczytuje dane z pliku porcjami w celu zwiększenia wydajności.

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 buf

`char BufferedFileReader::buf[ BFR_BUF_SIZE]`

Bufor do wczytywania danych.

3.1.2.2 file

```
FILE* BufferedFileReader::file
```

Uchwyt do pliku.

3.1.2.3 readCount

```
size_t BufferedFileReader::readCount
```

Ilość dostępnych znaków w buforze.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/Filter/ **BufferedFileReader.h**

3.2 Dokumentacja struktury LaunchConfig

```
#include <ParseArgs.h>
```

Atrybuty publiczne

- char **inFilePath** [FILENAME_MAX]
  ścieżka do pliku wejściowego.
- char **prefixFilePath** [FILENAME_MAX]
  ścieżka do pliku z przedrostkami.
- char **outFilePath** [FILENAME_MAX]
  ścieżka do pliku wyjściowego.

3.2.1 Opis szczegółowy

Informacje o konfiguracji uruchomieniowej programu.

3.2.2 Dokumentacja atrybutów składowych

3.2.2.1 inFilePath

```
char LaunchConfig::inFilePath[FILENAME_MAX]
```

ścieżka do pliku wejściowego.

3.2.2.2 outFilePath

```
char LaunchConfig::outFilePath[FILENAME_MAX]
```

ścieżka do pliku wyjściowego.

3.2.2.3 prefixFilePath

```
char LaunchConfig::prefixFilePath[FILENAME_MAX]
```

ścieżka do pliku z przedrostkami.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/Filter/ **ParseArgs.h**

3.3 Dokumentacja struktury PairOfStrings

```
#include <String.h>
```

Atrybuty publiczne

- **String left**
- **String right**

3.3.1 Opis szczegółowy

Para napisów.

3.3.2 Dokumentacja atrybutów składowych

3.3.2.1 left

```
String PairOfStrings::left
```

3.3.2.2 right

```
String PairOfStrings::right
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/Filter/ **String.h**

3.4 Dokumentacja struktury PrefixTree

```
#include <PrefixTree.h>
```

Atrybuty publiczne

- struct **PrefixTree** * **children** [**CHAR_COMBINATIONS**]
  Dzieci węzła.
- **String prefix**
  Zawartość (część prefixu) węzła.
- bool **isLeaf**
  Czy aktualny węzeł jest liściem drzewa?

3.4.1 Opis szczegółowy

Węzeł drzewa prefixowego.

3.4.2 Dokumentacja atrybutów składowych

3.4.2.1 children

```
struct PrefixTree* PrefixTree::children[ CHAR_COMBINATIONS]
```

Dzieci węzła.

3.4.2.2 isLeaf

```
bool PrefixTree::isLeaf
```

Czy aktualny węzeł jest liściem drzewa?

3.4.2.3 prefix

```
String PrefixTree::prefix
```

Zawartość (część prefixu) węzła.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/Filter/ **PrefixTree.h**

3.5 Dokumentacja struktury String

```
#include <String.h>
```

Atrybuty publiczne

- char * **data**
  Wskaźnik na początek znaków w napisie.
- size_t **len**
  Długość napis.
- size_t **capacity**
  Długość zarezerwowanej pamięci dla napisu (zawsze >= len).

3.5.1 Opis szczegółowy

Struktura do zarządzania napisami

3.5.2 Dokumentacja atrybutów składowych

3.5.2.1 capacity

```
size_t String::capacity
```

Długość zarezerwowanej pamięci dla napisu (zawsze >= len).

3.5.2.2 data

```
char* String::data
```

Wskaźnik na początek znaków w napisie.

3.5.2.3 len

```
size_t String::len
```

Długość napis.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/Filter/ **String.h**

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku include/Filter/App.h

```
#include <Filter/ParseArgs.h>
#include <stdbool.h>
```

Funkcje

- bool **run** (**LaunchConfig** ∗launchCfg_)

4.1.1 Dokumentacja funkcji

4.1.1.1 run()

```
bool run (
              LaunchConfig * launchCfg_ )
```

Uruchamia aplikację z odpowiednim ustawieniem.

Parametry

*launch↩ Cfg_*	ustawienie aplikacji

Zwraca

true jeśli wszystko przebiegło pomyślnie, w przeciwnym wypadku false

4.2 Dokumentacja pliku include/Filter/BufferedFileReader.h

