Travaux Pratiques Info 1

Informatique Industrielle GEII Nîmes - 1 septembre 2016

Dev-Cpp est une interface de développement intégré. Elle comprend :

- Un éditeur de texte permettant la saisie du code source écrit en langage C
- Un compilateur permettant de traduire le code source en langage objet (fichiers .o)
- Un éditeur de lien permettant de rassembler tous les fichiers objets et les appels aux fonctions de diverses bibliothèques pour générer le fichier exécutable (fichier binaire .exe)
- Un outil de déverminage ou déboggage permettant d'analyser et de suivre le fonctionnement d'un programme en vue de sa correction ou son amélioration.

Premier(s) programme(s) et prise en main de l'outil de développement Dev-Cpp

Voici le code source d'un petit programme écrit en langage C qui affiche à l'écran (dans une console) les mots « c'est chouette le C ! » :

```
#include<stdio.h>
// inclusion des entêtes de fonctions présentes dans les bibliothèques
void main(void)
{
        printf("c'est chouette le C!")
}
```

Voici la procédure pour générer un fichier binaire à partir d'un code source sous Dev-Cpp.

1. Créer un projet de type « langage C » dans la suite Dev-Cpp. Au préalable, créer un dossier à votre nom sur le bureau et ranger les fichiers et exécutables dans ce répertoire.
2. Saisir le code ci-dessus dans l'éditeur de texte intégré à Dev-Cpp et enregistrer le fichier avec une extension « .c ». On vient donc d'éditer le code source de notre programme.
3. Lancer la compilation du code source au sein de Dev-Cpp en cliquant sur l'icône « compiler ». Dans la fenêtre « messages » , le compilateur indique son état, les erreurs

ou avertissements (« warning ») qu'il a obtenus en analysant et compilant le fichier source.

4. Si l'étape précédente n'est pas validée c'est-à-dire obtenir une compilation réussie, on retourne à l'étape n°2, on effectue les corrections nécessaires et on re-compile (étape n° 3)

5. le fichier exécutable est généré dans votre répertoire où vous avez défini votre projet (dans votre répertoire à votre nom) et porte l'extension « .exe ». Exécuter le en double-cliquant dessus.

Travail demandé :

Après avoir saisi le code ci-dessus sous l'éditeur de Dev-Cpp, compiler et exécuter le fichier binaire :

- Que vous indique le compilateur?
- Comment modifier le code source pour que la compilation s'effectue sans encombre?
- Exécuter le fichier binaire. Que se passe-t-il?
- Modifier le code source en rajoutant après le printf(« c'est chouette le C »); l'instruction suivante :

```
system("PAUSE");
```

Compiler et exécuter le fichier binaire obtenu.

A quoi sert cette instruction ?

Premiers programmes

Calcul de la TVA

On souhaite automatiser le calcul de la TVA[1] pour une quantité de produit acheté dont on ne connaît que le prix à l'unité hors taxe.

L'algorithme est le suivant :

- demander le prix à l'unité du produit
- demander la quantité de produit
- calculer le prix total de quantité de produit acheté augmenté de la TVA soit le prix TTC vaut la quantité de produit * par le prix à l'unité du produit * par l'augmentation du taux de la TVA (20 %)
- afficher le résultat

Voici le code en langage C proposé :

```
void main(void)
        {
                float prixunit, prixtotal;
                unsigned int quantite,TVA=20;
                        // demander le prix à l'unité
                printf("Donner le prix unitaire de l'article : ");
                scanf("%d",&prixunit);
                        // demander le nombre d'articles
                printf("Donner la quantité d'articles achetée : ");
                scanf("%d",&quantite);
                        //calcul du prix total avec TVA
                prixtotal=(prixunit*quantite)*(1+TVA/100);
                        // affichage du résultat « prixtotal »
                printf("Le prix TTC pour une quantité %c d'articles est de %d euros.",prixtotal);
        }
```

Travail demandé :

Après avoir saisi le code ci-dessus, compiler le fichier source et exécuter le fichier binaire obtenu.

- Que vous indique le compilateur?
- Exécuter le fichier binaire. Que se passe-t-il?
- Modifier le code source, compiler et exécuter le programme pour répondre au problème posé.

[1] En France la TVA est de 20 %

Echange

Dans la recette d'une préparation d'un processus industriel, on souhaite échanger le contenu de 2 variables sans utiliser une autre variable intermédiaire car l'automate ne possède plus de ressource en mémoire de données.
Voici le programme proposé pour effectuer cet échange avec une telle contrainte :

```
void main(void)
{
        int mesure1=10, mesure2=45;
        printf("mesure1 vaut %d, mesure2 vaut %d ",mesure1, mesure2);
        mesure1 = mesure1+mesure2;
        mesure2 = mesure2-mesure1;
        mesure1=mesure1-mesure2;
        printf("maintenant mesure1 vaut %d et mesure2 vaut %d",mesure1,mesure2);
}
```

Travail demandé :

Après avoir saisi le code ci-dessus, compiler le fichier source et exécuter le fichier binaire obtenu.

- Que vous indique le compilateur?
- Exécuter le fichier binaire. Que se passe-t-il?
- Modifier le code source, compiler et exécuter le programme pour répondre au problème posé.

Géométrie

Proposer l'**algorithme**, puis le **code** qui permet à partir du rayon d'un cercle en mètre de fournir la circonférence (en mm), la surface du disque (en cm^2) et le volume du cylindre (en dm^3 et en l) qui contient ce cercle.

Calcul de moyenne d'un étudiant

Un étudiant en DUT GEII a obtenu les notes reportées dans le tableau suivant au cours du premier semestre.

