Structures conditionnelles [if] - Sujets d'examens

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 15 mai 2018

Table des matières

C++ - Sujets d'examens (Solution)

1 Problème de divisibilité (2 points)

Objectif
Affichez la divisibilité d'un entier.

(0.5 point) Écrivez un programme qui saisit un entier dans `n`.
Affichez l'invite :

```
Votre entier?
```

(1.5 point) Indiquez si `n` est divisible par 3 seulement, par 5 seulement, ni par 3 ni par 5, ou bien encore par 3 et par 5.

Testez.

2 Abonnements téléphoniques / pgtelecom (4 points)

Objectif

Vous désirez comparer deux offres d'abonnement téléphonique. La facture est calculée avec un fixe (somme à payer obligatoirement tous les mois) et une partie proportionnelle au temps (en minutes) passé à téléphoner.

Offre	Fixe	Prix à la minute
Telecom 1	10€	0.50€
Telecom 2	15€	0.40€

(0.5 point) Écrivez un programme qui saisit la consommation moyenne mensuelle (en minutes) dans `consom` (réel). Affichez l'invite :

```
Consommation moyenne (en minutes)?
```

(0.5 point) Calculez les tarifs des deux offres :

- Dans `tf1` (réel) pour Télécom 1.
- Dans `tf2` (réel) pour Télécom 2.

(0.5 point) Affichez (ou `[x]` désigne le contenu de `x`) :

```
Avec Telecom1 = [tf1] euros
Avec Telecom2 = [tf2] euros
```

(1.5 point) Affichez l'opérateur le plus intéressant, c.-à-d. l'**un** des messages :

```
==> Prenez Telecom1
==> Prenez Telecom2
==> Prenez l'un ou l'autre
```

(1 point) Calculez et affichez à partir de combien de minutes l'opérateur Telecom2 est plus intéressant que Telecom1, à savoir :

$$(fixe2 - fixe1)/(prix1 - prix2)$$

Testez. Exemples d'exécution :

```
Consommation moyenne (en mn)? 30
Avec Telecom1 = 25 euros
Avec Telecom2 = 27 euros
==> Prenez Telecom1
Prenez Telecom2 a partir de 50 mn
```

```
Consommation moyenne (en mn)? 50
Avec Telecom1 = 35 euros
Avec Telecom2 = 35 euros
==> Prenez l'un ou l'autre
Prenez Telecom2 a partir de 50 mn
```

Validez votre programme avec la solution.

Solution C++ @[pgtelecom1.cpp]

```cpp
#include <iostream>
using namespace std;

int main()
{
  const double FIXE1 = 10.0;
  const double FIXE2 = 15.0;
  const double PRIX1 = 0.5;
  const double PRIX2 = 0.4;
  double consom;
  cout<<"Consommation moyenne (en mn)? ";
  cin>>consom;
  double tf1 = FIXE1 + PRIX1 * consom;
  double tf2 = FIXE2 + PRIX2 * consom;
  cout<<"Avec Telecom1 = "<<tf1<<" euros"<<endl;
  cout<<"Avec Telecom2 = "<<tf2<<" euros"<<endl;
  if (tf1 < tf2)
  {
    cout<<"==> Prenez Telecom1"<<endl;
  }
  else if (tf1 > tf2)
  {
    cout<<"==> Prenez Telecom2"<<endl;
  }
  else
  {
    cout<<"==> Prenez l'un ou l'autre"<<endl;
  }
  double mt = (FIXE2 - FIXE1) / (PRIX1 - PRIX2);
  cout<<"Prenez Telecom2 a partir de "<<mt<<" mn"<<endl;
}
```

3 Téléphone mobile / pgmobile (4 points)

Objectif

Vous êtes l'heureux propriétaire d'un téléphone mobile. Vous disposez d'un abonnement vous coûtant mensuellement `a` (= 40 €) qui vous donne droit à `g` (= 65) minutes gratuites de conversation locale. Dès que vous dépassez ces `g` minutes, le tarif de la communication locale est de `loc` (= 45) centimes/minute. Par ailleurs, si vous faites des communications internationales, elles vous seront tarifées à `inter` (= 100) centimes/minutes (dès la première minute).

