A PROGRAMOZÁS ALAPJAI 1
(BMEVIEEAA00, 2024/25/1)
NAGYHÁZI FELADAT

Shanon-Fano kódoló és dekódoló program

készítette:

Ferencz Péter

(RFG7SN)

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Mérnökinformatikus Bsc

2024 Október

1. fejezet

Specifikáció

1.1. A program célja

A program célja tetszőleges adat tömörítése majd ezek kitömörítése információvesztés nélkül. Ennek megvalósítására a Shanon-Fano tömörítő algoritmust [1] [2] alkalmazza.

1.2. Felhasználói interakció

A felhasználó két üzemmódot választhat ki a program futtatásakor: kódolás vagy dekódolás. Ezeket az első parancssori argumentumban a 'kodol' és 'dekodol' kulcsszavakkal tudja kiválasztani.

1.2.1. kódolás (kodol)

Kódoló üzemmódban a bemenetet (lásd Bemenet) a Shanon-Fano kódoló algoritmust alkalmazva írja a kimenetre (lásd Kimenet) a kódolt adatot.

```
program kodol --bemenet <fájl> --kimenet <fájl>
```

1.2.2. dekódolás (dekodol)

Dekódoló üzemmódban a bemenetet (lásd Bemenet) a Shanon-Fano dekódoló algoritmust alkalmazva írja a kimenetre (lásd Kimenet) a dekódolt adatot.

```
program dekodol --bemenet <fájl> --kimenet <fájl>
```

[1] C. E. Shannon, „A Mathematical Theory of Communication", 1948
[2] Robert M. Fano, „The Transmittion of Information", 1949

1.3. A program által elfogadott kapcsolók

A program futása során tetszőleges futtatást befolyásoló kapcsolókat (flageket) beállíthatunk. Ezek sorrendje tetszőlegesen választható.

1.3.1. Bemenet

Parancssori megnevezés: `--bemenet <forrásfájl>`
Opcionális paraméter.
Ha nincs megadva, de a program egy figyelmeztető üzenet kíséretében folytatja a lefutást.
A fájl méretétől és tartalmától független a program lefutása.
Az azt követő paraméter megadja a forrásfájl elérési útvonalát. Ha nincs megadva, stdin-ról kér be új sorral lezárt szöveget.

1.3.2. Kimenet

Parancssori megnevezés: `--kimenet <célfájl>`
Opcionális paraméter.
Ha nincs megadva, de a program egy figyelmeztető üzenet kíséretében folytatja a lefutást.
A fájl méretétől és tartalmától független a program lefutása.
Az azt követő paraméter megadja a célfájl elérési útvonalát. Ha nincs megadva, stdout-ra írja ki a program a program kimenetét.

1.3.3. Kódtábla

Parancssori megnevezés: `--kodtabla`
Opcionális paraméter.
Azt szabályozza, hogy a kódtáblát kiírja-e a program a standard kimenetre.

1.3.4. Statisztika

Parancssori megnevezés: `--statisztika`
Opcionális paraméter.
Azt határozza meg, hogy a program kiírjon-e további számitásokat a program hatékonyságára vonatkozólag.
Az alábbi számítások történnek kiírásra:

- Tömörítés mértéke: bemenet mérete a tömörített adat méretéhez képest
- Kódtábal mérete: Egymástól eltérő kódok száma
- Kódok mérete: legrövidebb kód, leghosszabb kód, kódok átlagos mérete
- Fa mérete: A generált fa mérete

1.3.5. Segítség

Parancssori megnevezés: `--help`
Opcionális paraméter.
A felhasználót tájékoztatja a program helyes használatáról. Ha ez a kapcsoló meg van adva, akkor a program nem ellenőrzi a többi kötelező kapcsoló jelenlétét, kiírja a szöveget majd kódolás / dekódolás nélkül befejezi a futást.
Az alábbi szöveg íródik ki:

```
program [üzemmód] <...kapcsolók...>
Üzemmód: kodol, dekodol
Kapcsolók:
--bemenet <forrásfájl>: Bemeneti fájl (ha üres akkor stdin)
--kimenet <célfájl>: Bemeneti fájl (ha üres akkor stdout)
--kodtabla <fájl>: A kódtábla fájl (kötelező)
--statisztika: A tömörítés hatékonyságát értékelő statisztika (opcionális)
--help: Ezt az üzenetet írja ki (opcionális)
```

1.4. A program kimenete

Sikeres futtatás esetén a program a A program által elfogadott kapcsolók pontban meghatározott viselkedés szerint működik. Sikertelen futtatás esetén a konzolra kiíródik a probléma és egy nem nullás kilépési kóddal a program megáll.
A fájl ami generálódik a következőképpen épül fel: Kódtábla karaktereinek száma: Hány darab karaktert és annak kódolását tartalmaz a kódtábla. Lehetséges értékei: 0-255 –> 1-256
Illeszkedés hossza: A fájl végén hány darab 0 bit van a 8 bites fájlmentés kielégítéséhez.
Kódtábla, melynek minden eleme az alábbiakból épül fel:

- Karakter: nyolc bit, melyet tömörítünk
- a karaktert reprezentáló kód hossza 8 biten
- a kód, mely nullás és eggyesek sorozata

Kódolt adat

<table>
<tr><td rowspan="2">Kódolt karakterek hossza
(1-256)</td><td rowspan="2">Illeszkedés
hossza
(0-7)</td><td colspan="4">Kódtábla</td><td rowspan="2">Kódolt adat</td><td rowspan="2">Illeszkedés
(0)</td></tr>
<tr><td>karakter
ASCII</td><td>karakterkód
hossza</td><td>kód</td><td>. . . . .</td></tr>
<tr><td></td><td></td><td>8 bit</td><td>n = 8 bit</td><td>n bit</td><td></td><td></td><td></td></tr>
<tr><td>l = 8 bit</td><td>i = 3 bit</td><td colspan="4">legalább l * (8 + 8 + 1) bit</td><td>legalább l bit</td><td>i bit</td></tr>
</table>

2. fejezet

Adatszerkezet-mutató

2.1. Adatszerkezetek

Az összes adatszerkezet listája rövid leírásokkal:

3. fejezet

Fájlmutató

3.1. Fájllista

Az összes fájl listája rövid leírásokkal:

4. fejezet

Adatszerkezetek dokumentációja

4.1. Bits struktúrareferencia

Tetszőleges hosszú bitsorozat eltárolására alkalmas struktúra.

`#include <codeword.h>`

Adatmezők

- long long unsigned int b

  A tárolt szám A bitek jobbról balra értelmezendőek.
- size_t length

  A tárolt bitsorozat hossza.

4.1.1. Részletes leírás

Tetszőleges hosszú bitsorozat eltárolására alkalmas struktúra.

4.1.2. Adatmezők dokumentációja

4.1.2.1. b

`long long unsigned int Bits::b`

A tárolt szám A bitek jobbról balra értelmezendőek.

4.1.2.2. length

```
size_t Bits::length
```

A tárolt bitsorozat hossza.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

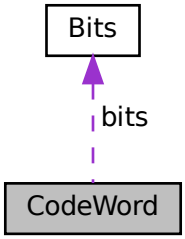
- lib/codeword.h

4.2. CodeWord struktúrareferencia

Karakter, és az azt kódoló bitsorozat.

```
#include <codeword.h>
```

A CodeWord osztály együttműködési diagramja:



Adatmezők

- uchar codeWord

  Egy byte, melyet a Shanon-Fano kódolás szerint kódolunk.
- Bits bits

  Egy bitsorozat, melyet a Shanon-Fano kódolás szerint a codeWord } kódolt változata.

4.2.1. Részletes leírás

Karakter, és az azt kódoló bitsorozat.

4.2.2. Adatmezők dokumentációja

4.2.2.1. bits

```
Bits CodeWord::bits
```

Egy bitsorozat, melyet a Shanon-Fano kódolás szerint a codeWord } kódolt változata.

4.2.2.2. codeWord

```
uchar CodeWord::codeWord
```

Egy byte, melyet a Shanon-Fano kódolás szerint kódolunk.

