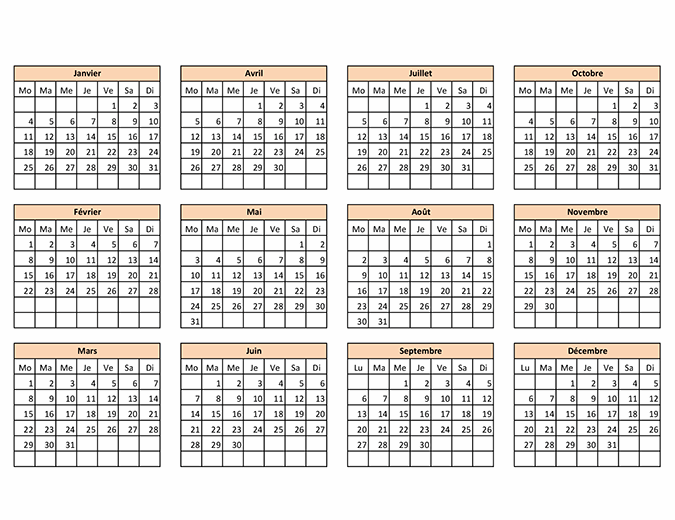
# Rapport de projet Calendrier perpétuel



Syoan ODOUHA| M1 Informatique| 20/11/2022

Table des matières

[Introduction 1](#_Toc119869033)

***Présentation de la problématique ...…………………………………………………………………………………2***

**Travail** **réalisé** ………………………………………………………………………………………………………….

**Conclusion**

**Annexe**

# **Introduction**

Ce rapport porte sur l’analyse du problème d’un calendrier perpétuel et de la démarche utilisé pour développer une application qui résout ce problème.

Afin de traiter ce problème il s’agira de l’analyser et présenter les solutions possibles afin de le résoudre.

Par la suite il sera question de comment est développé l’application qui va répondre à ce problème et comment cette application répond totalement ou non à la problématique.

# Présentation de la problématique

Un calendrier perpétuel indique le jour de la semaine pour n’importe quelle date donnée.

Il existe 2 calendriers, le calendrier julien et le calendrier grégorien. Plusieurs méthodes existent mais la méthode qui sera utilisée pour traiter de ce sujet est le calendrier perpétuel de G.D Moret.

Ce calendrier consiste à utiliser 3 tableaux dans lesquels seront déterminés le siècle l’année, le mois et un quatrième chiffre que l’on appelle le quantième qui est le jour du mois, par exemple, pour le 23/12/2010, le quantième est 23. On obtient un chiffre entre 0 et 6 soit 7 possibilités qui correspond au jour dans la semaine.

Pour ce calendrier de Moret il existe comme dit précédemment une version par tableau, une version mémorisable qui est une version qui simplifie la méthode par tableau et une version simplifiée de la version mémorisable. Pour la résolution du problème de calendrier perpétuel la méthode utilisée sera la version mémorisable.

Cette méthode sera appliquée au calendrier grégorien et non au calendrier julien.   
Le calendrier julien concerne toutes les dates jusqu’au 9 décembre 1582 et le calendrier commence le 20 décembre 1582.

Donc l’application devra respecter cette contrainte qui est de ne pas calculer une date inférieure au 20 décembre 1582.

# Travail réalisé

1. Résonnement algorithmique

Pour calculer le jour de la semaine on doit d’abord calculer 4 autres nombres :

* Le nombre séculaire : qui correspond au siècle de la date
* Le nombre annuel : qui correspond à l’année
* Le nombre mensuel : qui varie selon les mois et en fonction du type d’année (bissextile ou non bissextile)
* Le quantième : qui correspond au jour dans le mois, par exemple (21/11/2022), le quantième = 21
* Le nombre séculaire ne se calcule pas vraiment mais suit une suite logique :

1582 à 1599 : 1  
1600 à 1699 : 0  
1700 à 1799 : 5  
1800 à 1899 : 3  
1900 à 1999 : 1  
2000 à 2099 : 0  
2100 à 2199 : 5

On remarque que c’est une suite [1-0-5-3], qui répète à chaque fois.