```
#include <Filter/String.h>
#include <stdio.h>
#include <stdbool.h>
```

Komponenty

- struct **BufferedFileReader**

Definicje

- #define **BFR_BUF_SIZE** 1024 ∗ 1024

  *Rozmiar bloku danych wczytywanego przez **BufferedFileReader** (str. 5).*

Definicje typów

- typedef struct **BufferedFileReader** **BufferedFileReader**

Funkcje

- **BufferedFileReader** ∗ **bfrOpen** (const char ∗path_)
- **String** **bfrReadUntilWs** (**BufferedFileReader** ∗reader_)
- void **bfrClose** (**BufferedFileReader** ∗reader_)

4.2.1 Dokumentacja definicji

4.2.1.1 BFR_BUF_SIZE

```
#define BFR_BUF_SIZE 1024 * 1024
```

Rozmiar bloku danych wczytywanego przez **BufferedFileReader** (str. 5).

4.2.2 Dokumentacja definicji typów

4.2.2.1 BufferedFileReader

```
typedef struct BufferedFileReader BufferedFileReader
```

Buforowany wczytywacz (reader) danych z pliku. Wczytuje dane z pliku porcjami w celu zwiększenia wydajności.

4.2.3 Dokumentacja funkcji

4.2.3.1 bfrClose()

```
void bfrClose (
            BufferedFileReader * reader_ )
```

Usuwa z pamięci reader oraz zamyka plik.

Parametry

*reader_*↩	wskaźnik na reader

4.2.3.2 bfrOpen()

```
BufferedFileReader* bfrOpen (
          const char * path_ )
```

Otwiera buforowany reader wraz z danym plikiem.

Parametry

*path_*↩	ścieżka do pliku

Zwraca

wskaźnik na dynamicznie stworzony reader.

4.2.3.3 bfrReadUntilWs()

```
String bfrReadUntilWs (
            BufferedFileReader * reader_ )
```

Wczytuje dane do napotkania białego znaku.

Parametry

reader↩ _	wskaźnik na reader

Zwraca

Ciąg wczytanych znaków/danych, lub pusty **String** (str. 9), jeśli się nie udało wczytać.

4.3 Dokumentacja pliku include/Filter/Filter.h

```
#include <stdbool.h>
```

Definicje typów

- typedef struct **PrefixTree** **PrefixTree**

Funkcje

- bool **filterInputFile** (char const ∗inFilePath_, char const ∗outFilePath_, **PrefixTree** const ∗prefixTree_)

4.3.1 Dokumentacja definicji typów

4.3.1.1 PrefixTree

```
typedef struct PrefixTree PrefixTree
```

4.3.2 Dokumentacja funkcji

4.3.2.1 filterInputFile()

```
bool filterInputFile (
            char const * inFilePath_,
            char const * outFilePath_,
             PrefixTree const * prefixTree_ )
```

Filtruje plik źródłowy po przedrostkach zawartych w drzewie przedrostkowym. Zapisuje przefiltrowane dane do osobnego pliku.

Parametry

*inFilePath_*	ścieżka do pliku wejściowego
*outFilePath_*	ścieżka do pliku wyjściowego
*prefixTree_*	wskaźnik na korzeń drzewa przedrostkowego

4.4 Dokumentacja pliku include/Filter/ParseArgs.h

```
#include <stdio.h>
#include <stdbool.h>
```

Komponenty

- struct **LaunchConfig**

Definicje typów

- typedef struct **LaunchConfig** **LaunchConfig**

Funkcje

- bool **parseArgs** (int argc_, char *argv_[], **LaunchConfig** *config_)

4.4.1 Dokumentacja definicji typów

4.4.1.1 LaunchConfig

```
typedef struct LaunchConfig LaunchConfig
```

Informacje o konfiguracji uruchomieniowej programu.

4.4.2 Dokumentacja funkcji

4.4.2.1 parseArgs()

