

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

Shanon-Fano kódoló és dekódoló program

készítette:

Ferencz Péter

(RFG7SN)



M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Mérnök-informatikus Bsc

2024 Október-November

1. Specifikáció	1
1.1. A program célja	1
1.2. Felhasználói interakció	1
1.2.1. kódolás (kodol)	1
1.2.2. dekódolás (dekodol)	1
1.3. A program által elfogadott kapcsolók	2
1.3.1. Bemenet	2
1.3.2. Kimenet	2
1.3.3. Kódtábla	2
1.3.4. Statisztika	2
1.3.5. Segítség	3
1.4. A program kimenete	3
2. Adatszerkezet-mutató	5
2.1. Adatszerkezetek	5
3. Fájlmutató	7
3.1. Fájllista	7
4. Adatszerkezetek dokumentációja	9
4.1. Bits struktúrareferencia	9
4.1.1. Részletes leírás	9
4.1.2. Adatmezők dokumentációja	9
4.1.2.1. b	9
4.1.2.2. length	10
4.2. CodeWord struktúrareferencia	10
4.2.1. Részletes leírás	10
4.2.2. Adatmezők dokumentációja	10
4.2.2.1. bits	11
4.2.2.2. codeWord	11
4.3. codewordFrequency struktúrareferencia	11
4.3.1. Részletes leírás	12
4.3.2. Adatmezők dokumentációja	12
4.3.2.1. codeWord	12
4.3.2.2. freq	12
4.4. commandLineArguments struktúrareferencia	12
4.4.1. Részletes leírás	13
4.4.2. Adatmezők dokumentációja	13
4.4.2.1. displayStatistics	13
4.4.2.2. displayTable	13
4.4.2.3. infile	13
4.4.2.4. mode	13
4.4.2.5. outfile	13

4.5. InputFileBuffer struktúráreferencia	14
4.5.1. Részletes leírás	14
4.5.2. Adatmezők dokumentációja	14
4.5.2.1. currentBit	14
4.5.2.2. file	14
4.6. Node struktúráreferencia	15
4.6.1. Adatmezők dokumentációja	15
4.6.1.1. codeword	15
4.6.1.2. left_0	15
4.6.1.3. right_1	15
4.6.1.4. set	16
4.7. OutputFileBuffer struktúráreferencia	16
4.7.1. Részletes leírás	16
4.7.2. Adatmezők dokumentációja	16
4.7.2.1. bits	17
4.7.2.2. file	17
5. Fájlok dokumentációja	19
5.1. lib/bin.h fájlreferencia	19
5.1.1. Makródefiníciók dokumentációja	21
5.1.1.1. NULLBIT	21
5.1.2. Függvények dokumentációja	21
5.1.2.1. bits_cpy()	21
5.1.2.2. bits_equ()	21
5.1.2.3. bits_isNullbit()	21
5.1.2.4. bits_popBit()	22
5.1.2.5. bits_popBits()	22
5.1.2.6. bits_print()	23
5.1.2.7. bits_pushBit()	23
5.1.2.8. bits_pushBits()	23
5.1.2.9. getBitFromRight()	23
5.2. lib/codeword.h fájlreferencia	25
5.2.1. Típusdefiníciók dokumentációja	26
5.2.1.1. uchar	26
5.3. lib/debug/debug.h fájlreferencia	27
5.3.1. Makródefiníciók dokumentációja	27
5.3.1.1. CHECKMALLOCNULL	27
5.3.1.2. PRINTDEBUG_CORRUPTEDFILE	28
5.3.1.3. PRINTDEBUG_CUSTOM	28
5.3.1.4. PRINTDEBUG_FILEERR	28
5.3.1.5. PRINTDEBUG_MALLOCNULL	28
5.4. lib/decoder.h fájlreferencia	28

5.4.1.	Függvények dokumentációja	29
5.4.1.1.	decode()	30
5.5.	lib/encoder.h fájlreferencia	30
5.5.1.	Függvények dokumentációja	31
5.5.1.1.	encode()	31
5.6.	lib/fileBuffer.h fájlreferencia	32
5.6.1.	Függvények dokumentációja	34
5.6.1.1.	buff_createInputFileBuffer()	34
5.6.1.2.	buff_createOutputFileBuffer()	34
5.6.1.3.	buff_destroyInputFileBuffer()	35
5.6.1.4.	buff_destroyOutputFileBuffer()	35
5.6.1.5.	buff_flush()	35
5.6.1.6.	buff_readBit()	36
5.6.1.7.	buff_readBits()	36
5.6.1.8.	buff_readChar()	36
5.6.1.9.	buff_readInt()	37
5.6.1.10.	buff_rewind()	37
5.6.1.11.	buff_writeBit()	37
5.6.1.12.	buff_writeBits()	38
5.6.1.13.	buff_writeChar()	38
5.6.1.14.	buff_writeInt()	38
5.7.	lib/graph.h fájlreferencia	39
5.7.1.	Függvények dokumentációja	40
5.7.1.1.	freeTree()	40
5.7.1.2.	graph_countLeaves()	41
5.8.	lib/main.h fájlreferencia	41
5.8.1.	Enumerációk dokumentációja	42
5.8.1.1.	MODE	42
5.9.	lib/stats.h fájlreferencia	43
5.9.1.	Függvények dokumentációja	44
5.9.1.1.	stats_printCodetableArray()	44
5.9.1.2.	stats_printCodetableStatsArray()	44
5.9.1.3.	stats_printCodetableTree()	44
5.9.1.4.	stats_printCompression()	45
5.10.	src/bin.c fájlreferencia	45
5.10.1.	Függvények dokumentációja	46
5.10.1.1.	bits_cpy()	46
5.10.1.2.	bits_equ()	46
5.10.1.3.	bits_isNullbit()	47
5.10.1.4.	bits_popBit()	47
5.10.1.5.	bits_popBits()	47
5.10.1.6.	bits_print()	48

5.10.1.7. bits_pushBit()	48
5.10.1.8. bits_pushBits()	48
5.10.1.9. getBitFromRight()	49
5.11. src/decoder.c fájlreferencia	49
5.11.1. Függvények dokumentációja	50
5.11.1.1. appendCodeword()	50
5.11.1.2. createNodeIfNotexists()	50
5.11.1.3. decode()	50
5.11.1.4. displayTable()	51
5.11.1.5. freeTree()	51
5.12. src/encoder.c fájlreferencia	51
5.12.1. Függvények dokumentációja	52
5.12.1.1. codewordToBits()	52
5.12.1.2. compare_by_bitlength()	52
5.12.1.3. compare_by_freq()	53
5.12.1.4. encode()	53
5.12.1.5. setCodeWord()	53
5.13. src/fileBuffer.c fájlreferencia	54
5.13.1. Függvények dokumentációja	55
5.13.1.1. buff_createInputFileBuffer()	55
5.13.1.2. buff_createOutputFileBuffer()	55
5.13.1.3. buff_destroyInputFileBuffer()	56
5.13.1.4. buff_destroyOutputFileBuffer()	56
5.13.1.5. buff_flush()	56
5.13.1.6. buff_readBit()	57
5.13.1.7. buff_readBits()	57
5.13.1.8. buff_readChar()	57
5.13.1.9. buff_readInt()	58
5.13.1.10. buff_rewind()	58
5.13.1.11. buff_writeBit()	58
5.13.1.12. buff_writeBits()	59
5.13.1.13. buff_writeChar()	59
5.13.1.14. buff_writeInt()	59
5.14. src/graph.c fájlreferencia	60
5.14.1. Függvények dokumentációja	60
5.14.1.1. freeTree()	61
5.14.1.2. graph_countLeaves()	61
5.15. src/main.c fájlreferencia	61
5.15.1. Függvények dokumentációja	62
5.15.1.1. main()	62
5.15.1.2. parseCLA()	62
5.15.1.3. printHelp()	63

5.16. src/stats.c fájlreferencia	63
5.16.1. Függvények dokumentációja	64
5.16.1.1. calcFileSize()	64
5.16.1.2. prettyPrintChar()	64
5.16.1.3. printCodetableTreeRec()	64
5.16.1.4. stats_printCodetableArray()	65
5.16.1.5. stats_printCodetableStatsArray()	65
5.16.1.6. stats_printCodetableTree()	65
5.16.1.7. stats_printCompression()	66
Meta	67
5.17. Források, felhasznált irodalom	67
5.18. Felhasznált segédprogramok	67

