

ARIF Imran
OUTREBON Séraphin
Groupe 108

SAE 1.01 IMPLÉMENTATION D'UN BESOIN CLIENT



TABLE DES MATIÈRES

But du projet.....	2
La fiabilité du projet.....	2
Bilan.....	3
Annexe.....	5

En quoi consiste le projet ? :

Il s'agit de créer un interpréteur de commande qui gère des demandes d'intervention publiées par des opérateurs de fibre optique. Le but est de pouvoir stocker en mémoire plusieurs inscriptions d'entreprises, Missions afin de pouvoir les manipuler en fonction des commandes entrées par l'utilisateur.

Le projet permet à l'utilisateur de :

- S'inscrire en tant qu'Agence, Opérateur ou Intervenant.
- Proposer des missions en tant qu'Opérateur.
- Accepter une mission en tant qu'Agence ou Intervenant à condition de n'avoir pas déjà échoué à celle-ci.
- Effectuer une sous-traitance en tant qu'Agence (c'est-à-dire republier une mission en prenant le rôle d'un Opérateur).
- Effectuer un rapport (retour) d'une mission acceptée en tant qu'Agence ou Intervenant. Celui-ci peut être positif et mettre fin à la mission, ou négatif et donc en recréer une avec un nouveau prix.
- Consulter des missions non attribuées.
- Consulter les détails de n'importe quelle mission.
- Avoir un récapitulatif des statuts des missions de l'entreprise auquel l'utilisateur est rattaché.

Quelles méthodes avons-nous utilisé afin de vérifier la fiabilité de notre programme ?:

Tout d'abord pour vérifier la fiabilité des fonctions et corriger les bugs qu'il y avait, on testait chaque nouvelle fonction que nous venions de créer. C'est-à-dire que à chaque nouvelle fonction codée, on la testait avec un maximum de cas, et une multitude de manière pour éviter d'accumuler les erreurs au fur et à mesure du projet. Tout d'abord la

première méthode utilisée afin de trouver des erreurs était de simplement tester notre programme avec différents cas expliqués dans le cahier des charges, il fallait également faire bien attention à traiter certains cas précis (un exemple serait de bien renvoyer “aucune mission disponible” quand toutes les missions sont déjà acceptées). Nous avons également utilisé lors du développement du programme des asserts afin de mieux tester nos différentes fonctions. Cependant ces deux méthodes de vérification que nous venons de citer ne font que nous montrer qu’il y avait un problème, pour savoir où était le problème et comment le régler, nous avons dû utiliser d’autres méthodes. La première était de mettre des “printf” dans la fonction afin de suivre certaines valeurs importantes dans certaines parties du programme. Mais nous avons aussi utilisé le débogueur présent dans les logiciels “Clion” et “VisualStudio”, cela nous a permis de suivre l’entièreté des valeurs dans le programme et mieux comprendre les problèmes. Une fois le projet terminé et corrigé, nous avons fait une dernière vérification avec les sprint test donné sur moodle.

Bilan du projet :

Concernant les difficultés que nous avons rencontrées durant le développement du projet, la première compliquée était de créer une façon de stocker toutes les informations demandées au fur et à mesure du programme, c’est-à-dire de stocker les noms, les rôles, les prix de mission etc. Finalement nous avons décidé de stocker toutes les informations dans des tableaux. Une autre difficulté qui nous a causé bien des problèmes est les mauvais indices des tableaux. Comme les tableaux commencent à l’indice 0, et que l’affichage de l’id de la mission ou des inscrits commence à partir 1, il fallait souvent faire moins 1 ou plus 1 ce qui a causé beaucoup de confusion à certains endroits ou bien des oublis qui faussent totalement le résultat cherché. Nous avons également eu beaucoup de mal à stocker certains ID dans les cas où

des missions ont été sous-traitées ou bien quand une mission a échoué et qu'il fallait faire en sorte d'empêcher l'utilisateur ayant échoué de re-accepté la mission. Nous avons donc dû créer une structure de mission assez longue afin de tout stocker.

Et cette structure complexe qui est "mission", nous à ensuite causé des problèmes quand nous avons dû tester notre code sur Visual Studio (à la base nous avons codé le projet sur Clion), car elle causait un dépassement de mémoire utilisable. Cependant nous avons réussi à corriger ces bugs en optimisant mieux notre programme.

Concernant la qualité du programme, nous avons réussi à faire l'entièreté des commandes demandées dans le cahier des charges, cependant il reste juste une erreur que nous avons découvert lors du dernier sprint test noté (pendant le dernier cours de tp de SAE qui était dédié à tester nos projet)et que nous n'avons pas pu corriger.


Cependant, le programme pourrait encore bénéficier d'améliorations, par exemple nous pourrions optimiser encore plus les structures créées afin de prendre moins de mémoire ou alors créer des tableaux dynamiques afin d'améliorer la vitesse des fonctions et encore réduire la place en mémoire des tableaux qui n'auront plus à stocker des cases non utilisées .

Annexe programme:

Sprint test atteint: SP:

out sp4 base:

```
exitInscription realisee (1)
Inscription realisee (2)
Inscription realisee (3)
Mission publiee (1)
1 Dupont Orange 30.50 (0)
Acceptation enregistree
Rapport enregistre (2)
2 Dupont Orange 31.72 (0)
Recepteur defectueux
Sous-traitance enregistree (3)
3 Dupont FiberAgency 28.40 (1)
3 Dupont FiberAgency 28.40 (1)
Recepteur defectueux
* attribuees
  2 Dupont Orange 31.72 (0)
* terminees
  1 Dupont Orange 30.50 (0)
* non attribuees
  3 Dupont FiberAgency 28.40 (1)
* realisees
  1 Dupont Orange 30.50 (0)
Inscription realisee (4)
Acceptation enregistree
Rapport enregistre
Aucune mission disponible
* terminees
  1 Dupont Orange 30.50 (0)
  2 Dupont Orange 31.72 (0)
* terminees
  3 Dupont FiberAgency 28.40 (1)
* realisees
  3 Dupont FiberAgency 28.40 (1)
```

