Représentation interne des entiers PARTIE 2

4.Déclaration de variables (indiquer la grosseur du contenant)

Comment fait-on pour indiquer le nombre d'octets dont chaque variable a besoin ?

On ajoute une section au début du programme : la déclaration de variables.

Programme : DÉCLARATION DE VARIABLES

Aller chercher des données (lecture)

Traitement

Rendre disponible les résultats (affichage ou impression)

En plus d'indiquer le nombre d'octets nécessaires pour chaque variable, la déclaration de variables permet d'indiquer le type d'information qui sera emmagasinée dans ces octets (entier, nombre à virgule…)

Toujours avec le même exemple (somme et différence entre 2 nombres) :

Déclaration de variables :

A : entier, 1 octet

B : entier, 1 octet

SOMME : entier, 1 octet

DIFF : entier, 1 octet

Début des instructions :

Lire A,B

SOMME = A + B

DIFF = A – B

Afficher SOMME, DIFF

Effet en RAM :

	Déclarations
	A : entier, 1 octet
	B : entier, 1 octet
	SOMME : entier, 1 octet
	DIFF : entier, 1 octet
	Début du programme
	Lire A,B
	SOMME = A + B
	DIFF = A – B
	Afficher SOMME, DIFF
A	
B	
SOMME	
DIFF	

Ça fonctionne si A et B sont petits, mais si on veut que ça fonctionne pour des valeurs de A et B plus grandes, on pourrait par exemple réserver 4 octets pour les valeurs

A,B,SOMME,DIFF : entier, 4 octets

Rappel : valeurs possibles sur 4 octets (32 bits) = de -2^{31} à $2^{31}-1$ = de -2 147 483 648 à 2 147 483 647 (voir le fichier Représentation interne des entiers Partie 1, page 9)

Effet en RAM :

	Déclarations
	A : entier, 4 octets
	B : entier, 4 octets
	SOMME : entier, 4 octets
	DIFF : entier, 4 octets
	Début du programme
	Lire A,B
	SOMME = A + B
	DIFF = A – B
	Afficher SOMME, DIFF
A	
B	

SOMME	
DIFF	

Lien entre les notions vues dans le cours de C12 (Outils et matériels) et le cours de programmation (C11).

Notion vue en C12	Type correspondant lors de la déclaration de variables au début du programme
Entier 8 bits	char
Entier 16 bits	short
Entier 32 bits	int
Entier 64 bits	long

À la page suivante, vous voyez comment se code notre exemple en C :

Rappel : notre exemple :

Lire A,B

SOMME = A + B

DIFF = A – B

Afficher SOMME, DIFF

Programme en C :

```
#include <iostream>

using namespace std;

int main()

{

        // Définitions des variables

        int a, b, diff, somme;


        // Initialisations des variables

        a = 0;

        b = 0;


        // Lecture (sans traitement des erreurs)

        cout << "lecture du nombre #1: ";

        cin >> a; // lire a

        cout << "lecture du nombre #2: ";

        cin >> b; // lire b


        // Traitement

        somme = a + b;

        diff = a - b;


        // Affichage

        cout << "Somme: " << somme << ", Différence: " << diff << endl;

        system("pause");

}
```

Il existe en C d'autres autres types, par exemple :

1.Les entiers non signés

Type qui correspond seulement à des entiers positifs.

Représentation interne : mettre en binaire la valeur en base 10 et ajouter des zéros non significatifs si nécessaire.

Il n'y a **PAS** de notion de bit de signe avec ce type (on représente seulement des positifs)

Exemple1 : donner la représentation interne 8 bits de 48 (entier non signé) :

$48_{10} = 110000_2$ = 00110000 sur 8 bits

Exemple 2 : donner la représentation interne 8 bits de -64 (entier non signé) :

Impossible, car seulement les entiers positifs peuvent être représentés en entier non signé (il n'y a pas de place pour un bit de signe dans cette représentation, toutes les valeurs sont positives)

Chemin inverse :

Exemple : les valeurs suivantes représentent des entiers contenus dans des variables en RAM. Ce sont des entiers non signés. Si on demande à l'ordinateur d'afficher ces contenus, que sera-t-il affiché ?

a) $1\,1\,0\,1\,1\,1\,0\,0_2$
b) $0\,1\,0\,1\,0\,1\,0\,1_2$

a) $1101111_2 = 220_{10}$
Le nombre affiché sera 220

b) $01010101_2 = 85_{10}$
Le nombre affiché sera 85

Limite de représentation pour les entiers non signés :

Prenons un exemple sur 4 bits, voyons les valeurs possibles

0000

0001

0010

0011

0100

0101

0110

0111

1000

1001

1010

1011

1100

1101

1110

1111

Sur 4 bits, je peux représenter $2^4 = 16$ valeurs

Formule générale : sur n bits je peux représenter 2^n valeurs avec les entiers non signés.