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>
```

¹C. E. Shannon, „A Mathematical Theory of Communication”, 1948

²Robert M. Fano, „The Transmission 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: `--kodtábla`

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 számításokat a program hatékonyságára vonatkozólag.

Az alábbi számítások történnek kiírásra:

- Tömörítés mértéke: bemenet mérete a tömörített adat méretéhez képest
- Kódtábla mérete: Egymástól eltérő kódok száma
- Kódok mérete: legrövidebb kód, leghosszabb kód, kódok átlagos 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...>
Üzem mód: kodol, dekodol
Kapcsolók:
--bemenet <forrásfájl>: Bemeneti fájl (ha üres akkor stdin)
--kimenet <célfájl>: Bemeneti fájl (ha üres akkor stdout)
--kódtábla <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

Kódolt karakterek hossza (1-256)	Illeszkedés hossza (0-7)	Kódtábla				Kódolt adat	Illeszkedés (0)
		karakter ASCII	karakterkód hossza	kód	• • • • •		
		8 bit	n = 8 bit	n bit			
I = 8 bit	i = 3 bit	legalább I * (8 + 8 + 1) bit				legalább I bit	i bit

2. fejezet

Adatszerkezet-mutató

2.1. Adatszerkezetek

Az összes adatszerkezet listája rövid leírásokkal:

Bits	Tetszőleges hosszú bitsorozat eltárolására alkalmas struktúra	9
CodeWord	Karakter, és az azt kódoló bitsorozat	10
codewordFrequency	Kódolt szót és annak a szövegben előfordulásának gyakoriságát eltároló struktúra	11
commandLineArguments	A program parancssori argumentumait rendező struktúra	12
InputFileBuffer	Struktúra, mely lehetővé teszi a bitenkénti olvasást egy fájlból	14
Node	15
OutputFileBuffer	Struktúra, mely lehetővé teszi a bitenkénti írást egy fájlba	16

3. fejezet

Fájlmutató

3.1. Fájllista

Az összes fájl listája rövid leírásokkal:

lib/bin.h	19
lib/codeword.h	25
lib/decoder.h	28
lib/encoder.h	30
lib/fileBuffer.h	32
lib/graph.h	39
lib/main.h	41
lib/stats.h	43
lib/debug/debug.h	27
src/bin.c	45
src/decoder.c	49
src/encoder.c	51
src/fileBuffer.c	54
src/graph.c	60
src/main.c	61
src/stats.c	63

4. fejezet

Adatszerkezetek dokumentációja

4.1. Bits struktúrareferencia

Tetszőleges hosszú bitsorozat eltárolására alkalmas struktúra.

```
#include <bin.h>
```

Adatmezők

- long long unsigned int **b**
A tárolt szám A bitek jobbról balra értelmezendők.
- 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 értelmezendők.

4.1.2.2. length

```
size_t Bits::length
```

A tárolt bitsorozat hossza.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

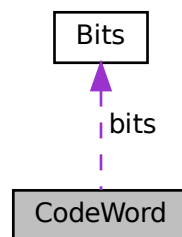
- lib/[bin.h](#)

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:

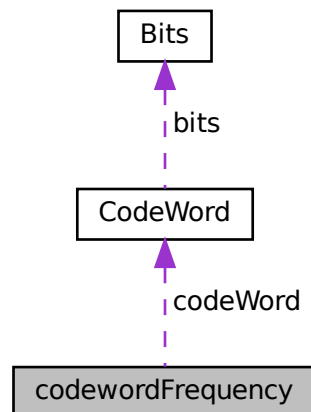
- lib/[codeword.h](#)

4.3. codewordFrequency struktúráreferencia

Kódolt szót és annak a szövegben előfordulásának gyakoriságát eltároló struktúra.

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

A codewordFrequency osztály együttműködési diagramja:



Adatmezők

- float [freq](#)
Az adott kódolt szó előfordulásának frekvenciája.
- [CodeWord](#) [codeWord](#)
Kódolt szó

4.3.1. Részletes leírás

Kódolt szót és annak a szövegben előfordulásának gyakoriságát eltároló struktúra.

4.3.2. Adatmezők dokumentációja

4.3.2.1. codeWord

```
CodeWord codewordFrequency::codeWord
```

Kódolt szó

4.3.2.2. freq

```
float codewordFrequency::freq
```

Az adott kódolt szó előfordulásának frekvenciája.

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

- const char * [infile](#)
A –bemenet kapcsoló által megadott stream.
- const char * [outfile](#)
A –kimenet kapcsoló által megadott stream.
- enum [MODE](#) [mode](#)
A program üzemmódja: kódolás vagy dekódolás.
- bool [displayTable](#)
Megajda, hogy a program kiírja-e a kódtáblát.
- bool [displayStatistics](#)
Megajda, hogy a program kiírjon-e további számításokat 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 számításokat 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

```
const char* commandLineArguments::infile
```

A –bemenet kapcsoló által megadott stream.

4.4.2.4. mode

```
enum MODE commandLineArguments::mode
```

A program üzemmódja: kódolás vagy dekódolás.

4.4.2.5. outfile

```
const char* commandLineArguments::outfile
```

A –kimenet kapcsoló által megadott stream.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- [lib/main.h](#)

4.5. InputFileBuffer struktúráreferencia

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.

4.5.1. Részletes leírás

Struktúra, mely lehetővé teszi a bitenkénti olvasást egy fájlból.

4.5.2. Adatmezők dokumentációja

4.5.2.1. currentBit

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

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

4.5.2.2. file

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

A fájl, melyből olvasunk.

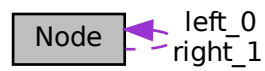
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- lib/[fileBuffer.h](#)

4.6. Node struktúráreferencia

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

A Node osztály együttműködési diagramja:



Adatmezők

- `uchar` `codeword`
- `bool` `set`
- `struct Node *` `left_0`
- `struct Node *` `right_1`

4.6.1. Adatmezők dokumentációja

4.6.1.1. `codeword`

```
uchar Node::codeword
```

4.6.1.2. `left_0`

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

4.6.1.3. `right_1`

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

4.6.1.4. set

```
bool Node::set
```

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

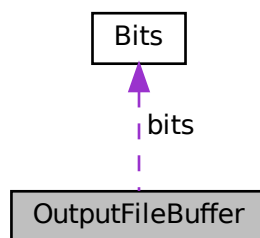
- [lib/graph.h](#)

4.7. 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.

4.7.1. Részletes leírás

Struktúra, mely lehetővé teszi a bitenkénti írást egy fájlba.

4.7.2. Adatmezők dokumentációja

4.7.2.1. bits

`Bits* OutputFileBuffer::bits`

A még nem a fájlba beírt bitek.

4.7.2.2. file

`FILE* OutputFileBuffer::file`

A fájl, melybe írunk.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- `lib/fileBuffer.h`

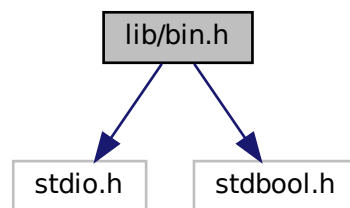
5. fejezet

Fájlok dokumentációja

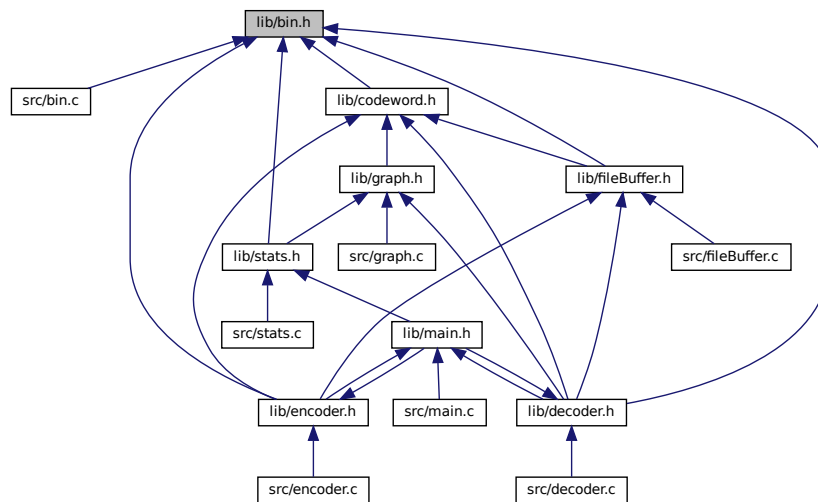
5.1. lib/bin.h fájlreferencia

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

A bin.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.

Makródefiníciók

- #define **NULLBIT** ((Bits){ .b = 0, .length = 0 })
Hibás kimenetet jelentő bitsorozat, melynek hossza 0.

Függvények

- **Bits** **getBitFromRight** (Bits bits, int n)
Adott bitsorozatnak megadja a jobbról számított n -edik bitjét.
- void **bits_pushBit** (Bits *bits, char b)
Egy bitsorozatot bővít egy b bittel.
- void **bits_pushBits** (Bits *bits, Bits append)
Egy bitsorozatot bővít egy másik bitsorozattal jobb oldalról.
- **Bits** **bits_popBit** (Bits *bits)
Egy bitsorozatból adja vissza a legkisebb helyiértéken álló bitet, majd azt eltávolítja.
- **Bits** **bits_popBits** (Bits *bits, int length)
Egy bitsorozatból adja vissza a legkisebb helyiértéktől számolva $length$ bitet, majd azokat eltávolítja.
- void **bits_cpy** (Bits src, Bits *dest)
- void **bits_print** (Bits bits)
Kiír egy bitsorozatot ASCII 0 és 1 karakterekkel.
- bool **bits_equ** (Bits b1, Bits b2)
Összehasonlít két bitsorozatot.
- bool **bits_isNullbit** (Bits b)
Megmondja, hogy egy adott bitsorozat értelmes-e.

5.1.1. Makródefiníciók dokumentációja

5.1.1.1. NULLBIT

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

Hibás kimenetet jelentő bitsorozat, melynek hossza 0.

5.1.2. Függvények dokumentációja

5.1.2.1. bits_cpy()

