PROJET D'ALGORITHMIQUE

Utilitaire de traitement de photo astronomique

CATALA ALEXANDRE
SOUBRY SOPHIE
DI GIOVANNI CÉLIAN

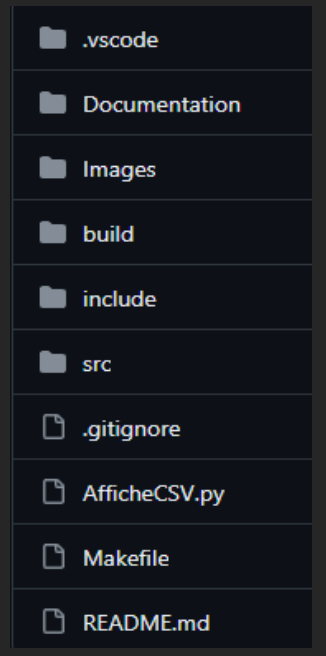
ISEN
ALL IS DIGITAL!
MÉDITERRANÉE
yncréa

SOMMAIRE

I. Présentation du projet

Objectif du projet

Travail en groupe avec GitHub



Architecture du projet

II. Fonctionnalités

- Lecture et construction du Header (stockage des données, association des valeurs et solution)
- Ajout d'une structure *FitStruct* pour répondre à nos besoins
- Conversion d'un fichier .fit au format .csv
- Opérations sur les images

1. Stockage des données des headers dans la structure Header

```
typedef struct Header
{
    char SIMPLE;
    int BITPIX;
    int NAXIS;
    int NAXIS1;
    int NAXIS2;
    int NAXIS3;
    int BZERO;
    int BSCALE;
} Header;
```

Structure Header

2. Association des valeurs du header à la structure pour pouvoir les conserver et les utiliser ultérieurement.

```c
void process_header(Header *mon_header, char packet80[LONGUEUR_LIGNES_HEADER])
{
    char cle[8];
    char valeur[72];
    if (sscanf(packet80, "%7s = %71[^\n]", cle, valeur) == 2)
    {
        // printf("CLE: %s\n", cle);
        // printf("VALEUR: %s\n", valeur);
        // if (!cle_valide(cle))
        //     return;
        if (!strncmp(cle, "NAXIS", 6))
        {
            mon_header->NAXIS = atoi(valeur);
            return;
        }
        if (!strncmp(cle, "NAXIS1", 6))
        {
            mon_header->NAXIS1 = atoi(valeur);
            return;
        }
        if (!strncmp(cle, "NAXIS2", 6))
        {
            mon_header->NAXIS2 = atoi(valeur);
            return;
        }
        if (!strncmp(cle, "NAXIS3", 6))
        {
            mon_header->NAXIS3 = atoi(valeur);
            return;
        }
        if (!strncmp(cle, "BZERO", 5))
        {
            mon_header->BZERO = atoi(valeur);
```

```c
            return;
        }
        if (!strncmp(cle, "BSCALE", 6))
        {
            mon_header->BSCALE = atoi(valeur);
            return;
        }
        if (!strncmp(cle, "BITPIX", 6))
        {
            mon_header->BITPIX = atoi(valeur);
            return;
        }
        if (!strncmp(cle, "SIMPLE", 6))
        {
            mon_header->SIMPLE = *cle;
            return;
        }
        if (!strncmp(cle, "SIMPLE", 6))
        {
            mon_header->SIMPLE = *cle;
            return;
        }
    }
}
```

3. Réponse à l'exigence PRIM_30

```
struct Header construct_header(FILE *mon_fichier)
{
    char *data = lire_donnees_header(mon_fichier);
    Header mon_header = {0};

    for (int i = 0; i < NB_LIGNES_HEADER; i++)
    {
        char ligne[LONGUEUR_LIGNES_HEADER + 1];
        strncpy(ligne, &data[i * LONGUEUR_LIGNES_HEADER], LONGUEUR_LIGNES_HEADER);
        ligne[LONGUEUR_LIGNES_HEADER] = '\0';

        process_header(&mon_header, ligne);
    }

    printf("\n*****************************************");
    printf("\nLes données importantes du header à retenir sont :");
    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);

    free(data);
    return mon_header;
}
```