>  main.c

out sp4 erreur :

```
exitInscription realisee (1)
Inscription realisee (2)
Inscription realisee (3)
Inscription realisee (4)
Mission publiee (1)
Remuneration incorrecte
Remuneration incorrecte
Entreprise incorrecte
Entreprise incorrecte
Entreprise incorrecte
Entreprise incorrecte
Mission incorrecte
Mission incorrecte
Sous-traitance enregistree (2)
Sous-traitance enregistree (3)
Sous-traitance enregistree (4)
Sous-traitance enregistree (5)
Sous-traitance enregistree (6)
Mission incorrecte
6 Dupont Agence1 10.00 (5)
|
C > main.c
```

Code :

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#pragma warning (disable : 4096 4996)
enum {TAILLE = 30, T_LIGNE = 200, ENTREPRISE = 50, MISSION_TAILLE = 500, SOUS_TRAITANCE = 5, NB_RAPPORT = 4,};
//création d'un Enum qui va contenir les différentes valeurs d'on
nous allons avoir besoin durant le le code
typedef struct { char ROLE[TAILLE]; char NOM[TAILLE]; } Inscription;
```

```

//Création d'une structure Inscription Contenant le rôle de
l'entreprise et le nom
typedef struct { char NOM[TAILLE]; char NOM_MISSION[TAILLE]; double
PRIX; int Attribue; int SOUS[SOUS_TRAITANCE]; int NOMBRE_SOUS; char
RAPPORT[T_LIGNE]; int Echec[ENTREPRISE];int PLACE; int OPE; int
ACC;} Mission;
//Création d'une structure Mission contenant le nom de la mission,
le nom de l'entreprise gérant la mission, le prix de la mission,
Son statut dans attribue (0 -> non attribué, 1 -> attribué, -1 ->
terminée),
//d'une liste de sous traitance (contenant les anciens ID (place
de l'entreprise dans le tableau) des entreprises), noltre
sous-traitance qui compte le nombre de sous-traitance,
// du rapport contenant le rapport de la mission, d'une liste
d'échec contenant les ID des inscrit qui ont échoué à la mission,
de l'indice de la mission dans Place,
//de l'indice de l'opérateur qui possède la mission et de l'indice
de l'utilisateur qui a accepté la mission (prend la valeur de
l'ID de l'utilisateur qui l'accepte et la valeur * (-1)
del'utilisateur quand il l'a réussi)
typedef struct {char SIGNIFICATION[ENTREPRISE]; double MAJORATION;}
Rapport;
//Création d'une structure Rapport contenant  texte coun char
contenant des messages en fonction de leur place, une majoration
exprimé en % en fonction de la place

void constru (Rapport rap[])
{
    unsigned int i = 0;
    strcpy(rap[i].SIGNIFICATION, "Succes");
    i+=1;
    strcpy(rap[i].SIGNIFICATION, "Local non accessible \n");
    rap[i].MAJORATION = 0;
    i+=1;
    strcpy(rap[i].SIGNIFICATION, "Pas de signal dans le boitier
general \n");
    rap[i].MAJORATION = 5.5;
    i+=1;
    strcpy(rap[i].SIGNIFICATION, "Recepteur defectueux \n");
    rap[i].MAJORATION = 4;
}
//Ce programme prends une tablea d'indice Rapport et permet de
construire le tableau de laconsigne, par exemple il met à la
place 0 ddu tableau signification "succes" et rien à la place
Majoration

void arret() {
    printf("Fin du programme");//affichage

```

```

}
//Renvois juste le message de fin du programme

int inscription(Inscription ins[],int i,char b[TAILLE],char
c[TAILLE]) {
    if (strcmp(b,"OP") == 0 || strcmp(b,"AG") == 0 ||
strcmp(b,"IN") == 0) { // regarde si le rôle inscrit est bon
        for (int j = 0; j < ENTREPRISE; ++j) {
            if (strcmp(ins[j].NOM, c) == 0) { //regarde si le nom
enregistré existe déjà, si oui on renvoie une erreur
                printf("Nom incorrect \n");//affichage
                return i;//renvoie du compte
            }
        }
        strcpy(ins[i].ROLE, b);//copy b dans ins[i].Role
        strcpy(ins[i].NOM, c);
        i+=1;//Augmente le compte de 1
        printf("Inscription realisee (%d) \n", i);//affichage
        return i;//renvoie du compte
    }
    else
    {
        printf("Role incorrect \n");//affichage
        return i;
    }
}

int mission(Inscription ins[ENTREPRISE], Mission
miss[MISSION_TAILLE],int i,int j,char b[TAILLE],char c[TAILLE],char
d[TAILLE])
{
    int indice = atoi(b);//transforme le char en int
    double prix = atof(d);//transforme le char en float
    if ((indice < 1 || indice > i) ||
strcmp(ins[indice-1].ROLE,"OP") != 0 )//regarde que le rôle ne
soit pas "OP"
    {
        printf("Identifiant incorrect \n");//affichage
        return j;//renvoie du nombre de mission
    }
    if (prix < 0.01)//regarde que le prix soit pas inférieur ou
égale à 0
    {
        printf("Remuneration incorrect \n");//affichage
        return j;
    }
    strcpy(miss[j].NOM, ins[indice-1].NOM);//copie du nom de
l'entreprise inscrit dans la mission à l'indice j

```



```

    strcpy(miss[j].NOM_MISSION, c); //donne le nom "c" à la nouvelle
mission à l'indice j
    miss[j].PRIX = prix; //donne le prix de la mission à l'indice j
    miss[j].Attribue = 0; //donne comme attribue 0 à la mission à
l'indice j
    strcpy(miss[j].RAPPORT, ""); //donne un rapport vide
    for (unsigned int p = 0 ; p < SOUS_TRAITANCE ; ++p) //initialise
le tableau de sous-traitance
    {
        miss[j].SOUS[p] = 0;
    }
    miss[j].NOMBRE_SOUS = 0;
    for (unsigned int k = 0 ; k < ENTREPRISE; ++k) //initialise le
tableau d'echec
    {
        miss[j].Echec[k] = 0;
    }
    miss[j].PLACE = j+1; //donne la place dans le tableau + 1
    miss[j].OPE = indice;
    miss[j].ACC = 0;
    j+=1; //augmente le nombre de mission avec le nouvel ajout
    printf("Mission publiee (%d) \n", j); //affichage
    return j; //renvois du nombre de mission
}

void consultation(int nb_miss, Mission miss[])
{
    if (nb_miss == 0) //regarde qu'il y est bien des missions
enregistrées
    {
        printf("Aucune mission disponible \n"); //affichage
    }
    else
    {
        int a = 1; //booléen qui indique si parmi toutes les
missions inscrites, il y a des missions non attribuées
        for (unsigned int i = 0; i < nb_miss ; ++i)
        {
            if (miss[i].Attribue == 0) { //regarde si la mission est
non attribué
                printf("%d %s %s %.2f (%d) \n", i+1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS); //affichage
                a = 0;
            }
        }
        if (a == 1)
            printf("Aucune mission disponible \n"); //affichage
    }
}

```

