Introduction au C++

Guillaume Fuhr

Guillaume.fuhr@univ-amu.fr

References

- ***Livres***
  - *Cours de programmation par objets — Principes et applications avec Eiffel et C++. de* M. Gautier, G. Masini et K. Proch.
  - *Conception et programmation par objets, Seconde édition* de B. Meyer.
  - *Effective C++, More Effective C++,* Scott Myers
  - *Exceptional C++, More Exceptional C++,* Herb Sutter
  - ***Thinking in C++, Bruce Eckel***
    - Disponible sur *http://mindview.net/Books/TICPP/ThinkingInCPP2e.html*
  - *The C++ Programming Language, 3rd edition*, de Bjarne Stroustrup

- ***Sites Web***
  - http://cpp.developpez.com/
  - http://www.cplusplus.com/
- **Exemples utilisés en cour :**
  - https://github.com/GFuhr/M2_FI

Premier exemple C++

```
1.  // operations avec des variables
2.  #include <iostream>
3.  using namespace std;

4.  int main (void)
5.  {
6.    // declaration des variables:
7.    int a=5;
8.    int b(2);
9.  // calcul:
10.   a = a + 3;
11. int result; // valeur init. indeterminee
12.   result = a - b;
13.   // affichage du resultat en console:
14.   cout << result;

15.   // fin du programme:
16.   return 0;
17. }
```

```
6
```

Classe Ccompte I

```
1.  // POO_banque.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v0.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1;
9.
10.     cout<<"le compte 1 a un solde de " << compte1.dSolde<<endl;
11.     cout<<"montant de la derniere operation " << compte1.dLastOp<<endl;

12.     compte1.dSolde=25;
13.     cout<<"le compte 1 a un solde de " << compte1.dSolde<<endl;
14.     cout<<"montant de la derniere operation " << compte1.dLastOp<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte.h
2.  class CCompte
3.  {
4.  public:
5.      double dSolde;
6.      double dLastOp;
7.  };
```

le compte 1 a un solde de 0

le montant de la derniere operation sur le compte 1 etait de **2.07336e-317**

le compte 1 a un solde de 25

le montant de la derniere operation surl e compte 1 etait de **2.07336e-317**

Classe Ccompte II

```
1.  // POO_banque_v0.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v0.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1;
9.

10.     cout<<"le compte 1 a un solde de " <<
    compte1.dSolde<<endl;
11.     cout<<"le montant de la derniere
    operation surl e compte 1 etait de " <<
    compte1.dLastOp<<endl;

12.     compte1.dSolde=25;
13.     cout<<"le compte 1 a un solde de " <<
    compte1.dSolde<<endl;
14.     cout<<"le montant de la derniere
    operation surl e compte 1 etait de " <<
    compte1.dLastOp<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v0.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  };
```

In file included from POO_banque_v0.cpp:4:0:
CCompte_v0.h: In function ‘int main(int, char**)’:
CCompte_v0.h:5:12**: error: ‘double CCompte::dSolde’ is private**
private
POO_banque_v0.cpp:13:51: error: within this context
In file included from POO_banque_v0.cpp:4:0:
CCompte_v0.h:6:12: error: ‘double CCompte::dLastOp’ is private
POO_banque_v0.cpp:14:86: error: within this context
In file included from POO_banque_v0.cpp:4:0:
CCompte_v0.h:5:12: error: ‘double CCompte::dSolde’ is private
POO_banque_v0.cpp:16:13: error: within this context
In file included from POO_banque_v0.cpp:4:0:
CCompte_v0.h:5:12: **error: ‘double CCompte::dSolde’ is private**
POO_banque_v0.cpp:17:51: error: within this context
In file included from POO_banque_v0.cpp:4:0:
CCompte_v0.h:6:12: error: ‘double CCompte::dLastOp’ is private
POO_banque_v0.cpp:18:86: error: within this context