```
void bits_cpy (  
    Bits src,  
    Bits * dest )
```

5.1.2.2. bits_equ()

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

Összehasonlít két bitsorozatot.

Paraméterek

<i>b1</i>	Az összehasonlítandó bitsorozat
<i>b2</i>	Az összehasonlítandó bitsorozat

Visszatérési érték

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

5.1.2.3. bits_isNullbit()

```
bool bits_isNullbit (  
    Bits b )
```

Megmondja, hogy egy adott bitsorozat értelmes-e.

Paraméterek

<i>b</i>	A vizsgálandó bitsorozat
----------	--------------------------

Visszatérési érték

igaz, hogyha a bitsorozat hossza 0, különben hamis

5.1.2.4. bits_popBit()

```
Bits bits_popBit (  
    Bits * bits )
```

Egy bitsorozatból adja vissza a legkisebb helyiértéken álló bitet, majd azt eltávolítja.

Paraméterek

<i>bits</i>	A bitsorozat, melyből kivesszük a bitet
-------------	---

Visszatérési érték

5.1.2.5. bits_popBits()

```
Bits bits_popBits (  
    Bits * bits,  
    int length )
```

Egy bitsorozatból adja vissza a legkisebb helyiértéktől számolva *length* bitet, majd azokat eltávolítja.

Paraméterek

<i>bits</i>	A bitsorozat, melyből kivesszük a biteket
<i>length</i>	A kivett bitek száma

Visszatérési érték

5.1.2.6. bits_print()

```
void bits_print (
    Bits bits )
```

Kiír egy bitsorozatot ASCII 0 és 1 karakterekkel.

Paraméterek

<i>bits</i>	A kiírandó bitsorozat
-------------	-----------------------

5.1.2.7. bits_pushBit()

```
void bits_pushBit (
    Bits * bits,
    char b )
```

Egy bitsorozatot bővít egy *b* bittel.

Paraméterek

<i>bits</i>	A bővítendő bitsorozat
<i>b</i>	a hozzáadott bit

5.1.2.8. bits_pushBits()

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

Egy bitsorozatot bővít egy másik bitsorozattal jobb oldalról.

Paraméterek

<i>bits</i>	A bővítendő bitsorozat
<i>append</i>	A hozzáfűzendő bitsorozat

5.1.2.9. getBitFromRight()

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

Adott bitsorozatnak megadja a jobbról számított n -edik bitjét.

Paraméterek

<i>bits</i>	A bitsorozat, melyből kiválasztjuk a bitet
<i>n</i>	Jobbról számítva hányadik bit

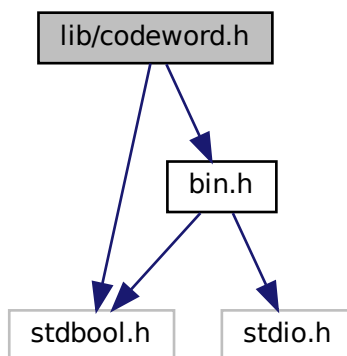
Visszatérési érték

A keresett bit

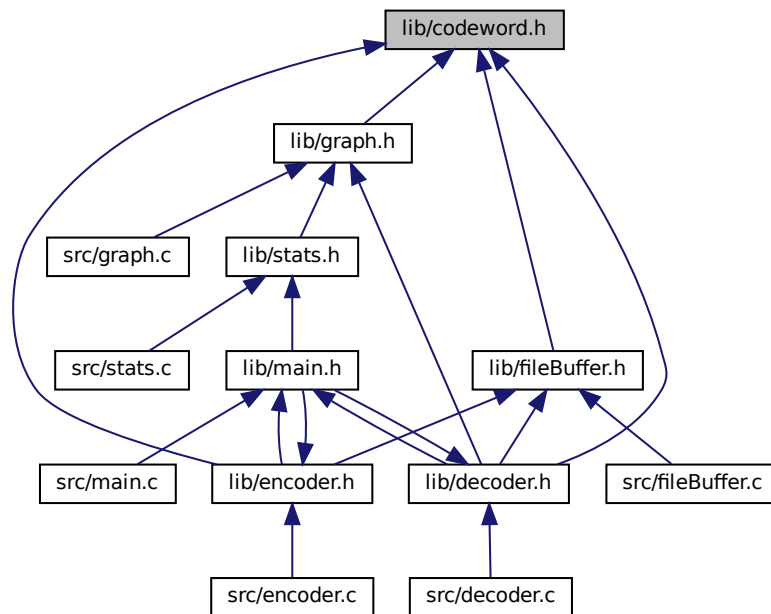
5.2. lib/codeword.h fájlreferencia

```
#include <stdbool.h>
#include "bin.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 [CodeWord](#)
Karakter, és az azt kódoló bitsorozat.

Típusdefiníciók

- typedef unsigned char [uchar](#)
Előjel nélküli 8 bites karakter.

5.2.1. Típusdefiníciók dokumentációja

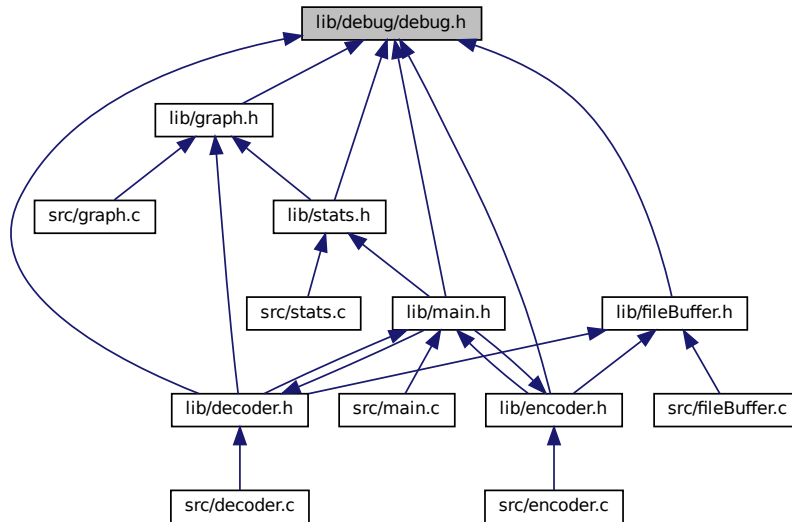
5.2.1.1. uchar

```
typedef unsigned char uchar
```

Előjel nélküli 8 bites karakter.

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 CHECKMALLOCNUL(ptr) if(ptr == NULL) { PRINTDEBUG_MALLOCNUL(); exit(1); }`
Megnézi, hogy `ptr` `NULL`-e. Ha igen, akkor kilép a programból 1-es értékkel, különben nem változtatja meg a futást.
- `#define PRINTDEBUG_MALLOCNUL() ;;`
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 talákoztunk.
- `#define PRINTDEBUG_CUSTOM(str, ...) ;;`
Általános hibakeresésre használható, konzolra való kiírásra alkalmas.

5.3.1. Makródefiníciók dokumentációja

5.3.1.1. CHECKMALLOCNUL

```

#define CHECKMALLOCNUL(
    ptr ) if(ptr == NULL) { PRINTDEBUG_MALLOCNUL(); exit(1); }
  
```

Megnézi, hogy `ptr` `NULL`-e. Ha igen, akkor kilép a programból 1-es értékkel, különben nem változtatja meg a futást.

5.3.1.2. PRINTDEBUG_CORRUPTEDFILE

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

Kiírja, hogy dekódolás közben nem várt karakterrel találkoztnk.

5.3.1.3. PRINTDEBUG_CUSTOM

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

Általános hibakeresésre használható, konzolra való kiírásra alkalmas.

5.3.1.4. PRINTDEBUG_FILEERR

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

Kiírja, hogy a fájlművelet sikertelen volt.

5.3.1.5. PRINTDEBUG_MALLOCNULL

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

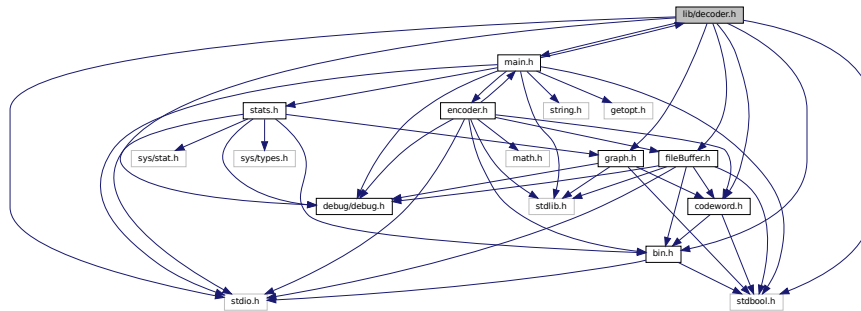
Kiírja hogy egy memóriafoglalás sikertelen volt.