* Le nombre annuel est calculer de la manière suivante :

(annee%100) + (annee%100)/4 – 5

* Le nombre annuel est défini en fonction du tableau suivant :

|  |  |
| --- | --- |
| Mois | Nombre mensuel |
| février (année non bissextile), mars, novembre | 0 |
| juin | 1 |
| septembre, décembre | 2 |
| janvier (année bissextile), avril, juillet | 3 |
| janvier (année non bissextile), octobre | 4 |
| mai | 5 |
| février (année bissextile), août | 6 |

* Le quantième est le numéro du jour dans le mois

1. Programmation de l’application

Dans l’application il y a 3 fonctionnalités, une qui détermine à quel jour de la semaine il s’agit, une qui liste les vendredis 13 dans une année, s’il y en a plusieurs et la dernière qui permet l’édition d’un calendrier.

Pour que ces 3 fonctionnalités il faut utiliser ces 4 fonctions suivantes :

*// fonction qui calcule le nombre seculaire*

*void* nbre\_seculaire(*int* *&jour*, *int* *&mois*, *int* *&annee*, *int* *&nbre\_secu*){

*int* siecle\_ref = 15;

*int* nb\_sec\_ref = 1;

*nbre\_secu* = nb\_sec\_ref;

*int* s = *annee*/100;

while (siecle\_ref != s)

{

if (*nbre\_secu* == 1)

*nbre\_secu* = 0;

else if (*nbre\_secu* == 0)

*nbre\_secu* = 5;

else if (*nbre\_secu* == 5)

*nbre\_secu* = 3;

else if (*nbre\_secu* == 3)

*nbre\_secu* = 1;

siecle\_ref++;

}

}

*// nombre annuel*

*void* nbre\_annuel(*int* *&jour*, *int* *&mois*, *int* *&annee*, *int* *&annuel*){

*annuel* = (*annee*%100) + (*annee*%100)/4 - 5;

}

Comme vu précédemment, cette fonction utilise la formule pour calculer le nombre annuel et renvoie ce résultat.

*// fontion qui calcule le nombre mensuel*

*void* nbre\_mensuel(*int* *&mois*, *int* *&annee*,*int* *&mensuel*){

*// si l'année est bisextile*

if(*annee* % 4 == 0 && *annee* % 100 != 0 || *annee* % 400 == 0){

if(*mois* == 1){

*mensuel* = 3;

}

if(*mois* == 2){

*mensuel* = 6;

}

}

else{

if(*mois* == 1){

*mensuel* = 4;

}

if(*mois* == 2){

*mensuel* = 0;

}

}

*// pour les autres mois quelque soit le type de l'année*

if (*mois* == 3 || *mois* == 11){

*mensuel* = 0;

}

if (*mois* == 6){

*mensuel* = 1;

}

if (*mois* == 9 || *mois* == 12){

*mensuel* = 2;

}

if (*mois* == 4 || *mois* == 7){

*mensuel* = 3;

}

if (*mois* == 10){

*mensuel* = 4;

}

if (*mois* == 5){

*mensuel* = 5;

}

if (*mois* == 8){

*mensuel* = 6;

}

}

La fonction attribue une valeur en fonction des mois et du type d’année (bissextile ou pas).

*// fonction qui calcule le quantième*

*void* nbre\_quantieme(*int* *&jour*, *int* *&quantieme*){

*quantieme* = *jour*;

}

Ces fonctions calculent les 4 nombres vu précédemment qui vont permettre les actions suivantes.

Pour calculer le jour de la semaine il faut la date entière, pour cela on a une fonction getDate qui va récupérer une date qui sera valide, c’est-à-dire une date qui va respecter toutes les règles d’un calendrier : pas de 29 février si l’année n’est pas bissextile, la saisie obligatoire d’une date supérieure au 20 décembre 1582.