Unités d'enseignement	**UE1**			**UE2**			**UE3**			
Modules	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M4**
Notes	8	10,5	15	12	10	13	9	12	16	10
Coefficients	4	3	3	3	4	3	3	3	2	5

Travail demandé :

Aidez-le à automatiser le calcul de sa moyenne pondérée pour ce semestre :

- en proposant un **algorithme** adéquat en utilisant 10 variables pour les notes et 10 pour les coefficients. On pourra demander à l'utilisateur de rentrer les 10 notes (les coefficients étant fixes).
- en proposant le **code** source associé
- en générant le fichier binaire qu'il devra utiliser pour calculer sa moyenne générale.

Soirée arrosée ?

Avant de reprendre la route, une personne qui sort d'une soirée (arrosée?) veut tester son taux d'alcoolémie.

Travaux demandés :

a) Proposez un **algorithme** puis le **code** écrit en langage C qui permet de calculer le taux d'alcoolémie en fonction :

- de la quantité de produit alcoolisé bue en ml,
- de la quantité d'alcool présent dans le produit alcoolisé en pourcentage
- du type d'individu : femme ou homme.

Le taux d'alcoolémie (T) est le rapport entre la masse d'alcool pur[2] ingérée (Ma) en gramme sur la masse de l'individu (Mi) en kilogramme selon son type (K, pour une femme ce coefficient vaut 0.6 et pour un homme, il vaut 0.7), soit :

T = Ma / (Mi en kg*K)

b) Sachant que la vitesse d'élimination de l'alcool dans un organisme est de 0.15 gramme par litre de sang d'un individu[3] par heure. Combien de temps mettra un individu à éliminer totalement la boisson alcoolisée ingérée?

Proposer l'**algorithme** et le **code** en langage C permettant de réaliser ce calcul de manière automatique suite au calcul du taux d'alcoolémie précédent.

[2] la masse volumique de l'alcool est de 800 grammes pour 1 litre

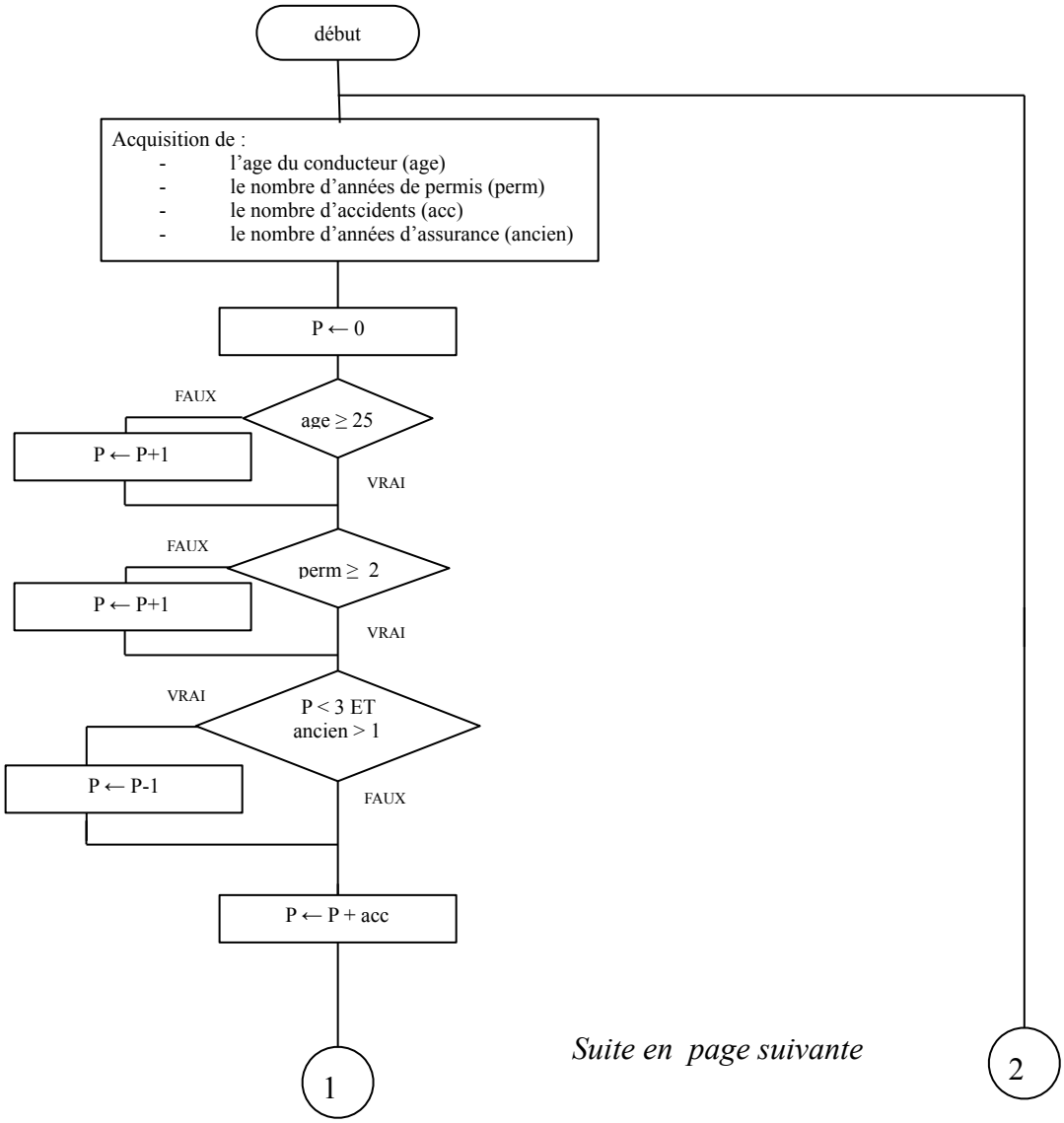
[3] Un individu normalement constitué possède 5 litres de sang

