# Objectifs du TP

L'objectif de la séance est de manipuler le débogueur sous QT Creator et de comprendre :

* comment sont codées les différentes variables en C++,
* comment s’effectuent les opérations entre variables,
* comment s’opèrent les appels de fonctions.

# Travail à réaliser

Une fois connecté sous la machine physique Linux Mint avec le login invite et le mot de passe invite, lancez QT Creator puis répondez aux questions.

# Utilisation du débogueur

Après avoir positionné un point d’arrêt (double clique dans la marge à côté de la ligne souhaitée), vous pouvez lancer le débogage en utilisant la barre des boutons et/ou les raccourcis clavier.



F5 lancer le débogage

F11 avancer pas à pas

## Codage des entiers naturels

1. Sachant qu’un entier court non signé est codé sur 16 bits, donner la plage de valeurs valides en décimal, en binaire et en hexadécimal.

La plage de valeurs valides est [0-65535] en décimal, [0-1111111111111111] en binaire et [0-FFFF] en hexadécimal

1. Soit le programme ci-dessus. Donnez la sortie écran attendue à l’exécution.

« Hello World ! »

« 0 »

« 65535 »

1. Vérifiez en lançant le programme dans Qt creator.
2. A présent, positionnez un point d’arrêt ligne 9 puis lancez le débogage F5 et observez l’évolution de nb au fil de l’exécution en mode pas à pas F11.
3. Notez la valeur de nb avant initialisation. A quoi correspond-elle ?

La valeur de nb avant initialisation est de « 0 » et elle correspond

1. Si ce n’est fait, arrêter le débogueur en cliquant sur le bouton STOP de la barre ad hoc  .*return* 0;
2. Relancez le débogueur et cliquez sur le bouton  pour basculer entre le code C++ et le code assembleur (jeu d'instructions de l'architecture x86 d'Intel et AMD).
3. D’après vous et en modifiant la valeur d’initialisation de nb, à quoi correspondent les 00 00 de la ligne 19 :

0x4015fb <+ 59> 66 c7 45 f6 00 00 movw $0x0,-0xa(%ebp)

Selon moi les 00 00 correspondent à l’initialisation de nb à « 0 »

1. Essayez avec la valeur 256. Quelle remarque pouvez-vous formuler ?

Les 00 00 de la ligne 17 sont passés à 00 01, on déduit qu’ils représentent la valeur binaire et hexadécimal de nb

## Hissez les drapeaux

1. Remettez l’initialisation de la variable nb à la valeur 0.
2. En mode débogueur, affichez les registres à l’aide du menu contextuel (voir ci-dessous).

Le registre eflags (drapeaux) donne les indications suivantes pour chaque bit : voir page Wikipedia <https://fr.wikipedia.org/wiki/RFLAGS>

1. Donnez les valeurs des différents drapeaux après initialisation de la valeur nb (ligne 9).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13…12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | NT | IOPL | OF | DF | IF | TF | SF | ZF | 0 | AF | 0 | PF | 1 | CF |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

1. Donnez les valeurs des différents drapeaux après incrémentation de la valeur nb (ligne 13).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13…12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | NT | IOPL | OF | DF | IF | TF | SF | ZF | 0 | AF | 0 | PF | 1 | CF |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

1. Même question mais en remplaçant nb-- par nb++.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13…12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | NT | IOPL | OF | DF | IF | TF | SF | ZF | 0 | AF | 0 | PF | 1 | CF |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

## Codage des entiers relatifs

1. Modifiez le programme pour travailler sur des entiers relatifs. Quelle est la nouvelle plage de valeurs possible ?

La nouvelle plage de valeurs est [−32768, 32767]

1. Quel résultat s’affiche à l’exécution du programme avec nb++ ?

« Hello World ! »

« 0 »

« 1 »

1. Quel résultat s’affiche à l’exécution du programme avec nb-- ?

« Hello World ! »

« 0 »

