TP2: Programmation Modulaire

**Introduction :**

L'objectif de ce TP est d'apprendre la programmation modulaire avec le langage C++. Le principe de la Programmation modulaire est de diviser le problème en plusieurs sous-problèmes et dont les solutions seront représentés sous formes de **plusieurs modules** (d'où on parle de modulaire) qui se communiquent et qui sont tous pilotés (ou convoqués) par un programme principal.

En C++ (comme en C), un programme peut être divisé en plusieurs fichiers sources. Ces fichiers représentent les différents modules. On distingue entre deux types de fichiers :

1) Fichier principal qui doit contenir le programme **main** et dont l'extension est « **.cpp** » (ou « .c » dans le cas de C)
2) Plusieurs fichiers de bibliothèques avec l'extension « **.h** » (dits aussi fichiers header ou fichiers d'entête). Ces fichiers sont soit prédéfinis dans le langage (comme **<stdio.h>**) ou programmés par le programmeur.

Pour articuler un programme selon une vision modulaire en C++, il faut avoir au moins un fichier **«.cpp»** contenant le main, ensuite plusieurs fichiers **«.h»** qui seront inclus dans le **main** par la directive : `#include <fichier.h>`

Sous Eclipse, après la création de projet (voir les deux figures 1, 2), on obtient l'ensemble de fichier nécessaire inclus le fichier «.cpp » contenant le programme principal.

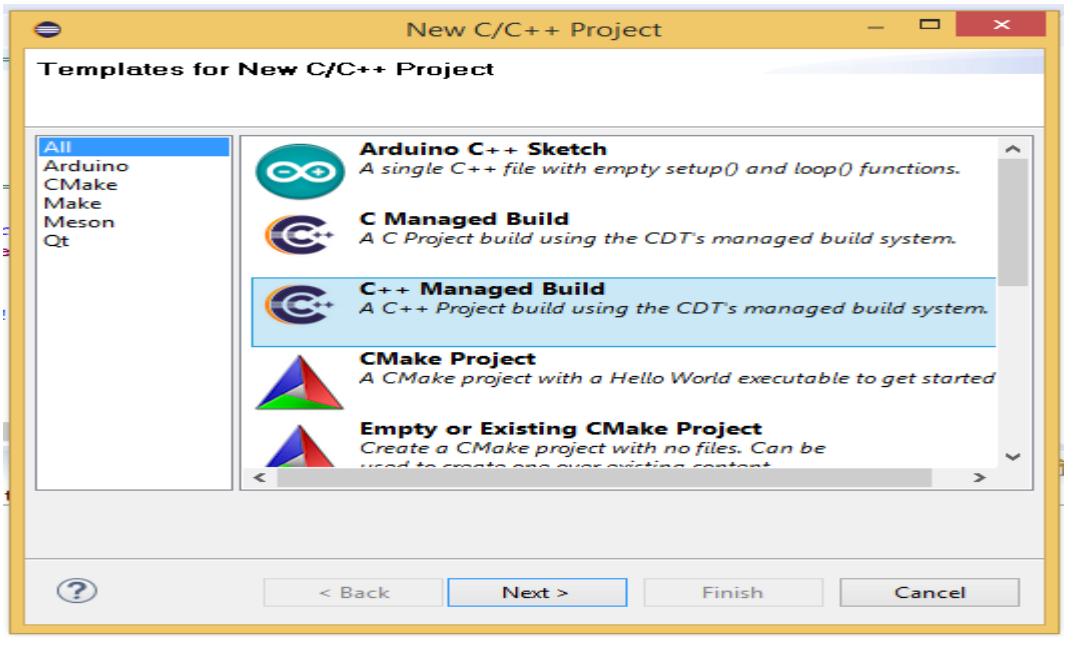


Figure 1 : créer un projet C++ sous Eclipse


Figure 2 : programme p2.cpp

Le programmeur peut ajouter des fichiers de type « .h » à son projet dans le répertoire src dans la hiérarchie des fichiers du projet. (Figure 3)

