Cahier de TD n° 3

Cahier de TD n° 3

THÈME 1 : MANIPULATIONS ET INITIALISATION DE POINTEURS

- Vrai/Faux
- Instructions correctes et incorrectes : adresses et pointeurs
- Partage de contenu
  - Variables liées
  - Que fait ce programme ?
  - un programme un peu tordu.

ARITHMÉTIQUE DES POINTEURS

THÈME 2 : ALLOCATION DYNAMIQUE, POINTEURS ET TABLEAUX

- Notations * et notation []
- inversion de tableau avec des pointeurs
- Allocation de tableau à 1 dimension
- Tableaux de caractères et allocation : commande de duplication de texte

Thème 1 : manipulations et initialisation de pointeurs

Vrai/Faux

Répondez par vrai ou faux aux affirmations suivantes :

- Une variable peut ne pas avoir d'adresse
- Soit x une variable d'un type simple (caractere, entier ou reel). La notation &x a un sens.
- Soit x une variable d'un type simple (caractere, entier ou reel). La notation *x a un sens.
- & signifie : adresse de
- Soit x un pointeur. Alors, x est une variable.
- Soit x un pointeur. On peut toujours utiliser la notation *x dans un programme.
- Une variable peut avoir plusieurs adresses
- signifie : pointeur
- signifie : contenu de

Instructions correctes et incorrectes : adresses et pointeurs

Dans le programme suivant, indiquez quelles instructions sont incorrectes et pourquoi. Attention ! toutes les instructions sont correctes syntaxiquement (c'est à dire qu'elles sont bien écrites), donc un compilateur acceptera ce programme, mais le résultat aboutit souvent à un plantage du programme !

```
int main()
{
      long val1, val2;
      long *p_ent1, *p_ent2;

      val1 = 5;
      val2 = -125;

      p_ent1 = p_ent2;
      *p_ent1 = val2;
      p_ent2 = &val1;
      *p_ent2 = val2;
      *p_ent1 = *p_ent2;
      p_ent1 = p_ent2;
      *p_ent1 = val1;
      val2 = *p_ent2;
}
```

Si l'on supprime du programme toutes les instructions incorrectes, quelles valeurs prendront les différentes variables ?

Partage de contenu

Variables liées

Ecrire un programme définissant une variable de type `caractere`, deux pointeurs vers des `caractere`, et qui fait en sorte que si l'on modifie le contenu de l'un des deux pointeurs ou la valeur de la variable, alors les contenus des deux pointeurs et de la variable prennent tous la même valeur, sans utiliser de test ni de boucle.

Faites une illustration des contenus et des valeurs des variables utilisées pour bien matérialiser l'effet des instructions utilisées.

Que fait ce programme ?

Indiquez les valeurs prises par les variables de ce programme :

```
int main()
{
      double a,b,c;
      double *p_a, *p_b;

      a = 0.001;
      b = 0.003;

      p_a  = &a;
      *p_a = *p_a * 2.0;
      p_b  = &b;
      c    = 3.0 * (*p_b - *p_a);
}
```

Un programme un peu tordu.

Notre ami Gilbert, toujours absent en amphi, a décidé d'écrire un programme, qui est correct, mais qui est vraiment tordu. Que fait ce programme ? Ecrivez-le de manière plus simple (en 4 à 5 lignes, normalement, c'est possible…)

```
int main()
{
      long a,b;
      long *ptr, *qtr;

      printf("entrez une valeur positive:");
      scanf("%ld",&a); // on suppose que la valeur entree est
                       // positive
      b = 0;

      ptr = &a;
```

```
        qtr = ptr – (&a - &b);

        while (*qtr < *ptr)
        {
              *qtr = *qtr +1;
        }

        printf("et voilà, b =%ld\n",b);
}
```

Arithmétique des pointeurs

Rappelez les effets des instructions du programme suivant (on suppose que les variables concernées sont situées à des adresses consécutives en mémoire). Utilisez la représentation avec flèches au besoin. (un réel occupe 8 octets, un pointeur occupe 4 octets).

```
int main()
{
      double val_a, val_b, val_c;
      double *pdoub;
      double *qdoub;

      val_a = 0.0;
      val_b = 3.1415;
      val_c = 1.E-50;

      qdoub = &val_b;
      pdoub = qdoub;

      pdoub = p_doub-1;

      printf("%ld\"n,qdoub-pdoub);

      qdoub = qdoub+1;

      printf("%lf\n",*qdoub);

      *qdoub =val_a;

      *pdoub = (*pdoub)+1;

      afficher("%ld - %lf\n",qdoub-pdoub,*pdoub);


      *qdoub = *pdoub;
      pdoub = pdoub+1;

      printf("%ld %lf %lf\n",qdoub-pdoub, *pdoub, *qdoub);

}
```

Thème 2 : Allocation dynamique, pointeurs et tableaux

Notations * et notation []

Soit le programme suivant :

```
programme pointeurs_et_tableaux

      char a[9] = {12, 23, 34, 45, 56, 67, 78, 89, 90};
      char *p;
      p = a;
```

Quelles valeurs ou adresses fournissent ces expressions ? On supposera que le tableau `a` est stocké à l'adresse 3000 en mémoire de l'ordinateur. Un caractère occupe un octet.

```
      a)    *p+2
      b)    *(p+2)
      c)    &p+1
      d)    &a[4]-3
      e)    a+3
      f)    &a[7]-p
      g)    p+(*p-10)
h)    *(p++*(p+8)-a[7])
```

(énoncé tiré de : http://www.ltam.lu/Tutoriel_Ansi_C/prg-c97.htm#Heading206)

Inversion de tableau avec des pointeurs

Ecrire un programme qui range les éléments d'un tableau `tab` du type `entier` dans l'ordre inverse. Le programme utilisera des pointeurs `p1` et `p2` et une variable `temp` pour la permutation des éléments.

Allocation de tableau à 1 dimension

Ecrire un programme qui saisit une valeur entière *n* et alloue une zone de mémoire pour stocker *n* entiers, et l'affecte à un pointeur. Quelle est la taille maximum de ce tableau dynamique ? quelle est la taille utile de ce tableau dynamique ?

Ecrire un programme qui effectue la saisie de n valeurs entières et les range dans la zone mémoire obtenue, que l'on peut considérer comme un tableau :

en utilisant la notation [] des tableaux

en utilisant les notations avec des pointeurs

Ecrire un programme qui recherche une valeur dans ce tableau, en utilisant les notations des pointeurs. (pas de crochets [] donc !).

On veut ajouter une nouvelle valeur à ce tableau. A quel problème se trouve-t-on confronté ? Trouvez une méthode permettant, à partir du premier tableau (c'est à dire la zone mémoire obtenue par reservation()), de créer un tableau ayant une variable de plus dans laquelle on stockera la nouvelle valeur.

De la même manière, traitez le cas de la suppression d'une valeur d'un tableau, la valeur étant donnée par une saisie de l'utilisateur.

Tableaux de caractères et allocation : commande de duplication de texte

On cherche à programmer la commande de copie de texte d'un tableau vers un autre tableau en utilisant **d'abord des indices puis des pointeurs.**

On dispose d'un **tableau statique** de caractères (de taille maximum 100), et l'on souhaite dupliquer ce tableau dans un autre texte, dont **la longueur correspond exactement au nombre de caractères à stocker (donc tableau dynamique**, sans utiliser la commande `strcpy`.