Structures conditionnelle et itérative

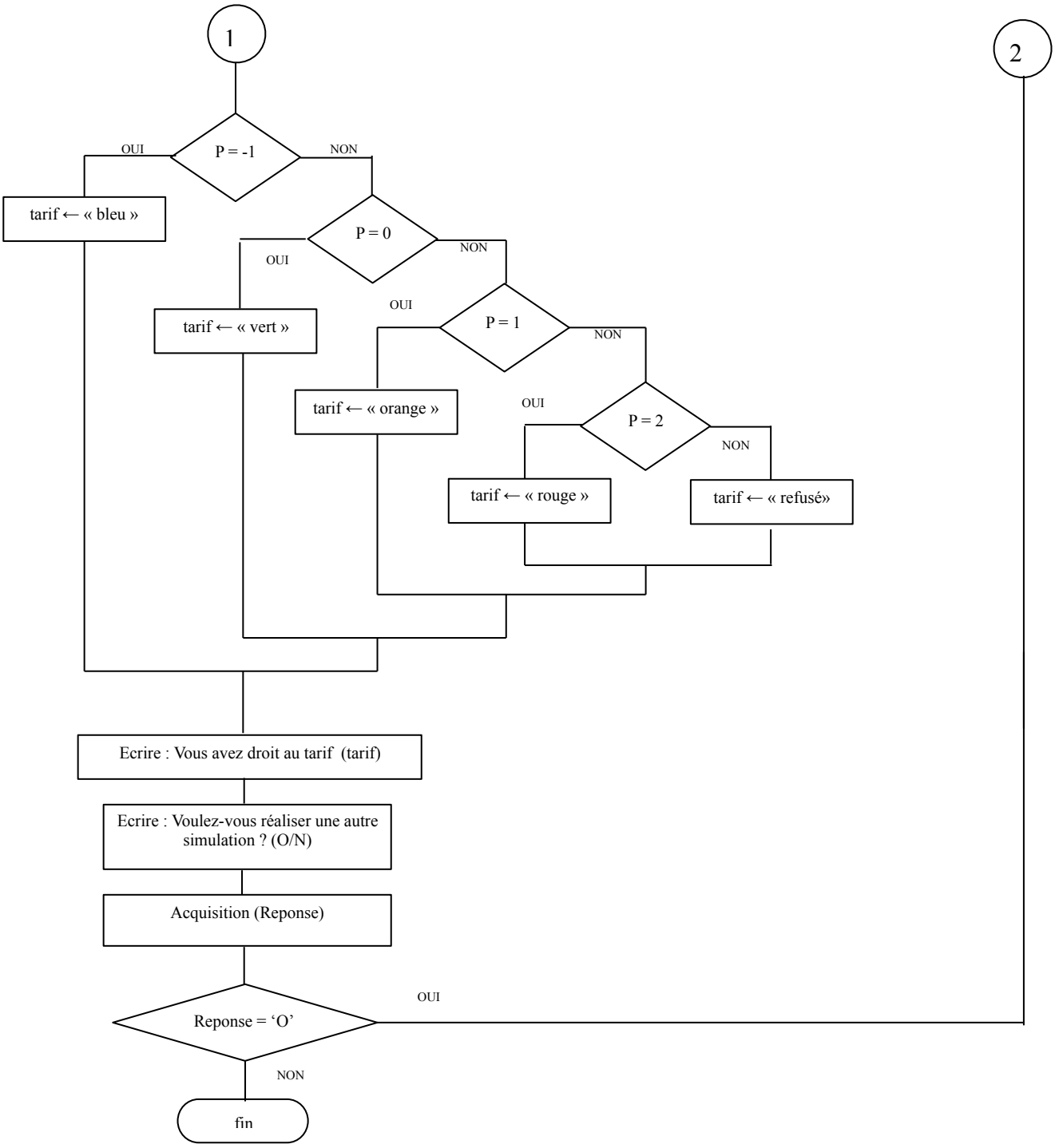
Tarifs d'assurance auto.

Une compagnie d'assurance automobile propose à ses clients quatre familles de tarifs identifiables par une couleur, du moins au plus onéreux : tarifs bleu, vert, orange et rouge.
Le tarif appliqué dépend de la situation du conducteur (**âge**, nombre d'années de **perm**is, nombre d'**acc**idents responsables, **ancien**neté, …)
Le calcul du tarif est réalisé au moyen de l'organigramme donné ci-dessous.

1. A partir de cet organigramme, déterminer les variables d'entrée et de sortie
2. On demande de coder cet organigramme en pseudo-code.
3. On demande de réaliser puis tester le code en langage C. Attention à l'indentation du code



Suite en page suivante


1
2
P = -1
OUI
NON
tarif ← « bleu »
P = 0
OUI
NON
tarif ← « vert »
P = 1
OUI
NON
tarif ← « orange »
P = 2
OUI
NON
tarif ← « rouge »
tarif ← « refusé»
Ecrire : Vous avez droit au tarif (tarif)
Ecrire : Voulez-vous réaliser une autre simulation ? (O/N)
Acquisition (Reponse)
Reponse = 'O'
OUI
NON
fin

Panneau d'affichage à Leds

Pour afficher le résultat de l'algorithme précédent, on dispose d'un afficheur matriciel à LEDs de 16 par 16 leds.

L'affichage du tarif se fera avec l'affichage des lettres 'o', 'b', 'v', 'r' et lorsque l'on est « refusé » par la lettre 'X'.

On symbolise par '1', l'état d'une led allumée et par '0' l'état d'une led éteinte. On ne s'intéressera qu'à la visualisation de la lettre 'X'. Voici l'exemple sur la lettre 'X' pour l'état « refusé » sur cet afficheur.

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Voici le pseudo code pour réaliser un tel affichage.

