Ecole préparatoire en Sciences et Techniques d'Oran
Module : Informatique 2
2^{me} année, Semestre 4, 2011/2012 — Examen de synthèse
Juin 2012 — Durée : 02h00mn

Partie cours : (5.50 pts)

Exercice 1 : les fichiers (3.00 pts)

1. Associer à chaque information de la colonne **A**, une et une seule information de la colonne **B** :

Colonne **A**	Colonne **B**
(1) ifstream	(a) écrire un caractère
(2) ofstream	(b) ouvrir le fichier en écriture
(3) eof	(c) test de la fin du fichier
(4) getline	(d) lire un caractère
(5) put	(e) fermer le fichier
(6) get	(f) lire une ligne complète
(7) close	(g) ouvrir le fichier en lecture

2. Donner un programme en C++ qui permet de saisir et mémoriser dans un fichier nommé *repertoire.txt*, le nom et le numéro de téléphone de 10 personnes,

Exercice 2 : la POO (2.50)pts

Soient le code en C++ suivant :

```
class Base
{
  public:
  Base() { cout << "Constructeur de la classe Base" << endl; };
  ~Base() { cout << "Destructeur de la classe Base" << endl; };
  MethodeB(){};
  protected:
  void FonctionB(){};
  private:
  void FB(){};
};
class Derive : public Base
{
  public:
  Derive() { cout << "Constructeur de la classe Derive" << endl; };
  ~Derive() { cout << "Destructeur de la classe Derive" << endl; };
};
void main()
{
  Derive D ;
}
```

1. Soit le tableau suivant :

	MethodeB()	FonctionB()	FB()
Classe *Derive*	...	...	...
Accès utilisateur (*main()*)	...	...	...

   (a) donnez le type d'accès par rapport à la classe dérivée "Derive" : publique, protégé ou privé,

   (b) dites si l'accès est possible par rapport au programme principal : Oui ou Non.

2. Quel est le résultat d'exécution du programme? (faire la trace)

Partie TD : (14.50 pts)

Exercice 1 : les piles (5.00 pts)

L'analyse syntaxique est une phase de la compilation qui nécessite une pile. Par exemple, le problème de la reconnaissance des mots bien parenthèsés, nous devons écrire un programme qui :
– accepte les mots comme (), ()(()) ou ((()())()) ;
– rejette les mots comme)(, ()(() ou ((()()))).
En utilisant les piles, donnez :

1. la structure de la pile ainsi que les fonctions *empiler* et *depiler*,
2. la fonction *correct* qui retourne TRUE si une chaîne de caractères est bien parenthèsée et FALSE sinon,
3. le programme principal qui permet de saisir la chaîne de caractères et de tester la fonction *correct*.

Exercice 2 : les arbres binaires (5.25 pts)

Un arbre binaire de recherche est tel que tout noeud a une valeur supérieure à celles des noeuds de son sous arbre gauche et inférieure à celles des noeuds de son sous arbre droit.

Exemple 1 *(arbre binaire de recherche)*
Pour une liste aléatoire de 14 entiers : 25, 60, 35, 10, 5, 20, 65, 45, 70, 40, 50, 55, 30, 15, on obtient l'arbre de recherche suivant :

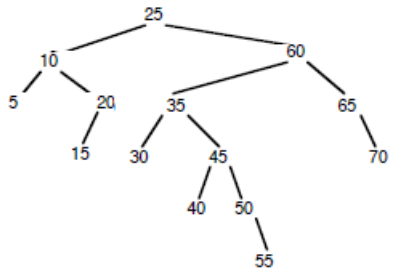

FIGURE 1 – Arbre binaire de recherche

1. donnez la structure de l'arbre, sachant que chaque noeud porte une valeur de type entier,
2. donnez les fonctions :
   – *create* qui permet de créer un noeud,
   – *insert* récursive qui insère un élément (une valeur entière) dans l'arbre en respectant le principe de l'arbre de recherche,
   – *hauteur* qui calcule la hauteur de l'arbre.
3. écrire le programme principal qui permet de tester les fonctions *insert* et *hauteur*.

