PROGRAMMATION STRUCTUREE EN C++

TP n° 1 – Ecriture et mise au point d'un programme
Utilisation du débogueur DDD (*Data Display Debugger*)
Le débuggeur est votre ami !

Nombre de séances de TP encadrés : 2

Exercice 1

1. Taper sous l'éditeur de texte *gedit* le programme C++ suivant et le sauvegarder sous le nom *exo1.C*.

```
#include "/usr/local/public/BIBLIOC++/entreeSortie.h"
#include "/usr/local/public/BIBLIOC++/chaine.h"

int main ()
{
    int v ;
    int r ;

    ecrire (uneChaine ("entrer une valeur entiere : ")) ;
    lire (v) ;
    r = 1 ;
    while (v > 1)
    {
        r = r*v ;
        v = v-1 ;
    }
    ecrireNL (r) ;
}
```

Remarque : Pour mettre en gras les mots réservés du langage C++ sous *gedit*, vérifier que le langage C++ est sélectionné.

2. Compiler le programme avec la commande suivante :

```
comp nom_de_fichier <RC>
```

où *nom_de_fichier* désigne le nom du fichier source à compiler sans l'extension.
S'il y a des erreurs de compilation, les modifier et recompiler le programme jusqu'à ce qu'il n'y ait plus aucune erreur.
Vérifier par la commande *ls* l'existence des fichiers *exo1.o* et *exo1.exe*. A quoi correspondent ces deux fichiers ?

3. Exécuter le programme en tapant ***./nom_de_fichier.exe*** (le nom du fichier exécutable).

On exécutera une première fois le programme en saisissant la valeur 3 puis une deuxième fois en saisissant la valeur 6. Que fait le programme ?

4. Le fichier *makefile* et l'utilitaire *make*

a) Création du fichier Makefile

Créer un fichier sous *gedit* appelé *makefile*. Sélectionner dans le menu *Preferences* l'option *Tabs…* et cliquer sur le bouton *Use tab characters in padding and emulated tabs*. Taper ensuite le texte suivant :

```
exo1.exe : exo1.C
    comp exo1
```

La deuxième ligne de ce fichier doit impérativement commencer par une tabulation obtenue par la touche associée. Sauvegarder le fichier puis quitter l'éditeur.
Ce fichier *makefile* contient les directives indiquant comment obtenir le fichier exécutable *exo1.exe*. Il faut l'interpréter de la manière suivante : *exo1.exe* dépend du fichier source *exo1.C*. Si *exo1.C* est modifié, il faut alors produire un nouvel exécutable *exo1.exe*. Pour obtenir le fichier exécutable *exo1.exe* il faut exécuter la commande *comp exo1*.

b) Utilisation de Make

Make est un utilitaire sous UNIX qui interprète automatiquement les lignes contenues dans le fichier *makefile*. Pour lancer cet utilitaire il suffit de taper la commande :

```
make <RC>
```

Supprimer du répertoire les fichiers *exo1.o* et *exo1.exe*, puis lancer l'utilitaire *make*. Que se passe-t-il ?
Lancer une deuxième fois l'utilitaire make. Le message suivant s'affiche :

```
make : 'exo1.exe'  is up to date.
```

Ce message indique que depuis la dernière compilation le fichier *exo1.C* n'a pas été modifié, par conséquent l'utilitaire n'a pas relancé la compilation pour produire un nouvel exécutable.
Modifier sous *gedit* le fichier source *exo1.C* pour y ajouter en commentaire le rôle du programme, puis relancer l'utilitaire *make*. Que se passe-t-il ? Justifier.

5. Exécution du programme pas à pas en utilisant le débogueur DDD

Définition : un débogueur est un outil (logiciel) permettant de contrôler l'exécution d'un programme pas à pas, de telle sorte qu'il est facile d'examiner l'état des variables, leur définition, de manière interactive pendant que le programme s'exécute.