Sur 8 bits, je peux représenter $2^8 = 256$ valeurs (de 0 à 255)

Valeurs minimales et maximales sur 8 bits : 0 et 255.

Pourquoi ce type ? (Historiquement) pour économiser de l'espace. Si on sait que notre valeur ne peut être que positive (un compteur, par exemple qui va de 0 à 200), on peut prendre un entier non signé au lieu d'un entier 16 bits.

En C, unsigned char, unsigned short, unsigned int, unsigned long (entier non signé 8,16,32,64 bits)

2.Les booléens

Il est quelquefois utile d'associer une valeur logique à une variable (vrai ou faux).

On utilise alors le type booléen (bool). L'espace mémoire réservé est d'un octet (même si on a en fait besoin d'un seul bit).

Résumé des représentations internes des entiers :

Entiers signés (représentés en RAM avec un bit de signe)

En C : char, short, int, long

Représentation interne :

Entiers positifs :

1. Transformer l'entier en binaire
2. Compléter avec des zéros non significatifs si nécessaire

Entiers négatifs :

1. Transformer la valeur absolue de l'entier en binaire
2. Compléter avec des zéros non significatifs si nécessaire
3. Complémenter (à partir de la droite, recopier les chiffres jusqu'au premier 1 inclusivement et inverser les autres)

Limites de représentation : sur n bits : de -2^{n-1} à $2^{n-1}-1$

Entiers non signés (pas de bit de signe dans la représentation en RAM)

En C : unsigned char, unsigned short, unsigned int, unsigned long

Représentation interne :

Entiers positifs :

1. Transformer l'entier en binaire
2. Compléter avec des zéros non significatifs si nécessaire

Entiers négatifs :

Ne peuvent pas être représentés

Limites de représentation : sur n bits : de 0 à 2^n

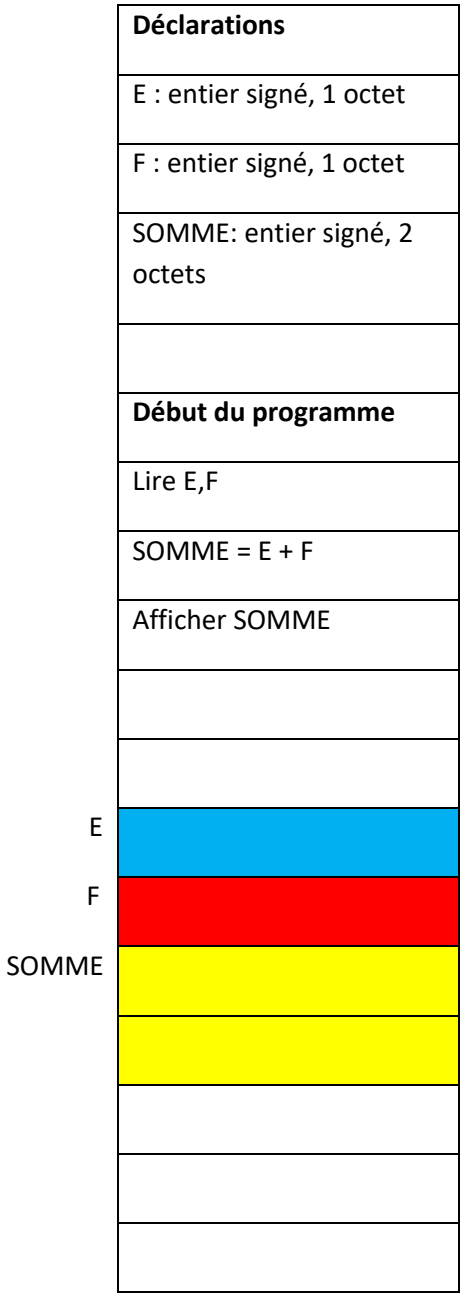
Exercices :

1.Soit le programme suivant en RAM :

Déclarations
E : entier signé, 1 octet
F : entier signé, 1 octet
SOMME: entier signé, 2 octets
Début du programme
Lire E,F
SOMME = E + F
Afficher SOMME

a) Donnez le schéma de la réservation des variables en RAM

Solution :



b) si les valeurs lues pour les variables E et F sont respectivement 6_{10} et -8_{10}, donnez le contenu des variables E, F et SOMME en RAM <u>après l'exécution du programme</u>

Solution :

	Déclarations
	E : entier signé, 1 octet
	F : entier signé, 1 octets
	SOMME: entier signé, 2 octets
	Début du programme
	Lire E,F
	SOMME = E + F
	Afficher SOMME
E	0 0 0 0 0 1 1 0
F	1 1 1 1 1 0 0 0
SOMME	1 1 1 1 1 1 1
	1 1 1 1 1 1 0

2.Soit le programme suivant en RAM ainsi que le contenu des variables <u>après l'exécution du programme :</u>

