Projet IAP 2020/2021

Liste des instructions

Lexèmes

Définition. *< entier > : Un nombre positif, composé d'une suite de chiffres parmi 0,1... 9, sans espace, ni virgule, ni aucun autre symbole. Sa valeur maximale est bornée à 10000 (inclus).*

Définition. *< mot > : Un mot, composé d'une suite de lettres parmi A,B..Z,a,b..z sans espace, ni virgule, ni aucun autre symbole (donc pas d'accentuation non plus). Sa longueur maximale est bornée à 35 caractères (inclus).*

Définition. mot clé *: Un mot réservé pour une instruction. Il ne peut être utilisé nulle part ailleurs. Noté en gras dans la suite.*

Spécialités

Instruction. ***developpe*** *< mot > < entier >*

Déclare une nouvelle spécialité *de tâches qui sera proposée par l'entreprise. Le mot correspond au nom de la spécialité ; l'entier indique le prix de facturation d'une heure de travail pour cette spécialité.*

Le système traite au maximum 10 spécialités.

Travailleurs

Instruction. ***embauche*** *< mot > < mot >*

Déclare un nouveau travailleur. Le premier mot correspond à son nom ; le second à sa spécialité. La spécialité doit avoir été précédement déclarée. Chaque travailleur doit avoir un nom distinct.

Peut aussi être utilisée pour un travailleur déjà déclaré. Dans ce cas, une nouvelle spécialité est ajoutée pour ce travailleur. La spécialité doit avoir été précédement déclarée.

Le système traite au maximum 50 travailleurs.

Clients

Instruction. ***demarche*** *< mot >*

Déclare un client. Chaque client doit avoir un nom distinct.

Le système traite au maximum 100 clients.

Commandes

Instruction. ***commande*** *< mot > < mot >*

Déclare une nouvelle commande. Le premier mot définit la commande. Chaque commande doit avoir un nom distinct. Le second mot correspond au nom du client effectuant la commande.

Le système traite au maximum 500 commandes.

Tâches

Instruction. ***tache*** *< mot > < mot > < entier >*

Déclare une tâche nécessaire à la réalisation d'une commande. Le premier mot désigne la commande ; le second, la spécialité concernée. L'entier indique le nombre d'heures nécessaires pour réaliser la tâche. Chaque commande peut avoir au maximum une tâche par spécialité.

Remarque : les tâches ne sont pas interdépendantes. Elles peuvent être realisées en parallèle et avancer chacune à son rythme.

Affectation

L'affectation d'une tâche à un travailleur est faite automatiquement par le système. La règle dépend du sprint : soit le premier travailleur compétent connu, soit celui ayant reçu le moins de tâches, soit celui ayant la plus petite charge de travail restante.

Execution

Instruction. ***progression*** *< mot > < mot > < entier >*

Déclare l'avancement d'une tâche. Le premier mot désigne la commande ; le second la spécialité ; l'entier indique le nombre d'heures à ajouter à son avancement.

Instruction.
progression *< mot > < mot > < entier >* ***passe***

Lorsque le mot clé ***passe*** *est ajouté, il indique qu'il faut, en plus du comportement ci-dessus, réaffecter la tâche.*

Remarque : il se peut que le même travailleur soit à nouveau sélectionné, s'il correspond encore au critère de choix lors de l'exécution de cette instruction.

Etat

Instruction. ***specialites***

Affiche la liste des spécialités, dans l'ordre de leur déclaration, ainsi que le coût associé.

Instruction. ***travailleurs*** *< mot >*

Affiche la liste des travailleurs compétents pour la spécialité choisie, par ordre de déclaration des travailleurs.

Instruction. ***travailleurs tous***

Affiche la liste des travailleurs pour toutes les spécialités. Les spécialités sont affichées dans l'ordre de leur déclaration ; pour une même spécialité, les travailleurs sont affichés par ordre de déclaration.

Instruction. ***client*** *< mot >*

Affiche la liste des commandes effectuées par le client choisi, selon leur ordre de déclaration.

Instruction. ***client tous***

Affiche la liste de toutes les commandes, par ordre de déclaration de client, et de commande.

Instruction. ***supervision***

Affiche la description de l'état d'avancement des commandes et des tâches associées, par ordre de commandes, et de spécialité.

Instruction. ***charge*** *< mot >*

Pour le travailleur choisi, affiche le nombre d'heures restantes pour l'accomplissement de chacune des tâches qui lui sont affectées. Les tâches sont affichées par ordre de commande, et de spécialité.

Facturation

Lorsqu'une commande est terminée, le système affiche automatiquement une facturation par spécialité effectivement utilisée.
Lorsque toutes les commandes sont terminées, le système affiche automatiquement une facturation globale.

Fin

Instruction. ***interruption***

Quitte brutalement le programme... L'usine est délocalisée.

