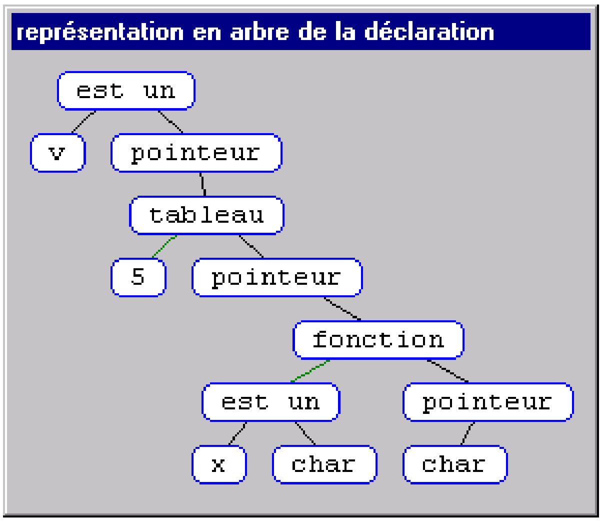
Université Pierre Mendès-France — *27 mars 2000*
DESS DCISS — *3 heures, documents autorisés*

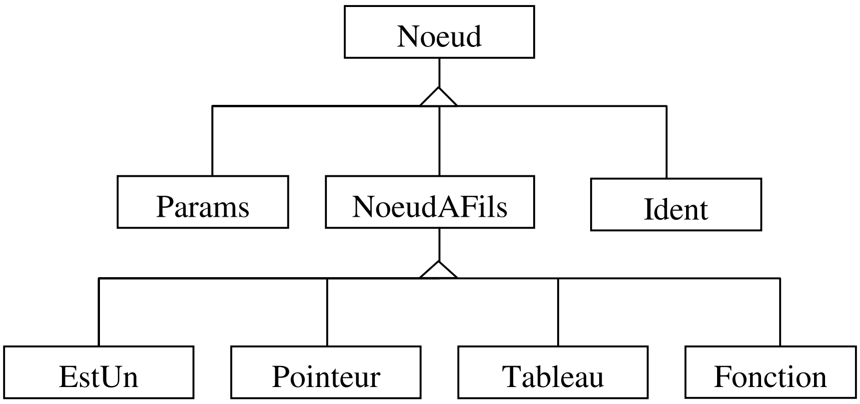
programmation examen

Les déclarations en C++ ne sont pas toujours facile à écrire ou à lire : quel est le type de *v* dans la déclaration `char *(* (*v)[5])(char x)`? Nous supposerons qu'un compilateur a su traduire la déclaration C++ en un arbre représentant cette déclaration. Pour l'exemple précédent nous aurions la représentation par l'arbre ci-contre:



Nous nous proposons, entre autre, d'écrire la définition de fonctions qui affichent la déclaration en clair (`v est un pointeur sur un tableau de 5 pointeur sur un fonction de paramètre( x est un char)qui renvoie un pointeur sur un char)`

Pour représenter une déclaration nous utiliserons la hiérarchie d'objets ci-contre :
Voir en annexe pour la représentation en C++.



Question 1.(4 points)
Ecrire les définitions des destructeurs des différentes classes de la hiérarchie.

Question 2. (4 points)
Ecrire les définitions des fonctions *int nbParam()* membres de la classe *Fonction* et de la classe *Params*, qui renvoient le nombre de paramètres de la fonction. Une fonction sans paramètre a son pointeur *param* égal à NULL.

Question 3. (6 points)
Ecrire les définitions des fonctions *AnsiString ecrireClair()* ; qui renvoie le texte en « clair » correspondant à la déclaration en C++. La déclaration *char * (* (*v)[5])[6]* sera « traduite » en : *v est un pointeur sur un tableau de 5 pointeur sur un tableau de 6 pointeur sur un char.*

Question 4. (6 points)
Ecrire les définitions des fonctions *bool verif() ;* qui vérifient que la déclaration est correcte : une variable ou un paramètre de type *void* est une déclaration incorrecte, un tableau

d'éléments de type *void* ou de fonctions est une déclaration incorrecte, et une fonction ne peut pas retourner un tableau, ni une fonction.

ANNEXE :

```
class Noeud{
  public :
     virtual ~Noeud(){}
     virtual AnsiString
                  ecrireClair()=0;
     virtual bool verif()=0;
};
```

```
class NoeudAFils : public Noeud{
  protected :
     Noeud * fils;
  public :
      NoeudAFils(Noeud * f)
          :fils(f){}
      ~NoeudAFils() ;
};
```

```
class Ident: public Noeud{
       char * nom;
    public :
       Ident(char * n)
           {nom = strdup(n);}
       ~Ident();
       AnsiString ecrireClair();
       bool verif();
       AnsiString getNom(){
                return nom ;}
};
```

```
class EstUn : public NoeudAFils{
       Ident * nom;
    public :
       EstUn(Ident * n, Noeud * f)
          : NoeudAFils(f),
            nom(n){}
       ~EstUn();
       AnsiString ecrireClair();
       bool verif();
};
```

```
class Pointeur:public NoeudAFils{
    public :
       Pointeur( Noeud * f)
               : NoeudAFils(f) {}
       ~Pointeur();
       AnsiString ecrireClair();
       bool verif();
};
```

```
class Tableau : public NoeudAFils{
      int nombre;
    public :
       Tableau( Noeud * f, int n)
        : NoeudAFils(f), nombre(n)
        { }
       ~Tableau();
       AnsiString ecrireClair();
       bool verif();
};
```

```
class Fonction :public NoeudAFils{
       Params * param;
    public :
       Fonction( Noeud * f,
                          Params*p)
        : NoeudAFils(f),
          param (p){ }
       ~ Fonction ();
       AnsiString ecrireClair();
       bool verif();
       int nbParam() ;
};
```

```
class Params : public Noeud{
      Noeud * unParametre;
      Params * suite;
   public :
      Params(Noeud *f, Params * p)
        : unParametre (f),
          suite(p){ };
      ~ Params () ;
      AnsiString ecrireClair();
      bool verif();
      int nbParam() ;
};
```