```

    }

}

void detail(Mission miss[], int nb_miss ,char b[TAILLE])
{
    int indice = atoi(b);
    if (indice < 0 || indice > nb_miss)//regarde si l'indice
indiqué existe
    {
        printf("Identifiant incorrect \n");//affichage
    }
    else
    {
        printf("%d %s %s %.2f (%d)
\n",indice,miss[indice-1].NOM_MISSION,miss[indice-1].NOM
,miss[indice-1].PRIX,miss[indice-1].NOMBRE_SOUS);
        printf("%s",miss[indice-1].RAPPORT);//affichage
    }
}

void acceptation(Inscription ins[], Mission miss[],char b[TAILLE],
char c[TAILLE], int nb_ins, int nb_miss )
{
    int ope = atoi(b);
    int indice = atoi(c);
    int echec = 0;//booleen qui regarde si dans le tableau d'echec
de la mission, l'indice de l'utilisateur souhaitant accepté la
mission existe
    for (unsigned int k = 0; k < ENTREPRISE;++k)
    {
        if (miss[indice-1].Echec[k] == ope) {
            echec = 1;
        }
    }
    if (ope < 1 || ope > nb_ins || strcmp(ins[ope-1].ROLE,"OP") ==
0 || echec != 0 )//regarde si l'opérateur donné n'existe pas,
qu'il n'a pas comme rôle "OP"
    {
        printf("Entreprise incorrecte \n");//affichage
    }
    else if (indice < 1 || indice > nb_miss ||
miss[indice-1].Attribue != 0)//regarde si l'indice de la mission
donné n'existe pas et si l'utilisateur a déjà échoué la mission
    {
        printf("Mission incorrecte \n");//affichage
    }
}

```

```

else
{
    printf("Acceptation enregistree \n");//affichage
    miss[indice-1].Attribue = 1;//donne à l'attribue de la
mission la valeur 1
    miss[indice-1].ACC = ope;//donne l'indice de l'utilisateur
à la mission accepté
}
}

int sous_traitance(Inscription ins[],Mission miss[],char
b[TAILLE], char c[TAILLE],char d[TAILLE],int nb_miss, int nb_ins)
{
    int ope = atoi(b);
    int indice = atoi(c);
    double prix = atof(d);
    if (strcmp(ins[ope-1].ROLE,"AG") == 1 || ope < 1 || ope >
nb_ins)//regarde si l'entrepise qui souhaite sous-traité est pas
"AG" et si l'opérateur donné n'existe pas
    {
        printf("Entreprise incorrecte \n");//affichage
        return nb_miss;//renvois le nombre de mission
    }
    else if (prix < 0.01)//regarde si le prix donné est incorrecte
    {
        printf("Remuneration incorrecte \n");//affichage
        return nb_miss;//renvoie le nombre de mission
    }
    else if (miss[indice-1].NOMBRE_SOUS == 5 || indice < 1 ||
indice > nb_miss || miss[indice-1].Attribue != 0 )//regarde si la
mission a déjà été sous-traité 5 fois, si la mission est déjà
accepté, si la mission n'existe pas
    {
        printf("Mission incorrecte \n");//affichage
        return nb_miss;//renvoie le nombre de mission
    }
    else
    {
        miss[nb_miss] = miss[indice - 1];//copie la mission
sous-traité dans une nouvelle mission à l'indie nombre de mission
        miss[indice-1].Attribue = 1;//on attribue la mission qui a
été copié la valeur 1 pour dire qu'elle est terminé
        strcpy(miss[nb_miss].NOM, ins[ope - 1].NOM);//On change le
nom d'entreprise de la nouvelle mission sous-traité par
l'utilisateur
        miss[nb_miss].PRIX = prix;//on change le prix de la mission
sous-traité
    }
}

```

```

        miss[nb_miss].SOUS[miss[indice-1].NOMBRE_SOUS] =
miss[indice - 1].PLACE;//on inscript la place de la mission copié
dans le tableau sous-traitance de la nouvelle mission en fonction
du nombre de sous-traitance
        miss[nb_miss].NOMBRE_SOUS = miss[indice-1].NOMBRE_SOUS +
1;//on ajoute 1 au nombre de sous-traitance de la nouvelle mission
        miss[nb_miss].OPE = ope ;//on change l'opérateur de la
nouvelle mission par celui de l'utilisateur ayant fait la
sous-traitance
        miss[nb_miss].PLACE = nb_miss+1;//on change la place de la
nouvelle mission
        nb_miss += 1;//on rajoute 1 au compteur de mission
        printf("Sous-traitance enregistrée (%d) \n",
nb_miss);//affichage
        return nb_miss;//on renvoie le nombre de mission
    }
}

int rapport(Mission miss[], Inscription ins[], Rapport rap[], char
b[], char c[], int nb_miss, int nb_ins)
{
    int indice = atoi(b);
    int code = atoi(c);
    if (indice < 1 || indice > nb_miss || miss[indice-1].Attribue <
1)//regarde si l'indice donné n'existe pas et si la mission de
l'indice i n'est pas attribué
    {
        printf("Mission incorrecte \n");//affichage
        return nb_miss;//retourne le nombre de missions
    }
    else if (code < 0 || code > NB_RAPPORT-1)//regarde si le code
donné est incorrecte
    {
        printf("Code de retour incorrect \n");//affichage
        return nb_miss;//retourne le nombre de missions
    }
    else
    {
        if (code == 0)//Si le code de la mission est 0, c'est à
dire si la mission est réussie
        {
            printf("Rapport enregistré \n");//affichage
            miss[indice-1].Attribue = -1;//on donne la valeur -1
pour signifier que la mission est terminée
            strcpy(miss[indice - 1].RAPPORT,
rap[code].SIGNIFICATION);//on met un rapport dans la mission (ici
"succès")

```

