Chapter 7

Exemple de conception et implantation par raffinement successif

Énoncé

On veut écrire un programme qui fait la mise à jour de votre compte en banque. Le programme traite les transactions (dépôt (d), retrait (r) et terminer (t)) une à la fois, ajuste le solde et garde trace du nombre de transactions de chaque type. À la fin on imprime le solde de départ, le nouveau solde et le nombre de transactions de chaque type.

On doit refuser un retrait si le solde devient négatif.

7.1 Analyse et conception globale

Analyse

Entrée :
- Solde de départ (réel)
- Suite de transactions comprenant :
  - code (caractère)
  - montant (réel).

Sortie :
- Solde de départ (réel)
- Solde courant (réel)
- Nombre de dépôts (entier)
- Nombre de retraits (entier)

formules :

Simple addition et soustraction

Constantes :

'd' pour dépôt

'r' pour retrait

't' pour terminer

Conception : diagramme structurel

Le diagramme structurel pour le niveau principal ressemble à ce qui suit :

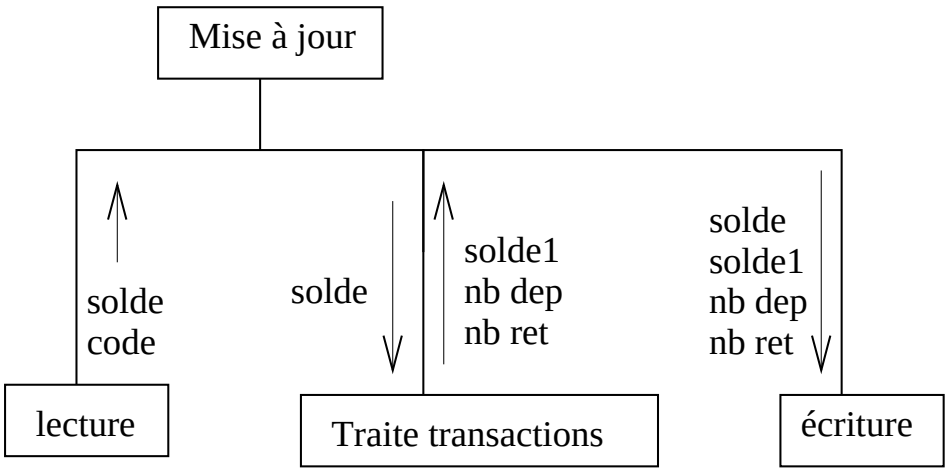

Figure 7.1: Diagramme structurel niveau 1

On veut valider la lecture. Ainsi le module de lecture retourne un code d'erreur si il y a eu un problème lors de la lecture. On veut traiter les transactions seulement si la lecture initiale s'est faite correctement. Cela ne se reflète pas dans le diagramme.

7.2 Analyse, conception et implantation du module Mise à jour

Analyse

Entrée :
(lecture) Solde de départ (réel)
Code d'erreur (entier)
Sortie : rien

Conception : Algorithme

1. Lecture du solde initial
2. Si la lecture initiale est correcte.
   - Traite toutes les transactions
   - Impression des résultats.