```
bool parseArgs (
            int argc_,
            char * argv_[],
             LaunchConfig * config_ )
```

Wypełnia ustawienie programu na podstawie argumentów.

Parametry

argc↩_	ilość argumentów
argv↩_	zawartość argumentów
config↩_	wskaźnik na ustawienie programu, do wypełnienia.

4.5 Dokumentacja pliku include/Filter/PrefixTree.h

```
#include <Filter/String.h>
#include <stdio.h>
#include <stdbool.h>
```

Komponenty

- struct **PrefixTree**

Definicje

- #define **CHAR_COMBINATIONS** 256

Definicje typów

- typedef struct **PrefixTree** **PrefixTree**

Funkcje

- **PrefixTree** ∗ **buildPrefixTree** (const char ∗prefixFilePath_)
- void **destroyPrefixTree** (**PrefixTree** ∗prefixTree_)
- bool **prefixFilter** (**PrefixTree** const ∗root_, **String** str_)

4.5.1 Dokumentacja definicji

4.5.1.1 CHAR_COMBINATIONS

```
#define CHAR_COMBINATIONS 256
```

4.5.2 Dokumentacja definicji typów

4.5.2.1 PrefixTree

```
typedef struct  PrefixTree  PrefixTree
```

Węzeł drzewa prefixowego.

4.5.3 Dokumentacja funkcji

4.5.3.1 buildPrefixTree()

```
PrefixTree* buildPrefixTree (
            const char * prefixFilePath_ )
```

Buduje drzewo przedrostkowe z wyrazów wczytanych z pliku.

Parametry

*prefixFile↩ Path_*	ścieżka do pliku z wyrazami

Zwraca

Stworzone dynamicznie drzewo.

4.5.3.2 destroyPrefixTree()

```
void destroyPrefixTree (
             PrefixTree * prefixTree_ )
```

Zwalnia z pamięci zawartość drzewa. Nie zwalnia korzenia!

Parametry

*prefix↩ Tree_*	korzeń drzewa przedrostkowego.

4.5.3.3 prefixFilter()

```
bool prefixFilter (
              PrefixTree const * root_,
              String str_ )
```

Sprawdza, czy wyraz powinien zostać zaakceptowany przez filtr przedrostkowy.

Parametry

*root_*	korzeń drzewa przedrostkowego
*str_*	wyraz do sprawdzenia

Zwraca

true jeśli wyraz posiada przedrostek znajdujący się w drzewie, w przeciwnym wypadku false.

4.6 Dokumentacja pliku include/Filter/String.h

```
#include <stdlib.h>
#include <inttypes.h>
```

Komponenty

- struct **String**
- struct **PairOfStrings**

Definicje typów

- typedef struct **String** **String**
- typedef struct **PairOfStrings** **PairOfStrings**

Funkcje

- **String** **makeString** ()
- **String** **makeStringWith** (char const ∗bytes_, size_t len_)
- **String** **wrapWithString** (char ∗bytes_, size_t len_)
- void **destroyString** (**String** ∗str_)
- int64_t **findWsInString** (**String** const ∗str_, size_t startPos_)
- int64_t **findNonWsInString** (**String** const ∗str_, size_t startPos_)
- **PairOfStrings** **splitString** (**String** ∗src_, size_t where_)
- void **appendString** (**String** ∗str_, **String** const ∗toAppend_)
- void **appendStringRaw** (**String** ∗str_, char const ∗bytes_, size_t len_)
- void **reserveStringCapacity** (**String** ∗str_, size_t newCapacity_)
- void **shrinkStringToFit** (**String** ∗str_)

4.6.1 Dokumentacja definicji typów

4.6.1.1 PairOfStrings

```
typedef struct  PairOfStrings  PairOfStrings
```

Para napisów.

4.6.1.2 String

```
typedef struct  String  String
```

Struktura do zarządzania napisami

4.6.2 Dokumentacja funkcji

4.6.2.1 appendString()

```
void appendString (
              String * str_,
              String const * toAppend_ )
```

Dopisuje napis na koniec drugiego.

Parametry

*str_*	napis, do którego zostanie dopisany inny
to↩ Append↩ _	dopisywany napis

4.6.2.2 appendStringRaw()

