Cahier de TD n°3

Séance 9 : 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 utiliser sans risque 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 !

```
programme adr_pointeurs

entier val1, val2;
entier *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 ?

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…)

```
programme super_gilbert_pointeurs

entier a,b;
entier *ptr, *qtr;

afficher("entrez une valeur positive:");
saisir(a); // on suppose que la valeur entree est positive
b ← 0;

ptr ← &a;
qtr ← ptr - (&a - &b);

tant que (*qtr < *ptr)
{
     *qtr ← *qtr +1;
}

afficher("et voilà, b =",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).

```
programme arith_pointeurs


reel val_a, val_b, val_c;
reel *pdoub;
reel *qdoub;

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

qdoub ← &val_b;
pdoub ← qdoub;

pdoub ← p_doub-1;

afficher(qdoub-pdoub,"\n");
```

```
qdoub ← qdoub+1;

afficher(*qdoub,"\n");

*qdoub ←val_a;

*pdoub ← (*pdoub)+1;

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


*qdoub ← *pdoub;
pdoub ← pdoub+1;

afficher(qdoub-pdoub," ", *pdoub," ", *qdoub,"\n");
```

Séance 10 : Allocation dynamique, pointeurs et tableaux

Notations * et notation []

Soit le programme suivant :

```
programme pointeurs_et_tableaux

caractere a[9] ← {12, 23, 34, 45, 56, 67, 78, 89, 90};
caractere *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)

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 ?

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.

Le triangle de Pascal

Le triangle de Pascal est un tableau particulier qui stocke un ensemble de coefficients entiers que l'on appelle également : coefficient du binôme.

Dans sa ligne *i*, ce tableau stocke tous les coefficients de la forme développée de la formule : $(a+b)^i$

Exemples :

$(a+b)^0 = 1$: coefficient : 1

$(a+b)^1 = 1.a + 1.b$: coefficients 1, 1

$(a+b)^2 = 1.a^2+2.ab+1.b^2$: coefficients 1, 2, 1

En prenant les exemple des puissances 3 et 4, écrivez le triangle de Pascal et établissez une relation simple entre les coefficients permettant de passer d'une ligne à l'autre.

Avec des tableaux statiques

Ecrire un programme qui calcule et affiche les coefficients du triangle de Pascal jusqu'à un certain rang *n* en utilisant un tableau statique.

Avec des tableaux dynamiques

Ecrire un programme qui calcule et affiche les coefficients du triangle de Pascal jusqu'à un certain rang *n* en utilisant un tableau dynamique qui n'utilise que la mémoire strictement nécessaire à son stockage.

En fait, le coefficient situé à la colonne *j* de la ligne *i* correspond au nombre de tirages possibles de *j* éléments (sans remise) parmi *i*. Jusqu'à quel rang devriez-vous calculer le triangle de pascal pour obtenir le nombre de combinaisons possibles de tirage au loto ?

Ce nombre est également donnée par : i!/ [(i-j)! × j!] : écrivez un programme qui calcule cette quantité en essayant de simplifier la formule. De toute manière, un ordinateur ne peut pas calculer une factorielle supérieure à 25 !. Le programme que vous écrivez devra être capable de calculer 49 ! / [(49-6)! × 6!]