

Projet d'algorithmique - CATALA Alexandre - DI GIOVANNI Celian - SOUBRY
Sophie

Generated by Doxygen 1.9.8

1	Projet d'algorithmique - CATALA Alexandre - DI GIOVANNI Celian - SOUBRY Sophie	1
1.1	Utilitaire de traitement de photo astronomiques	1
1.2	Contenu du projet	1
1.3	Compilation et Exécution	1
1.4	Exemple d'Utilisation	2
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	FitStruct Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Field Documentation	7
4.1.2.1	data	7
4.1.2.2	header_fichier	7
4.2	Header Struct Reference	8
4.2.1	Detailed Description	8
4.2.2	Field Documentation	8
4.2.2.1	BITPIX	8
4.2.2.2	BSCALE	8
4.2.2.3	BZERO	8
4.2.2.4	NAXIS	9
4.2.2.5	NAXIS1	9
4.2.2.6	NAXIS2	9
4.2.2.7	NAXIS3	9
4.2.2.8	SIMPLE	9
5	File Documentation	11
5.1	AfficheCSV.py	11
5.2	include/ecriture.h File Reference	11
5.2.1	Detailed Description	12
5.2.2	Function Documentation	12
5.2.2.1	ecrire_fit_file()	12
5.2.2.2	ecrire_pixels_csv()	12
5.3	ecriture.h	13
5.4	include/fitstruct.h File Reference	13
5.4.1	Detailed Description	14
5.4.2	Function Documentation	14
5.4.2.1	construct_fitstruct()	14
5.5	fitstruct.h	14
5.6	include/header.h File Reference	15

5.6.1 Detailed Description	16
5.6.2 Macro Definition Documentation	16
5.6.2.1 LONGUEUR_LIGNES_HEADER	16
5.6.2.2 NB_CLES_VALIDES	16
5.6.2.3 NB_LIGNES_HEADER	16
5.6.2.4 OCTETS_HEADER	16
5.6.3 Function Documentation	16
5.6.3.1 afficher_header()	16
5.6.3.2 cle_valide()	17
5.6.3.3 construct_header()	17
5.6.3.4 process_header()	17
5.7 header.h	18
5.8 include/lecture.h File Reference	18
5.8.1 Detailed Description	19
5.8.2 Function Documentation	19
5.8.2.1 lire_donnees_header()	19
5.8.2.2 lire_donnees_image()	19
5.8.2.3 ouvrir_fichier()	20
5.9 lecture.h	20
5.10 include/memoire.h File Reference	20
5.10.1 Detailed Description	21
5.10.2 Function Documentation	21
5.10.2.1 allouer_malloc()	21
5.11 memoire.h	22
5.12 include/menu.h File Reference	22
5.12.1 Detailed Description	22
5.12.2 Function Documentation	23
5.12.2.1 menu()	23
5.13 menu.h	23
5.14 include/operation.h File Reference	23
5.14.1 Detailed Description	24
5.14.2 Function Documentation	24
5.14.2.1 afficher_premieres_valeurs()	24
5.14.2.2 diviser_image()	24
5.14.2.3 headers_compatible()	25
5.14.2.4 moyenne_image()	25
5.14.2.5 somme_image()	26
5.14.2.6 soustraire_image()	26
5.15 operation.h	26
5.16 src/ecriture.c File Reference	27
5.16.1 Detailed Description	27
5.16.2 Function Documentation	27

5.16.2.1 <code>ecrire_fit_file()</code>	27
5.16.2.2 <code>ecrire_pixels_csv()</code>	28
5.17 <code>ecriture.c</code>	28
5.18 <code>src/fitstruct.c</code> File Reference	29
5.18.1 Detailed Description	29
5.18.2 Function Documentation	29
5.18.2.1 <code>construct_fitstruct()</code>	29
5.19 <code>fitstruct.c</code>	30
5.20 <code>src/header.c</code> File Reference	30
5.20.1 Detailed Description	31
5.20.2 Function Documentation	31
5.20.2.1 <code>afficher_header()</code>	31
5.20.2.2 <code>cle_valide()</code>	31
5.20.2.3 <code>construct_header()</code>	32
5.20.2.4 <code>process_header()</code>	32
5.21 <code>header.c</code>	32
5.22 <code>src/lecture.c</code> File Reference	34
5.22.1 Detailed Description	34
5.22.2 Function Documentation	35
5.22.2.1 <code>lire_donnees_header()</code>	35
5.22.2.2 <code>lire_donnees_image()</code>	35
5.22.2.3 <code>ouvrir_fichier()</code>	35
5.23 <code>lecture.c</code>	36
5.24 <code>src/main.c</code> File Reference	36
5.24.1 Detailed Description	37
5.24.2 Function Documentation	37
5.24.2.1 <code>main()</code>	37
5.25 <code>main.c</code>	38
5.26 <code>src/memoire.c</code> File Reference	39
5.26.1 Detailed Description	39
5.26.2 Function Documentation	39
5.26.2.1 <code>allouer_malloc()</code>	39
5.27 <code>memoire.c</code>	40
5.28 <code>src/menu.c</code> File Reference	40
5.28.1 Detailed Description	40
5.28.2 Function Documentation	41
5.28.2.1 <code>menu()</code>	41
5.29 <code>menu.c</code>	41
5.30 <code>operation.c</code>	42
Index	45

Chapter 1

Projet d'algorithmique - CATALA Alexandre - DI GIOVANNI Celian - SOUBRY Sophie

1.1 Utilitaire de traitement de photo astronomiques

1.2 Contenu du projet

1. **src** : Ce répertoire contient le code source de l'application.

- `main.c` : Fichier principal contenant le menu interactif.
- `operation.c` : Fichier contenant les opérations à effectuer sur les fichier .fit.
- ...

1. **include** : Ce répertoire contient les fichiers d'en-tête.

- `fitstruct.h` : Définition de la structure `FitStruct`.
- `header.h` : Définition de la structure `Header`.
- ...

1. **Images** : Ce répertoire contient les images FITS utilisées pour les tests.

2. **Documentation** : Ce répertoire contient la documentation du projet.

3. **build** : Ce répertoire contient les fichiers générés lors de la compilation.

1.3 Compilation et Exécution

1. Cloner le dépôt

2. Lancer la commande `make clean && make run` dans votre terminal en étant à la racine du projet

1.4 Exemple d'Utilisation

1. Sélectionnez l'option dans le menu interactif.
2. Suivez les instructions pour fournir les fichiers FITS nécessaires.
3. Obtenez les résultats dans votre terminal.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

FitStruct

[FitStruct](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un fichier à savoir les données du header et les données de l'image 7

Header

Structure du [Header](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un header 8

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

AfficheCSV.py	11
include/ ecriture.h	
Fichier ecriture.h contenant l'ensemble des prototypes et des déclarations pour les procédures d'écriture	11
include/ fitstruct.h	
Fichier fitstruct.h contenant l'ensemble des prototypes et des déclarations pour les fitstruct	13
include/ header.h	
Fichier header.h contenant l'ensemble des prototypes et des déclarations pour les header	15
include/ lecture.h	
Fichier lecture.h contenant l'ensemble des prototypes et des déclarations pour les lecture de fichier et de header	18
include/ memoire.h	
Fichier memoire.h contenant l'ensemble des prototypes et des déclarations pour les procédures de gestion de mémoire	20
include/ menu.h	
Fichier menu.h contenant l'ensemble des prototypes et des déclarations pour la procédure menu	22
include/ operation.h	
Fichier operation.h contenant l'ensemble des prototypes et des déclarations pour les procédures d'opérations	23
src/ ecriture.c	
Fichier ecriture.c contenant l'ensemble des fonctions pour la lecture de fichier / données	27
src/ fitstruct.c	
Fichier fitstruct.c contenant l'ensemble des fonctions pour traiter les fitstruct	29
src/ header.c	
Fichier header.c contenant l'ensemble des fonctions pour traiter les header	30
src/ lecture.c	
Fichier lecture.c contenant l'ensemble des fonctions pour la lecture de fichier et de header	34
src/ main.c	
Fichier main.c contenant l'ensemble de nos procédures de tests	36
src/ memoire.c	
Fichier memoire.c contenant l'ensemble des fonctions pour les opérations concernant la mémoire	39
src/ menu.c	
Fichier menu.c contenant la fonction void menu() - Répond à l'exigence optionnelle OPT_30	40
src/ operation.c	42

Chapter 4

Data Structure Documentation

4.1 FitStruct Struct Reference

[FitStruct](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un fichier à savoir les données du header et les données de l'image.

```
#include <fitstruct.h>
```

Data Fields

- `int16_t * data`
- `Header header_fichier`

4.1.1 Detailed Description

[FitStruct](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un fichier à savoir les données du header et les données de l'image.