```
void appendStringRaw (
              String * str_,
             char const * bytes_,
             size_t len_ )
```

Dopisuje napis na koniec drugiego (wersja dla czystego C-stringa)

Parametry

*str_*	napis, do którego zostanie dopisany inny
*bytes_*	początek napisu do dopisania
*len_*	ilość znaków do dopisania

4.6.2.3 destroyString()

```
void destroyString (
              String * str_ )
```

Usuwa zawartość istniejącego obiektu **String** (str. 9). Nie usuwa przekazanego obiektu z pamięci!

Parametry

*str_*	**String** (str. 9), z którego zostanie zwolniona pamięć

4.6.2.4 findNonWsInString()

```
int64_t findNonWsInString (
              String const * str_,
             size_t startPos_ )
```

Znajduje indeks pierwszego elementu nie będącego białym znakiem w napisie.

Parametry

*str_*	rozpatrywany napis
*startPos_*	pozycja, od której rozpoczynamy wyszukiwanie

Zwraca

pozycja znalezionego znaku, lub -1 gdy nie znaleziono.

4.6.2.5 findWsInString()

```
int64_t findWsInString (
              String const * str_,
             size_t startPos_ )
```

Znajduje indeks pierwszego białego znaku w napisie.

Parametry

*str_*	rozpatrywany napis
*start↩ Pos_*	pozycja, od której rozpoczynamy wyszukiwanie

Zwraca

pozycja znalezionego białego znaku, lub -1 gdy nie znaleziono.

4.6.2.6 makeString()

```
String makeString ( )
```

Tworzy pusty **String** (str. 9).

Zwraca

pusty **String** (str. 9).

4.6.2.7 makeStringWith()

```
String makeStringWith (
            char const * bytes_,
            size_t len_ )
```

Tworzy **String** (str. 9) kopiując inny istniejący.

Parametry

bytes↩ _	źródło do kopiowania danych
*len_*	ilość bajtów do skopiowania

Zwraca

nowo utworzony **String** (str. 9).

4.6.2.8 reserveStringCapacity()

```
void reserveStringCapacity (
             String * str_,
            size_t newCapacity_ )
```

Rezerwuje pamięć wewnątrz Stringa. Nigdy nie zmniejsza zaalokowanej pamięci.

Parametry

*str_*	napis, któremu rezerwujemy pamięć
*new↩ Capacity_*	nowa pojemność pamięci.

4.6.2.9 shrinkStringToFit()

```
void shrinkStringToFit (
              String * str_ )
```

Ucina zarezerwowana pamięć do ilości znaków faktycznie używanych przez Stringa.

Parametry

str↩ _	modyfikowany napis

4.6.2.10 splitString()

```
PairOfStrings splitString (
              String * src_,
             size_t where_ )
```

Rozbija napis na dwa osobne napisy w następujący sposób: lewy zawiera indeksy [0, where_), prawy zawiera indeksy [where_, length). Nie zwalnia rozbijanego zapisu z pamięci!

Parametry

*src_*	rozbijany napis
where↩ _	pozycja do rozbicia napisu

Zwraca

struktura zawierająca lewą i prawą część napisu

4.6.2.11 wrapWithString()

```
String wrapWithString (
            char * bytes_,
            size_t len_ )
```

Tworzy **String** (str. 9) opakowując istniejące dane.

Parametry

*bytes_*	źródło danych
*len_*	ilość bajtów

Zwraca

nowo utworzony **String** (str. 9) (opakowujący istniejące dane).

4.7 Dokumentacja pliku src/App.c

```
#include <Filter/App.h>
#include <Filter/PrefixTree.h>
#include <Filter/Filter.h>
#include <stdlib.h>
#include <assert.h>
```

Funkcje

- bool **run** (**LaunchConfig** *launchCfg_)

4.7.1 Dokumentacja funkcji

4.7.1.1 run()

```
bool run (
            LaunchConfig * launchCfg_ )
```

Uruchamia aplikację z odpowiednim ustawieniem.

Parametry

*launchCfg_*	ustawienie aplikacji

Zwraca

true jeśli wszystko przebiegło pomyślnie, w przeciwnym wypadku false

4.8 Dokumentacja pliku src/BufferedFileReader.c