Liste des sprints

Sprint 1 : La boucle des instructions

Chaque instruction provoque un affichage selon les formats ci-dessous. Chaque affichage est fait sur une ligne et se termine par un passage à la ligne, sauf pour le mot clé **passe** qui est fait sur deux lignes distinctes. L'instruction **interruption** provoque ensuite l'arrêt du programme ; les autres instructions n'ont pas d'effet.

developpe *mot entier*	*## nouvelle specialite "*mot*" ; cout horaire "*entier*"*
embauche *mot1 mot2*	*## nouveau travailleur "*mot1*" competent pour la specialite "*mot2*"*
demarche *mot*	*## nouveau client "*mot*"*
commande *mot1 mot2*	*## nouvelle commande "*mot1*", par client "*mot2*"*
tache *mot1 mot2 entier*	*## la commande "*mot1*" requiere la specialite "*mot2*" (nombre d'heures "*entier*")*
progression *mot1 mot2 entier*	*## pour la commande "*mot1*", pour la specialite "*mot2*" : "*entier*" heures de plus ont ete realisees*
progression *mot1 mot1 entier* **passe**	*## pour la commande "*mot1*", pour la specialite "*mot2*" : "*entier*" heures de plus ont ete realisees* *## une reallocation est requise*
specialites	*## consultation des specialites*
travailleurs *mot*	*## consultation des travailleurs competents pour la specialite "*mot*"*
travailleurs tous	*## consultation des travailleurs competents pour chaque specialite*
client *mot*	*## consultation des commandes effectuees par "*mot*"*
client tous	*## consultation des commandes effectuees par chaque client*
supervision	*## consultation de l'avancement des commandes*
charge *mot*	*## consultation de la charge de travail de "*mot*"*
interruption	*## fin de programme*

Versions : **alpha** les mots clé **passe** et **tous** ne sont pas traités (i.e. absents du jeu de test). **beta** Le mot clé **passe** n'est pas traité. **release** Tous les mots clés sont traités.

Guide technique

```
typedef enum {FAUX=0,VRAI=1} Booleen;
Booleen EchoActif = FAUX;
// Messages emis par les instructions ----------------------------------------
#define MSG_DEVELOPPE     "## nouvelle specialite \"%s\" ; cout horaire \"%d\"\n"
#define MSG_INTERRUPTION "## fin de programme\n"
// Lexemes -------------------------------------------------------------------
#define LGMOT           35
#define NBCHIFFREMAX   5
typedef char Mot [LGMOT+1];
void get_id(Mot id){
  scanf("%s", id);
  if (EchoActif) printf(">>echo %s\n", id);                                  }
int get_int(){
  char buffer [NBCHIFFREMAX+1]; scanf("%s", buffer);
  if (EchoActif) printf(">>echo %s\n", buffer);
  return atoi(buffer);                                                       }
// Instructions --------------------------------------------------------------
// developpe -----------------------------
void traite_developpe(){
  Mot nom_specialite; get_id(nom_specialite);     int cout_horaire = get_int();
  printf(MSG_DEVELOPPE, nom_specialite, cout_horaire);                       }
// interruption --------------------------
void traite_interruption(){ printf(MSG_INTERRUPTION);                        }
// Boucle principale ---------------------------------------------------------
int main(int argc, char *argv[]) {
  if(argc>=2 && strcmp("echo",argv[1])==0 ) EchoActif = VRAI;
  Mot buffer;
  while(VRAI){
    get_id(buffer);
    if(strcmp(buffer,"developpe")==0)     { traite_developpe(); continue; }
    if(strcmp(buffer,"interruption")==0) { traite_interruption(); break; }
    printf("!!! instruction inconnue >%s< !!!\n", buffer);
  }
  return 0;
}
```

Sprint 2 : L'entreprise et ses clients

Les instructions non modifiées ici produisent le même effet qu'au sprint précédent.

developpe *mot entier* **embauche** *mot mot* **demarche** *mot*	Ces instructions ne provoquent plus d'affichage. Les informations sont mémorisées.
specialites	*specialites traitees :* liste Où *liste* est une suite de couples *mots*/*entier* correspondants aux noms des spécialités et leur coût horaire, par ordre de déclaration des spécialités.
travailleurs *mot*	*la specialite* mot *peut etre prise en charge par :* liste Où *mot* est une spécialité, et *liste* une suite de mots, correspondants aux noms des travailleurs compétents.
travailleurs tous	*la specialite* mot *peut etre prise en charge par :* liste ⋮ *la specialite* mot *peut etre prise en charge par :* liste
client *mot*	*le client* mot *a commande :* liste Où *mot* est un client, et *liste* une suite de mots, correspondants aux identifiants des commandes effectuées. Pour ce sprint, la *liste* est donc vide (chaine vide).
client tous	*le client* mot *a commande :* liste ⋮ *le client* mot *a commande :* liste

Versions : **alpha** Chaque travailleur n'a qu'une seule compétence. **release** Les travailleurs peuvent avoir plusieurs compétences.