Definition at line [25](#) of file [fitstruct.h](#).

4.1.2 Field Documentation

4.1.2.1 data

```
int16_t* data
```

Definition at line [27](#) of file [fitstruct.h](#).

4.1.2.2 header_fichier

```
Header header_fichier
```

Definition at line [28](#) of file [fitstruct.h](#).

The documentation for this struct was generated from the following file:

- `include/fitstruct.h`

4.2 Header Struct Reference

Structure du [Header](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un header.

```
#include <header.h>
```

Data Fields

- char [SIMPLE](#)
- int [BITPIX](#)
- int [NAXIS](#)
- int [NAXIS1](#)
- int [NAXIS2](#)
- int [NAXIS3](#)
- int [BZERO](#)
- int [BSCALE](#)

4.2.1 Detailed Description

Structure du [Header](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un header.

Definition at line [28](#) of file [header.h](#).

4.2.2 Field Documentation

4.2.2.1 BITPIX

```
int BITPIX
```

Definition at line [31](#) of file [header.h](#).

4.2.2.2 BSCALE

```
int BSCALE
```

Definition at line [37](#) of file [header.h](#).

4.2.2.3 BZERO

```
int BZERO
```

Definition at line [36](#) of file [header.h](#).

4.2.2.4 NAXIS

```
int NAXIS
```

Definition at line 32 of file [header.h](#).

4.2.2.5 NAXIS1

```
int NAXIS1
```

Definition at line 33 of file [header.h](#).

4.2.2.6 NAXIS2

```
int NAXIS2
```

Definition at line 34 of file [header.h](#).

4.2.2.7 NAXIS3

```
int NAXIS3
```

Definition at line 35 of file [header.h](#).

4.2.2.8 SIMPLE

```
char SIMPLE
```

Definition at line 30 of file [header.h](#).

The documentation for this struct was generated from the following file:

- [include/header.h](#)

Chapter 5

File Documentation

5.1 AfficheCSV.py

```
00001 #!/bin/python3
00002 import csv
00003 import matplotlib.pyplot as plt
00004 import numpy as np
00005 import sys
00006
00007 print("Lecture de ", sys.argv[1]);
00008 csvReader = csv.reader(open(sys.argv[1]), delimiter=',')
00009 dataArray = np.zeros([2900, 4144], int)
00010 for i, rowText in enumerate(csvReader) :
00011     dataArray[i, :] = np.array([int(x) for x in rowText[:-1]])
00012
00013 print("Affichage");
00014 fig = plt.figure()
00015 fig.add_axes()
00016 ax = plt.gca()
00017 ax.imshow(dataArray)
00018 plt.show()
```

5.2 include/ecriture.h File Reference

Fichier [ecriture.h](#) contenant l'ensemble des prototypes et des déclarations pour les procédures d'écriture.

```
#include "fitstruct.h"
#include "lecture.h"
```

Functions

- void [ecrire_pixels_csv](#) ([FitStruct](#) fitStruct, char *nom_fichier_csv)
*Écrit les valeurs réelles d'une image dans un fichier CSV valeur_réelle = BZERO + BSCALE × valeur_enregistrée.
Répond à l'exigence **PRIM_40***
- void [ecrire_fit_file](#) ([FitStruct](#) fitStruct, char *filename)
Convertir une [FitStruct](#) en fichier .fit.

5.2.1 Detailed Description

Fichier [écriture.h](#) contenant l'ensemble des prototypes et des déclarations pour les procédures d'écriture.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [écriture.h](#).

5.2.2 Function Documentation

5.2.2.1 `ecrire_fit_file()`

```
void écrire_fit_file (
    FitStruct fitStruct,
    char * filename )
```

Convertir une [FitStruct](#) en fichier .fit.

Parameters

<i>fitStruct</i>	Structure fitStruct d'une image .fit
<i>filenameLe</i>	Le chemin du fichier .fit dans lequel on va écrire

Definition at line [47](#) of file [écriture.c](#).

5.2.2.2 `ecrire_pixels_csv()`

```
void écrire_pixels_csv (
    FitStruct fitStruct,
    char * nom_fichier_csv )
```

Écrit les valeurs réelles d'une image dans un fichier CSV $\text{valeur_réelle} = \text{BZERO} + \text{BSCALE} \times \text{valeur_enregistrée}$. Répond à l'exigence **PRIM_40**

Parameters

<i>fitStruct</i>	Structure fitStruct d'une image .fit
<i>nom_fichier_csv</i>	Le chemin du fichier .csv dans lequel on va écrire

Definition at line 21 of file [ecriture.c](#).

5.3 ecriture.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef ECRITURE_H
00012 #define ECRITURE_H
00013 #include "fitstruct.h"
00014 #include "lecture.h"
00015
00016 void ecrire_pixels_csv(FitStruct fitStruct, char *nom_fichier_csv);
00017 void ecrire_fit_file(FitStruct fitStruct, char *filename);
00018
00019 #endif // ECRITURE_H

```

5.4 include/fitstruct.h File Reference

Fichier [fitstruct.h](#) contenant l'ensemble des prototypes et des déclarations pour les fitstruct.

```

#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include "header.h"

```

Data Structures

- struct [FitStruct](#)

[FitStruct](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un fichier à savoir les données du header et les données de l'image.

Typedefs

- typedef struct [FitStruct](#) **FitStruct**

[FitStruct](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un fichier à savoir les données du header et les données de l'image.

Functions

- struct [FitStruct](#) [construct_fitstruct](#) (FILE *fichier)

On passe un fichier en paramètre et on construit sa structure [FitStruct](#). Dans cette structure on stockera les données de l'image et son header. Cette structure nous permettra de faire les opérations sur les images et l'écriture du CSV.

5.4.1 Detailed Description

Fichier [fitstruct.h](#) contenant l'ensemble des prototypes et des déclarations pour les fitstruct.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [fitstruct.h](#).

5.4.2 Function Documentation

5.4.2.1 construct_fitstruct()

```
struct FitStruct construct_fitstruct (
    FILE * fichier )
```

On passe un fichier en paramètre et on construit sa structure [FitStruct](#). Dans cette structure on stockera les données de l'image et son header. Cette structure nous permettra de faire les opérations sur les images et l'écriture du CSV.

Parameters

<i>fichier</i>	Le chemin du fichier .fit pour lequel on veut obtenir sa structure FitStruct .
----------------	--

Returns

struct [FitStruct](#)

Definition at line 21 of file [fitstruct.c](#).

5.5 fitstruct.h

[Go to the documentation of this file.](#)

```
00001
00011 #include <string.h>
```

```

00012 #include <stdlib.h>
00013 #include <math.h>
00014 #include <stdint.h>
00015 #include <stdio.h>
00016 #include "header.h"
00017
00018 #ifndef FITSTRUCT_H
00019 #define FITSTRUCT_H
00020
00025 typedef struct FitStruct
00026 {
00027     int16_t *data;
00028     Header header_fichier;
00029 } FitStruct;
00030
00031 struct FitStruct construct_fitstruct(FILE *fichier);
00032 #endif // FITSTRUCT_H

```

5.6 include/header.h File Reference

Fichier [header.h](#) contenant l'ensemble des prototypes et des déclarations pour les header.