Classe Ccompte III

```cpp
// POO_banque_v2.cpp
//
#include <iostream>
#include "CCompte_v2.h"

using namespace std;

int main(int argc, char* argv[])
{
    CCompte compte1;

    cout<<"le compte 1 a un solde de " << compte1.recup_solde()<<endl;
    cout<<"le montant de la derniere operation sur le compte 1 etait de " << compte1.recup_lastop()<<endl;

    compte1.placer(25);
    cout<<"le compte 1 a un solde de " << compte1.recup_solde()<<endl;
    cout<<"le montant de la derniere operation sur le compte 1 etait de " << compte1.recup_lastop()<<endl;
return 0;
}
```

```cpp
// Ccompte_v2.h
class CCompte
{
private:
    double dSolde;
    double dLastOp;
public:
    double recup_solde( void)
    {   return dSolde;   };
    double recup_lastop( void)
    {    return dLastOp; };
    void placer( double dValeur)
    {
        dSolde=dSolde+dValeur;
        dLastOp=dValeur;
    };
};

double Ccompte::retirer( double dValeur)
    {
        dLastOp=dValeur;
        return dSolde-dValeur;
    }
```

Classe Ccompte IV

```
1.  // POO_banque_v2.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v0.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1;
9.
10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;

12.     compte1.placer(25);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v2.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
8.   CCompte( void )
9.      {dSolde=dLastOp=0.0;
10.      std::cout<<"***"<<std::endl<<
    "constructeur de la classe CCompte
    "<<std::endl<<"***"<<std::endl;
11.     }
12.     ~CCompte( void )
13.     {
14.        std::cout<<"++++"<<std::endl<<
    "destructeur de la classe CCompte
    "<<std::endl<<"++++"<<std::endl;
15.     }
16.     double recup_solde( void);
17.     double recup_lastop( void);
18.     void placer( double dValeur);
19.     double retirer( double dValeur);
20. };
```

Classe Ccompte IV

```
***
constructeur de la classe CCompte
***
le compte 1 a un solde de 0
le montant de la derniere operation sur le compte 1 etait de 0
le compte 1 a un solde de 25
le montant de la derniere operation sur le compte 1 etait de 25
++++
destructeur de la classe CCompte
++++
```

Classe Ccompte V

```
1.  // POO_banque_v4.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v4.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1(30);
9.
10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;

12.     compte1.placer(25);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v4.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
■   CCompte(double sommeinit)
■     {dSolde=dLastOp=sommeinit;
■       std::cout<<"***"<<std::endl<<"constructeur de
    la classe CCompte avec argument
    "<<std::endl<<"***"<<std::endl;
■     }    ~CCompte( void );

1.      double recup_solde( void);
2.      double recup_lastop( void);
3.      void placer( double dValeur);
4.      double retirer( double dValeur);
5.  };
```

Classe Ccompte V

***

constructeur de la classe Ccompte avec argument

***

le compte 1 a un solde de **30**

le montant de la derniere operation sur le compte 1 etait de **30**

le compte 1 a un solde de 55

le montant de la derniere operation sur le compte 1 etait de 25

++++

destructeur de la classe CCompte

++++

Classe Ccompte VI

```
1.  // POO_banque_v4b.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v4.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1;
9.
10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;

12.     compte1.placer(25);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v4.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
```

- CCompte(double sommeinit)
- {dSolde=dLastOp=sommeinit;
- std::cout<<"***"<<std::endl<<"constructeur de la classe CCompte avec argument "<<std::endl<<"***"<<std::endl;
- } ~CCompte(void);

```
1.      double recup_solde( void);
2.      double recup_lastop( void);
3.      void placer( double dValeur);
4.      double retirer( double dValeur);
5.  };
```

Classe Ccompte VI

POO_banque_v4b.cpp: In function ‘int main(int, char**)’:

POO_banque_v4b.cpp:10:13: **error: no matching function for call to ‘CCompte::CCompte()’**

POO_banque_v4b.cpp:10:13: note: candidates are:

In file included from POO_banque_v4b.cpp:4:0:

CCompte_v4.h:10:5: note: CCompte::CCompte(double)

CCompte_v4.h:10:5: note: **candidate expects 1 argument, 0 provided**

Classe Ccompte VII

```
1.  // POO_banque_v5.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v0.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1;
9.
10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;

12.     compte1.placer(25);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v5.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
8.   CCompte( void )
9.   {dSolde=dLastOp=0.0;
10.         std::cout<<"***"<<std::endl<<
    "constructeur de la classe CCompte avec
    argument "<<std::endl<<"***"<<std::endl;
11.     }

12.   CCompte( double sommeinit)
13.   {dSolde=dLastOp=sommeinit;
14.         std::cout<<"***"<<std::endl<<
    "constructeur de la classe CCompte avec
    argument "<<std::endl<<"***"<<std::endl;
15.     }
16.   ~CCompte( void );
17.   double recup_solde( void);
18.   double recup_lastop( void);
19.   void placer( double dValeur);
20.   double retirer( double dValeur);
21. };
```