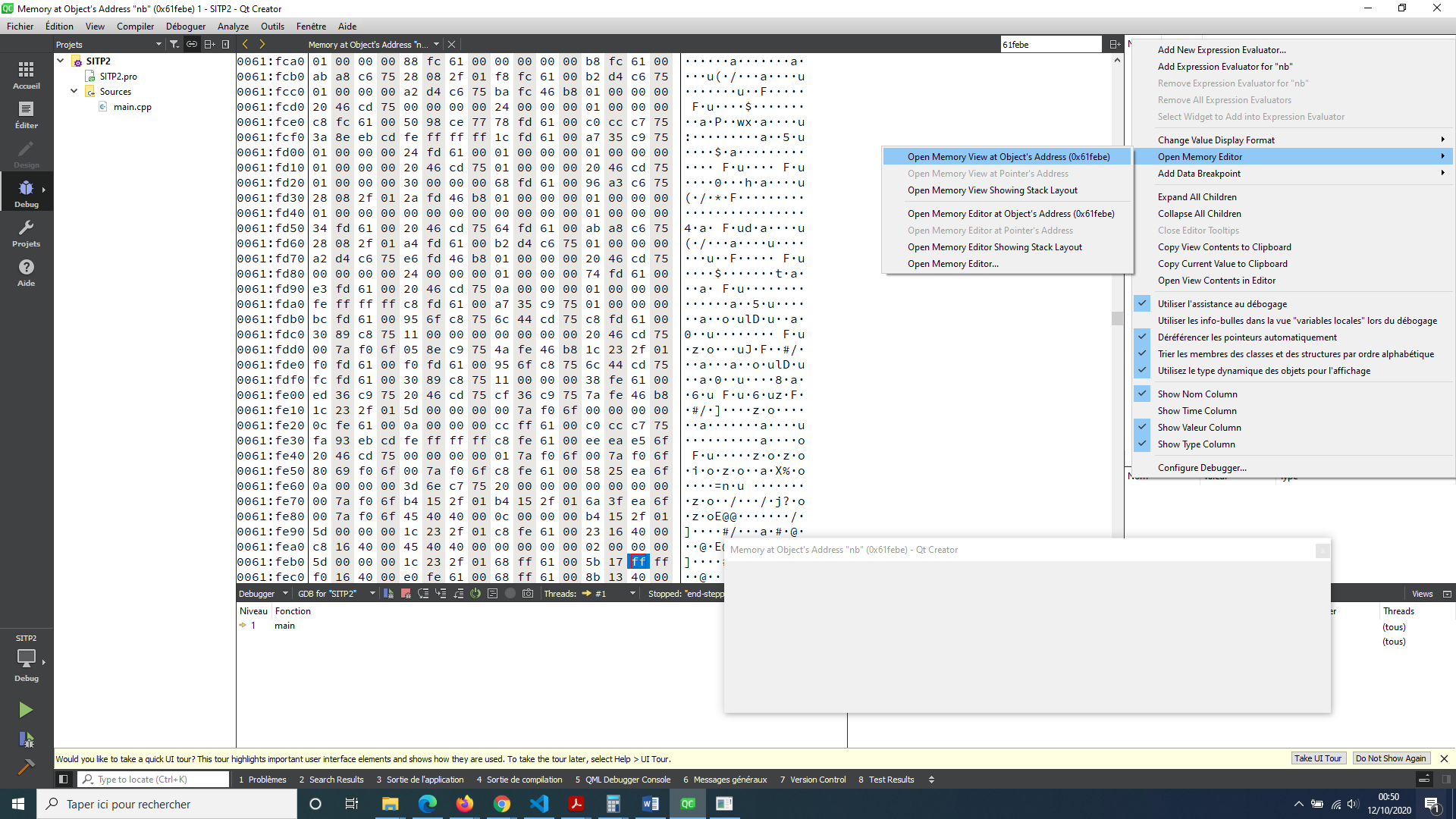
« -1 »

1. Donnez la valeur des flags juste après l’opération et justifiez leur valeur.

La valeur du flag pour n++ est « 202 »(16)

La valeur du flag pour n-- est « 297 »(16)

1. En mode débogueur, affichez une vue de la mémoire pour la variable nb.



1. A l’aide du débogueur, donnez la valeur en hexadécimal et en binaire de -1.

La valeur en hexadécimal de -1 est FFFF et en binaire 1111111111111111

1. Vérifiez en calculant le complément à 2.

**0000 0000 0000 0001(2) = 1(10)**

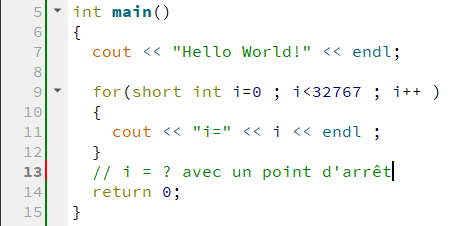
**complément à 1 = 1111 1111 1111 1110**

**complément à 2 = 1111 1111 1111 1110 + 1 = 1111 1111 1111 1111**

1. Soit 2 entiers courts signés nb1 et nb2, on effectue respectivement l’addition et la soustraction des valeurs suivantes. Complétez le tableau en donnant les valeurs attendues et les valeurs affichées ainsi qu’en donnant les valeurs des flags donnés.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| nb1 | nb2 | add=nb1+nb2 | | OF | SF | ZF | PF | CF | sub=nb1-nb2 | | OF | SF | ZF | PF | CF |
| attendue | affichée | attendue | affichée |
| 333 | 666 | 999 | 999 | 0 | 0 | 0 | 1 | 0 | -333 | -333 | 0 |  |  |  |  |
| 33333 | 0 | 33 333 | -32203 | 0 | 0 | 0 | 0 | 1 | 33 333 | -32203 |  |  |  |  |  |
| 32768 | 1 | 32769 | -32767 |  |  |  |  |  | 32767 | -32769 |  |  |  |  |  |
| 16500 | 15500 | 32000 | 32000 |  |  |  |  |  | 1000 | 1000 |  |  |  |  |  |
| 17500 | 16500 | 34000 | 34000 |  |  |  |  |  | 1000 | 1000 |  |  |  |  |  |
| 16500 | 17500 | 34000 | 34000 |  |  |  |  |  | -1000 | -1000 |  |  |  |  |  |
| 32767 | 32767 | 65534 | 65534 |  |  |  |  |  | 0 | 0 |  |  |  |  |  |

