FICHE RESUME SUR LES STRUCTURES EN C++

Les structures permettent de regrouper des éléments de types différents dans un même ensemble.

1 Programme élémentaire

```
#include <iostream>     // bibliothèque de gestion des E/S
#include <conio.h>      // gestion de la console (ici _getch())
#include <string>

using namespace std;

struct Personne
{
        string nom;
        string prenom;
        string telephone[2];
        int age;

};

/*****************************
Fonction de saisie
*****************************/
Personne saisie()
{
        Personne tmp;
        cout << endl;
        cout << "NOM ? ";               getline(cin, tmp.nom);
        cout << "Prenom  ?";            getline(cin, tmp.prenom);
        cout << "Tel fixe ? ";          getline(cin, tmp.telephone[0]);
        cout << "Tel portable ? ";      getline(cin, tmp.telephone[1]);
        cout << "Age : ? ";             cin >> tmp.age;
        fflush(stdin); // vider buffer clavier
        return tmp;
}

/*****************************
Fonction d'affichage
Passage par référence (&) pour éviter de dupliquer la variable lors du passage de paramètre
const permet d'interdire la modification du paramètre
*****************************/
void affichage(const Personne &toto)
{
        cout << endl ;
        cout << "NOM : " << toto.nom << endl;
        cout << "Prenom : " << toto.prenom << endl;
        cout << "Tel fixe : " << toto.telephone[0] << endl;
        cout << "Tel portable : " << toto.telephone[1] << endl;
        cout << "Age : " << toto.age << endl;
}

/*****************************
Fonction principale
*****************************/
int main()      // Fonction principale
{
        Personne test1 = { "DUPONT", "Albert", { "0301010101","0601010101" }, 20 };
        affichage(test1);

        Personne titi;
        titi = saisie();
        affichage(titi);


        _getch();
        return 0;
}
```

Affichez la 3ème lettre du prénom de titi

```
cout <<.....
```

Affichez le 4ème chiffre du numéro de portable de titi

```
cout <<.....
```

2 Notion de sous-structure (ajouts au programme précédent)

```
/**************************

***************************/
#include <iostream>     // bibliothèque de gestion des E/S
#include <conio.h>      // gestion de la console (ici _getch())
#include <Windows.h>
#include <string>

using namespace std;

struct Personne
{
        …

};

struct Etudiant
{
        Personne individu;
        string numeroINE;
        string classe;
};


/****************************
Fonction de saisie
****************************/
Personne saisie()
{
        …
}

/****************************
Fonction de saisie etudiant
****************************/
Etudiant saisieEtudiant()
{
        Etudiant tmp;

        fflush(stdin);
        tmp.individu = saisie();

        cout << "Numero INE ? ";
        getline(cin, tmp.numeroINE);
        cout << "Classe? ";
        getline(cin, tmp.classe);

        return tmp;
}
/****************************
Fonction d'affichage d'une personne
****************************/
void affichage(const Personne &toto)
{

        …
}

/****************************
Fonction d'affichage d'un étudiant
Passage par référence (&) pour éviter de dupliquer
la variable lors du passage de paramètre

const permet d'interdire la modification du paramètre
****************************/
void affichage(const Etudiant &toto)
{
        affichage(toto.individu);       // appel de la fonction d'affichage d'un personne
        cout << "INE=" << toto.numeroINE << endl;
        cout << "CLASSE=" << toto.classe << endl;

}
```

Le champ individu correspond à une personne. il se décompose donc en :

- individu.nom
- individu.prenom
- individu.telephone
- individu.age

On appelle la fonction d'affichage du dessus avec l'individu associé comme paramètre

```
/*****************************
Fonction principale
*****************************/
int main()      // Fonction principale
{
        // La commande ci-dessous permet d'obtenir les accents
        // dans la console. Mais il faut choisir la police de
        // caractères Consolas ou Lucida
        SetConsoleOutputCP(1252);

        …


        Etudiant test2 = { { "NEYMAR", "JEAN", { "0301010101", "0601010101" },21 }, "1234567890G", "SNIR2" };
        affichage(test2);

        Etudiant test3;
        test3=saisieEtudiant();
        affichage(test3);

        _getch();
        return 0;
}
```