Classe Ccompte VII

```
1.  // POO_banque_v2.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v0.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1;
9.
10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;

12.     compte1.placer(25);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le montant de la derniere
    operation sur le compte 1 etait de " <<
    compte1.recup_lastop()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v2.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
8.   CCompte( void )
9.      {dSolde=dLastOp=0.0;
10.        std::cout<<"***"<<std::endl<<
    "constructeur de la classe CCompte avec
    argument "<<std::endl<<"***"<<std::endl;
11.     }

12.         CCompte( double sommeinit)
13.     {dSolde=dLastOp=sommeinit;
14.        std::cout<<"***"<<std::endl<<
    "constructeur de la classe CCompte avec
    argument "<<std::endl<<"***"<<std::endl;
15.     }
16. ~CCompte( void );
17.     double recup_solde( void);
18.     double recup_lastop( void);
19.     void placer( double dValeur);
20.     double retirer( double dValeur);
21. };
```

Classe Ccompte VII

```
POO_banque_v4b.cpp: In function ‘int main(int, char**)’:
POO_banque_v4b.cpp:10:13: error: no matching function for call to
      ‘CCompte::CCompte()’
POO_banque_v4b.cpp:10:13: note: candidates are:
In file included from POO_banque_v4b.cpp:4:0:
CCompte_v4.h:10:5: note: CCompte::CCompte(double)
CCompte_v4.h:10:5: note:   candidate expects 1 argument, 0 provided
```

Classe Ccompte VIII

```
1.  // POO_banque_v2.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v6.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1(30);
9.      CCompte compte2(12);

10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;

12.     compte1.virement(compte2,12);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v2.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
8.   CCompte( void );

9.   CCompte( double sommeinit);

10. ~CCompte( void );
11.     double recup_solde( void);
12.     double recup_lastop( void);
13.     void placer( double dValeur);
14.     double retirer( double dValeur);
15.     double virement( CCompte autrecompte,
    double somme)
16.     {
17.         dLastOp = somme;
18.         dSolde=dSolde+somme;
19.         autrecompte.dLastOp = -somme;
20.         autrecompte.dSolde -= somme;
21.     }

22. };
```

Classe Ccompte VIII

```
1.  // POO_banque_v2.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v6.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1(30);
9.      CCompte compte2(12);

10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;

12.     compte1.virement(compte2,12);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;
15. return 0;
16. }
```

```
***
constructeur de la classe CCompte avec argument
***
***
constructeur de la classe CCompte avec argument
***
le compte 1 a un solde de 30
le compte 2 a un solde de 12
++++
destructeur de la classe CCompte
++++
le compte 1 a un solde de 42
le compte 2 a un solde de 12
++++
destructeur de la classe CCompte
++++
++++
destructeur de la classe CCompte
++++
```

2 constructeurs appelés en apparence
3 destructeurs appelés

Classe Ccompte IX

```
1.  // POO_banque_v2.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v6.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1(30);
9.      CCompte compte2(12);

10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;

12.     compte1.virement(compte2,12);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v7.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
8.   CCompte( void );
9.   CCompte( double sommeinit);

10. ~CCompte( void );
11.     double recup_solde( void);
12.     double recup_lastop( void);
13.     void placer( double dValeur);
14.     double retirer( double dValeur);
15.     double virement( CCompte autrecompte,
    double somme);

16.  CCompte(CCompte const &  comptebase)
17.     {
18.         dSolde=comptebase.dSolde;
19.         dLastOp=comptebase.dLastOp;
20.         std::cout<<"***"<<std::endl<<
    "constructeur de la classe CCompte par
    copie "<<std::endl<<"***"<<std::endl;
21.     }
22. };
```

Classe Ccompte IX

```
In file included from POO_banque_v7.cpp:4:0:
CCompte_v7.h:20:42: error: invalid constructor; you probably meant ‘CCompte
      (const CCompte&)’
```

Classe Ccompte X

***
constructeur de la classe CCompte avec argument
***
***
constructeur de la classe CCompte avec argument
***
le compte 1 a un solde de **30**
le compte 2 a un solde de **12**
***
constructeur de la classe CCompte par copie
***
++++
destructeur de la classe CCompte
++++
le compte 1 a un solde de **42**
le compte 2 a un solde de **12**
++++
destructeur de la classe CCompte
++++
++++
destructeur de la classe CCompte
++++

