**INCOMPATIBILITES ENTRE C ET C++**

1. Ecrire correctement en **C++** ce programme qui est correct en **C**. Affichez à l’écran ses résultats

**#include<stdio.h>**

**#include<stdlib.h>**

**#define nb 10**

**#define exclus 5**

**int main(void)**

**{ int valeurs[nb], i, nbval = 0 ;**

**printf(˝Saisissez %d valeurs : \n˝, nb);**

**for(i=0 ; i<nb ; i++)scanf(“%d”, valeurs+i);**

**for(i=0 ; i<nb ; i++)**

**switch (valeurs[i])**

**{ case exclus-1  :**

**case exclus   :**

**case exclus+1 : nbval ++;**

**}**

**printf(˝%d valeurs sont interdites \n˝, nbval);**

**}**

**SPECIFICITES DE C++**

1. Ecrivez un programme C++ qui déclare 2 variables **n** et **p** de type **double**, une référence **refN** qui fait référence à **n**, puis un pointeur **pointP** qui pointe sur l’adresse de **p**. Rangez respectivement les valeurs 25 et 13 dans **n** et **p**. Puis modifiez ces valeurs indirectement à partir des adresse **refN** et **pointP**.
2. Ecrivez un programme C++ dans lequel : on déclare un tableau d’entiers **tab** de dimension 10. Deux fonctions, **remplitab** et **printab**, permettent de remplir et imprimer le tableau, utilisez ces fonctions dans le programme. Déclarez une référence vers l’élément 5 du tableau et incrémentez sa valeur. Déclarez un pointeur vers l’élément 7, incrémentez le pointeur et la valeur pointée.
3. Ecrivez un programme C++ dans lequel :

* On déclare un entier **nb = 10** ;
* Déclarez une référence vers cet entier ;
* Déclarez un pointeur vers l’adresse de cet entier ;
* Dans les 2 cas, affichez la variable, l’adresse de la variable et la valeur pointée.

5. Ecrivez une fonction qui renvoie dans 2 paramètres passés par référence la surface **area** et la circonférence **circumf.** Et faites un jeu d’essai dans lequel on saisit le rayon **r** et affiche les résultats : surface et circonférence.d’un cercle.

6. Soit le modèle de structure : **struct essai**

**{ int n; double x;**

**}**

Ecrivez une fonction **raz** permettant de remettre à zéro les 2 champs d’une structure de ce type, transmise en argument : par pointeur, puis par référence. Dans les 2 cas, on écrira un programme d’essai de la fonction ; il affichera les valeurs d’une structure de ce type, après appel de ladite fonction.

7. Ecrivez en C++, les instructions suivantes en utilisant les opérateurs **new** et **delete**.

**#include<stdio.h>**

**#include<malloc.h>**

**int main(void){ int nbval, \*adi; double \*add, \*adtab;**

**adi = (int\*)malloc(sizeof(int));**

**add = (double\*)malloc(sizeof(double)\*100);**

**printf(˝ Combien de valeurs ? ˝); scanf(˝%d˝, &nbval);**

**adtab = (double\*) malloc(sizeof(double)\*nbval);**

**…**

**}**

8. Transformez le programme suivant pour que la fonction devienne une fonction en ligne.

**#include<iostream>**

**using namespace std;**

**int main()**

**{ int fct (char, int);**

**int p, n=150; char c= ‘s’;**

**p=fct(c, n);**

**cout<<˝fct (\’ ˝<<c<<˝\’, ˝<<n<< ˝) vaut : ˝<<p;**

**return 1;**

**}**

**int fct (char car, int nb)**

**{ int res;**

**if(car==’a’)res=nb+car;**

**else if(car==’s’)res=nb-car;**

**else res=nb\*car;**

**return res;**

**}**

b) Comment faudrait-il procéder si l’on souhaitait que la fonction **fct** soit compilée séparément ?

**Notions de classe, constructeur et destructeur**

9. Réaliser une classe **Rational** permettant de manipuler un rationnel de 2 nombres entiers. On prévoira un constructeur recevant en argument le numérateur et le dénominateur du rationnel, une fonction membre **convert** retournant le résultat du rationnel en réel double précision, une fonction membre **invert** qui inverse le numérateur en dénominateur et vice-versa, une fonction membre **print** se contentant d’afficher le rationnel comme (num/den).