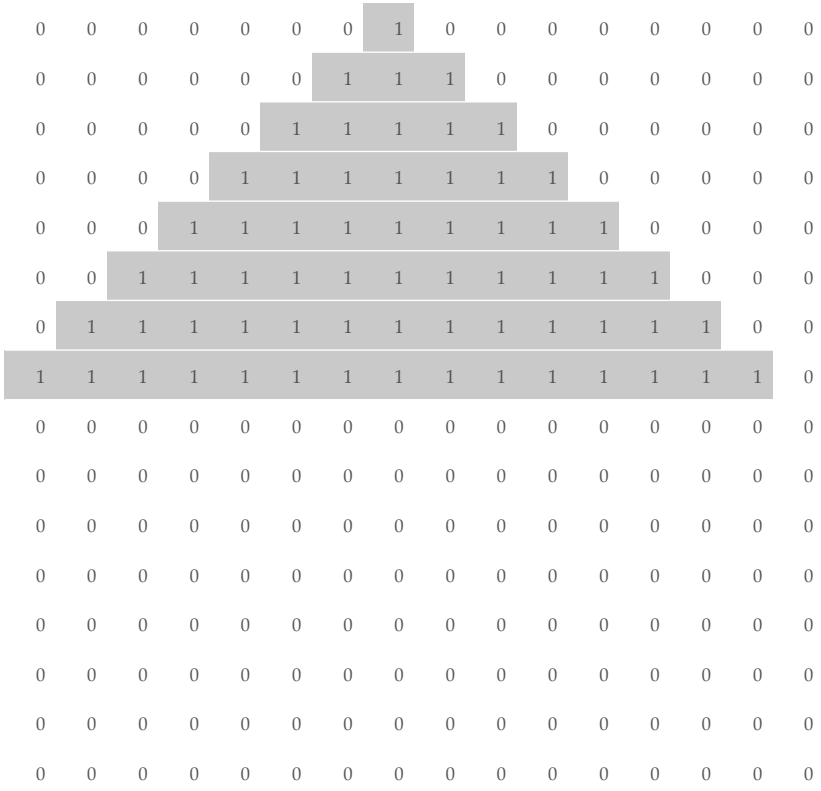
```
Début
        Pour j allant de 0 jusqu'à 15 faire
                Pour i allant de 0 jusqu'à 15 faire
                        Si (i=j) ou (i=15-j)
                                Ecrire "1"
                        Sinon
                                Ecrire "0"
                        Finsi
                Finpour
                Ecrire "\n"
        Finpour
Fin
```

Ecrire et tester le programme en C qui réalise cet affichage.

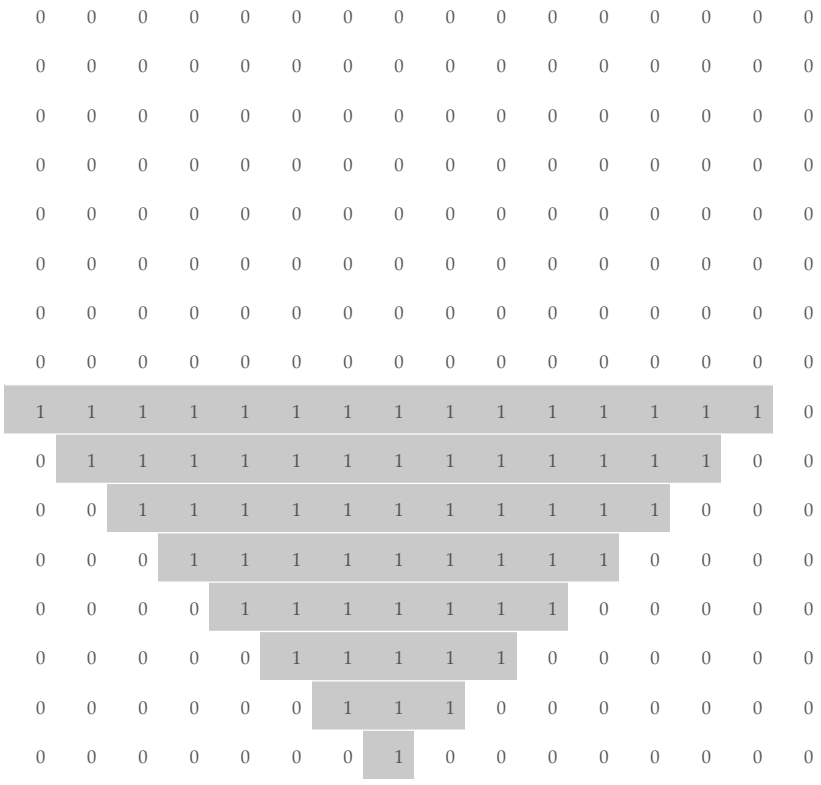
Evolution du tarif

Chaque année les tarifs évoluent. Aussi, on souhaite afficher les symboles suivants.
A partir de l'algorithme précédent, proposer le programme en C pour ces affichages. Tester.

- « flèche vers le haut » pour une hausse du tarif de la catégorie



- « flèche vers le bas » pour une baisse du tarif de la catégorie



Tableaux

Tri par permutation

A partir d'un tableau de moyennes d'étudiants (26 étudiants) rangées de manière aléatoire, on souhaite ranger ces moyennes par ordre décroissant.

- dans un premier temps, on va générer un tableau rempli aléatoirement de nombres réels compris entre 0,0 et 20,0. Pour cela on pourra utiliser la fonction « rand() » qui génère un nombre aléatoire compris entre 0 et 32767.

Exemple de programme utilisant la fonctions RAND

```
#include <stdio.h>
#include <time.h>
int main()
{
        int x = 0;
        srand(time(NULL)); // initialisation du générateur de nombre aléatoire
        for (x = 0; x < 10; ++x)
                printf ("Nombre %d : %d\n", x + 1, rand()%100); // affichage d'un
                                                        // nombre aléatoire compris entre 0 et 99
        return 0;
}
```

- puis on proposera un code qui permet d'échanger le contenu de deux variables.
- Enfin on proposera le code qui permet de classer le tableau de moyennes par ordre de mérite.

