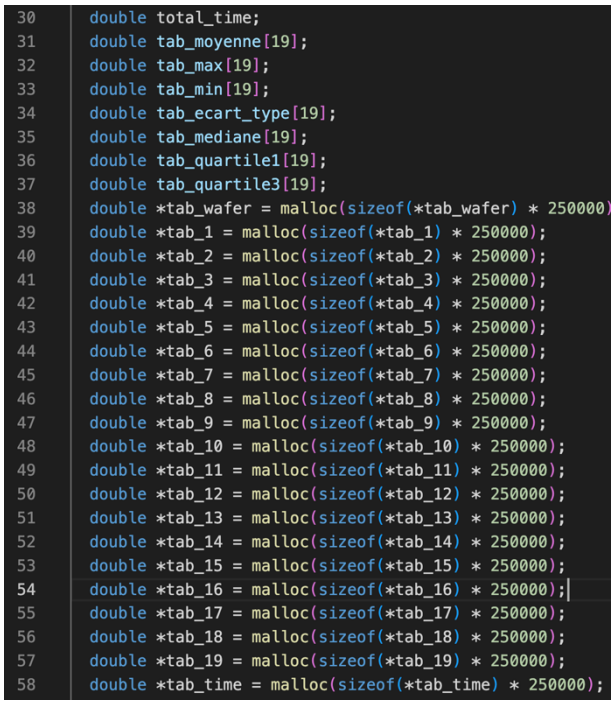
Projet d'Algorithmique et programmation

Jérémy Louis
Gaël-Mehdi Le Lay

Pour compiler nos fichiers, taper : makefile ProjAlgoProg dans le terminal.
Ensuite exécuter : ./ProjAlgoProg dans le terminal.

Question 1

Pour chaque fichier, les données d'une colonne sont extraites dans un tableau, l'extraction des données d'un fichier créé donc 19 tableaux, et cela via la fonction main du fichier ProjAlgoProg.c.



```
30  double total_time;
31  double tab_moyenne[19];
32  double tab_max[19];
33  double tab_min[19];
34  double tab_ecart_type[19];
35  double tab_mediane[19];
36  double tab_quartile1[19];
37  double tab_quartile3[19];
38  double *tab_wafer = malloc(sizeof(*tab_wafer) * 250000)
39  double *tab_1 = malloc(sizeof(*tab_1) * 250000);
40  double *tab_2 = malloc(sizeof(*tab_2) * 250000);
41  double *tab_3 = malloc(sizeof(*tab_3) * 250000);
42  double *tab_4 = malloc(sizeof(*tab_4) * 250000);
43  double *tab_5 = malloc(sizeof(*tab_5) * 250000);
44  double *tab_6 = malloc(sizeof(*tab_6) * 250000);
45  double *tab_7 = malloc(sizeof(*tab_7) * 250000);
46  double *tab_8 = malloc(sizeof(*tab_8) * 250000);
47  double *tab_9 = malloc(sizeof(*tab_9) * 250000);
48  double *tab_10 = malloc(sizeof(*tab_10) * 250000);
49  double *tab_11 = malloc(sizeof(*tab_11) * 250000);
50  double *tab_12 = malloc(sizeof(*tab_12) * 250000);
51  double *tab_13 = malloc(sizeof(*tab_13) * 250000);
52  double *tab_14 = malloc(sizeof(*tab_14) * 250000);
53  double *tab_15 = malloc(sizeof(*tab_15) * 250000);
54  double *tab_16 = malloc(sizeof(*tab_16) * 250000);
55  double *tab_17 = malloc(sizeof(*tab_17) * 250000);
56  double *tab_18 = malloc(sizeof(*tab_18) * 250000);
57  double *tab_19 = malloc(sizeof(*tab_19) * 250000);
58  double *tab_time = malloc(sizeof(*tab_time) * 250000);
```

Le tableau tab_1 correspond à la colonne USAGE_OF_BACKING_FILM, tab_2 correspond à la colonne USAGE_OF_DRESSER, cela jusqu'au tableau 19 qui correspond à la colonne EDGE_AIR_BAG_PRESSURE.