```

        miss[indice-1].ACC = miss[indice-1].ACC*(-1); //on fait
        fois *1 ACC de la mission (ça servira pour recapitulatif afin de
        savoir les missions réalisé)
        for (unsigned int i = 0; i <
miss[indice-1].NOMBRE_SOUS; ++i) //ici on parcour toutes les
        sous-traitances y compris la mission originale pour toute leur
        donné la valeur -1 à "attribue" pour qu'elles soient terminées
        {
            int l = miss[indice - 1].SOUS[i];
            miss[l-1].Attribue = -1;
            strcpy(miss[indice - 1].RAPPORT,
rap[code].SIGNIFICATION);
        }
        return nb_miss; //retourne le nombre de missions
    }
    else //Si jamais la mission a échoué
    {
        unsigned int j = 0;
        while(j < ENTREPRISE && miss[indice-1].Echec[j] != 0
) //Ici on récupère le dernière indice du tableau d'échec où
        quelque chose y est inscrit et on fait plus 1, cela sert à savoir
        où on va insérer l'entreprise qui vient d'échouer
            j+=1;
        miss[indice - 1].Echec[j] = miss[indice - 1].ACC; //on
        inscrit l'indice de l'opérateur dans le tableau d'échec de la
        mission à l'indice j

        strcat(miss[indice-1].RAPPORT, rap[code].SIGNIFICATION); //on ajoute
        le rapport
        strcpy(miss[indice - 1].RAPPORT,
rap[code].SIGNIFICATION); //on ajoute le rapport
        miss[nb_miss] = miss[indice - 1]; //on créer une
        nouvelle mission copié de celle qui à échoué
        miss[indice - 1].Attribue = -1; //on attribue à la
        mission qui a échoué la valeur -1 pour signifier qu'elle est
        terminée

        miss[indice-1].ACC = miss[indice-1].ACC*(-1);
        miss[nb_miss].ACC = 0;
        miss[nb_miss].Attribue = 0; //on donne à l'attribue de
        la nouvelle mission la valeur 0 pour signifier qu'elle n'est pas
        accepté

        miss[nb_miss].PRIX = miss[nb_miss].PRIX +
(miss[nb_miss].PRIX * (rap[code].MAJORATION / 100)); //on majore la
        prix en fonction du code d'erreur
        miss[nb_miss].PLACE = nb_miss+1; //on donne la place de
        la nouvelle mission
        nb_miss += 1; //on augmente de 1 le compteur de missions
    }
}

```

```

        printf("Rapport enregistre (%d)
\n",nb_miss);//affichage
        return nb_miss;//retourne le nombre de missions
    }
}

void recapitulatif (Inscription ins[],Mission miss[], char b[],
int nb_ins, int nb_miss) {
    int ope = atoi(b);
    if (ope < 1 || ope > nb_ins)//regarde si l'entreprise donnée
est incorrecte
        printf("Entreprise incorrecte \n");
    int identifiant;
    if (strcmp(ins[ope - 1].ROLE, "OP") == 0)
        identifiant = 1;
    else if (strcmp(ins[ope - 1].ROLE, "AG") == 0)
        identifiant = 2;
    else
        identifiant = 3;
    int a0 = 0;
    int a1 = 0;
    int a2 = 0;
    int a3 = 0;
    int a4 = 0;//initialise des booléen qui servent à savoir si une
entreprise ou utilisateur est renseigné dans une étape afin de
rajouté un message au début
    if (identifiant == 1 || identifiant == 2) {
        for (unsigned int i = 0; i < nb_miss; ++i) {
            if (miss[i].Attribue == 0 && miss[i].OPE == ope)//on
cherche les missions non attriubées de l'utilisateur
            {
                if (a0 == 0) {
                    printf("* non attribuees \n");
                    a0 = 1;
                }
                printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
                    miss[i].NOMBRE_SOUS);
            }
        }

        for (unsigned int i = 0; i < nb_miss; ++i) {
            if (miss[i].Attribue == 1 && miss[i].OPE == ope)//on
cherche les missions attriubées de l'utilisateur
            {
                if (a1 == 0) {
                    printf("* attribuees \n");

```

```

        a1 = 1;
    }
    printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS);
    }

}

for (unsigned int i = 0; i < nb_miss; ++i) {
    if (miss[i].Attribue == -1 && miss[i].OPE == ope) //on
cherche les missions terminées de l'utilisateur
    {
        if (a2 == 0) {
            printf("* terminees \n");
            a2 = 1;
        }
        printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS);
    }

}

}

if (identifiant == 2 || identifiant == 3)
{
    for (unsigned int i = 0; i < nb_miss; ++i) {
        if (miss[i].ACC == ope) //on regarde les missions
accepté mais pas terminé de l'utilisateur
        {
            if (a3 == 0) {
                printf("* a realiser \n");
                a3 = 1;
            }
            printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS);
        }

    }

    for (unsigned int i = 0; i < nb_miss; ++i) {
        if (miss[i].ACC == ope * (-1)) ///on regarde les
missions réalisé de l'utilisateur
        {
            if (a4 == 0) {
                printf("* realisees \n");
                a4 = 1;
            }
        }
    }
}

```