5.4. lib/decoder.h fájlreferencia

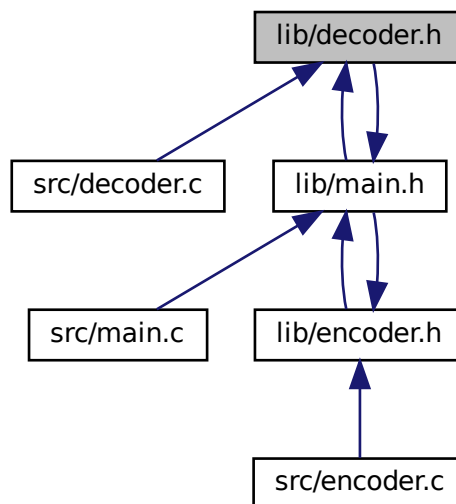
```
#include <stdio.h>  
#include <stdbool.h>  
#include "debug/debug.h"  
#include "main.h"  
#include "bin.h"  
#include "fileBuffer.h"  
#include "codeword.h"
```

```
#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.

5.4.1. Függvények dokumentációja

5.4.1.1. decode()

```
int decode (
    CommandLineArguments args )
```

Dekódol egy Shannon-Fano algoritmussal kódolt fájlt.

Paraméterek

<code>args</code>	Parancssori bemenet, amely a dekódolás folyamatát módosítja
-------------------	---

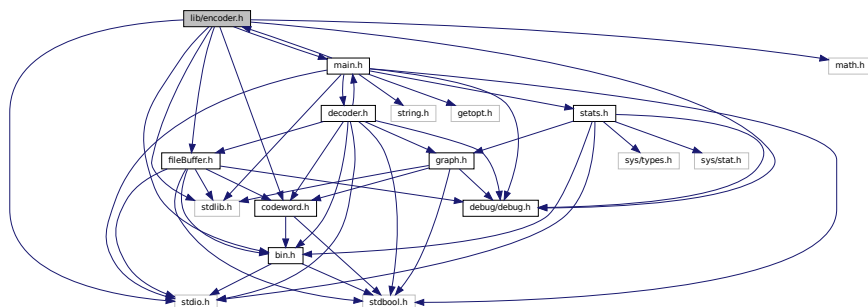
Visszatérési érték

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

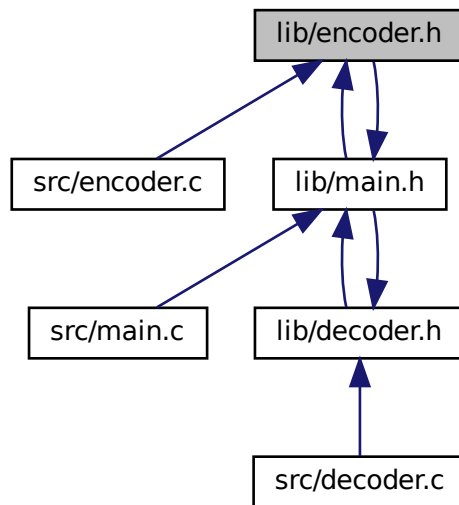
5.5. lib/encoder.h fájlreferencia

```
#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`

Kódolt szót és annak a szövegben előfordulásának gyakoriságát eltároló struktúra.

Függvények

- int `encode` (`commandLineArguments` args)

Kódol Shannon-Fano algoritmus alkalmazásával egy fájlt.

5.5.1. Függvények dokumentációja

5.5.1.1. `encode()`

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

Kódol Shannon-Fano algoritmus alkalmazásával egy fájlt.

Paraméterek

args	Parancssori bemenet, amely a dekódolás folyamatát módosítja
------	---

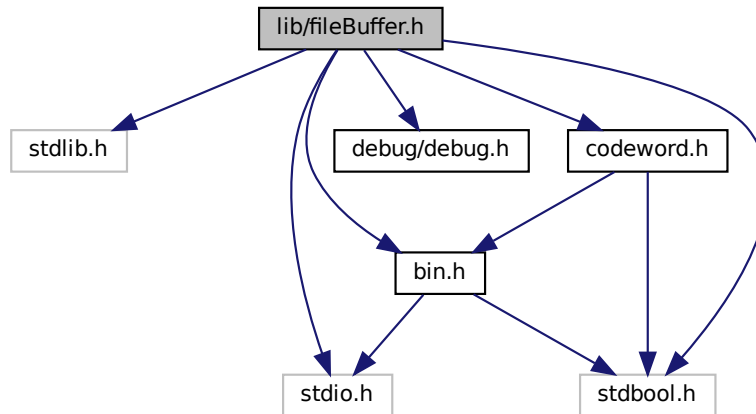
Visszatérési érték

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

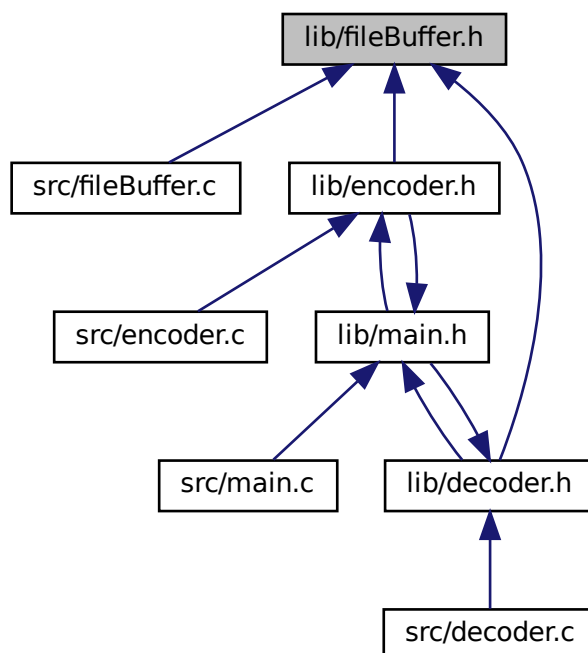
5.6. lib/fileBuffer.h fájlreferencia

```
#include <stdlib.h>
#include <stdbool.h>
#include <stdio.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](#) (const char *path)
Készít egy bitek olvasására alkalmas puffert.
- [OutputFileBuffer buff_createOutputFileBuffer](#) (const char *path)
Készít egy bitek írására alkalmas puffert.
- void [buff_destroyInputFileBuffer](#) ([InputFileBuffer](#) buffer)
Bezárja a puffer által megnyitott fájlt, és felszabadítja az az által lefoglalt memóriát.
- void [buff_destroyOutputFileBuffer](#) ([OutputFileBuffer](#) buffer)
Bezárja a puffer által megnyitott fájlt, és felszabadítja az az által lefoglalt memóriát.
- void [buff_rewind](#) ([InputFileBuffer](#) buffer)
A fájl újboli olvasására készíti fel a puffer.
- bool [buff_writeBits](#) ([OutputFileBuffer](#) buff, Bits bit)
Egy fájlba ír biteket.
- bool [buff_writeBit](#) ([OutputFileBuffer](#) buff, Bits bit)

- Egy fájlba ír 1 bitet.*
- bool `buff_writeChar` (`OutputFileBuffer` buff, `uchar` val)
- Egy fájlba ír 1 karaktert.*
- bool `buff_writeInt` (`OutputFileBuffer` buff, int val)
- Egy fájlba ír 1 egész számot.*
- bool `buff_flush` (`OutputFileBuffer` buff)
- Beírja a fájlba a puffer tartalmát, 0val kiegészítve.*
- Bits `buff_readBit` (`InputFileBuffer` buff)
- Egy fájlból olvas 1 bitet.*
- Bits `buff_readBits` (`InputFileBuffer` buff, int bitCount)
- Egy fájlból olvas bitCount darab bitet.*
- Bits `buff_readChar` (`InputFileBuffer` buff)
- Egy fájlból olvas 1 karaktert.*
- Bits `buff_readInt` (`InputFileBuffer` buff)
- Egy fájlból olvas 1 egész számot.*

5.6.1. Függvények dokumentációja

5.6.1.1. buff_createInputFileBuffer()

```
InputFileBuffer buff_createInputFileBuffer (
    const char * path )
```

Készít egy bitek olvasására alkalmas puffert.

Paraméterek

<i>path</i>	A fájl elérési útvonala
-------------	-------------------------

Visszatérési érték

Az elkészített puffer

5.6.1.2. buff_createOutputFileBuffer()