Les valeurs de chaque fichier .csv sont intégrées aux tableaux via la fonction fscanf.
Nous avons utilisé la fonction malloc avec chaque tableau créé car ceux-ci contiennent plus de 10000 valeurs, ce qui force l'utilisation de la fonction malloc pour ne pas avoir de segmentaton fault.

```
67      tab_moyenne[0]=moyenne(tab_1,Nblignes);
68      tab_max[0]=maximum(tab_1,Nblignes);
69      tab_min[0]=minimum(tab_1,Nblignes);
70      tab_ecart_type[0]=ecart_type(tab_1,Nblignes);
71      tab_mediane[0]=mediane(tab_1,Nblignes);
72      tab_quartile1[0]=quartile1(tab_1,Nblignes);
73      tab_quartile3[0]=quartile3(tab_1,Nblignes);
```

Nous calculons une par une les caractéristiques des 19 tableaux (soit 19 fois l'image ci-dessus), nous n'avons pas trouvé d'autre moyen pour faire cela car nous avons essayé de faire un tableau à 2 dimensions mais cela n'a pas fonctionné.

Après avoir créé un tableau par série statistique, nous calculons les valeurs caractéristiques associées à chaque série puis les mettons dans un tableau pour chaque caractéristiques (moyenne,ecart-type…), puis nous écrivons ses tableaux dans un fichier statistique.txt.

```
219  FILE * fichier ; // d é claration d ' un pointeur de type FILE
220  fichier = fopen ( "statistique.txt" ,"w+" );
221  fprintf ( fichier , "Moyenne \n");
222    for ( int i =0; i < 19 ; i ++) // ecriture du tableau
223          {fprintf ( fichier , "%lf \t " , tab_moyenne[i]);}
224          fprintf(fichier,"\n");
225  fprintf ( fichier , "Max \n");
226    for ( int i =0; i < 19 ; i ++) // ecriture du tableau
227          {fprintf ( fichier , "%lf \t" , tab_max[i]);}
228          fprintf(fichier,"\n");
229  fprintf ( fichier , "Min \n");
230    for ( int i =0; i < 19 ; i ++) // ecriture du tableau
231          {fprintf ( fichier , "%lf \t" , tab_min[i]);}
232          fprintf(fichier,"\n");
233  fprintf ( fichier , "Ecart Type \n");
234    for ( int i =0; i < 19 ; i ++) // ecriture du tableau
235          {fprintf ( fichier , "%lf \t" , tab_ecart_type[i]);}
236          fprintf(fichier,"\n");
237  fprintf ( fichier , "Mediane\n");
238    for ( int i =0; i < 19 ; i ++) // ecriture du tableau
239          {fprintf ( fichier , "%lf \t" , tab_mediane[i]);}
240          fprintf(fichier,"\n");
241  fprintf ( fichier , "Quartile1\n");
242    for ( int i =0; i < 19 ; i ++) // ecriture du tableau
243          {fprintf ( fichier , "%lf \t" , tab_quartile1[i]);}
244          fprintf(fichier,"\n");
245  fprintf ( fichier , "Quartile3 \n");
246    for ( int i =0; i < 19 ; i ++) // ecriture du tableau
247          {fprintf ( fichier , "%lf \t" , tab_quartile3[i]);}
248          fprintf(fichier,"\n");
249  printf("les statistiques ont ete enregistrees dans le fichier statistique.txt \n");}
```

Question 2

```
248  int tri_liste2(double tabl[],int k,int Nblignes){
249      Colonne col;
250      col.taille=Nblignes;
251      for (int t=0;t<col.taille;t++){
252      col.tab[t]=tabl[t];
253      }
254      double ecart=ecart_type(col.tab,col.taille);
255      double moy=moyenne(col.tab,col.taille);
256      int fin;
257      int i=0;
258      while(i<col.taille){
259          if (col.tab[i]< moy-k*ecart || col.tab[i]> moy+k*ecart){
260              fin=col.tab[i];
261              col.tab[i]=col.tab[col.taille-1];
262              col.tab[col.taille-1]=fin;
263              col.taille--;
264              i--;
265          }
266      i++;}
267      printf("nous avons enlevée les elements %d \n",(Nblignes-col.taille));
268      for (int t=0;t<col.taille;t++){
269      tabl[t]=col.tab[t];
270      }
271      return col.taille;
272  }
```

```
38  typedef struct{
39  double tab[250000];
40  int taille;} Colonne;
```

Pour cette fonction nous avons crée une structure Colonne qui contient la taille du tableau et le tableau afin de le trier plus facilement. En effet, dans la fonction tri_liste2, nous copions le tableau dans une structure colonne qui est locale pour enfin trier la liste. Pour cela, nous utilisons une boucle while et lorsque qu'une valeur ne respecte pas le critère, nous l'échangeons avec la valeur en fin de tableau puis décrémentons la taille du tableau que nous retournons. Nous décrémentons aussi i dans ce cas afin de vérifier la valeur échanger.

