Trimestre Automne, 2024

Mohamed Lokbani

IFT1169 – Examen Final –

Inscrivez tout de suite : votre nom et le code permanent.

Nom : _____| Prénom(s) : _____|

Signature : _____| Matricule : _____|

Date : mercredi 11 décembre 2024

Durée : 3 heures (de 18h30 à 21h30)

Local : Z-310, Pavillon Claire McNicoll

Directives :

- Toute documentation est permise.
- Appareils électroniques **non** permis.

- Répondre directement sur le questionnaire.
- Les réponses **doivent être brèves, précises, claires et nettement présentées.**

1._____/10 (1.1)

2._____/15 (2.1)

3._____/18 (3.1)

4._____/18 (4.1, 4.2, 4.3)

5._____/19 (5.1)

6._____/20 (6.1, 6.2, 6.3, 6.4, 6.5)

Total : _____/100

Directives officielles

* Interdiction de toute communication verbale pendant l'examen.

* Interdiction de quitter la salle pendant la première heure.

* L'étudiant qui doit s'absenter après la première heure remettra sa carte d'étudiant au surveillant, l'absence ne devant pas dépasser 5 minutes. Un seul étudiant à la fois peut quitter la salle.

* Toute infraction relative à une fraude, un plagiat ou un copiage est signalée par le surveillant au directeur de département ou au professeur qui suspend l'évaluation.

F.A.S

Exercice 1 (10 points) trouvez la meilleure correspondance pour les descriptions ci-dessous en y associant un nombre de 1 à 11. Le nombre ne peut-être utilisé qu'une seule fois. Nous avons complété une description à titre d'exemple.

[_____] en [C++ (2)] sont essentiels pour organiser le code et éviter les [conflits] de noms.

La déclaration d'un espace de noms se fait à l'aide du mot clé [_____]. Il définit [_____] pour les identificateurs comme les variables, les fonctions et les classes.

On utilise [_____] pour accéder à un élément dans un namespace.

Les namespaces peuvent être [_____]. On peut aussi créer des [_____] pour raccourcir les références.

La directive [_____] permet d'importer tous les éléments d'un namespace. Cependant, cela peut parfois causer [_____].

Enfin, il est important de noter que le namespace [_____] est l'espace par défaut pour tous les identificateurs qui ne sont pas explicitement placés dans un namespace nommé.

L'espace [_____] regroupe les flux standards cin, cout et cerr.

Les mots sont à l'infinitif, sans les articles, au masculin et au singulier.

alias	1	
~~c++~~	~~2~~	X
conflit	3	
espace de noms	4	
global	5	
imbriqué	~~6~~	
namespace	7	
opérateur de portée	8	
portée	9	
std	10	
using	11	

Exercice 2 (15 points) ce programme compile et s'exécute correctement.

```
 1 #include <iostream>
 2 #include <exception>
 3 #include <string>
 4
 5 class Test {
 6 public:
 7        Test(){std::cout << "C1\n";};
 8        ~Test(){std::cout << "D1\n";};
 9        const char* Appel() { return "R1\n"; }
10 };
11 void Exemple(){
12        std::cout<< "R2\n";
13        throw Test();
14 }
15 int main(){
16        std::cout << "I1\n";
17        try{
18               std::cout << "T1\n";
19               Exemple();
20        }
21        catch( Test E ){
22               std::cout << "H1\n";
23               std::cout << E.Appel();
24        }
25        catch(char* str ){
26               std::cout << str;
27        }
28        std::cout << "I2\n";
29        return 0;
30 }
```

Tout en justifiant votre réponse, que va-t-il afficher en sortie?

Exercice 3 (18 points) ce programme compile et s'exécute correctement.

```
 1 #include <iostream>
 2
 3 template <class T> T Sortie(T i, T j){std::cout << "1\n"; return i;}
 4 template <class T> T Sortie(T i, int j){std::cout << "2\n"; return i;}
 5 int Sortie(int i, int j){std::cout << "3\n"; return i;}
 6
 7 int main() {
 8
 9       double a = 2.0; int b = 2; long int c = 2; char d = 2;
10
11       Sortie(a,a);
12       Sortie(a,b);
13       Sortie(b,a);
14       Sortie(b,b);
15       Sortie(b,d);
16       Sortie(d,d);
17       Sortie(a,c);
18       Sortie(c,c);
19       Sortie(d,a);
20
21       return 0;
22 }
```

Tout en justifiant votre réponse, que va-t-il afficher en sortie?

Exercice 4 (18 points) nous avons déclaré dans la fonction « main » deux threads « t1 » et « t2 » qui font appel à la fonction « Traitement ». Soit le fragment de code suivant :

```
int balance = 0;

int FaireUnDepot(int montant) {
       int ancbalance = balance;
       balance = ancbalance + montant;
       return ancbalance;
}
void Traitement() {
       FaireUnDepot(10);
       std::cout << "Balance = " << balance \
          << " dans le thread " \
          << std::this_thread::get_id() \
          << "\n";
       return;
}
```

La fonction « FaireUnDepot » sert à déposer un certain « montant » dans un compte; à calculer la nouvelle balance et retourner l'ancienne valeur vers la fonction appelante.

La fonction « Traitement » est la fonction exécutée par les deux threads « t1 » et « t2 ». Elle sert à afficher la balance et l'identifiant du thread exécuté.