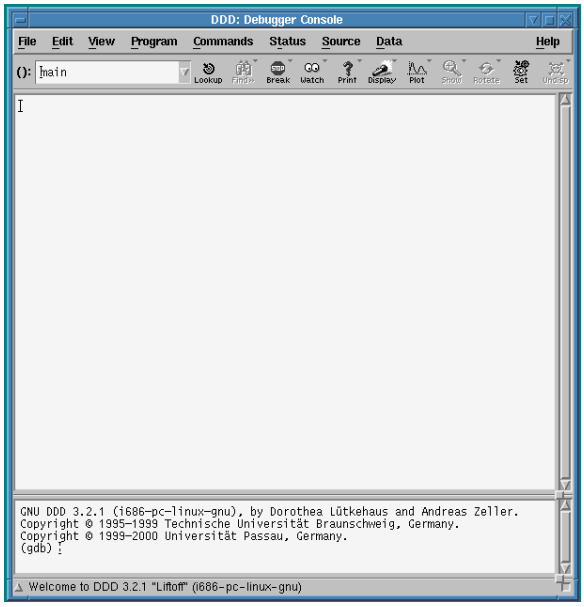
DDD est un débogueur sur système d'exploitation LINUX qui fournit une interface conviviale pour déboguer des programmes écrits en C, C++, Java, Perl, Fortran, Ada, … Nous pourrons ainsi l'utiliser pour différents langages de programmation.

Remarque importante :
Le débogueur nécessitant beaucoup de ressources, il est indispensable de refermer toutes les fenêtres déjà ouvertes pour ne garder que la fenêtre nacre. Attention également à ne pas conserver de fenêtre iconifiée. Cette manipulation est à faire chaque fois que se présentera l'occasion de manipuler le débogueur que ce soit dans ce TP ou dans ceux qui suivront.

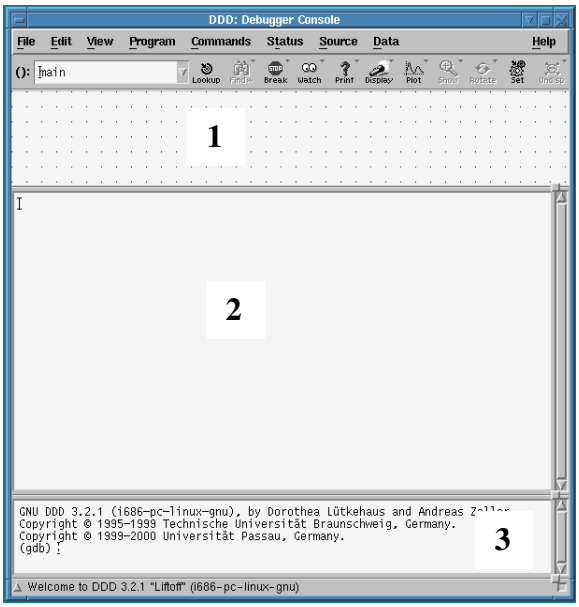
Lancer le débogueur *DDD* en tapant la commande :

```
ddd & <RC>
```

La fenêtre suivante s'affiche :



Sélectionner dans le menu *View* l'option *Data Window*. La fenêtre du débogueur est alors composée de trois parties :

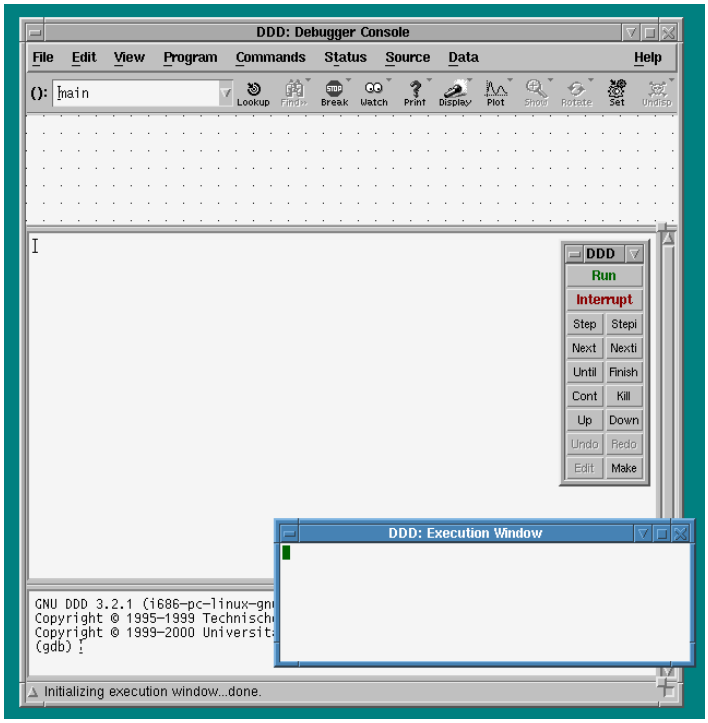


- en haut (1), une zone permettant l'affichage des variables du programme que nous appellerons « *Fenêtre des données* »,
- au milieu (2), une zone ou s'affichera le programme que nous appellerons « *Fenêtre du programme source* »,
- en bas (3), une zone de dialogue avec le débogueur que nous appellerons « *Console du débogueur* ».