Classe Ccompte XI

```
1.  // POO_banque_v2.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v6.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1(30);
9.      CCompte compte2(12);

10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;

12.     compte1.virement(compte2,12);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_v2.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
8.   CCompte( void );
9.   CCompte( double sommeinit);

10. ~CCompte( void );
11.     double recup_solde( void);
12.     double recup_lastop( void);
13.     void placer( double dValeur);
14.     double retirer( double dValeur);
15.     double virement( CCompte &
    autrecompte, double somme)
16.  {
17.         dLastOp = somme;
18.         dSolde=dSolde+somme;
19.         autrecompte.dLastOp = -somme;
20.         autrecompte.dSolde -= somme;
21.     }

22.  CCompte(CCompte const &  comptebase);
23. };
```

Classe Ccompte XI

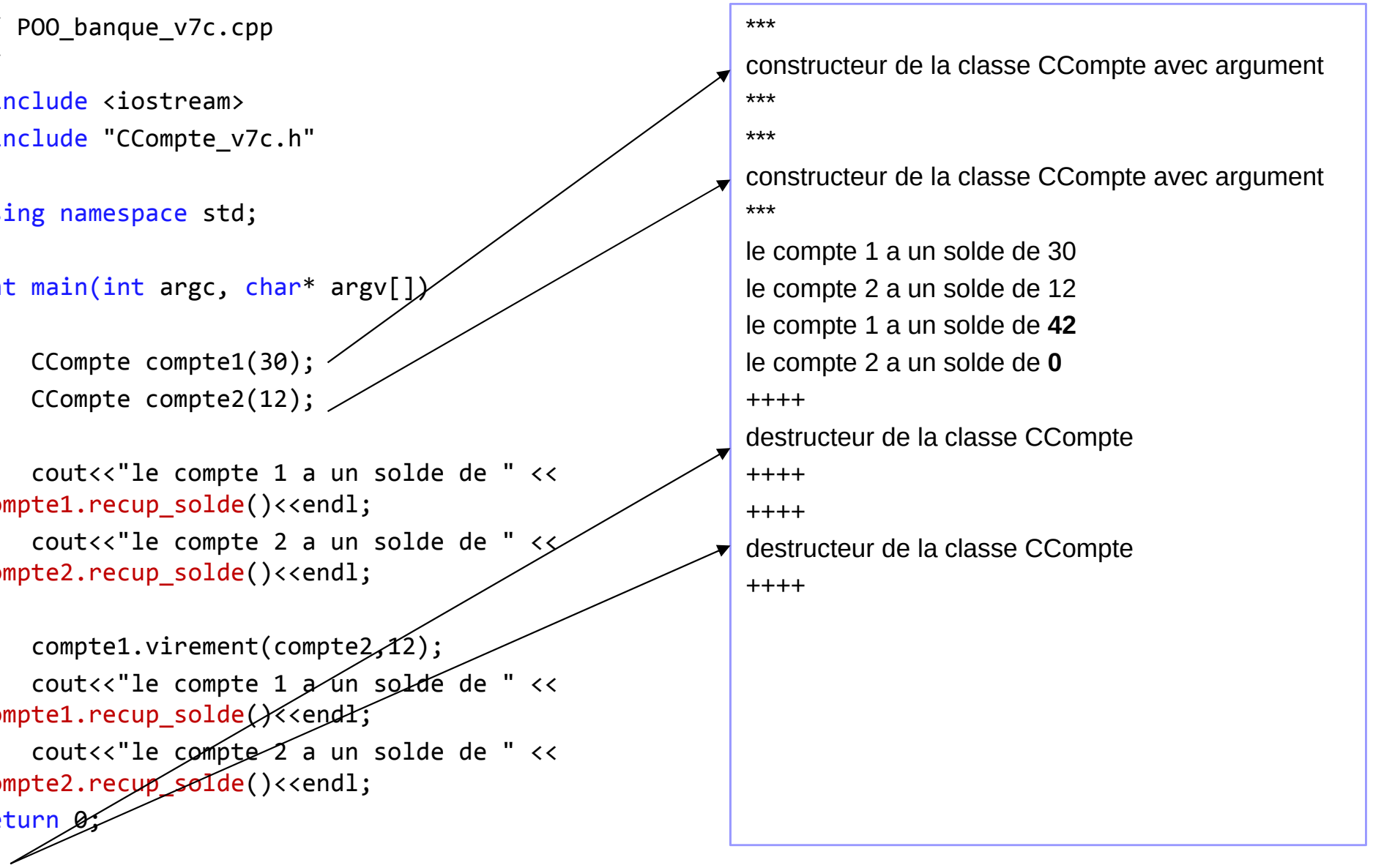
```
1.  // POO_banque_v7c.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v7c.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1(30);
9.      CCompte compte2(12);

10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;

12.     compte1.virement(compte2,12);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;
15. return 0;
16. }
```



