TD2. Pointeurs et tableaux

Dimitri Galayko

1 Pointeurs

Exo 1.1. Considérez les déclarations suivantes et expliquez quel est l'objet défini par chaque ligne.

```
int *pi1;

double a;

double *pd1=&a;

int *ac={1, 2, 3};

int bc[]={1, 4, 10, 11};

int *pi2=ac+2;

int ad[3][2]={{1, 2}, {1, 4}, {5, 6}};

int  *pi3=ad[1];
```

Exo 1.2. On continue le programme commencé par le code de la question précédente.

Analyser le code et commentez le. Détectez les cas où une erreur ou un comportement incorrect se produira...

```
*pi1=1;

*pi1=&a;

pi1[0]=2;

printf("%d\n", a);

pi1[1]=3;

bc[3]=-25;

printf("%d\n", pi2[-1]);

printf("%d\n", pi2[3]);

printf("%d\n", pi3[1]);

printf("%d\n", *(pi3+2));
```

2 Utilisation des pointeurs dans les expressions

Une chaîne de caractère est définie de la façon suivante :

```
char str[100];
printf("Entrez une phrase pas très longue : ");
scanf("%s", str);
```

On souhaite analyser la chaine caractère par caractère. Par exemple, on souhaite connaître le nombre d'occurrence de la lettre 'o'.

On peut utiliser le parcours du tableau d'une manière élégante :

```
i=0;
```

```
char c; // variable intermédiaire
int counter=0; // compteur d'occurrences

while(c=str[i++])
if (c=='o') counter++;
printf("Le nombre d'occurences
 de 'o' est %d", counter);
```

Analysez et commentez le code.

3 Tableau de chaînes de caractère

On déclare un tableau de chaînes de caractère comme suit :

char tab_str[][LENGTH_MAX]={"stalker", "writer", "scientist", "zona"};

Expliquez comment les données sont stoquées dans la mémoire.

Ecrivez deux fonctions. La première affichera les chaînes de caractères du tableaux. L'autre affichera *tous les caractères* du tableau bidimentionnel tab_str. On affichera "\0" si une cellule de mémoire contient une valeur 0.

Les deux fonctions prendront le tableau en argument. Aucune variable globale ne doit être utilisée, sauf les constantes désignant les dimensions du tableau.

Expliquez la structure de l'expression de la condition de poursuite de boucle.

Exo 3.1. Soit tab est un tableau de 10 entiers. Expliquez pourquoi le programme suivant génère une erreur :

```
int i;
for (i=0; tab[i]!=234 && i<=9 ; i++)
{ /* corps de boucle*/}
```

Exo 3.2. Donnez le résultat des opérations :

```
int i=0;
int tab={0,1,2,3}
i++<25\&&i!=0+15|3*-tab[i|2]

int i=0;
int tab={0,1,2,3}
*&tab[2]<<1**tab;
```

4 Applications

Exo 4.1. Ecrivez une fonction qui prend en argument les trois coefficients d'une équation quadratique et qui retourne ses racines réelles.

Le problème : on connaît pas d'avance le nombre des racines – il dépend du signe de déterminant. Ainsi, la fonction doit retourner un nombre d'argument variable. Pour cela, vous écrirez une fonction qui retournera un tableau des doubles, dont la premier élément donnera le nombre des racines, et les autres éléments, éventuellement, les valeurs des racines. Ainsi, selon le cas, vous aurez un tableau d'un, deux ou trois éléments.

Le prototype de la fonction s'écrit :

double * racines(double a, double b, double c)

Exo 4.2. Ecrivez les fonctions suivantes qui permettent de manipuler des chaînes de caractères:

int str_length(char* s)

retourne la longueur d'une chaîne de caractères;

void str_copy(char* str1, char * str2)

copie la chaine str1 vers la chaîne str2.

int str_substring(char* str1, char * str2)

retourne 1 si la chaîne str1 est une sous-chaine de str2.