Structures conditionnelles [if]
Exercices de cours

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 14 mai 2018

Table des matières

C - Exercices de cours (Solution)

Mots-Clés Structures conditionnelles ■
Difficulté • ○ ○ (2 h) ■

1 Appréhender le cours

1.1 Évaluation d'expressions logiques / qzlogiques

Objectif
Cet exercice évalue des expressions logiques.

Écrivez un programme qui saisit deux entiers dans `n1` et `n2`.
Affichez l'invite :

```
Deux entiers?
```

On saisit 15 et 4.
Évaluez les conditions simples (où <- désigne l'affectation) :

```
c1 <- n1 < n2
c2 <- 'A' < 'B'
c3 <- "Arb uste" < "Arbre"
```

Solution simple
(c1) `eval(15<4)` donne `Faux`.
(c2) `eval('A'<'B')` donne `Vrai` car l'ordre alphabétique est respecté dans les codes ASCII attribués aux lettres.
(c3) `eval("Arb uste"<"Arbre")` donne `Vrai` car l'espace est avant les lettres.

Quelle(s) est/sont la/les caractéristique(s) de l'évaluation des expressions logiques ?

Solution simple
Il y en a deux :

- Elle s'effectue de la gauche vers la droite.
- Il y a une évaluation paresseuse des opérateurs `Et` et `Ou`.

Évaluez les expressions logiques :

```
b1 <- c1 Et c2
b2 <- c1 Ou c2
b3 <- (c1 Et c2) Ou c3
b4 <- c1 Et (c2 Ou c3)
```

Solution simple
(b1) `c1` est `Faux` et `c2` est `Vrai` ce qui donne `Faux Et Vrai` donc `Faux`.
(b2) Ici on a `Faux Ou Vrai` donc `Vrai`.
(b3) Le groupe de conditions situé à gauche donne le résultat intermédiaire `Faux`, et `c3` est `Vrai`. Le résultat global est `Faux Ou Vrai` donc `Vrai`.
(b4) `c1` est `Faux` et le groupe de conditions qui suit donne `Vrai`. Le résultat global est `Faux Et Vrai` donc `Faux`.

Écrivez les expressions logiques de la question précédente dans votre langage.

Affichez les variables (où `[x]` désigne le contenu de `x`) :

```
c1=[c1] c2=[c2] c3=c3
b1=[b1] b2=[b2] b3=[b3] b4=[b4]
```

Testez.

Solution simple

```
Deux entiers? 15 4
c1=Faux c2=Vrai c3=Vrai
b1=Faux b2=Vrai b3=Vrai b4=Faux
```

(En C/C++, le `Vrai` est affiché 1 et le `Faux` en 0.)

Validez votre programme avec la solution.

Solution C @[qzlogiques1.c]

```c
#include <stdio.h>
#include <stdbool.h>
int main()
{
  int n1, n2;
  printf("Deux entiers? ");
  scanf("%d%d",&n1,&n2);
  bool c1 = n1 < n2;
  bool c2 = 'A' < 'B';
  bool c3 = "Arb uste" < "Arbre";
  bool b1 = c1 && c2;
  bool b2 = c1 || c2;
  bool b3 = ( c1 && c2 ) || c3;
  bool b4 = c1 && ( c2 || c3 );
  printf("c1=%d c2=%d c3=%d\n",c1,c2,c3);
  printf("b1=%d b2=%d b3=%d b4=%d\n",b1,b2,b3,b4);
}
```

1.2 Inverse d'un entier / pginverse

Écrivez un programme qui saisit un entier.
Affichez l'invite :

```
Votre entier?
```

En utilisant une structure `Si`, affichez :

- L'inverse de l'entier s'il n'est pas nul.
- Sinon le message « l'entier est nul ».

Rappel de cours C/C++
Attention à la division « entier/entier ».

Testez. Exemples d'exécution :

```
Un entier? 12
1/12 vaut 0.0833333
```

```
Un entier? 0
L'entier est nul
```

Validez votre programme avec la solution.