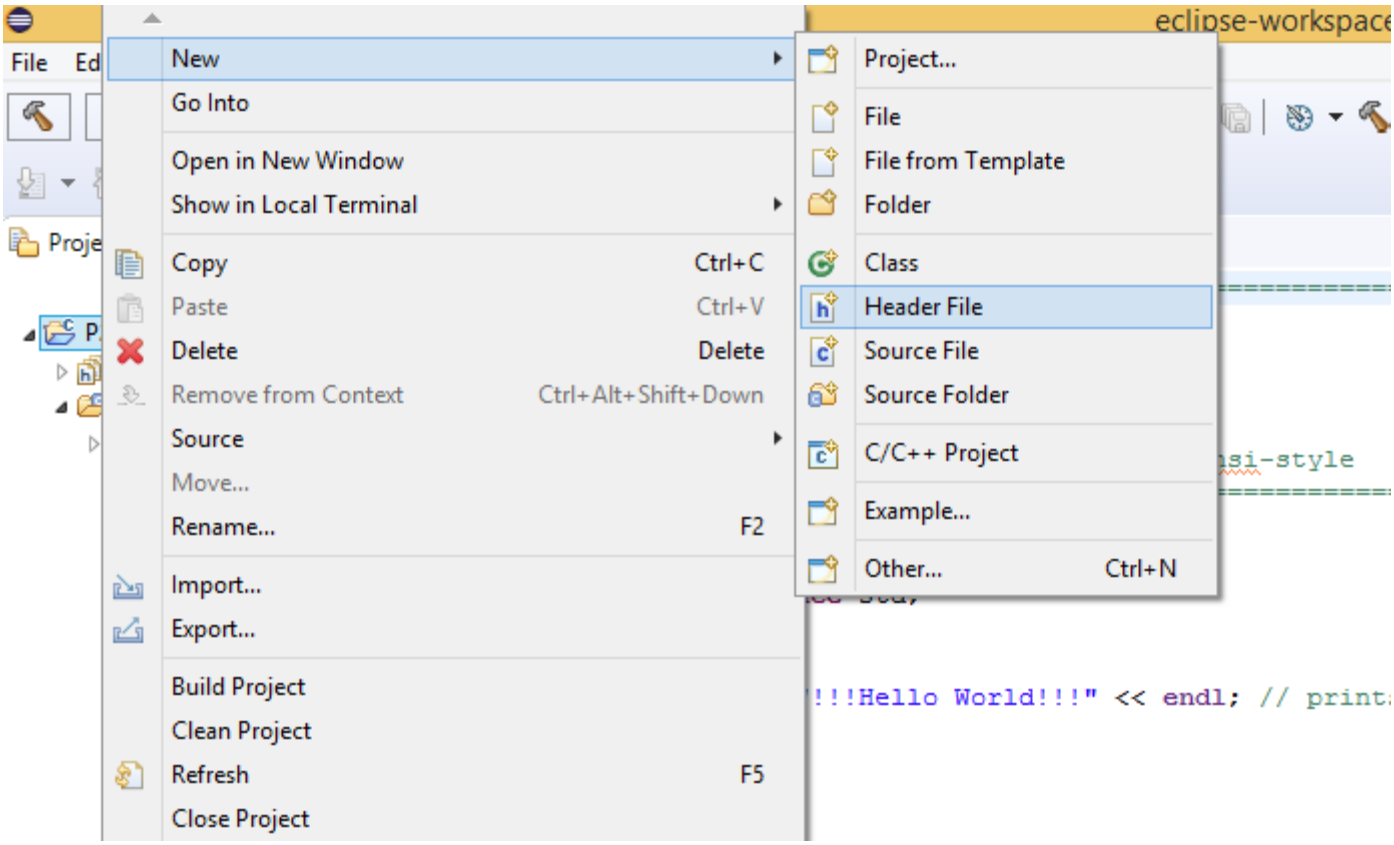


Figure 3 : ajouter un fichier header au projet C++, sous Eclipse

Sur la figure 4, on fixe le nom du fichier header. **Il faut l'écrire avec l'extension « .h » comme sur la figure.**

