Projet ASD2

APPLICATION D'EVOLUTION DU COVID-19

Blidi Abderrahmane - Chouria Mohamed amine

ISG TUNIS | 1LNIG

Sommaire

Tableau des figures

I. Étude de l'existence

Covid-19 fait référence à « Coronavirus Disease » 2019. Elle a émergé en décembre 2019 dans la ville de Wuhan en Chine. Elle est une maladie respiratoire pouvant être mortelle chez les patients fragilisés par l'âge ou une autre maladie chronique. Cette maladie s'est rapidement propagée, d'abord dans toute la Chine, puis à l'étranger provoquant une épidémie mondiale. En Tunisie, la première apparition du virus été en mars 2020.

Dans le cadre du Projet d'Algorithmique et Structures de Données du deuxième semestre on va traiter ce sujet. L'idée est de créer une application qui va présenter l'évolution quotidienne des statistiques concernant la corona virus du tous les 24 gouvernorats de Tunisie.

II. SOLUTION

Il s'agit de développer un programme en langage C avec une interface qui permet l'utilisateur à choisir entre des plusieurs fonctionnalités sur les statistiques.

L'utilisateur peut :

- Saisir pour chaque jour, pour chaque gouvernorat les chiffres relatifs à (Cas positifs, Guérisons, Décès, Nombre de vaccinés)
- Supprimer ou modifier n'importe quelle information.
- Afficher pour un gouvernorat donné, l'évolution de ses statistiques sur les 4 critères.
- Établir et afficher le classement des 24 gouvernorats triés par ordre croissant ou décroissant selon les 4 critères
- Remplir toutes les informations automatiquement à partir d'un fichier texte.

III. Réalisation

Dans la fonction main on a affiché un menu principal qui va demander de choisir entre la saisie, la suppression ou la modification du n'importe quelle information après la saisie, l'affichage de statistiques d'un gouvernorat, l'affichage des informations de tous les gouvernorats selon une date précise, ou la saisie à partir d'un fichier texte selon un certain format qui nous l'expliquerons plus tard.

Selon le numéro saisit par l'utilisateur le programme va appeler l'un des procédures :

- Saisir
- afficheEvol
- Affiche
- saisirTxt
- Modifier
- Supp

```
do
{
    vide=estVide(tab[0]);
    printf("||-------------------------------------------------------------------------------------------------------||\n");
    printf("||                                          CHOISISSEZ                                                   ||\n");
    printf("||-------------------------------------------------------------------------------------------------------||\n");
    printf("||                      1- SAISIR                                                                        ||\n");
    printf("||                      2- AFFICHER L'EVOLUTION DE STATISTIQUES D'UN GOUVERNORAT                         ||\n");
    printf("||                      3- AFFICHER LES INFORMATIONS DES GOUVERNORATS VIA UNE DATE                       ||\n");
    printf("||                      4- SAISIR A PARTIR D'UN FICHIER TEXTE                                            ||\n");
    if (vide==0)
    {
        printf("||                          5- MODIFIER                                                                  ||\n");
        printf("||                          6- SUPPRIMER                                                                 ||\n");
    }
    printf("||-------------------------------------------------------------------------------------------------------||\n");
    printf("||                                          7- QUITTER                                                   ||\n");
    printf("||-------------------------------------------------------------------------------------------------------||\n");
    c=getchar();
    switch (c)
    {
        case '1' : saisir(tab,noms); break;
        case '2' : afficheEvol(tab,noms); break;
        case '3' : affiche(tab,noms); break;
        case '4' : saisirTxt(tab); break;
        case '5' : if (vide==0) {modifier(tab,noms);} break;
        case '6' : if (vide==0) {supp(tab);} break;
        case '7' : quitter(tab);stop=1; break;
    }
    getchar();
} while ((c<'1')||(c>'7')||(stop==0));
```

Figure 1: menu principale

On va utiliser deux tableaux : un tableau des chaines de caractères contenant les noms de 24 gouvernorats et l'autre un tableau de 24 Listes bidirectionnelles. Chacune des cases du tableau concerne un gouvernorat. Chaque case de la liste d'un gouvernorat est relative à une date. Les listes ont la même taille car on suppose que nous avons les mêmes statistiques pour tous les gouvernorats. Les deux tableaux considèrent l'ordre alphabétique.

