**INFO1101 Test2 Le 4 décembre 2018**

**13h30 à 15h30**

**Durée 2 heures**

Le lien pour votre répertoire de test est :

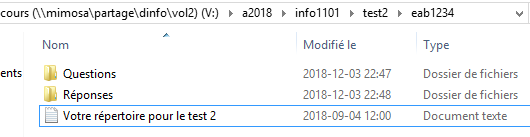
[\\MIMOSA.cm.umoncton.ca\PARTAGE\DINFO\VOL2\cours\a2018\info1101\test2\eab1234](file:///\\MIMOSA.cm.umoncton.ca\PARTAGE\DINFO\VOL2\cours\a2018\info1101\test2\eab1234)

**Le lien pour la pratique du 26 novembre 2019 :**

[\\MIMOSA.cm.umoncton.ca\PARTAGE\DINFO\VOL2\cours\a2019\info1101\pratique\eab1234](\\\\MIMOSA.cm.umoncton.ca\\PARTAGE\\DINFO\\VOL2\\cours\\a2019\\info1101\\pratique\\eab1234)

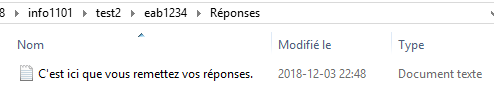
où eab1234 est remplacé par votre compte.

Dans ce répertoire, vous trouverez le répertoire de Questions et celui pour les Réponses.

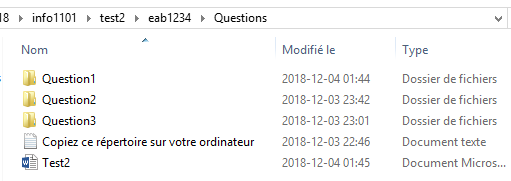


Je vous demande de copier le répertoire Questions sur votre ordinateur. Si jamais vous gâchez un fichier, vous pourrez toujours en faire une nouvelle copie grâce à l’original qui est demeuré sur le serveur mimosa.

Le répertoire Réponses accueillera vos réponses au test. Au début, ce répertoire sera presque vide. Afin d’éviter un mauvais transfert de fichiers, je vous recommande de copier vos trois projets au complet dans ce répertoire.

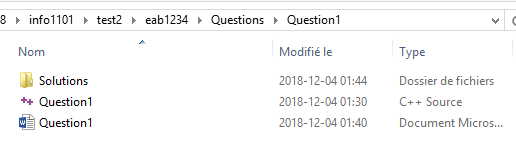


Le répertoire Questions contient trois sous-répertoires, un pour chacune des questions.



Le fichier Word Test2.docx est essentiellement ce texte-ci plus les textes des trois questions. Vous aurez une copie papier de ce document pour que vous n’ayez pas besoin de l’avoir ouvert à l’écran.

Le répertoire Question1 contient deux fichiers et un sous-répertoire.



Le contenu de Question1.docx vous sera donné sous format imprimé.

Question1.cpp est le fichier C++ dans lequel j’ai introduit des erreurs dans trois lignes de code. Vous aurez à trouver, expliquer et corriger ces trois lignes de code. Voici un exemple de ce à quoi devrait ressembler vos réponses.

Voici comment la correction doit être démontrée :

// ERREUR : cout >> variable >> endl ;

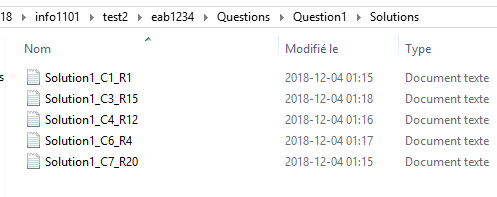
// EXPLICATION : Il y a erreur car on a utilisé des opérateurs

// d’extraction de flux au lieu d’opérateur d’insertion de flux.

cout << variable << endl ; // CORRECTION

Le fait qu’il y avait deux opérateurs incorrects ne comptent que pour une erreur.

Le sous-répertoire Solutions montrent des exemples de sorties produit par le programme s’il ne contenait pas d’erreurs.



Des explications sont données dans l’énoncé de la question.

Le répertoire Question2 est bâti sur le même plan :

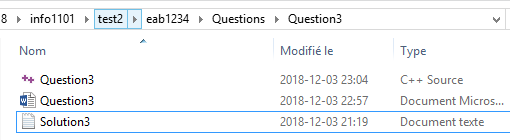


Question2.docx contient l’énoncé de la question, Question2.cpp contient le code à modifier et le sous -répertoire Solutions montrent des exemples de sorties.