Chaînes de caractères

A la chaîne !

On remarque que le code qui recherche la taille d'une chaine de caractères est déjà disponible dans une bibliothèque dénommée « string.h ». Ce code (fonction) est lui même nommé « strlen ». Vous trouverez en annexe une liste de fonctions disponibles pour traiter les chaînes de caractères présente dans la bibliothèque.
Par exemple en utilisant la fonction « strlen » qui retourne la longueur d'une chaine de caractère :

```
#include<stdio.h>
#include<string.h>

void main(void)
{
        char chaine[10]=“bonjour“;
        printf(“%d“,strlen(chaine)); // affiche la valeur 7
}
```

A partir de ces fonctions, on souhaite réaliser un programme qui conjugue des verbes du premier groupe.

- L'utilisateur saisit un verbe à l'infinitif.
- On extrait la base du verbe (on supprime le suffixe « er »).
- On rajoute les suffixes adéquats pour chaque personne.

On donnera un affichage du type (exemple pour le verbe « conjuguer ») :

je conjugu-e
tu conjugu-es
il/elle conjugu-e
nous conjugu-ons
vous conjugu-ez
ils/elles conjugu-ent

On utilisera trois tableaux qui contiennent :

- les personnes,
- la base du verbe
- et les suffixes.

Message codé

Le code de Vigenère est une technique de codage utilisant le décalage de caractères,. Tous les caractères ne sont pas décalés de la même valeur et dépendent d'une clé, en générale un mot ou une phrase.

Par exemple, si la clé est "BONJOUR", les lettres du message seront décalés de :

- 2 lettres ('B' est la deuxième lettre de l'alphabet)
- 15 lettre ('O' est la quinzième lettre de l'alphabet)s
- 14 lettres ('N' est la quatorzième lettre de l'alphabet)
- 10 lettres ('J' est la dixième lettre de l'alphabet)

- 15 lettres ('O' est la quinzième lettre de l'alphabet)
- 21 lettres ('U' est la vingt-et-unième lettre de l'alphabet)
- 18 lettres ('R' est la dix-huitième lettre de l'alphabet)

Pour coder le message "vive le langage c", on répète la clé autant de fois que nécessaire et procède comme suit :

```
          VIVE  LE LANGAGE C
    +     BONJ OU BONJOUR  B
          ____________________

          XXJO AZ NPBQPBW E
```

Pour simplifier, nous allons supposer que la clé ne contient que des lettres ascii en majuscule **et que la chaine à coder ne contient que des lettres ascii en majuscules**.

a) Dans un premier temps, on va supprimer les caractère espaces des 2 chaines de caractères, ainsi on va proposer un code qui stocke dans une chaine de caractère :

VIVELELANGAGEC (14 lettres)

b) puis, on va créer une nouvelle chaine de caractère qui représente la clé répétée autant de fois pour avoir une chaine de caractère de même taille que la chaine contenant le message.

BONJOURBONJOUR (14 lettres)

c) puis, créer le code qui effectue calcul le décalage de toutes les lettre du message par la position de la lettre issue de la clé .

d) Enfin, écrire le programme qui va coder un message selon la clé fournie par un utilisateur.

e) proposer le programme pour le décodage d'une chaine de caractère crypté selon cette méthode.

f) proposer une modification qui permettra de choisir le mode de fonctionnement du programme soit en codage ou en décodage.

Annexe - Bibliothèque « string.h »

1 - *void *memchr(const void *str, int c, size_t n)*
Searches for the first occurrence of the character c (an unsigned char) in the first n bytes of the string pointed to by the argument str.

2 - *int memcmp(const void *str1, const void *str2, size_t n)*
Compares the first n bytes of str1 and str2.

3 - *void *memcpy(void *dest, const void *src, size_t n)*
Copies n characters from src to dest.

4 - *void *memmove(void *dest, const void *src, size_t n)*
Another function to copy n characters from str2 to str1.

5 - *void *memset(void *str, int c, size_t n)*
Copies the character c (an unsigned char) to the first n characters of the string pointed to by the argument str.

6 - *char *strcat(char *dest, const char *src)*
Appends the string pointed to by src to the end of the string pointed to by dest.

7 - *char *strncat(char *dest, const char *src, size_t n)*
Appends the string pointed to by src to the end of the string pointed to by dest up to n characters long.

8 - *char *strchr(const char *str, int c)*
Searches for the first occurrence of the character c (an unsigned char) in the string pointed to by the argument str.

9 - *int strcmp(const char *str1, const char *str2)*
Compares the string pointed to by str1 to the string pointed to by str2.

10 - *int strncmp(const char *str1, const char *str2, size_t n)*
Compares at most the first n bytes of str1 and str2.

11 - *int strcoll(const char *str1, const char *str2)*
Compares string str1 to str2. The result is dependent on the LC_COLLATE setting of the location.

12 - *char *strcpy(char *dest, const char *src)*
Copies the string pointed to by src to dest.

13 - *char *strncpy(char *dest, const char *src, size_t n)*
Copies up to n characters from the string pointed to by src to dest.

14 - *size_t strcspn(const char *str1, const char *str2)*
Calculates the length of the initial segment of str1 which consists entirely of characters not in str2.

15 - *char *strerror(int errnum)*
Searches an internal array for the error number errnum and returns a pointer to an error message string.

16 - *size_t strlen(const char *str)*
Computes the length of the string str up to but not including the terminating null character.

17 - *char *strpbrk(const char *str1, const char *str2)*
Finds the first character in the string str1 that matches any character specified in str2.

18 - *char *strrchr(const char *str, int c)*
Searches for the last occurrence of the character c (an unsigned char) in the string pointed to by the argument str.