Implantation

On implante le niveau principal. On implante aussi le module d'impression car il est relativement simple.

```
/*************************************************************/
/*                                                           */
/*  Compte.cc                                                */
/*                                                           */
/*  Ce programme fait la mise a jour mensuel d'un compte en */
/*  banque                                                   */
/*                                                           */
/*  Auteur : Gabriel Girard                                  */
/*                                                           */
/*  Entree : solde_depart (float)                            */
/*  Sortie :                                                 */
/*                                                           */
/*************************************************************/

#include <iostream>

using namespace std;
```

```
int main()
{
  // Fonctions utiles
  int lecture_solde(float &);
  int traite_transactions(float, float &, int &, int &);
  void affiche_resultat(float, float, int, int);

  // Definition des variables
  int nb_depot = 0, nb_retrait = 0;
  int code;                 // pour les codes d'erreurs

  float solde_depart, solde_courant;

  // on lit le solde de depart
  code = lecture_solde(solde_depart);

  // on traite toutes les transactions

  if (code == 0)
     {
       // on traite toutes les transactions
       code = traite_transactions(solde_depart, solde_courant,
                                  nb_depot, nb_retrait);

      // on affiche le resume des transactions et le solde
      affiche_resultat(solde_depart, solde_courant,
                       nb_depot, nb_retrait);
     }

  else cout << "Erreur lors de la lecture --- "
            << "traitement annule\n";

}

/*************************************************************/
/*                                                           */
/*                                                           */
/*  Cette fonction imprime le resume des transactions        */
/*                                                           */
/*                                                           */
/*  Entree et sortie : solde_depart (float)                  */
/*                     solde_courant (char)                  */
/*                     nb_depots (int)                       */
/*                     nb_retraits (int)                     */
/*                                                           */
```

```
/*********************************************************/
#include <iostream>

void affiche_resultat(float solde1, float solde2,
                      int   nb_dep, int   nb_ret)
{
   cout << "Resume des transactions du mois. " << endl;
   cout << "Solde initial      ==> " << solde1 << endl;
   cout << "Solde final        ==> " << solde2 << endl;
   cout << "Nombre de depots   ==> " << nb_dep << endl;
   cout << "Nombre de retraits ==> " << nb_ret << endl;
}
```

7.3 Lecture initiale

Maintenant que l'on a fait l'analyse, la conception et l'implantation du module principal, on peut s'attaquer à l'analyse, la conception et l'implantion du premier sous-module, le module de lecture.

Analyse Lecture initiale

Entrée : rien
Sortie:
solde initial (réel)
code d'erreur (0 - correct, 1 - montant invalide)

Conception Lecture initiale

On peut donner à l'utilisateur trois chances pour entrer des données valides. Après ces trois chances le module retourne un code d'erreur.

Pour un certain nombre d'essais maximum
lire solde initial
verifie si valide pour fin boucle
si solde valide on retourne le solde
sinon retourne erreur

Implantation

```
/*********************************************************/
/*  Lecture_solde.cc                                     */
/*                                                       */
/*  Cette fonction lit le solde initial pour le compte   */
/*                                                       */
/*  Auteur : Gabriel Girard                              */
/*                                                       */
/*  Sortie : solde_depart (float)                        */
```

```
/*           code (int)                                        */
/*************************************************************/

#include <iostream>


int lecture_solde(float & solde_depart)
{
  // definition de la constante
  const int NB_ESSAI = 3;
  // Definition des variables
  int code=0, cpt=0;              // pour les codes d'erreurs
  bool valide = false;

  while (cpt < NB_ESSAI && !valide)
  {
    cout << "Entrer le solde de depart (il doit etre > 0) : ";
    cin  >> solde_depart;
    if (solde_depart > 0) valide = true;
    cpt++;
  }
  if (cpt >= NB_ESSAI) code = 1;

  return code;
}
```

7.4 Module traite transactions

Le premier sous-module terminé, on fait le second. On applique la méthode de conception descendante par raffinement successif.

Analyse du module traite transactions

Entrée
: (paramètre) solde initial (réel)

Sortie
: (paramètre) solde courant (réel)
: (paramètre) nombre de dépôts (entier)
: (paramètre) nombre de retraits (entier)

Constantes :
: 'd' pour dépôt
: 'r' pour retrait
: 't' pour terminer

Conception traite_transactions

Cette fonction reçoit un code et un montant si le code est valide. Elle traite toutes les transactions. Elle se décompose de nouveau en plusieurs modules : un pour la lecture, un pour traiter une seule transaction et un pour imprimer un résumé du traitement. Tous ces modules sont appelés jusqu'à ce que toutes les transactions soient traitées.

Voici donc le diagramme sructurel pour ce niveau.