La structure *FitStruct*

```c
typedef struct FitStruct
{
    int16_t *data;
    Header header_fichier;
} FitStruct;
```

Structure FitStruct

```c
struct FitStruct construct_fitstruct(FILE *fichier)
{
    FitStruct fit_fichier;

    Header mon_header = construct_header(fichier);
    fit_fichier.header_fichier = mon_header;
    fit_fichier.data = lire_donnees_image(fichier, mon_header.NAXIS1, mon_header.NAXIS2);

    return fit_fichier;
}
```

Fonction permettant de créer une structure FitStruct d'une image .fit

Lecture des données du .fit

```
int16_t *lire_donnees_image(FILE *fichier, int naxis1, int naxis2)
{
    fseek(fichier, 2880, SEEK_SET);
    size_t total_pixels = naxis1 * naxis2 * 2;
    int16_t *buffer = (int16_t *)allouer_malloc(total_pixels * sizeof(int16_t));

    uint16_t pixel;
    for (size_t i = 0; i < total_pixels; i++)
    {
        if (fread(&pixel, sizeof(pixel), 1, fichier) == 1)
        {
            pixel = (pixel » 8) | (pixel « 8);
            buffer[i] = pixel;
        }
    }

    return buffer;
}
```

PRIM_40 : Conversion des données au format CSV

```
void ecrire_pixels_csv(FitStruct fitStruct, char *nom_fichier_csv)
{
    FILE *fichier_csv = ouvrir_fichier(nom_fichier_csv, "w");

    int16_t *pixels = fitStruct.data;
    for (int i = 0; i < fitStruct.header_fichier.NAXIS2; i++)
    {
        for (int j = 0; j < fitStruct.header_fichier.NAXIS1; j++)
        {
            fprintf(fichier_csv, "%hd;", (fitStruct.header_fichier.BZERO +
fitStruct.header_fichier.BSCALE * pixels[i * fitStruct.header_fichier.NAXIS1 + j]));
            if (j == fitStruct.header_fichier.NAXIS1 - 1)
            {
                fprintf(fichier_csv, "\n");
            }
        }
    }

    fclose(fichier_csv);
}
```

Les opérations : PRIM_50 à PRIM_70

```c
FitStruct somme_image(FitStruct *images, int nombre_images)
{
    // Vérifier la compatibilité des headers
    if (!headers_compatible(images, nombre_images))
    {
        fprintf(stderr, "Les headers des images ne sont pas compatibles.\n");
    }
    FitStruct resultat;
    int naxis1 = images[0].header_fichier.NAXIS1;
    int naxis2 = images[0].header_fichier.NAXIS2;

    // Initialiser la structure résultat et allouer la mémoire pour les données
    resultat.header_fichier = images[0].header_fichier; // Copier le header de la première image
    resultat.data = (int16_t *)malloc(naxis1 * naxis2 * sizeof(int16_t));

    // Initialiser les données à 0
    memset(resultat.data, 0, naxis1 * naxis2 * sizeof(uint16_t));
```

La somme

```
    // Sommation des images
    for (int i = 0; i < naxis2; i++)
    {
        for (int j = 0; j < naxis1; j++)
        {
            int32_t somme = 0;
            for (int k = 0; k < nombre_images; k++)
            {
                somme += images[k].data[i * naxis1 + j];
            }
            resultat.data[i * naxis1 + j] = (somme > INT16_MAX) ? INT16_MAX : ((somme < INT16_MIN) ?
INT16_MIN : somme);
        }
    }

    return resultat;
}
```