```

#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <stdint.h>
#include <stdio.h>

```

Data Structures

- struct [Header](#)

Structure du [Header](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un header.

Macros

- #define [NB_LIGNES_HEADER](#) 36
- #define [LONGUEUR_LIGNES_HEADER](#) 80
- #define [OCTETS_HEADER](#) 2880
- #define [NB_CLES_VALIDES](#) 7

Typedefs

- typedef struct [Header](#) [Header](#)

Structure du [Header](#) d'un fichier .fit. Cette structure nous permettra de stocker les informations importantes d'un header.

Functions

- int [cle_valide](#) (char *cle)
Renvoie 1 si la clé passé en paramètre est contenue dans la liste, 0 sinon.
- void [afficher_header](#) (char header[OCTETS_HEADER])
Affiche le header passé en paramètre.
- void [process_header](#) ([Header](#) *mon_header, char packet80[LONGUEUR_LIGNES_HEADER])
Associe les valeurs contenues dans le header aux variables de la structure pour pouvoir les conserver et les utiliser ultérieurement.
- struct [Header](#) [construct_header](#) (FILE *mon_fichier)
Construit la structure du [Header](#) associée au fichier passé en paramètre. Répond a l'exigence **PRIM_30**

5.6.1 Detailed Description

Fichier [header.h](#) contenant l'ensemble des prototypes et des déclarations pour les header.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2023-12-24

Copyright

Copyright (c) 2023

Definition in file [header.h](#).

5.6.2 Macro Definition Documentation

5.6.2.1 LONGUEUR_LIGNES_HEADER

```
#define LONGUEUR_LIGNES_HEADER 80
```

Definition at line 20 of file [header.h](#).

5.6.2.2 NB_CLES_VALIDES

```
#define NB_CLES_VALIDES 7
```

Definition at line 22 of file [header.h](#).

5.6.2.3 NB_LIGNES_HEADER

```
#define NB_LIGNES_HEADER 36
```

Definition at line 19 of file [header.h](#).

5.6.2.4 OCTETS_HEADER

```
#define OCTETS_HEADER 2880
```

Definition at line 21 of file [header.h](#).

5.6.3 Function Documentation

5.6.3.1 afficher_header()

```
void afficher_header (
    char header[OCTETS_HEADER] )
```

Affiche le header passé en paramètre.

Parameters

<i>header</i>	
---------------	--

Definition at line 19 of file [header.c](#).

5.6.3.2 cle_valide()

```
int cle_valide (
    char * cle )
```

Renvoie 1 si la clé passé en paramètre est contenue dans la liste, 0 sinon.

Parameters

<i>cle</i>	
------------	--

Returns

int

Definition at line 33 of file [header.c](#).

5.6.3.3 construct_header()

```
struct Header construct_header (
    FILE * mon_fichier )
```

Construit la structure du [Header](#) associée au fichier passé en paramètre. Répond a l'exigence **PRIM_30**

Parameters

<i>mon_fichier</i>	Le chemin du fichier .fit pour lequel on veut obtenir sa structure Header .
--------------------	---

Returns

struct [Header](#)

Definition at line 119 of file [header.c](#).

5.6.3.4 process_header()

```
void process_header (
    Header * mon_header,
    char packet80[LONGUEUR_LIGNES_HEADER] )
```

Associe les valeurs contenues dans le header aux variables de la structure pour pouvoir les conserver et les utiliser ultérieurement.

Parameters

<i>mon_header</i>	
<i>packet80</i>	

Definition at line 55 of file [header.c](#).

5.7 header.h

[Go to the documentation of this file.](#)

```

00001
00011 #include <string.h>
00012 #include <stdlib.h>
00013 #include <math.h>
00014 #include <stdint.h>
00015 #include <stdio.h>
00016
00017 #ifndef HEADER_H
00018 #define HEADER_H
00019 #define NB_LIGNES_HEADER 36
00020 #define LONGUEUR_LIGNES_HEADER 80
00021 #define OCTETS_HEADER 2880
00022 #define NB_CLES_VALIDES 7
00023
00028 typedef struct Header
00029 {
00030     char SIMPLE;
00031     int BITPIX;
00032     int NAXIS;
00033     int NAXIS1;
00034     int NAXIS2;
00035     int NAXIS3;
00036     int BZERO;
00037     int BSCALE;
00038 } Header;
00039
00040 int cle_valide(char *cle);
00041 void afficher_header(char header[OCTETS_HEADER]);
00042 void process_header(Header *mon_header, char packet80[LONGUEUR_LIGNES_HEADER]);
00043 struct Header construct_header(FILE *mon_fichier);
00044
00045 #endif // HEADER_H

```

5.8 include/lecture.h File Reference

Fichier [lecture.h](#) contenant l'ensemble des prototypes et des déclarations pour les lecture de fichier et de header.

```

#include <stdio.h>
#include <stdint.h>
#include "fitstruct.h"

```

Functions

- char * [lire_donnees_header](#) (FILE *fichier)
Retourne les données du header du fichier.
- FILE * [ouvrir_fichier](#) (char *chemin_fichier, char *option)
Ouvre le fichier passé en paramètre avec son option.
- int16_t * [lire_donnees_image](#) (FILE *fichier, int naxis1, int naxis2)
Lit les données brutes de l'images en format big endian.

5.8.1 Detailed Description

Fichier [lecture.h](#) contenant l'ensemble des prototypes et des déclarations pour les lecture de fichier et de header.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2023-12-24

Copyright

Copyright (c) 2023

Definition in file [lecture.h](#).

5.8.2 Function Documentation

5.8.2.1 lire_donnees_header()

```
char * lire_donnees_header (
    FILE * fichier )
```

Retourne les données du header du fichier.

Parameters

<i>fichier</i>	Le chemin du fichier .fit pour lequel on veut lire les données du header
----------------	--

Returns

char*

Definition at line [24](#) of file [lecture.c](#).

5.8.2.2 lire_donnees_image()

```
int16_t * lire_donnees_image (
    FILE * fichier,
    int naxis1,
    int naxis2 )
```

Lit les données brutes de l'images en format big endian.

Parameters

<i>fichier</i>	Le chemin du fichier .fit pour lequel on veut lire les données
<i>naxis1</i>	La valeur NAXIS1 de ce fichier
<i>naxis2</i>	La valeur NAXIS2 de ce fichier

Returns

uint16_t*

Definition at line 60 of file [lecture.c](#).

5.8.2.3 ouvrir_fichier()

```
FILE * ouvrir_fichier (
    char * chemin_fichier,
    char * option )
```

Ouvre le fichier passé en paramètre avec son option.

Parameters

<i>chemin_fichier</i>	
<i>option</i>	

Returns

FILE*

Definition at line 40 of file [lecture.c](#).

5.9 lecture.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef LECTURE_H
00012 #define LECTURE_H
00013 #include <stdio.h>
00014 #include <stdint.h>
00015 #include "fitstruct.h"
00016
00017 char *lire_donnees_header(FILE *fichier);
00018
00019 FILE *ouvrir_fichier(char *chemin_fichier, char *option);
00020 int16_t *lire_donnees_image(FILE *fichier, int naxis1, int naxis2);
00021
00022 #endif // LECTURE_H
```

5.10 include/memoire.h File Reference

Fichier [memoire.h](#) contenant l'ensemble des prototypes et des déclarations pour les procédures de gestion de mémoire.

Functions

- void * [allouer_malloc](#) (int taille)

Permet de faire un malloc tout en prenant en compte les erreurs de pointeurs NULL.

5.10.1 Detailed Description

Fichier [memoire.h](#) contenant l'ensemble des prototypes et des déclarations pour les procédures de gestion de mémoire.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [memoire.h](#).

5.10.2 Function Documentation

5.10.2.1 [allouer_malloc\(\)](#)

```
void * allouer_malloc (  
    int n )
```

Permet de faire un malloc tout en prenant en compte les erreurs de pointeurs NULL.

Parameters

<i>n</i>	
----------	--

Returns

void*

Definition at line 20 of file [memoire.c](#).

5.11 memoire.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef MEMOIRE_H
00012 #define MEMOIRE_H
00013
00014 void *allouer_malloc(int taille);
00015
00016 #endif // MEMOIRE_H
```

5.12 include/menu.h File Reference

Fichier [menu.h](#) contenant l'ensemble des prototypes et des déclarations pour la procédure menu.