4.1 Expliquez le comportement des deux threads pour le scénario suivant :

t1	t2	Balance
Traitement()		0
FaireUnDepot(10)		0
ancbalance = balance		0
balance = ancbalance + montant		10
return ancbalance		10
	Traitement()	10
	FaireUnDepot(10)	10
	ancbalance = balance	10
	balance = ancbalance + montant	20
	return ancbalance	20

4.2 Expliquez le comportement des deux threads pour le scénario suivant :

t1	t2	Balance
`Traitement()`		0
`FaireUnDepot(10)`		0
`ancbalance = balance`		0
	`Traitement()`	0
	`FaireUnDepot(10)`	0
	`ancbalance = balance`	0
	`balance = ancbalance + montant`	10
	`return ancbalance`	10
`balance = ancbalance + montant`		10
`return ancbalance`		10

4.3 tout en justifiant votre réponse, à partir des deux scénarios « 4.1 » et « 4.2 », pensez-vous qu'il est nécessaire d'introduire des modifications dans le précédent fragment de code? Si oui, lesquelles?

Exercice 5 (19 points) ce fragment de code est syntaxiquement correct.

```
 1 MaFrame(const wxString& title) : wxFrame(NULL, wxID_ANY, title, \
 2                                     wxDefaultPosition, wxSize(500, 400)) {
 3
 4    wxPanel* panel = new wxPanel(this, wxID_ANY);
 5
 6    wxBoxSizer* mainSizer = new wxBoxSizer(wxVERTICAL);
 7
 8    wxStaticText* header = new wxStaticText(panel, wxID_ANY, wxT("Un exemple"),\
 9                               wxDefaultPosition, wxDefaultSize, wxALIGN_CENTER);
10
11    mainSizer->Add(header, 0, wxEXPAND | wxALL, 10);
12
13    wxBoxSizer* hbox = new wxBoxSizer(wxHORIZONTAL);
14    wxTextCtrl* text1 = new wxTextCtrl(panel, wxID_ANY);
15    wxTextCtrl* text2 = new wxTextCtrl(panel, wxID_ANY);
16    hbox->Add(text1, 1, wxEXPAND | wxALL, 5);
17    hbox->Add(text2, 1, wxEXPAND | wxALL, 5);
18
19    mainSizer->Add(hbox, 0, wxEXPAND | wxALL, 10);
20
21    wxGridSizer* gridSizer = new wxGridSizer(2, 2, 10, 10);
22
23    wxButton* btn1 = new wxButton(panel, wxID_ANY, wxT("Bouton 1"));
24    wxButton* btn2 = new wxButton(panel, wxID_ANY, wxT("Bouton 2"));
25    wxButton* btn3 = new wxButton(panel, wxID_ANY, wxT("Bouton 3"));
26    wxButton* btn4 = new wxButton(panel, wxID_ANY, wxT("Bouton 4"));
27
28    gridSizer->Add(btn1, 0, wxEXPAND);
29    gridSizer->Add(btn2, 0, wxEXPAND);
30    gridSizer->Add(btn3, 0, wxEXPAND);
31    gridSizer->Add(btn4, 0, wxEXPAND);
32
33    mainSizer->Add(gridSizer, 1, wxEXPAND | wxALL, 10);
34
35    panel->SetSizer(mainSizer);
36
37    this->Fit();
38 }
```

À noter que la frame « MaFrame » a comme titre « Exo05 ».

Dessiner cette « frame » tout en expliquant votre démarche.

Exercice 6 (20 points) le fichier « fruits.txt » contient ce qui suit :

```
Raisins: 2
Bananes: 10
Pommes: 8
Poires: 7
Oranges: 2
Mandarines: 1
Figues: 1
```

Le premier élément du fichier représente le type de fruit, le second élément représente la quantité disponible. On vous demande d'écrire un programme qui réalise une suite d'opérations dans l'ordre mentionné.

Lors de la correction, je vais tenir compte de la clarté de votre code, de son optimisation, de l'utilisation maximale de tout ce qui gravite autour des STL (conteneurs, algorithmes, C++11, etc.). Faites l'effort d'optimiser votre code.

6.1 Ce programme peut-être exécuté sous les deux formes suivantes :

```
exo6.exe unfichier.txt
exo6.exe unfichier.txt A
```

Le programme doit vérifier la présence des arguments, ouvrir le fichier « unfichier.txt » afin de lire son contenu et le stocker dans un conteneur approprié. En cas d'erreur, le programme doit s'arrêter avec un message approprié. Le 2e argument (la lettre A) est optionnel. Si l'argument est présent, les valeurs associées aux fruits sont ajustées. Cet ajustement est expliqué dans **6.3**.

6.2 Afficher le contenu du conteneur dans un ordre alphabétique par le nom des fruits. Vous allez obtenir cet affichage à l'écran pour le fichier « fruits.txt » :

```
Tri par le nom du produit:
Bananes: 10
Figues: 1
Mandarines: 1
Oranges: 2
Poires: 7
Pommes: 8
Raisins: 2
```

6.3 Nous allons appliquer maintenant l'effet de l'option A, en suivant cet algorithme :

Valeur dans le fichier	Nouvelle valeur
1	**2** (on la remplace par 2)
2	**4** (on la remplace par 4)
> 2	**+3** (on lui ajoute 3)

Le résultat d'un tel traitement est présenté avec la question **6.5**.

6.4 Trier maintenant le conteneur sur le nombre de jours dans un ordre croissant. Ce tri a lieu quand même indépendamment de la présence ou pas de l'option A.

6.5 Afficher le contenu du conteneur sur la sortie standard. Vous allez obtenir l'affichage suivant, en fonction de la présence ou pas de l'option A :

Sans l’option A	Avec l’option A
Ordonner sur le nombre de jours: Figues: 1 Mandarines: 1 Oranges: 2 Raisins: 2 Poires: 7 Pommes: 8 Bananes: 10	Ordonner sur le nombre de jours: Figues: 2 Mandarines: 2 Oranges: 4 Raisins: 4 Poires: 10 Pommes: 11 Bananes: 13

Joyeuses Fêtes et Bonne Année!