Algorithmique
Correction du DS1

IUT Informatique de Lens, 1ère Année
Université d'Artois

Frédéric Koriche
koriche@cril.fr
2011 - Semestre 1

Exercice 1

Un service de reprographie facture :

- 0.10 € par photocopie si vous faites moins de 10 photocopies
- 0.08 € par photocopie si vous faites entre 10 et 100 photocopies
- 0.05 € par photocopie si vous faites plus de 100 photocopies

Ecrire un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées, et qui affiche la facture correspondante.

Correction de l'exercice 1

nombreDePhotocopies

```
variables
    // Nombre de photocopies
    entier nbPhotocopies

début
    afficher "Nombre de photocopies : "
    lire nbPhotocopies
    afficher "Prix total : "
    si nbPhotocopies < 10 alors
        // Moins de 10 photocopies
        afficher 0.10 × nbPhotocopies
    sinon
        si nbPhotocopies ≤ 100 alors
            // Au moins 10 photocopies, et au plus 100 photocopies
            afficher 0.08 × nbPhotocopies
        sinon
            // Plus de 100 photocopies
            afficher 0.05 × nbPhotocopies
        fin
    fin
fin
```

Exercice 2

Monsieur Boulier, responsable de la comptabilité d'un service de vente en-ligne, souhaite construire un programme simulant une caisse automatique : le programme lit le prix hors taxe (HT) des articles du consommateur, ajoute à chaque article la TVA à 5.5 %, et retourne la somme totale (TTC) à payer lorsque l'utilisateur entre la valeur 0. On supposera que les prix saisis sont positifs. Corriger les erreurs du code C++ ci-dessous.

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

int main()
{
float prix_ht, prix_ttc;

do
   {
   cout << "Saisir le prix (HT) de l'article : ";
   cin >> prix_ht;
   prix_ttc = prix_ht + 5.5;
   somme = somme + prix_ttc;
   }
while(prix > 0)

cout >> "La somme totale est : " << somme << endl;
return 0;
}
```

caisseAutomatique

Correction de l'exercice 2

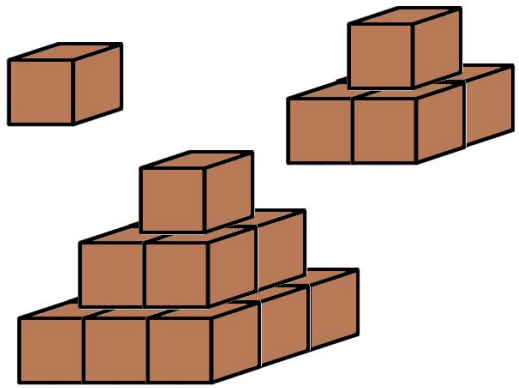
```
#include <iostream>
using namespace std;

int main()
{
float prix_ht, prix_ttc, somme;

somme = 0;
do
    {
    cout << "Saisir le prix (HT) de l'article : ";
    cin >> prix_ht;
    prix_ttc = 1.055 * prix_ht;
    somme = somme + prix_ttc;
    }
while(prix_ht> 0);

cout <<"La somme totale est : " << somme << endl;
return 0;
}
```

caisseAutomatique

Exercice 3

Numérobis, architecte égyptien, souhaite résoudre le problème suivant :

- Données : un nombre n de pierres taillées disponibles
- Résultat : la hauteur de la plus grande pyramide pouvant être construite avec ces pierres



Comme l'indique la figure ci-dessus, les pyramides égyptiennes sont stratifiées :

- une pyramide de hauteur 1 compte une pierre (une seule strate),
- une pyramide de hauteur 2 compte 5 pierres : 1 pour la première strate et 4 pour la seconde,
- une pyramide de hauteur 3 compte 14 pierres : 1 pour la première strate, 4 pour la seconde, 9 pour la troisième,

et ainsi de suite. Afin de résoudre le problème, remplacez dans l'algorithme ci-dessous les points d'interrogation (**?**) par les bonnes expressions.

Algorithme 2: hauteurDePyramide

```
variables
    // hauteur de la pyramide
    entier hauteur
    // nombre de pierres disponibles
    entier nbPierresDisponibles
    // nombre de pierres par strate
    entier nbPierresParStrate
    // nombre de pierres utilisées dans la pyramide
    entier nbPierresUtilisées

début
    afficher "Entrez le nombre de pierres disponibles : "
    lire ?
    hauteur ← ?
    nbPierresUtilisées ← ?
    répéter
        nbPierresParStrate ← ?
        nbPierresUtilisées ← nbPierresUtilisées + ?
        si nbPierresUtilisées ≤ nbPierresDisponibles alors
            hauteur ← hauteur + ?
        fin
    jusqu'à ?
    afficher "La hauteur maximale est : ", ?
fin
```

Algorithme 3: hauteurDePyramide (une solution possible)

```
variables
    // hauteur de la pyramide
    entier hauteur
    // nombre de pierres disponibles
    entier nbPierresDisponibles
    // nombre de pierres par strate
    entier nbPierresParStrate
    // nombre de pierres utilisées dans la pyramide
    entier nbPierresUtilisées

début
    afficher "Entrez le nombre de pierres disponibles : "
    lire nbPierresDisponibles
    hauteur ← 0
    nbPierresUtilisées ← 0
    répéter
        nbPierresParStrate ← hauteur × hauteur
        nbPierresUtilisées ← nbPierresUtilisées + nbPierresParStrate
        si nbPierresUtilisées ≤ nbPierresDisponibles alors
            hauteur ← hauteur + 1
        fin

    jusqu'à nbPierresUtilisées > nbPierresDisponibles
    afficher "La hauteur maximale est : ", hauteur - 1
fin
```

Algorithme 4: hauteurDePyramide (une autre solution possible)

```
variables
    // hauteur de la pyramide
    entier hauteur
    // nombre de pierres disponibles
    entier nbPierresDisponibles
    // nombre de pierres par strate
    entier nbPierresParStrate
    // nombre de pierres utilisées dans la pyramide
    entier nbPierresUtilisées

début
    afficher "Entrez le nombre de pierres disponibles : "
    lire nbPierresDisponibles
    hauteur ← 0
    nbPierresUtilisées ← 0
    répéter
        nbPierresParStrate ← (hauteur + 1) × (hauteur + 1)
        nbPierresUtilisées ← nbPierresUtilisées + nbPierresParStrate
        si nbPierresUtilisées ≤ nbPierresDisponibles alors
            hauteur ← hauteur + 1
        fin

    jusqu'à nbPierresUtilisées ≥ nbPierresDisponibles
    afficher "La hauteur maximale est : ", hauteur
fin
```