```
#include "fitstruct.h"
#include "lecture.h"
#include "operation.h"
#include "header.h"
#include "ecriture.h"
```

Functions

- void [menu](#) ()

Permet d'effectuer une suite d'opération choisi par l'utilisateur sur 2 images présélectionnées.

5.12.1 Detailed Description

Fichier [menu.h](#) contenant l'ensemble des prototypes et des déclarations pour la procédure menu.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [menu.h](#).

5.12.2 Function Documentation

5.12.2.1 menu()

```
void menu ( )
```

Permet d'effectuer une suite d'opération choisi par l'utilisateur sur 2 images présélectionnées.

Definition at line 17 of file [menu.c](#).

5.13 menu.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef MENU_H
00012 #define MENU_H
00013 #include "fitstruct.h"
00014 #include "lecture.h"
00015 #include "operation.h"
00016 #include "header.h"
00017 #include "ecriture.h"
00018
00019 void menu();
00020
00021 #endif // MENU_H
```

5.14 include/operation.h File Reference

Fichier [operation.h](#) contenant l'ensemble des prototypes et des déclarations pour les procédures d'opérations.

```
#include "fitstruct.h"
```

Functions

- [int headers_compatible](#) ([FitStruct](#) *images, int nombre_images)
Retourne 1 si les header sont compatibles, 0 sinon.
- [FitStruct somme_image](#) ([FitStruct](#) *images, int nombre_images)
*Renvoie la somme des images en fonction d'une liste d'image passés en paramètre. Répond à l'exigence **PRIM_50**.*
- [FitStruct moyenne_image](#) ([FitStruct](#) *images, int nombre_images)
*Renvoie la moyenne des images en fonction d'une liste d'image passés en paramètre. Répond à l'exigence **PRIM_60**.*
- [FitStruct diviser_image](#) ([FitStruct](#) images1, [FitStruct](#) images2)
*Renvoie la division de 2 images passés en paramètre. Répond à l'exigence **PRIM_80**.*
- [FitStruct soustraire_image](#) ([FitStruct](#) images1, [FitStruct](#) images2)
*Renvoie la différence de 2 images passés en paramètre. Répond à l'exigence **PRIM_70**.*
- [void afficher_premieres_valeurs](#) ([FitStruct](#) somme, int nombre_valeurs)
Affiche le nombre de valeurs d'une [FitStruct](#), permet de vérifier les calculs des opérations.

5.14.1 Detailed Description

Fichier [operation.h](#) contenant l'ensemble des prototypes et des déclarations pour les procédures d'opérations.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [operation.h](#).

5.14.2 Function Documentation

5.14.2.1 `afficher_premieres_valeurs()`

```
void afficher_premieres_valeurs (
    FitStruct image,
    int nombre_valeurs )
```

Affiche le nombre de valeurs d'une [FitStruct](#), permet de vérifier les calculs des opérations.

Parameters

<i>image</i>	
<i>nombre_valeurs</i>	

Definition at line 75 of file [operation.c](#).

5.14.2.2 `diviser_image()`

```
FitStruct diviser_image (
    FitStruct images1,
    FitStruct images2 )
```

Renvoie la division de 2 images passés en paramètre. Répond à l'exigence **PRIM_80**.

Parameters

<i>images1</i>	
<i>images2</i>	

Returns

[FitStruct](#)

Definition at line 134 of file [operation.c](#).

5.14.2.3 headers_compatible()

```
int headers_compatible (
    FitStruct * images,
    int nombre_images )
```

Retourne 1 si les header sont compatibles, 0 sinon.

Parameters

<i>images</i>	
<i>nombre_images</i>	

Returns

int

Definition at line 12 of file [operation.c](#).

5.14.2.4 moyenne_image()

```
FitStruct moyenne_image (
    FitStruct * images,
    int nombre_images )
```

Renvoie la moyenne des images en fonction d'une liste d'image passés en paramètre. Répond à l'exigence **PRIM**↔
_60.

Parameters

<i>images</i>	
<i>nombre_images</i>	

Returns

[FitStruct](#)

Definition at line 90 of file [operation.c](#).

5.14.2.5 somme_image()

```
FitStruct somme_image (
    FitStruct * images,
    int nombre_images )
```

Renvoie la somme des images en fonction d'une liste d'image passés en paramètre. Répond à l'exigence **PRIM_50**.

Parameters

<i>images</i>	
<i>nombre_images</i>	

Returns

[FitStruct](#)

Definition at line [34](#) of file [operation.c](#).

5.14.2.6 soustraire_image()

```
FitStruct soustraire_image (
    FitStruct images1,
    FitStruct images2 )
```

Renvoie la différence de 2 images passés en paramètre. Répond à l'exigence **PRIM_70**.

Parameters

<i>images1</i>	
<i>images2</i>	

Returns

[FitStruct](#)

Definition at line [182](#) of file [operation.c](#).

5.15 operation.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef OPERATION_H
00012 #define OPERATION_H
00013 #include "fitstruct.h"
00014
00015 int headers_compatible(FitStruct *images, int nombre_images);
00016 FitStruct somme_image(FitStruct *images, int nombre_images);
00017 FitStruct moyenne_image(FitStruct *images, int nombre_images);
00018 FitStruct diviser_image(FitStruct images1, FitStruct images2);
00019 FitStruct soustraire_image(FitStruct images1, FitStruct images2);
00020
00021 void afficher_premieres_valeurs(FitStruct somme, int nombre_valeurs);
00022
00023 #endif // OPERATION_H
```


5.16 src/ecriture.c File Reference

Fichier [ecriture.c](#) contenant l'ensemble des fonctions pour la lecture de fichier / données.

```
#include "ecriture.h"
#include "header.h"
#include "lecture.h"
```

Functions

- void [ecrire_pixels_csv](#) ([FitStruct](#) fitStruct, char *nom_fichier_csv)
*Écrit les valeurs réelles d'une image dans un fichier CSV $\text{valeur_réelle} = \text{BZERO} + \text{BSCALE} \times \text{valeur_enregistrée}$. Répond à l'exigence **PRIM_40***
- void [ecrire_fit_file](#) ([FitStruct](#) fitStruct, char *filename)
Convertir une [FitStruct](#) en fichier .fit.

5.16.1 Detailed Description

Fichier [ecriture.c](#) contenant l'ensemble des fonctions pour la lecture de fichier / données.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [ecriture.c](#).

5.16.2 Function Documentation

5.16.2.1 [ecrire_fit_file\(\)](#)

```
void écrire_fit_file (
    FitStruct fitStruct,
    char * filename )
```

Convertir une [FitStruct](#) en fichier .fit.

Parameters

<i>fitStruct</i>	Structure fitStruct d'une image .fit
<i>filenameLe</i>	Le chemin du fichier .fit dans lequel on va écrire

Definition at line 47 of file [ecriture.c](#).

5.16.2.2 `ecrire_pixels_csv()`

```
void écrire_pixels_csv (
    FitStruct fitStruct,
    char * nom_fichier_csv )
```

Écrit les valeurs réelles d'une image dans un fichier CSV valeur_réelle = BZERO + BSCALE × valeur_enregistrée.
Répond à l'exigence **PRIM_40**

Parameters

<i>fitStruct</i>	Structure fitStruct d'une image .fit
<i>nom_fichier_csv</i>	Le chemin du fichier .csv dans lequel on va écrire

Definition at line 21 of file [ecriture.c](#).

5.17 `ecriture.c`

[Go to the documentation of this file.](#)

```
00001
00011 #include "ecriture.h"
00012 #include "header.h"
00013 #include "lecture.h"
00014
00021 void écrire_pixels_csv(FitStruct fitStruct, char *nom_fichier_csv)
00022 {
00023     FILE *fichier_csv = ouvrir_fichier(nom_fichier_csv, "w");
00024
00025     int16_t *pixels = fitStruct.data;
00026     for (int i = 0; i < fitStruct.header_fichier.NAXIS2; i++)
00027     {
00028         for (int j = 0; j < fitStruct.header_fichier.NAXIS1; j++)
00029         {
00030             fprintf(fichier_csv, "%hd;", (fitStruct.header_fichier.BZERO +
fitStruct.header_fichier.BSCALE * pixels[i * fitStruct.header_fichier.NAXIS1 + j]));
00031             if (j == fitStruct.header_fichier.NAXIS1 - 1)
00032             {
00033                 fprintf(fichier_csv, "\n");
00034             }
00035         }
00036     }
00037
00038     fclose(fichier_csv);
00039 }
00040
00047 void écrire_fit_file(FitStruct fitStruct, char *filename)
00048 {
00049     FILE *output_file = fopen(filename, "wb");
00050
00051     // Écrire le header dans le fichier
00052     fwrite(&fitStruct.header_fichier, sizeof(Header), 1, output_file);
00053
00054     // Écrire les données dans le fichier
00055     fwrite(fitStruct.data, sizeof(int16_t), fitStruct.header_fichier.NAXIS1 *
fitStruct.header_fichier.NAXIS2, output_file);
00056
00057     fclose(output_file);
00058 }
```

5.18 src/fitstruct.c File Reference

Fichier [fitstruct.c](#) contenant l'ensemble des fonctions pour traiter les fitstruct.