Guide technique

```
// Donnees ----------------------------------------------------

// specialites ----------------------
#define MAX_SPECIALITES    10

typedef struct{
        Mot nom;
        int cout_horaire;
} Specialite;

typedef struct{
        Specialite tab_specialites [MAX_SPECIALITES];
        unsigned int nb_specialites;
} Specialites;

// travailleurs ---------------------
#define MAX_TRAVAILLEURS  50

typedef struct{
        Mot nom;
        Booleen tags_competences [MAX_SPECIALITES];
} Travailleur;

typedef struct{
        Travailleur tab_travailleurs [MAX_TRAVAILLEURS];
        unsigned int nb_travailleurs;
} Travailleurs;

// client ---------------------------
#define MAX_CLIENTS        10

typedef struct{
        Mot tab_clients [MAX_CLIENTS];
        unsigned int nb_clients;
} Clients;
```

Sprint 3 : Les commandes

Les instructions non modifiées ici produisent le même effet qu'au sprint précédent.

commande *mot mot* **tache** *mot mot entier* **progression** *mot mot entier* **progression** *mot mot entier* **passe**	Ces instructions ne provoquent plus d'affichage. Les informations sont mémorisées. le mot clé **passe** n'a aucun effet.
client *mot* **client tous**	Les listes de commandes sont à présent renseignées.
supervision	*etat des taches pour* mot *:* liste : : *etat des taches pour* mot *:* liste Où *mot* est le nom d'une commande, et *liste* correspond à une suite de triplets *mot*:*entier*/*entier*, séparés par des virgules, où chaque *mot* est le nom d'une spécialité pour laquelle une tâche est requise, le premier *entier* est le total d'heures effectuées, et le second *entier* est le nombre d'heures requises. Les commandes et les spécialités sont affichées par ordre de déclaration.

Versions : **release** Tous les mots clés sont traités (sauf **passe**).

Guide technique

```
// Donnees ----------------------------------------------------------

//...

// commandes ------------------------
#define MAX_COMMANDES 500

typedef struct{
   unsigned int nb_heures_requises;
   unsigned int nb_heures_effectuees;
} Tache;
typedef struct{
        Mot nom;
        unsigned int idx_client;
        Tache taches_par_specialite [MAX_SPECIALITES]; // nb_heures_requises==0 <=> pas de tache pour cette specialite
} Commande;

typedef struct{
        Commande tab_commandes [MAX_COMMANDES];
        unsigned int nb_commandes;
} Commandes;
```

Sprint 4 : L'affectation automatique

Les instructions non modifiées ici produisent le même effet qu'au sprint précédent.

tache	L'affectation automatique de la tâche à un travailleur est implémentée
charge	*charge de travail pour* mot *:* liste ⋮ *charge de travail pour* mot *:* liste Où *mot* est le nom d'un travailleur, et *liste* correspond à une suite de triplets *mot1*/*mot2*/*entier*heure(s), séparés par des virgules, où *mot1* est le nom d'une commande, *mot2* est le nom d'une spécialité, et *entier* le nombre d'heures restantes pour compléter cette tâche. L'affichage est fait par ordre de déclaration de travailleurs d'abord, puis de commandes, puis de spécialités. Seules les tâches définies et non achevées sont affichées.

Versions : **alpha** La tâche est affectée au premier travailleur embauché ayant la compétence. **beta** La tâche est affectée au premier travailleur embauché compétent ayant reçu le moins de tâches jusqu'à présent. **release** La tâche est affectée au premier travailleur compétent embauché ayant le moins d'heures à faire pour compléter toutes ses tâches.
Remarques : Le mot clé **passe** n'est toujours pas traité. **Lisez les sprints suivants.**

Sprint 5 : La facturation

La **détection** de commande complétée est réalisée. Elle correspond au cas où toutes les tâches requises pour une commande ont été accomplies. Le programme affiche alors:

facturation mot *:* liste où *mot* est le nom de la commande, et *liste* est une suite de couples *mot*:*gros_ entier* séparés par des virgules, indiquant pour chaque spécialité, le coût associé. Les spécialités sont mentionnées par ordre de déclaration ; uniquement si une tâche leur était effectivement associée.

Remarque : *gros_ entier* est un entier autorisé à dépasser la limite spécifiée pour *entier*.

La **détection** de fin est réalisée. Elle correspond au cas où toutes les commandes sont terminées. Le programme affiche alors:

facturations : liste où *liste* est une suite de couples *mot*:*gros_ entier* séparés par des virgules, indiquant le nom du client et le coût total de ses commandes, par ordre de clients.

Remarque : Le programme s'arrête après cet affichage, ignorant donc toute eventuelle autre entrée.
Versions : les trois versions sont admissibles, selon la meilleure version du sprint 4 obtenue.

Sprint 6 : Le mot clé *passe*

La mot clé **passe** est implémenté.
Versions : les versions **beta** et **release** sont admissibles, selon la meilleure version du sprint 4 obtenue.