Introduction de la programmation   
Projet Premier semestre  
Sujet : Guide des prénoms de Bordeaux

**Justifications**

**techniques**

Sommaire

[A. Fonction principale 3](#_Toc408997250)

[a. La structure Prenom 3](#_Toc408997251)

[b. Le menu (cf. Annexe 1) 3](#_Toc408997252)

[B. Affichage des textes 4](#_Toc408997253)

[a. La structure Texte 4](#_Toc408997254)

[b. Gestion de l’affichage 4](#_Toc408997255)

[c. Choix de la langue 4](#_Toc408997256)

[ANNEXE 6](#_Toc408997257)

[Annexe 1 : Code de la boucle du Menu 6](#_Toc408997258)

# Fonction principale

La fonction principale dans notre programme est la fonction **main**. Cette dernière permettra de charger l’ensemble des données contenues dans le fichier texte, de les mettre sous forme de structure et de les stocker dans un tableau

## La structure Prenom

Lors de l’initialisation de notre programme, chaque ligne du fichier texte est traitée et stockée dans une structure **Prenom** qui est composée de 4 variables :

* Un entier **annee** qui contiendra l’année du prénom.
* Une chaîne de caractères **prenom.**
* Un entier **nombrePrenom** qui contiendra le total de nouveau-né portant ce prénom sur une année.
* Un entier **ordre** qui contiendra l’ordre sur 100 de ce prénom

Nous avons choisi ce format de structure car c’est celui qui se rapproche le plus de la présentation es informations d’origines. Nous stockons ensuite l’ensemble de ces structures dans un fichier texte.

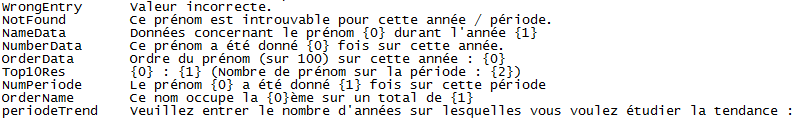
## Le menu (cf. Annexe 1)

Une fois les traitements d’initialisation du programme terminés, nous avons mis en place un menu. Ce menu prend la forme **d’une boucle while** qui tourne en boucle jusqu’à ce que l’utilisateur quitte le programme. Utiliser une fonctionnalité du programme renverra donc l’utilisateur au menu.  
Dans ce menu nous proposons les différents choix possibles à l’utilisateur puis, en fonction de sa réponse, nous utilisons une **structure switch** afin de lancer le traitement voulu. Une fois le traitement terminé, nous réaffichons ce menu.

# Affichage des textes

## La structure Texte

Afin d’optimiser au maximum notre programme, nous avons décidé d’utiliser un système particulier pour gérer l’affichage des informations qui apparaîtront à l’écran. Nous avons stocké l’ensemble des textes de notre programme dans un fichier texte qui possède la structure suivante :



Le premier mot correspond à **l’identifiant** du texte, puis après une tabulation, on trouve le texte qui sera affiché à l’écran.

Lors de l’initialisation de notre programme, nous lisons ce fichier texte et stockons chaque ligne dans une structure **Texte**  qui possède les variables suivantes :

* Une chaîne de caractères **nomTexte** qui contiendra l’identifiant du texte.
* Une chaîne de caractères **texte** qui contiendra le texte à afficher.

Nous stockons ensuite ces structures dans un tableau de **Texte.**

## Gestion de l’affichage

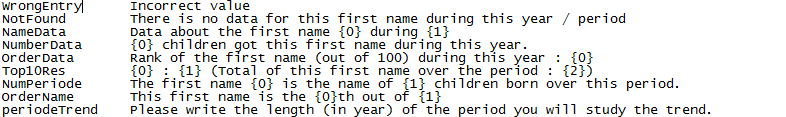
Afin de gérer l’affichage grâce à nos structures **Texte**, nous avons eu besoin de créer une nouvelle fonction. C’est le rôle de la fonction **affichageTexte** : cette fonction prendra en paramètre l’identifiant du texte que l’on souhaite afficher, et après une recherche dans le tableau contenant l’ensemble de nos structure **Texte**, affichera le texte correspondant à l’identifiant.

En utilisant le mot-clé **params** dans la définition de la fonction, il nous a été possible de créer un tableau variable qui contiendra l’ensemble des paramètres que l’on souhaite afficher dans la chaîne de caractères. Par défaut, si l’identifiant n’est pas trouvé dans le tableau, la fonction affichera ce dernier.

L’avantage de cette méthode est qu’on désolidarise en partie le texte affiché du programme. En effet si l’on souhaite modifier le texte il suffit de modifier directement le fichier texte et non plus le texte en brut dans le programme.

## Choix de la langue

L’autre intérêt de cette méthode est qu’il permet au programme de supporter plusieurs langues. Par exemple, dans le cadre de notre programme, nous avons ajouté l’anglais comme langue disponible. Cela s’est traduit par l’ajout d’un second fichier langue qui possède les même identifiant pour le début de chaque ligne mais dont le texte affiché sera différent.



Ainsi si l’on souhaite traduire notre programme dans une autre langue, au lieu de traduire en brut dans le programme ou de recopier une traduction faite par un tiers, le développeur aura juste à rajouter une option correspondant à la nouvelle langue dans la fonction **choixLangue** et donc au nouveau fichier.

Actuellement le souci majeur de notre solution est que les fichiers contenant le texte ne sont pas cryptés et donc modifiables. Pour sécuriser un minimum notre programme et donc cette solution, il faudrait crypter ces fichiers de langues.

# ANNEXE

## Annexe 1 : Code de la boucle du Menu

while (!quitte)

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.Green;

affichageTexte("TitleLine1", texteProgramme);

affichageTexte("TitleLine2", texteProgramme);

affichageTexte("TitleLine3", texteProgramme);

Console.ResetColor();

affichageTexte("MenuLine1", texteProgramme);

affichageTexte("MenuChoice1", texteProgramme);

affichageTexte("MenuChoice2", texteProgramme);

affichageTexte("MenuChoice3", texteProgramme);

affichageTexte("MenuChoice4", texteProgramme);

affichageTexte("MenuChoice5", texteProgramme);

affichageTexte("MenuChoice0", texteProgramme);

choixOk = false;

while (!choixOk){

choixOk = true;

try{

affichageTexte("EnterChoice", texteProgramme);

choix = int.Parse(Console.ReadLine());

}

catch{

affichageTexte("IncorrectValue", texteProgramme);

choixOk = false;

}

}

switch (choix){

case 1:

prenomSurUneAnnee(Donnees, texteProgramme);

break;

case 2:

top10PrenomsPeriode(Donnees, texteProgramme);

break;

case 3:

nomPeriodeDonnee(Donnees, texteProgramme);

break;

case 4:

tendancePrenom(Donnees, texteProgramme);

break;

case 5:

langue = choixLangue();

recuperationTexteProgramme(langue, texteProgramme);

break;

case 6:

triRapideSurPrenom(DonneesTrieSurPrenom, 0, DonneesTrieSurPrenom.Length - 1);

for (int k = 0; k < 500; k++){

Console.WriteLine(DonneesTrieSurPrenom[k].prenom);

}

break;

case 0:

quitte = true;

break;

}