Solution C @[pginverse1.c]

```c
#include <stdio.h>
int main()
{
  int x;
  printf("Un entier? ");
  scanf("%d",&x);
  if (x != 0)
  {
    printf("1/%d vaut %g\n",x,1.0 / x);
  }
  else
  {
    printf("L'entier est nul\n");
  }
}
```

1.3 Valeur absolue d'un entier / pgvabsolue

Écrivez un programme qui saisit un entier.
Affichez l'invite :

```
Un entier?
```

Calculez puis affichez sa valeur absolue.

Testez. Exemples d'exécution :

```
Votre entier? 6
==> La valeur absolue est 6
```

```
Votre entier? -8
==> Inversion du signe
==> La valeur absolue est 8
```

Validez votre programme avec la solution.

Solution C @[pgvabsolue1.c]

```c
#include <stdio.h>
int main()
{
  int n;
  printf("Votre entier? ");
  scanf("%d",&n);
  if (n < 0)
  {
    n = -n;
    printf("==> Inversion du signe\n");
  }
  printf("==> La valeur absolue est %d\n",n);
}
```

Solution commentée

L'alternative `Si-Alors` évite de mettre du code pour rien. C'est le cas ici : si la valeur est positive, il n'y a rien à faire. Par contre si elle est négative, il faut inverser le signe pour la rendre positive. C'est ce qu'effectue l'algorithme après la saisie de la valeur entière.

2 Appliquer le cours

Méthode

Reprenons la méthode élémentaire d'élaboration d'algorithme vue dans le module @[Structures de base] et complétons-la en y ajoutant deux nouveaux points (N) :

- Faites attention au problème posé.....
- Résolvez les exercices...
- (N) Ayez à l'esprit le fonctionnement des nouvelles instructions étudiées.
- Transcrivez l'analyse en un programme.
- Soumettez votre programme à une ou deux traces d'exécution.
- (N) Vérifiez si votre programme permet de traiter correctement tous les cas permis par l'énoncé.

Vous vous apercevrez que le style de rédaction des énoncés des exercices change notamment sur trois points :

1. Les énoncés sont rédigés dans un style moins académique, plus proche du langage de tous les jours, que ceux des modules précédent. Par exemple, au lieu d'écrire « écrivez un programme qui demande un nombre représentant une température... » nous écrirons « écrivez un programme qui saisit une température.... ». Ou encore, au lieu de « écrivez un programme qui demande un nombre compris entre 0 et 20 représentant une note.... » nous écrirons plus volontiers « écrivez un programme qui saisit une note comprise entre 0 et 20.... », ou encore « écrivez un programme qui saisit une note.... ».
2. Ces énoncés ne mettent plus en avant les valeurs en entrée et les valeurs en sortie. Il faut le plus souvent les déduire de l'énoncé et de l'exemple d'exécution.
3. Ces énoncés sont plus imprécis. Cela correspond davantage à la manière dont les utilisateurs énoncent réellement leurs problèmes. C'est au concepteur de lever les ambiguïtés, soit par déduction logique, soit en se documentant, soit en interrogeant l'utilisateur. Nos imprécisions seront néanmoins limitées. Par exemple, s'il n'y a pas d'ambiguïté, les chaînes de caractères ne sont pas toujours entourées de guillemets.

2.1 Différence positive de deux entiers / pgpositive

Écrivez un programme qui saisit deux entiers dans `n1` et `n2`.
Affichez l'invite :

```
Deux entiers?
```

Calculez la différence **positive** des deux entiers dans `rs` (entier).

Aide méthodologique

Au choix :

- Si `n1<n2` alors c'est `n2-n1` sinon c'est `n1<n2`.
- Calculez la différence des deux entiers dans `rs` puis utilisez une structure `Si-Alors` pour positiver le résultat.

Affichez (où `[x]` désigne le contenu de `x`) :

```
La difference positive est [rs]
```

Testez. Exemples d'exécution :

```
Deux entiers? -6 3
La difference positive est 9
```

```
Deux entiers? 10 2
La difference positive est 8
```

Validez votre programme avec la solution.