```
OutputFileBuffer buff_createOutputFileBuffer (
    const char * path )
```

Készít egy bitek írására alkalmas puffert.

Paraméterek

<i>path</i>	A fájl elérési útvonala
-------------	-------------------------

Visszatérési érték

Az elkészített puffer

5.6.1.3. buff_destroyInputFileBuffer()

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

Bezárja a puffer által megnyitott fájlt, és felszabadítja az az által lefoglalt memóriát.

Paraméterek

<i>buffer</i>	A felszabadítandó puffer
---------------	--------------------------

5.6.1.4. buff_destroyOutputFileBuffer()

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

Bezárja a puffer által megnyitott fájlt, és felszabadítja az az által lefoglalt memóriát.

Paraméterek

<i>buffer</i>	A felszabadítandó puffer
---------------	--------------------------

5.6.1.5. buff_flush()

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

Beírja a fájlba a puffer tartalmát, 0val kiegészítve.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
-------------	------------------------

Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.6.1.6. buff_readBit()

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

Egy fájlból olvas 1 bitet.

Paraméterek

<i>buff</i>	A puffer, amiből olvasunk
-------------	---------------------------

Visszatérési érték

NULLBIT, ha EOF vagy fájl olvasási hiba lépett fell, különben az olvasott bit

5.6.1.7. buff_readBits()

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

Egy fájlból olvas *bitCount* darab bitet.

Paraméterek

<i>buff</i>	A puffer, amiből olvasunk
<i>bitCount</i>	Hány darab bitet olvassunk

Visszatérési érték

A beolvasott bitsorozat

5.6.1.8. buff_readChar()

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

Egy fájlból olvas 1 karaktert.

Paraméterek

<i>buff</i>	A puffer, amiből olvasunk
-------------	---------------------------

Visszatérési érték

A beolvasott karakter

5.6.1.9. buff_readInt()

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

Egy fájlból olvas 1 egész számot.

Paraméterek

<i>buff</i>	A puffer, amiből olvasunk
-------------	---------------------------

Visszatérési érték

A beolvasott szám

5.6.1.10. buff_rewind()

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

A fájl újboli olvasására készíti fel a puffer.

Paraméterek

<i>buffer</i>	A visszahízandó puffer
---------------	------------------------

5.6.1.11. buff_writeBit()

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

Egy fájlba ír 1 bitet.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
<i>bit</i>	A beírandó bit. Hogyha a bitsorozat hossza nem 1, akkor a

Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.6.1.12. buff_writeBits()

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

Egy fájlba ír biteket.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
<i>bit</i>	Egy tetszőleges hosszúságú bitsorozat

Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.6.1.13. buff_writeChar()

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

Egy fájlba ír 1 karaktert.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
<i>val</i>	A beírandó érték

Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.6.1.14. buff_writeln()

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

Egy fájlba ír 1 egész számot.

Paraméterek

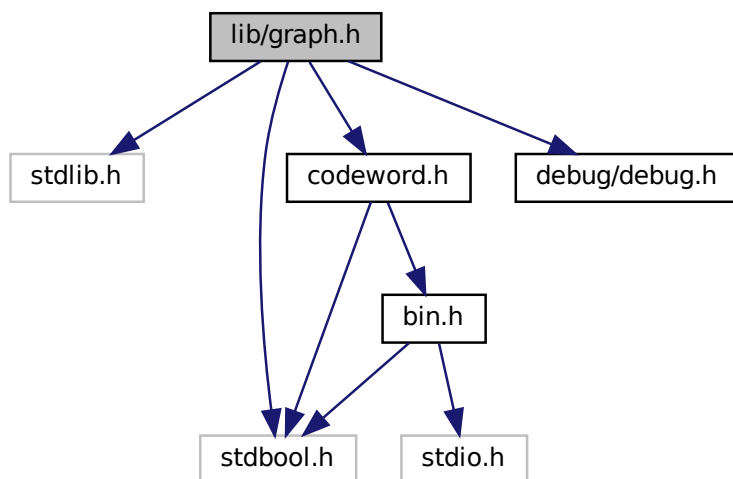
<i>buff</i>	A puffer, melybe írunk
<i>val</i>	A beírandó érték

Visszatérési érték

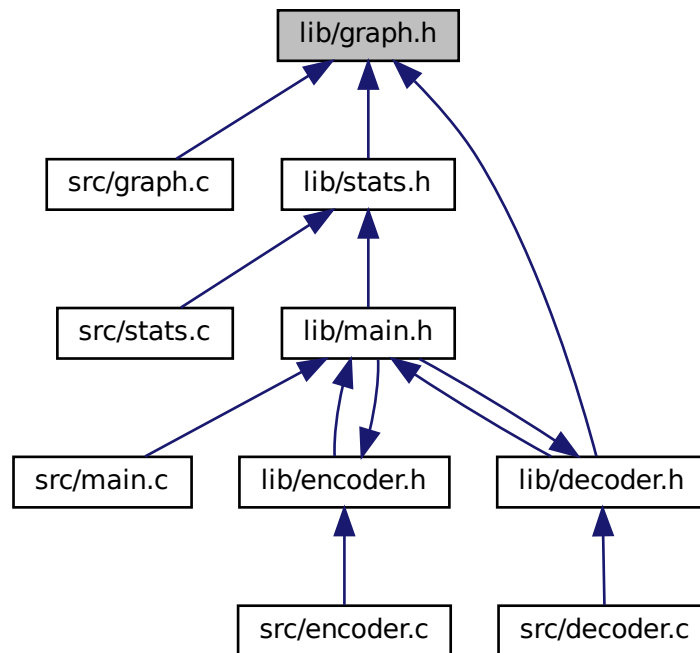
'false', ha sikeres a művelet, különben 'true'

5.7. lib/graph.h fájlreferencia

```
#include <stdlib.h>
#include <stdbool.h>
#include "codeword.h"
#include "debug/debug.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)
Megszámolja egy bináris fagráf leveleinek számát.
- void [freeTree](#) ([Node](#) *root)
Rekurzívan felszabadít egy fagráft.

5.7.1. Függvények dokumentációja

5.7.1.1. freeTree()

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

Rekurzívan felszabadít egy fagráft.

Paraméterek

<i>root</i>	a felszabadítandó gráf gyökere
-------------	--------------------------------

5.7.1.2. graph_countLeaves()

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

Megszámolja egy bináris fagráf leveleinek számát.

Paraméterek

<i>root</i>	A fagráf gyökere
-------------	------------------

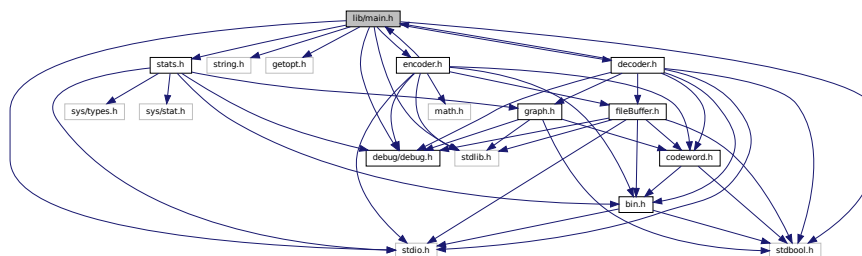
Visszatérési érték

A fagráf leveleinek száma

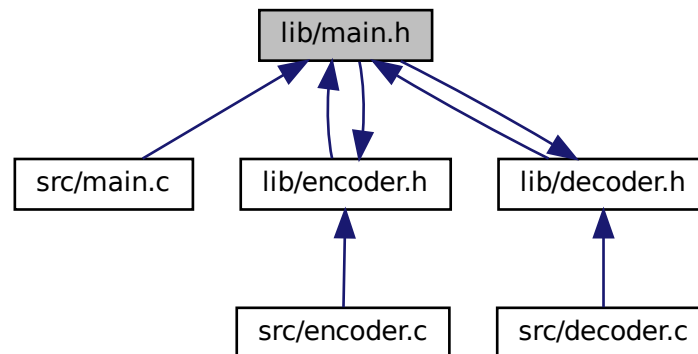
5.8. lib/main.h fájlreferencia

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

5.8.1. Enumerációk dokumentációja

5.8.1.1. MODE

```
enum MODE
```

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

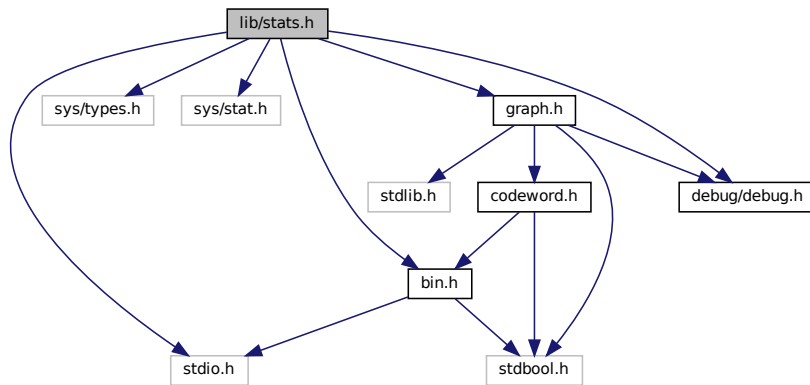
Enumeráció-értékek

ENCODE	
DECODE	
UNSET	

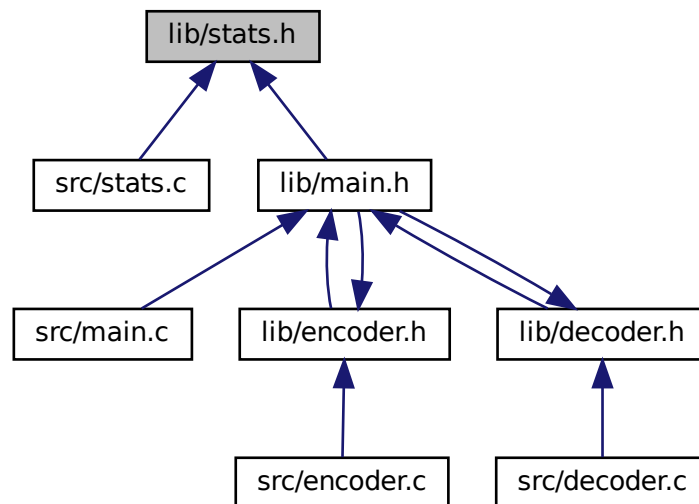
5.9. lib/stats.h fájltreferencia

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "graph.h"
#include "debug/debug.h"
#include "bin.h"
```

A stats.h definíciós fájl függési gráfja:



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



Függvények

- void [stats_printCompression](#) (const char *src, const char *dest)

- *Kiszámolja és kiírja két fájl mérete közötti különbséget.*
- void `stats_printCodetableTree` (`Node *root`)
Egy fagráf esetén kiírja a kódolt karaktereket és azok kódját.
- void `stats_printCodetableArray` (`CodeWord array[]`, int `elements`)
Egy kódtömb esetén kiírja a kódolt karaktereket és azok kódját.
- void `stats_printCodetableStatsArray` (`CodeWord array[]`, int `elements`)
Kiírja egy kódtömb statisztikáját.

5.9.1. Függvények dokumentációja

5.9.1.1. stats_printCodetableArray()

