

TD 2: Pointeurs et structures

Algorithmie

Master EEA 1^{ère} année 2006

1. Introduction

1.1. Edition du fichier

Afin d'écrire votre script C, vous pouvez utiliser n'importe quel éditeur de texte, cependant xemacs ou nedit paraissent être un bon choix. On lance l'éditeur à partir d'une console (« nedit & »). Le fichier aura une extension .c.

1.2. Compilation

Ouvrez un terminal et placez-vous dans le répertoire dans lequel vous avez enregistré votre script C. Afin de compiler votre script, utiliser la commande :

```
gcc nom_du_fichier.cpp -o nom_executable
```

Ce qui donne, par exemple, si votre script du exo1 se nomme exo1.cpp et que vous voulez que votre exécutable se nomme exo1:

```
gcc exo1.cpp -o exo1
```

Le compilateur génère alors un fichier exo1 que vous pouvez exécuter en utilisant la commande ./exo1

2. Exercice 1

```
float value;  
value = 10;
```

```
cout << "value = " << value << endl ;      (1)
```

```
cout << "&value = " << &value << endl ;    (2)
```

1. Que représente value ? Que va afficher la ligne 1 ?
2. Que représente &value ? Que va afficher la ligne 2 ?

3. Exercice 2

```
float *value;
```

1. Comment allouer la mémoire de value ?
2. Que représente &value ?
3. Que représente value ?
4. Que représente *value ?

4. Exercice 3

```
#include <iostream>
using namespace std;

struct point {
    int x;
    int y;
};

void init (int *a, int *b);

int main(int argc, char* argv[])
{
    struct point p, *pp;
    ...
}

void init (int *a, int *b) {
    *a = 10 ;
    *b = 20 ;
}
```

1. Comment afficher la valeur de x de p ?
2. Comment afficher l'adresse mémoire dans laquelle est stockée la valeur de y ?
3. Comment allouer l'espace mémoire de pp ?
4. Comment afficher la valeur de x de pp ?
5. Comment afficher l'adresse mémoire de pp ?
6. Comment afficher l'adresse mémoire de y ?

5. Exercice 4

On n'utilisera que des pointeurs dans cet exercice.

Un pointeur est un espace mémoire pouvant contenir une adresse mémoire. On dit que cette adresse mémoire pointe sur une valeur d'un type donné (défini le nombre de bits à lire).

1. Créer un fichier « exo4.cpp » dans lequel on va calculer le prix d'un article après déduction (en pourcentage) d'une remise et votre profit selon la marge.
On aura comme pointeurs : *prixInitial, *prixSolde et *profit.
On utilisera malloc pour initialiser au moins un des pointeurs (« malloc.h »).
On fixe la remise à 20%, et la marge à 5%.
2. Afficher prix_solde, &prix_solde et *prix_solde. Que représentent ces valeurs ?
3. Pouvez-vous reprendre la fonction « prix_solde » du TD1 (variables) pour calculer le prix soldé ? Testez-la.
4. Est-il possible de créer une fonction « calcul_solde_et_marge » ?
Si oui, afin de diminuer les erreurs de codage, il est conseillé de faire appel à la fonction « calcul_remise »
Afficher l'adresse mémoire des pointeurs passés en paramètre à l'extérieur et à l'intérieur de cette fonction. Que constatez-vous ?

6. Exercice 3

Ajouter un pointeur *nom. Ce pointeur est de type char, il contient dans un premier temps le nom de l'article vendu, par exemple « pantalon ».

Créez une fonction « ajoutSolde (char *label) » qui va rajouter « en solde » à *nom.

Pour rappel, les chaînes de caractères se terminent par « \0 », et la fonction de réallocation mémoire est « realloc() ».