```
#include <Filter/BufferedFileReader.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>
```

Funkcje

- bool **bfrReadNext** (**BufferedFileReader** ∗reader_)
- void **bfrConsume** (**BufferedFileReader** ∗reader_, size_t count_)
- **BufferedFileReader** ∗ **bfrOpen** (const char ∗path_)
- **String** **bfrReadUntilWs** (**BufferedFileReader** ∗reader_)
- void **bfrClose** (**BufferedFileReader** ∗reader_)

4.8.1 Dokumentacja funkcji

4.8.1.1 bfrClose()

```
void bfrClose (
              BufferedFileReader * reader_ )
```

Usuwa z pamięci reader oraz zamyka plik.

Parametry

*reader_*	wskaźnik na reader

4.8.1.2 bfrConsume()

```
void bfrConsume (
              BufferedFileReader * reader_,
             size_t count_ )
```

Usuwa z bufora określoną liczbę znaków.

Parametry

*reader_*	wskaźnik na reader
*count_*	liczba znaków

4.8.1.3 bfrOpen()

```
BufferedFileReader* bfrOpen (
            const char * path_ )
```

Otwiera buforowany reader wraz z danym plikiem.

Parametry

*path_*	ścieżka do pliku

Zwraca

wskaźnik na dynamicznie stworzony reader.

4.8.1.4 bfrReadNext()

```
bool bfrReadNext (
              BufferedFileReader * reader_ )
```

Wczytuje następny blok danych. Rozmiar bloku jest określany makrem

Zobacz również

BFR_BUF_SIZE (str. 12)

Parametry

*reader_*	wskaźnik na reader

Zwraca

true jeśli udało się wczytać BFR_BUF_SIZE bajtów, w innym wypadku false.

4.8.1.5 bfrReadUntilWs()

```
String bfrReadUntilWs (
              BufferedFileReader * reader_ )
```

Wczytuje dane do napotkania białego znaku.

Parametry

reader↩ _	wskaźnik na reader

Zwraca

Ciąg wczytanych znaków/danych, lub pusty **String** (str. 9), jeśli się nie udało wczytać.

4.9 Dokumentacja pliku src/Filter.c

```
#include <Filter/Filter.h>
#include <Filter/PrefixTree.h>
#include <Filter/BufferedFileReader.h>
#include <string.h>
#include <assert.h>
```

Definicje

- #define **OUTPUT_BUF_SIZE** 1024 ∗ 1024

Funkcje

- bool **filterInputFile** (char const ∗inFilePath_, char const ∗outFilePath_, **PrefixTree** const ∗prefixTree_)

4.9.1 Dokumentacja definicji

4.9.1.1 OUTPUT_BUF_SIZE

```
#define OUTPUT_BUF_SIZE 1024 * 1024
```

Rozmiar bufora dla pliku wyjściowego

4.9.2 Dokumentacja funkcji

4.9.2.1 filterInputFile()

```
bool filterInputFile (
            char const * inFilePath_,
            char const * outFilePath_,
             PrefixTree const * prefixTree_ )
```

Filtruje plik źródłowy po przedrostkach zawartych w drzewie przedrostkowym. Zapisuje przefiltrowane dane do osobnego pliku.

Parametry

*inFilePath_*	ścieżka do pliku wejściowego
*outFilePath_*	ścieżka do pliku wyjściowego
*prefixTree_*	wskaźnik na korzeń drzewa przedrostkowego

4.10 Dokumentacja pliku src/Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <Filter/ParseArgs.h>
#include <Filter/App.h>
```

Funkcje

- int **main** (int argc, char ∗argv[ ])

4.10.1 Dokumentacja funkcji

4.10.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Funkcja główna programu. Parsuje argumenty i uruchamia właściwą logikę programu.

Parametry

argc	ilość argumentów
argv	zawartość argumentów

4.11 Dokumentacja pliku src/ParseArgs.c

```
#include <Filter/ParseArgs.h>
#include <assert.h>
#include <string.h>
```

Funkcje

- bool **parseArgs** (int argc, char ∗argv[ ], **LaunchConfig** ∗config_)

4.11.1 Dokumentacja funkcji

4.11.1.1 parseArgs()

```
bool parseArgs (
            int argc_,
            char * argv_[],
             LaunchConfig * config_ )
```

Wypełnia ustawienie programu na podstawie argumentów.