*// fonction qui recupère une date valide*

*void* getDate(*int* *&jour*, *int* *&mois*, *int* *&annee*)

{

cout << "Entrez une date valide (jj/mm/aaaa) supérieur ou égale au 20 décembre 1582 : \n";

cin >> *jour*;

cin.ignore();

cin >> *mois*;

cin.ignore();

cin >> *annee*;

*// année bissextile*

if (*annee* % 4 == 0 && *annee* % 100 != 0 || *annee* % 400 == 0)

{

if (*mois* == 2 && *jour* > 29)

{

cout << "Date invalide" << endl;

getDate(*jour*, *mois*, *annee*);

}

}

else

{

if (*mois* == 2 && *jour* > 28)

{

cout << "Date invalide" << endl;

getDate(*jour*, *mois*, *annee*);

}

}

*// mois de 30 jours*

if (*mois* == 4 || *mois* == 6 || *mois* == 9 || *mois* == 11)

{

if (*jour* > 30)

{

cout << "Date invalide" << endl;

getDate(*jour*, *mois*, *annee*);

}

}

*// mois de 31 jours*

if (*mois* == 1 || *mois* == 3 || *mois* == 5 || *mois* == 7 || *mois* == 8 || *mois* == 10 || *mois* == 12)

{

if (*jour* > 31)

{

cout << "Date invalide" << endl;

getDate(*jour*, *mois*, *annee*);

}

}

*// mois invalide*

if (*mois* > 12 || *mois* < 1)

{

cout << "Date invalide" << endl;

getDate(*jour*, *mois*, *annee*);

}

*// date inférieure au 20 décembre 1582*

if (*annee* < 1582)

{

cout << "Date invalide" << endl;

getDate(*jour*, *mois*, *annee*);

}

else if (*annee* == 1582)

{

if (*mois* < 12)

{

cout << "Date invalide" << endl;

getDate(*jour*, *mois*, *annee*);

}

else if (*mois* == 12)

{

if (*jour* < 20)

{

cout << "Date invalide" << endl;

getDate(*jour*, *mois*, *annee*);

}

}

}

}

1. Résultats

Lorsque l’on a calculé les 4 nombres, on peut calculer le jour de la semaine. La formule pour calculer le jour est la division euclidienne de la somme de tous les nombres obtenus par 7. Le reste de la division est le numéro du jour de la semaine

*void* jour\_semaine(*int* *&nbre\_secu*, *int* *&annuel*, *int* *&mensuel*, *int* *&quantieme*, *int&res*){

*res* = (*nbre\_secu* + *annuel* + *mensuel* + *quantieme*) % 7;

}

Pour l’affichage, on convertit le numéro obtenu par le jour de la semaine

*// fonction qui affiche le jour de la semaine*

*void* affiche\_jour\_semaine(*int* *&res*){

if(*res* == 0){

cout << "Dimanche" << endl;

}

if(*res* == 1){

cout << "Lundi" << endl;

}

if(*res* == 2){

cout << "Mardi" << endl;

}

if(*res* == 3){

cout << "Mercredi" << endl;

}

if(*res* == 4){

cout << "Jeudi" << endl;

}

if(*res* == 5){

cout << "Vendredi" << endl;

}

if(*res* == 6){

cout << "Samedi" << endl;

}

}

Les autres options reprennent les mêmes fonctions. Pour savoir dans une année quel est le ou les vendredis 13 on calcule les 13 jours pour tous les mois, ensuite on détermine quels sont ceux qui sont des vendredis.

*// fonction 2 : vendredi 13*

*void* option2(*int* *&jour*, *int* *&mois*, *int* *&annee*, *int* *&nbre\_secu*, *int* *&annuel*, *int* *&mensuel*, *int* *&quantieme*, *int* *&res*){

*// saisir une année*

cout << "Saisir une année : ";

cin >> *annee*;

*// calculer le jour de la semaine pour tous les 13 du mois*

for (*int* i = 1; i <= 12; i++)

{

*mois* = i;

*jour* = 13;

nbre\_seculaire(*jour*, *mois*, *annee*, *nbre\_secu*);

nbre\_annuel(*jour*, *mois*, *annee*, *annuel*);

nbre\_mensuel(*mois*, *annee*, *mensuel*);

nbre\_quantieme(*jour*, *quantieme*);

jour\_semaine(*nbre\_secu*, *annuel*, *mensuel*, *quantieme*, *res*);

if (*res* == 5)

{

cout << "Le 13 " << *mois* << " " << *annee* << " est un vendredi" << endl;

}

}

}

Pour l’édition de calendrier, on calcule tous les jours de l’année et on écrit toutes les dates dans un fichier.