Les case de liste contient un élément. Cet élément est composé par :

- Une date de type d
- Nombre des cas positifs (cp) de type entier
- Nombre des guérisons (grs) de type entier
- Nombre des décès (dec) de type entier
- Nombre de vaccinés (nva) de type entier

```
typedef struct
{
    int j;
    int m;
    int a;
}d;

typedef struct
{
    d date;
    int cp;
    int grs;
    int dec;
    int nva;
}corona,*ELEMENT;
```

Figure 2 : les structures

Ce schéma montre un exemple de 3 dates pour 3 gouvernorats

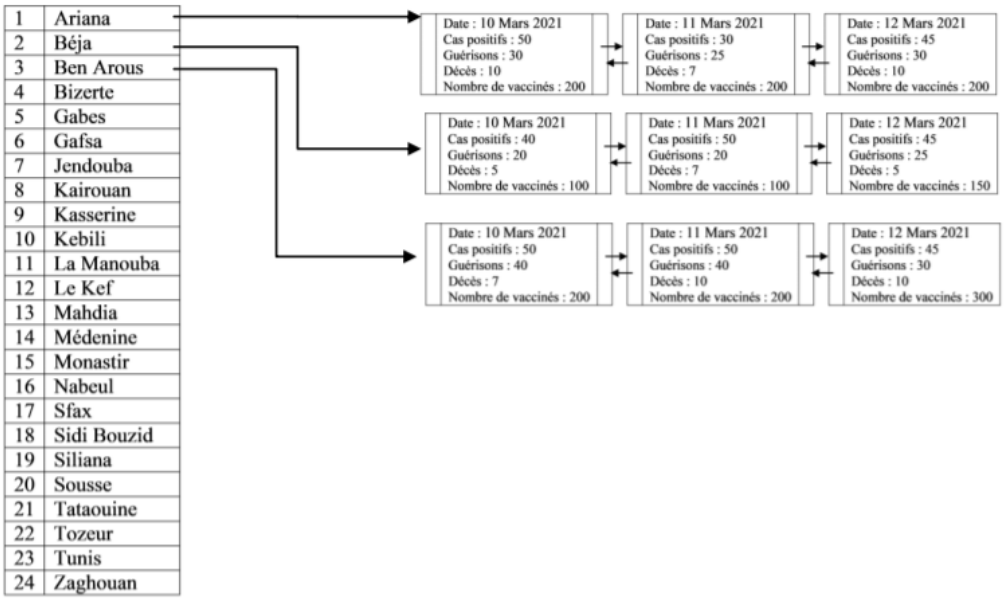

Figure 3 : la structure du programme

Pour utiliser des listes bidirectionnelles on doit créer quatre fichiers :

- « LSTGOV.h» : Déclaration de la structure de données choisie du TDA LISTE par liste doublement chainée
- « LSTSDD.h » : Inclusion du fichier où est déclarée la structure de données adoptée pour réaliser le TDA LISTE

- « LSTPRIM.h» : Déclaration des primitives du TDA LISTE
- « LSTGOV.c » : Définition des primitives pour une réalisation du TDA LISTE

Et quatre fichiers pour l'ELEMENT : « ELTCRN.h», « ELTSDD.h», « ELTPRIM.h», « ELTCRN.c».

```
ELEMENT elementCreer(void);
void elementLire(ELEMENT*);
void elementDetruire(ELEMENT);
void elementAfficher(ELEMENT);
void elementAffecter(ELEMENT*,ELEMENT)
void elementCopier(ELEMENT*,ELEMENT);
int elementComparer(ELEMENT,ELEMENT);
int isValid (d);
```

```
NOEUD noeudCreer(ELEMENT);
void noeudDetruire(NOEUD);
LISTE listeCreer(void);
void listeDetruire(LISTE);
void listeAfficher(LISTE);
int estVide(LISTE);
int estSaturee(LISTE);
int listeTaille(LISTE);
int inserer(LISTE,ELEMENT,int);
int supprimer(LISTE,int);
ELEMENT recuperer(LISTE,int);
LISTE listeCopier(LISTE);
int listeComparer(LISTE,LISTE);
void saisir(LISTE[24],char[24][20]);
void afficheEvol(LISTE[24],char[24][20]);
void supp(LISTE[24]);
void modifier(LISTE[24]);
void affiche(LISTE[24],char[24][20]);
void saisirTxt(LISTE[24]);
```

Figure 4 : les primitives

▪ Dans le programme on a utilisé system (« cls ») plusieurs fois. Il sert à effacer tout ce qu'il y a à l'écran.