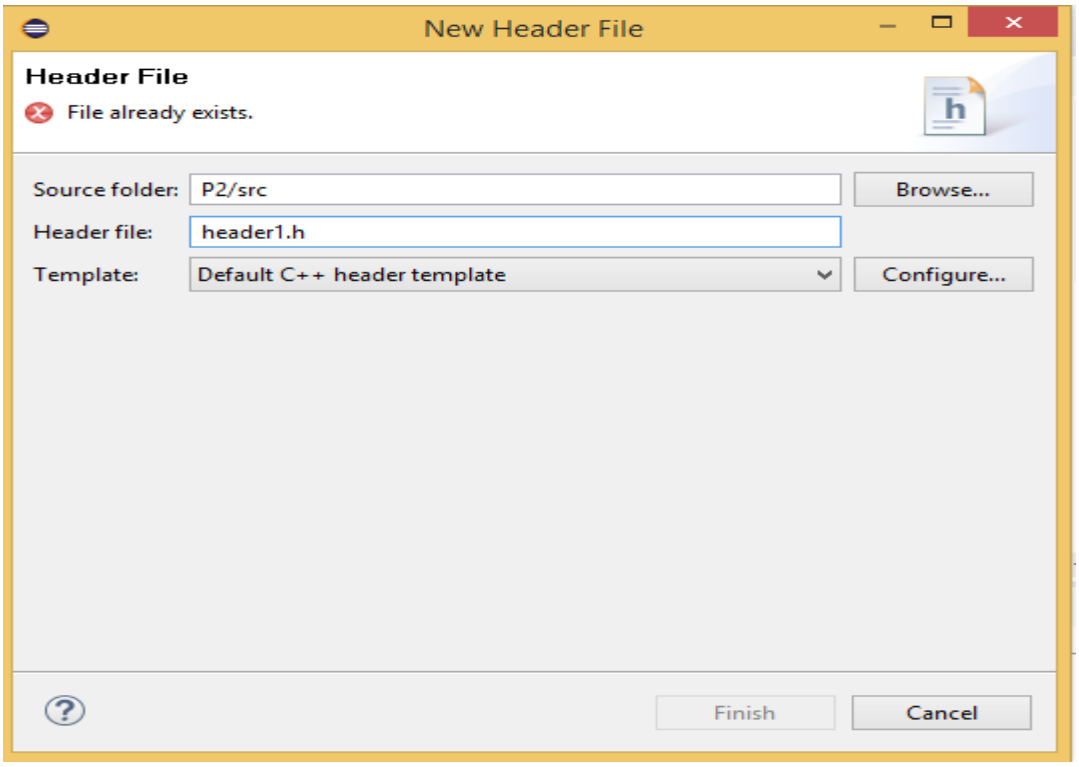


Figure 4 : Donner le nom du fichier « .h »

Après la création du fichier header, il sera ajouté à l'ensemble des fichiers sources comme sur la figure 5.

