UNIVERSITÉ PARIS DESCARTES

DUT Informatique - 1ère année
SDA - Structures de Données et Algorithmes

TRAVAUX PRATIQUES – Semaine n°1

Thèmes

- Les entrées-sorties en Langage C++
- Algorithmique

Exercice 1. Affichage – Taille et domaine de variation des types

Comme vous l'avez déjà écrit pour le langage C, écrivez le programme qui affiche pour chaque type natif du langage C++, la taille mémoire (en nombre d'octets) prise par une variable de ce type et le domaine de validité de ce type. Le domaine est caractérisé par une borne minimale et une borne maximale. Ces bornes sont définies (par des macro instructions) dans le fichier d'entête `climits`. Les constantes que vous devez employer pour chacun des types sont les suivantes :

Type	*Borne min.*	*Borne max.*
`char`	`CHAR_MIN`	`CHAR_MAX`
`unsigned char`	*Non définie*	`UCHAR_MAX`
`wchar_t`	*Non définies*	
`short`	`SHRT_MIN`	`SHRT_MAX`
`unsigned short`	*Non définie*	`USHRT_MAX`
`int`	`INT_MIN`	`INT_MAX`
`unsigned int`	*Non définie*	`UINT_MAX`
`long`	`LONG_MIN`	`LONG_MAX`
`unsigned long`	*Non définie*	`ULONG_MAX`
`float`	*Non définies*	
`double`	*Non définies*	

Votre programme devra provoquer l'affichage suivant :

```
Types : coût-mémoire (en octets) et domaine de variation
Bool : 1 - false .. true
char : 1 - Domaine : -128 .. 127
unsigned char : 1 - Domaine : 0 .. 255
wchar_t : 2
short : 2 - Domaine : -32768 .. 32767
unsigned short : 2 - Domaine : 0 .. 65535
int : 4 - Domaine : -2147483648 .. 2147483647
unsigned int : 4 - Domaine : 0 .. 4294967295
long : 4 - Domaine : -2147483648 .. 2147483647
unsigned long : 4 - Domaine : 0 .. 4294967295
float : 4
double : 8
```

- Le fichier d'entête à inclure est `climits`.
- Les instructions d'affichage doivent employer l'opérateur d'insertion (`<<`) dans le flot de sortie standard (`cout`) vu en cours.
- Le type `bool` doit être pris en compte (ses bornes minimales et maximales ne sont pas définies dans `climits`). Affichez les deux constantes (`false` et `true`) du type. Vous obtiendrez les deux constantes en alphanumérique avec l'inclusion `<iomanip>` et en insérant dans le flot de sortie standard `boolalpha` avant d'afficher la première constante.

Exercice 2. Affichage – Table ASCII

Le code ASCII établit une correspondance entre un entier naturel (codé sur un octet) et un caractère. Le code ASCII établit cette correspondance sur la plage 0..127 ; de 128 à 255, on parle de code ASCII étendu. Les 31 premiers caractères ne sont pas affichables. Ecrivez un programme en C++ qui affiche la table ASCII pour les entiers de 32 à 255.

- Vous pourrez utiliser la conversion explicite d'un entier en caractère pour l'affichage demandé. Exemple : `(char) 33` donne le caractère ' !'
- Affichez la table avec 8 caractères par ligne sous le format suivant :

```
32->' '.. 33->'!'.. 34->'"'.. 35->'#'.. 36->'$'.. 37->'%'.. 38->'&'.. 39->'''..
```

- L'entier (i) sera cadré à droite sur 3 caractères. Vous utiliserez les manipulateurs de flots avec l'inclusion suivante : `#include <iomanip>` Exemple d'utilisation des manipulateurs pour un cadrage à droite (`setiosflags(ios::right))et un affichage` sur une largeur de 3 caractères (setw(3)) :

```
cout << setiosflags(ios::right) << setw(3) << i;
```

Repérez les plages de contigüité de codage des chiffres et des lettres de l'alphabet (minuscules et majuscules).

Exercice 3. Implémentation d'algorithmes

On implémentera les algorithmes donnés en langage pseudo-naturel et on comparera leur efficacité aux solutions étudiées au TD n°1.

3.1. ALGORITHME DE LA FONCTION `palindrome`

Il est facile de décider si un nombre est un palindrome en testant si le premier chiffre a la même valeur que le dernier, puis si le second a la même valeur que l'avant dernier et ainsi de suite. Pour pouvoir accéder à la valeur de chaque chiffre, vous pouvez préalablement construire une table contenant chacun d'eux en suivant l'algorithme suivant : soient n un entier, t un tableau, i un indice.

```
i ← 0 ;
tant que (n > 0) faire
     t[i] ← n modulo 10 ;
     i ← i + 1 ;
     n ← n / 10 ;
fin tant que
```

Notez bien qu'à la fin de l'algorithme, `i` indique le nombre de chiffres et que les unités sont à l'indice 0 du tableau `t`, les dizaines à l'indice 1, etc. Notez aussi la façon dont est traité le cas particulier où `n` est égal à 0.

Comparez ce nouvel algorithme par rapport à celui à l'ancien. Programmez-le et testez-le. Utilisez le débogueur si nécessaire.

3.2. ALGORITHME DE LA FONCTION `miroir`

L'algorithme permettant d'extraire les chiffres d'un nombre peut aussi être employé dans la fonction `miroir`. En effet, il permet d'obtenir l'équivalent de la chaîne inverse du nombre en un parcours (alors qu'il en faut deux dans la version actuelle de `miroir`). La construction du miroir peut être obtenue en multipliant chaque chiffre par la puissance de 10 qui lui correspond et en sommant le tout. C'est le rôle de l'algorithme suivant : soient `t` un tableau produit par l'algorithme précédent, `i` le nombre de chiffres présents dans `t` et `r` et `p` deux variables entières.

```
r ← 0 ;  p ← 1 ; i ← i - 1 ;
tant que (i >= 0) faire
     r ← r + (p * t[i]) ;
     p ← p * 10 ;
     i ← i - 1 ;
fin tant que
```

Notez que le résultat est stocké dans `r`. Programmez cette nouvelle version de la fonction `miroir` et testez-la. Utilisez le débogueur si nécessaire.