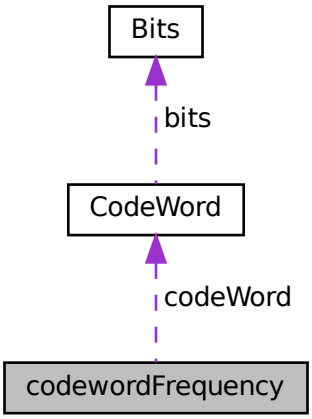
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- lib/codeword.h

4.3. codewordFrequency struktúrareferencia

```
#include <encoder.h>
```

A codewordFrequency osztály együttműködési diagramja:



Adatmezők

- float freq
- CodeWord codeWord

4.3.1. Adatmezők dokumentációja

4.3.1.1. codeWord

```
CodeWord codewordFrequency::codeWord
```

4.3.1.2. freq

```
float codewordFrequency::freq
```

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- lib/encoder.h

4.4. commandLineArguments struktúrareferencia

A program parancssori argumentumait rendező struktúra.

```
#include <main.h>
```

Adatmezők

- FILE * infile

  A –bemenet kapcsoló által megadott stream.
- FILE * outfile

  A –kimenet kapcsoló által megadott stream.
- bool displayTable

  Megajda, hogy a program kiírja-e a kódtáblát.
- bool displayStatistics

  Megajda, hogy a program kiírjon-e további számitásokat a program hatékonyságára vonatkozólag.

4.4.1. Részletes leírás

A program parancssori argumentumait rendező struktúra.

4.4.2. Adatmezők dokumentációja

4.4.2.1. displayStatistics

```
bool commandLineArguments::displayStatistics
```

Megajda, hogy a program kiírjon-e további számitásokat a program hatékonyságára vonatkozólag.

4.4.2.2. displayTable

```
bool commandLineArguments::displayTable
```

Megajda, hogy a program kiírja-e a kódtáblát.

4.4.2.3. infile

```
FILE* commandLineArguments::infile
```

A –bemenet kapcsoló által megadott stream.

4.4.2.4. outfile

```
FILE* commandLineArguments::outfile
```

A –kimenet kapcsoló által megadott stream.

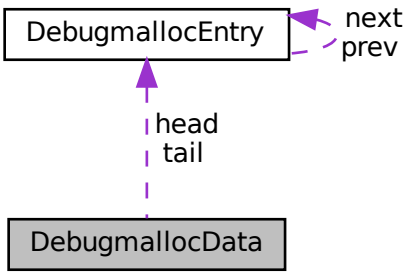
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- lib/main.h

4.5. DebugmallocData struktúrareferencia

```
#include <debugmalloc.h>
```

A DebugmallocData osztály együttműködési diagramja:



Adatmezők

- char logfile [256]
- long max_block_size
- long alloc_count
- long long alloc_bytes
- long all_alloc_count
- long long all_alloc_bytes
- DebugmallocEntry head [debugmalloc_tablesize]
- DebugmallocEntry tail [debugmalloc_tablesize]

4.5.1. Adatmezők dokumentációja

4.5.1.1. all_alloc_bytes

```
long long DebugmallocData::all_alloc_bytes
```

4.5.1.2. all_alloc_count

```
long DebugmallocData::all_alloc_count
```

4.5.1.3. alloc_bytes

```
long long DebugmallocData::alloc_bytes
```

4.5.1.4. alloc_count

```
long DebugmallocData::alloc_count
```

4.5.1.5. head

```
DebugmallocEntry DebugmallocData::head[debugmalloc_tablesize]
```

4.5.1.6. logfile

```
char DebugmallocData::logfile[256]
```

4.5.1.7. max_block_size

```
long DebugmallocData::max_block_size
```

4.5.1.8. tail

```
DebugmallocEntry DebugmallocData::tail[debugmalloc_tablesize]
```

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- lib/debug/debugmalloc.h

4.6. DebugmallocEntry struktúrareferencia

```
#include <debugmalloc.h>
```

A DebugmallocEntry osztály együttműködési diagramja:

Adatmezők

- void * real_mem
- void * user_mem
- size_t size
- char file [64]
- unsigned line
- char func [32]
- char expr [128]
- struct DebugmallocEntry * prev
- struct DebugmallocEntry * next

4.6.1. Adatmezők dokumentációja

4.6.1.1. expr

```
char DebugmallocEntry::expr[128]
```

4.6.1.2. file

```
char DebugmallocEntry::file[64]
```

4.6.1.3. func

```
char DebugmallocEntry::func[32]
```

4.6.1.4. line

```
unsigned DebugmallocEntry::line
```

4.6.1.5. next

```
struct DebugmallocEntry * DebugmallocEntry::next
```

4.6.1.6. prev

```
struct DebugmallocEntry* DebugmallocEntry::prev
```

4.6.1.7. real_mem

```
void* DebugmallocEntry::real_mem
```

4.6.1.8. size

```
size_t DebugmallocEntry::size
```

4.6.1.9. user_mem

```
void* DebugmallocEntry::user_mem
```

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- lib/debug/debugmalloc.h

4.7. InputFileBuffer struktúrareferencia

Struktúra, mely lehetővé teszi a bitenkénti olvasást egy fájlból.

```
#include <fileBuffer.h>
```

Adatmezők

- FILE * file
  A fájl, melyből olvasunk.
- uchar * currentBit
  Megadja, hogy az adott fájl olvasásánál hanyadik bitnél tartunk. Értéke 0 és 7 közötti.

4.7.1. Részletes leírás

Struktúra, mely lehetővé teszi a bitenkénti olvasást egy fájlból.

4.7.2. Adatmezők dokumentációja

4.7.2.1. currentBit

```
uchar* InputFileBuffer::currentBit
```

Megadja, hogy az adott fájl olvasásánál hanyadik bitnél tartunk. Értéke 0 és 7 közötti.

4.7.2.2. file

```
FILE* InputFileBuffer::file
```

A fájl, melyből olvasunk.

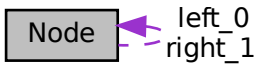
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- lib/fileBuffer.h

4.8. Node struktúrareferencia

```
#include <graph.h>
```

A Node osztály együttműködési diagramja:



Adatmezők

- char codeword
- struct Node * left_0
- struct Node * right_1

4.8.1. Adatmezők dokumentációja

4.8.1.1. codeword

```
char Node::codeword
```

4.8.1.2. left_0

```
struct Node* Node::left_0
```

4.8.1.3. right_1

```
struct Node* Node::right_1
```

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

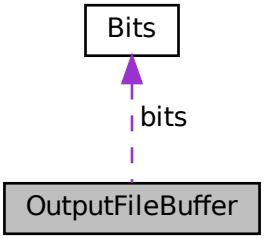
- lib/graph.h

4.9. OutputFileBuffer struktúrareferencia

Struktúra, mely lehetővé teszi a bitenkénti írást egy fájlba.

```
#include <fileBuffer.h>
```

Az OutputFileBuffer osztály együttműködési diagramja:



Adatmezők

- FILE * file

  A fájl, melybe írunk.
- Bits * bits

  A még nem a fájlba beírt bitek.

4.9.1. Részletes leírás

Struktúra, mely lehetővé teszi a bitenkénti írást egy fájlba.

4.9.2. Adatmezők dokumentációja

4.9.2.1. bits

```
Bits* OutputFileBuffer::bits
```

A még nem a fájlba beírt bitek.