Sélectionner dans le menu *View* l'option *Command Tool*. Une mini-fenêtre apparaît. Celle-ci peut être déplacée indépendamment de la grande et contient une série de boutons de commandes que nous explorerons dans la suite. Nous l'appellerons « *Outil de commande* ».

Sélectionner dans le menu *View* l'option *Execution Window*. Une autre fenêtre apparaît. Elle servira à la saisie et à l'affichage de données pendant l'exécution du programme. Nous l'appellerons « *Fenêtre d'exécution* ».

On obtient alors les fenêtres suivantes :



A partir de maintenant, on va pouvoir déboguer un programme.

a) Choix du programme à déboguer

Choisir le programme à déboguer en sélectionnant dans le menu *File* l'option *Open Program*. Une fenêtre de dialogue s'affiche. Sélectionner alors le programme *exo1.exe*. Le texte correspondant au programme source *exo1.C* apparaît alors dans la fenêtre 2.

b) Exécution normale du programme

Exécuter le programme en cliquant sur le bouton *Run* de l'outil de commande. Pour saisir la valeur de l'entier, cliquer dans la fenêtre d'exécution, taper la valeur et valider par un retour chariot. On peut voir le résultat obtenu dans cette même fenêtre. On peut voir également que le programme s'est déroulé normalement en regardant le message affiché dans la console du débogueur *« Program exited normally »*.

c) Exécution pas à pas du programme

Pour exécuter pas à pas le programme, il faut commencer par mettre un *point d'arrêt* (*breakpoint*) à l'endroit où l'on veut commencer l'observation du déroulement du programme. Si nous voulons commencer l'observation du programme depuis le début, il faut placer un point d'arrêt avant la première instruction qui suit le main. Cliquer sur le début de cette ligne avec le bouton droit et sélectionner l'option *Set Breakpoint* dans le menu qui apparaît. Un point d'arrêt apparaît sur la ligne sélectionnée, symbolisé par un panneau STOP.

Remarque : Pour enlever un point d'arrêt, cliquer avec le bouton droit sur ce point d'arrêt et sélectionner l'option *Delete Breakpoint* dans le menu qui apparaît.

Exécuter le programme (*cf. b*). il s'arrête désormais à la première instruction suivant le point d'arrêt. La flèche verte dans la fenêtre du programme source indique la prochaine ligne qui doit être exécutée.

En positionnant le curseur sur le nom d'une variable, on peut visualiser son contenu au moment où le programme s'est arrêté. Vérifier le contenu des variables *v* et *r*.

Pour exécuter la ligne suivante du programme, cliquer sur *Next* de l'outil de commande. Après l'exécution de l'instruction *r = 1 ;* vérifier le contenu des variables *v* et *r*.