Solution C : Première méthode @[pgpositive1.c]

```c
#include <stdio.h>
int main()
{
  int n1, n2;
  printf("Deux entiers? ");
  scanf("%d%d",&n1,&n2);
  int rs;
  if (n1 < n2)
  {
    rs = n2 - n1;
  }
  else
  {
    rs = n1 - n2;
  }
  printf("La difference positive est %d\n",rs);
}
```

Solution C : Deuxième méthode @[pgpositive2.c]

```c
#include <stdio.h>
int main()
{
  int n1, n2;
  printf("Deux entiers? ");
  scanf("%d%d",&n1,&n2);
  int rs =  n2 - n1;
  if (rs < 0)
```

```
  {
    rs = - rs;
  }
  printf("La difference positive est %d\n",rs);
}
```

2.2 Majorité d'un individu / pgmajorite

Écrivez un programme qui saisit l'âge d'un individu dans `age` (entier).
Affichez l'invite :

```
Votre age?
```

Définissez une constante `MAJORITE=18` (âge de la majorité).

En utilisant une structure `Si`, affichez :

- S'il est ou non majeur (âge supérieur ou égal à `MAJORITE`).
- Ainsi que le nombre d'années depuis ou dans pour être majeur.

Testez. Exemples d'exécution :

```
Votre age? 12
==> Vous n'êtes pas majeur
==> Majorite dans 6 annees
```

```
Votre age? 40
==> Vous êtes majeur
==> Majorite depuis 22 annees
```

Définissez maintenant les constantes :

- `RETRAITE=62` (âge minimum du départ à la retraite).
- `COTISEE=42` (nombre d'années de cotisation).

Saisissez la durée cotisée dans `duree` (entier).
Affichez l'invite :

```
Durée cotisation?
```

Affichez alors s'il peut partir à la retraite (âge supérieur ou égal à `RETRAITE` et durée cotisée supérieure ou égale à `COTISEE`). Dans le cas de la négative, affichez le nombre d'années restants.

Testez. Exemple d'exécution :

```
Votre age? 40
==> Vous êtes majeur
Majorite depuis 22 annees
Durée cotisation? 22
Vous ne pouvez pas encore partir à la retraite
Depart prevu dans 22 annees
```

Validez votre programme avec la solution.

Solution C @[pgmajorite1.c]

```c
#include <stdio.h>
const int MAJORITE = 18;
const int RETRAITE = 62;
const int COTISEE = 42;
int main()
{
  int age;
  printf("Votre age? ");
  scanf("%d",&age);
  if (age >= MAJORITE)
  {
    printf("==> Vous etes majeur\n");
    printf("Majorite depuis %d annees\n",(age-MAJORITE));
  }
  else
  {
    printf("==> Vous n'etes pas majeur\n");
    printf("Majorite dans %d annees\n",(MAJORITE-age));
  }
  int duree;
  printf("Duree cotisation? ");
  scanf("%d",&duree);
  if (age >= RETRAITE && duree >= COTISEE)
  {
    printf("Vous pouvez partir a la retraite\n");
  }
  else
  {
    printf("Vous ne pouvez pas encore partir a la retraite\n");
    printf("Depart prevu dans %d annees\n",(RETRAITE-age));
  }
}
```

2.3 Validation d'un module / pgvmodule

Objectif

Un module est sanctionné par une note d'oral de coefficient 1 et une note d'écrit de coefficient 2. La moyenne obtenue doit être supérieure ou égale à 10 pour valider le module. Cet exercice calcule le résultat (reçu, refusé) d'un étudiant à un module.

Écrivez un programme qui saisit :

- La note d'orale d'un étudiant dans `no` (réel).
- Sa note d'écrit dans `ne` (réel).

Affichez les invites :

```
Note d'écrit?
Note d'oral?
```

Supposez des notes positives valides entre 0 et 20.

Calculez la moyenne du module dans `moyenne` (réel) définie par :

$$moyenne = (2ne + no)/3$$

Affichez (où `[x]` désigne le contenu de `x`) :

```
==> Moyenne [moyenne]
```

Affichez le résultat (`recu`, `refus`) en comparant sa moyenne à 10 au moyen de l'alternative `Si`.

Testez. Exemples d'exécution :

```
Note d'écrit? 8.5
Note d'oral? 14
==> Moyenne 10.3333333333
==> reçu
```

```
Note d'écrit? 11
Note d'oral? 5
==> Moyenne 9
==> refusé
```

Validez votre programme avec la solution.