4.9.2.2. file

```
FILE* OutputFileBuffer::file
```

A fájl, melybe írunk.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- lib/fileBuffer.h

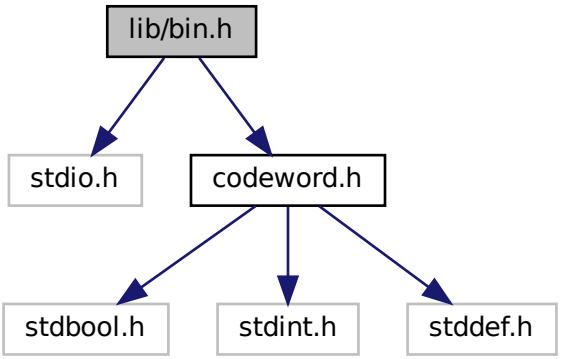
5. fejezet

Fájlok dokumentációja

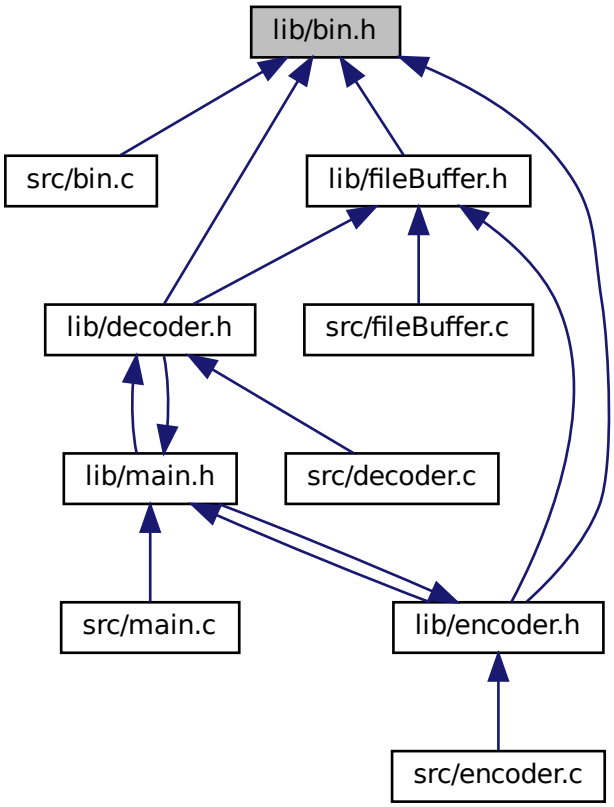
5.1. lib/bin.h fájlreferencia

```
#include <stdio.h>
#include "codeword.h"
```

A bin.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Függvények

- Bits getBitFromRight (Bits bits, int n)

  Adott bitsorozatnak megadja a jobról számított n *-edik bitjét.*
- void bits_pushBit (Bits ∗bits, Bits append)

  Egy bitsorozatot bővít egy másik bitsorozattal jobb oldalról.
- void print_bits (Bits bits)

  Kiír egy bitsorozatot ASCII 0 és 1 karakterekkel.

5.1.1. Függvények dokumentációja

5.1.1.1. bits_pushBit()

```
void bits_pushBit (
            Bits * bits,
            Bits append )
```

Egy bitsorozatot bővít egy másik bitsorozattal jobb oldalról.

Paraméterek

bits	a bővítendő bitsorozat
append	A hozzáfűzendő bitsorozat

5.1.1.2. getBitFromRight()

```
Bits getBitFromRight (
            Bits bits,
            int n )
```

Adott bitsorozatnak megadja a jobról számított `n` -edik bitjét.

Paraméterek

bits	A bitsorozat, melyből kiválasztjuk a bitet
n	Jobbról számítva hányadik bit

Visszatérési érték

A keresett bit

5.1.1.3. print_bits()

```
void print_bits (
            Bits bits )
```

Kiír egy bitsorozatot ASCII 0 és 1 karakterekkel.

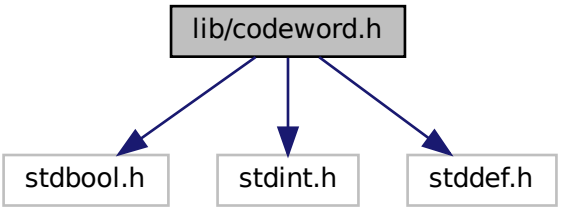
Paraméterek

bits	A kiírandó bitsorozat

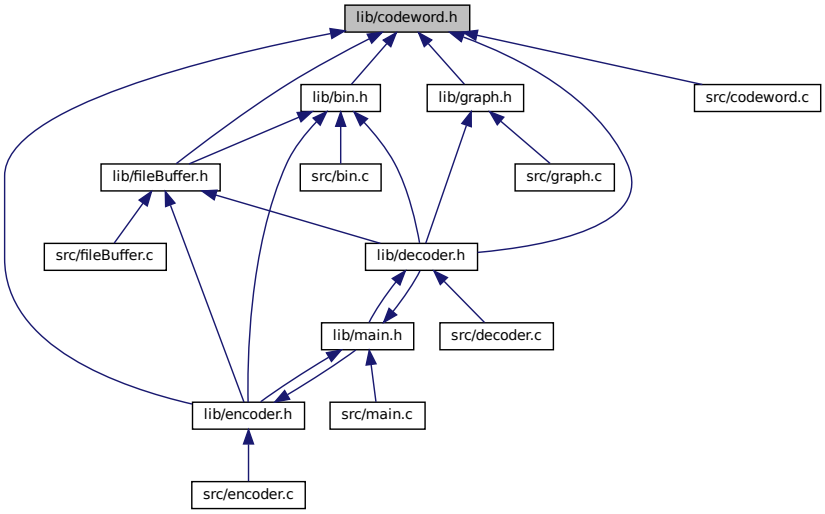
5.2. lib/codeword.h fájlreferencia

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

A codeword.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- struct Bits

  Tetszőleges hosszú bitsorozat eltárolására alkalmas struktúra.
- struct CodeWord

  Karakter, és az azt kódoló bitsorozat.

Makródefiníciók

- #define NULLBIT ((Bits){ .b = 0, .length = 0 })

  Hibás kimenetet jelentő bitsorozat, melynek hossza 0.

Típusdefiníciók

- typedef unsigned char uchar

  Előjel nélküli 8 bites karakter.

Függvények

- bool bits_equ (Bits b1, Bits b2)

  Összehasonlít két bitsorozatot.
- bool isNullbit (Bits b)

  Megmondja, hogy egy adott bitsorozat értelmes-e.

5.2.1. Makródefiníciók dokumentációja

5.2.1.1. NULLBIT

```
#define NULLBIT ((Bits){ .b = 0, .length = 0 })
```

Hibás kimenetet jelentő bitsorozat, melynek hossza 0.

5.2.2. Típusdefiníciók dokumentációja

5.2.2.1. uchar

```
typedef unsigned char uchar
```

Előjel nélküli 8 bites karakter.

5.2.3. Függvények dokumentációja

5.2.3.1. bits_equ()

```
bool bits_equ (
            Bits b1,
            Bits b2 )
```

Összehasonlít két bitsorozatot.

Paraméterek

b1	Az összehasonlítandó bitsorozat
b2	Az összehasonlítandó bitsorozat

Visszatérési érték

igaz, hogyha a bitsorozatok hossza és bitjei megegyeznek, különben hamis

5.2.3.2. isNullbit()

```
bool isNullbit (
            Bits b )
```

Megmondja, hogy egy adott bitsorozat értelmes-e.

Paraméterek

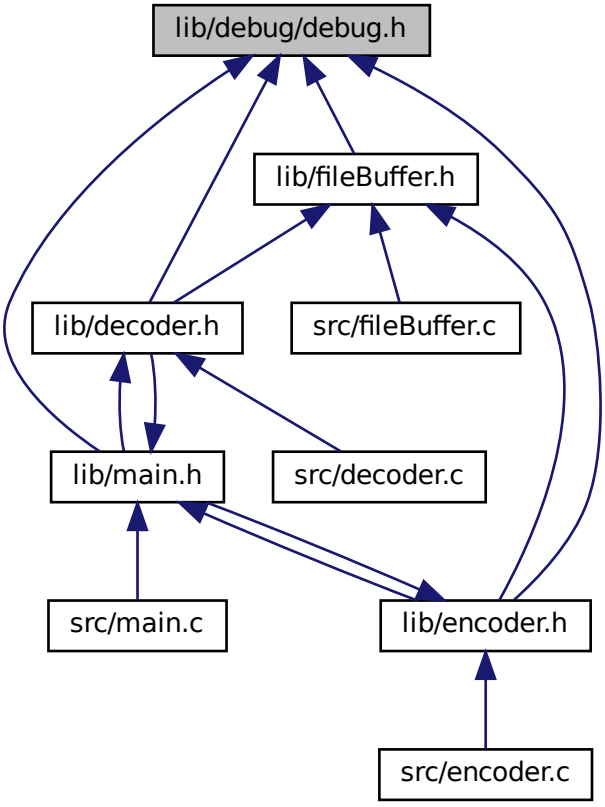
b	A vizsgálandó bitsorozat

Visszatérési érték

igaz, hogyha a bitsorozat hossza 0, különben hamis

5.3. lib/debug/debug.h fájlreferencia

Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Makródefiníciók

- #define PRINTDEBUG_MALLOCNULL() ;;

  Kiírja hogy egy memóriafoglalás sikertelen volt.
- #define PRINTDEBUG_FILEERR() ;;

  Kiírja, hogy a fájlművelet sikertelen volt.
- #define PRINTDEBUG_CORRUPTEDFILE() ;;

  Kiírja, hogy dekódolás közben nem várt karakterrel találkoztnk.
- #define PRINTDEBUG_CUSTOM(str, ...) ;;

  Általános hibakeresésre használható, konzolra való kiírásra alkalmas.

