Exercices pratiques de passage de paramètres aux fonctions

Notions abordées :

- Passage de tableaux
- Passage de valeurs
- Pointeurs
- Allocations mémoire

Passage de tableaux à une fonction

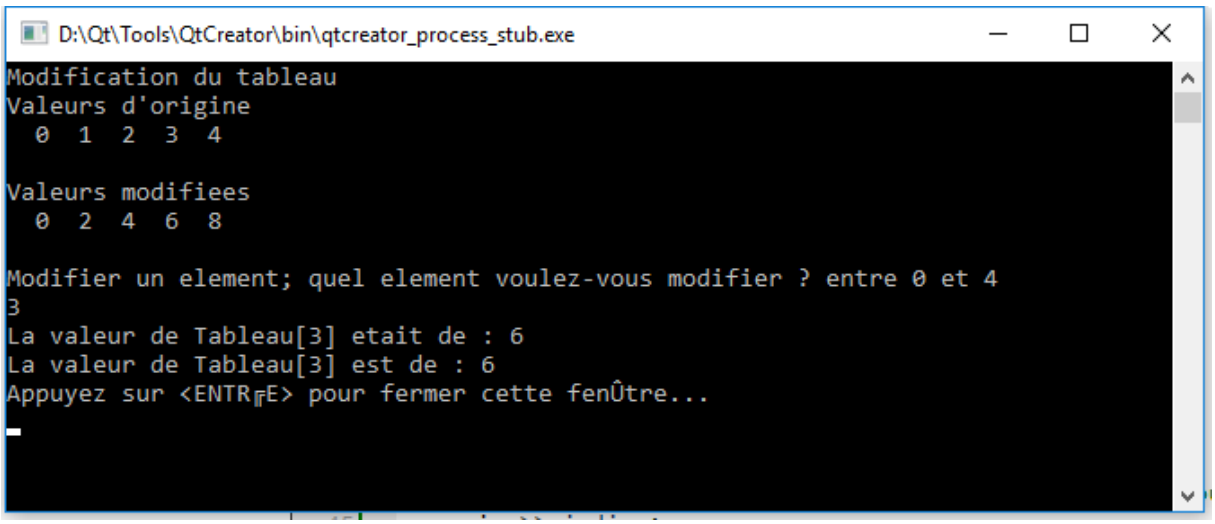
Voici trois fonctions : la fonction *main()* et deux fonctions de traitement *modifierTableau()* et *modifierElement()*

```
#include <iostream>
#include <iomanip>

using namespace std;

void modifierTableau(int Tab[],int taille)
{
    for (int i = 0; i < taille; i++)
    {
        Tab[i] *= 2;
    }
}

void modifierElement(int e)
{
    e *= 2;
}

int main()
{
    const int tailleTableau = 5;
    int Tableau[tailleTableau] = {0,1,2,3,4};
    int indice;

    cout << "Modification du tableau" << endl;
    cout << "Valeurs d'origine" << endl;
    for (int j = 0; j < tailleTableau;j++)
    {
        cout << setw(3) << Tableau[j];
    }

    cout << endl<< endl;

    modifierTableau(Tableau, tailleTableau);

    cout << "Valeurs modifiees" << endl;
    for (int j = 0; j < tailleTableau;j++)
    {
        cout << setw(3) << Tableau[j];
    }
    cout << endl<< endl;


    cout << "Modifier un element; quel element voulez-vous modifier ? entre
0 et " << tailleTableau-1 << endl;
    cin >> indice;
```

```
    cout << "La valeur de Tableau["<< indice<< "] etait de : " <<
Tableau[indice]  << endl;
    modifierElement( Tableau[indice]);
    cout << "La valeur de Tableau["<< indice<< "] est de : " <<
Tableau[indice] << endl;
    return 0;
}
```

Saisir le code précédent dans un *« non Qt Project » « Plain C++ Project »*

Compiler et exécuter ce programme. Vous obtenez ceci :



Pourquoi Tableau[3] n'a-t-il pas été modifié ?