```
#include "fitstruct.h"
#include "lecture.h"
#include "header.h"
```

Functions

- struct [FitStruct](#) [construct_fitstruct](#) (FILE *fichier)

On passe un fichier en paramètre et on construit sa structure [FitStruct](#). Dans cette structure on stockera les données de l'image et son header. Cette structure nous permettra de faire les opérations sur les images et l'écriture du CSV.

5.18.1 Detailed Description

Fichier [fitstruct.c](#) contenant l'ensemble des fonctions pour traiter les fitstruct.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [fitstruct.c](#).

5.18.2 Function Documentation

5.18.2.1 [construct_fitstruct\(\)](#)

```
struct FitStruct construct\_fitstruct (
    FILE * fichier )
```

On passe un fichier en paramètre et on construit sa structure [FitStruct](#). Dans cette structure on stockera les données de l'image et son header. Cette structure nous permettra de faire les opérations sur les images et l'écriture du CSV.

Parameters

<i>fichier</i>	Le chemin du fichier .fit pour lequel on veut obtenir sa structure FitStruct .
----------------	--

Returns

struct [FitStruct](#)

Definition at line 21 of file [fitstruct.c](#).

5.19 fitstruct.c

[Go to the documentation of this file.](#)

```

00001
00011 #include "fitstruct.h"
00012 #include "lecture.h"
00013 #include "header.h"
00014
00021 struct FitStruct construct_fitstruct(FILE *fichier)
00022 {
00023     FitStruct fit_fichier;
00024
00025     Header mon_header = construct_header(fichier);
00026     fit_fichier.header_fichier = mon_header;
00027     fit_fichier.data = lire_donnees_image(fichier, mon_header.NAXIS1, mon_header.NAXIS2);
00028
00029     return fit_fichier;
00030 }
```

5.20 src/header.c File Reference

Fichier [header.c](#) contenant l'ensemble des fonctions pour traiter les header.

```
#include "lecture.h"
#include "header.h"
```

Functions

- void [afficher_header](#) (char header[OCTETS_HEADER])
Affiche le header passé en paramètre.
- int [cle_valide](#) (char *cle)
Renvoie 1 si la clé passé en paramètre est contenue dans la liste, 0 sinon.
- void [process_header](#) ([Header](#) *mon_header, char packet80[LONGUEUR_LIGNES_HEADER])
Associe les valeurs contenues dans le header aux variables de la structure pour pouvoir les conserver et les utiliser ultérieurement.
- struct [Header](#) [construct_header](#) (FILE *mon_fichier)
*Construit la structure du [Header](#) associée au fichier passé en paramètre. Répond a l'exigence **PRIM_30***

5.20.1 Detailed Description

Fichier [header.c](#) contenant l'ensemble des fonctions pour traiter les header.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2023-12-24

Copyright

Copyright (c) 2023

Definition in file [header.c](#).

5.20.2 Function Documentation

5.20.2.1 afficher_header()

```
void afficher_header (
    char header[OCTETS_HEADER] )
```

Affiche le header passé en paramètre.

Parameters

<i>header</i>	
---------------	--

Definition at line 19 of file [header.c](#).

5.20.2.2 cle_valide()

```
int cle_valide (
    char * cle )
```

Renvoie 1 si la clé passé en paramètre est contenue dans la liste, 0 sinon.

Parameters

<i>cle</i>	
------------	--

Returns

int

Definition at line 33 of file [header.c](#).**5.20.2.3 construct_header()**

```
struct Header construct_header (
    FILE * mon_fichier )
```

Construit la structure du [Header](#) associée au fichier passé en paramètre. Répond a l'exigence **PRIM_30**

Parameters

<i>mon_fichier</i>	Le chemin du fichier .fit pour lequel on veut obtenir sa structure Header .
--------------------	---

Returnsstruct [Header](#)Definition at line 119 of file [header.c](#).**5.20.2.4 process_header()**

```
void process_header (
    Header * mon_header,
    char packet80[LONGUEUR_LIGNES_HEADER] )
```

Associe les valeurs contenues dans le header aux variables de la structure pour pouvoir les conserver et les utiliser ultérieurement.

Parameters

<i>mon_header</i>	
<i>packet80</i>	

Definition at line 55 of file [header.c](#).**5.21 header.c**

[Go to the documentation of this file.](#)

```
00001
00011 #include "lecture.h"
00012 #include "header.h"
00013
00019 void afficher_header(char header[OCTETS_HEADER])
00020 {
00021     for (int i = 0; i < NB_LIGNES_HEADER * LONGUEUR_LIGNES_HEADER; i++)
00022     {
00023         printf("%c%s", header[i], (i + 1) % LONGUEUR_LIGNES_HEADER ? " " : "\n");
```

```

00024     }
00025 }
00026
00033 int cle_valide(char *cle)
00034 {
00035     int estValide = 0;
00036
00037     char *cles_valides[NB_CLES_VALIDES] = {"BITPIX", "NAXIS", "NAXIS1", "NAXIS2", "NAXIS3", "BZERO",
"BSCALE"};
00038
00039     for (int i = 0; i < NB_CLES_VALIDES; i++)
00040     {
00041         if (strcmp(cle, cles_valides[i]) == 0)
00042         {
00043             estValide = 1;
00044         }
00045     }
00046     return estValide;
00047 }
00048
00055 void process_header(Header *mon_header, char packet80[LONGUEUR_LIGNES_HEADER])
00056 {
00057     char cle[8];
00058     char valeur[72];
00059     if (sscanf(packet80, "%7s = %71[^\n]", cle, valeur) == 2)
00060     {
00061         // printf("CLE: %s\n", cle);
00062         // printf("VALEUR: %s\n", valeur);
00063         // if (!cle_valide(cle))
00064         //     return;
00065         if (!strncmp(cle, "NAXIS", 6))
00066         {
00067             mon_header->NAXIS = atoi(valeur);
00068             return;
00069         }
00070         if (!strncmp(cle, "NAXIS1", 6))
00071         {
00072             mon_header->NAXIS1 = atoi(valeur);
00073             return;
00074         }
00075         if (!strncmp(cle, "NAXIS2", 6))
00076         {
00077             mon_header->NAXIS2 = atoi(valeur);
00078             return;
00079         }
00080         if (!strncmp(cle, "NAXIS3", 6))
00081         {
00082             mon_header->NAXIS3 = atoi(valeur);
00083             return;
00084         }
00085         if (!strncmp(cle, "BZERO", 5))
00086         {
00087             mon_header->BZERO = atoi(valeur);
00088             return;
00089         }
00090         if (!strncmp(cle, "BSCALE", 6))
00091         {
00092             mon_header->BSCALE = atoi(valeur);
00093             return;
00094         }
00095         if (!strncmp(cle, "BITPIX", 6))
00096         {
00097             mon_header->BITPIX = atoi(valeur);
00098             return;
00099         }
00100         if (!strncmp(cle, "SIMPLE", 6))
00101         {
00102             mon_header->SIMPLE = *cle;
00103             return;
00104         }
00105         if (!strncmp(cle, "SIMPLE", 6))
00106         {
00107             mon_header->SIMPLE = *cle;
00108             return;
00109         }
00110     }
00111 }
00112
00119 struct Header construct_header(FILE *mon_fichier)
00120 {
00121     char *data = lire_donnees_header(mon_fichier);
00122     Header mon_header = {0};
00123
00124     for (int i = 0; i < NB_LIGNES_HEADER; i++)
00125     {
00126         char ligne[LONGUEUR_LIGNES_HEADER + 1];
00127         strncpy(ligne, &data[i * LONGUEUR_LIGNES_HEADER], LONGUEUR_LIGNES_HEADER);

```

```

00128     ligne[LONGUEUR_LIGNES_HEADER] = '\0';
00129
00130     process_header(&mon_header, ligne);
00131 }
00132
00133     printf("\n*****");
00134     printf("\nLes données importantes du header à retenir sont :");
00135     printf("\nBITPIX = %d, NAXIS = %d, NAXIS1 = %d, NAXIS2= %d, NAXIS3= %d, BZERO = %d, BSCALE =
%d, SIMPLE = %c\n", mon_header.BITPIX, mon_header.NAXIS, mon_header.NAXIS1, mon_header.NAXIS2,
mon_header.NAXIS3, mon_header.BZERO, mon_header.BSCALE, mon_header.SIMPLE);
00136
00137     free(data);
00138     return mon_header;
00139 }

```

5.22 src/lecture.c File Reference

Fichier [lecture.c](#) contenant l'ensemble des fonctions pour la lecture de fichier et de header.