5.3.1. Makródefiníciók dokumentációja

5.3.1.1. PRINTDEBUG_CORRUPTEDFILE

```
#define PRINTDEBUG_CORRUPTEDFILE( ) ;;
```

Kiírja, hogy dekódolás közben nem várt karakterrel találkoztnk.

5.3.1.2. PRINTDEBUG_CUSTOM

```
#define PRINTDEBUG_CUSTOM(
              str,
              ...  ) ;;
```

Általános hibakeresésre használható, konzolra való kiírásra alkalmas.

5.3.1.3. PRINTDEBUG_FILEERR

```
#define PRINTDEBUG_FILEERR( ) ;;
```

Kiírja, hogy a fájlművelet sikertelen volt.

5.3.1.4. PRINTDEBUG_MALLOCNULL

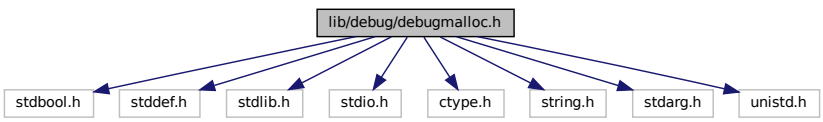
```
#define PRINTDEBUG_MALLOCNULL( ) ;;
```

Kiírja hogy egy memóriafoglalás sikertelen volt.

5.4. lib/debug/debugmalloc.h fájlreferencia

```
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdarg.h>
#include <unistd.h>
```

A debugmalloc.h definíciós fájl függési gráfja:



Adatszerkezetek

- struct DebugmallocEntry
- struct DebugmallocData

Makródefiníciók

- #define malloc(S) debugmalloc_malloc_full((S), "malloc", #S, __FILE__, __LINE__, false)
- #define calloc(N, S) debugmalloc_malloc_full((N)∗(S), "calloc", #N ", " #S, __FILE__, __LINE__, true)
- #define realloc(P, S) debugmalloc_realloc_full((P), (S), "realloc", #S, __FILE__, __LINE__)
- #define free(P) debugmalloc_free_full((P), "free", __FILE__, __LINE__)

Enumerációk

- enum { debugmalloc_canary_size = 64 , debugmalloc_canary_char = 'K' , debugmalloc_tablesize = 256 , debugmalloc_max_block_size_default = 1048576 }

5.4.1. Makródefiníciók dokumentációja

5.4.1.1. calloc

```
#define calloc(
              N,
              S ) debugmalloc_malloc_full((N)*(S), "calloc", #N ", " #S, __FILE__, __LINE__↩
, true)
```

5.4.1.2. free

```
#define free(
              P ) debugmalloc_free_full((P), "free", __FILE__, __LINE__)
```

5.4.1.3. malloc

```
#define malloc(
              S ) debugmalloc_malloc_full((S), "malloc", #S, __FILE__, __LINE__, false)
```

5.4.1.4. realloc

```
#define realloc(
              P,
              S ) debugmalloc_realloc_full((P), (S), "realloc", #S, __FILE__, __LINE__)
```

5.4.2. Enumerációk dokumentációja

5.4.2.1. anonymous enum

```
anonymous enum
```

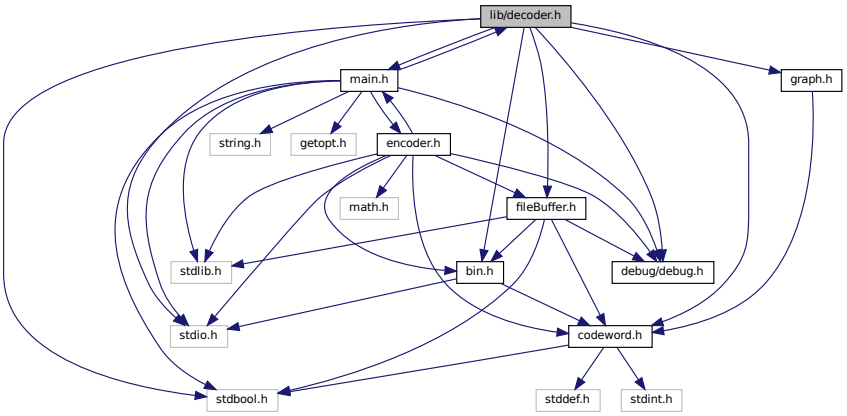
Enumeráció-értékek

debugmalloc_canary_size	
debugmalloc_canary_char	
debugmalloc_tablesize	
debugmalloc_max_block_size_default	

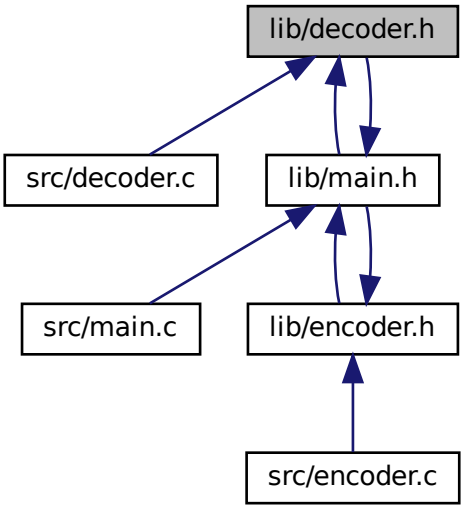
5.5. lib/decoder.h fájlreferencia

```
#include <stdio.h>
#include <stdbool.h>
#include "debug/debug.h"
#include "main.h"
#include "bin.h"
#include "fileBuffer.h"
#include "codeword.h"
#include "graph.h"
```

A decoder.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Függvények

- int decode (commandLineArguments args)

  Dekódol egy Shanon-Fano algoritmussal kódolt fájlt.

5.5.1. Függvények dokumentációja

5.5.1.1. decode()

```
int decode (
            commandLineArguments args )
```

Dekódol egy Shanon-Fano algoritmussal kódolt fájlt.

Paraméterek

args	Parancssori bemenet, amely a dekódolás folyamatát módosítja

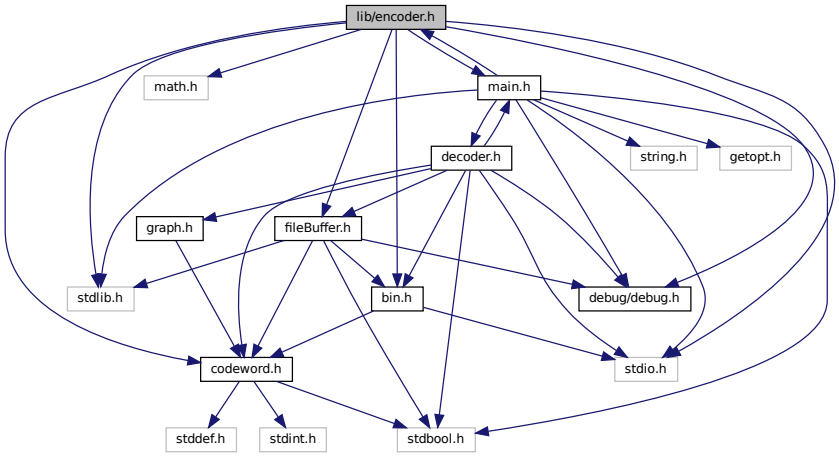
Visszatérési érték

0, ha a dekódolás sikeres volt. Minden más érték sikertelen

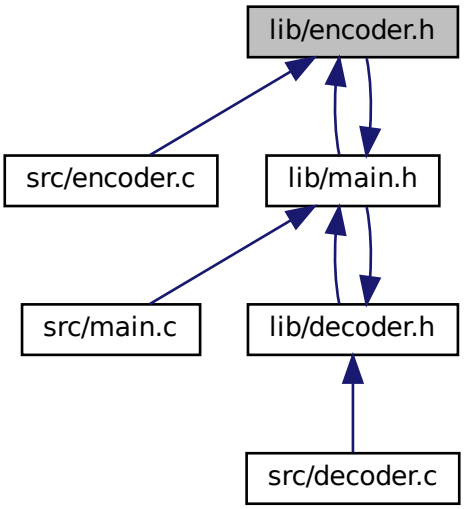
5.6. lib/encoder.h fájlreferencia

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "main.h"
#include "fileBuffer.h"
#include "codeword.h"
#include "bin.h"
#include "debug/debug.h"
```

Az encoder.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- struct codewordFrequency

Függvények

- int encode (commandLineArguments args)

  Kódol Shanon-Fano algoritmus alkalmazásával egy fájlt.

5.6.1. Függvények dokumentációja

5.6.1.1. encode()

