ASR TD3
Passage de paramètres, Pointeurs, `const`

1 Passage de paramètres

1.1 Types primitifs et structs

En C, les variables de type primitif ou structure sont passées par copie. Pour le vérifier on peut modifier leur valeur dans la fonction et vérifier que la valeur n'a pas été modifiée par l'appel de fonction. Une autre méthode consiste à afficher les adresses des variables et vérifier que celles-ci sont différentes, on ne modifie donc pas la même zone mémoire dans la fonction :

```
void fct0( int i ) {
  printf( "&i = %x\n", &i );
}
void fct1( struct point p ) {
  printf( "&p = %x\n", &p );
}

int i = 2;
printf( "&i = %x\n", &i );
fct0( i );
point p;
printf( "&p = %x\n", &p);
fct1( p );
```

1.2 Pointeurs

Pour pouvoir modifier une variable passée en paramètre à une fonction, il faut passer son adresse à la fonction, c'est à dire définir un paramètre de type pointeur :

```
void fct0( int * p_a ) {
  *p_a += 1; // modif. de la valeur a l'adresse contenue dans p_a
}
...
int a = 5; // a dans la pile
fct0( &a ); // adresse de a
printf( "a = %i\n", a );
```

1.3 const

En C, le mot clé `const` permet de définir une constante. Utilisé dans les paramètres de fonctions, il permet de restreindre les opérations applicables aux paramètres durant l'appel de la fonction :

```
// On souhaite interdire la modification de la valeur pointee par p_a
void fct0( int const * p_a ) {
  *p_a += 1; // erreur de compilation
}
// On souhaite ne pas pouvoir modifier le pointeur
void fct1( int * const p_a ) {
  *p_a += 1 // ok
  p_a = (int *)malloc( sizeof( int ) ); // erreur
}
void fct2( int const * const p_a ) {
  *p_a += 1; // erreur
  p_a = (int *)malloc( sizeof( int ) ); // erreur
}
```

1.4 Différences C / Java

1. Quelles valeurs doivent être affichées par le code suivant :

```
public class Params0
{
  int i;

  Params0( int i ) {
    this.i = i;
  }

  public static void fct0( int i ) {
    i += 1;
  }

  public static void fct1( Params0 p0 ) {
    p0.i += 1;
  }

  public static void fct2( Params0 p0 ) {
    p0.i += 1;
    p0 = new Params0( 123 );
  }

  public static void main( String[] args ) {
    int i = 3;
    fct0( i );
    System.out.println( "i = " + i );

    Params0 p0 = new Params0( 1 );
    fct1( p0 );
    System.out.println( "p0.i = " + p0.i );

    fct2( p0 );
    System.out.println( "p0.i = " + p0.i );
  }
}
```

2. Compiler et exécuter ce code pour vérifier vos hypothèses.

3. Quelles règles en déduire entre le passage de paramètres de variables types primitifs et de types complexes ?

4. Qu'est devenue la valeur 123 ?

1.5 Exercices

1. Créer une structure nommée `vec3d` contenant 3 attributs de type flottant nommés x, y, z.

2. Créer une fonction nommée `zero` qui remet à 0 les attributs d'un vecteur.

3. Créer une fonction nommée `randomize` qui initialise les attributs d'un vecteur avec des valeurs aléatoires (`man 3 rand`) entre 0.0f et 10.0f.

4. Créer une fonction nommée `display` qui affiche un vecteur.

5. Créer une fonction nommée `add` qui effectue la somme de 2 vecteurs passés en paramètre et retourne ce nouveau vecteur.

6. Créer une fonction qui retourne un tableau de vecteurs dont la taille est passée en paramètre et dont les valeurs sont initialisées à l'aide de la fonction `randomize`.