Dans ce problème, vous aurez à produire des cas tests pour tester une fonction spécifique.

Les détails vous seront donnés dans l’énoncé de la question 2.

Le répertoire Question3 déroge un peu au plan en ce qu’il n’y a qu’un seul fichier d’exemple de sortie.



Question3.docx contient l’énoncé de la question, Question3.cpp contient le code C++ à compléter et Solution3.txt montre ce que le programme devrait produit si vous complétez correctement le code de la fonction spécifiée. Les détails de ce que la fonction fait sont donnés dans l’énoncé du problème.

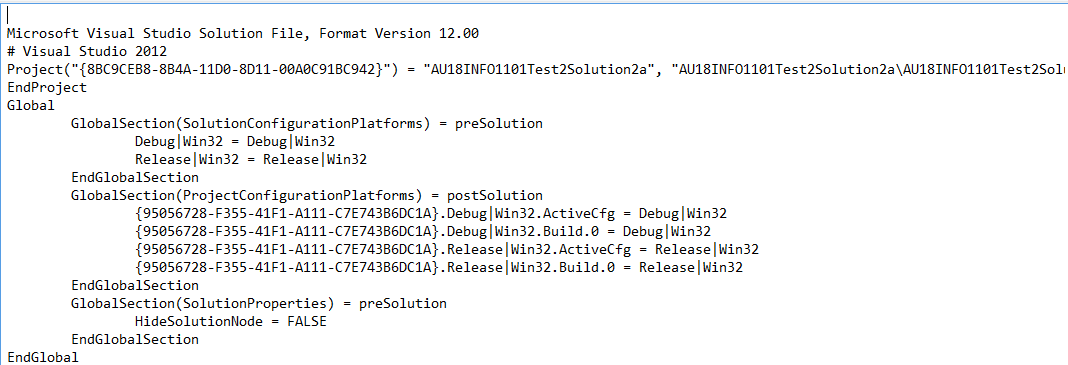
En comparant la sortie de votre programme et le contenu du fichier Solution3.txt, vous pourrez vérifier si vous êtes sur la bonne voie.

Dès que vous avez terminé une question, je vous encourage à copier votre répertoire de projet dans votre répertoire Réponses sur mimosa. N’attendez pas à la fin pour le faire.

Sous le coup du stress des dernières minutes, les gens font parfois des erreurs en remettant les mauvais fichiers. Par exemple, par le passé, certains m’ont remis des fichier .sln :



Voici ce à quoi ressemble le contenu d’un fichier .sln :



Ce n’est pas du C++.

Ou encore des fichiers .vcxproj.



Voici un exemple de contenu de ce genre de fichiers :



Encore une fois, ce n’est pas du C++.

Donc, SVP, faire attention. Remettez-moi le répertoire complet. Comme ça, les chances de ne pas remettre les bons fichiers sont beaucoup moindres.

**Consignes :**

Arrivez avant 13h30 pour pouvoir démarrer votre ordinateur (vous savez que parfois ça prend du temps) puis démarrer l’environnement Visual Studio.

Vous devriez être une personne par table, assis sur la chaise la plus près du mur.

Je ne permettrai deux personnes par table si toutes les tables sont occupées et seulement sur les tables à quatre chaises. Je choisirai sur quelle table ces « deuxièmes personnes » pourront s’asseoir. Je ne permettrai qu’une personne par table à trois chaises ou à deux chaises. Raison de plus pour arriver de bonne heure.

On ne parle pas pendant un test car ça sent la tentative de tricherie.

On garde ses yeux sur son écran et son clavier. Si vous voulez vous reposer les yeux, regardez au plafond ou fermez-les.

Tout se répond sur l’ordinateur.

Le seul papier permis sur votre table sera l’énoncé du test. Vous pouvez utiliser le verso des feuilles pour utiliser comme brouillon.

Quand j’annoncerai la fin du test, remettez vos répertoires, ne tardez pas. Ce n’est plus le temps d’écrire du code.

Conseil :

Si vous bloquez sur une question, allez travailler sur une autre question. Vous pourrez y revenir plus tard.

**Question 1 (36 points)**

Dans le fichier Question1.cpp, des erreurs ont été ajoutées de façons à avoir trois lignes de code erronées.

Vous devez trouver ces trois lignes erronées.