```
int encode (
            commandLineArguments args )
```

Kódol Shanon-Fano algoritmus alkalmazásával egy fájlt.

Paraméterek

args	Parancssori bemenet, amely a dekódolás folyamatát módosítja

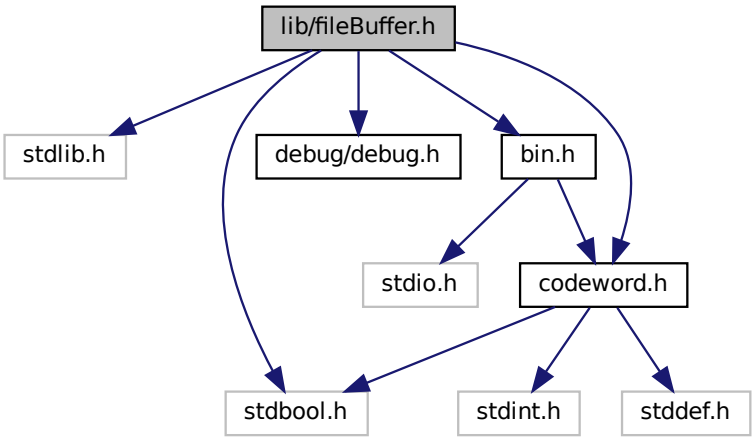
Visszatérési érték

0, ha a dekódolás sikeres volt. Minden más érték sikertelen

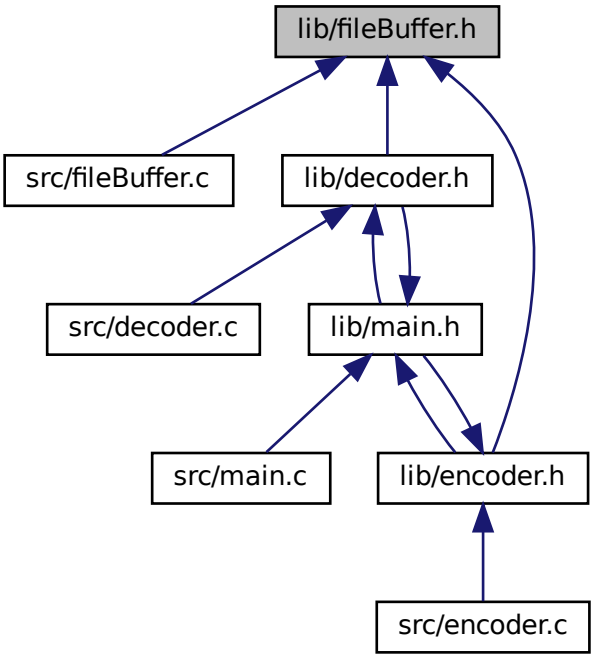
5.7. lib/fileBuffer.h fájlreferencia

```
#include <stdlib.h>
#include <stdbool.h>
#include "debug/debug.h"
#include "bin.h"
#include "codeword.h"
```

A fileBuffer.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- struct InputFileBuffer

Struktúra, mely lehetővé teszi a bitenkénti olvasást egy fájlból.

- struct OutputFileBuffer

  Struktúra, mely lehetővé teszi a bitenkénti írást egy fájlba.

Függvények

- InputFileBuffer buff_createInputFileBuffer (FILE *file)
- OutputFileBuffer buff_createOutputFileBuffer (FILE *file)
- void buff_destroyInputFileBuffer (InputFileBuffer buffer)
- void buff_destroyOutputFileBuffer (OutputFileBuffer buffer)
- void buff_rewind (InputFileBuffer buffer)
- bool buff_writeBits (OutputFileBuffer buff, Bits bit)
- bool buff_writeBit (OutputFileBuffer buff, Bits bit)
- bool buff_writeChar (OutputFileBuffer buff, uchar val)
- bool buff_writeInt (OutputFileBuffer buff, int val)
- bool buff_flush (OutputFileBuffer buff)
- Bits buff_readBit (InputFileBuffer buff)
- Bits buff_readBits (InputFileBuffer buff, int bitCount)
- Bits buff_readChar (InputFileBuffer buff)
- Bits buff_readInt (InputFileBuffer buff)

5.7.1. Függvények dokumentációja

5.7.1.1. buff_createInputFileBuffer()

```
InputFileBuffer buff_createInputFileBuffer (
            FILE * file )
```

Paraméterek

file	Az olvasandó file stream-je

Visszatérési érték

A kreált puffer

5.7.1.2. buff_createOutputFileBuffer()

```
OutputFileBuffer buff_createOutputFileBuffer (
            FILE * file )
```

5.7.1.3. buff_destroyInputFileBuffer()

```
void buff_destroyInputFileBuffer (
            InputFileBuffer buffer )
```

5.7.1.4. buff_destroyOutputFileBuffer()

```
void buff_destroyOutputFileBuffer (
            OutputFileBuffer buffer )
```

5.7.1.5. buff_flush()

```
bool buff_flush (
            OutputFileBuffer buff )
```

5.7.1.6. buff_readBit()

```
Bits buff_readBit (
            InputFileBuffer buff )
```

5.7.1.7. buff_readBits()

```
Bits buff_readBits (
            InputFileBuffer buff,
            int bitCount )
```

5.7.1.8. buff_readChar()

```
Bits buff_readChar (
            InputFileBuffer buff )
```

5.7.1.9. buff_readInt()

```
Bits buff_readInt (
            InputFileBuffer buff )
```

5.7.1.10. buff_rewind()

```
void buff_rewind (
            InputFileBuffer buffer )
```

5.7.1.11. buff_writeBit()

```
bool buff_writeBit (
            OutputFileBuffer buff,
            Bits bit )
```

5.7.1.12. buff_writeBits()

```
bool buff_writeBits (
            OutputFileBuffer buff,
            Bits bit )
```

5.7.1.13. buff_writeChar()

```
bool buff_writeChar (
            OutputFileBuffer buff,
            uchar val )
```

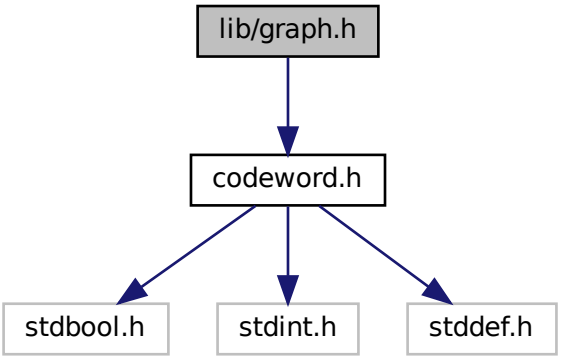
5.7.1.14. buff_writeInt()