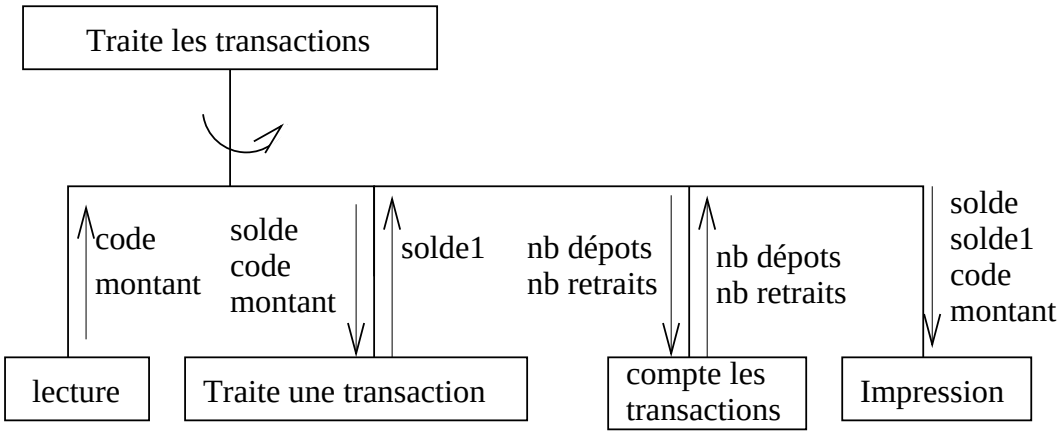

Figure 7.2: Diagramme structurel niveau 2

Algorithme

Pour chacune des transactions

1. Lecture code transaction et montant de la transaction
2. Si lecture correcte
   a. Traite la transaction
   b. On compte le nombre de transactions
   c. Imprime le nouveau solde

Implantation

On implante ici le module traite_transactions et le module d'impression étant donné sa simplicité.

```
/*************************************************************/
/*                                                           */
/*  Traite_transactions.cc                                   */
/*                                                           */
/*  Ce programme traite toutes les transactions sur un compte */
/*  en banque.  Il manipule par l'intermediaire de fonctions  */
/*  les donnees suivantes:                                   */
/*               solde (float)                               */
/*               solde_cour (float)                          */
/*               nb_dep, nb_ret (int)                        */
/*   Entree :    solde (float)                               */
/*   E/S         solde_cour (float)                          */
/*               nb_dep, nb_ret (int)                        */
/*                                                           */
/*  Auteur : Gabriel Girard                                  */
/*                                                           */
/*************************************************************/

#include <iostream>

traite_transactions(float solde, float& solde_cour,
                    int& nb_dep, int& nb_ret)
{
  // Fonctions utiles
  int lire_trans(char &, float &);
  int traite_une_trans(float, char, float, float &);
  void affiche_resume(float, char, float, float);

  // Definition des variables
  char code_trans;

  int code;                  // pour les codes d'erreurs
  float montant;

  solde_cour = solde;

  // on lit la transaction
  code = lire_trans(code_trans, montant);

  while ((code == 0) && (code_trans != 't'))
  {
     // on traite une transaction
     traite_une_trans(solde, code_trans,
                      montant, solde_cour);

     // mise a jour du nombre de depots ou retraits
```

```
     if (code_trans == 'r' && solde != solde_cour) nb_ret++;
     else nb_dep++;

     // on affiche le resume
     affiche_resume(solde, code_trans,
                    montant, solde_cour);

     // mise a jour du solde
     solde = solde_cour;

     // on lit la prochaine transaction
     code = lire_trans(code_trans, montant);

  }

  if (code != 0)
           cout << "\n\t *** Mauvaise transaction --- "
                << "traitement interrompu ***\n";
  else     cout << "\nFin du traitement des transactions\n"
                << "-----------------------------------" << endl;

  return code;
}

/*********************************************************/
/*                                                       */
/*  Cette fonction imprime le resume de la transaction   */
/*                                                       */
/*                                                       */
/*  Entree : solde_depart (float)                        */
/*           code_trans (char)                           */
/*           montant (float)                             */
/*           solde_courant (float)                       */
/*                                                       */
/*********************************************************/

void affiche_resume(float solde1, char code_t,
                    float montant, float solde2)
{
  switch (code_t)
  {
    case 'r' : cout << "Retrait de $" << montant << endl;
               cout << "solde initial ==> " << solde1 << endl;
               cout << "solde final   ==> " << solde2 << endl;
               break;
```

```
    case 'd' : cout << "depot de $" << montant << endl;
               cout << "solde initial ==> " << solde1 << endl;
               cout << "solde final   ==> " << solde2 << endl;

   }
}
```