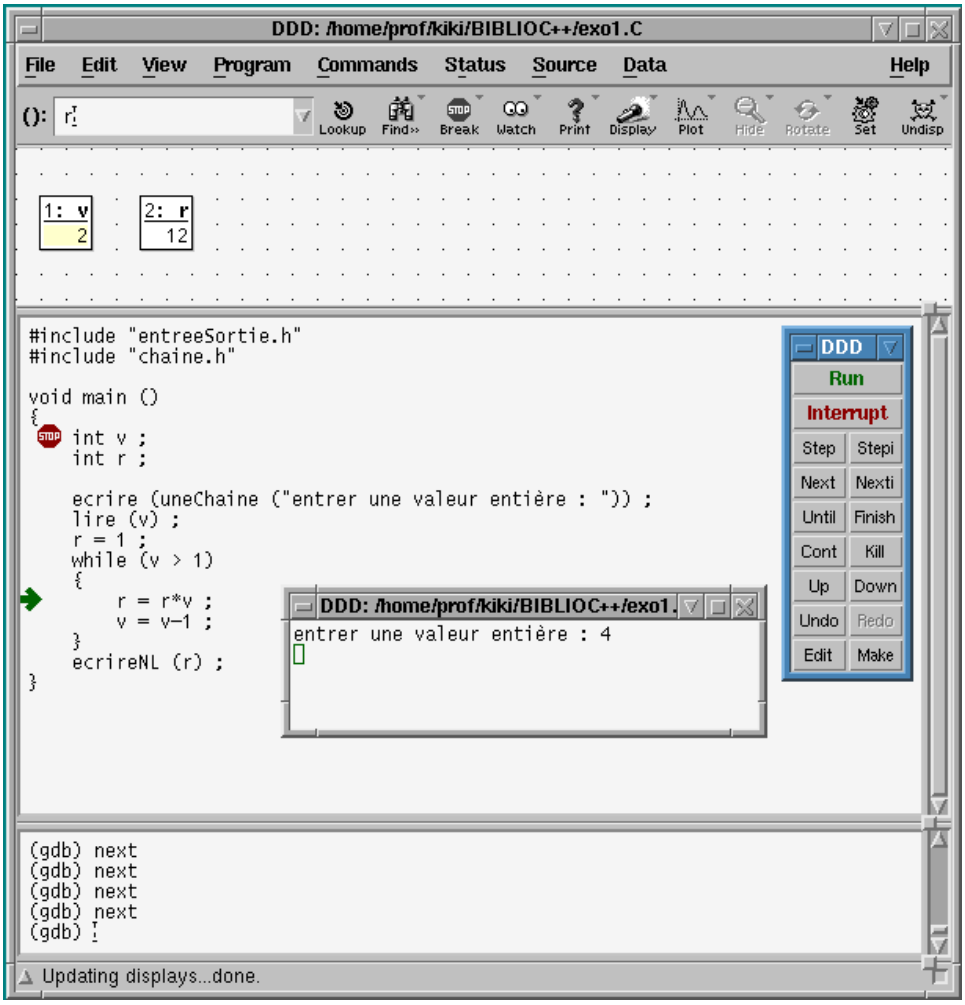
Pour continuer le programme jusqu'à la fin, sans s'arrêter à chaque ligne, cliquer sur *Cont* dans l'outil de commande. Faire plusieurs exécutions consécutives pas à pas pour vérifier le contenu des variables *v* et *r*.

Remarque : On peut cliquer sur le bouton *Kill* de l'outil de commande pour arrêter l'exécution du programme.

d) Exécution pas à pas du programme avec visualisation permanente du contenu des variables

On peut visualiser de manière permanente le contenu de chaque variable dans la fenêtre de données afin d'observer l'évolution de leurs valeurs au cours de l'exécution du programme.

Pour visualiser une variable, il faut tout d'abord que l'exécution ait été lancée, cliquer ensuite sur la variable à visualiser avec le bouton droit et choisir l'option *Display* dans le menu qui apparaît. Visualiser les variables *v* et *r*. A l'aide de la souris les ramener dans la fenêtre de données l'une à côté de l'autre pour éviter d'utiliser l'ascenseur (*cf. fig ci-dessous*).



Exécuter pas à pas le programme pour observer les changements des valeurs *v* et *r*. Ils sont matérialisés dans la fenêtre des données par la couleur jaune.

Remarque : Pour supprimer la visualisation d'une variable, cliquer avec le bouton droit sur la variable à supprimer dans la fenêtre des données et choisir l'option *Undisplay* dans le menu qui apparaît.

e) Modification du programme source

Pour modifier le programme source sans sortir du débogueur, cliquer sur le bouton *Edit* de l'outil de commande. La fenêtre d'un éditeur source s'affiche. Elle contient le fichier source.

Modifier ce dernier pour remplacer toutes les occurrences de la variables *r* par *f*. Quitter l'éditeur après avoir effectué la sauvegarde du fichier pour revenir sous le débogueur. Nous pouvons remarquer que la fenêtre du débogueur contenant le fichier source est automatiquement mise à jour en fonction des modifications apportées au fichier.

f) Recompilation du programme après modification du programme source

Pour recompiler le programme, cliquer sur le bouton *Make* de l'outil de commande. Cette action a pour effet d'appeler l'utilitaire *make* (vu dans le point 4). Cette fonctionnalité suppose bien entendu l'existence d'un fichier *makefile* associé. La console du débogueur affiche les résultats de la compilation.

g) Sortie du débogueur

Quitter le débogueur en sélectionnant l'option *Exit* du menu *File*.