```
void stats_printCodetableArray (
    CodeWord array[],
    int elements )
```

Egy kódtömb esetén kiírja a kódolt karaktereket és azok kódját.

Paraméterek

<i>array</i>	A kódtömb
<i>elements</i>	A tömb elemeinek száma

5.9.1.2. stats_printCodetableStatsArray()

```
void stats_printCodetableStatsArray (
    CodeWord array[],
    int elements )
```

Kiírja egy kódtömb statisztikáját.

Paraméterek

<i>array</i>	A kódtömb
<i>elements</i>	A tömb elemeinek száma

5.9.1.3. stats_printCodetableTree()

```
void stats_printCodetableTree (
    Node * root )
```

Egy fagráf esetén kiírja a kódolt karaktereket és azok kódját.

Paraméterek

<i>root</i>	A fagráf gyökere
-------------	------------------

5.9.1.4. stats_printCompression()

```
void stats_printCompression (
    const char * src,
    const char * dest )
```

Kiszámolja és kiírja két fájl mérete közötti különbséget.

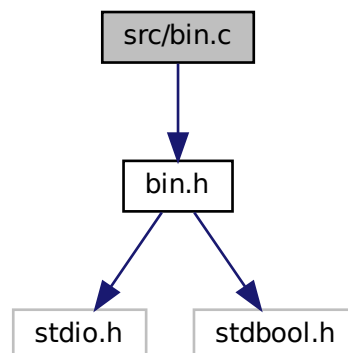
Paraméterek

<i>src</i>	A bemeneti fájl elérési útvonala
<i>dest</i>	A kimeneti fájl elérési útvonala

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 jobbról számított *n*-edik bitjét.
- `void bits_pushBits (Bits *bits, Bits append)`

- Egy bitsorozatot bővít egy másik bitsorozattal jobb oldalról.
void `bits_pushBit` (`Bits` *bits, char b)
- Egy bitsorozatot bővít egy *b* bittel.
`Bits` `bits_popBit` (`Bits` *bits)
- Egy bitsorozatból adja vissza a legkisebb helyiértéken álló bitet, majd azt eltávolítja.
`Bits` `bits_popBits` (`Bits` *bits, int length)
- Egy bitsorozatból adja vissza a legkisebb helyiértéktől számolva *length* bitet, majd azokat eltávolítja.
void `bits_cpy` (`Bits` src, `Bits` *dest)
- void `bits_print` (`Bits` bits)
Kiír egy bitsorozatot ASCII 0 és 1 karakterekkel.
- bool `bits_equ` (`Bits` b1, `Bits` b2)
Összehasonlít két bitsorozatot.
- bool `bits_isNullbit` (`Bits` b)
Megmondja, hogy egy adott bitsorozat értelmes-e.

5.10.1. Függvények dokumentációja

5.10.1.1. `bits_cpy()`

```
void bits_cpy (
    Bits src,
    Bits * dest )
```

5.10.1.2. `bits_equ()`

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

Összehasonlít két bitsorozatot.

Paraméterek

<i>b1</i>	Az összehasonlítandó bitsorozat
<i>b2</i>	Az összehasonlítandó bitsorozat

Visszatérési érték

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

5.10.1.3. bits_isNullbit()

```
bool bits_isNullbit (
    Bits b )
```

Megmondja, hogy egy adott bitsorozat értelmes-e.

Paraméterek

<i>b</i>	A vizsgálandó bitsorozat
----------	--------------------------

Visszatérési érték

igaz, hogyha a bitsorozat hossza 0, különben hamis

5.10.1.4. bits_popBit()

```
Bits bits_popBit (
    Bits * bits )
```

Egy bitsorozatból adja vissza a legkisebb helyiértéken álló bitet, majd azt eltávolítja.

Paraméterek

<i>bits</i>	A bitsorozat, melyből kivesszük a bitet
-------------	---

Visszatérési érték

5.10.1.5. bits_popBits()

```
Bits bits_popBits (
    Bits * bits,
    int length )
```

Egy bitsorozatból adja vissza a legkisebb helyiértéktől számolva *length* bitet, majd azokat eltávolítja.

Paraméterek

<i>bits</i>	A bitsorozat, melyből kivesszük a biteket
<i>length</i>	A kivett bitek száma

Visszatérési érték

5.10.1.6. bits_print()

```
void bits_print (
    Bits bits )
```

Kiír egy bitsorozatot ASCII 0 és 1 karakterekkel.

Paraméterek

<i>bits</i>	A kiírandó bitsorozat
-------------	-----------------------

5.10.1.7. bits_pushBit()

```
void bits_pushBit (
    Bits * bits,
    char b )
```

Egy bitsorozatot bővít egy *b* bittel.

Paraméterek

<i>bits</i>	A bővítendő bitsorozat
<i>b</i>	a hozzáadott bit

5.10.1.8. bits_pushBits()

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

Egy bitsorozatot bővít egy másik bitsorozattal jobb oldalról.

Paraméterek

<i>bits</i>	A bővítendő bitsorozat
<i>append</i>	A hozzáfűzendő bitsorozat

5.10.1.9. getBitFromRight()

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

Adott bitsorozatnak megadja a jobbról számított n -edik bitjét.

Paraméterek

<i>bits</i>	A bitsorozat, melyből kiválasztjuk a bitet
<i>n</i>	Jobbról számítva hányadik bit

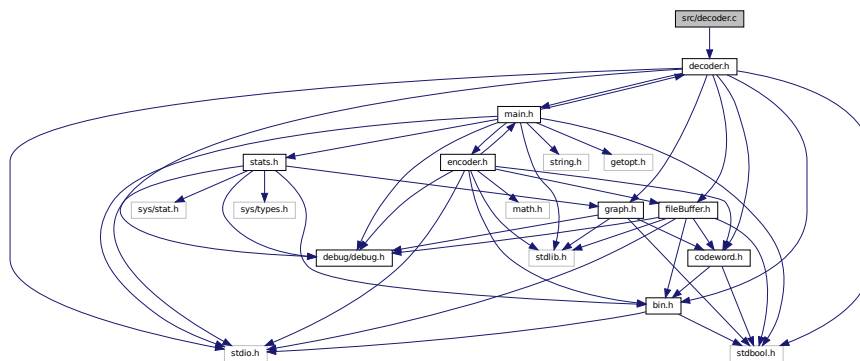
Visszatérési érték

A keresett bit

5.11. 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 meghatározott úton továbbhaladni. Ha az nem létezik, létrehozza azt.
- void **displayTable** (Node *root, Bits *_path)
- void **freeTree** (Node *root)
Rekurzívan felszabadít egy fagráfot.
- int **decode** (commandLineArguments args)
Dekódol egy Shannon-Fano algoritmussal kódolt fájlt.

5.11.1. Függvények dokumentációja

5.11.1.1. 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

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

5.11.1.2. createNodeIfNotexists()

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

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

Paraméterek

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

Visszatérési érték

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

5.11.1.3. decode()

```
int decode (
    CommandLineArguments args )
```

Dekódol egy Shannon-Fano algoritmussal kódolt fájlt.

Paraméterek

<i>args</i>	Parancssori bemenet, amely a dekódolás folyamatát módosítja
-------------	---

Visszatérési érték

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

5.11.1.4. displayTable()

```
void displayTable (
    Node * root,
    Bits * _path )
```

5.11.1.5. freeTree()

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

Rekurzívan felszabadít egy fagráfot.

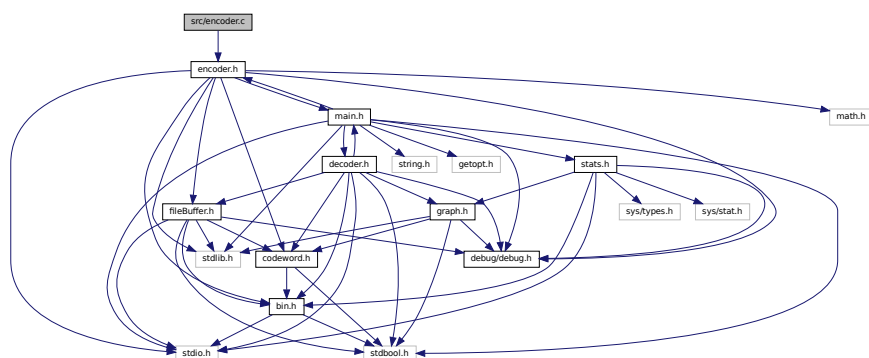
Paraméterek

<i>root</i>	a felszabadítandó gráf gyökere
-------------	--------------------------------

5.12. 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 összehasonlítja 2 frekvenciával rendelkező karakterkódolást.
- int `compare_by_bitlength` (const void *a, const void *b)
Kódolásuk hossza alapján összehasonlítja 2 rendelkező karakterkódolást.
- int `encode` (`commandLineArguments` args)
Kódol Shanon-Fano algoritmus alkalmazásával egy fájlt.

5.12.1. Függvények dokumentációja

5.12.1.1. codewordToBits()

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

Megkeresi code tömbben, find karaktert.

Paraméterek

<i>code</i>	A kódtömb
<i>codesLength</i>	A kódtömb hossza
<i>find</i>	A keresett karakter

Visszatérési érték

A keresett kódolás vagy NULLBIT

5.12.1.2. compare_by_bitlength()

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

Kódolásuk hossza alapján összehasonlítja 2 rendelkező karakterkódolást.

Paraméterek

<i>a</i>	Az összehasonlítandó karakterkódolás
<i>b</i>	Az összehasonlítandó karakterkódolás

Visszatérési érték

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

5.12.1.3. compare_by_freq()

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

Frekvenciájuk alapján összehasonlítja 2 frekvenciával rendelkező karakterkódolást.

Paraméterek

<i>a</i>	Az összehasonlítandó karakterkódolás
<i>b</i>	Az összehasonlítandó karakterkódolás

Visszatérési érték

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

5.12.1.4. encode()