```

#include "lecture.h"
#include "memoire.h"
#include "header.h"
#include "fitstruct.h"
#include <stdio.h>
#include <stdlib.h>

```

Functions

- char * [lire_donnees_header](#) (FILE *fichier)
Retourne les données du header du fichier.
- FILE * [ouvrir_fichier](#) (char *chemin_fichier, char *option)
Ouvre le fichier passé en paramètre avec son option.
- int16_t * [lire_donnees_image](#) (FILE *fichier, int naxis1, int naxis2)
Lit les données brutes de l'images en format big endian.

5.22.1 Detailed Description

Fichier [lecture.c](#) contenant l'ensemble des fonctions pour la lecture de fichier et de header.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2023-12-24

Copyright

Copyright (c) 2023

Definition in file [lecture.c](#).

5.22.2 Function Documentation

5.22.2.1 lire_donnees_header()

```
char * lire_donnees_header (
    FILE * fichier )
```

Retourne les données du header du fichier.

Parameters

<i>fichier</i>	Le chemin du fichier .fit pour lequel on veut lire les données du header
----------------	--

Returns

char*

Definition at line 24 of file [lecture.c](#).

5.22.2.2 lire_donnees_image()

```
int16_t * lire_donnees_image (
    FILE * fichier,
    int naxis1,
    int naxis2 )
```

Lit les données brutes de l'images en format big endian.

Parameters

<i>fichier</i>	Le chemin du fichier .fit pour lequel on veut lire les données
<i>naxis1</i>	La valeur NAXIS1 de ce fichier
<i>naxis2</i>	La valeur NAXIS2 de ce fichier

Returns

uint16_t*

Definition at line 60 of file [lecture.c](#).

5.22.2.3 ouvrir_fichier()

```
FILE * ouvrir_fichier (
    char * chemin_fichier,
    char * option )
```

Ouvre le fichier passé en paramètre avec son option.

Parameters

<i>chemin_fichier</i>	
<i>option</i>	

Returns

FILE*

Definition at line 40 of file [lecture.c](#).

5.23 lecture.c

[Go to the documentation of this file.](#)

```

00001
00011 #include "lecture.h"
00012 #include "memoire.h"
00013 #include "header.h"
00014 #include "fitstruct.h"
00015 #include <stdio.h>
00016 #include <stdlib.h>
00017
00024 char *lire_donnees_header(FILE *fichier)
00025 {
00026     char *buffer = (char *)allouer_malloc(sizeof(char) * NB_LIGNES_HEADER * LONGUEUR_LIGNES_HEADER);
00027
00028     fread(buffer, OCTETS_HEADER, 1, fichier);
00029
00030     return buffer;
00031 }
00032
00040 FILE *ouvrir_fichier(char *chemin_fichier, char *option)
00041 {
00042     FILE *mon_fichier = fopen(chemin_fichier, option);
00043
00044     if (mon_fichier == NULL)
00045     {
00046         fprintf(stderr, "Impossible d'ouvrir le fichier %s\n", chemin_fichier);
00047         exit(EXIT_FAILURE);
00048     }
00049     return mon_fichier;
00050 }
00051
00060 int16_t *lire_donnees_image(FILE *fichier, int naxis1, int naxis2)
00061 {
00062     fseek(fichier, 2880, SEEK_SET);
00063     size_t total_pixels = naxis1 * naxis2 * 2;
00064     int16_t *buffer = (int16_t *)allouer_malloc(total_pixels * sizeof(int16_t));
00065
00066     uint16_t pixel;
00067     for (size_t i = 0; i < total_pixels; i++)
00068     {
00069         if (fread(&pixel, sizeof(pixel), 1, fichier) == 1)
00070         {
00071             pixel = (pixel » 8) | (pixel « 8);
00072             buffer[i] = pixel;
00073         }
00074     }
00075
00076     return buffer;
00077 }

```

5.24 src/main.c File Reference

Fichier [main.c](#) contenant l'ensemble de nos procédures de tests.

```
#include <stdio.h>
#include <stdlib.h>
#include "header.h"
#include "lecture.h"
#include "fitstruct.h"
#include "ecriture.h"
#include "operation.h"
#include "menu.h"
```

Functions

- int [main](#) ()

5.24.1 Detailed Description

Fichier [main.c](#) contenant l'ensemble de nos procédures de tests.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2023-12-24

Copyright

Copyright (c) 2023

Definition in file [main.c](#).

5.24.2 Function Documentation

5.24.2.1 [main\(\)](#)

```
int main ( )
```

Definition at line [20](#) of file [main.c](#).

5.25 main.c

[Go to the documentation of this file.](#)

```

00001
00011 #include <stdio.h>
00012 #include <stdlib.h>
00013 #include "header.h"
00014 #include "lecture.h"
00015 #include "fitstruct.h"
00016 #include "ecriture.h"
00017 #include "operation.h"
00018 #include "menu.h"
00019
00020 int main()
00021 {
00022     //***** DECLARATION ET OUVERTURE DES FICHIERS *****
00023     // FILE *mon_fichier = ouvrir_fichier("Images//lights/r_lights_00001.fit", "rb");
00024     // FILE *mon_fichier2 = ouvrir_fichier("Images//lights/r_lights_00002.fit", "rb");
00025
00026     // //***** CONSTRUCTION DES HEADERS DES FICHIERS *****
00027     // // construct_header(mon_fichier);
00028
00029     // //***** CONSTRUCTION DES FitStruct DES FICHIERS *****
00030     // FitStruct maFitStruct = construct_fitstruct(mon_fichier);
00031     // FitStruct maFitStruct2 = construct_fitstruct(mon_fichier2);
00032     // FitStruct resultat_somme, resultat_moyenne, resultat_division, resultat_soustraction;
00033
00034     // //***** DECLARATION LISTE DE FITSTRUCT *****
00035     // FitStruct fitStructs[] = {maFitStruct, maFitStruct2};
00036
00037     //***** TEST HEADERS IDENTIQUE DE 2 FICHIERS *****
00038     // if (headers_compatible(fitStructs, 2))
00039     // {
00040     //     printf("Les headers sont compatibles.\n");
00041     // }
00042     // else
00043     // {
00044     //     printf("Les headers ne sont pas compatibles.\n");
00045     // }
00046
00047     //***** TEST ELABORATION .CSV *****
00048     // ecrire_pixels_csv(maFitStruct, "test.csv");
00049
00050     //***** AFFICHAGE DES PREMIERS PIXELS DES IMAGES *****
00051     // printf("\n*****IMAGE N°1*****\n");
00052     // afficher_premieres_valeurs(maFitStruct, 10);
00053
00054     // printf("\n*****IMAGE N°2*****\n");
00055     // afficher_premieres_valeurs(maFitStruct2, 10);
00056     // //***** TEST OPERATIONS *****
00057     // // Tester la somme
00058     // resultat_somme = somme_image(fitStructs, 2);
00059     // printf("\n*****SOMME*****\n");
00060     // afficher_premieres_valeurs(resultat_somme, 10);
00061
00062     // // Tester la moyenne
00063     // resultat_moyenne = moyenne_image(fitStructs, 2);
00064     // printf("\n*****MOYENNE*****\n");
00065     // afficher_premieres_valeurs(resultat_moyenne, 10);
00066
00067     // // Tester la division
00068     // resultat_division = diviser_image(maFitStruct, maFitStruct2);
00069     // printf("\n*****DIVISION*****\n");
00070     // afficher_premieres_valeurs(resultat_division, 10);
00071
00072     // // Tester la soustraction
00073     // resultat_soustraction = soustraire_image(maFitStruct, maFitStruct2);
00074     // printf("\n*****SOUSTRACTION*****\n");
00075     // afficher_premieres_valeurs(resultat_soustraction, 10);
00076
00077     // fclose(mon_fichier);
00078     // fclose(mon_fichier2);
00079
00080     //***** CREER .FIT A PARTIR D'UNE FITSTRUCT *****
00081     // FILE *mon_fichier = ouvrir_fichier("Images//lights/r_lights_00001.fit", "rb");
00082     // FitStruct maFitStruct = construct_fitstruct(mon_fichier);
00083     // ecrire_fit_file(maFitStruct, "nouveau_fichier.fit");
00084
00085     menu();
00086
00087     return 0;
00088 }

```

5.26 src/memoire.c File Reference

Fichier [memoire.c](#) contenant l'ensemble des fonctions pour les opérations concernant la mémoire.