(4 points) Écrivez un programme qui demande le nombre de minutes passées en conversations locales (dans `loc`) et le nombre de minutes passées en conversations internationales (dans `inter`), calcule le payant local `p` et le montant `m` à payer, puis affiche `m` ainsi que le message adéquat des minutes non consommées. Supposerez que l'utilisateur introduit des données correctes : il n'est donc pas nécessaire de faire des tests d'intégrité sur les données introduites.

Solution simple

On définit des constantes afin de représenter les valeurs littérales. Dans le cas d'une communication internationale, l'énoncé stipule qu'elle est tarifée dès la première première minute. Si on saisit des entiers, n signifiera donc que l'on consomme $n + 1$ minute en international.

Testez. Exemples d'exécution :

```
# min. appels locaux et inter? 64 10
Montant a payer = 51 Euros
Zut... 1 min. non consommee
```

```
# min. appels locaux et inter? 75 17
Montant a payer = 62.5 Euros
Comprend 10 min. en local
```

Validez votre programme avec la solution.

Solution C++ @[pgmobile1.cpp]

```cpp
#include <iostream>
using namespace std;

int main()
{
  const int TARIF_LOCAL = 45;
  const int TARIF_INTER = 100;
  const int FORFAIT = 65;
  const int ABONNEMENT = 40;
  // Saisit les données
```

```
  int loc, inter;
  cout<<"# min. appels locaux et inter? ";
  cin>>loc>>inter;
  // Calcule les résultats
  int p = loc - FORFAIT;
  double m1oc = (p > 0 ? p * TARIF_LOCAL : 0.0);
  double minter = (inter + 1) * TARIF_INTER;
  double m = ABONNEMENT + (m1oc + minter) / 100.0;
  // Affiche les résultats
  cout<<"Montant a payer = "<<m<<" Euros"<<endl;
  if (p < 0)
  {
    cout<<"Zut... "<<(-p)<<" min. non consommee"<<endl;
  }
  else if (p == 0)
  {
    cout<<"Pile forfait !"<<endl;
  }
  else
  {
    cout<<"Comprend "<<p<<" min. en local"<<endl;
  }
}
```

4 Le stationnement alternatif / pgalternatif (4 points)

Objectif

Dans une rue où se pratique le stationnement alternatif, on se gare :

- Du 1 au 15 du mois : du côté des maisons ayant un numéro impair.
- Et le reste du mois : de l'autre côté.

Sur la base du numéro de jour et du numéro de maison, indiquez si le stationnement est valide.

(0.5 point) Écrivez un programme qui saisit le numéro du jour dans `jr` (entier supposé compris entre 1 et 31). Affichez l'invite :

```
Numero du jour?
```

(0.5 point) Selon la valeur de `jr`, affichez de quel côté (impair ou pair) on doit se garer. Exemple : Pour `jr` valant 20 :

```
Garez-vous du cote pair
```

(0.5 point) Saisissez le numéro de maison devant laquelle vous vous êtes arrêté dans `num` (entier). Affichez l'invite :

```
Numero de maison?
```

(1 point) Finalement, indiquez dans un booléen `b` si vous êtes bien stationné (valeur `Vrai`) ou non (valeur `Faux`).

(0.5 point) Affichez (où `[x]` signifie le contenu de `x`) :

```
Stationnement correct : [b]
```

(Le C/C++ indiquera 1 pour `true` et 0 pour `false`.)

Testez. Exemples d'exécution :

```
Numero du jour? 20
Garez-vous du cote pair
Numero de maison? 67
Stationnement correct : 0
```

```
Numero du jour? 5
Garez-vous du cote impair
Numero de maison? 9
Stationnement correct : 1
```

Validez votre programme avec la solution.

Solution C++ @[pgalternatif1.cpp]

```cpp
#include <iostream>
using namespace std;

int main()
{
  int jr;
  cout<<"Numero du jour? ";
  cin>>jr;
  if (1 <= jr && jr <= 15)
  {
    cout<<"Garez-vous du cote impair"<<endl;
  }
  else
  {
    cout<<"Garez-vous du cote pair"<<endl;
  }
  int num;
  cout<<"Numero de maison? ";
  cin>>num;
  bool b = ((1 <= jr && jr <= 15 && num % 2 == 1) || (16 <= jr && jr <= 31 && num % 2 ==
    0));
  cout<<"Stationnement correct : "<<b<<endl;
}
```

5 Une minute plus tard / pgminute (4 points)

Objectif
Cet exercice affiche l'horaire (heure, minute) qu'il sera une minute plus tard.
(Une montre à affichage digital effectue un calcul semblable toutes les minutes.)