	Déclarations
	G : entier signé, 1 octet
	H : entier signé, 1 octet
	SOMME: entier signé, 2 octets
	Début du programme
	Lire G,H
	SOMME = G + H
	Afficher SOMME
	Afficher G,H
G	04
H	F9
SOMME	FF
	FD

a)Quelle sera la somme affichée par le programme ? (en base 10)

b)Quelles seront les valeurs affichées pour les variables G et H ? (en base 10)

Solution :

a) SOMME = $FFFD_{16}$ = 1111111111111101_2
Complément de 1111111111111101_2 = 0000000000000011_2

0000000000000011_2 = 3

Comme la valeur est négative, SOMME = -3

b) G = 04_{16} = 00000100_2 = 4_{10}

H = $F9_{16}$ = 11111001_2
Complément de 11111001_2 = 00000111_2

00000111_2 = -7_{10}

3.Soit la déclaration de variables suivante :

A : entier signé, 4 octets

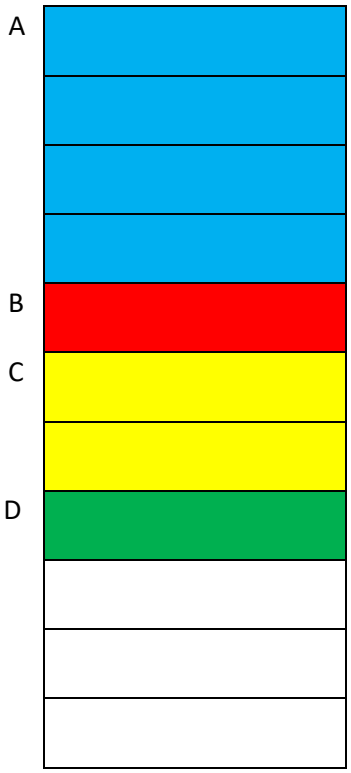
B : entier signé, 1 octet

C : entier signé, 2 octets

D : entier signé, 1 octet

Donnez le schéma de la réservation des variables en RAM

Solution :



5. Soient le programme et les valeurs suivantes en RAM :

	Lire A,B
	SOMME = A + B
	DIFF = A – B
	Afficher SOMME, DIFF
A	08
	02
	B6
	FA
	1B
	00
	14
	78

a) Quelles seront les valeurs affichées par le programme si A et B sont déclarées ainsi : entier signé, 1 octet (on prend pour acquis que A et B sont contigües) ?

Solution :A contient 8_{10} et B contient 2_{10}.

Valeur de SOMME = 10_{10}

Valeur de DIFF = 6_{10}

Ce qui sera affiché : 10 et 6

b) Quelles seront les valeurs affichées par le programme si A et B sont déclarées ainsi : entier non signé, 2 octets (on prend pour acquis que A et B sont contigües) ?

Solution :A contient 2050_{10} et B contient 46842_{10}.
Valeur de SOMME = 48892_{10}
Valeur de DIFF = -44792_{10}
Ce qui sera affiché : 48892 et -44792

c) Quelles seront les valeurs affichées par le programme si A et B sont déclarées ainsi : entier signé, 4 octets (on prend pour acquis que A et B sont contigües) ?

Solution :A contient $134\ 395\ 642_{10}$ et B contient $452\ 990\ 072_{10}$.

Valeur de SOMME = $587\ 385\ 714_{10}$

Valeur de DIFF = $-318\ 594\ 430_{10}$

Ce qui sera affiché : 587 385 714 et -318 594 430

Conclusion :

L'importance de choisir le bon type de variable était plus grande avant dans tous les cas. Il fallait faire des économies de RAM (RAM de plus petite capacité à cause du coût de la RAM et de la technologie pour stocker 1 bit).

Aujourd'hui, c'est encore important dans plusieurs cas, pour des raisons différentes.

Ex : jeux vidéos en ligne, il faut penser à minimiser le trafic sur le réseau. Les variables qui contiennent les caractéristiques des joueurs ne doivent pas être plus grandes que nécessaire.

Par contre, elles doivent être assez grandes :

Ex1 : Jeux vidéos (compteurs de points)

Points : dans une variable de type entier non signé 8 bits.

Type entier :ok

Non signé : ok

8 bits : ???

Nb de points : 0 00000000

1 00000001
2 00000010
3 00000011
4 00000100

…….

253 11111101

245 11111110

255 11111111

256 255 + 1 donc 11111111 + 1 = 1 00000000

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 (255_{10})

\+ 1

---

1 0 0 0 0 0 0 0 0

Et le joueur se retrouvait avec 0 comme total de points accumulés au lieu de 256

Ex2 Gangnam style

https://arstechnica.com/information-technology/2014/12/gangnam-style-overflows-int_max-forces-youtube-to-go-64-bit/

Donc, ayez toujours en tête de programmer avec la plus grande efficacité possible.

Alain va vous donner des exemples concrets des types à employer en programmation.