Les numérateur et dénominateur sont des données privées de la classe. Ecrivez un programme d’essai, déclarant un rationnel, l’affichant, calculant son résultat en réel double, l’inversant et l’affichant de nouveau.

10. Changez la classe **Rational** afin qu’on puisse créer par le constructeur un objet sans paramètre, avec un seul paramètre et avec 2 paramètres (utilisez des valeurs d’arguments par défaut).

11. Fonctions d’accès de la classe **Rational** sont des fonctions membres publiques donnant accès aux données membres privées ou publiques.

Réalisez la même classe **Rational**, mais ne comportant pas la fonction **print**. Pour respecter le principe d’encapsulation des données, prévoyez 2 fonctions membres publiques (nommées **numerator** et **denominator**) fournissant en retour le numérateur et le dénominateur du rationnel. Ecrivez un programme d’essai pour cette nouvelle classe.

12. Soit la classe **Rational** créée à l’exercice 10. Adaptez cette classe de manière à ce qu’elle gère un compteur qui comptabilise le nombre d’objets de cette classe.

Utilisez un programme d’essai pour qu’il affiche le résultat suivant :

Maintenant il y a 2 rationnels,

Maintenant il y a 6 rationnels,

Maintenant il y a 2 rationnels,

Maintenant il y a 3 rationnels,

13, Implémentez une classe **chaine**. Chaque objet de cette classe représente une chaine de caractères. Les données membres sont la longueur et la chaine de caractères en elle-même. En plus des constructeur, destructeur, fonctions d’accès et fonction d’affichage, définissez une fonction d’extraction de caractère.

14.Ecrire un programme c++ qui déclare une classe **HEURE** contenant comme données privées entières : **heure** et **minute**, un constructeur, un destructeur qui affiche l’information **« Appel du destructeur »**, une fonction appelée **ecritInfo** qui affiche l’heure qu’il sera dans une minute plus tard. Utiliser un programme d’essai.

Exemple : quand vous déclarez un objet de type **HEURE**, en appelant la fonction **ecritInfo**, si l’objet a reçu 21 puis 32, le programme affichera **« Dans une minute, il sera 21 heures 33 »**.

**Surcharge d’opérateurs et héritage**

15. Définir une classe permettant d’additionner, de soustraire, de multiplier, de diviser et de donner l’opposé (-) des fractions (rationnels comprenant numérateur et dénominateur) en utilisant la surcharge de ces opérateurs. Créer un programme permettant de tester cette classe.

16. Soit la classe **vecteur3d** qui contient les coordonnées du vecteur (double), un constructeur initialisant les données membres avec des valeurs par défaut à 0. Il faut définir l’opérateur **+** pour qu’il fournisse la somme de deux vecteurs et l’opérateur binaire **\*** pour qu’il fournisse le produit scalaire de deux vecteurs. Créer un programme permettant de tester cette classe.

17. Créer une classe **personnel** comprenant 3 données membres (nom, position, salaire), un constructeur initialisant les données membres et une fonction **affiche\_personnel** qui affiche les informations sur le personnel. Créer une classe patron dérivant publiquement de la classe personnel et ayant 2 données membres (bonus\_annuel, marque de voiture de fonction), un constructeur initialisant les données membres et une fonction affiche\_patron qui affiche les informations sur le patron (entant que membre du personnel et patron). Créer un programme permettant de tester ces classes.

18. Modifier les classes définies dans l’exercice 17 en utilisant la fonction **affiche** dans la classe de base et la classe dérivée. Modifier le programme pour tester les modifications.

b) Créer dans le programme principal un pointeur sur la classe **personnel** et un objet de la classe **patron**. Le pointeur sur la classe de base contient l’adresse de la classe dérivée. Appeler la fonction **affiche** en utilisant le pointeur sur la classe de base (il faut qu’elle affiche les informations sur la classe dérivée).

19. Créer une classe **nombre** formée d’une unique variable entière et d’une méthode **affiche\_nb** puis définir 3 classes dérivées afin d’afficher la variable aux formats hexadécimal, octal et décimal à l’aide d’une méthode **affiche\_nb**.