▪ La fonction prédéfinit « sleep (nbre de secondes) » permet de faire attendre le programme pendant un certain nombre de secondes avant de continuer à exécuter les instructions suivantes.

1) La procédure saisir :

D'abord on a demandé de donner la date avec un contrôle de saisie pour que la date soit valide et ne se répète pas. Ici on va créer une fonction « exValid » qui rendre 0 si la date est non valide ou la dernière date saisie est inférieur ou égale à la date saisit. La fonction « isValid » vérifie si la date s'écrit sous la forme jj/mm/aaaa en respectant les nombre de jour pour chaque mois.

```
int exValid(d dt,LISTE tab[24])
{
    int succee=1;
    if (!isValid(dt))
        succee=0;
    else
    {
        if (!estVide(tab[0]))
        {
            d dt2=(recuperer(tab[0],tab[0]->lg))->date;
            if (dt.a<dt2.a)
                succee=0;
            else if ((dt.a==dt2.a)&&(dt.m<dt2.m))
                succee=0;
            else if ((dt.a==dt2.a)&&(dt.m==dt2.m)&&(dt.j<=dt2.j))
                succee=0;
        }
    }
    return succee;
}
```

Figure 5 : fonction exValid

Puis on fait une boucle pour parcourir le tableau des éléments. Chaque fois on crée un élément on lit les informations (cas positifs, guérisons, décès, nombre de vaccinés) on affecte la date choisit dans l'élément. Puis on insère tout l'élément dans le tableau. À la fin, l'utilisateur a le choix de réinsérer des informations d'une autre date.

```
void saisir(LISTE tab[24],char noms[24][20])
{
    system("cls");
    char c;
    d dt;
    int i,j;
    ELEMENT e;
    do
    {
        printf("DONNER UNE DATE POUR LA SAISIE : \n");
        printf("JOUR  :  ");scanf("%i",&(dt.j));
        printf("MOIS  :  ");scanf("%i",&(dt.m));
        printf("ANNEE :  ");scanf("%i",&(dt.a));
    } while (exValid(dt,tab)==0);

    for (i=0;i<24;i++)
    {
        e=elementCreer();
        printf("\nDONNEZ LES INFORMATIONS DU %s :  \n",noms[i]);
        elementLire(&e);
        e->date=dt;
        inserer(tab[i],e,(tab[i]->lg)+1);
    }
    do
    {
        printf("\nVOULEZ VOUS ECRIRE LES INFORMATIONS D'UNE AUTRE DATE : ");
        printf("1- OUI     2- NON\n");
        getchar();
        c=getchar();
    } while ((c<'1')||(c>'2'));
    if (c=='1')
        saisir(tab,noms);
    system("cls");
}
```

Figure 6 : procédure saisir

2) La procédure afficheEvol :

Cette procédure permet d'afficher l'évolution de toutes les informations d'un gouvernorat durant toutes les dates.

On demande à l'utilisateur de donner le numéro de gouvernorat et on appelle la procédure prédéfinit « listeAfficher » pour afficher la liste correspondante à ce numéro. À la fin, l'utilisateur peut afficher l'évolution de statistique d'un autre gouvernorat.

```
void afficheEvol(LISTE tab[24],char noms[24][20])
{
    system("cls");
    int ch;
    char c;
    do{
        system("cls");
        printf("1-ARIANA | 2-BEJA | 3-BEN AROUS | 4-BIZERTE | 5-GABES | 6-GAFSA | 7-JENDOUBA | 8-KAIROUAN\n");
        printf("9-KASSERINE | 10-KEBILI | 11-LA MANOUBA | 12-LE KEF | 13-MAHDIA | 14-MEDENINE | 15-MONASTIR | 16-NABEUL\n");
        printf("17-SFAX | 18-SIDI BOUZID | 19-SILIANA | 20-SOUSSE | 21-TATAOUINE | 22-TOZEUR | 23-TUNIS | 24-ZAGHOUAN\n\n");
        printf("CHOISIR UN NUMERO DE GOUVERNORAT :  ");
        scanf("%i",&ch);
    } while ((ch<1)||(ch>24));
    system("cls");
    printf("----------------  %s  ---------------\n\n",noms[ch-1]);
    listeAfficher(tab[ch-1]);
    do
    {
        printf("\n\nVOULEZ VOUS AFFICHER L'EVOLUTION DE STATISTIQUES D'UN AUTRE GOUVERNORAT : ");
        printf("1- OUI     2- NON\n");
        getchar();
        c=getchar();
    } while ((c<'1')||(c>'2'));
    if (c=='1')
        afficheEvol(tab,noms);
    system("cls");
}
```

Figure 7 : procédure afficheEvol

3) La procédure affiche :

Cette procédure permet d'afficher le classement des 24 gouvernorats triés par ordre croissant ou décroissant selon les 4 critères.