Exercice 2

1. Taper sous l'éditeur de texte *gedit* le programme C++ suivant et le sauvegarder sous le nom *exo2.C*. Modifier le fichier *makefile* déjà existant pour remplacer toutes les occurrences de *exo1* par *exo2*. Compiler le programme et l'exécuter à partir de l'utilitaire *make*. Que fait-il ?

```
#include "/usr/local/public/BIBLIOC++/entreeSortie.h"
#include "/usr/local/public/BIBLIOC++/chaine.h"

int fonction (const int n) ;

int main ()
{
    int n ;
    int f ;

    ecrire (uneChaine ("entrer une valeur entiere : ")) ;
    lire (n) ;
    f = fonction (n) ;
    ecrireNL (f) ;
}

int fonction (const int n)
{
    int v ;
    int r ;

    v = n ;
    r = 1 ;
    while (v > 1)
    {
        r = r*v ;
        v = v-1 ;
    }
    return (r) ;
}
```

2. Lancer le débogueur *DDD*, puis ouvrir les fenêtres qui vont servir à déboguer le programme.

a) Choix du programme à déboguer

Choisir le programme *exo2.exe* à déboguer.

b) Exécution pas à pas du programme

Exécuter pas à pas le programme *exo2.exe* après avoir placé un point d'arrêt au début du programme principal.

c) Exécution instruction par instruction du programme

Si au cours d'une exécution pas à pas on clique sur le bouton *Step* au lieu de cliquer sur le bouton *Next* de l'outil de commande et si l'instruction à exécuter correspond à un appel de sous-programme, le programme s'arrête sur la première ligne du sous-programme correspondant (changement de contexte). Exécuter pas à pas le programme *exo2.exe* en utilisant le bouton *Next*. Lorsque la flèche verte est positionnée sur la ligne *f = fonction (n)*, au lieu de cliquer sur le bouton *Next* de l'outil de commande, cliquer sur le bouton *Step*, et regarder ce qui se passe.

Remarque : Le bouton *Up* de l'outil de commande permet de connaître la ligne qui a appelé la fonction en train d'être exécutée (contexte de l'appelant), le bouton *Down* de l'outil de commande permet de revenir à la fonction en train d'être exécutée (contexte de l'appelé).

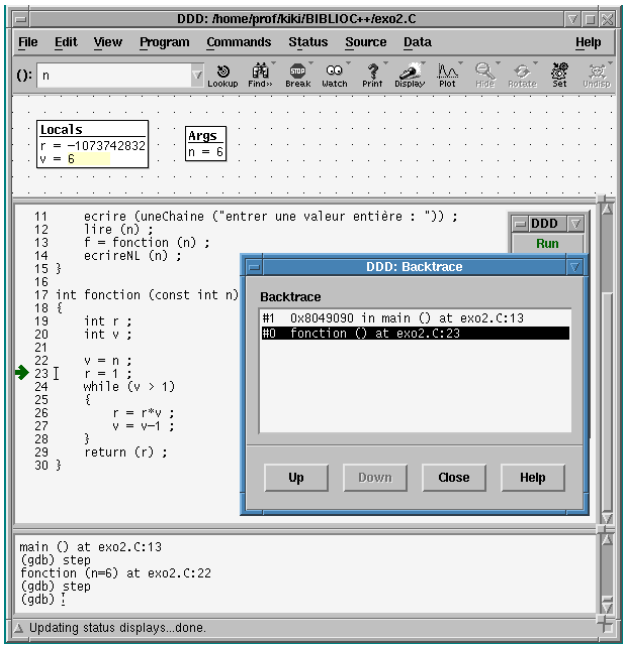
Remarque : Il est impossible de visualiser les lignes de code d'un fichier n'appartenant pas à l'utilisateur. C'est le cas des opérations d'entrée-sortie et des opérations sur les chaînes de caractères.

d) Exécution instruction par instruction du programme en affichant le contenu des variables dans les différents contextes

Sélectionner dans le menu *Data* l'option *Display Local Variables* et dans le même menu l'option *Display Arguments*. Exécuter le programme *exo2.exe* pas à pas en utilisant le bouton *Next* jusqu'à la ligne 13 puis *Step* par la suite et observer la fenêtre de données. Que se passe-t-il ?

e) Traçabilité

Dans le menu *Status*, sélectionner l'option *Backtrace*. Une nouvelle fenêtre apparaît : la *fenêtre de trace*. Réexécuter le programme comme dans d). Dans cette fenêtre apparaît le numéro de la prochaine ligne exécutée et son contexte. Par exemple, lorsque la flèche verte est positionnée sur la ligne 23, la fenêtre de trace affiche que la prochaine instruction à exécuter se trouve dans le fichier source *exo2.C* à la ligne 23 dans la fonction C++ *fonction()* qui elle-même a été appelée à la ligne 13 dans le fichier source *exo2.C* dans la fonction *main()* (*cf. fig. suivante*).