Solution C @[pgvmodule1.c]

```c
#include <stdio.h>
int main()
{
  double ne;
  printf("Note d'ecrit? ");
  scanf("%lf",&ne);
  double no;
  printf("Note d'oral? ");
  scanf("%lf",&no);
  double moyenne = (2*ne+no)/3;
  printf("==> Moyenne %g\n",moyenne);
  if (moyenne>=10)
  {
    printf("==> recu\n");
  }
  else
  {
    printf("==> refuse\n");
  }
}
```

2.4 Facturation avec remise / pgremise

Écrivez un programme qui saisit un prix hors taxes.

Calculez le prix TTC correspondant (avec un taux de TVA constant de 19.6%).

Établissez ensuite une remise dont le taux dépend de la valeur ainsi obtenue, à savoir :

- 0% pour un montant inférieur à 1 000€.
- 1% pour un montant supérieur ou égal à 1 000€ et inférieur à 2 000€.
- 3% pour un montant supérieur ou égal à 1 000€ et inférieur à 5 000€.
- 5% pour un montant supérieur ou égal à 5 000€.

Affichez le prix TTC ainsi que la remise.

Testez.

Validez votre programme avec la solution.

Solution C @[pgremise1.c]

```c
#include <stdio.h>
#define TAUX_TVA 19.6
int main(void)
{
   double ht, ttc, net, tauxr, remise;
   printf("Prix hors taxes? ");
   scanf("%lf",&ht);

   ttc = ht * (1.0 + TAUX_TVA/100.0);
   if (ttc < 1000.0)
   {
      tauxr = 0.0;
   }
   else if (ttc < 2000.0)
   {
      tauxr = 1.0;
   }
   else if (ttc < 5000.0)
   {
      tauxr = 3.0;
   }
   else
   {
      tauxr = 5.0;
   }
   remise = ttc * tauxr / 100.0;
   net = ttc - remise;

   printf("Prix ttc    %10.2lf\n",ttc);
```

```
    printf("Remise      %10.2lf\n",remise);
    printf("Net a payer %10.2lf\n",net);
    return 0;
}
```

3 Approfondir le cours

3.1 Sexe d'un individu / pgsexe

Écrivez un programme qui saisit un caractère parmi 'm', 'h', 'g', 'f' dans `sexe`.
Affichez l'invite :

```
Votre sexe parmi m,h,g,f?
```

Affichez le sexe de l'individu selon :

- « Masculin » si `'m'` (Masculin), `'h'` (Homme) ou `'g'` (Garçon)
- « Féminin » si `'f'` (Féminin, Femme, Fille)
- « Saisie incorrecte » dans tous les autres cas.

Testez. Exemples d'exécution :

```
Votre sexe parmi m,h,g,f? h
Vous êtes du sexe Masculin
```

```
Votre sexe parmi m,h,g,f? x
Saisie incorrecte
```

Validez votre programme avec la solution.

Solution C : Structure Selon] @[pgsexe1.c]

```c
#include <stdio.h>
int main()
{
  char sexe;
  printf("Votre sexe parmi m,h,g,f? ");
  scanf("%c",&sexe);
  switch (sexe)
  {
    case 'm':
    case 'h':
    case 'g':
      printf("Vous etes du sexe Masculin\n");
      break;
    case 'f':
      printf("Vous etes du sexe Feminin\n");
      break;
    default:
      printf("Saisie incorrecte\n");
  }
}
```

Solution C : Structure Si @[pgsexe2.c]

```c
#include <stdio.h>
int main()
{
  char sexe;
  printf("Votre sexe parmi m,h,g,f? ");
  scanf("%c",&sexe);
  if (sexe == 'm' || sexe == 'h' || sexe == 'g')
  {
      printf("Vous etes du sexe Masculin\n");
  }
  else if (sexe == 'f')
  {
      printf("Vous etes du sexe Feminin\n");
  }
  else
  {
      printf("Saisie incorrecte\n");
  }
}
```

3.2 Calculette algébrique / pgcalcul

Écrivez un programme qui saisit :

- Un entier dans `n1`.
- Un caractère dans `op`.
- Un deuxième entier dans `n2`.

Affichez l'invite :

```
n1 @ n2?
```

Déclarez un entier `rs` qui mémorisera le calcul de l'opération correspondante.

Déclarez un booléen `ok` et initialisez-le à `Vrai` :
il vaudra `Vrai` si le calcul a été effectué, `Faux` sinon.