(0.5 point) Écrivez un programme qui saisit un horaire exprimé par deux entiers dans `hr` (heure) et `mn` (minute). Affichez l'invite :

```
Votre horaire (hr mn)?
```

(1.5 point) Calculez l'horaire (heure, minute) qu'il sera une minute plus tard, **dans ces mêmes entiers** `hr` et `mn`.

Aide simple
On peut envisager deux familles de solutions :

- Soit ajouter 1 au nombre des minutes puis changer l'heure si on arrive à 60.
- Soit traiter différemment le cas où le nombre des minutes est 59 (changement d'heure) de celui où il n'est pas 59 (pas de changement d'heure).

Dans les deux cas, on se méfiera de 23 heures et 59 minutes.

(2 points) Affichez l'horaire en tenant compte du cas spécial de minuit et des minutes nulles.

Testez. Exemples d'exécution :

```
Votre horaire (hr, mn)? 11 32
Une minute plus tard : 11h33'
```

```
Votre horaire (hr, mn)? 23 59
Une minute plus tard : Minuit
```

```
Votre horaire (hr, mn)? 3 59
Une minute plus tard : 4h
```

Validez votre programme avec la solution.

Solution C++ @[pgminute1.cpp]

```cpp
#include <iostream>
using namespace std;

int main()
{
  int hr, mn;
```

```
  cout<<"Votre horaire (hr, mn)? ";
  cin>>hr>>mn;
  ++mn;
  if (mn == 60)
  {
    mn = 0;
    ++hr;
    if (hr == 24)
    {
      hr = 0;
    }
  }
  cout<<"Une minute plus tard: ";
  if (hr == 0 && mn == 0)
  {
    cout<<"Minuit"<<endl;
  }
  else
  {
    cout<<hr<<"h";
    if (mn != 0)
    {
      cout<<mn<<"'";
    }
    cout<<endl;
  }
}
```

Solution C++ : Autre solution @[pgminute2.cpp]

```
#include <iostream>
using namespace std;

int main()
{
  int hr, mn;
  cout<<"Votre horaire (hr, mn)? ";
  cin>>hr>>mn;
  if (mn < 59)
  {
    ++mn;
  }
  else
  {
    mn = 0;
    if (hr < 23)
    {
      ++hr;
    }
    else
    {
      hr = 0;
    }
  }
  cout<<"Une minute plus tard: ";
  if (hr == 0 && mn == 0)
  {
```

```
    cout<<"Minuit"<<endl;
  }
  else
  {
    cout<<hr<<"h";
    if (mn != 0)
    {
      cout<<mn<<"'";
    }
    cout<<endl;
  }
}
```

6 L'horloge / pghorloge (4 points)

Objectif
Cet exercice fait avancer une horloge de 1 seconde. Cette horloge est représentée par un triplet de trois variables (heure, minute, seconde). Exemple : si l'horloge indique `15:12:59`, une seconde plus tard, elle devra indiquer `15:13:00`. On suppose que le passage au jour suivant remet l'horaire à `00:00:00`.

(0.5 point) Écrivez un programme qui saisit un triplet d'entiers dans `hr` (heure), `mn` (minute) et `ss` (seconde). Affichez l'invite :

```
Votre horloge (hr mn ss)?
```

(1.5 point) Dans ces mêmes entiers `hr`, `mn`, `ss` supposés valides,

Calculez l'horaire (heure, minute, seconde) qu'il sera une seconde plus tard.

Aide simple
On peut envisager deux familles de solutions :

- Soit ajouter 1 au nombre de secondes puis changer les minutes si on arrive à 60.
- Soit traiter différemment le cas où le nombre de secondes est 59 (changement de minutes) de celui où il n'est pas 59 (pas de changement de minutes).

(2 points) Affichez l'horaire en tenant compte du cas spécial de minuit et des minutes nulles.

Testez.

Validez votre programme avec la solution.

7 Références générales

Comprend [Dabancourt-PG1 :c2 :ex1], [Felea-PG1 :c3 :ex22] ■