*// fonction 3 : edition de calendrier*

*void* option3(*int* *&jour*, *int* *&mois*, *int* *&annee*, *int* *&nbre\_secu*, *int* *&annuel*, *int* *&mensuel*, *int* *&quantieme*, *int* *&res*){

*// saisir une année*

cout << "Saisir une année : ";

cin >> *annee*;

*//convertir le res en jour de la semaine stocker dans un tableau*

string jours[7] = {"Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"};

*//ecriture dans un fichier*

ofstream fichier("calendrier.txt", ios::out | ios::trunc); *//déclaration du flux et ouverture du fichier*

*// si l'ouverture a réussi*

if(fichier)

{

*// gerer les mois qui ont 31 jours, 30 jours, 28 ou 29 jours pour fevrier*

for (*int* i = 1; i <= 12; i++)

{

*mois* = i;

if (*mois* == 1 || *mois* == 3 || *mois* == 5 || *mois* == 7 || *mois* == 8 || *mois* == 10 || *mois* == 12)

{

for (*int* j = 1; j <= 31; j++)

{

*jour* = j;

nbre\_seculaire(*jour*, *mois*, *annee*, *nbre\_secu*);

nbre\_annuel(*jour*, *mois*, *annee*, *annuel*);

nbre\_mensuel(*mois*, *annee*, *mensuel*);

nbre\_quantieme(*jour*, *quantieme*);

jour\_semaine(*nbre\_secu*, *annuel*, *mensuel*, *quantieme*, *res*);

fichier << *jour* << "-" << *mois* << "-" << *annee* << " " << jours[*res*] << endl;

}

}

if (*mois* == 4 || *mois* == 6 || *mois* == 9 || *mois* == 11)

{

for (*int* j = 1; j <= 30; j++)

{

*jour* = j;

nbre\_seculaire(*jour*, *mois*, *annee*, *nbre\_secu*);

nbre\_annuel(*jour*, *mois*, *annee*, *annuel*);

nbre\_mensuel(*mois*, *annee*, *mensuel*);

nbre\_quantieme(*jour*, *quantieme*);

jour\_semaine(*nbre\_secu*, *annuel*, *mensuel*, *quantieme*, *res*);

fichier << *jour* << "-" << *mois* << "-" << *annee* << " " << jours[*res*] << endl;

}

}

if (*mois* == 2)

{

if(*annee* % 4 == 0 && *annee* % 100 != 0 || *annee* % 400 == 0){

for (*int* j = 1; j <= 29; j++)

{

*jour* = j;

nbre\_seculaire(*jour*, *mois*, *annee*, *nbre\_secu*);

nbre\_annuel(*jour*, *mois*, *annee*, *annuel*);

nbre\_mensuel(*mois*, *annee*, *mensuel*);

nbre\_quantieme(*jour*, *quantieme*);

jour\_semaine(*nbre\_secu*, *annuel*, *mensuel*, *quantieme*, *res*);

fichier << *jour* << "-" << *mois* << "-" << *annee* << " " << jours[*res*] << endl;

}

}

else{

for (*int* j = 1; j <= 28; j++)

{

*jour* = j;

nbre\_seculaire(*jour*, *mois*, *annee*, *nbre\_secu*);

nbre\_annuel(*jour*, *mois*, *annee*, *annuel*);

nbre\_mensuel(*mois*, *annee*, *mensuel*);

nbre\_quantieme(*jour*, *quantieme*);

jour\_semaine(*nbre\_secu*, *annuel*, *mensuel*, *quantieme*, *res*);

fichier << *jour* << "-" << *mois* << "-" << *annee* << " " << jours[*res*] << endl;

}

}

}

}

*// on ferme le fichier*

fichier.close();

}

else *// sinon*

cerr << "Impossible d'ouvrir le fichier !" << endl;

}

# Conclusion

# Annexe