La moyenne

```
    // Moyenne des images
    for (int i = 0; i < naxis2; i++)
    {
        for (int j = 0; j < naxis1; j++)
        {
            double moyenne = 0.0;
            for (int k = 0; k < nombre_images; k++)
            {
                moyenne += images[k].data[i * naxis1 + j];
            }
            moyenne /= nombre_images; // Calcul de la moyenne
            resultat.data[i * naxis1 + j] = (moyenne > INT16_MAX) ? INT16_MAX : (moyenne < INT16_MIN)
? INT16_MIN

: (int16_t)moyenne; // Gérer les débordements : on vérifie si la moyenne ne dépasse pas les valeurs
min ou max
        }
    }

    return resultat;
}
```

La division

```
    // Division des images
    for (int i = 0; i < naxis2; i++)
    {

      for (int j = 0; j < naxis1; j++)
      {
          int32_t numerateur = images1.data[i * naxis1 + j];
          int32_t denominateur = images2.data[i * naxis1 + j];

          if (denominateur != 0)
          {
              double resultat_division = (double)numerateur / denominateur;
              resultat.data[i * naxis1 + j] = (resultat_division > INT16_MAX) ? INT16_MAX :
esultat_division < INT16_MIN) ? INT16_MIN

(int16_t)resultat_division;
          }
          else
          {
              // si on divise par 0, la valeur par défaut sera 0
              resultat.data[i * naxis1 + j] = 0;
          }
      }
  }

  return resultat;
```

La soustraction

```
    // Soustraction des images
    for (int i = 0; i < naxis2; i++)
    {
        for (int j = 0; j < naxis1; j++)
        {
            int32_t difference = images1.data[i * naxis1 + j] - images2.data[i * naxis1 + j];
            resultat.data[i * naxis1 + j] = (difference < INT16_MIN) ? INT16_MIN : (difference >
INT16_MAX) ? INT16_MAX

: (int16_t)difference;
        }
    }

    return resultat;
}
```

III. Démonstration du projet

- Présentation du menu
- Récupération du header, écriture dans un .csv ...

```
void menu()
{
    FILE *mon_fichier1 = ouvrir_fichier("Images//lights/r_lights_00001.fit", "rb");
    FILE *mon_fichier2 = ouvrir_fichier("Images//lights/r_lights_00002.fit", "rb");

    printf("\nINFORMATIONS HEADER 1ère IMAGE:");
    FitStruct maFitStruct1 = construct_fitstruct(mon_fichier1);
    printf("\nINFORMATIONS HEADER 2nde IMAGE:");
    FitStruct maFitStruct2 = construct_fitstruct(mon_fichier2);

    FitStruct fitStructs[] = {maFitStruct1, maFitStruct2};

    int choix;
    int continuer = 1;

    while (continuer)
    {
        printf("\n===== Menu =====\n");
        printf("1. Faire le CSV de la 1ère image\n");
        printf("2. Faire le CSV de la 2ème image\n");
        printf("3. Somme d'images\n");
```

Extrait de la fonction menu()

IV. Conclusion

- Bilan du projet
- Perspectives d'améliorations

```
void ecrire_fit_file(FitStruct fitStruct, char *filename)
{
    FILE *output_file = fopen(filename, "wb");

    // Écrire le header dans le fichier
    fwrite(&fitStruct.header_fichier, sizeof(Header), 1, output_file);

    // Écrire les données dans le fichier
    fwrite(fitStruct.data, sizeof(int16_t), fitStruct.header_fichier.NAXIS1 *
fitStruct.header_fichier.NAXIS2, output_file);

    fclose(output_file);
}
```

Fonction ecrire_fit_file() qui n'est pas au point

MERCI DE NOUS AVOIR ÉCOUTÉ

AVEZ-VOUS DES QUESTIONS ?

Documentation du projet

- Lien du dépôt GitHub :

https://github.com/AviMcCartney/projetalgoS1.git

- Documentation des fichier.fit:

https://github.com/AviMcCartney/projetalgoS1/blob/main/Documentation/fits_standard30.pdf