Parametry

argc↩ _	ilość argumentów
argv↩ _	zawartość argumentów
config↩ _	wskaźnik na ustawienie programu, do wypełnienia.

4.12 Dokumentacja pliku src/PrefixTree.c

```
#include <Filter/PrefixTree.h>
#include <Filter/BufferedFileReader.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
```

Funkcje

- **PrefixTree** ∗ **makePrefixTree** ()
- void **insertPrefixIntoTree** (**PrefixTree** ∗root_, **String** prefix_)
- bool **shouldContinueDown** (**PrefixTree** ∗node_, **String** prefix_, size_t ∗startCharacter_)
- void **initPrefixTreeNode** (**PrefixTree** ∗node_)
- void **initPrefixTreeNodeWith** (**PrefixTree** ∗node_, **String** str_)
- **PrefixTree** ∗ **buildPrefixTree** (const char ∗prefixFilePath)
- void **destroyPrefixTree** (**PrefixTree** ∗prefixTree_)
- bool **prefixFilter** (**PrefixTree** const ∗root_, **String** str_)

4.12.1 Dokumentacja funkcji

4.12.1.1 buildPrefixTree()

```
PrefixTree* buildPrefixTree (
            const char * prefixFilePath_ )
```

Buduje drzewo przedrostkowe z wyrazów wczytanych z pliku.

Parametry

*prefixFile↩ Path_*	ścieżka do pliku z wyrazami

Zwraca

Stworzone dynamicznie drzewo.

4.12.1.2 destroyPrefixTree()

```
void destroyPrefixTree (
             PrefixTree * prefixTree_ )
```

Zwalnia z pamięci zawartość drzewa. Nie zwalnia korzenia!

Parametry

*prefix↩ Tree_*	korzeń drzewa przedrostkowego.

4.12.1.3 initPrefixTreeNode()

```
void initPrefixTreeNode (
             PrefixTree * node_ )
```

4.12.1.4 initPrefixTreeNodeWith()

```
void initPrefixTreeNodeWith (
             PrefixTree * node_,
             String str_ )
```

4.12.1.5 insertPrefixIntoTree()

```
void insertPrefixIntoTree (
              PrefixTree * root_,
              String prefix_ )
```

Dodaje przedrostek do drzewa. Zarządza czasem życia `prefix_`!

Parametry

*root_*	korzeń drzewa przedrostkowego
*prefix_*	prefix do dodania

4.12.1.6 makePrefixTree()

```
PrefixTree * makePrefixTree ( )
```

Dynamicznie tworzy korzeń drzewa prefixowego.

Zwraca

Korzeń stworzonego drzewa.

4.12.1.7 prefixFilter()

```
bool prefixFilter (
              PrefixTree const * root_,
              String str_ )
```

Sprawdza, czy wyraz powinien zostać zaakceptowany przez filtr przedrostkowy.

Parametry

*root_*	korzeń drzewa przedrostkowego
*str_*	wyraz do sprawdzenia

Zwraca

true jeśli wyraz posiada przedrostek znajdujący się w drzewie, w przeciwnym wypadku false.

4.12.1.8 shouldContinueDown()

```
bool shouldContinueDown (
              PrefixTree * node_,
              String prefix_,
             size_t * startCharacter_ )
```

Sprawdza czy dodając prefix powinniśmy iść dalej w dół drzewa, czy rozbić aktualny węzeł i zrobić gałąź.

Parametry

*node_*	aktualnie rozpatrywany węzeł
*prefix_*	dodawany prefix
*start↩ Character_*	aktualna pozycja do porównywania prefixu

4.13 Dokumentacja pliku src/String.c

```
#include <Filter/String.h>
#include <stdlib.h>
#include <ctype.h>
#include <assert.h>
```

Funkcje

- **String makeString** ()
- **String makeStringWith** (char const ∗bytes_, size_t len_)
- **String wrapWithString** (char ∗bytes_, size_t len_)
- void **destroyString** (**String** ∗str_)
- int64_t **findWsInString** (**String** const ∗str_, size_t startPos_)
- int64_t **findNonWsInString** (**String** const ∗str_, size_t startPos_)
- **PairOfStrings splitString** (**String** ∗src_, size_t where_)
- void **appendString** (**String** ∗str_, **String** const ∗toAppend_)
- void **appendStringRaw** (**String** ∗str_, char const ∗bytes_, size_t len_)
- void **reserveStringCapacity** (**String** ∗str_, size_t newCapacity_)
- void **shrinkStringToFit** (**String** ∗str_)

4.13.1 Dokumentacja funkcji

4.13.1.1 appendString()

```
void appendString (
              String * str_,
              String const * toAppend_ )
```

Dopisuje napis na koniec drugiego.