19 - *size_t strspn(const char *str1, const char *str2)*

calculates the length of the initial segment of str1 which consists entirely of characters in str2.

20 - *char *strstr(const char *haystack, const char *needle)*

Finds the first occurrence of the entire string needle (not including the terminating null character) which appears in the string haystack.

21 - *char *strtok(char *str, const char *delim)*

Breaks string str into a series of tokens separated by delim.

22 - *size_t strxfrm(char *dest, const char *src, size_t n)*

Transforms the first n characters of the string src into corrent locale and place them in the string dest

Tableau et sous programmes

Exercice 1 :

On souhaite écrire un programme qui effectue le calcul de la moyenne générale d'un étudiant selon les coefficients des modules du tableau suivant :

UE1			UE2			UE3		UE4			
M1	M2	M3	M1	M2	M3	M1	M2	M1	M2	M3	M4
1	1	1	1	2	2	1	3	1	2	2	2

L'écriture du programme se fera par la mise en place de 3 sous-programmes :

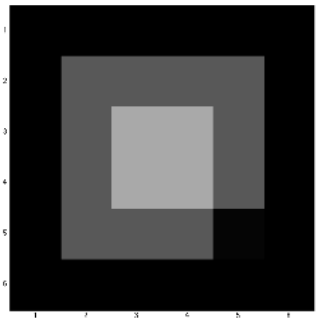
- on demandera à l'utilisateur de rentrer les notes pour les différents modules et UE ainsi que son nom et prénom.
- on effectuera le calcul de la moyenne pondérée
- on affichera les notes rentrées par UE, la moyenne ainsi que les nom et prénom de l'étudiant.

Exercice 2 :

Enoncé :

Une image numérique en « niveau de gris » est en fait un tableau à 2 dimensions : dans chacune des cases de ce tableau - que l'on nomme aussi pixels -, on code la valeur de la couleur (du blanc au noir en passant par les nuances de gris) par un nombre entier. Cette correspondance est unique et dépend de la résolution du convertisseur analogique vers numérique associé au capteur d'image. Dans notre cas, on a affaire à un convertisseur de 8 bits ce qui nous fournit 256 valeurs entières qui correspondent à 256 « couleurs » ou « niveaux de gris » que l'on peut représenter pour fabriquer une image.
Ci-dessous, un exemple d'image de taille 6 par 6 pixels à 256 niveaux de gris.

0	0	0	0	0	0
0	128	128	128	128	0
0	128	200	200	128	0
0	128	200	200	128	0
0	128	128	128	10	0
0	0	0	0	0	0

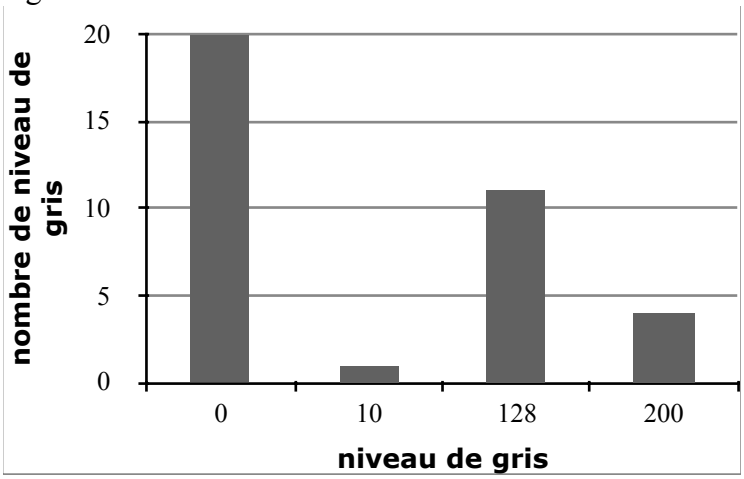


Ainsi le tableau ne contient que des valeurs comprises entre 0 et 255, la valeur 0 étant associée à la couleur « noire » et la valeur 255 à la couleur « blanche », toutes les autres valeurs représentent une couleur comprise entre le blanc et le noir.
Afin d'effectuer des traitements sur les images numériques, il est possible de travailler sur les histogrammes issus des images.

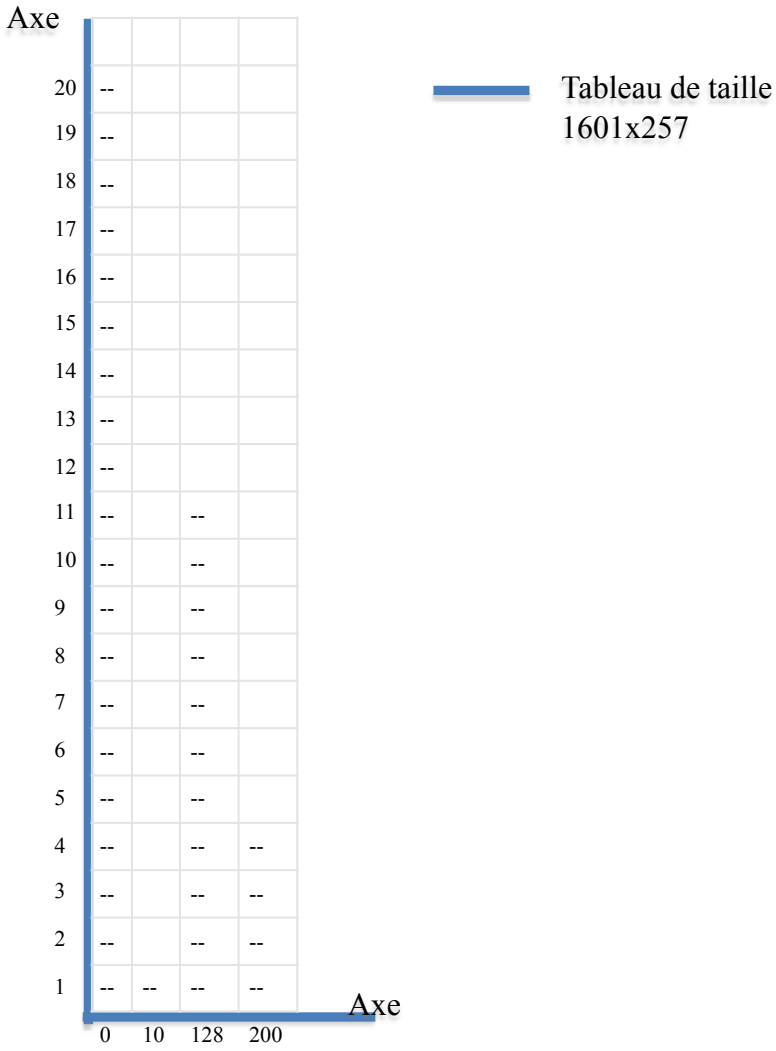
Un histogramme est une représentation sous forme de diagramme « bâtons » du nombres de pixels ayant le même niveau de gris dans l'image.
Ainsi si on reprend l'exemple ci-dessus :