```
#include "memoire.h"
#include <stdlib.h>
```

Functions

- void * [allouer_malloc](#) (int n)
Permet de faire un malloc tout en prenant en compte les erreurs de pointeurs NULL.

5.26.1 Detailed Description

Fichier [memoire.c](#) contenant l'ensemble des fonctions pour les opérations concernant la mémoire.

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [memoire.c](#).

5.26.2 Function Documentation

5.26.2.1 [allouer_malloc\(\)](#)

```
void * allouer_malloc (
    int n )
```

Permet de faire un malloc tout en prenant en compte les erreurs de pointeurs NULL.

Parameters

<i>n</i>	
----------	--

Returns

void*

Definition at line 20 of file [memoire.c](#).

5.27 memoire.c

[Go to the documentation of this file.](#)

```
00001
00011 #include "memoire.h"
00012 #include <stdlib.h>
00013
00020 void *allouer_malloc(int n)
00021 {
00022     void *pointeur;
00023
00024     pointeur = malloc(n);
00025
00026     if (pointeur == NULL)
00027     {
00028         exit(EXIT_FAILURE);
00029     }
00030
00031     return pointeur;
00032 }
```

5.28 src/menu.c File Reference

Fichier [menu.c](#) contenant la fonction void [menu\(\)](#) - Répond à l'exigence optionnelle **OPT_30**

```
#include "menu.h"
```

Functions

- void [menu](#) ()
Permet d'effectuer une suite d'opération choisi par l'utilisateur sur 2 images présélectionnées.

5.28.1 Detailed Description

Fichier [menu.c](#) contenant la fonction void [menu\(\)](#) - Répond à l'exigence optionnelle **OPT_30**

Author

Alexandre, Célian, Sophie

Version

0.1

Date

2024-01-06

Copyright

Copyright (c) 2024

Definition in file [menu.c](#).

5.28.2 Function Documentation

5.28.2.1 menu()

```
void menu ( )
```

Permet d'effectuer une suite d'opération choisi par l'utilisateur sur 2 images présélectionnées.

Definition at line 17 of file [menu.c](#).

5.29 menu.c

[Go to the documentation of this file.](#)

```
00001
00011 #include "menu.h"
00012
00017 void menu()
00018 {
00019     FILE *mon_fichier1 = ouvrir_fichier("Images//lights/r_lights_00001.fit", "rb");
00020     FILE *mon_fichier2 = ouvrir_fichier("Images//lights/r_lights_00002.fit", "rb");
00021
00022     printf("\nINFORMATIONS HEADER 1ère IMAGE:");
00023     FitStruct maFitStruct1 = construct_fitstruct(mon_fichier1);
00024     printf("\nINFORMATIONS HEADER 2nde IMAGE:");
00025     FitStruct maFitStruct2 = construct_fitstruct(mon_fichier2);
00026
00027     FitStruct fitStructs[] = {maFitStruct1, maFitStruct2};
00028
00029     int choix;
00030     int continuer = 1;
00031
00032     while (continuer)
00033     {
00034         printf("\n===== Menu =====\n");
00035         printf("1. Faire le CSV de la 1ère image\n");
00036         printf("2. Faire le CSV de la 2ème image\n");
00037         printf("3. Somme d'images\n");
00038         printf("4. Moyenne d'images\n");
00039         printf("5. Division d'images\n");
00040         printf("6. Soustraction d'images\n");
00041         printf("0. Quitter\n");
00042         printf("Choix: ");
00043         scanf("%d", &choix);
00044
00045         switch (choix)
00046         {
00047             case 1:
00048                 // Afficher les headers de la première image
00049                 ecrire_pixels_csv(maFitStruct1, "image1.csv");
00050
00051                 break;
00052             case 2:
00053                 // Afficher les headers de la deuxième image
00054                 ecrire_pixels_csv(maFitStruct2, "image2.csv");
00055
00056                 break;
00057             case 3:
00058                 // Somme d'images
00059                 FitStruct resultat_somme = somme_image(fitStructs, 2);
00060                 afficher_premieres_valeurs(resultat_somme, 10);
00061                 ecrire_fit_file(resultat_somme, "somme.fit");
00062
00063                 break;
00064             case 4:
00065                 // Moyenne d'images
00066                 FitStruct resultat_moyenne = moyenne_image(fitStructs, 2);
00067                 afficher_premieres_valeurs(resultat_moyenne, 10);
00068                 ecrire_fit_file(resultat_moyenne, "moyenne.fit");
00069
00070                 break;
00071             case 5:
00072                 // Division d'images
00073                 FitStruct resultat_division = diviser_image(maFitStruct1, maFitStruct2);
00074                 afficher_premieres_valeurs(resultat_division, 10);
00075                 ecrire_fit_file(resultat_division, "division.fit");
00076
00077                 break;
00078             case 0:
00079                 continuer = 0;
00080                 break;
00081             default:
00082                 printf("Choix invalide\n");
00083                 break;
00084         }
00085     }
00086 }
```

```

00077         break;
00078     case 6:
00079         // Soustraction d'images
00080         FitStruct resultat_soustraction = soustraire_image(maFitStruct1, maFitStruct2);
00081         afficher_premieres_valeurs(resultat_soustraction, 10);
00082         ecrire_fit_file(resultat_soustraction, "soustraction.fit");
00083     case 0:
00084         break;
00085     case 0:
00086         // Quitter
00087         continuer = 0;
00088     case 0:
00089         break;
00090     default:
00091         printf("Choix non valide. Veuillez réessayer.\n");
00092     }
00093 }
00094
00095 fclose(mon_fichier1);
00096 fclose(mon_fichier2);
00097 }

```

5.30 operation.c

```

00001 #include "operation.h"
00002 #include "fitstruct.h"
00003 #include <limits.h>
00004
00012 int headers_compatible(FitStruct *images, int nombre_images)
00013 {
00014     for (int i = 1; i < nombre_images; i++)
00015     {
00016         if (images[0].header_fichier.NAXIS1 != images[i].header_fichier.NAXIS1 ||
00017             images[0].header_fichier.NAXIS2 != images[i].header_fichier.NAXIS2 ||
00018             images[0].header_fichier.BITPIX != images[i].header_fichier.BITPIX)
00019         {
00020             return 0;
00021         }
00022     }
00023     return 1;
00024 }
00025
00034 FitStruct somme_image(FitStruct *images, int nombre_images)
00035 {
00036     // Vérifier la compatibilité des headers
00037     if (!headers_compatible(images, nombre_images))
00038     {
00039         fprintf(stderr, "Les headers des images ne sont pas compatibles.\n");
00040     }
00041     FitStruct resultat;
00042     int naxis1 = images[0].header_fichier.NAXIS1;
00043     int naxis2 = images[0].header_fichier.NAXIS2;
00044
00045     // Initialiser la structure résultat et allouer la mémoire pour les données
00046     resultat.header_fichier = images[0].header_fichier; // Copier le header de la première image
00047     resultat.data = (int16_t *)malloc(naxis1 * naxis2 * sizeof(int16_t));
00048
00049     // Initialiser les données à 0
00050     memset(resultat.data, 0, naxis1 * naxis2 * sizeof(uint16_t));
00051
00052     // Somme des images
00053     for (int i = 0; i < naxis2; i++)
00054     {
00055         for (int j = 0; j < naxis1; j++)
00056         {
00057             int32_t somme = 0;
00058             for (int k = 0; k < nombre_images; k++)
00059             {
00060                 somme += images[k].data[i * naxis1 + j];
00061             }
00062             resultat.data[i * naxis1 + j] = (somme > INT16_MAX) ? INT16_MAX : ((somme < INT16_MIN) ?
INT16_MIN : somme);
00063         }
00064     }
00065
00066     return resultat;
00067 }
00068
00075 void afficher_premieres_valeurs(FitStruct image, int nombre_valeurs)
00076 {
00077     for (int i = 0; i < nombre_valeurs; i++)
00078     {
00079         printf("Valeur brute %d: %hd\n", i, image.data[i]);

```