Classe Ccompte finale

```
1.  // POO_banque_v2.cpp
2.  //
3.  #include <iostream>
4.  #include "CCompte_v6.h"

5.  using namespace std;

6.  int main(int argc, char* argv[])
7.  {
8.      CCompte compte1(30);
9.      CCompte compte2(12);

10.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
11.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;

12.     compte1.virement(compte2,12);
13.     cout<<"le compte 1 a un solde de " <<
    compte1.recup_solde()<<endl;
14.     cout<<"le compte 2 a un solde de " <<
    compte2.recup_solde()<<endl;
15. return 0;
16. }
```

```
1.  // Ccompte_vf.h
2.  class CCompte
3.  {
4.  private:
5.      double dSolde;
6.      double dLastOp;
7.  public:
8.      CCompte( void );
9.      CCompte( double const sommeinit);
10.     CCompte(CCompte const &  comptebase);
11.     ~CCompte( void );
12.     double recup_solde( void ) const;
13.     double recup_lastop( void ) const;
14.     void placer( double const dValeur);
15.     double retirer( double const
    dValeur);
16.     double virement( CCompte &
    autrecompte, double const  somme);
17. };
```

Heritage de classes

```
1.  #include <iostream>
2.  #include <string>
3.  #include "CCompte_vf.h"
4.  class CPersonne
5.  {
6.  protected:
7.      int id;
8.      std::string name;
9.  public:
10.     Cpersonne ( std::string pers_name="JohnDoe", int pers_id=0 );
11.     CPersonne( CPersonne const & autrepersonne);
12.     ~CPersonne(void);
13.     std::string  getName( void) const;
14.     void setName( std::string newname);
15.     int  getID( void) const;
16. };

17. class CClient: public CPersonne
18. {  private:
19.     CCompte ComptePerso;
20. public:
21.     CClient( void):CPersonne()
22.     {  std::cout<<"*** Personne : [ "<<name<<" ] devenue client ***"<<std::endl;     };
23.     CClient( std::string client_name, double soldeinit=0.);
24.     CClient( CClient const & client);
25.     CCompte &   donner_id_compte(void) ;
26.     ~CClient(void);
27. };

28. class CEmploye : public CPersonne
29. { public:
30.     ~CEmploye(void)
31.     {         std::cout<<"+++ Employe : [ "<<name<<" ] disparu +++"<<std::endl;     };
32.     CEmploye( void ):CPersonne()
33.     {         std::cout<<"*** Personne : [ "<<name<<" ] devenue employe ***"<<std::endl;
    };

34.     CEmploye( std::string employe_name);
35.     CEmploye( CEmploye const & employe);

36.     double solde_client( CClient  &client) const;
37.     void virement_compte( CClient &  client_dest, CClient &  client_src, double const somme);
38.     void operation_bancaire( CClient &  client, double const somme);
39. };
```

Heritage de classes

```
1.  CPersonne:: CPersonne( std::string pers_name,
    int pers_id)
2.  {
3.      name=pers_name;   id = pers_id;
4.      std::cout<<"*** nouvelle personne cree : [
    "<<name<<" ] ***"<<std::endl; }

5.  CPersonne:: CPersonne( CPersonne const &
    autrepersonne)
6.  {
7.      name=autrepersonne.name;      id =
    autrepersonne.id;
8.      std::cout <<"*** nouvelle personne cree par
    copie : [ "<<name<<" ] ***"<< std::endl;
9.  }

10. CPersonne:: ~CPersonne( void )
11. {     std::cout<<"+++ Personne : [ "<<name<<" ]
    disparue +++"<<std::endl; }

12. std::string  CPersonne::  getName( void ) const
13. {     return name; }

14. void  CPersonne:: setName( std::string newname)
15. {     name=newname; }
16. int  CPersonne:: getID( void ) const
17. {     return id; }
```

```
18. CClient::CClient(std::string client_name,
    double soldeinit):CPersonne(client_name) {
19.     name=client_name;
20.     ComptePerso.placer(soldeinit);
21.     std::cout<<"*** Personne : [ "<<name<<" ]
    devenue client ***"<<std::endl;
22. }

23. CClient::CClient( CClient const &
    client):CPersonne( client ) {
24.     name=client.name;
25.     id=client.id;
26.     ComptePerso.placer(
    client.ComptePerso.recup_solde() );
27.     std::cout<<"*** Client : [ "<<name<<" ]
    cree par copie ***"<<std::endl;
28. }


29. CCompte &   CClient::donner_id_compte(void)
30. {     return ComptePerso; };

31. CClient::~CClient( void )
32. {     std::cout<<"+++ Client : [ "<<name<<" ]
    disparue +++"<<std::endl;
33. }
```