Remarque : pour obtenir les numéro de ligne dans le programme source, sélectionner l'option *Display Line Numbers* dans le menu *Source*.

f) Modification du source

Sans quitter le débogueur, modifier le source pour ajouter en commentaire le rôle du programme puis recompiler le programme pour vérifier s'il ne comporte pas d'erreur.

h) Sortie du débogueur

Exercice 3

1. Taper sous l'éditeur de texte *gedit* le programme C++ suivant et le sauvegarder sous le nom *exo3.C*.

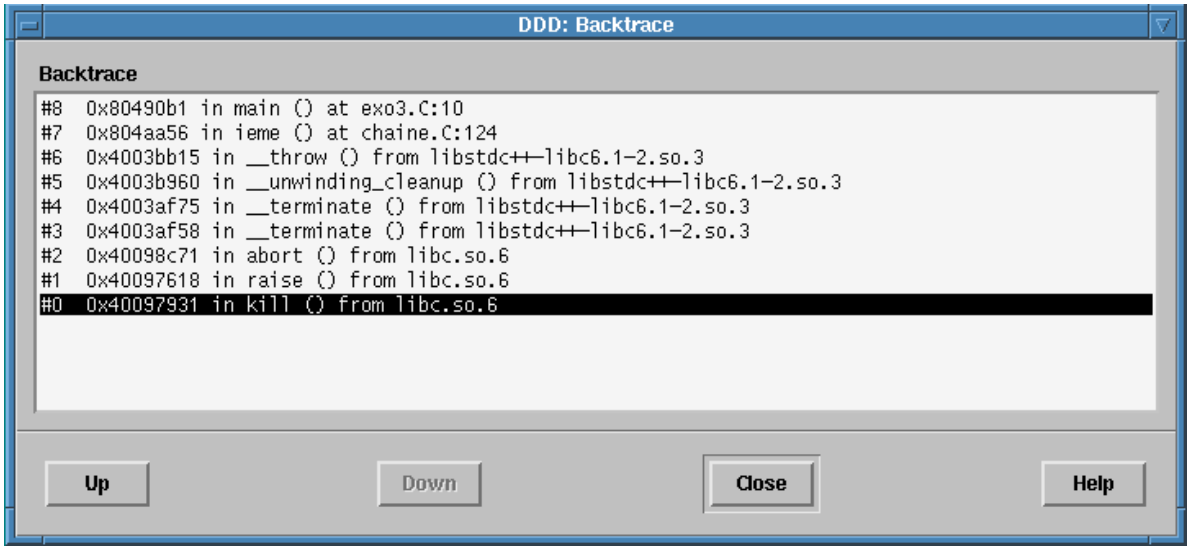
```
#include "/usr/local/public/BIBLIOC++/entreeSortie.h"
#include "/usr/local/public/BIBLIOC++/chaine.h"

int main ()
{
    Chaine ch ;

    ecrire (uneChaine ("entrer une chaine : ")) ;
    lire (ch) ;
    ecrireNL (ieme (ch, 6)) ;
}
```

Modifier le fichier *makefile* déjà existant pour qu'il permette la compilation du fichier *exo3.C*.
Que fait ce programme ?
Compiler le programme à l'aide de l'utilitaire *Make* et exécutez-le en tapant la première fois la chaîne de caractères « *bonjour* » et la deuxième fois en tapant la chaîne de caractères « *aaa* ». On peut remarquer que la première exécution se passe normalement alors que la seconde affiche le message :

```
Abort
```

Même s'il est aisé de trouver pourquoi et à quel endroit une erreur s'est produite au cours de la deuxième exécution, nous allons essayer de la retrouver par l'intermédiaire du débogueur.