```
bool buff_writeInt (
            OutputFileBuffer buff,
            int val )
```

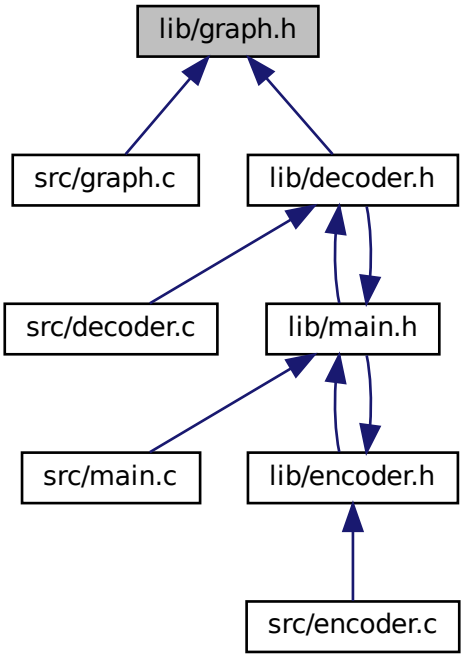
5.8. lib/graph.h fájlreferencia

```
#include "codeword.h"
```

A graph.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- struct Node

Függvények

- int graph_countLeaves (Node *root)

5.8.1. Függvények dokumentációja

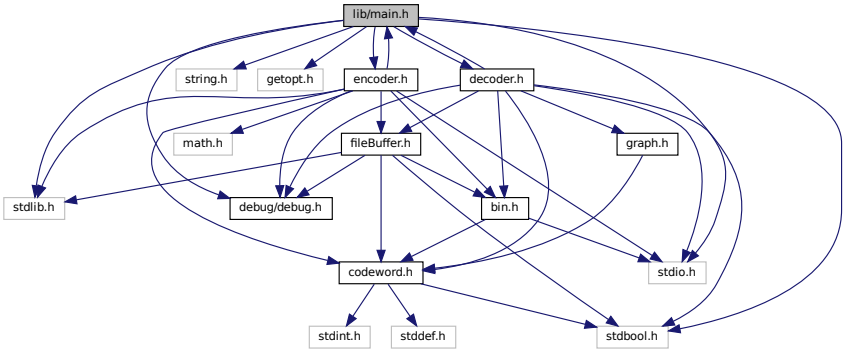
5.8.1.1. graph_countLeaves()

```
int graph_countLeaves (
            Node * root )
```

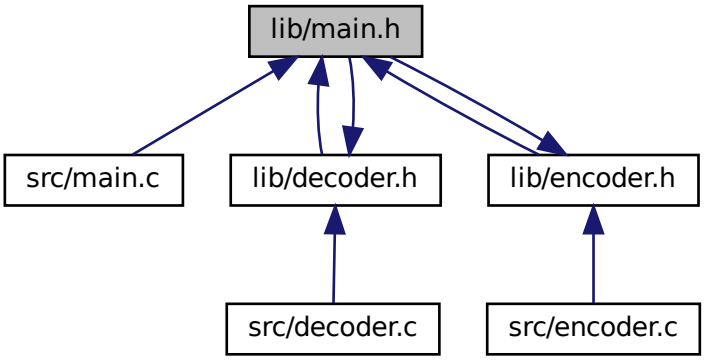
5.9. lib/main.h fájlreferencia

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <stdbool.h>
#include "debug/debug.h"
#include "encoder.h"
#include "decoder.h"
```

A main.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- struct commandLineArguments

  A program parancssori argumentumait rendező struktúra.

Enumerációk

- enum MODE { ENCODE = 0 , DECODE = 1 , UNSET = -1 }

  Megadja, hogy a program a 'kodol' vagy 'dekodol' paraméterrel lett meghívva.

5.9.1. Enumerációk dokumentációja

5.9.1.1. MODE

`enum MODE`

Megadja, hogy a program a 'kodol' vagy 'dekodol' paraméterrel lett meghívva.

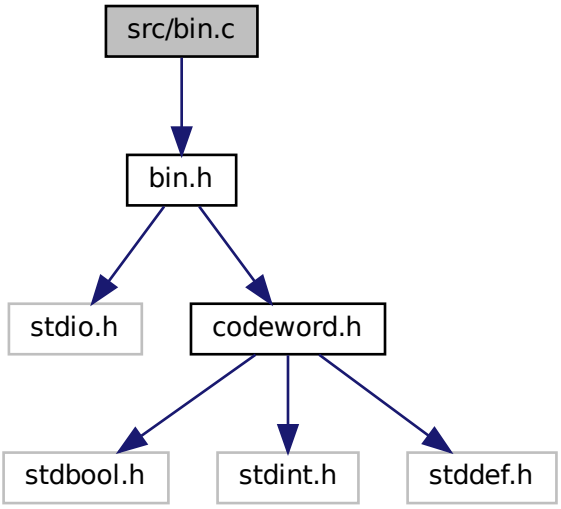
Enumeráció-értékek

ENCODE	
DECODE	
UNSET	

5.10. src/bin.c fájlreferencia

```
#include "bin.h"
```

A bin.c definíciós fájl függési gráfja:



Függvények

- Bits getBitFromRight (Bits bits, int n)

  Adott bitsorozatnak megadja a jobról számított n -edik bitjét.
- void bits_pushBit (Bits *bits, Bits append)

  Egy bitsorozatot bővít egy másik bitsorozattal jobb oldalról.
- void print_bits (Bits bits)

  Kiír egy bitsorozatot ASCII 0 és 1 karakterekkel.

5.10.1. Függvények dokumentációja

5.10.1.1. bits_pushBit()

```
void bits_pushBit (
            Bits * bits,
            Bits append )
```

Egy bitsorozatot bővít egy másik bitsorozattal jobb oldalról.

Paraméterek

bits	a bővítendő bitsorozat
append	A hozzáfűzendő bitsorozat

5.10.1.2. getBitFromRight()

```
Bits getBitFromRight (
            Bits bits,
            int n )
```

Adott bitsorozatnak megadja a jobról számított `n` -edik bitjét.

Paraméterek

bits	A bitsorozat, melyből kiválasztjuk a bitet
n	Jobbról számítva hányadik bit

Visszatérési érték

A keresett bit

5.10.1.3. print_bits()

```
void print_bits (
            Bits bits )
```

Kiír egy bitsorozatot ASCII 0 és 1 karakterekkel.

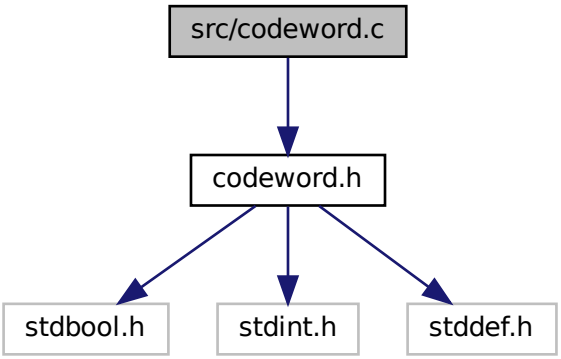
Paraméterek

bits	A kiírandó bitsorozat

5.11. src/codeword.c fájlreferencia

```
#include "codeword.h"
```

A codeword.c definíciós fájl függési gráfja:



Függvények

- bool bits_equ (Bits b1, Bits b2)

  Összehasonlít két bitsorozatot.
- bool isNullbit (Bits b)

  Megmondja, hogy egy adott bitsorozat értelmes-e.

5.11.1. Függvények dokumentációja

5.11.1.1. bits_equ()

```
bool bits_equ (
            Bits b1,
            Bits b2 )
```

Összehasonlít két bitsorozatot.

Paraméterek

b1	Az összehasonlítandó bitsorozat
b2	Az összehasonlítandó bitsorozat

Visszatérési érték

igaz, hogyha a bitsorozatok hossza és bitjei megegyeznek, különben hamis

5.11.1.2. isNullbit()

```
bool isNullbit (
            Bits b )
```

Megmondja, hogy egy adott bitsorozat értelmes-e.