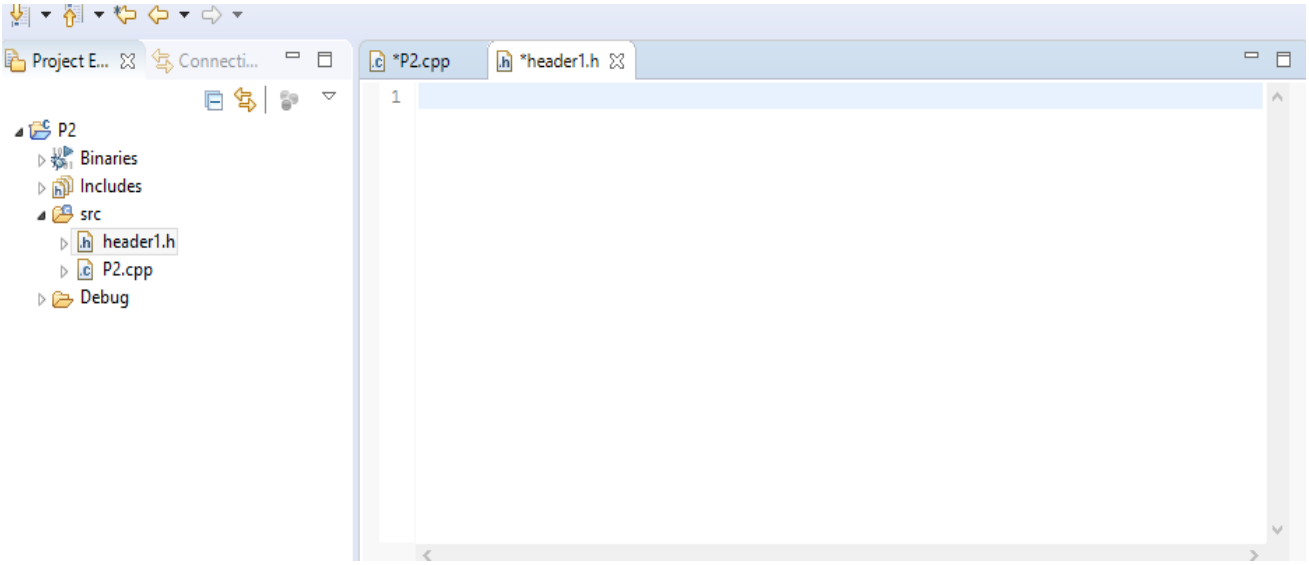


Figure 5 : Créer le fichier header : header1.h

Maintenant, c'est le temps de l'inclure dans le fichier principal .cpp. Voir la figure 6. **Il faut remarquer que cette inclusion s'écrit comme suit : `#include "header1.h"` et non pas comme : `#include <header1.h>`**