7.5 Module lire une transaction

Nous avons fini l'analyse de tous les modules de niveau 1. On s'attaque maintenant aux modules du niveau inférieur, ceux de niveau 2. À ce niveau, on retrouve seulement les sous-modules de «traite_transactions». Le premier module à analyser est celui de lecture.

Analyse de lire une transaction

Sortie
: code de transaction (caractère)
: montant (réel)

Conception : algorithme

```
Pour un nombre d'essais maximal
     lire et valider le code
Si code correct alors lire montant
sinon retourner erreur
```

Implantation

```
/*********************************************************/
/*                                                       */
/*  Lire_une_transaction.cc                              */
/*                                                       */
/*  Cette fonction lit la prochaine transaction          */
/*                                                       */
/*                                                       */
/*  Sortie : code_trans (char)                           */
/*           montant (float)                             */
/*                                                       */
/*********************************************************/
#include <iostream>
```

```
int lire_trans(char& code_trans, float& montant)
{
  // definition de la constante
  const int NB_ESSAI = 3;

  // Definition des variables

  int code=0, cpt=0;             // pour les codes d'erreurs
  bool valide = false;

  while (cpt < NB_ESSAI && !valide)
  {
    cout << "Entrer le code de la transaction\n";
    cout << "         r -- retrait\n";
    cout << "         d -- depot\n";
    cout << "         t -- pour terminer\n\n";
    cin  >> code_trans;
    if (code_trans == 'd' || code_trans == 'r'
                          || code_trans == 't')
         valide = true;
    else cout << "Code de transaction invalide; "
              <<  "Recommencer.\n";
    cpt++;
  }
  if (cpt < NB_ESSAI && code_trans != 't')
  {   cout << "Entrer le montant de la transaction : ";
      cin >> montant;
  }
  else if (cpt >= NB_ESSAI) code = 1;

  return code;
}
```

7.6 Module traite une transaction

Analyse de traite une transaction

Entrée
- code (caractère)
- montant (réel)
- solde de départ (réel)

Entrée/Sortie

solde (réel)

formule
Solde = Solde précédent + montant du dépôt
Solde = Solde précédent - montant du retrait

Conception : algorithme
Si code de retrait
alors effectuer retrait
sinon effectuer dépôt

Implantation

```
/*************************************************************/
/*                                                           */
/*  traite_une_trans.cc                                      */
/*                                                           */
/*  Cette fonction traite une transaction sur le compte      */
/*                                                           */
/*                                                           */
/*  Entree : solde_depart (float)                            */
/*           code_trans (char)                               */
/*           montant (float)                                 */
/*  Sortie : solde_courant (float)                           */
/*                                                           */
/*************************************************************/
#include <iostream>

void traite_une_trans(float solde, char code_trans,
                      float montant, float& solde_courant)
{
   float temp;

   switch (code_trans) {

   case 'd' : solde_courant = solde + montant;
              break;

   case 'r' : temp = solde - montant;
              if (temp < 0)
                  cout << " ATTENTION -- solde negatif "
                       << "------ transaction refusee \n";
              else solde_courant = temp;
   }
}
```