⇨ La fonction chargée de modifier l'élément d'indice 3 du tableau (le 4ème élément) est *modifierElement()*

```
void modifierElement(int e)
{
    e *= 2;
}
```

e est passée par valeur : le compilateur prend le contenu de la variable à l'appel et la copie dans e.

L'appel de la fonction est :

modifierElement(Tableau[indice]);

Donc le compilateur prend ce qui se troue dans le tableau à l'indice « indice » (ici 3), par conséquent la valeur 6 et la copie dans e.

e n'est qu'une varaible locale à la fonction *modifierElement()* ; elle n'existe que pendant la durée de l'exécution de la fonction. On a beau modifier le contenu de e (e *= 2) ça ne change pas le contenu de Tableau[indice]

Pourquoi les valeurs du tableau sont modifiées par la fonction *modifierTableau* ?

⇨ Tout simplement parce qu'on passe à la fonction, non pas les valeurs du tableau (copie des valeurs comme pour e) mais l'adresse du premier élément du tableau. Par conséquent, quand on modifie un élément du tableau, en fait, on accède à l'emplacement mémoire connu de toutes les fonctions (et notamment la fonction appelante)

```
void modifierTableau(int Tab[],int taille)
{
    for (int i = 0; i < taille; i++)
    {
        Tab[i] *= 2;
    }
}
```

L'écriture « Tab[i] *= 2 » pourrait s'écrire « *(Tab+i) *= 2 » ; c'est-à-dire « Tab+i » c'est l'adresse Tab incrémentée de i. « *(Tab+i) » signifie le contenu mémoire pointé par l'adresse (Tab+i)

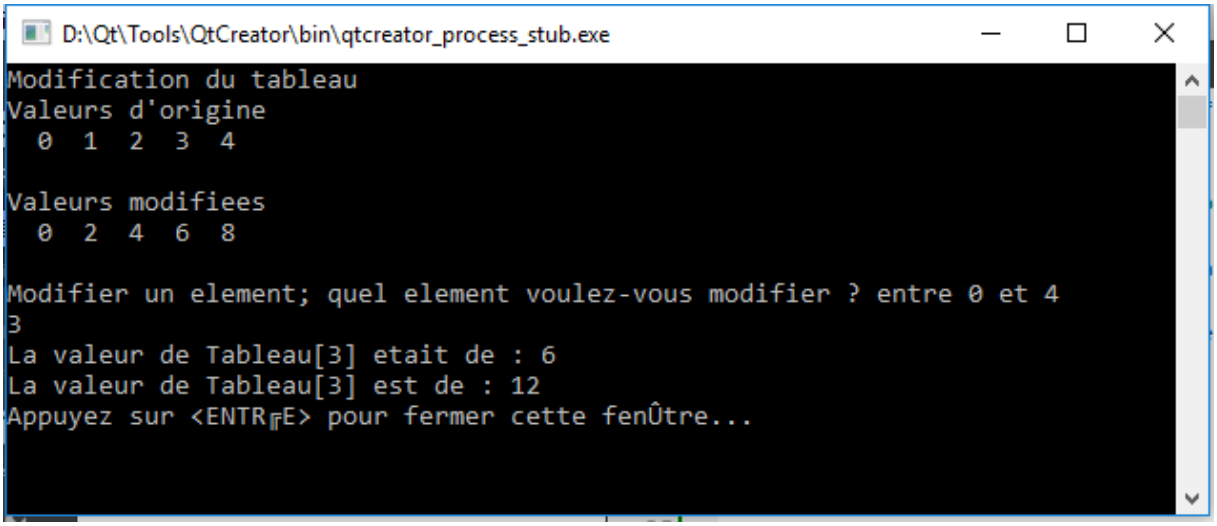
Remarque : on aurait pu écrire aussi void **modifierTableau**(int *Tab,int taille) En effet, l'écriture Tab[] et *Tab sont équivalente dans la définition de la fonction.