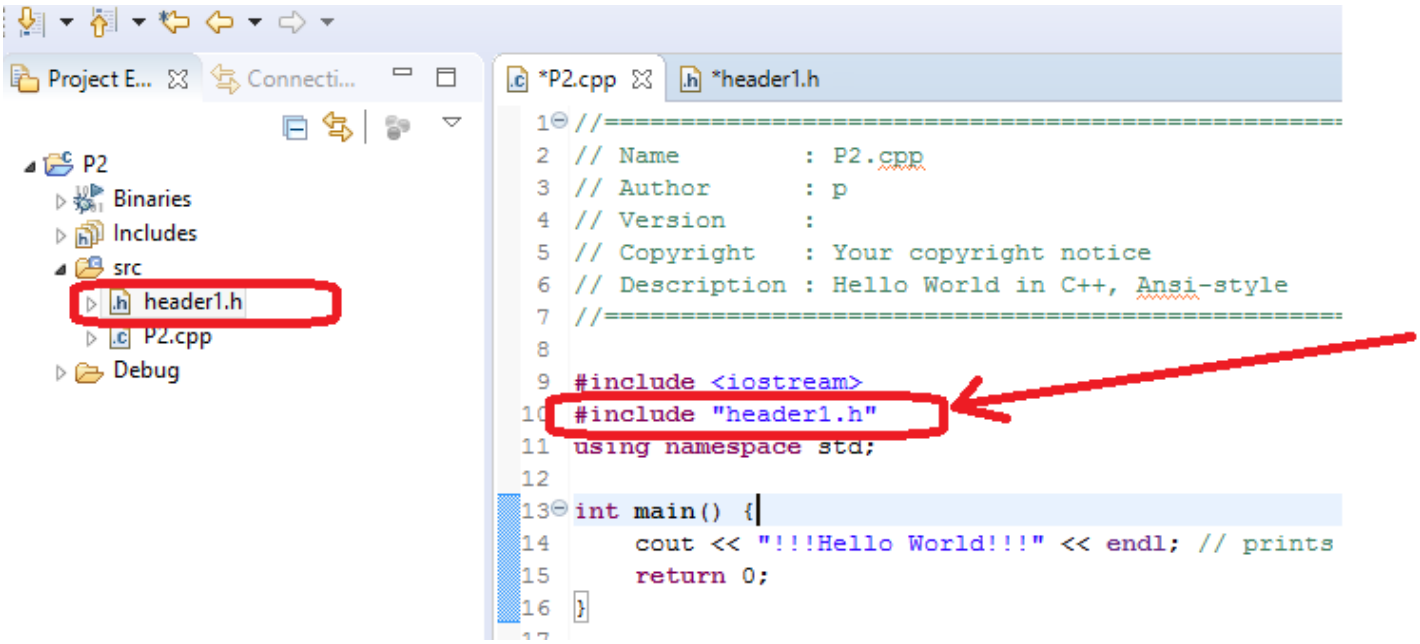


Figure 6 : Créer le fichier header : header1.h

**Exemple simple** : les deux figure 7 et 8 montre un simple usage d'une bibliothèque header1.h contenant une simple fonction LecureX() et qui sera utilisée dans le programme P2.cpp