Alors, on va d'abord, après la saisie de la date, de lire les priorités de chaque critère et les remplir dans un tableau t de quatre cases.

Puis l'utilisateur choisit l'indice de tri (0 : décroissant/1 : croissant). Ensuite on remplit tous le contenu des listes dans une matrice m.

Pour le tri on a choisi le tri à bulle. Il consiste à comparer répétitivement les éléments consécutifs d'un tableau, et à les permuter lorsqu'ils sont mal triés. Si on trouve une égalité entre deux valeurs on avance au suivant priorité dans le tableau ainsi de suite jusqu'à la quatrième priorité. S'ils sont encore égaux la matrice sera triée par ordre alphabétique selon les noms des gouvernorats. On fait la même chose pour les deux cas croissants ou décroissant.

Par la suite, on parcourt toute la matrice pour l'afficher.

Finalement, l'utilisateur peut afficher les informations d'une autre date en appelant la même procédure.

```
for (j=0;j<23;j++)
{
    if (ord==1)
    {
        if (m[j][t[i]]>m[j+1][t[i]])
        {
            permute(m[j],m[j+1]);
            b=0;
        }
        else if (m[j][t[i]]==m[j+1][t[i]])
        {
            if (m[j][t[i+1]]>m[j+1][t[i+1]])
            {
                permute(m[j],m[j+1]);
                b=0;
            }
            else if (m[j][t[i+1]]==m[j+1][t[i+1]])
            {
                if (m[j][t[i+2]]>m[j+1][t[i+2]])
                {
                    permute(m[j],m[j+1]);
                    b=0;
                }
                else if (m[j][t[i+2]]==m[j+1][t[i+2]])
                {
                    if (m[j][t[i+3]]>m[j+1][t[i+3]])
                    {
                        permute(m[j],m[j+1]);
                        b=0;
                    }

                }
            }
        }
    }
```

Figure 8 : le tri en cas de croissance

4) La procédure saisirTxt :

Cette procédure permet de remplir les listes automatiquement à partir d'un fichier texte préalablement saisi et qui a ce format :

Date	Gouvernorat	Cas positifs	Guérisons	Décès	Nombre de vaccinés
10 Mars 2021	Ariana	50	30	10	200
10 Mars 2021	Béja	40	20	5	100
10 Mars 2021	Ben Arous	50	40	7	200
…					
11 Mars 2021	Ariana	30	25	7	200
11 Mars 2021	Béja	50	20	7	100
11 Mars 2021	Ben Arous	50	40	10	200
….					
12 Mars 2021	Ariana	45	30	10	200
12 Mars 2021	Béja	45	25	5	150
12 Mars 2021	Ben Arous	45	30	10	300

Figure 9 : format de fichier

La fonction va supprimer les anciennes valeurs des listes et les remplacer par les informations du fichier.

D'abord, on va demander à l'utilisateur de saisir le nom de fichier avec un contrôle de saisit pourque ce dernier ne doit être vide ou d'un fichier inexistant. Puis, on supprime le contenu du tableau des listes avec une boucle pour qui va parcourir tous les gouvernorats et une boucle tant que qui va parcourir et supprimer les nœuds de liste. Ensuite, on ouvre le fichier texte avec « FILE * fopen (const char * filename, const char * accessMode) ; ». Cette fonction permet d'ouvrir un flux de caractère basé sur fichier. Vous pouvez choisir entre différents modes d'ouverture du fichier (ouverture en lecture(r), en écriture(w), en ajout(a), ...) en fonction de vos

```
while (fgets(ligne,100,fichier)!=NULL)
```

besoins. La fonction retourne une variable de type file ou Null si le fichier n'existe pas. La fonction « fgets (char* chaine, int nbreDeCaracteresALire, FILE* pointeurSurFichier) lit le fichier ligne par ligne et l'affecte dans une chaine. Pour parcourir le fichier, on utilise une boucle

Les informations dans le fichier sont séparées avec « – » et « / ». Alors, quand on trouve un séparateur on remplit une chaine de caractère (on trouve des chaines pour chaque information : jour, mois, année, numéro de gouvernorat, nbre des cas positifs, nbre des guérisons, nbre de décès et le nombre des vaccinés). On convertit ces chaines en entier et les affecte dans un élément e qui va être inséré dans le tableau des listes.

