Exercices sur les pointeurs

420-C21-IN | Automne 2021, par Godefroy Borduas

Opérations élémentaires sur les pointeurs

1. Valeurs des pointeurs

```
#include <iostream>

using namespace std;

int main(void) {
      int A = 1;
      int B = 2;
      int C = 3;
      int *P1, *P2;
      P1=&A;
      P2=&C;
      *P1=(*P2)++;
      P1=P2;
      P2=&B;
      *P1-=*P2;
      ++*P2;
      *P1*=*P2;
      A=++*P2**P1;
      P1=&A;
      *P2=*P1/=*P2;
      return 0;
   }
```

À partir du tableau précédent, complétez le tableau qui suit. Dans chaque case, donnez la valeur de la variable (dans la colonne) après l'exécution de l'instruction (de la ligne).

	A	**B**	**C**	**P1**	**P2**
Initialisation	1	2	3	-	-
`P1 = &A`	1	2	3	&A	-
`P2 = &C`					
`*P1 = (*P2)++`					
`P1 = P2`					
`P2 = &B`					
`*P1 -= *P2`					

	A	B	C	P1	P2
`++*P2`					
`*P1 *= *P2`					
`A = ++*P2**P1`					
`P1 = &A`					
`*P2=*P1/=*P2`					

2. Résultat

À partir du code suivant, déterminer la valeur affichée pour :

a. `t = 5` b. `t = 42` c. `t = 3`

```
#include <iostream>

using namespace std;

void Foo(int t) {
   float i = 4 * t, j = t * (t - 1); // 1
   float *ptr = &i; // 2

   *ptr = j * (*ptr) + 5 * (*ptr--); // 3
   ptr = &j; // 4
   *ptr = i * t; // 5
   j = *ptr * 2; // 6
   ptr = &i; // 7
   *ptr = j / *ptr; // 8

   cout << *ptr << endl;
}
```

3. Valeur d'un pointeur après plusieurs fonctions

Donnez la valeur de `ptr` et de `*ptr` à la fin du code suivant :

```
#include <iostream>

using namespace std;

void Foo(int*, int);
void Bar(int*, int, int, int);

int main(void) {
   int i = 2;
   int *ptr = &i;
```

```
    Foo(ptr, i);
    Bar(ptr, 1, 2, 3);
    Foo(ptr, 0);
    Bar(ptr, 3, 2, 0);
}

void Foo(int *pointeur, int t) {
    *pointeur += t * t;
}

void Bar(int *adresse, int a, int b, int c) {
    if (c < 0)
        *adresse = (a * b + (*adresse) * 2) + c * *adresse;
    else if (c > 0)
        *adresse = (b * c + (*adresse) * 2) + b * *adresse;
    else
        *adresse *= (a + b);
}
```

4. Adresses et valeurs

Soit P un pointeur qui 'pointe' sur un tableau A:

```
int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};
int *P;
P = A;
```

Quelles valeurs ou adresses fournissent ces expressions:

a) `*P+2` b) `*(P+2)` c) `&A[4]-3` d) `A+3` e) `&A[7]-P` f) `P+(*P-10)` g) `*(P+*(P+8)-A[7])`

Opérations avec tableau

5. Résultat d'une application

Qu'est-ce que le code suivant affichera ?

```
#include <iostream>

using namespace std;

int main(void) {
    int t[3];
    int i, j = 0;
    int *adt;

    for (i = 0; i < 3; i++) t[i] = j++ + i; /* 1 */

    for (i = 0; i < 3; i++) cout << t[i] << " "; /* 2 */
```

```
    cout << endl;

    for (i = 0; i < 3; i++) cout << *(t + i) - 2 << " "; /* 3 */
    cout << endl;

    for (adt = t; adt < t + 3; adt++) cout << *adt << " "; /* 4 */
    cout << endl;

    for (adt = t + 2; adt >= t; adt--) cout << *adt << " "; /* 5 */
}
```

6. Fonction de tableau

Soit deux tableaux `t1` et `t2` déclarés ainsi :

```
float t1[10], t2[10];
```

Supposons que seul le tableau t2 est initialisé avec des valeurs quelconques. Écrivez une fonction `Foo` qui prend en paramètre deux tableaux d'entier (`t1` et `t2`) sous le formalisme des pointeurs (n'oubliez pas l'indicateur de limite des tableaux). Cette fonction recopie toutes les valeurs positives de `t2` dans `t1`. Par la suite, toutes les cases qui n'auront pas été remplies dans `t1` devront être mises à 0. Enfin, imaginer l'appelle de la fonction comme suit :

```
Foo(t1, t2, 10);
```

7. Résultat d'un code de tableau

Quel sera le résultat du programme suivant :

```
#include <iostream>

using namespace std;

int main(void) {
   int t[4] = {10, 20, 30, 40};
   int *ad[4];
   int i;

   for(i = 0; i < 4; i++) ad[i] = t + i; /* 1 */
   for(i = 0; i < 4; i++) cout << ad[i] << " "; /* 2 */
   cout << endl;

   for(i = 0; i < 4; i++) cout << *ad[i] << " "; /* 3 */
   cout << endl;
```

```
    cout << *(ad[1] + 1) << " " << *ad[2] << endl; /* 4 */
}
```

8. Fonction somme

Écrivez une fonction `somme` qui renvoie la somme des éléments d'un tableau de nombre flottant. Afin de minimiser la copie en mémoire, utilisez le formalisme des pointeurs.

Écrivez un petit programme d'essai.

9. Utilisez une structure

Prenez l'énumération et la structure suivantes :

```
enum TypeValeur_e {
   MAXIMUM, MINIMUM
};

struct ExtremeTableau_s {
   TypeValeur_e TypeExtreme;
   float Valeur;
   int Position;
   float *Reference;
}
```

Nous souhaitons créer une fonction `maxmin` qui ne retourne aucune valeur. Cette fonction aura deux tâches. Déterminer la plus grande valeur entre deux tableaux de nombre flottant `t1` et `t2` ainsi que la plus petite valeur. Les deux tableaux ont le même nombre d'éléments. Vous devrez utiliser la structure `ExtremeTableau_s` pour renvoyer la plus petite valeur et la plus grande valeur. Chaque structure doit contenir le type d'extrême, la valeur en question, la position dans le tableau et la référence vers le tableau en question. Partez du prototype suivant :

```
void maxmin(float *t1, float *t2, int NbElement, ExtremeTableau_s *max,
ExtremeTableau_s *min);
```

10. Recherche de motif ADN

Écrivez une fonction qui prend en paramètre un tableau de caractère. Vérifiez, en premier lieu, que le tableau correspond à une chaîne ADN valide (contient uniquement les lettres `A`, `C`, `G` et `T`). Dans un deuxième temps, vérifiez si le motif décrit dans un second tableau de caractère est présent dans la chaîne. Retourner `true` si le motif existe. Retournez `false` si le motif n'existe pas ou si la chaîne d'ADN n'est pas valide.

N'oubliez pas d'appliquer les principes du *Clean Code* et de vérifier si le motif est une chaîne ADN valide. Votre fonction doit utiliser uniquement des pointeurs.

Tableaux multidimensionnels

11. Somme 2D

Écrivez une fonction qui fournit en retour la somme des valeurs d'un tableau de nombre flottant à deux indices dont les dimensions sont fournies en argument. Utilisez le formalisme des pointeurs.