Cours MOOC EPFL d'introduction à la programmation illustré en C++

Troisième devoir noté
Boucles et itérations

V. Lepetit, J.-C. Chappelier & J. Sam

du 11 octobre au 28 octobre 2013

1 Exercice 1 – jeux de nombres

1.1 Description

Considérons les opérations suivantes applicables à un nombre entier :
– si ce nombre est divisible par 3, on lui ajoute 4 ;
– s'il n'est pas divisible par 3 mais divisible par 4, on le divise par 2 ;
– s'il n'est divisible ni par 3, ni par 4, on lui soustrait 1.
On répète ces opérations successivement jusqu'à arriver à 0. Concrètement, partant d'un entier n_0, on applique les opérations à n_0 pour obtenir n_1, puis si n_1 n'est pas nul, on lui applique à nouveau les opérations précédentes, et ainsi de suite jusqu'à obtenir un nombre n_k valant 0.

Par exemple, si on part de 7, on trouve successivement les valeurs : 6, 10, 9, 13, 12, 16, 8, 4, 2, 1 et 0. Le nombre k de répétitions des opérations décrites ci-dessus est alors 11. Par contre, si on part de 1, on tombe tout de suite sur 0, et le nombre de répétitions est alors de $k = 1$.

On vous demande d'écrire un programme qui affiche le nombre de répétitions des opérations précédentes nécessaires pour tomber à 0, en partant tour à tour de chacun des entiers compris entre deux entiers saisis au clavier.

SUITE AU DOS ⟶

Par exemple si l'on demande ce nombre de répétitions pour chaque entier entre 1 et 7, l'affichage produit par votre programme devra être :

```
1 -> 1
2 -> 2
3 -> 12
4 -> 3
5 -> 4
6 -> 10
7 -> 11
```

Ici, 1 et 7 (dans la 1re colonne) sont les bornes entrées au clavier, et la seconde colonne correspond au nombre de répétitions nécessaires pour arriver à 0 en partant de chacun des nombres de la 1re colonne, par exemple 11 répétitions pour 7.

Si l'on demande ce nombre de répétitions pour les entiers entre 99 et 100, l'affichage du programme devra être :

```
99 -> 18
100 -> 17
```

Note : pour tester si un nombre `n` est divisible par `p`, il suffit de tester si `n % p` vaut zéro.

1.2 Marche à suivre

Télécharger le programme `suite.cc` fourni [1] et le compléter.

ATTENTION : vous ne devez modifier ni le début ni la fin du programme, juste ajouter vos propres lignes à l'endroit indiqué. Il est donc primordial de respecter la procédure suivante :

1. sauvegarder le fichier téléchargé sous le nom `suite.cc` ou `suite.cpp` ;
2. écrire le code à fournir entre ces deux commentaires :

```
/*****************************************
 * Completez le programme a partir d'ici.
 *****************************************/

/*****************************************
 * Ne rien modifier apres cette ligne.
 *****************************************/
```

1. Adresse : `https://d396qusza40orc.cloudfront.net/intro-cpp-fr/assignments-data/suite.cc`

3. sauvegarder et tester son programme pour être sûr(e) qu'il fonctionne correctement, par exemple avec les valeurs données plus haut ;
4. rendre le fichier modifié (toujours `suite.cc` ou `suite.cpp`) dans « OUTPUT submission » (et non pas dans «Additional»).

2 Exercice 2 – saut en parachute

On s'intéresse ici à écrire un programme permettant de calculer les paramètres de la chute d'un parachutiste.

Télécharger le programme `parachutiste.cc` fourni [2] et le compléter.

ATTENTION : vous ne devez modifier ni le début ni la fin du programme, juste ajouter vos propres lignes à l'endroit indiqué. Il est donc primordial de respecter la procédure suivante :

1. sauvegarder le fichier téléchargé sous le nom `parachutiste.cc` ou `parachutiste.cpp` ;
2. écrire le code à fournir (voir ci-dessous) entre ces deux commentaires :

```
/*****************************************
 * Completez le programme a partir d'ici.
 *****************************************/

/*****************************************
 * Ne rien modifier apres cette ligne.
 *****************************************/
```

3. sauvegarder et tester son programme pour être sûr(e) qu'il fonctionne correctement, par exemple avec les valeurs données plus bas ;
4. rendre le fichier modifié (toujours `parachutiste.cc` ou `parachutiste.cpp`) dans « OUTPUT submission » (et non pas dans «Additional»).

2.1 Modélisation du parachutiste

Le parachutiste sera représenté dans le programme par sa masse, sa vitesse de chute, son accélération, son altitude et la surface de son corps exposée aux frottements de l'air (celle-ci variera lorsque le parachute s'ouvrira).

Dans le programme téléchargé :

2. Adresse : `https://d396qusza40orc.cloudfront.net/intro-cpp-fr/assignments-data/parachutiste.cc`

1. commencez par déclarer dans le `main` une constante `g` de valeur 9.81, et deux variables de type `double` (qui seront modifiées lors de l'ouverture du parachute) : `v0`, initialisée à 0 et `t0`, initialisée à 0.
2. définissez ensuite les variables nécessaires à la description du parachutiste telle que donnée ci-dessus, autre que sa masse que nous avons déjà définie (voir le début du `main` dans le fichier téléchargé).