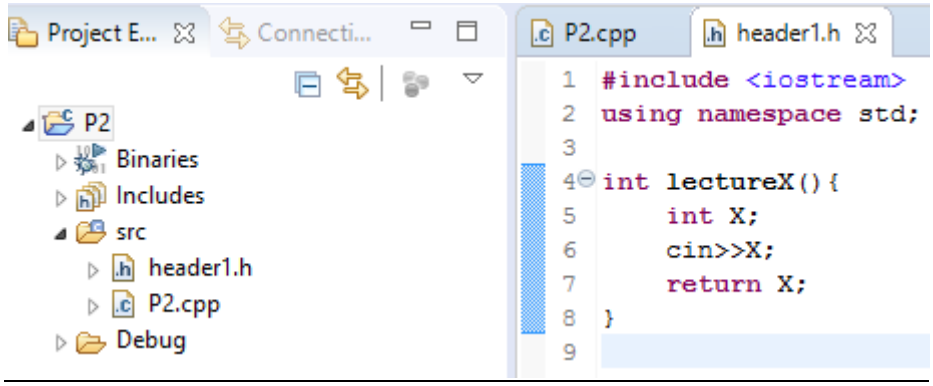


Figure 7 : Une fonction de lecture définie dans header1.h

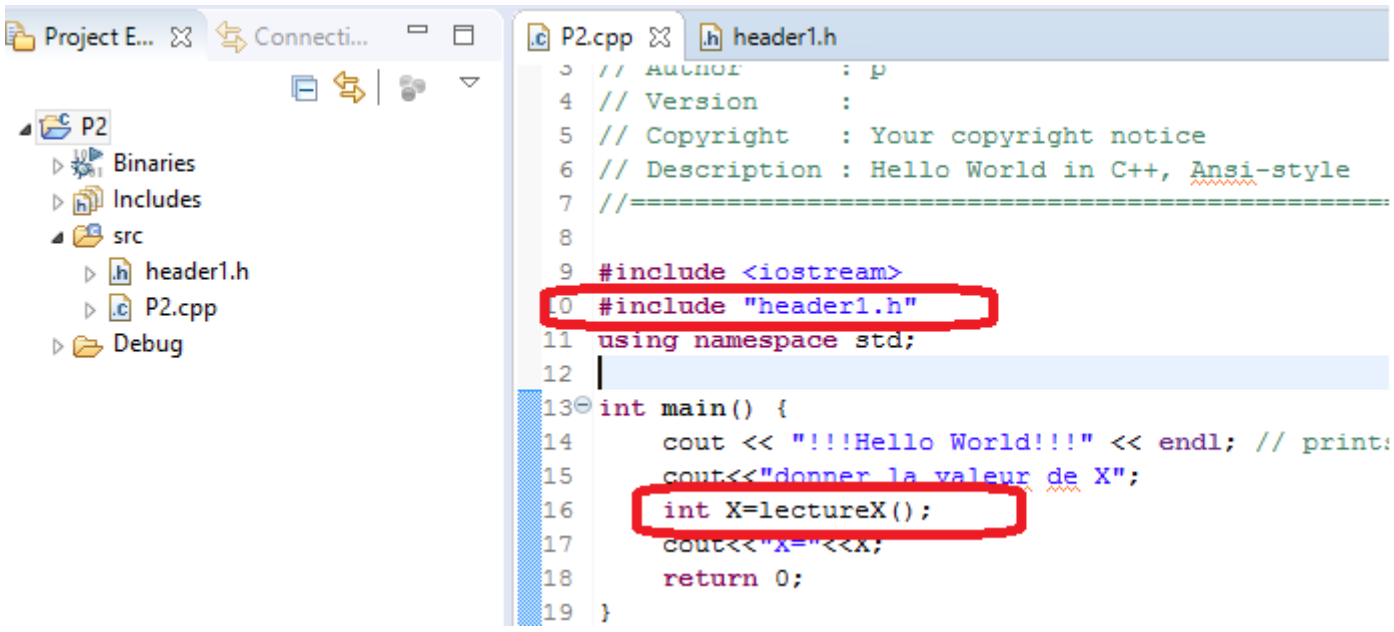


Figure 8 : Une fonction de lecture définie dans header1.h et utilisé dans P2.cpp

**Exercice 1:**

On veut créer une bibliothèque qui contient des traitements sur les tableaux. On va appeler cette bibliothèque : **tableaux.h**. Elle doit contenir les procédures/fonctions suivantes :

1) Lecture et affichage d'un tableau.
2) Recherche de la valeur min et max dans un tableau.
3) Tri d'un tableau par les différentes techniques : tri par insertion, tri par sélection, tris rapide, tri à bulle.
4) Inversement de tableau.
5) Recherche d'une valeur donnée dans un tableau : recherche séquentielle, recherche binaire, recherche dichotomique.

Ecrire un programme principal dans un fichier « .cpp » qui inclut `"tableaux.h"` et qui utilise les différentes fonctions et procédures de cette bibliothèque.

**Exercice 2:**

L'objectif est de réaliser un programme qui simule un jeu dit « pendu ». L'utilisateur joue avec la machine un jeu d'intelligence pour découvrir un mot secret. On vous propose les règles suivantes de ce jeu :

1) Quand l'utilisateur commence le jeu :
   - L'ordinateur génère un mot au hasard et qui sera caché de l'utilisateur
   - L'ordinateur fixe un nombre de tentatives maximal pour cet utilisateur.
2) L'utilisateur entre ensuite un seul caractère (par exemple A) ;
3) Deux cas sont possibles :
   - Le mot contient le caractère : dans ce cas, l'ordinateur affiche le mot avec caractères cachés par *, par exemple : ***A***
   - Le mot ne contient pas le caractère : l'ordinateur affiche : « erreur » et affiche aussi : « il vous reste seulement X tentative ». X est le nombre de tentatives restantes avant que l'utilisateur perd ou gagne.
     La figure suivante montre un scénario possible du jeu.