```

        printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS);
    }

}

}

}

}

int main() {
    Inscription ins[ENTREPRISE]; //on créer un tableau d'inscription
    Mission miss[MISSION_TAILLE]; //on créer un tableau de mission
    Rapport rap[NB_RAPPORT]; //on créer un tableau de rapport
    constru(rap); //on construit le tableau rapport
    int nb_ins = 0; //on initialise le nombre d'inscrit à 0
    int nb_miss = 0; //on initialise le nombre de mission à 0
    char a[TAILLE];
    char b[TAILLE];
    char c[TAILLE];
    char d[TAILLE]; //on initialise 4 variable char que l'on
    utilisera plus tard
    while (strcmp(a, "exit") != 0) { //boucle qui continu tant que
    l'utilisateur n'a pas écrit "exit"
        scanf("%s", a); //on récupère le premier mot (en fonction des
    espaces)
        if (strcmp(a, "inscription") == 0) //si l'utilisateur à tapé
    "inscription"
        {
            scanf("%s", b); //on récupère le deuxième mot (en fonction
    des espaces)
            scanf("%s", c); //on récupère le troisième mot (en fonction
    des espaces)
            nb_ins = inscription(ins, nb_ins, b, c); //on appelle la
    fonction inscription en lui donnant le tableau d'inscrit, le
    nombre d'inscrit et les valeurs b et c
        }
        else if (strcmp(a, "mission") == 0) //si l'utilisateur a tapé
    "mission"
        {
            scanf("%s", b);
            scanf("%s", c);
            scanf("%s", d);
            nb_miss = mission(ins, miss, nb_ins, nb_miss, b, c, d); //on
    appelle la fonction mission en lui donnant le tableau d'inscrit,
    le tableau de mission, le nombre d'inscrits, le nombre de
    missions, b, c et d
        }
    }
}

```



```

    else if (strcmp (a,"consultation") == 0)//si l'utilisateur a
tapé "consultation"
    {
        consultation(nb_miss,miss);//on appelle la fonction
consultation en lui donnant le nombre de mission et le tableau de
mission
    }
    else if (strcmp (a,"detail") == 0)//si l'utilisateur a tapé
"detail"
    {
        scanf("%s",b);
        detail(miss, nb_miss, b);//on appelle la fonction detail en
lui donnant le tableau de mission, le nombre de mission et b
    }
    else if (strcmp (a,"acceptation") == 0)//si l'utilisateur a
tapé "acceptation"
    {
        scanf("%s",b);
        scanf("%s",c);
        acceptation(ins, miss, b, c, nb_ins, nb_miss);//on appelle
la fonction acceptation en lui donnant le tableau d'inscrit, le
tableau de mission, le nombre d'inscrits, le nombre de missions, b
et c
    }
    else if (strcmp (a,"sous-traitance") == 0)//si l'utilisateur a
tapé "sous-traitance"
    {
        scanf("%s",b);
        scanf("%s",c);
        scanf("%s",d);
        nb_miss = sous_traitance(ins, miss, b, c, d, nb_miss,
nb_ins);//on appelle la fonction sous-traitance en lui donnant le
tableau d'inscrit, le tableau de mission, le nombre d'inscrits, le
nombre de missions, b, c et d
    }
    else if (strcmp (a,"rapport") == 0)//si l'utilisateur a tapé
"rapport"
    {
        scanf("%s",b);
        scanf("%s",c);
        nb_miss = rapport(miss, ins, rap, b, c,
nb_miss,nb_ins);//on appelle la fonction rapport en lui donnant le
tableau de mission, le tableau d'inscrit, le tableau de rapport,
le nombre de mission, le nombre d'inscrit, b et c
    }
    else if (strcmp (a,"recapitulatif") == 0)//si l'utilisateur a
tapé "recapitulatif"
    {

```

```

        scanf("%s",b);
        recapitulatif(ins,miss, b, nb_ins, nb_miss);//on appelle la
fonction recapitulatif en lui donnant le tableau de mission, le
nombre d'inscrits, le nombre de missions et b
    }
}
    arret();//on affiche l'arret
}
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#pragma warning (disable : 4096 4996)
enum {TAILLE = 30, T_LIGNE = 200, ENTREPRISE = 50, MISSION_TAILLE =
500, SOUS_TRAITANCE = 5, NB_RAPPORT = 4,};
//création d'un Enum qui va contenir les différentes valeurs d'on
nous allons avoir besoin durant le le code
typedef struct { char ROLE[TAILLE]; char NOM[TAILLE];} Inscription;
//Création d'une structure Inscription Contenant le rôle de
l'entreprise et le nom
typedef struct { char NOM[TAILLE]; char NOM_MISSION[TAILLE]; double
PRIX; int Attribue; int SOUS[SOUS_TRAITANCE]; int NOMBRE_SOUS; char
RAPPORT[T_LIGNE]; int Echec[ENTREPRISE];int PLACE; int OPE; int
ACC;} Mission;
//Création d'une structure Mission contenant le nom de la mission,
le nom de l'entreprise gérant la mission, le prix de la mission,
Son statut dans attribue (0 -> non attribué, 1 -> attribué, -1 ->
terminée),
//d'une liste de sous traitance (contenant les anciens ID (place
de l'entreprise dans le tableau) des entreprises), nolbre
sous-traitance qui compte le nombre de sous-traitance,
// du rapport contenant le rapport de la mission, d'une liste
d'échec contenant les ID des inscrit qui ont échoué à la mission,
de l'indice de la mission dans Place,
//de l'indice de l'opérateur qui possède la mission et de l'indice
de l'utilisateur qui a accepté la mission (prend la valeur de
l'ID de l'utilisateur qui l'accepte et la valeur * (-1)
del'utilisateur quand il l'a réussi)
typedef struct {char SIGNIFICATION[ENTREPRISE]; double MAJORATION;}
Rapport;
//Création d'une structure Rapport contenant  texte coun char
contenant des messages en fonction de leur place, une majoration
exprimé en % en fonction de la place

void constru (Rapport rap[])
{
    unsigned int i = 0;
    strcpy(rap[i].SIGNIFICATION, "Succes");
    i+=1;

```

