TP NUMERATION

Compte-rendu informatique à rendre

Objectifs du TP :

- ***vérifier les notions vues en cours et en TD sur le codage de données de différents types ;***
- ***être capable de choisir les types des données manipulées par un programme.***

Rappel des 5 règles de codage :

- Mettre un entête
- Mettre des commentaires
- Respecter l'indentation
- Choisir des noms de variables explicites
- Choisir des noms de fichiers explicites

Espace d'adressage d'un programme :

Quand il exécute un programme, le système accède aux informations en mémoire (instructions et données) au travers d'un ensemble d'adresses (constituant **l'espace d'adressage**).

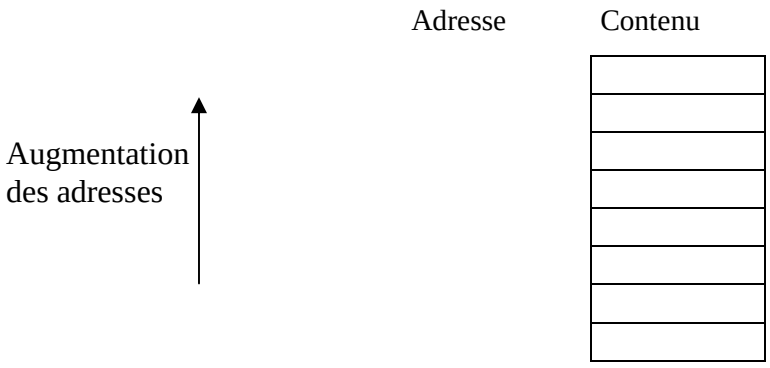

Fig 1 : Représentation de l'espace d'adressage d'un programme

Une **case mémoire** contient un **octet** (chaque octet de la mémoire possède sa propre adresse).

Par exemple, sur une machine à adressage sur 32 bits, 2^{32} adresses sont disponibles entre 0 et $2^{32} - 1$. Donc une machine 32 bits est capable d'adresser 2^{32} octets différents, alors qu'une machine 64 bits est capable d'en adresser 2^{64}.

Notions d'adresse et de valeur d'une variable :

Une déclaration d'une variable **réserve un emplacement dans la mémoire (i.e. une ou plusieurs cases mémoires)** pour cette variable. Cet emplacement est repéré par une **adresse** choisie par le système. On ne peut pas la modifier.

Une affectation dans une variable modifie la **valeur** stockée dans celle-ci : **les cases mémoire réservées pour cette variable** sont **modifiées**.

Les opérateurs sizeof et & en langage C++

L'opérateur unaire sizeof renvoie la dimension en octets de l'espace mémoire occupé par l'opérande.
L'opérateur unaire & renvoie l'adresse de l'opérande.

Exemple d'utilisation de ces opérateurs :

```
int n;            // declaration de la variable de nom n et de type int
cout<<"Adresse de n:"<<&n<<endl;    // affichage de l'adresse de la
                                    // variable n
cout <<"Nombre d'octets occupes par la variable n:"<<sizeof(n)<<endl;
                        // affichage de la dimension de l'espace
                        // mémoire occupé par la variable n
```

Documents à rendre :

- Un listing de **l'ensemble du programme** réalisé pour répondre aux questions ;
- Un compte rendu réalisé avec Libre Office Writer.

TRAVAIL A REALISER :

1. Etude du type unsigned short int

- déclarer une variable du type unsigned short int, relever son adresse et le nombre d'octets qu'elle occupe ;
- réaliser le programme permettant de compléter le tableau suivant :

Valeur affectée	Valeur en mémoire	Valeur affichée (*)
54		
128		
	0x2A5E	
68000		
- 10		

(*) : sur la console

- Quel est le système de codage utilisé par le type unsigned short int ?
- Quel est le format (système de numération) des données affichées sur la console ?
- Quelles sont les valeurs minimale et maximale d'un entier de type unsigned short int ? (justifier)
- Justifier les valeurs en mémoire des 2 premières lignes.
- Vérifier la correspondance entre les valeurs affichées et les valeurs en mémoire pour les 2 dernières lignes.
- Convertir en hexadécimal la valeur décimale 68000. Conclusion ?

2. Etude du type short int

- déclarer une variable du type short int, relever son adresse et le nombre d'octets qu'elle occupe ;
- **compléter** votre programme afin de remplir le tableau suivant :

Valeur affectée	Valeur en mémoire	Valeur affichée
- 63		
47		
	0x2A5E	
	0xA9EF	

- Quel est le système de codage utilisé par le type short int ?
- Quelles sont les valeurs minimale et maximale d'un entier de type short int ? (justifier)
- Justifier les valeurs en mémoire des 2 premières lignes.

3. Etude des types réels

3.1 Précision des variables de type float et double

- déclarer une variable du type float, relever le nombre d'octets qu'elle occupe ;
- déclarer une variable du type double, relever le nombre d'octets qu'elle occupe ;
- déterminer expérimentalement la précision de ces variables.

Remarque : l'instruction cout << setprecision(20); permet de fixer à 20 chiffres la précision d'**affichage** des nombres réels.

3.2 Etude du type float

Le type float utilise la norme IEEE754 pour coder des nombres réels sur 32 bits.

- Faire une recherche sur Internet pour comprendre la norme IEEE754;
- en utilisant une variable de type float, **compléter** votre programme afin de remplir le tableau suivant :

Valeur affectée	Valeur en mémoire	Valeur affichée
8.5		
-26.5625		
	0x2E310000	
	0xD1870000	

- Justifier les valeurs en mémoire des 2 premières lignes.