Parametry

*str_*	napis, do którego zostanie dopisany inny
to↩ Append↩ _	dopisywany napis

4.13.1.2 appendStringRaw()

```
void appendStringRaw (
              String * str_,
             char const * bytes_,
             size_t len_ )
```

Dopisuje napis na koniec drugiego (wersja dla czystego C-stringa)

Parametry

*str_*	napis, do którego zostanie dopisany inny
bytes↩ _	początek napisu do dopisania
*len_*	ilość znaków do dopisania

4.13.1.3 destroyString()

```
void destroyString (
              String * str_ )
```

Usuwa zawartość istniejącego obiektu **String** (str. 9). Nie usuwa przekazanego obiektu z pamięci!

Parametry

str↩ _	**String** (str. 9), z którego zostanie zwolniona pamięć

4.13.1.4 findNonWsInString()

```
int64_t findNonWsInString (
              String const * str_,
             size_t startPos_ )
```

Znajduje indeks pierwszego elementu nie będącego białym znakiem w napisie.

Parametry

*str_*	rozpatrywany napis
*start↩ Pos_*	pozycja, od której rozpoczynamy wyszukiwanie

Zwraca

pozycja znalezionego znaku, lub -1 gdy nie znaleziono.

4.13.1.5 findWsInString()

```
int64_t findWsInString (
              String const * str_,
             size_t startPos_ )
```

Znajduje indeks pierwszego białego znaku w napisie.

Parametry

*str_*	rozpatrywany napis
*start↩ Pos_*	pozycja, od której rozpoczynamy wyszukiwanie

Zwraca

pozycja znalezionego białego znaku, lub -1 gdy nie znaleziono.

4.13.1.6 makeString()

```
String makeString ( )
```

Tworzy pusty **String** (str. 9).

Zwraca

pusty **String** (str. 9).

4.13.1.7 makeStringWith()

```
String makeStringWith (
            char const * bytes_,
            size_t len_ )
```

Tworzy **String** (str. 9) kopiując inny istniejący.

Parametry

*bytes_*	źródło do kopiowania danych
*len_*	ilość bajtów do skopiowania

Zwraca

nowo utworzony **String** (str. 9).

4.13.1.8 reserveStringCapacity()

```
void reserveStringCapacity (
              String * str_,
             size_t newCapacity_ )
```

Rezerwuje pamięć wewnątrz Stringa. Nigdy nie zmniejsza zaalokowanej pamięci.

Parametry

*str_*	napis, któremu rezerwujemy pamięć
*newCapacity_*	nowa pojemność pamięci.

4.13.1.9 shrinkStringToFit()

```
void shrinkStringToFit (
              String * str_ )
```

Ucina zarezerwowana pamięć do ilości znaków faktycznie używanych przez Stringa.

Parametry

*str_*	modyfikowany napis

4.13.1.10 splitString()

```
PairOfStrings splitString (
              String * src_,
             size_t where_ )
```

Rozbija napis na dwa osobne napisy w następujący sposób: lewy zawiera indeksy [0, where_), prawy zawiera indeksy [where_, length). Nie zwalnia rozbijanego zapisu z pamięci!

Parametry

*src_*	rozbijany napis
*where_*	pozycja do rozbicia napisu

Zwraca

struktura zawierająca lewą i prawą część napisu

4.13.1.11 wrapWithString()

```
String wrapWithString (
            char * bytes_,
            size_t len_ )
```

Tworzy **String** (str. 9) opakowując istniejące dane.

Parametry

*bytes_*	źródło danych
*len_*	ilość bajtów

Zwraca

nowo utworzony **String** (str. 9) (opakowujący istniejące dane).

Indeks