- Il y a 20 pixels dont le niveau de gris est à '0'
- Il y a 1 pixels dont le niveau de gris est à '10'
- Il y a 11 pixels dont le niveau de gris est à '128'
- Il y a 4 pixels dont le niveau de gris est à '200'

On obtient ainsi l'histogramme ci-dessous :



A partir d'une image numérique de taille 40 par 40 pixels à 256 niveaux de gris, vous trouverez la définition du tableau à deux dimensions dans le fichier texte « image.txt » sur le bureau représentant cette image, il est demandé de réaliser le programme qui permet l'affichage de l'histogramme de ce fichier dans la console, selon le format de présentation ci-dessous réalisé à partir de l'exemple précédent.



Pour réaliser ceci

- Dans un premier temps, il est demandé de visualiser sous forme de symboles (« char ») le tableau à 2 dimensions « image » de type « unsigned char » qui est une représentation de l'image proposée de taille 40 par 40 pixels. Ceci afin de nous donner une idée sur le contenu de l'image convertie en tableau de pixels dont chaque valeur est un niveau de gris.
- Dans un deuxième temps, on déclarera un tableau à 1 dimension (« histo ») de taille 256 puisque l'on a 256 niveaux de gris possibles. On stockera ainsi le nombre d'apparition d'un niveau de gris dans le tableau image et on initialisera au préalable ce tableau avec des valeurs égales à « 0 ». On proposera un bout de code permettant de réaliser le remplissage de ce tableau avec le nombre d'apparition des pixels ayant un même niveau de gris.
- On affichera dans la console le contenu du tableau « histo » à des fins de vérifications.
- Afin de limiter l'affichage de la taille de l'histogramme, proposer le code qui permet de déterminer le maximum (« max ») du tableau « histo » précédemment rempli. Afficher ce maximum sur la console pour vérification.
- Toujours dans un soucis de limitation de l'affichage au strict minimum dans la console, on va se restreindre à n'afficher que les colonnes de l'histogramme où le nombre de pixels

ayant le même niveau de gris est non nul. Pour cela, proposer le code qui remplit un tableau (« indice ») de taille maximum de 256 qui va contenir que les valeurs de pixels ayant un nombre d'apparition non nul. Dans le cas de l'exemple on stockerait dans ce tableau les valeurs 0, 10, 128 et 200. Afficher ce tableau pour vérification

- Une fois que l'on possède l'histogramme dans le tableau « histo » et définit les bornes d'affichage de ce même histogramme (tableau « indice » et la valeur « max »), l'objectif maintenant est de créer un tableau à 2 dimension (« image_histo) de taille 1601 (40x40 pixels) par 257 (256 niveaux de couleurs+1). Ce tableau sera en fait une image représentant sous forme de symbole (« - » et espace) l'histogramme contenu dans le tableau « histo ». On va donc « dessiner » le diagramme en « bâtons » dans ce tableau à 2 dimensions.
    - Dans un premier temps, proposer le code qui permet de remplir la première colonne de ce tableau « image_histo » par les valeurs allant de 1 à la valeur « max » trouvée dans le tableau « histo ».
    - De même pour la première ligne proposer le programme qui permettra de remplir celle-ci avec tous les niveaux de gris pouvant être présents.
    - Ensuite à partir de la deuxième ligne et deuxième colonne, proposer le code qui va remplir le tableau de symboles « - » et de symboles espace de la manière suivante. On va positionner dans le tableau :
        - le symbole « - » pour toutes colonnes dont les pixels présentent une valeur d'apparition non nulle, et ce, sur autant de lignes que la valeur de ce nombre. Exemple si pour le pixel de valeur 128 il y a 11 apparitions de ce même pixels dans l'image, on mettra en colonne 128, 11 lignes de symbole « - »
        - partout ailleurs, on positionnera le symbole espace.

      Il est conseillé, au préalable, de remplir entièrement le tableau « image_histo » de symboles espace …
- Enfin proposer le code qui affichera ce tableau en prenant soin de n'afficher que les colonnes dont les valeurs de pixels possèdent un nombre d'apparition non nul. On n'oubliera pas non plus les axes (ordonnées et abscisses).

NB : l'instruction system("mode con cols=260 lines=1600"); permet d'augmenter la taille d'affichage de la console.