Comment permettre la modification d'un élément du tableau dans la fonction *modifierElement() ?*

Il suffit de passer l'adresse de la variable dont on veut modifier son contenu :

```
void modifierElement(int *e)
{
    *e *= 2;
}
```

« *e *= 2 » signifie le contenu pointé par e
On aurait pu écrire : « *e = *e*2 ; » mais ça n'est pas très lisible !
Pour utiliser cette fonction, il faut l'appeler en lui passant l'adresse de la variable dont on veut modifier son contenu :
modifierElement(&Tableau[indice]);
On spécifie ici, qu'on passe l'adresse mémoire de l'élément Tableau[indice] à l'aide de l'opérateur &
Faites les modifications nécessaires pour obtenir le résultat suivant :



Le 4ème élément a bien été modifié par la fonction *modifierElement()*

Création d'un répertoire avec tri automatique

On se propose, très modestement, de créer un répertoire de noms. On crée d'abord un répertoire en saisissant trois noms. Le programme affiche le contenu du répertoire trié. Puis on insère un quatrième nom. Le programme ré-affiche le répertoire de nouveau trié.

Il y a 5 fonctions : la fonction *main()* et 4 fonctions agissant sur le tableau *tRepertoire*

`int RemplirRepertoire(string *, int, int);` => Elle se charge de remplir le tableau passé en paramètre

`int InsererUnNom(string *, int*, int);` => Elle permet d'insérer un nom dans le tableau. Attention : le nombre de noms du tableau est augmenté de 1

`int RetrierRepertoire(string *, int, int);` => Elle trie le tableau dans l'ordre croissant

`void AfficherRepertoire(string *, int, int);` => Elle affiche le répertoire

Remarque : la taille TAILLE est le nombre maximal de noms dans le tableau. Par contre *nb_elem* est le nombre actuel de noms enregistrés. *nb_elem* doit toujours être inférieur ou égal à TAILLE.

Compléter le code suivant :

```
#include <iostream>
#include <algorithm>

using namespace std;

int RemplirRepertoire(string *, int, int);
int InsererUnNom(string *, int*, int);
int RetrierRepertoire(string *, int, int);
void AfficherRepertoire(string *, int, int);

int main()
{
    const int TAILLE = 30;
    string tRepertoire[TAILLE];
    int nb_elem = 3;
    RemplirRepertoire(tRepertoire, nb_elem, TAILLE);

    AfficherRepertoire(tRepertoire, nb_elem , TAILLE);

    InsererUnNom(tRepertoire, &nb_elem, TAILLE);

    AfficherRepertoire(tRepertoire, nb_elem , TAILLE);

    return 0;
}


int RemplirRepertoire(string *tString, int nbElem, int tailleMax)
{
    if (nbElem <= tailleMax)
    {
        for (int i = 0 ; i < nbElem ; i++)
        {
            cout << "Nom : ";
            cin >> ……
```

```
            cout << "Suivant .." << endl;
        }
        return 0;
    }
    else
    {
        return -1;
    }
}

int RetrierRepertoire(string *tString, int nbElem, int tailleMax)
{
    if (nbElem <= tailleMax)
    {
        sort(……, ……);

        return 0;
    }
    else
    {
       return -1;
    }
}

int InsererUnNom(string *tString, int* nbElem, int tailleMax)
{
    if (……<= tailleMax)
    {
        cout << "Ajout d'un nom dans le repertoire : ";
        cin >> ……
        *nbElem = …… + ……
        RetrierRepertoire(tString, *nbElem, tailleMax);
        return 0;
    }
    else
    {
       return -1;
    }
}

void AfficherRepertoire(string *tString, int nbElem, int tailleMax)
{
    if (nbElem <= tailleMax)
    {
        cout << "Repertoire ..." << endl;

        for (int i = 0 ; i < nbElem ; i++)
        {
            cout << tString[i] << endl;
        }
    }
}
```