Paraméterek

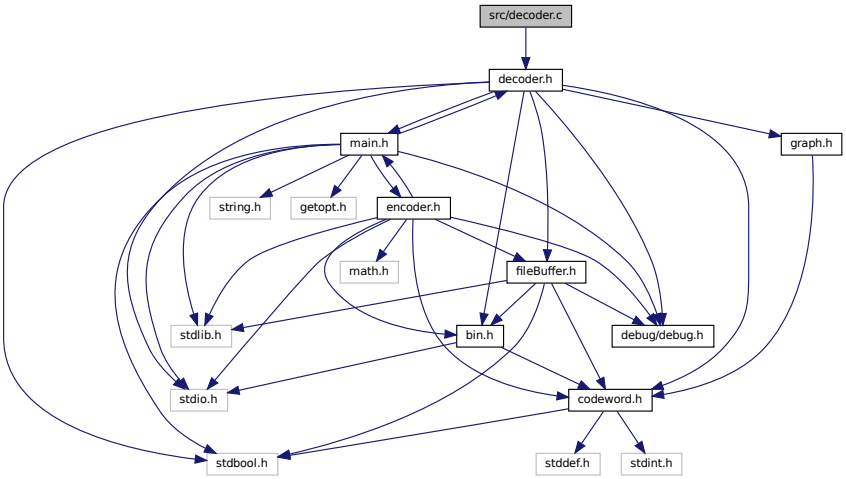
b	A vizsgálandó bitsorozat

Visszatérési érték

igaz, hogyha a bitsorozat hossza 0, különben hamis

5.12. src/decoder.c fájlreferencia

`#include "decoder.h"`
A decoder.c definíciós fájl függési gráfja:



Függvények

- void appendCodeword (Node ∗root, Bits codeword, char set)

  Egy fagráfhoz, `codeword` bitjeinek bejárása alapján beállítja egy elem kódolt karakterét. Ha az adott elérés nem létezik, a függvény létrehozza azt.
- Node ∗ createNodeIfNotexists (Node ∗parent, int dir)

  Egy fagráf adott eleméből megpróbál `dir` által meghatároott úton továbbhaladni. Ha az nem létezik, létrehozza azt.
- void freeTree (Node ∗root)

  Rekurzívan felszabadít egy fagráfot.
- int decode (commandLineArguments args)

  Dekódol egy Shanon-Fano algoritmussal kódolt fájlt.

5.12.1. Függvények dokumentációja

5.12.1.1. appendCodeword()

```
void appendCodeword (
            Node * root,
            Bits codeword,
            char set )
```

Egy fagráfhoz, `codeword` bitjeinek bejárása alapján beállítja egy elem kódolt karakterét. Ha az adott elérés nem létezik, a függvény létrehozza azt.

Paraméterek

root	A fagráf gyökere
codeword	A bejárás bitjei: 0 = bal, 1 (minden más) = jobb
set	A beállítandó karakter, melyhez elérkeztünk a bejárás végén

5.12.1.2. createNodeIfNotexists()

```
Node * createNodeIfNotexists (
            Node * parent,
            int dir )
```

Egy fagráf adott eleméből megpróbál `dir` által meghatároott úton továbbhaladni. Ha az nem létezik, létrehozza azt.

Paraméterek

parent	Az elem, melyből kiindulunk
dir	Az irány: 0 = bal, 1 (minden más) = jobb

Visszatérési érték

A gráf azon eleme, mely `parent` -től `dir` irányba helyezkedik el

5.12.1.3. decode()

```
int decode (
            commandLineArguments args )
```

Dekódol egy Shanon-Fano algoritmussal kódolt fájlt.

Paraméterek

args	Parancssori bemenet, amely a dekódolás folyamatát módosítja

Visszatérési érték

0, ha a dekódolás sikeres volt. Minden más érték sikertelen

5.12.1.4. freeTree()

```
void freeTree (
            Node * root )
```

Rekurzívan felszabadít egy fagráfot.

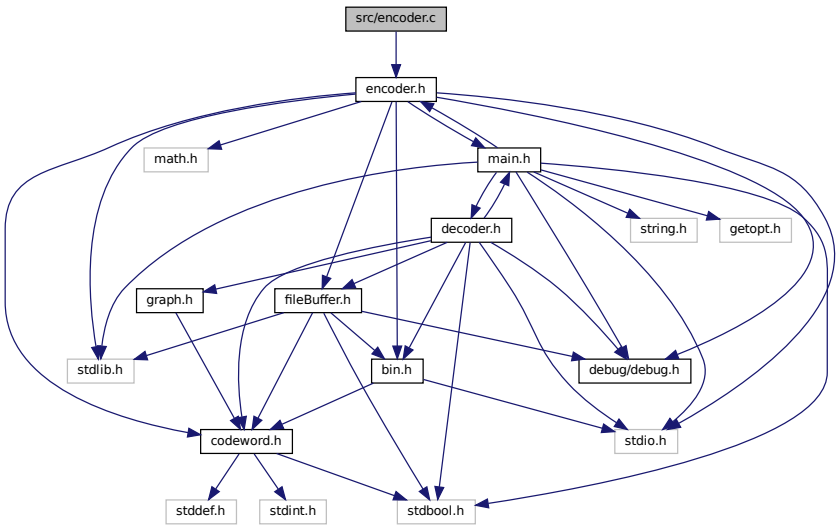
Paraméterek

root	a felszabadítandó gráf gyökere

5.13. src/encoder.c fájlreferencia

```
#include "encoder.h"
```

Az encoder.c definíciós fájl függési gráfja:



Függvények

- void setCodeWord (codewordFrequency codes[ ], int i, int j)

Rekurzívan beállítja egy kód tömbön az adott karakter Shanon-Fano algoritmus szerinti kódját.

- Bits codewordToBits (codewordFrequency code[ ], int codesLength, uchar find)

  Megkeresi `code` tömbben, `find` karaktert.
- int compare_by_freq (const void ∗a, const void ∗b)

  Frekvenciájuk alapján összehasonlít 2 frekvenciával rendelkező karakterkódolást.
- int compare_by_bitlength (const void ∗a, const void ∗b)

  Kódolásuk hossza alapján összehasonlít 2 rendelkező karakterkódolást.
- int encode (commandLineArguments args)

  Kódol Shanon-Fano algoritmus alkalmazásával egy fájlt.

5.13.1. Függvények dokumentációja

5.13.1.1. codewordToBits()

```
Bits codewordToBits (
            codewordFrequency code[],
            int codesLength,
            uchar find )
```

Megkeresi `code` tömbben, `find` karaktert.

Paraméterek

code	A kódtömb
codesLength	A kódtömb hossza
find	A keresett karakter

Visszatérési érték

A keresett kódolás vagy NULLBIT

5.13.1.2. compare_by_bitlength()

```
int compare_by_bitlength (
            const void * a,
            const void * b )
```

Kódolásuk hossza alapján összehasonlít 2 rendelkező karakterkódolást.

Paraméterek

a	Az összehasonlítandó karakterkódolás
b	Az összehasonlítandó karakterkódolás

Visszatérési érték

0 = egyeznek, >0 = a kódja hosszabb, <0 b kódja hosszabb

5.13.1.3. compare_by_freq()

```
int compare_by_freq (
            const void * a,
            const void * b )
```

Frekvenciájuk alapján összehasonlít 2 frekvenciával rendelkező karakterkódolást.

Paraméterek

a	Az összehasonlítandó karakterkódolás
b	Az összehasonlítandó karakterkódolás

Visszatérési érték

0 = egyeznek, >0 = b frekvenciája nagyobb, <0 a frekvenciája nagyobb

5.13.1.4. encode()

```
int encode (
            commandLineArguments args )
```

Kódol Shanon-Fano algoritmus alkalmazásával egy fájlt.

Paraméterek

args	Parancssori bemenet, amely a dekódolás folyamatát módosítja

Visszatérési érték

0, ha a dekódolás sikeres volt. Minden más érték sikertelen

5.13.1.5. setCodeWord()

```
void setCodeWord (
            codewordFrequency codes[],
            int i,
            int j )
```

Rekurzívan beállítja egy kód tömbön az adott karakter Shanon-Fano algoritmus szerinti kódját.