```

00080     }
00081 }
00082
00090 FitStruct moyenne_image(FitStruct *images, int nombre_images)
00091 {
00092     // Vérifier la compatibilité des headers
00093     if (!headers_compatible(images, nombre_images))
00094     {
00095         fprintf(stderr, "Les headers des images ne sont pas compatibles.\n");
00096     }
00097     FitStruct resultat;
00098     int naxis1 = images[0].header_fichier.NAXIS1;
00099     int naxis2 = images[0].header_fichier.NAXIS2;
00100
00101     // Initialiser la structure résultat et allouer la mémoire pour les données
00102     resultat.header_fichier = images[0].header_fichier; // Copier le header de la première image
00103     resultat.data = (int16_t *)malloc(naxis1 * naxis2 * sizeof(int16_t));
00104
00105     // Initialiser les données à 0
00106     memset(resultat.data, 0, naxis1 * naxis2 * sizeof(uint16_t));
00107
00108     // Moyenne des images
00109     for (int i = 0; i < naxis2; i++)
00110     {
00111         for (int j = 0; j < naxis1; j++)
00112         {
00113             double moyenne = 0.0;
00114             for (int k = 0; k < nombre_images; k++)
00115             {
00116                 moyenne += images[k].data[i * naxis1 + j];
00117             }
00118             moyenne /= nombre_images; // Calcul de la moyenne
00119             resultat.data[i * naxis1 + j] = (moyenne > INT16_MAX) ? INT16_MAX : (moyenne < INT16_MIN)
00120 ? INT16_MIN
00121 : (int16_t)moyenne; // Gérer les débordements : on vérifie si la moyenne ne dépasse pas les valeurs
00122 min ou max
00123         }
00124     }
00125     return resultat;
00126 }
00134 FitStruct diviser_image(FitStruct images1, FitStruct images2)
00135 {
00136     // Vérifier la compatibilité des headers
00137     if (!headers_compatible(&images1, 1) || !headers_compatible(&images2, 1))
00138     {
00139         fprintf(stderr, "Les headers des images ne sont pas compatibles.\n");
00140     }
00141
00142     FitStruct resultat;
00143     int naxis1 = images1.header_fichier.NAXIS1;
00144     int naxis2 = images1.header_fichier.NAXIS2;
00145
00146     // Initialiser la structure résultat et allouer la mémoire pour les données
00147     resultat.header_fichier = images1.header_fichier;
00148     resultat.data = (int16_t *)malloc(naxis1 * naxis2 * sizeof(int16_t));
00149
00150     // Division des images
00151     for (int i = 0; i < naxis2; i++)
00152     {
00153         for (int j = 0; j < naxis1; j++)
00154         {
00155             int32_t numerateur = images1.data[i * naxis1 + j];
00156             int32_t denominateur = images2.data[i * naxis1 + j];
00157
00158             if (denominateur != 0)
00159             {
00160                 double resultat_division = (double)numerateur / denominateur;
00161                 resultat.data[i * naxis1 + j] = (resultat_division > INT16_MAX) ? INT16_MAX :
00162 (resultat_division < INT16_MIN) ? INT16_MIN
00163 : (int16_t)resultat_division;
00164             }
00165             else
00166             {
00167                 // si on divise par 0, la valeur par défaut sera 0
00168                 resultat.data[i * naxis1 + j] = 0;
00169             }
00170         }
00171     }
00172     return resultat;
00173 }
00174
00182 FitStruct soustraire_image(FitStruct images1, FitStruct images2)

```

```
00183 {
00184     // Vérifier la compatibilité des headers
00185     if (!headers_compatible(&images1, 1) || !headers_compatible(&images2, 1))
00186     {
00187         fprintf(stderr, "Les headers des images ne sont pas compatibles.\n");
00188     }
00189
00190     FitStruct resultat;
00191     int naxis1 = images1.header_fichier.NAXIS1;
00192     int naxis2 = images1.header_fichier.NAXIS2;
00193
00194     // Initialiser la structure résultat et allouer la mémoire pour les données
00195     resultat.header_fichier = images1.header_fichier;
00196     resultat.data = (int16_t *)malloc(naxis1 * naxis2 * sizeof(int16_t));
00197
00198     // Soustraction des images
00199     for (int i = 0; i < naxis2; i++)
00200     {
00201         for (int j = 0; j < naxis1; j++)
00202         {
00203             int32_t difference = images1.data[i * naxis1 + j] - images2.data[i * naxis1 + j];
00204             resultat.data[i * naxis1 + j] = (difference < INT16_MIN) ? INT16_MIN : (difference >
00205             INT16_MAX) ? INT16_MAX
00206             : (int16_t)difference;
00207         }
00208     }
00209     return resultat;
00210 }
```

Index

- afficher_header
 - header.c, [31](#)
 - header.h, [16](#)
- afficher_premieres_valeurs
 - operation.h, [24](#)
- allouer_malloc
 - memoire.c, [39](#)
 - memoire.h, [21](#)
- BITPIX
 - Header, [8](#)
- BSCALE
 - Header, [8](#)
- BZERO
 - Header, [8](#)
- cle_valide
 - header.c, [31](#)
 - header.h, [17](#)
- construct_fitstruct
 - fitstruct.c, [29](#)
 - fitstruct.h, [14](#)
- construct_header
 - header.c, [32](#)
 - header.h, [17](#)
- data
 - FitStruct, [7](#)
- diviser_image
 - operation.h, [24](#)
- ecrire_fit_file
 - ecriture.c, [27](#)
 - ecriture.h, [12](#)
- ecrire_pixels_csv
 - ecriture.c, [28](#)
 - ecriture.h, [12](#)
- ecriture.c
 - ecrire_fit_file, [27](#)
 - ecrire_pixels_csv, [28](#)
- ecriture.h
 - ecrire_fit_file, [12](#)
 - ecrire_pixels_csv, [12](#)
- FitStruct, [7](#)
 - data, [7](#)
 - header_fichier, [7](#)
- fitstruct.c
 - construct_fitstruct, [29](#)
- fitstruct.h
 - construct_fitstruct, [14](#)
- Header, [8](#)
 - BITPIX, [8](#)
 - BSCALE, [8](#)
 - BZERO, [8](#)
 - NAXIS, [8](#)
 - NAXIS1, [9](#)
 - NAXIS2, [9](#)
 - NAXIS3, [9](#)
 - SIMPLE, [9](#)
- header.c
 - afficher_header, [31](#)
 - cle_valide, [31](#)
 - construct_header, [32](#)
 - process_header, [32](#)
- header.h
 - afficher_header, [16](#)
 - cle_valide, [17](#)
 - construct_header, [17](#)
 - LONGUEUR_LIGNES_HEADER, [16](#)
 - NB_CLES_VALIDES, [16](#)
 - NB_LIGNES_HEADER, [16](#)
 - OCTETS_HEADER, [16](#)
 - process_header, [17](#)
- header_fichier
 - FitStruct, [7](#)
- headers_compatible
 - operation.h, [25](#)
- include/ecriture.h, [11](#), [13](#)
- include/fitstruct.h, [13](#), [14](#)
- include/header.h, [15](#), [18](#)
- include/lecture.h, [18](#), [20](#)
- include/memoire.h, [20](#), [22](#)
- include/menu.h, [22](#), [23](#)
- include/operation.h, [23](#), [26](#)
- lecture.c
 - lire_donnees_header, [35](#)
 - lire_donnees_image, [35](#)
 - ouvrir_fichier, [35](#)
- lecture.h
 - lire_donnees_header, [19](#)
 - lire_donnees_image, [19](#)
 - ouvrir_fichier, [20](#)
- lire_donnees_header
 - lecture.c, [35](#)
 - lecture.h, [19](#)
- lire_donnees_image
 - lecture.c, [35](#)
 - lecture.h, [19](#)

LONGUEUR_LIGNES_HEADER
 header.h, 16

main
 main.c, 37

main.c
 main, 37

memoire.c
 allouer_malloc, 39

memoire.h
 allouer_malloc, 21

menu
 menu.c, 41
 menu.h, 23

menu.c
 menu, 41

menu.h
 menu, 23

moyenne_image
 operation.h, 25

NAXIS
 Header, 8

NAXIS1
 Header, 9

NAXIS2
 Header, 9

NAXIS3
 Header, 9

NB_CLES_VALIDES
 header.h, 16

NB_LIGNES_HEADER
 header.h, 16

OCTETS_HEADER
 header.h, 16

operation.h
 afficher_premieres_valeurs, 24
 diviser_image, 24
 headers_compatible, 25
 moyenne_image, 25
 somme_image, 25
 soustraire_image, 26

ouvrir_fichier
 lecture.c, 35
 lecture.h, 20

process_header
 header.c, 32
 header.h, 17

Projet d'algorithmique - CATALA Alexandre - DI GIO-
 VANNI Celian - SOUBRY Sophie, 1

SIMPLE
 Header, 9

somme_image
 operation.h, 25

soustraire_image
 operation.h, 26

src/ecriture.c, 27, 28
src/fitstruct.c, 29, 30
src/header.c, 30, 32
src/lecture.c, 34, 36
src/main.c, 36, 38
src/memoire.c, 39, 40
src/menu.c, 40, 41
src/operation.c, 42