## Une petite boucle

Soit le programme suivant :

1. Quelle sera la valeur de i à l’issue de la boucle ?

**32766**

1. Même question avec i<32768 :

i varie de façon infini

1. Vérifiez avec le débogueur. Expliquez.

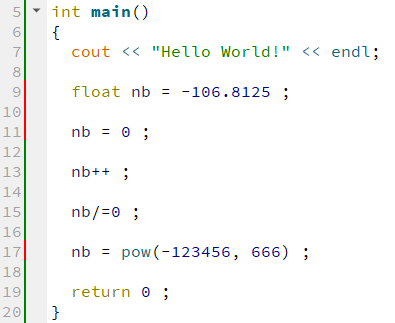
Avec un nombre courts on peut avoir que 32768 possibilités et on commence à 0 donc on cherche à aller jusqu’à 32769 alors que ce n’est pas possible

## Codage des réels norme IEE754

On s’intéresse maintenant aux nombres réels codés au format IEEE754.

1. Sur combien de bits est codé un float ? Rappelez le format.

32 bits soit 4 octets

Soit le programme suivant :

1. Donnez les valeurs de nb en décimal, en binaire et en hexadécimal. Vous pouvez vous aider avec un convertisseur en ligne comme celui-ci : <https://www.h-schmidt.net/FloatConverter/IEEE754.html>.

**1) Hexadécimal : 0xc2d5a000 et Binaire :11000010110101011010000000000000**

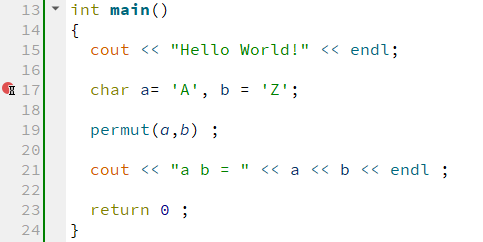
**2) Hexadécimal : 0 et binaire : 0**

**3) Hexadécimal : 0x3f800000 et binaire : 00111111100000000000000000000000**

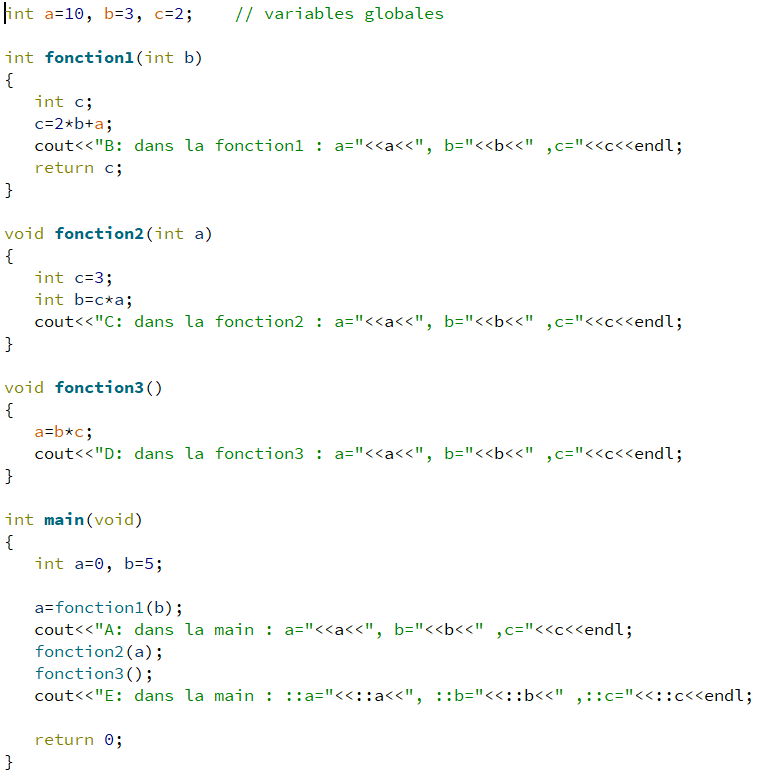
**4) infini / hexadécimal : 0x7f800000 et binaire : 01111111100000000000000000000000**

1. A l’aide du débogueur, vérifiez ces valeurs. Y a-t-il des flags levés pour les différentes affectations ? Pourquoi ?

## Fonctions : passage de paramètres copie VS référence

1. Ecrivez la fonction void permut(char char1, char char2) qui permute les valeurs de char1 et char2.
2. Rappelez la valeur du caractère ‘A’ et ‘Z’ en décimal, en hexadécimal et en binaire.
3. Testez en mode débogueur avec le code suivant et affichez la vue de la mémoire afin de suivre l’évolution des variables a et b.
4. Donnez la sortie écran obtenue :
5. Même question en modifiant la déclaration de la fonction : void permut(char& char1, char& char2). Donnez la sortie écran obtenue. Expliquez la différence avec la réponse précédente.

## Fonctions : passage de paramètres, valeur de retour et variables globales

1. A l’aide du débogueur, suivez le fil d’exécution du code suivant à l’aide du débugueur afin de comprendre la portée des variables.