2. Lancer le débogueur *DDD*, puis ouvrir les fenêtres qui vont servir à déboguer le programme et sélectionner le programme *exo3.exe*.
Ouvrir la fenêtre de trace et exécuter le programme en tapant la chaîne de caractères « *aaa* ».
Voici ce que l'on obtient :



Nous pouvons constater que le programme s'est arrêté avant d'exécuter la ligne 124 de la fonction *ieme ()* qui se trouve dans le fichier source *chaine.C*. La fonction *ieme ()* a été appelée dans la fonction *main ()* du fichier source *exo3.exe* à la ligne 10. Rechercher ce qu'il s'est donc passé et indiquer l'exception levée qui a interrompu l'exécution du programme.

3. En restant dans le débogueur *DDD*, modifier le code source avec le code ci-dessous, puis réexécuter pas à pas le programme pour voir ce qui se passe.

```
#include "/usr/local/public/BIBLIOC++/entreeSortie.h"
#include "/usr/local/public/BIBLIOC++/chaine.h"

int main ()
{
    try
    {
        Chaine ch ;

        ecrire (uneChaine ("entrer une chaine : ")) ;
        lire (ch) ;
        ecrireNL (ieme (ch, 6)) ;
    }
    catch (const Chaine message)
    {
        ecrireNL (message) ;
    }
}
```

4. En restant dans le débogueur *DDD*, modifier le code source pour que l'exception ne soit plus levée.

5. Quitter le débogueur *DDD*.

Exercice 4

Taper sous l'éditeur de texte *gedit* le programme C++ suivant et le sauvegarder sous le nom *exo4.C*.

```
int main ()
{
    int a, b ;

    a = 1024 ;
    b = a*a*a-1 ;
    a = 2*b ;
    b = a+1 ;
    a = b+1 ;
    b = 4*b ;
    a = 2*a ;
    b = b/a ;
}
```

Créer le fichier *makefile* puis compiler le programme et l'exécuter. Une exception système a été levée et le programme le signale par le message :

```
Floating Point exception
```

Utiliser le débogueur *DDD* pour localiser la ligne où s'est produit l'erreur. Exécuter pas à pas le programme pour déterminer à quel moment les résultats des calculs deviennent aberrants et pourquoi une exception arithmétique a été levée.

Exercice 5

Taper sous l'éditeur de texte *nedit* le programme C++ suivant et le sauvegarder sous le nom *exo5.C*. Créer le fichier *makefile* puis compiler le programme pour vérifier qu'il n'y a aucune erreur.

```
typedef int Tableau [3][3] ;

int main ()
{
    Tableau t ;

    for (int i = 0 ; i < 3 ; i++)
    {
        for (int j = 0 ; j < 3 ; j++)
        {
            t[i][j] = (i*10)+1+j+1 ;
        }
    }
}
```

Appeler le débogueur *DDD* en précisant en argument de la commande le nom du fichier exécutable à déboguer. Mettre un point d'arrêt en début de programme. Afficher les variables locales par l'option *Display Local Variables* du menu *Data* et exécuter pas à pas le programme. Nous pouvons constater que seuls le contenu des variables de types prédéfinis est affiché (le contenu du tableau *t* ne l'est pas).

Pour afficher le contenu de toutes les variables, relancer l'exécution du programme puis cliquer avec le bouton droit sur le mot *Locals* de la fenêtre de données et sélectionner l'option *Show All* dans le menu qui apparaît. On peut voir ainsi apparaître le contenu du tableau. Exécuter le programme pas à pas pour vérifier les modifications du tableau. Au besoin, on agrandira la fenêtre des données.

Que fait le programme ?

Mettre un deuxième point d'arrêt devant l'avant dernière accolade du programme.

Lancer l'exécution du programme. Au lieu de l'exécuter pas à pas (bouton *Next*), utiliser le bouton *Cont* de l'outil de commande. Que se passe-t-il ?

Exercice 6

Modifier à partir de *DDD* le programme source *exo5.C* pour obtenir les lignes suivantes :

```
typedef int Tableau [3][3] ;

int main ()
{
    Tableau t ;

    for (int i = 0 ; i < 3 ; i++)
    {
        for (int j = 0 ; j != 3 ; j = j+2)
        {
            t[i][j] = (i*10)+1+j+1 ;
        }
    }
}
```

Que fait le programme à l'exécution ? En utilisant les commandes du débogueur, déterminer pourquoi le programme boucle indéfiniment.

Exercice 7

Explorer les différents menus de l'outil *DDD* pour découvrir de nouvelles fonctionnalités et les noter au fur et à mesure sur une feuille de papier.