```

    strcpy(rap[i].SIGNIFICATION, "Local non accessible \n");
    rap[i].MAJORATION = 0;
    i+=1;
    strcpy(rap[i].SIGNIFICATION, "Pas de signal dans le boitier
general \n");
    rap[i].MAJORATION = 5.5;
    i+=1;
    strcpy(rap[i].SIGNIFICATION, "Recepteur defectueux \n");
    rap[i].MAJORATION = 4;
}
//Ce programme prends une tablea d'indice Rapport et permet de
construire le tableau de laconsigne, par exemple il met à la
place 0 ddu tableau signification "succes" et rien à la place
Majoration

int inscription(Inscription ins[],int i,char b[TAILLE],char
c[TAILLE]) {
    if (strcmp(b,"OP") == 0 || strcmp(b,"AG") == 0 ||
strcmp(b,"IN") == 0) { // regarde si le rôle inscript est bon
        for (int j = 0; j < ENTREPRISE; ++j) {
            if (strcmp(ins[j].NOM, c) == 0) { //regarde si le nom
enregistré existe déjà, si oui on renvoie une erreur
                printf("Nom incorrect \n");//affichage
                return i;//renvoie du conteur
            }
        }
        strcpy(ins[i].ROLE, b);//copy b dans ins[i].Role
        strcpy(ins[i].NOM, c);
        i+=1;//Augmente le conteur de 1
        printf("Inscription realisee (%d) \n", i);//affichage
        return i;//renvoie du conteur
    }
    else
    {
        printf("Role incorrect \n");//affichage
        return i;
    }
}

int mission(Inscription ins[ENTREPRISE], Mission
miss[MISSION_TAILLE],int i,int j,char b[TAILLE],char c[TAILLE],char
d[TAILLE])
{
    int indice = atoi(b);//transforme le char en int
    double prix = atof(d);//transforme le char en float
    if ((indice < 1 || indice > i) ||
strcmp(ins[indice-1].ROLE,"OP") != 0 )//regarde que le rôle ne
soit pas "OP"

```

```

{
    printf("Identifiant incorrect \n");//affichage
    return j;//renvois du nombre de mission
}
if (prix < 0.01)//regarde que le prix soit pas inférieur ou
égale à 0
{
    printf("Remuneration incorrect \n");//affichage
    return j;
}
strcpy(miss[j].NOM, ins[indice-1].NOM);//copie du nom de
l'entreprise inscrit dans la mission à l'indice j
strcpy(miss[j].NOM_MISSION, c);//donne le nom "c" à la nouvelle
mission à l'indice j
miss[j].PRIX = prix;//donne le prix de la mission à l'indice j
miss[j].Attribue = 0;//donne comme attribue 0 à la mission à
l'indice j
strcpy(miss[j].RAPPORT, "");//donne un rapport vide
for (unsigned int p = 0 ; p < SOUS_TRAITANCE ; ++p)//initialise
le tableau de sous-traitance
{
    miss[j].SOUS[p] = 0;
}
miss[j].NOMBRE_SOUS = 0;
for (unsigned int k = 0 ; k < ENTREPRISE; ++k)//initialise le
tableau d'echec
{
    miss[j].Echec[k] = 0;
}
miss[j].PLACE = j+1;//donne la place dans le tableau + 1
miss[j].OPE = indice;
miss[j].ACC = 0;
j+=1;//augmente le nombre de mission avec le nouvel ajout
printf("Mission publiee (%d) \n",j);//affichage
return j;//renvois du nombre de mission
}

void consultation(int nb_miss, Mission miss[])
{
    if (nb_miss == 0)//regarde qu'il y est bien des missions
enregistrées
    {
        printf("Aucune mission disponible \n");//affichage
    }
    else
    {
        int a = 1;//booléen qui indique si parmi toutes les
missions inscrites, il y a des missions non attribuées

```

```

        for (unsigned int i = 0; i < nb_miss ; ++i)
        {
            if (miss[i].Attribue == 0) { //regarde si la mission est
non attribué
                printf("%d %s %s %.2f (%d) \n", i+1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS); //affichage
                a = 0;
            }
        }
        if (a == 1)
            printf("Aucune mission disponible \n"); //affichage
    }
}

void detail(Mission miss[], int nb_miss , char b[TAILLE])
{
    int indice = atoi(b);
    if (indice < 0 || indice > nb_miss) //regarde si l'indice
indiqué existe
    {
        printf("Identifiant incorrect \n"); //affichage
    }
    else
    {
        printf("%d %s %s %.2f (%d)
\n", indice, miss[indice-1].NOM_MISSION, miss[indice-1].NOM
, miss[indice-1].PRIX, miss[indice-1].NOMBRE_SOUS);
        printf("%s", miss[indice-1].RAPPORT); //affichage
    }
}

void acceptation(Inscription ins[], Mission miss[], char b[TAILLE],
char c[TAILLE], int nb_ins, int nb_miss )
{
    int ope = atoi(b);
    int indice = atoi(c);
    int echec = 0; //booleen qui regarde si dans le tableau d'echec
de la mission, l'indice de l'utilisateur souhaitant accepté la
mission existe
    for (unsigned int k = 0; k < ENTREPRISE; ++k)
    {
        if (miss[indice-1].Echec[k] == ope) {
            echec = 1;
        }
    }
}

```

```

    if (ope < 1 || ope > nb_ins || strcmp(ins[ope-1].ROLE,"OP") ==
0 || echec != 0 )//regarde si l'opérateur donné n'existe pas,
qu'il n'a pas comme rôle "OP"
    {
        printf("Entreprise incorrecte \n");//affichage
    }
    else if (indice < 1 || indice > nb_miss ||
miss[indice-1].Attribue != 0)//regarde si l'indice de la mission
donné n'existe pas et si l'utilisateur a déjà échoué la mission
    {
        printf("Mission incorrecte \n");//affichage
    }
    else
    {
        printf("Acceptation enregistree \n");//affichage
        miss[indice-1].Attribue = 1;//donne à l'attribue de la
mission la valeur 1
        miss[indice-1].ACC = ope;//donne l'indice de l'utilisateur
à la mission accepté
    }
}

int sous_traitance(Inscription ins[],Mission miss[],char
b[TAILLE], char c[TAILLE],char d[TAILLE],int nb_miss, int nb_ins)
{
    int ope = atoi(b);
    int indice = atoi(c);
    double prix = atof(d);
    if (strcmp(ins[ope-1].ROLE,"AG") == 1 || ope < 1 || ope >
nb_ins)//regarde si l'entreprise qui souhaite sous-traité est pas
"AG" et si l'opérateur donné n'existe pas
    {
        printf("Entreprise incorrecte \n");//affichage
        return nb_miss;//renvois le nombre de mission
    }
    else if (prix < 0.01)//regarde si le prix donné est incorrecte
    {
        printf("Remuneration incorrecte \n");//affichage
        return nb_miss;//renvoie le nombre de mission
    }
    else if (miss[indice-1].NOMBRE_SOUS == 5 || indice < 1 ||
indice > nb_miss || miss[indice-1].Attribue != 0 )//regarde si la
mission a déjà été sous-traité 5 fois, si la mission est déjà
accepté, si la mission n'existe pas
    {
        printf("Mission incorrecte \n");//affichage
        return nb_miss;//renvoie le nombre de mission
    }
}

```