En utilisant une structure conditionnelle (`Selon` ou `Si`) :

- Traitez les cas où `op` est l'un des opérateurs `'+'` (addition), `'-'` (soustraction), `'*'` ou `'x'` (multiplication) en mémorisant le résultat du calcul dans `rs`.
- Traitez les cas où `op` est l'opérateur `'/'` (division) ou `'%'` (modulo). Dans ces deux cas, l'opération est réalisable si `n2` n'est pas nul. Mettez `Faux` dans `ok` si elle n'a pas été effectuée.
- Enfin dans tous les autres cas, l'opération n'est pas valide :
  mettez `Faux` dans `ok`.

Affichez l'opération effectuée si `ok` est resté à `Vrai`, le message d'erreur sinon.

```
[n1] [op] [n2] donne [rs] # si ok
Opérateur erroné ou division par zéro # message d'erreur
```

Testez. Exemples d'exécution :

```
n1 @ n2? 3 x -3
3 x -3 donne -9
```

```
n1 @ n2? 6 / 0
Opérateur erroné ou division par zéro
```

Validez votre programme avec la solution.

Solution C @[pgcalcul1.c]

```c
#include <stdio.h>
#include <stdbool.h>
int main()
{
  int n1, n2;
  char op;
  printf("n1@n2? ");
  scanf("%d%c%d",&n1,&op,&n2);
  int rs = 0;
  bool ok = true;
  switch (op)
  {
    case '+':
      rs = n1 + n2;
      break;
    case '-':
      rs = n1 - n2;
      break;
    case '*':
    case 'x':
      rs = n1 * n2;
      break;
    case '/':
      if (n2 != 0)
      {
        rs = n1 / n2;
      }
      else
      {
        ok = false;
      }
      break;
    case '%':
      if (n2 != 0)
      {
        rs = n1 % n2;
      }
      else
      {
        ok = false;
      }
      break;
    default:
      ok = false;
  }
  if (ok)
  {
    printf("%d %c %d donne %d\n",n1,op,n2,rs);
  }
  else
  {
    printf("Operateur errone ou division par zero\n");
  }
}
```

3.3 Bonjour / pgbonjour

Écrivez un programme qui saisit une heure dans `hr` (entier).
Affichez l'invite :

```
Quelle heure est-il?
```

Affichez la salutation selon :

- « Bonjour » si `hr` est dans [0..18[
- « Bonsoir » si `hr` est dans [18..22[
- « Bonne nuit » si `hr` est dans [22..24[
- « Heure invalide » dans tous les autres cas

Testez. Exemples d'exécution :

```
Quelle heure est-il? 19
Bonsoir
```

```
Quelle heure est-il? 25
Heure invalide
```

Validez votre programme avec la solution.

Solution C : Première solution @[pgbonjour0.c]

```c
#include <stdio.h>
int main()
{
  int hr;
  printf("Quelle heure est-il? ");
  scanf("%d",&hr);
  if (0 <= hr && hr < 18)
  {
    printf("Bonjour\n");
  }
  else if (18 <= hr && hr < 22)
  {
    printf("Bonsoir\n");
  }
  else if (22 <= hr && hr < 24)
  {
    printf("Bonne nuit\n");
  }
  else
  {
    printf("Heure invalide\n");
  }
}
```

Solution C : Solution améliorée @[pgbonjour1.c]

```c
#include <stdio.h>
int main()
{
  int hr;
  printf("Quelle heure est-il? ");
  scanf("%d",&hr);
  if (hr < 0 || 24 <= hr)
  {
    printf("Heure invalide\n");
  }
  else if (hr < 18)
  {
    printf("Bonjour\n");
  }
  else if (hr < 22)
  {
    printf("Bonsoir\n");
  }
  else
  {
    printf("Bonne nuit\n");
  }
}
```

Solution commentée

- Première solution : Notez qu'il faut effectivement vérifier que l'heure est dans l'intervalle. En n'écrivant que la deuxième partie de la condition, pour une heure négative (donc invalide), on afficherait :

  ```
  Bonsoir
  ```

- Solution améliorée : Cette version est une bonne solution car elle effectue le minimum de tests. Chaque condition est simple, outre la première qui vérifie que l'heure est correcte, contrairement à la version naïve qui emploie trois conditions composées.