Heritage de classes

```
1.  #include <iostream>
2.  #include "CEmploye_v1.h"

3.  using namespace std;

4.  int main(int argc, char* argv[])
5.  {
6.       CEmploye employe1;
7.      CClient client1("Francois");
8.
9.      employe1.operation_bancaire(
    client1, 42);
10.     employe1.solde_client( client1 );
11.
12.     CClient client2;
13.     employe1.operation_bancaire(
    client2, 30);
14.     employe1.virement_compte(client1,
    client2, 15);
15.     employe1.solde_client( client1);
16.     employe1.solde_client( client2);
17.
20.     CPersonne p1( "toto«  );
21.     CPersonne p2=p1;
22.     cout<< "personne 2 " <<p2.getName()
    <<std::endl;
23.     p1.setName("tata");
24.     cout <<"personne 2 " <<p2.getName()
    <<std::endl;

25.     employe1.operation_bancaire( client1,42);

26.     CClient client3(client1);
27.     employe1.solde_client( client1);
28.     employe1.solde_client( client3);

29.     return 0;
30. }
```

Heritage de classes

1. *** nouvelle personne cree : [JohnDoe] ***
2. *** nouvelle personne cree : [Francois] ***
3. *** constructeur de la classe CCompte par defaut ***
4. *** Personne : [Francois] devenue client ***
5. solde du client [Francois] :42
6. *** nouvelle personne cree : [JohnDoe] ***
7. *** constructeur de la classe CCompte par defaut ***
8. *** Personne : [JohnDoe] devenue client ***
9. solde du client [Francois] :57
10. solde du client [JohnDoe] :15
11. *** nouvelle personne cree : [toto] ***
12. *** nouvelle personne cree par copie : [toto] ***
13. personne 2 toto
14. personne 2 toto
15. *** nouvelle personne cree par copie : [Francois] ***
16. *** constructeur de la classe CCompte par defaut ***
17. *** Client : [Francois] cree par copie ***
18. solde du client [Francois] :99
19. solde du client [Francois] :99
20. +++ Client : [Francois] disparue +++
21. ++++ destructeur de la classe CCompte ++++
22. +++ Personne : [Francois] disparue +++
23. +++ Personne : [toto] disparue +++
24. +++ Personne : [tata] disparue +++
25. +++ Client : [JohnDoe] disparue +++
26. ++++ destructeur de la classe CCompte ++++
27. +++ Personne : [JohnDoe] disparue +++
28. +++ Client : [Francois] disparue +++
29. ++++ destructeur de la classe CCompte ++++
30. +++ Personne : [Francois] disparue +++
31. +++ Employe : [JohnDoe] disparue +++
32. +++ Personne : [JohnDoe] disparue +++

Heritage et Constructeurs

```
1.  CPersonne:: CPersonne( std::string pers_name="JohnDoe", int pers_id=0)
2.  {
3.      name=pers_name;   id = pers_id;
4.      std::cout<<"*** nouvelle personne cree : [ "<<name<<" ] ***"<<std::endl; }

5.  CPersonne:: CPersonne( CPersonne const & autrepersonne)
6.  {
7.      name=autrepersonne.name;      id = autrepersonne.id;
8.      std::cout<<"*** nouvelle personne cree par copie : [ "<<name<<" ] ***"<<std::endl;
9.  }
```

```
10. class CClient:public CPersonne
11. {
12. private:
13.     CCompte ComptePerso;

14. public:
15.     CClient( void ):CPersonne()     {
16.         std::cout<<"*** Personne : [ "<<name<<" ] devenue client ***"<<std::endl; };

17.     CClient( std::string client_name, double soldeinit=0 ) : CPersonne( client_name ) {
18.         name=client_name;
19.         ComptePerso.placer( soldeinit );
20.         std::cout<<"*** Personne : [ "<<name<<" ] devenue client ***"<<std::endl; };

21.         CClient( CClient const & client): CPersonne( client ) {
22.         name=client.name;
23.         std::cout<<"*** Client : [ "<<name<<" ] cree par copie ***"<<std::endl; };
24.
25.     CCompte  &  donner_id_compte( void ) ;

26. };
```