Pour chaque ligne erronée de code, vous devrez

1. Mettre la ligne erronée en commentaire en la précédant de « // ERREUR : ».
2. Ajouter une ligne de commentaire expliquant pourquoi c’est une erreur. Cette ligne sera précédée de « // EXPLICATION : ». Il se peut que votre commentaire utilise plus d’une ligne.
3. Ajouter une ligne au-dessous de la ligne contenant l’explication et y écrire la ligne de code corrigée. Mettre le commentaire « // CORRECTION » à la fin de la ligne.

Voir l’exemple ci-dessous.

Exemple

Si on avait la ligne suivante de code:

cout >> variable >> endl ;

Voici comment la correction doit être démontrée :

// ERREUR : cout >> variable >> endl ;

// EXPLICATION : Il y a erreur car on a utilisé des opérateurs

// d’extraction de flux au lieu d’opérateur d’insertion de flux.

cout << variable << endl ; // CORRECTION

Le fait qu’il y avait deux opérateurs incorrects ne comptent que pour une erreur.

**Votre travail est de trouver les trois lignes de code contenant des erreurs.**

Le répertoire Solutions contient des fichiers qui vous permettront de comparer ce qui est attendu de ce que vous obtenez. Les différences devraient pouvoir vous donner des indications sur ce qui ne va pas.

Le programme est supposé créer une table de multiplications avec un certain nombre de colonnes et un certain nombre de rangées. Ces colonnes et rangées font références aux nombres de colonnes et de rangées de produits (réponses).

Les fichiers suivants sont des exemples de sortie de ce programme (lorsqu’il sera corrigé correctement.)

Solution1\_C1\_R1.txt  : Une table de 1 colonne par 1 rangée.

Solution1\_C3\_R15 : Une table de 3 colonnes par 15 rangées.

Solution1\_C4\_R12 : Une table de 4 colonnes par 12 rangées.

Solution1\_C6\_R4 : Une table de 6 colonnes par 4 rangées.

Solution1\_C7\_R20 : Une table de 7 colonnes par 20 rangées.

**RAPPEL : Vous êtes évalués sur vos habiletés à trouver des erreurs.**

**Vous devez rendre vos réponses évidentes au lecteur que je serai.**

**Suivez les directives au début de cette question. Relisez l’exemple.**

**Les points sont attribués comme suit :**

1. **12 points par ligne erronée**
2. **Ces douze points sont ventilés comme suit :**
   1. **4 points pour avoir identifié une ligne contenant une ou plusieurs erreurs.**
   2. **4 points pour avoir donnée une explication correcte.**
   3. **4 points pour avoir corrigé la ligne de code.**

**Assurez-vous que vos réponses soient complètes.**

**Question 2 ( 32 points)**

Le fichier Question2.cpp est un pilote pour tester la fonction TriPartielBulles().

TriPartielBulles() est l’implantation de l’algorithme suivant :

|  |
| --- |
| **Niveau de travail**  Entrée : T[ ], un tableau de données  Debut et Fin, le début et la fin de l’intervalle de travail.  Sortie : T[]  Tâche : Balayer l’intervalle en poussant des « grands » éléments vers la fin de l’intervalle, en particulier, le plus grand élément sera poussé en position Fin.  TriPartielBubble( T[], Debut, Fin )  pour K = Debut jusqu’à (Fin-1)  si( T[K] > T[K+1] )  echanger( T[K], T[K+1] ) |

**Votre travail : Écrire quatre tests spécifiquement pour détecter les erreurs par un sur le contrôle de la boucle for qui se trouve dans la fonction TriPartielBulles().**

Chacun des tests a la forme suivante :

// Test #4 - détecter si le contrôle finit trop tard

cout << endl << "Test #4 - detecter si le controle finit trop tard."

<< endl << endl;

debut = 3;

fin = 7;

/\*

Proposez un test qui va permettre la détection que le contrôle

de l'intervalle commence trop tôt.

\*/

// #0 1 2 3 4 5 6 7 8 9

long Tableau4[10] = { 7, 7, 7, 7, 7 ,7, 7, 7, 7, 7 };

// Original - sera modifié par l'appel à la fonction TriPartielBulles).

long Attendu4[10] = { 7, 7, 7, 7, 7 ,7, 7, 7, 7, 7 };

// Attendu (après l'appel à la fonction TriPartielBulles).