```
int encode (
    CommandLineArguments args )
```

Kódol Shannon-Fano algoritmus alkalmazásával egy fájl.

Paraméterek

<i>args</i>	Parancssori bemenet, amely a dekódolás folyamatát módosítja
-------------	---

Visszatérési érték

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

5.12.1.5. setCodeWord()

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

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

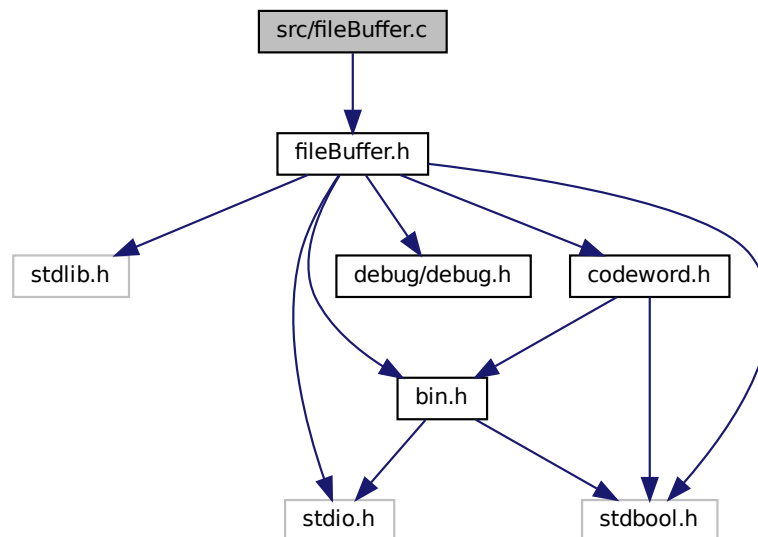
Paraméterek

<i>codes</i>	A kódtömb
<i>i</i>	A tömb kezdeti indexe (inkluzív)
<i>j</i>	A tömb vgső indexe (inkluzív)

5.13. src/fileBuffer.c fájlreferencia

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

A fileBuffer.c definíciós fájl függési gráfja:



Függvények

- [InputFileBuffer buff_createInputFileBuffer](#) (const char *path)
Készít egy bitek olvasására alkalmas puffert.
- [OutputFileBuffer buff_createOutputFileBuffer](#) (const char *path)
Készít egy bitek írására alkalmas puffert.
- void [buff_destroyInputFileBuffer](#) (InputFileBuffer buffer)
Bezárja a puffer által megnyitott fájlt, és felszabadítja az az által lefoglalt memóriát.
- void [buff_destroyOutputFileBuffer](#) (OutputFileBuffer buffer)
Bezárja a puffer által megnyitott fájlt, és felszabadítja az az által lefoglalt memóriát.
- void [buff_rewind](#) (InputFileBuffer buffer)
A fájl újboli olvasására készíti fel a puffer.
- bool [buff_writeBits](#) (OutputFileBuffer buff, Bits bit)
Egy fájlba ír biteket.
- bool [buff_writeBit](#) (OutputFileBuffer buff, Bits bit)

- Egy fájlba ír 1 bitet.*
- bool `buff_writeChar` (`OutputFileBuffer` buff, `uchar` val)
- Egy fájlba ír 1 karaktert.*
- bool `buff_writeInt` (`OutputFileBuffer` buff, int val)
- Egy fájlba ír 1 egész számot.*
- bool `buff_flush` (`OutputFileBuffer` buff)
- Beírja a fájlba a puffer tartalmát, 0val kiegészítve.*
- Bits `buff_readBit` (`InputFileBuffer` buff)
- Egy fájlból olvas 1 bitet.*
- Bits `buff_readBits` (`InputFileBuffer` buff, int bitCount)
- Egy fájlból olvas bitCount darab bitet.*
- Bits `buff_readChar` (`InputFileBuffer` buff)
- Egy fájlból olvas 1 karaktert.*
- Bits `buff_readInt` (`InputFileBuffer` buff)
- Egy fájlból olvas 1 egész számot.*

5.13.1. Függvények dokumentációja

5.13.1.1. buff_createInputFileBuffer()

```
InputFileBuffer buff_createInputFileBuffer (
    const char * path )
```

Készít egy bitek olvasására alkalmas puffert.

Paraméterek

<i>path</i>	A fájl elérési útvonala
-------------	-------------------------

Visszatérési érték

Az elkészített puffer

5.13.1.2. buff_createOutputFileBuffer()