Exercice 3 : POO (4.25 pts)

1. Ecrire une classe *personne* permettant de décrire complètement une personne, sachant que l'on souhaite avoir autant d'informations que dans la phrase suivante : "M. Benali est né en 1965, il est célibataire",
2. Ajouter un constructeur à la classe *personne*,
3. Ajouter à la classe *personne*, une méthode de nom *retourneinfos*. Cette méthode doit afficher une chaîne de caractères similaire à la phrase donnée ci-dessus,
4. Ajouter une méthode *age* qui renvoie l'âge de l'individu en fonction d'une année donnée en paramètre,
5. Ecrire un programme qui déclare 3 variables de type *personne*, crée 3 instances de personne et teste les fonctions *retourneinfos* et *age*.

N.B : faire attention à l'emplacement des mot-clès *public*, *private* et *protected*.

FZ.LEBBAH

Ecole préparatoire en Sciences et Techniques d'Oran
Module : Informatique 2
2^{me} année, Semestre 4, 2011/2012 — Corrigé type de l'examen de synthèse
Juin 2012 — Durée : 02h30mn

Partie cours : (5.50 pts)

Exercice 1 : les fichiers (1.75+ 1.25= 3.00pts)

1. ifstream ⟶ (g) ouvrir le fichier en lecture
2. ofstream ⟶ (b) ouvrir le fichier en écriture
3. eof ⟶ (c) test de la fin du fichier
4. getline ⟶ (f) lire une ligne complète
5. put ⟶ (a) écrire un caractère
6. get ⟶ (d) lire un caractère
7. close ⟶ (e) fermer le fichier

Le programme :

```
# include <fstream.h>
main ( )
{
  string nom, tel;
  ofstream fichrep ("repertoire.txt"); // ouverture du répertoire en écriture
  for(int i=0; i<10; i++)
  {
    cout << "\npersonne "<< i+1 << ":\n"
    cout << "nom? ";  cin >> nom;  fichrep <<nom << " ";
    cout << "\ntelephone?"; cin >> tel; fichrep << tel << "\n";
  }
  fichrep.close();
}
```

Exercice 2 : la POO (1.50+1.00=2.50pts)

1. Le tableau :

	MethodeB()	FonctionB()	FB()
Classe *Derive*	publique	protégé	privé
Accès utilisateur (*main()*)	Oui	Non	Non

2. Le résultat d'exécution du programme :

```
Constructeur de la classe Base
Constructeur de la classe Derive
Destructeur de la classe Derive
Destructeur de la classe Base
```

Partie TD : (14.50 pts)

Exercice 1 : les piles (5.00 pts)

```
#include <iostream>  // (0.25 pt pour les includes)
#include <stdlib.h>
using namespace std;
struct cellule  // (0.5 pt pour la structure)
{
  char cc;
```

```
  cellule *next;
};
cellule* sommet = NULL;

cellule* cellnew(char c)     // (1.00 pt)
{
  cellule* cel = new cellule;
  if (cel == 0) {
  cout << "Plus de mémoire" << endl;
  exit(1);
 };
  cel->cc = c;
  return cel;
}

void empiler(cellule * &l, char c)   // (0.50 pt)
{
  cellule* cel = cellnew(c);
  cel->next = l;
  l = cel;
}

void depiler(cellule * &l) (0.50 pt)
{
  if (l)
  {
   cellule * cel = l;
   cout << "caractère dépilé : " << l->cc << endl;
   l = l->next;
   delete cel;
  }
  else cout << "pile vide !" << endl;
}


bool correct(cellule * &s, char *t)  // (1.25 pts)
{
  for (int i=0; t[i]!='\0'; i++)
  {
    if (t[i] == ')')
    {
     if (!s) return false;
     else depiler(s);
    }
    else
    if (t[i] == '(') empiler(s, t[i]);
  }
  if (s) return false;
  else return true;
}

main()    // (1.00 pt)
{
  char t[30];
  cout << "chaîne de caractères = "; cin >> t;
  bool cr = correct(sommet, t);
  if (cr) cout << "la chaîne de caractères est correcte" << endl;
```

```
  else
  cout << "la chaîne de caractères est incorrecte" << endl;
}
```