TriPartielBulles(Tableau4, debut, fin);

if (!sontEgauxDeuxTableaux(Tableau4, Attendu4, debut, fin))

{

// affichage d’information. Voir le fichier Question2.cpp.

}

Pour fins de présentation, les tableaux ont été remplis avec la valeur 7 dans toutes les positions et pour tous les tests. L’intervalle sous considération est l’intervalle [3, 7] dans les quatre cas.

Comme vous pouvez le voir, la plupart du code des quatre tests est déjà écrit. **Mais vous devez proposer ou modifier le contenu des tableaux** pour détecter les cas spécifiques mentionnés. C’est-à-dire qu’il faut remplacer les contenus des tableaux pour créer les cas tests.

**Vous avez 8 tableaux à modifier :**

**Tableau1 et Attendu1 pour le Test #1 - détecter si le contrôle commence trop tôt ;**

**Tableau2 et Attendu2 pour le Test #2 - détecter si le contrôle commence trop tard ;**

**Tableau3 et Attendu3 pour le Test #3 - détecter si le contrôle finit trop tôt ;**

**Tableau4 et Attendu4 pour le Test #4 - détecter si le contrôle finit trop tard.**

**Les quatre TableauX sont les contenus de départ de votre tests et les quatre AttenduX sont les contenus que les TableauX correspondants devraient avoir après l’appel à TriPartielBulles() si la fonction fait bien son travail.**

Pour vous aider à vérifier que vos tests détectent quelque chose, je vous ai fourni un moyen de mettre des erreurs dans le code. Il suffit de choisir quel « for » n’est pas en commentaires.

Pour la version « correcte ».

|  |
| --- |
| void TriPartielBulles(long T[], long Debut, long Fin)  {  /\*  Vous pouvez choisir le "bon bug" en enlevant les // devant l'un des cinq for  suivants tout en gardant un seul des for sans le // au début.  \*/  // for( long K = Debut-1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tôt  // for( long K = Debut+1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tard  // for( long K = Debut ; K < Fin-1 ; K++ ) // Version avec erreur : finit trop tôt  // for( long K = Debut ; K <= Fin ; K++ ) // Version avec erreur : finit trop tard  for (long K = Debut; K < Fin; K++) // Version correcte  if (T[K] > T[K + 1])  echanger(T[K], T[K + 1]);  } |

Version avec erreur : le contrôle commence trop tôt.

|  |
| --- |
| void TriPartielBulles(long T[], long Debut, long Fin)  {  /\*  Vous pouvez choisir le "bon bug" en enlevant les // devant l'un des cinq for  suivants tout en gardant un seul des for sans le // au début.  \*/  for( long K = Debut-1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tôt  // for( long K = Debut+1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tard  // for( long K = Debut ; K < Fin-1 ; K++ ) // Version avec erreur : finit trop tôt  // for( long K = Debut ; K <= Fin ; K++ ) // Version avec erreur : finit trop tard  // for (long K = Debut; K < Fin; K++) // Version correcte  if (T[K] > T[K + 1])  echanger(T[K], T[K + 1]);  } |

Version avec erreur : le contrôle commence trop tard.

|  |
| --- |
| void TriPartielBulles(long T[], long Debut, long Fin)  {  /\*  Vous pouvez choisir le "bon bug" en enlevant les // devant l'un des cinq for  suivants tout en gardant un seul des for sans le // au début.  \*/  // for( long K = Debut-1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tôt  for( long K = Debut+1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tard  // for( long K = Debut ; K < Fin-1 ; K++ ) // Version avec erreur : finit trop tôt  // for( long K = Debut ; K <= Fin ; K++ ) // Version avec erreur : finit trop tard  // for (long K = Debut; K < Fin; K++) // Version correcte  if (T[K] > T[K + 1])  echanger(T[K], T[K + 1]);  } |

Version avec erreur : le contrôle finit trop tôt.

|  |
| --- |
| void TriPartielBulles(long T[], long Debut, long Fin)  {  /\*  Vous pouvez choisir le "bon bug" en enlevant les // devant l'un des cinq for  suivants tout en gardant un seul des for sans le // au début.  \*/  // for( long K = Debut-1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tôt  // for( long K = Debut+1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tard  for( long K = Debut ; K < Fin-1 ; K++ ) // Version avec erreur : finit trop tôt  // for( long K = Debut ; K <= Fin ; K++ ) // Version avec erreur : finit trop tard  // for (long K = Debut; K < Fin; K++) // Version correcte  if (T[K] > T[K + 1])  echanger(T[K], T[K + 1]);  } |