Question 3

Pour construire chaque histogramme, nous avons construit une fonction nommée histogramme. Celle-ci sépare chaque série statistique demandée en 5 bins. Dans le tableau tableau_inter, nous positionnons les 4 indices séparant les 5 bins.
Après avoir calculé les fréquences associées à chaque bins, nous affichons un histogramme fait d'étoiles.

```c
    double m=minimum(tab,Nblignes);
    double M=maximum(tab,Nblignes);
    double inter = (M-m)/5;
    Tri_fusion(tab,0,Nblignes);
    int tableau_inter[4]={0,0,0,0};
    int j=0;
    for(int i=0;i<4;i++){
        while(tab[j]<=m+i*inter)
        {
            tableau_inter[i]=j;
            j=j+1;
        }
    }
```

Pour tracer les histogrammes, nous utilisons 19 fois la fonction histogramme, et cela à la main.

Question 4

Pour la question 4, nous utilisons une fonction nommée lecture_fichierq4. Cette-ci est basée sur le même principe que ce que nous faisons pour la question 1.

Nous utilisons ensuite un tableau local dans lequel nous insérons les valeurs des séries statistiques pour chaque wafer, pour ensuite en calculer les moyennes, et insérer ceci dans un fichier texte nommé StatistiquesWafer.txt. Nous réutilisons ce même tableau local pour chaque wafer.

```c
    int idebut=0;
    int ifin=0;
    double tab_temp[5000];
    double tab_moyenne[19];
    FILE *fp;
    long i = 0;
    int k;
    char fichier[21];
    for(k=0;k<59;k++){
    if (k<10){
        sprintf(fichier,"CMP-training-00%d.csv",k);
    }
    else (sprintf(fichier,"CMP-training-0%d.csv",k));
    //printf("%s \n",fichier);
    fp = fopen(fichier,"r");
    //printf("lecture du fichier numero %d \n",k);
    fscanf(fp,"%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s");
```

Pour cela, nous avons utilisé une fonction une fonction copie_tab, pour copier, dans notre tableau local les valeurs de chaque wafer une par une.

```c
void copie_tab(double tab_deb[],double tab_fin[],int idebut,int ifin){
    for(int i=idebut;i<ifin+1;i++){
        tab_fin[i-idebut]=tab_deb[i];
    }
}
```

Nous calculons la moyenne 19 fois, une par série par wafer, à l’aide du tableau tab_temp.

```
400     while((fscanf(fp, "%*d%*d%*lf%lf%*s%*lf%lf%lf%lf%lf%lf
401     if(wafer_id[i]!=wafer_id[idebut]){
402         ifin =i-1;
403         copie_tab(tab_1,tab_temp,idebut,ifin);
404         tab_moyenne[0]=moyenne(tab_temp,ifin-idebut);
405         copie_tab(tab_2,tab_temp,idebut,ifin);
406         tab_moyenne[1]=moyenne(tab_temp,ifin-idebut);
407         copie_tab(tab_3,tab_temp,idebut,ifin);
408         tab_moyenne[2]=moyenne(tab_temp,ifin-idebut);
409         copie_tab(tab_4,tab_temp,idebut,ifin);
```

Nous écrivons ensuite les valeurs dans le fichier StatistiquesWafer.txt, à l'aide du morceau de code ci-dessous.

```
442         new_fich=fopen("StatistiquesWafer.txt","a");
443         fprintf(new_fich,"Wafer ID :\t%lf\n ",wafer_id[idebut]);
444         fprintf (new_fich, "Moyenne \n");
445         for (int i =0; i < 19 ; i ++){ // ecriture du tableau
446             fprintf (new_fich , "%lf \t " , tab_moyenne[i]);
447             }
448         fprintf(new_fich,"\n");
449         idebut=i;
```