La partie individu en jaune

Affichez le numéro INE de **test2**

cout <<……

Affichez le 5eme chiffre du numéro INE de **test2**

cout <<……

Affichez le prénom de **test2**

cout <<……

Affichez le 4ème chiffre du numéro de portable de **test2**

cout <<……

3 Les pointeurs et les structures (ajout dans la fonction principale)

```
Personne *p = new Personne;
p->nom = "FONFEK";          // équivalent à (*p).nom="FONFEK"
p->prenom = "SOPHIE";
p->telephone[0] = "0345659878";
p->telephone[1] = "0645789865";
p->age = 25;
affichage(……);
```

Données			Adresses
*p	nom		p
	prenom		
	telephone	[0]	
		[1]	
	age		

```
// Sophie s'inscrit comme étudiante en SNIR1
// Son INE est "123456789Q"

Etudiant *q = new Etudiant;

…… ;

…… ;

…… ;

affichage(……) ;


delete p;     // libération mémoire
delete q;
```

Données				Adresses
*q	individu	nom		q
		prenom		
		telephone	[0]	
			[1]	
		age		
	numeroINE			
	classe			

4 Les tableaux de structures (ajouts dans programme principal)

4.1 Version 1 : tableau avec allocation de mémoire statique (nombre de cases du tableau défini dans le programme source)

```
// Création d'un tableau de 3 etudiants
Etudiant tab1[3] = { { { "ASSIN", "Marc", { "0301010101", "0601010101" }, 21 }, "1234567890G", "SNIR2" } ,
                     { { "DUBOIS", "Yvan", { "0302010101", "0602010101" }, 22 }, "2234567890G", "SNIR2" } ,
                     { { "AIRE", "Axel", { "0303010101", "0603010101" }, 23 }, "3234567890G", "SNIR2" } };


//Affichage de tous les étudiants
for (unsigned i = 0; i < 3; i++)
{
        cout << "\nEtudiant " << i;
        affichage(tab1[i]);
}
```

Affichez le numéro INE du 2ème étudiant (le n°1)

```
cout <<.....
```

Affichez le 5eme chiffre du numéro INE du 2ème étudiant

```
cout <<.....
```

Affichez le prénom du 3ème étudiant

```
cout <<.....
```

Affichez le 4ème chiffre du numéro de portable du 3ème étudiant

```
cout <<.....
```

4.2 Version 2 : tableau avec allocation de mémoire dynamique

```
Etudiant *lesEtudiants;
unsigned n;

cout << "Saisir le nombre d'étudiants : ";
cin >> n;

lesEtudiants = new Etudiant[n];
for (unsigned i = 0; i < n; i++)
        lesEtudiants[i] = saisieEtudiant();

//Reaffichage des étudiants
for (unsigned i = 0; i < n; i++)
{
        cout << "\nEtudiant " << i;
        affichage(lesEtudiants[i]);
}

delete[] lesEtudiants;
```

Avantage :
Le nombre d'étudiants n'a pas à être connu avant le lancement du programme.

Inconvénient :
On ne peut plus ajouter d'étudiant par la suite ni en supprimer.

4.3 Version 3 : Utilisation des vecteurs de la STL (inclure <vector>)

```
vector<Etudiant> lesEtudiants; // vecteur vide au départ
Etudiant tmp;
char encore;

do
{
        tmp = saisieEtudiant();
        lesEtudiants.push_back(tmp);    // ajout au vecteur
        cout << "Encore ? ";
        cin >> encore;
}
while (encore != 'n' && encore != 'N');

//Reaffichage des étudiants
for (unsigned i = 0; i < lesEtudiants.size(); i++)
{
        cout << "\nEtudiant " << i;
        affichage(lesEtudiants[i]);
}
```

On peut à tout moment ajouter ou supprimer un étudiant du tableau. ☺