Finalement, on ferme le fichier et donne la possibilité de remplir les listes à partir d'un autre fichier texte.

5) La procédure modifier :

Cette procédure permet de modifier n'importe quelle information de n'importe quel gouvernorat dans n'importe quelle date. D'abord l'utilisateur donne la date et l'indice de gouvernorat. On récupère les informations qui leur appartiennent et les affecte dans une variable e. On affiche les anciennes informations et demande à l'utilisateur de donner l'indice de l'information qu'il va changer. Selon cet indice on change l'information auxquelles il se réfère dans e. Ensuite on pointe sur le nœud de cette date avec une boucle pour et affecte ce nouvel

élément dans le nœud. Et finalement, l'utilisateur a le choix de modifier d'autres informations.

```
    printf("CHOISIR UN NUMERO DE GOUVERNORAT :  ");
    scanf("%i",&ch);
} while ((ch<1)||(ch>24));
e=recuperer(tab[ch-1],pos);
do
{
    printf("----------------  %s  ---------------\n\n",noms[ch-1]);
    printf("1-CAS POSITIFS : %i\n",e->cp);
    printf("2-GUERISONS : %i\n",e->grs);
    printf("3-DECES : %i\n",e->dec);
    printf("4-NOMBRE DE VACCINES : %i\n",e->nva);
    printf("\nSAISIR LE NUMERO D'INFORMATION QUE VOUS ALLEZ LE MODIFIER :  ");
    scanf("%i",&inf);
} while ((inf<1)||(inf>4));
do {
    printf("\nSAISIR LA NOUVELLE VALEUR :  ");
    scanf("%i",&val);
} while (val<0);
switch (inf)
{
    case 1 : e->cp=val;break;
    case 2 : e->grs=val;break;
    case 3 : e->dec=val;break;
    case 4 : e->nva=val;break;
}
NOEUD p=tab[ch-1]->tete;
for (i=1;i<pos;i++)
{
    p=p->suivant;
}
p->info=e;
```

Figure 10 : procédure modifier

6) La procédure supp :

Cette procédure permet de supprimer le contenu de tous les éléments d'une seule date pour tous les gouvernorats. Le principe est simple, l'utilisateur donne une date avec un contrôle de saisie qui vérifie

l'existence de la date. On affecte dans une variable pos la position de la date dans la liste avec une fonction « posDate ». Cette fonction rend la position de la date s'il elle existe ou 0 sinon. Puis on parcourt les 24 gouvernorats et on supprime l'élément correspondant à cette date. Et finalement, comme d'habitude on donne à l'utilisateur la possibilité de supprimer les informations d'une autre date.

```
void supp(LISTE tab[24])
{
    d dt;
    int i;
    char c;
    system("cls");
    do
    {
        printf("DONNER UNE DATE POUR LA SUPPRESSION : \n");
        printf("JOUR  :  ");scanf("%i",&(dt.j));
        printf("MOIS  :  ");scanf("%i",&(dt.m));
        printf("ANNEE :  ");scanf("%i",&(dt.a));
    } while (posDate(dt,tab[0])==0);
    int pos=posDate(dt,tab[0]);
    for (i=0;i<24;i++)
    {
        supprimer(tab[i],pos);
    }
    printf("LES DONNEES CORRESPONDANTS AU %i/%i/%i SONT SUPPRIMES AVEC SUCCES",dt.j,dt.m,dt.a);
    do
    {
        printf("\n\nVOULEZ VOUS SUPPRIMER LES INFORMATIONS D'UNE AUTRE DATE : ");
        printf("1- OUI     2- NON\n");
        getchar();
        c=getchar();
    } while ((c<'1')||(c>'2'));
    if (c=='1')
        supp(tab);
    system("cls");
}
```

Figure 11 : procédure supp

```
int posDate (d dt,LISTE L)
{
    int i=1,stop=0, pos=0;
    ELEMENT e=elementCreer();
    while ((i<=L->lg)&&(stop==0))
    {
        e=recuperer(L,i);
        if ((e->date.j==dt.j)&&(e->date.m==dt.m)&&(e->date.a==dt.a))
        {
            pos=i;
            stop=1;
        }
        i++;
    }
    return pos;
}
```

Figure 12 : fonction posDate

7) La procédure Quitter :