Version avec erreur : le contrôle finit trop tard.

|  |
| --- |
| void TriPartielBulles(long T[], long Debut, long Fin)  {  /\*  Vous pouvez choisir le "bon bug" en enlevant les // devant l'un des cinq for  suivants tout en gardant un seul des for sans le // au début.  \*/  // for( long K = Debut-1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tôt  // for( long K = Debut+1 ; K < Fin ; K++ ) // Version avec erreur : commence trop tard  // for( long K = Debut ; K < Fin-1 ; K++ ) // Version avec erreur : finit trop tôt  for( long K = Debut ; K <= Fin ; K++ ) // Version avec erreur : finit trop tard  // for (long K = Debut; K < Fin; K++) // Version correcte  if (T[K] > T[K + 1])  echanger(T[K], T[K + 1]);  } |

**Vous êtes évalués sur vos choix de tests. Est-ce que chacun de vos cas tests forcent la détection du cas en question.**

Vous trouverez dans le sous-répertoire Solution 5 fichiers Document texte :

* solution - erreur - commence trop tard.txt
* solution - erreur - commence trop tôt.txt
* solution - erreur - finit trop tard.txt
* solution - erreur - finit trop tôt.txt
* solution - version correcte.txt

Ces fichiers montrent ce à quoi pourrait ressembler vos sorties de programmes.

**Notez que j’ai édité les fichiers pour que tous les éléments des tableaux soient des 7 pour ne pas vous donnez mes exemples de cas tests.**

**N’allez pas écrire des cas tests tels que chacune des erreurs proposées déclenchent les quatre détections.**

**Vos cas tests devraient être tels qu’une erreur ne devrait déclencher de préférence qu’une des détections et au plus deux (sinon, c’est du n’importe quoi.)**

**Lors de la correction, je vais essayer votre programme avec les quatre erreurs et la version correcte pour voir comment vos cas tests se défendent.**

**Il y a 4 tests pour 32 points, donc 8 points par test.**

**Question 3 (32 points)**

Complétez le code tel que demandé.

Dans le fichier Question3.cpp, vous trouvez la fonction triPartielBullesSpecial() dont le code est manquant.

**Vous devrez en écrire le code en implantant l’algorithme suivant.**

triPartielBullesSpecial( T[], debut, delta, fin )

pour K = debut jusqu’à (fin-delta) par étape de delta

if( T[K] > T[K+delta])

echanger( T[K], T[K+delta] )

Le fichier Solution3.txt montre ce que le programme Qestion2.cpp est supposé produire à l’écran.

Ci-après, vous verrez les mêmes informations mais avec des valeurs surlignées pour vous montrer quels éléments sont balayés par la boucle de la fonction triPartielBullesSpecial(). Ce ne sont pas tous les éléments qui sont balayés.

Test 1

debut = 0 delta = 2 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 7 8 5 6 3 4 1 2 9 0

Test 2

debut = 1 delta = 2 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 6 7 4 5 2 3 0 1 8

Test 3

debut = 0 delta = 3 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 6 8 7 3 5 4 0 2 1 9

Test 4

debut = 1 delta = 3 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 5 7 6 2 4 3 8 1 0

Test 5

debut = 2 delta = 3 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 8 4 6 5 1 3 2 7 0

Test 6

debut = 0 delta = 4 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 5 8 7 6 1 4 3 2 9 0

Test 7

debut = 1 delta = 4 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 4 7 6 5 0 3 2 1 8

Test 8

debut = 2 delta = 4 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 8 3 6 5 4 7 2 1 0

Test 9

debut = 3 delta = 4 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 8 7 2 5 4 3 6 1 0

Test 10

debut = 3 delta = 4 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 8 7 2 5 4 3 6 1 0

Test 11

debut = 0 delta = 5 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 4 8 7 6 5 9 3 2 1 0

Test 12

debut = 1 delta = 5 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 3 7 6 5 4 8 2 1 0

Test 13

debut = 2 delta = 5 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 8 2 6 5 4 3 7 1 0

Test 14

debut = 3 delta = 5 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 8 7 1 5 4 3 2 6 0

Test 15

debut = 4 delta = 5 fin = 9

Indice : 0 1 2 3 4 5 6 7 8 9

Avant : 9 8 7 6 5 4 3 2 1 0

Apres : 9 8 7 6 0 4 3 2 1 5