```

else
{
    miss[nb_miss] = miss[indice - 1]; //copie la mission
    sous-traité dans une nouvelle mission à l'indice nombre de mission
    miss[indice-1].Attribue = 1; //on attribue la mission qui a
    été copié la valeur 1 pour dire qu'elle est terminée
    strcpy(miss[nb_miss].NOM, ins[oep - 1].NOM); //On change le
    nom d'entreprise de la nouvelle mission sous-traité par
    l'utilisateur
    miss[nb_miss].PRIX = prix; //on change le prix de la mission
    sous-traité
    miss[nb_miss].SOUS[miss[indice-1].NOMBRE_SOUS] =
    miss[indice - 1].PLACE; //on inscrit la place de la mission copié
    dans le tableau sous-traitance de la nouvelle mission en fonction
    du nombre de sous-traitance
    miss[nb_miss].NOMBRE_SOUS = miss[indice-1].NOMBRE_SOUS +
    1; //on ajoute 1 au nombre de sous-traitance de la nouvelle mission
    miss[nb_miss].OPE = ope ; //on change l'opérateur de la
    nouvelle mission par celui de l'utilisateur ayant fait la
    sous-traitance
    miss[nb_miss].PLACE = nb_miss+1; //on change la place de la
    nouvelle mission
    nb_miss += 1; //on rajoute 1 au compteur de mission
    printf("Sous-traitance enregistrée (%d) \n",
    nb_miss); //affichage
    return nb_miss; //on renvoie le nombre de mission

}
}

int rapport(Mission miss[], Inscription ins[], Rapport rap[], char
b[], char c[], int nb_miss, int nb_ins)
{
    int indice = atoi(b);
    int code = atoi(c);
    if (indice < 1 || indice > nb_miss || miss[indice-1].Attribue <
1) //regarde si l'indice donné n'existe pas et si la mission de
l'indice i n'est pas attribué
    {
        printf("Mission incorrecte \n"); //affichage
        return nb_miss; //retourne le nombre de missions
    }
    else if (code < 0 || code > NB_RAPPORT-1) //regarde si le code
donné est incorrecte
    {
        printf("Code de retour incorrect \n"); //affichage
        return nb_miss; //retourne le nombre de missions
    }
}

```

```

else
{
    if (code == 0) // Si le code de la mission est 0, c'est à
dire si la mission est réussie
    {
        printf("Rapport enregistré \n"); // affichage
        miss[indice-1].Attribue = -1; // on donne la valeur -1
pour signifier que la mission est terminée
        strcpy(miss[indice - 1].RAPPORT,
rap[code].SIGNIFICATION); // on met un rapport dans la mission (ici
"succès")
        miss[indice-1].ACC = miss[indice-1].ACC*(-1); // on fait
fois *1 ACC de la mission (ça servira pour récapitulatif afin de
savoir les missions réalisées)
        for (unsigned int i = 0; i <
miss[indice-1].NOMBRE_SOUS; ++i) // ici on parcourt toutes les
sous-traitances y compris la mission originale pour toute leur
donné la valeur -1 à "attribue" pour qu'elles soient terminées
        {
            int l = miss[indice - 1].SOUS[i];
            miss[l-1].Attribue = -1;
            strcpy(miss[indice - 1].RAPPORT,
rap[code].SIGNIFICATION);
        }
        return nb_miss; // retourne le nombre de missions
    }
    else // Si jamais la mission a échoué
    {
        unsigned int j = 0;
        while(j < ENTREPRISE && miss[indice-1].Echec[j] != 0
) // Ici on récupère le dernière indice du tableau d'échec où
quelque chose y est inscrit et on fait plus 1, cela sert à savoir
où on va insérer l'entreprise qui vient d'échouer
            j++;
        miss[indice - 1].Echec[j] = miss[indice - 1].ACC; // on
inscrit l'indice de l'opérateur dans le tableau d'échec de la
mission à l'indice j

        strcat(miss[indice-1].RAPPORT, rap[code].SIGNIFICATION); // on ajoute
le rapport
        strcpy(miss[indice - 1].RAPPORT,
rap[code].SIGNIFICATION); // on ajoute le rapport
        miss[nb_miss] = miss[indice - 1]; // on crée une
nouvelle mission copiée de celle qui a échoué
        miss[indice - 1].Attribue = -1; // on attribue à la
mission qui a échoué la valeur -1 pour signifier qu'elle est
terminée
        miss[indice-1].ACC = miss[indice-1].ACC*(-1);

```



```

        miss[nb_miss].ACC = 0;
        miss[nb_miss].Attribue = 0;//on donne à l'attribue de
la nouvelle mission la valeur 0 pour signifier qu'elle n'est pas
accepté
        miss[nb_miss].PRIX = miss[nb_miss].PRIX +
(miss[nb_miss].PRIX * (rap[code].MAJORATION / 100));//on majore la
prix en fonction du code d'erreur
        miss[nb_miss].PLACE = nb_miss+1;//on donne la place de
la nouvelle mission
        nb_miss += 1;//on augmente de 1 le compteur de missions
        printf("Rapport enregistre (%d)
\n",nb_miss);//affichage
        return nb_miss;//retourne le nombre de missions
    }
}

void recapitulatif (Inscription ins[],Mission miss[], char b[],
int nb_ins, int nb_miss) {
    int ope = atoi(b);
    if (ope < 1 || ope > nb_ins)//regarde si l'entreprise donnée
est incorrecte
        printf("Entreprise incorrecte \n");
    int identifiant;
    if (strcmp(ins[ope - 1].ROLE, "OP") == 0)
        identifiant = 1;
    else if (strcmp(ins[ope - 1].ROLE, "AG") == 0)
        identifiant = 2;
    else
        identifiant = 3;
    int a0 = 0;
    int a1 = 0;
    int a2 = 0;
    int a3 = 0;
    int a4 = 0;//initialise des booléen qui servent à savoir si une
entreprise ou utilisateur est renseigné dans une étape afin de
rajouté un message au début
    if (identifiant == 1 || identifiant == 2) {
        for (unsigned int i = 0; i < nb_miss; ++i) {
            if (miss[i].Attribue == 0 && miss[i].OPE == ope)//on
cherche les missions non attribuees de l'utilisateur
            {
                if (a0 == 0) {
                    printf("* non attribuees \n");
                    a0 = 1;
                }
                printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,

```