   On initialisera la surface du parachutiste à 2.0 m^2, son altitude avec la valeur de `h0`, sa vitesse avec celle de `v0` et son accélération avec celle de `g`.

   Définissez enfin une variable `t` initialisée à la valeur de `t0`.
3. Affichez les valeurs initiales telles que définies ci-dessus en respectant *strictement* le format suivant :

   temps, altitude, vitesse, accélération

   Avec les valeurs initiales données ci-dessus, une masse de 80 kg et une altitude de départ de 39'000 m, le programme affichera à ce stade :

   ```
   0, 39000, 0, 9.81
   ```

2.2 Chute libre

Pour calculer l'évolution du sportif en chute libre nous aurons besoin des deux expressions suivantes :
– s qui est la surface du sportif divisée par sa masse ;
– un « terme » noté q et valant $q = \exp(-s \times (t - t_0))$, où t représente le temps courant et t_0 le temps initial de la chute, initialisé à 0 dans la question précédente.

Note : la fonction exp s'écrit simplement `exp` en C++, par exemple : `exp(x)`.
L'évolution du sportif en chute libre s'exprime alors suivant :

$$v(t) = \frac{g}{s} \times (1 - q) + v_0 \times q$$
$$h(t) = h_0 - \frac{g}{s} \times (t - t_0) - \frac{v_0 - g/s}{s} \times (1 - q)$$
$$a(t) = g - s \times v(t)$$

où v est la vitesse du sportif, h son altitude, a son accélération, $g = 9.81$, et v_0, h_0 et t_0 correspondent aux trois variables définies à la question précédente [3].

On vous demande de compléter votre programme précédent de sorte à calculer l'évolution de la chute du sportif tel qu'initialisé dans la question précédente :

3. Elles pourront changer par la suite, donc même si certaines sont pour le moment nulles, il est important de toutes les garder explicitement dans les formules calculées.

faites le calcul, de seconde en seconde (c'est-à-dire ajouter à chaque fois `1` au temps `t`), tant que le sportif n'atteint pas le sol, c'est-à-dire tant que son altitude `h` est positive.

Affichez les caractéristiques du sportif à chaque seconde en respectant le format de la question précédente.

Testez votre programme avec une masse de 80 kg et une altitude de départ de 39'000 m ; il devrait donner les résultats suivants :

```
0, 39000, 0, 9.81
1, 38995.1, 9.68839, 9.56779
2, 38980.7, 19.1376, 9.33156
3, 38956.9, 28.3535, 9.10116
...
137, 426.306, 379.628, 0.319308
138, 46.5205, 379.943, 0.311425
```

2.3 Vitesse du son et vitesse limite

On vous demande maintenant d'étendre votre programme précédent de sorte que :

5. dès que la vitesse du sportif dépasse la vitesse du son (343 m/s), le programme affiche (en plus, mais qu'*une seule fois*) le message suivant :
   `## Felix depasse la vitesse du son`
   Ce message doit s'afficher *AVANT* les informations de temps, altitude, vitesse et accélération :
   ```
   ...
   82, 20498.6, 341.884, 1.26289
   ## Felix depasse la vitesse du son
   83, 20156.1, 343.132, 1.23171
   ...
   ```
6. dès que son accélération est inférieure à 0.5 m/s^2, le programme affiche (en plus, mais qu'*une seule fois*) le message suivant :
   `## Felix a atteint sa vitesse maximale`
   Ce message doit s'afficher *AVANT* les informations de temps, altitude, vitesse et accélération :
   ```
   ...
   119, 7199.16, 372.369, 0.500775
   ## Felix a atteint sa vitesse maximale
   120, 6826.54, 372.864, 0.488411
   ...
   ```

Pour tester : avec les valeurs précédentes (80 kg et 39'000 m), la vitesse du son est atteinte au bout de 83 s et la vitesse maximale ($\simeq$ 372 m/s) au bout de 120 s comme montré dans les deux exemples ci-dessus.

2.4 Ouverture du parachute

On vous demande finalement d'étendre une dernière fois votre programme précédent de sorte que dès que l'altitude du sportif est plus petite que 2500 m, le programme change la valeur de la surface du sportif de 2.0 m^2 (avant l'ouverture du parachute) à 25.0 m^2 (après l'ouverture du parachute). Il faut aussi changer les « conditions initiales » `t0`, `v0` et `h0` avec les valeurs actuelles du sportif (de sorte que les équations d'évolution soient correctes pour la suite de la chute).

De plus, le programme doit afficher le message suivant :

```
## Felix ouvre son parachute
```

Ce message doit s'afficher *AVANT* les informations de temps, altitude, vitesse et accélération :

```
...
131, 2698.03, 377.561, 0.370983
## Felix ouvre son parachute
132, 2320.28, 377.927, 0.361824
133, 1991.28, 284.922, -79.2283
...
```

Notez que donc l'accélération devient négative *deux* lignes après l'affichage de ce message.

Pour tester : avec les valeurs précédentes (80 kg et 39'000 m), le parachute est ouvert au bout de 132 s et la simulation se termine au bout de 170 s :

```
...
131, 2698.03, 377.561, 0.370983
## Felix ouvre son parachute
132, 2320.28, 377.927, 0.361824
133, 1991.28, 284.922, -79.2283
...
170, 18.4814, 31.3944, -0.000753963
```