```
OutputFileBuffer buff_createOutputFileBuffer (
    const char * path )
```

Készít egy bitek írására alkalmas puffert.

Paraméterek

<i>path</i>	A fájl elérési útvonala
-------------	-------------------------

Visszatérési érték

Az elkészített puffer

5.13.1.3. buff_destroyInputFileBuffer()

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

Bezárja a puffer által megnyitott fájlt, és felszabadítja az az által lefoglalt memóriát.

Paraméterek

<i>buffer</i>	A felszabadítandó puffer
---------------	--------------------------

5.13.1.4. buff_destroyOutputFileBuffer()

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

Bezárja a puffer által megnyitott fájlt, és felszabadítja az az által lefoglalt memóriát.

Paraméterek

<i>buffer</i>	A felszabadítandó puffer
---------------	--------------------------

5.13.1.5. buff_flush()

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

Beírja a fájlba a puffer tartalmát, Oval kiegészítve.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
-------------	------------------------

Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.13.1.6. buff_readBit()

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

Egy fálból olvas 1 bitet.

Paraméterek

<i>buff</i>	A puffer, amiből olvasunk
-------------	---------------------------

Visszatérési érték

NULLBIT, ha EOF vagy fájl olvasási hiba lépett fell, különben az olvasott bit

5.13.1.7. buff_readBits()

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

Egy fálból olvas *bitCount* darab bitet.

Paraméterek

<i>buff</i>	A puffer, amiből olvasunk
<i>bitCount</i>	Hány darab bitet olvassunk

Visszatérési érték

A beolvasott bitsorozat

5.13.1.8. buff_readChar()

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

Egy fálból olvas 1 karaktert.

Paraméterek

<i>buff</i>	A puffer, amiből olvasunk
-------------	---------------------------

Visszatérési érték

A beolvasott karakter

5.13.1.9. buff_readInt()

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

Egy fájlból olvas 1 egész számot.

Paraméterek

<i>buff</i>	A puffer, amiből olvasunk
-------------	---------------------------

Visszatérési érték

A beolvasott szám

5.13.1.10. buff_rewind()

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

A fájl újboli olvasására készíti fel a puffer.

Paraméterek

<i>buffer</i>	A visszahízandó puffer
---------------	------------------------

5.13.1.11. buff_writeBit()

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

Egy fájlba ír 1 bitet.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
<i>bit</i>	A beírandó bit. Hogyha a bitsorozat hossza nem 1, akkor a

Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.13.1.12. buff_writeBits()

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

Egy fájlba ír biteket.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
<i>bit</i>	Egy tetszőleges hosszúságú bitsorozat

Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.13.1.13. buff_writeChar()

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

Egy fájlba ír 1 karaktert.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
<i>val</i>	A beírandó érték

Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.13.1.14. buff_writeln()

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

Egy fájlba ír 1 egész számot.

Paraméterek

<i>buff</i>	A puffer, melybe írunk
<i>val</i>	A beírandó érték

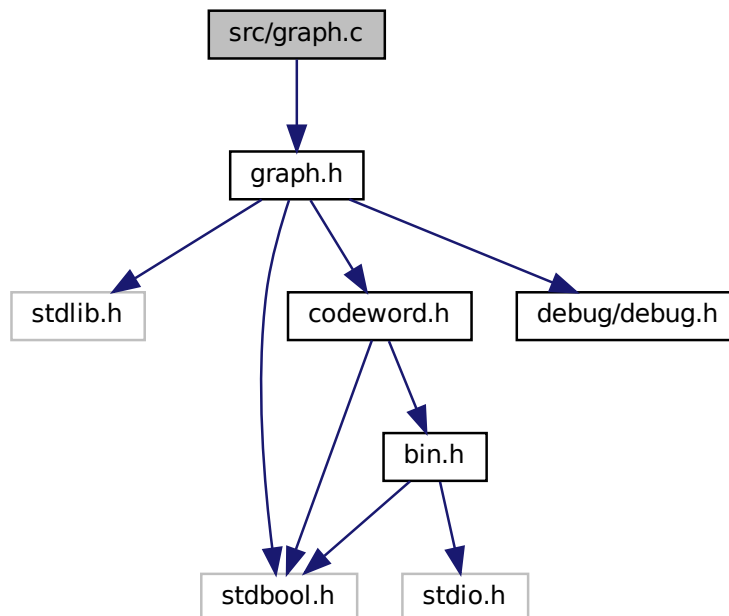
Visszatérési érték

'false', ha sikeres a művelet, különben 'true'

5.14. src/graph.c fájlreferencia

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

A graph.c definíciós fájl függési gráfja:



Függvények

- int `graph_countLeaves` (`Node *root`)
Megszámolja egy bináris fagráf leveleinek számát.
- void `freeTree` (`Node *root`)
Rekurzívan felszabadít egy fagráft.

5.14.1. Függvények dokumentációja

5.14.1.1. freeTree()

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

Rekurzívan felszabadít egy fagráft.

Paraméterek

<i>root</i>	a felszabadítandó gráf gyökere
-------------	--------------------------------

5.14.1.2. graph_countLeaves()

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

Megszámolja egy bináris fagráf leveleinek számát.

Paraméterek

<i>root</i>	A fagráf gyökere
-------------	------------------

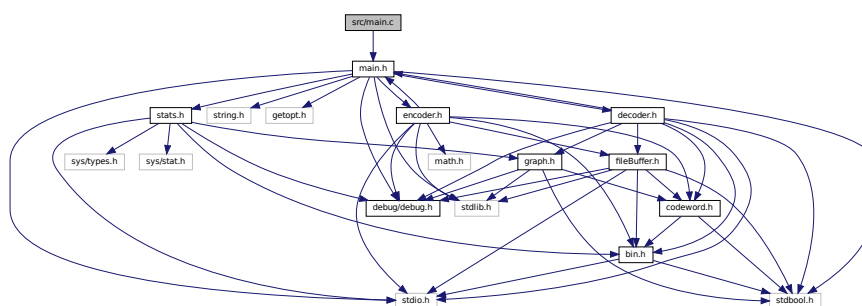
Visszatérési érték

A fagráf leveleinek száma

5.15. src/main.c fájlreferencia

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

A main.c definíciós fájl függési gráfja:



Függvények

- void `printHelp` ()
Kiírja a standard outputra a program elfogadott paramétereit és kapcsolókat.
- int `parseCLA` (int argc, char **argv, `commandLineArguments` *args)
A parancssori argumentumokat állítja be args paramétereként és alakítja át azokat `commandLineArguments` kapcsolókat feldolgozó függvény.
- int `main` (int argc, char **argv)
A program belépési pontja.

5.15.1. Függvények dokumentációja

5.15.1.1. main()

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

A program belépési pontja.

Paraméterek

<code>argc</code>	argv parancssori argumentumok hossza
<code>argv</code>	parancssori argumentumok

Visszatérési érték

A program futásának eredménye. 0 = Rendeltetésszerű futás, bármilyen más esetben sikertelen a program futása

5.15.1.2. parseCLA()

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

A parancssori argumentumokat állítja be args paramétereként és alakítja át azokat `commandLineArguments` kapcsolókat feldolgozó függvény.

Paraméterek

<code>argc</code>	argc hossza
<code>argv</code>	A programnak átadott parancssori paraméterek
<code>args</code>	A függvény ide tölti be a feldolgozott kapcsolókat. args értéke megváltozhat annak ellenére, hogy a függvény nem 0 kimenettel tér vissza

Visszatérési érték

0, ha sikeres volt az argumentumok elemzése, és minden kötelező paraméter meg lett adva különben ettől eltérő érték

5.15.1.3. printHelp()

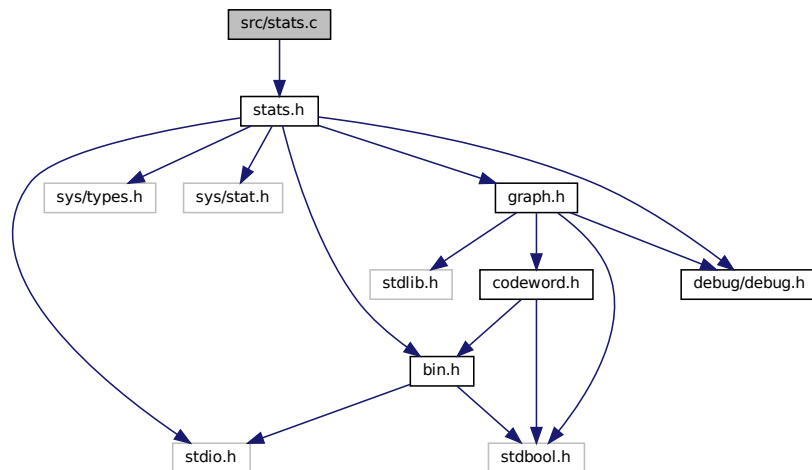
```
void printHelp ( )
```

Kiírja a standard outputra a program elfogadott paramétereit és kapcsolókat.

5.16. src/stats.c fájlreferencia

```
#include "stats.h"
```

A stats.c definíciós fájl függési gráfja:



Függvények

- void [printCodetableTreeRec](#) (Node *root, Bits *_path, int *_count, [CodeWord](#) *_min, [CodeWord](#) *_max, float *_avg)
Rekurzívan kiírja egy bináris fa gráf esetén a kódolt karaktereket és azok kódját. A [stats_printCodetableTree](#) segéd-függvénye.
- long [calcFileSize](#) (const char *file)
Egy adott fájl méretét adja meg.
- void [prettyPrintChar](#) (uchar c)
Egy karakter beszédesebb formáját írja ki a standard kimenetre.
- void [stats_printCompression](#) (const char *src, const char *dest)
Kiszámolja és kiírja két fájl mérete közötti különbséget.
- void [stats_printCodetableTree](#) (Node *root)
Egy fa gráf esetén kiírja a kódolt karaktereket és azok kódját.
- void [stats_printCodetableArray](#) (CodeWord array[], int elements)
Egy kódtömb esetén kiírja a kódolt karaktereket és azok kódját.
- void [stats_printCodetableStatsArray](#) (CodeWord array[], int elements)
Kiírja egy kódtömb statisztikáját.

5.16.1. Függvények dokumentációja

5.16.1.1. calcFileSize()

```
long calcFileSize (
    const char * file )
```

Egy adott fájl méretét adja meg.

Paraméterek

<i>file</i>	A fájl elérési útvonala
-------------	-------------------------

Visszatérési érték

A fájl mérete bájtokban

5.16.1.2. prettyPrintChar()

```
void prettyPrintChar (
    uchar c )
```

Egy karakter beszédesebb formáját írja ki a standard kimenetre.

Paraméterek

<i>c</i>	A kiírandó karakter
----------	---------------------

5.16.1.3. printCodetableTreeRec()

```
void printCodetableTreeRec (
    Node * root,
    Bits * _path,
    int * _count,
    CodeWord * _min,
    CodeWord * _max,
    float * _avg )
```

Rekurzívan kiírja egy bináris fastruktúra esetén a kódolt karaktereket és azok kódját. A [stats_printCodetableTree](#) segéd-függvénye.

Paraméterek

<i>root</i>	A fagráf gyökere
<i>_path</i>	A fagráf jelenlegi elérési útvonala
<i>_count</i>	A fagráf beállított elemeinek száma
<i>_min</i>	A fagráf legrövidebben kódolt karaktere
<i>_max</i>	A fagráf leghosszabban kódolt karaktere
<i>_avg</i>	A kódolt karakterek hosszának átlaga

5.16.1.4. stats_printCodetableArray()

```
void stats_printCodetableArray (
    CodeWord array[],
    int elements )
```

Egy kódtömb esetén kiírja a kódolt karaktereket és azok kódját.

Paraméterek

<i>array</i>	A kódtömb
<i>elements</i>	A tömb elemeinek száma

5.16.1.5. stats_printCodetableStatsArray()

```
void stats_printCodetableStatsArray (
    CodeWord array[],
    int elements )
```

Kiírja egy kódtömb statisztikáját.

Paraméterek

<i>array</i>	A kódtömb
<i>elements</i>	A tömb elemeinek száma

5.16.1.6. stats_printCodetableTree()

```
void stats_printCodetableTree (
    Node * root )
```

Egy fagráf esetén kiírja a kódolt karaktereket és azok kódját.

Paraméterek

<i>root</i>	A fagráf gyökere
-------------	------------------

5.16.1.7. stats_printCompression()

```
void stats_printCompression (
    const char * src,
    const char * dest )
```

Kiszámolja és kiírja két fájl mérete közötti különbséget.

Paraméterek

<i>src</i>	A bemeneti fájl elérési útvonala
<i>dest</i>	A kimeneti fájl elérési útvonala

Meta

5.17. Források, felhasznált irodalom

- Wayback Machine: C. E. Shannon, „A Mathematical Theory of Communication”, 1948 (<https://web.archive.org/web/1998071501labs.com/cm/ms/what/shannonday/shannon1948.pdf>)
- Halley's Comet software: Robert M. Fano, „The Transmission 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/>)