

PoP C++ Série 4 corrigé

Première partie : opérateurs bit à bit

Exercice 1.2 (niveau 1) : fonctions de manipulation à l'échelle du bit ou groupe de bits.

```
unsigned int f1(unsigned int n, unsigned int index)
{
    if(index < 32)
    {
        return (n & (1<<index))>> index;
    }

    cout << "Index must be within [0, 31] instead of : "
         << index << endl;

    exit(0);
}

unsigned int f2(unsigned int n, unsigned int index,
               unsigned int val)
{
    if(index < 32 and val < 2)
    {
        return (n & ~(1<<index)) | (val << index);
    }

    cout << "Index must be within [0, 31] "
         << " and val must be only 0 or 1" << endl;

    exit(0);
}

unsigned int f3(unsigned int n, unsigned int width,
               unsigned int shift, unsigned int val)
{
    if( 1< width and width < 32 and
        shift < 32-width and val < (1<<width))
    {
        unsigned int mask((1<<width) -1);
        return (n & ~(mask<<shift)) | (val << shift);
    }

    cout << "width must be within [1, 32], "
         << " shift within [0, 32-width]"
         << " and val within [0, 2^width-1]" << endl;

    exit(0);
}
```

Seconde partie : static à l'échelle d'un module

Exercice 2.2 (niveau 2) : gestion d'un agenda

On testera les modules **date** et **event** au fur et à mesure des questions avec un module de niveau supérieur **test** qui inclut leur interface et appelle les fonctions exportées.

b) module **date** gérant une classe **Date** de calendrier

Ecrire un module responsable d'une classe **Date** avec 3 attributs privés : **jour, mois, annee**.

Interface du module : date.h

```
#ifndef DATE_H
#define DATE_H
// version : 1.0

class Date
{
public:
    Date(unsigned dd, unsigned mm, unsigned yy);
    unsigned nb_of_days_between_dates(const Date& d) const;
    bool operator==(const Date& b);
    bool operator!=(const Date& b);
    void affiche() const;

private:
    unsigned day;
    unsigned month;
    unsigned year;
};

#endif
```

Implémentation du module : date.cc

On remarquera que les fonctions utilitaires sont déclarées en static et donc sont invisible en dehors du module.

```
// fichier : date.cc
// version : 1.0
//
#include <iostream>
#include <limits>
#include <vector>

using namespace std;
#include "date.h"

// zero ref for current calendar
static constexpr unsigned starting_year(1600);

static bool date_correct(unsigned dd, unsigned mm, unsigned yy);
static unsigned yy_is_leap_year(unsigned yy);
static unsigned day_per_month (unsigned mm, unsigned yy);
static unsigned day_per_year(unsigned yy);
static unsigned nb_of_days_since_starting_year(unsigned dd,
                                                unsigned mm, unsigned yy);

Date::Date(unsigned dd, unsigned mm, unsigned yy)
{
    if(date_correct(dd,mm,yy))
```

```

        {
            day   = dd;
            month = mm;
            year  = yy;
        }
    else
    {
        cout << "Date impossible dans le calendrier actuel\n" << endl;
        cout << "La Date est initialisée au 1.1.1600 \n" << endl;
        day   = 1;
        month = 1;
        year  = 1600;
    }
}

bool Date::operator==(const Date& b)
{
    return(day   == b.day   &&
           month == b.month &&
           year  == b.year );
}

bool Date::operator!=(const Date& b)
{
    return( not ((*this) == b));
}

// the special character '\t' is a tabulation
void Date::affiche() const
{
    cout << day << " \t" << month << " \t" << year << endl;
}

unsigned Date::nb_of_days_between_dates(const Date& d) const
{
    return abs(int(nb_of_days_since_starting_year(day   ,month, year)) -
              int(nb_of_days_since_starting_year(d.day,d.month,d.year)));
}

static bool date_correct(unsigned dd, unsigned mm, unsigned yy)
{
    if ((mm < 1) || (mm > 12))
        return false;

    if ((dd < 1) || (dd > day_per_month (mm, yy)))
        return false;

    if(yy < starting_year)
        return false;

    return true;
}

static unsigned yy_is_leap_year(unsigned yy)
{
    if((yy % 400) == 0)
        return true;
    else if((yy % 100) == 0)
        return false;
    else
        return ((yy % 4) == 0);
}

static unsigned day_per_year(unsigned yy)
{
    return yy_is_leap_year (yy) ? 366 : 365;
}

```

```

static unsigned day_per_month (unsigned mm, unsigned yy)
{
    switch (mm)
    {
        case 4 :
        case 6 :
        case 9 :
        case 11:
            return 30;
        case 1 :
        case 3 :
        case 5 :
        case 7 :
        case 8 :
        case 10:
        case 12:
            return 31;
        case 2 :
            return yy_is_leap_year (yy) ? 29 : 28;
        default:
            cout << "Erreur: je ne connais pas le mois "
                 << mm << endl;
            return 0;
    }
}
static unsigned nb_of_days_since_starting_year(unsigned dd, unsigned mm, unsigned
yy)
{
    unsigned total(0);

    for (unsigned i(starting_year); i < yy; i++)
        total += day_per_year (i);

    for (unsigned i(1); i < mm; i++)
        total += day_per_month (i, yy);

    return total + dd - 1;
}

```

e) module event gérant une classe Event et un ensemble d'instance de ce type dans un tableau dynamique **agenda** restant confidentiel au niveau du module event.

La classe Event contient les attributs privés : **nom** et **lieu** de type string, et **date** de type Date.

Interface du module : event.h

```

#ifndef EVENT_H
#define EVENT_H
// version : 1.0
#include <string>
#include "date.h"

class Event
{
public:
    Event(string nom, string lieu, Date date)
        :nom(nom), lieu(lieu), date(date){};
    bool event_add_to_database();
    bool operator==(const Event& b);
    bool operator!=(const Event& b);
    void affiche() const;

private:
    string nom;
    string lieu;
    Date date;
};
void event_display_all();
#endif

```

Implémentation du module : event.cc

```
// fichier : event.cc
// version : 1.0
//
#include <iostream>
#include <limits>
#include <vector>

using namespace std;
#include "event.h"

static vector<Event> tab; // vide à l'initialisation

bool Event::event_add_to_database()
{
    for(size_t i(0) ; i < tab.size() ; ++i)
    {
        if(tab[i] == *this )
        {
            cout << "Erreur: Event déjà présent dans agenda !" << endl ;
            return false;
        }

        if(tab[i].lieu == this->lieu)
        {
            if(tab[i].date.nb_of_days_between_dates(this->date) < 7)
            {
                cout << "Cet autre Event est trop proche !" ;
                cout << endl ;
                tab[i].affiche();
                cout << endl;
                return false;
            }
        }
    }
    tab.push_back(*this);
    return true;
}

bool Event::operator==(const Event& b)
{
    return( nom == b.nom &&
           lieu == b.lieu &&
           date == b.date);
}

bool Event::operator!=(const Event& b)
{
    return( not ((*this)==b));
}

void Event::affiche() const
{
    cout << endl << "Nom : " << nom << endl
         << "Lieu : " << lieu << endl
         << "Date : " ;
    date.affiche() ;
}

// the special character '\t' is a tabulation
void event_display_all()
{
    for( const Event& e: tab) e.affiche();
}
```