Exercice 2 : les arbres binaires (5.25 pts)

```
#include <iostream>  // (0.25 pt)
#include <stdlib.h>

using namespace std; (0.5 pt)
struct noeud
{
  int val;
  noeud *gauche;
  noeud *droit;
};
noeud *racine = NULL;

noeud *create(int v)   // (1.00 pts)
{
  noeud *nd = new noeud;
  if (nd == NULL) {
  cout << "Plus de mémoire!" << endl;
  exit(1);
 };
 nd->val = v;
 nd->gauche = NULL;
 nd->droit = NULL;
 return nd;
}

void insert(noeud * &r, int v) (1.50 pts)
{
  if (r==NULL)
  {
    cout << "racine" << endl;
    noeud *nt = create(v);
    r=nt;
  }
  else
  {
    if (r->val<v) insert(r->droit, v);
    else insert(r->gauche,v);
  }
}
unsigned int max(unsigned int i1, unsigned int i2)   // (0.25 pt)
{
  if (i1>i2) return i1;
  else return i2;
}
unsigned int hauteur(noeud * T)   // (1.00 pt)
{
  if (T==NULL) return 0;
  else return(1+max(hauteur(T->gauche), hauteur(T->droit)));
}
main()   // (0.75 pt)
{
```

```
  int h,n;
  cout << "Nombre de noeuds = "; cin >> n;
  for(int i=0; i<n; i++)
  {
    cout << "val = ";
    cin >> h;
    insert(racine,h);
  }
  cout << "La hauteur de l'arbre = " << hauteur(racine) << endl;
}
```

Exercice 3 : POO (04.25 pts)

```
#include <iostream>  // (0.25 pt)
#include <string.h>
using namespace std;

class personne   // (1.25 pts)
{
 private :
 string nom_pren;
 unsigned int annee_nais;
 string cel_mari;
 public :
 personne(string, unsigned int, string);
 void retourneinfo( );
 unsigned int age(unsigned int);
};
personne::personne(string np, unsigned int an, string cm)  //  (0.75 pt)
{
nom_pren=np;
annee_nais = an;
cel_mari = cm;
}

void personne::retourneinfo( ) (0.50 pt)
{
cout << nom_pren << " est né en " << annee_nais << ", il est " << cel_mari << endl;
}

unsigned int personne::age(unsigned int a)  // (0.25 pt)
{
return a-annee_nais;
}
main()  // (1.25 pts)
{
 string s1,s2;
 unsigned int annee;
 cout << "Nom_prénom = "; cin >> s1;
 cout << "année de naissance = "; cin >> annee;
 cout << "situation familiale = "; cin >> s2;
 personne p1(s1,annee,s2);
 cout << "Nom_prénom = "; cin >> s1;
 cout << "année de naissance = "; cin >> annee;
 cout << "situation familiale = "; cin >> s2;
 personne p2(s1,annee,s2);
```

```
 cout << "Nom_prénom = "; cin >> s1;
 cout << "année de naissance = "; cin >> annee;
 cout << "situation familiale = "; cin >> s2;
 personne p3(s1,annee,s2);
 p1.retourneinfo();
 cout << "agé de : " << p1.age(2012) << "ans" << endl;
 p2.retourneinfo();
 cout << "agé de : " << p2.age(2012) << "ans" << endl;
 p3.retourneinfo();
 cout << "agé de : " << p3.age(2012) << "ans" << endl;
}
```

FZ.LEBBAH