```

        miss[i].NOMBRE_SOUS);
    }

}

for (unsigned int i = 0; i < nb_miss; ++i) {
    if (miss[i].Attribue == 1 && miss[i].OPE == ope) //on
cherche les missions attribuees de l'utilisateur
    {
        if (a1 == 0) {
            printf("* attribuees \n");
            a1 = 1;
        }
        printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS);
    }

}

for (unsigned int i = 0; i < nb_miss; ++i) {
    if (miss[i].Attribue == -1 && miss[i].OPE == ope) //on
cherche les missions terminees de l'utilisateur
    {
        if (a2 == 0) {
            printf("* terminees \n");
            a2 = 1;
        }
        printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS);
    }

}

}

if (identifiant == 2 || identifiant == 3)
{
    for (unsigned int i = 0; i < nb_miss; ++i) {
        if (miss[i].ACC == ope) //on regarde les missions
accepte mais pas termine de l'utilisateur
        {
            if (a3 == 0) {
                printf("* a realiser \n");
                a3 = 1;
            }
            printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS);
        }
    }
}

```

```

    }
    for (unsigned int i = 0; i < nb_miss; ++i) {
        if (miss[i].ACC == ope * (-1)) ///on regarde les
missions réalisé de l'utilisateur
        {
            if (a4 == 0) {
                printf("* realisees \n");
                a4 = 1;
            }
            printf("  %d %s %s %.2f (%d) \n", i + 1,
miss[i].NOM_MISSION, miss[i].NOM, miss[i].PRIX,
miss[i].NOMBRE_SOUS);
        }
    }
}
}

int main() {
    Inscription ins[ENTREPRISE]; ///on créer un tableau d'inscription
    Mission miss[MISSION_TAILLE]; ///on créer un tableau de mission
    Rapport rap[NB_RAPPORT]; ///on créer un tableau de rapport
    constru(rap); ///on construit le tableau rapport
    int nb_ins = 0; ///on initialise le nombre d'inscrit à 0
    int nb_miss = 0; ///on initialise le nombre de mission à 0
    char a[TAILLE];
    char b[TAILLE];
    char c[TAILLE];
    char d[TAILLE]; ///on initialise 4 variable char que l'on
utilisera plus tard
    while (strcmp(a, "exit") != 0) { ///boucle qui continu tant que
l'utilisateur n'a pas écrit "exit"
        scanf("%s", a); ///on récupère le premier mot (en fonction des
espaces)
        if (strcmp(a, "inscription") == 0) ///si l'utilisateur a tapé
"inscription"
        {
            scanf("%s", b); ///on récupère le deuxième mot (en fonction
des espaces)
            scanf("%s", c); ///on récupère le troisième mot (en fonction
des espaces)
            nb_ins = inscription(ins, nb_ins, b, c); ///on appelle la
fonction inscription en lui donnant le tableau d'inscrit, le
nombre d'inscrit et les valeurs b et c
        }
        else if (strcmp(a, "mission") == 0) ///si l'utilisateur a tapé
"mission"

```

```

{
    scanf("%s", b);
    scanf("%s", c);
    scanf("%s", d);
    nb_miss = mission(ins, miss, nb_ins, nb_miss, b, c, d); //on
appelle la fonction mission en lui donnant le tableau d'inscrit,
le tableau de mission, le nombre d'inscrits, le nombre de
missions, b, c et d
}
else if (strcmp (a, "consultation") == 0) //si l'utilisateur a
tapé "consultation"
{
    consultation(nb_miss, miss); //on appelle la fonction
consultation en lui donnant le nombre de mission et le tableau de
mission
}
else if (strcmp (a, "detail") == 0) //si l'utilisateur a tapé
"detail"
{
    scanf("%s", b);
    detail(miss, nb_miss, b); //on appelle la fonction detail en
lui donnant le tableau de mission, le nombre de mission et b
}
else if (strcmp (a, "acceptation") == 0) //si l'utilisateur a
tapé "acceptation"
{
    scanf("%s", b);
    scanf("%s", c);
    acceptation(ins, miss, b, c, nb_ins, nb_miss); //on appelle
la fonction acceptation en lui donnant le tableau d'inscrit, le
tableau de mission, le nombre d'inscrits, le nombre de missions, b
et c
}
else if (strcmp (a, "sous-traitance") == 0) //si l'utilisateur a
tapé "sous-traitance"
{
    scanf("%s", b);
    scanf("%s", c);
    scanf("%s", d);
    nb_miss = sous_traitance(ins, miss, b, c, d, nb_miss,
nb_ins); //on appelle la fonction sous-traitance en lui donnant le
tableau d'inscrit, le tableau de mission, le nombre d'inscrits, le
nombre de missions, b, c et d
}
else if (strcmp (a, "rapport") == 0) //si l'utilisateur a tapé
"rapport"
{
    scanf("%s", b);

```

```

        scanf("%s",c);
        nb_miss = rapport(miss, ins, rap, b, c,
nb_miss,nb_ins);//on appelle la fonction rapport en lui donnant le
tableau de mission, le tableau d'inscrit, le tableau de rapport,
le nombre de mission, le nombre d'inscrit, b et c
    }
    else if (strcmp (a,"recapitulatif") == 0)//si l'utilisateur a
tapé "recapitulatif"
    {
        scanf("%s",b);
        recapitulatif(ins,miss, b, nb_ins, nb_miss);//on appelle la
fonction recapitulatif en lui donnant le tableau de mission, le
nombre d'inscrits, le nombre de missions et b
    }
}

```