Entrées / sorties sur fichier

Introduction : Un fichier permet de stocker sur le disque dur, une disquette, un CD, etc, des informations sous formes binaire ou texte. La nature des informations peut être très diverse : on peut stocker dans un même fichier des entiers, des réels, du texte…. Les fichiers sont à accès séquentiel, c'est à dire que l'on accède aux éléments du fichier les uns après les autres. Il est néanmoins possible d'avoir accès à un élément particulier grâce à la fonction « ***fseek*** ». La bibliothèque standard reconnaît trois fichiers particuliers :

- *stdin* : le périphérique par défaut pour lire des caractères, c'est à dire le clavier (utilisé par scanf).
- *stdout* : le périphérique par défaut pour écrire des caractères, c'est à dire l'écran (utilisé par printf).
- *stderr* : le périphérique par défaut pour écrire les messages d'erreur, c'est à dire l'écran.

Ces trois fichiers viennent du monde UNIX.
Pour accéder à un fichier, il faut inclure le fichier <*stdio.h*>. L'accès au fichier doit être encadré par

```
FILE * file = fopen("dico.txt","r"); /* ouverture du fichier */
        lecture du fichier ...
fclose(file); /* fermeture du fichier */
```

On passe à la fonction ***fopen*** le nom du fichier (sous forme d'une chaîne de caractères, ici « dico.txt ») et le type d'accès que l'on désire ("*r*" = en lecture (read), "*w*" = en écriture (write)[4], "*a*"= en écriture à la fin du fichier (append)). La fonction renvoie un pointeur sur un descripteur de fichier (une structure complexe). Ce descripteur de fichier identifie dans le programme le fichier auquel on est en train d'accéder.
Pour écrire et lire dans un fichier, on utilise ***fprintf*** et ***fscanf***. Il y a d'autres fonctions qui permettent de lire /d'écrire...

```
int entier=2;
fprintf(file,"%d",entier); /* écriture d'un entier dans le fichier */
fscanf(file,"%d",&entier); /* lecture d'un entier du fichier */
```

A chaque fois qu'on lit un élément dans un fichier, on avance à l'élément suivant, on peut ainsi récupérer tous les éléments d'un fichier. La fin du fichier est spécifiée par la fonction ***feof(file)*** ou la variable ***EOF (End Of File)***. Le parcours d'un fichier est donc généralement bâti sur le modèle :

```
while(!feof(file)) {
        // lire un élément;
}
```

Chaque fois que vous ouvrez un fichier, il existe en effet un curseur qui indique votre position dans le fichier. Vous pouvez imaginer que c'est exactement comme le curseur de votre éditeur de texte (tel Bloc-Notes). Il indique où vous êtes dans le fichier, et donc où vous allez écrire.

[4] si on utilise le commutateur « w+ », on accède au fichier en lecture **et** écriture

En résumé, le système de curseur vous permet d'aller lire et écrire à une position précise dans le fichier.
Il existe trois fonctions à connaître :

- *ftell* : indique à quelle position vous êtes actuellement dans le fichier ;
  long ftell(FILE pointeurSurFichier);*
  Le nombre renvoyé indique donc la position du curseur dans le fichier.

- *fseek* : positionne le curseur à un endroit précis ;
  int fseek(FILE pointeurSurFichier, long deplacement, int origine);*
  La fonction fseek permet de déplacer le curseur d'un certain nombre de caractères (indiqué par deplacement) à partir de la position indiquée par origine.
  Le nombre deplacement peut être un nombre positif (pour se déplacer en avant), nul (= 0) ou négatif (pour se déplacer en arrière).
  Quant au nombre origine, vous pouvez mettre comme valeur l'une des trois constantes (généralement des define) listées ci-dessous :
  - *SEEK_SET* : indique le début du fichier ;
  - *SEEK_CUR* : indique la position actuelle du curseur ;
  - *SEEK_END* : indique la fin du fichier.

- *rewind* : remet le curseur au début du fichier (c'est équivalent à demander à la fonction « fseek » de positionner le curseur au début).
  void rewind(FILE pointeurSurFichier);*

Enfin on peut clore le fichier en utilisant la fonction « ***fclose*** » définie comme suit :
*fclose(FILE *pointeurSurFichier);*

Exercice : Devin

- Proposer et tester le programme qui écrit un certain nombre de valeurs aléatoires comprises entre 0 et 200 dans un fichier texte. Le nombre de valeurs à écrire dans le fichier sera fourni par l'utilisateur au moment de l'exécution du programme. Chaque valeur sera écrite sur une nouvelle ligne du fichier.
- modifier le code précédent pour que l'on recherche dans le fichier une valeur fournie par l'utilisateur et indique le numéro de la ligne où se trouve celle-ci dans le fichier.

Exercice : Pendu

A partir d'un fichier « dico.txt », piocher un mot au hasard dans ce même fichier.
Le but ici est de faire deviner à un joueur le mot à trouver.
Lorsque le joueur propose une lettre, le programme indique si cette lettre est contenue dans le mot à trouver et affiche où se trouve le caractère dans le mot à deviner.
Au préalable, le programme calculera la taille du mot pioché. On fixera un nombre d'essai pour la divination du mot.
Par exemple :

le mot à deviner est « automatique » pioché dans le fichier « dico.txt »

Le joueur propose la lettre « e », celle-ci est bien dans le motets se trouve en dernière position, le programme affichera et on ne décomptera pas le nombre d'essai.

- - - - - - - - - - e

Le joueur propose la lettre « k », celle ci n'y est pas, on décompte le nombre d'essai. Si le nombre d'essai est à 0 le joueur a perdu !

le joueur propose la lettre « a », alors le programme affichera

a - - - - a - - - - e

Et ainsi de suite jusqu'à ce que toutes les lettres soient trouvées avant le décompte d'essai.

Modifier le code pour que l'on stocke dans un fichier « score.txt », le nom et le score d'un joueur en fonction des différentes parties qui seront jouées :

si un mot est deviné avant le décompte d'essai, le joueur gagne 10 points,
si on devine 90 % du mot, le joueur gagne 5 points
si on devine 50 % du mot, le joueur gagne 1 points
dans tous les autres cas, on ajoute aucun point.