La fonction quitter permet de stocker les informations situées dans le tableau dans un fichier texte "latestVersion.txt" qui trouve dans le dossier de projet et afficher un message lors de sortie de programme.

```
void quitter(LISTE tab[24])
{
    FILE *fichier;
    fichier=fopen("latestVersion.txt","w");
    ELEMENT e;
    int i,j;
    for (i=0;i<24;i++)
    {
        for (j=1;j<=tab[i]->lg;j++)
        {
            e=elementCreer();
            e=recuperer(tab[i],j);
            fprintf(fichier,"%i/%i/%i-%i-%i-%i-%i-%i\n",e->date.j,e->date.m,e->date.a,i,e->cp,e->grs,e->dec,e->nva);
        }
    }
    fclose(fichier);
    printf("||-------------------------------------------------------------------------------------------------||\n");
    printf("||           LES INFORMATIONS ACTUELLES SONT ENREGISTREES DANS LE FICHIER 'latestVersion.txt'       ||\n");
    printf("||-------------------------------------------------------------------------------------------------||\n");
    printf("||                                        AU REVOIR                                                ||\n");
                                                                                                                \n");
```

Figure 13 : procédure quitter

La fonction « int fprintf (FILE * stream, const char *format, ...) ; » permet d'écrire dans un fichier texte. Puis, on affiche un petit message

8) L'interface graphique :

Pour l'interface graphique on a utilisé un générateur (ASCII Generator) qui convertit les images en des caractères ASCII. On a affiché le logo du covid-19 en début et la carte de Tunisie à la fin du programme.

Printf (R"EOF (message) EOF") nous permet d'afficher d'un message contenant plusieurs lignes.

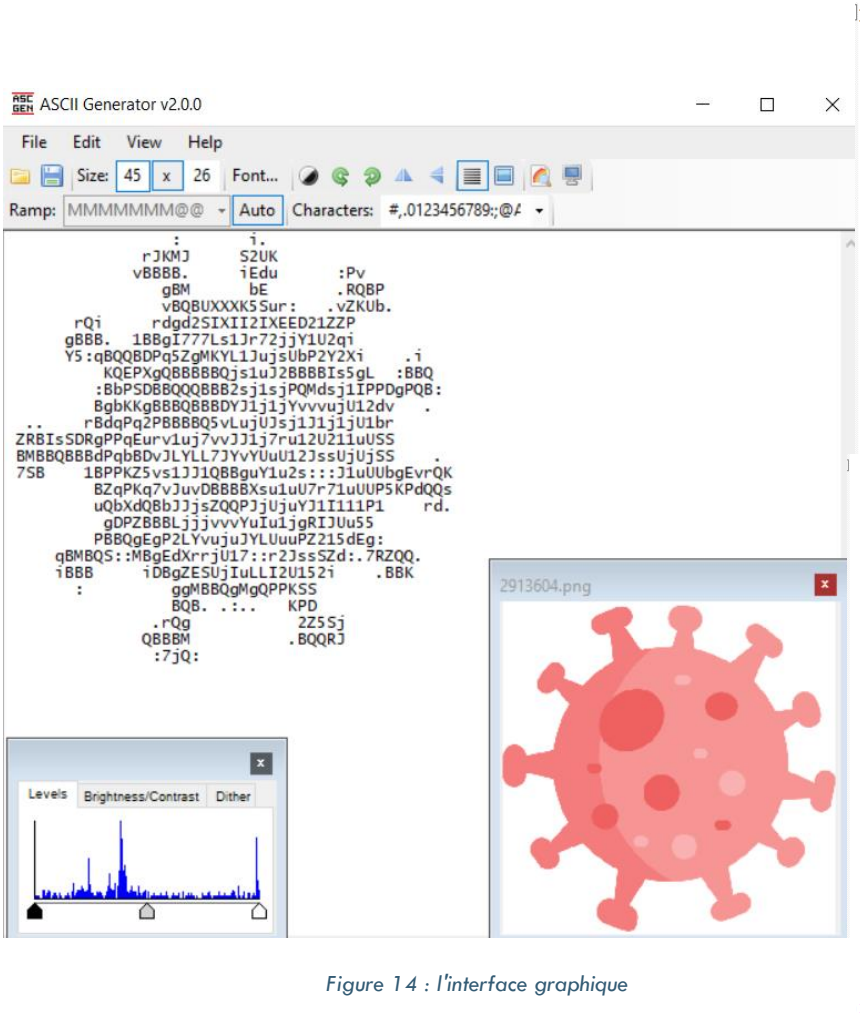

Figure 14 : l'interface graphique