

Rapport de projet

Calendrier perpétuel

Janvier	Février	Mars	Avril	Mai	Juin	Juillet	Octobre
Mo Ma Me Je Ve Sa Di	Mo Ma Me Je Ve Sa Di	Mo Ma Me Je Ve Sa Di	Mo Ma Me Je Ve Sa Di	Mo Ma Me Je Ve Sa Di	Mo Ma Me Je Ve Sa Di	Mo Ma Me Je Ve Sa Di	Mo Ma Me Je Ve Sa Di
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31



Syoan ODOUHA | M1 Informatique | 20/11/2022

Table des matières

<i>Introduction</i>	3
<i>Présentation de la problématique</i>	4
Travail réalisé	5
1)Raisonnement algorithmique	
2)Programmation de l'application	
3)Résultats	
Conclusion	15
Annexe	16

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 correspondent 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 :
 $(\text{annee} \% 100) + (\text{annee} \% 100) / 4 - 5$
- 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

2) 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.

```
// fonction 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);
            }
        }
    }
}
}

```

3) 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 supérieure à 1582 : ";
    cin >> annee;

    // verifier si l'annee est supérieure à 1582
    while(annee < 1582)
    {
        cout << "Date invalide" << endl;
        cout << "Saisir une année supérieure à 1582 : ";
        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 au format txt.

```

// 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;

    while(annee < 1582)
    {
        cout << "Date invalide" << endl;
        cout << "Saisir une année supérieure à 1582 : ";
        cin >> annee;
    }

    //convertir le res en jour de la semaine stocker dans un tableau
    string jours[7] = {"Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
"Vendredi", "Samedi"};

    //écriture 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

La méthode utilisée fonctionne parfaitement pour les années supérieurs à 1582. Une intégration avec la méthode de julien aurait permis de ne pas avoir de contraintes de date minimum. Certaines fonctionnalités peuvent être améliorés comme la recherche de vendredis 13.

Annexe

```
M1_calendrierP -- main -- 80x24
syo@Macbook M1_calendrierP % ./main
1. Jour de la semaine
2. Vendredi 13
3. Edition de calendrier
4. Quitter
Votre choix :
█
```

```
M1_calendrierP -- zsh -- 80x24
syo@Macbook M1_calendrierP % ./main
1. Jour de la semaine
2. Vendredi 13
3. Edition de calendrier
4. Quitter
Votre choix :
1
Entrez une date valide (jj/mm/aaaa) supérieur ou égale au 20 décembre 1582 :
21/11/2022
Lundi
syo@Macbook M1_calendrierP % █
```



```
M1_calendrierP — zsh — 80x24
syo@Macbook M1_calendrierP % ./main
1. Jour de la semaine
2. Vendredi 13
3. Edition de calendrier
4. Quitter
Votre choix :
2
Saisir une année supérieure à 1582 : 2015
Le 13 2 2015 est un vendredi
Le 13 3 2015 est un vendredi
Le 13 11 2015 est un vendredi
syo@Macbook M1_calendrierP %
```

```
M1_calendrierP — zsh — 80x24
syo@Macbook M1_calendrierP % ./main
1. Jour de la semaine
2. Vendredi 13
3. Edition de calendrier
4. Quitter
Votre choix :
3
Saisir une année : 2022
syo@Macbook M1_calendrierP %
```

```
1 1-1-2022 Samedi
2 2-1-2022 Dimanche
3 3-1-2022 Lundi
4 4-1-2022 Mardi
5 5-1-2022 Mercredi
6 6-1-2022 Jeudi
7 7-1-2022 Vendredi
8 8-1-2022 Samedi
9 9-1-2022 Dimanche
10 10-1-2022 Lundi
11 11-1-2022 Mardi
12 12-1-2022 Mercredi
13 13-1-2022 Jeudi
14 14-1-2022 Vendredi
15 15-1-2022 Samedi
16 16-1-2022 Dimanche
17 17-1-2022 Lundi
18 18-1-2022 Mardi
19 19-1-2022 Mercredi
20 20-1-2022 Jeudi
21 21-1-2022 Vendredi
22 22-1-2022 Samedi
23 23-1-2022 Dimanche
24 24-1-2022 Lundi
25 25-1-2022 Mardi
26 26-1-2022 Mercredi
27 27-1-2022 Jeudi
28 28-1-2022 Vendredi
29 29-1-2022 Samedi
30 30-1-2022 Dimanche
31 31-1-2022 Lundi
32 1-2-2022 Mardi
33 2-2-2022 Mercredi
34 3-2-2022 Jeudi
35 4-2-2022 Vendredi
36 5-2-2022 Samedi
37 6-2-2022 Dimanche
```

Lien github : https://github.com/kemuelos/M1_calendrierP