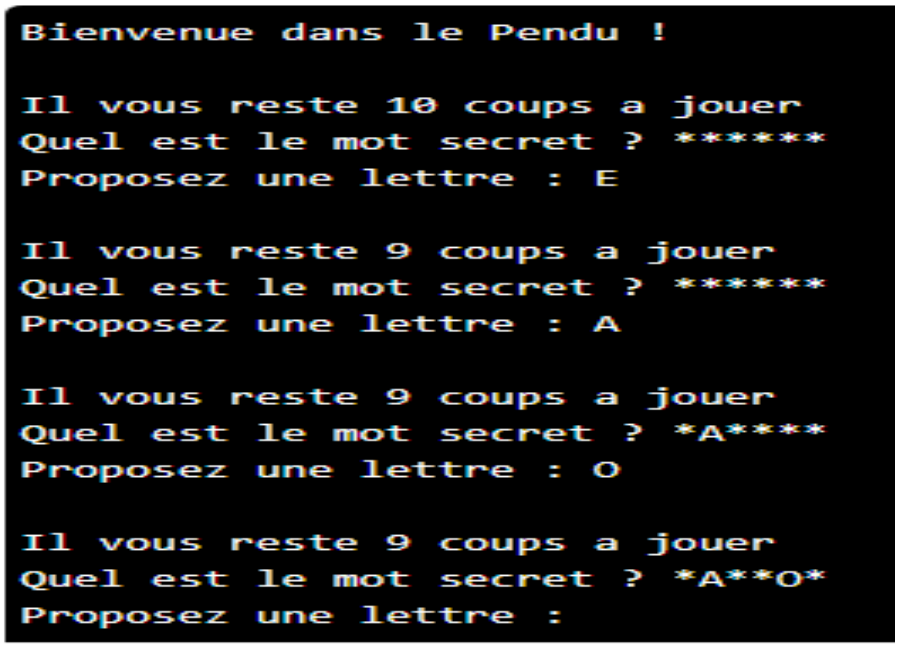


Figure 10. Scénario d'exécution du jeu de pendu.

4) Si à un moment donné, le joueur arrive à connaitre tous les caractères, il est déclaré gagné avec un score : nombre de tentative. Voir la figure suivante :

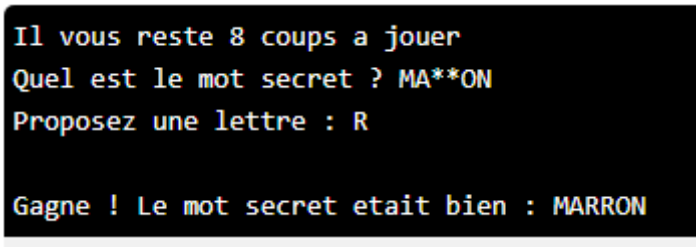


Figure 11. Le joueur a gagné.

**Remarques sur la réalisation de ce jeu :**

1) Tous les mots secrets qu'on peut utiliser dans le jeu sont stockés dans un fichier : il faut remplir ce fichier texte
2) Au début du jeu, l'ordinateur tire au hasard un mot de ce fichier texte.
3) Réaliser un fichier header `"chaines_car.h"` séparé permettant de faire les fonctions suivantes:
   - Retourner un mot (chaîne de caractères) depuis un fichier texte contenant des chaînes de caractères
   - Tester si un mot contient ou non un caractère
4) Le programme principal utilise le fichier header pour réaliser le **main** du jeu
5) Améliorer le jeu dans les sens suivants:
   - Créer un fichier qui va contenir les <noms, score> de tous les joueurs qui jouent le jeu
   - Au début, l'utilisateur doit saisir son nom avant de commencer le jeu
   - L'ordinateur stocke le nom dans le fichier des joueurs, s'il n'est pas déjà stocké
   - A la fin de chaque jeu, le score du joueur sera stocké dans le fichier
   - Il est possible de visualiser au début le dernier score de l'utilisateur si cet utilisateur a déjà joué à ce jeu

   **Remarque** : toutes procédures ou fonctions concernant les fichiers doivent être ajoutées dans le fichier header « chaine_car.h »