AP1 TP

4

Semaine 4 : Introduction aux structures de contrôle - 2/2

Exercice 1 : Portées des variables

1. Copiez chez vous le programme `/net/Bibliotheque/AP1/TPSem4/porteeIdent.cc`.
2. Lisez le code source et devinez l'affichage produit par l'exécution.
3. Exécutez ensuite ce programme.
4. Comparez avec votre intuition. Que remarquez-vous sur la variable `i` ? Expliquez.

Exercice 2 : Dicdacticiel de calcul

Avant de commencer, nous allons introduire : la structure de contrôle conditionnelle `switch`.

Structure de contrôle conditionnelle - switch

```
switch ( <variable> )
  {
    case <valeur_1> :
      <instruction_1_1>
      <instruction_1_2>
      ...
      break;
    case <valeur_2> :
      <instruction_2_1>
      <instruction_2_2>
      ...
      break;
    ...
    default :
      <instruction_d_1>
      <instruction_d_2>
  }
```

Exemple :

```
1  int n;
2  cin >> n;
3  switch ( n )
4    {
5      case 1 :
6        cout << "le jeu commence !" << endl;
7        jouer();
8        break;
9      case 2 :
10       solution();
11       break;
12     default :
```

```
13        cout << "entree incorrecte" << endl;
14    }
```

Ce didacticiel doit permettre de tester les connaissances d'élèves de CM1 en calcul mental. Le programme que vous allez réaliser propose à l'élève de résoudre des opérations et évalue les réponses.

1. **Menu : affichage et gestion des choix**. Écrire en C++ un programme qui affiche à l'écran le menu suivant :

   ```
   1. Exercice sur l'addition.
   2. Exercice sur la soustraction.
   3. Exercice sur la multiplication.

                  Votre Choix :
   ```

   Selon le choix de l'élève le programme affiche un message indiquant l'opération choisie.
   Pour répondre à cette question, vous écrirez l'action `menu` qui affiche les lignes du menu. Le prototype de l'action `menu` est la suivante :

   ```
   void menu()
   ```

2. **Mise en œuvre des opérations**. Étendez le programme précédent en demandant à l'élève d'entrer les opérandes des opérations. Selon chaque cas, le résultat de l'opération sera affiché.

Tests	Résultat(s) attendu(s)	Résultat(s) observé(s)
1		
2		
3		
0		

3. **Génération des opérandes**. Nous voulons créer un didactiel : les opérandes doivent être générés par le programme et non entrés par l'élève. Modifiez le programme pour que les opérandes soient générés de manière aléatoire par le programme. Pour cela vous écrirez une fonction `genOperande` dont voici le prototype :

   ```
   int genOperande(int max)
   ```

   La fonction `genOperande` retourne un entier compris entre 0 et `max` ; cet entier est généré de manière aléatoire.
   Génération aléatoire des opérandes. On utilisera le générateur de nombres aléatoires du C++ : `rand()`. La fonction `rand()` renvoie un entier aléatoire entre 0 et `RAND_MAX`. Pour générer un entier aléatoire `a` entre 0 et 10, il faut donc utiliser l'instruction : `a = (int) (10.0 * rand() / RAND_MAX)`. La fonction `rand()` est définie dans la bibliothèque `cstdlib` que vous devez inclure dans votre programme : `#include <cstdlib>` au début du fichier source.
   Le fichier `/net/exemples/AP1/TPSem4/genalea.cc` est un exemple de programme générant des nombres aléatoires.

   (a) Testez *plusieurs fois* le programme. Que remarquez-vous ?
   (b) Décommentez la ligne `srand(time(NULL))`. Que remarquez-vous ?

4. **Didactitiel**. Ce programme étant un didactitiel, le résultat calculé dans le programme ne doit pas être affiché ! Après génération aléatoire des opérandes et affichage de l'opération, **c'est l'élève qui propose un résultat**. Le programme vérifie alors le résultat proposé et donne un commentaire du type $Bravo$ ou $Perdu$.
   Modifiez le programme en conséquence.
5. **Tests et commentaires**. C'est fini ? Pas tout à fait ! Vérifiez sur plusieurs tests bien choisis que votre programme fonctionne et commentez le code pour faire apparaître les différentes étapes.

Exercice 3 : Suite de Syracuse

On considère la suite entière définie, pour une valeur u_1 positive fixée, par :

$$u_n = \begin{cases} 3u_{n-1} + 1 & \text{si } u_{n-1} \text{ est impair} \\ u_{n-1}/2 & \text{sinon} \end{cases}$$

On conjecture que, quel que soit $u_1 > 0$, cette suite converge vers le cycle $(1; 4; 2)$. On appelle *vol* associé à une valeur donnée de u_1 la liste des valeurs prises par la suite entre u_1 et u_k, où k est le plus petit entier tel que $u_k = 1$.

1. Récupérez, compilez et exécutez le programme `syracuse.cc`.
   Il se trouve dans le répertoire `/net/exemples/AP1/TPSem4`.
2. Modifiez le programme précédent pour qu'il affiche la longueur du vol, c'est-à-dire le premier k tel que $u_k = 1$.

Tests	Résultat(s) attendu(s)	Résultat(s) observé(s)
U_1=1 ;k=8		
U_1=3 ;k=8		
U_1=-10 ;k=8		

3. Modifiez le programme précédent pour qu'il affiche aussi l'altitude maximale, c'est-à-dire le plus grand élément du vol.

Tests	Résultat(s) attendu(s)	Résultat(s) observé(s)
U_1=1 ;k=8		
U_1=3 ;k=8		
U_1=-10 ;k=8		