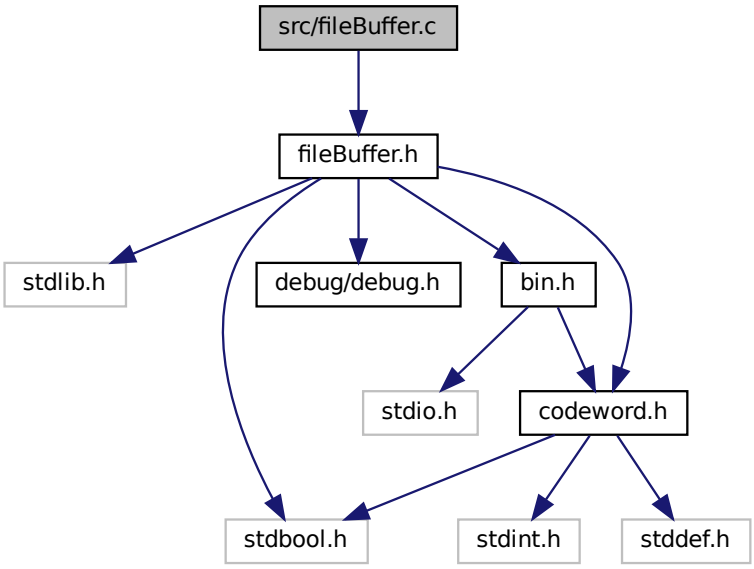
Paraméterek

codes	A kódtömb
i	A tömb kezdeti indexe (inkluzív)
j	A tömb vegső indexe (inkluzív)

5.14. src/fileBuffer.c fájlreferencia

```
#include "fileBuffer.h"
```

A fileBuffer.c definíciós fájl függési gráfja:



Függvények

- InputFileBuffer buff_createInputFileBuffer (FILE *file)
- OutputFileBuffer buff_createOutputFileBuffer (FILE *file)
- void buff_destroyInputFileBuffer (InputFileBuffer buffer)
- void buff_destroyOutputFileBuffer (OutputFileBuffer buffer)
- void buff_rewind (InputFileBuffer buffer)
- bool buff_writeBits (OutputFileBuffer buff, Bits bit)
- bool buff_writeBit (OutputFileBuffer buff, Bits bit)
- bool buff_writeChar (OutputFileBuffer buff, uchar val)
- bool buff_writeInt (OutputFileBuffer buff, int val)
- bool buff_flush (OutputFileBuffer buff)
- Bits buff_readBit (InputFileBuffer buff)
- Bits buff_readBits (InputFileBuffer buff, int bitCount)
- Bits buff_readChar (InputFileBuffer buff)
- Bits buff_readInt (InputFileBuffer buff)

5.14.1. Függvények dokumentációja

5.14.1.1. buff_createInputFileBuffer()

```
InputFileBuffer buff_createInputFileBuffer (
            FILE * file )
```

Paraméterek

file	Az olvasandó file stream-je

Visszatérési érték

A kreált puffer

5.14.1.2. buff_createOutputFileBuffer()

```
OutputFileBuffer buff_createOutputFileBuffer (
            FILE * file )
```

5.14.1.3. buff_destroyInputFileBuffer()

```
void buff_destroyInputFileBuffer (
            InputFileBuffer buffer )
```

5.14.1.4. buff_destroyOutputFileBuffer()

```
void buff_destroyOutputFileBuffer (
            OutputFileBuffer buffer )
```

5.14.1.5. buff_flush()

```
bool buff_flush (
            OutputFileBuffer buff )
```

5.14.1.6. buff_readBit()

```
Bits buff_readBit (
            InputFileBuffer buff )
```

5.14.1.7. buff_readBits()

```
Bits buff_readBits (
            InputFileBuffer buff,
            int bitCount )
```

5.14.1.8. buff_readChar()

```
Bits buff_readChar (
            InputFileBuffer buff )
```

5.14.1.9. buff_readInt()

```
Bits buff_readInt (
            InputFileBuffer buff )
```

5.14.1.10. buff_rewind()

```
void buff_rewind (
            InputFileBuffer buffer )
```

5.14.1.11. buff_writeBit()

```
bool buff_writeBit (
            OutputFileBuffer buff,
            Bits bit )
```

5.14.1.12. buff_writeBits()

```
bool buff_writeBits (
            OutputFileBuffer buff,
            Bits bit )
```

5.14.1.13. buff_writeChar()

```
bool buff_writeChar (
            OutputFileBuffer buff,
            uchar val )
```

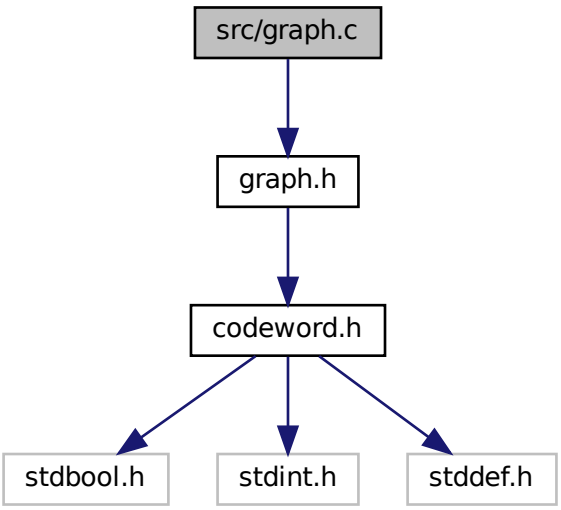
5.14.1.14. buff_writeInt()

```
bool buff_writeInt (
            OutputFileBuffer buff,
            int val )
```

5.15. src/graph.c fájlreferencia

```
#include "graph.h"
```

A graph.c definíciós fájl függési gráfja:



Függvények

- int graph_countLeaves (Node *root)

5.15.1. Függvények dokumentációja

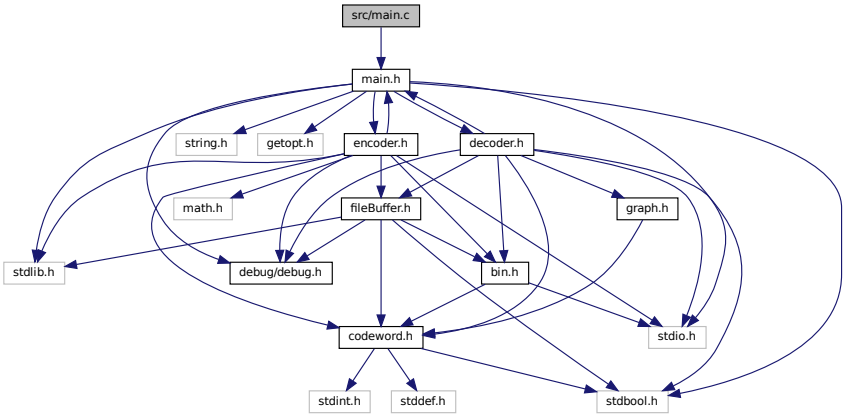
5.15.1.1. graph_countLeaves()

```
int graph_countLeaves (
            Node * root )
```

5.16. src/main.c fájlreferencia

```
#include "main.h"
```

A main.c definíciós fájl függési gráfja:



Függvények

- void printHelp ()
- int parseCLA (int argc, char **argv, commandLineArguments *args, enum MODE *mode)

  A bemeneti paraméterek feldolgozására szolgáló függvény.
- int main (int argc, char **argv)

5.16.1. Függvények dokumentációja

5.16.1.1. main()

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

5.16.1.2. parseCLA()

```
int parseCLA (
             int argc,
             char ** argv,
             commandLineArguments * args,
             enum MODE * mode )
```

A bemeneti paraméterek feldolgozására szolgáló függvény.

Paraméterek

argc	'argv' hossza
argv	parancssori argumentumok
args	

Visszatérési érték

0, ha sikeres volt az argumentumok elemzése, különben ettől eltérő

5.16.1.3. printHelp()

```
void printHelp ( )
```

Meta

5.17. Források, felhasznált irodalom

- Wayback Machine: C. E. Shannon, „A Mathematical Theory of Communication”, 1948 (https://web.archive.org/web/19980715014
  labs.com/cm/ms/what/shannonday/shannon1948.pdf)
- Halley’s Comet software: Robert M. Fano, „The Transmittion of Information”, 1949 (https://hcs64.com/files/fano-tr65-ocr-only.pdf)
- Linux man pages online: (https://man7.org/linux/man-pages/index.html)
- BME InfoC: (https://infoc.eet.bme.hu)

5.18. Felhasznált segédprogramok

- Fejlesztői környezet: Visual Studio Code (https://code.visualstudio.com/)
- C compiler: GNU Compiler Collection (GCC) (https://gcc.gnu.org/)
- Projekt fordítása: Make (https://www.gnu.org/software/make/)
- Dokumentáció: Doxygen (https://www.doxygen.nl/)