Heritage et Constructeurs

```
1.  #include <iostream>
2.  #include "CEmploye_v2.h"

3.  using namespace std;

4.  int main( int argc, char* argv[])
5.  {
6.       CEmploye employe1;
7.      CClient client1("Francois");
8.
9.      employe1.operation_bancaire(
    client1, 42);
10.     employe1.solde_client( client1);
11.
12.     CClient client2;
13.     employe1.operation_bancaire(
    client2, 30);
14.     employe1.virement_compte( client1,
    client2, 15);
15.     employe1.solde_client( client1 );
16.     employe1.solde_client( client2 );
17.
```

```
20.     CPersonne p1("toto");
21.     CPersonne p2=p1;
22.     cout <<"personne 2
    "<<p2.getName()<<std::endl;
23.     p1.setName("tata");
24.     cout <<"personne 2
    "<<p2.getName()<<std::endl;

25.     employe1.operation_bancaire( client1, 42);

26.     CClient client3( client1 );
27.     employe1.solde_client( client1 );
28.     employe1.solde_client( client3 );

29.     return 0;
30. }
```

Heritage de classes

1. *** nouvelle personne cree : [JohnDoe] ***
2. *** nouvelle personne cree : [Francois] ***
3. *** constructeur de la classe CCompte par defaut ***
4. *** Personne : [Francois] devenue client ***
5. solde du client [Francois] :42
6. *** nouvelle personne cree : [JohnDoe] ***
7. *** constructeur de la classe CCompte par defaut ***
8. *** Personne : [JohnDoe] devenue client ***
9. solde du client [Francois] :57
10. solde du client [JohnDoe] :15
11. *** nouvelle personne cree : [toto] ***
12. *** nouvelle personne cree par copie : [toto] ***
13. personne 2 toto
14. personne 2 toto
15. *** nouvelle personne cree par copie : [Francois] ***
16. *** constructeur de la classe CCompte par defaut ***
17. *** Client : [Francois] cree par copie ***
18. solde du client [Francois] :99
19. solde du client [Francois] :99
20. +++ Client : [Francois] disparue +++
21. ++++ destructeur de la classe CCompte ++++
22. +++ Personne : [Francois] disparue +++
23. +++ Personne : [toto] disparue +++
24. +++ Personne : [tata] disparue +++
25. +++ Client : [JohnDoe] disparue +++
26. ++++ destructeur de la classe CCompte ++++
27. +++ Personne : [JohnDoe] disparue +++
28. +++ Client : [Francois] disparue +++
29. ++++ destructeur de la classe CCompte ++++
30. +++ Personne : [Francois] disparue +++
31. +++ Employe : [JohnDoe] disparue +++
32. +++ Personne : [JohnDoe] disparue +++

Classe CEmploye

```
1.  class CEmploye: public CPersonne {
2.  public:

3.      void operation_bancaire( CClient &  client, double const somme)   {
4.          CCompte & comptetemp = client.donner_id_compte();
5.          if (somme>0.)
6.              comptetemp.placer( somme );
7.          else
8.              comptetemp.retirer( somme );
9.      };

10.     void virement_compte( CClient &  client_dest, CClient &
    client_src, double const somme)     {
11.         CCompte & comptetemp_dest = client_dest.donner_id_compte();
12.         CCompte & comptetemp_src = client_src.donner_id_compte();
13.         comptetemp_dest.virement( comptetemp_src,somme );
14.     };

15.     double solde_client( CClient & client)   {
16.         CCompte &comptetemp = client.donner_id_compte();
17.         std::cout<<"solde du client [ "<<client.getName() <<" ]
    :"<<comptetemp.recup_solde()<<std::endl;
18.         return comptetemp.recup_solde();
19.     };
■   };
```

Masquage de Noms

```
1.  // POO_banque_Herit_v3.cpp :  .
2.  #include <iostream>
3.  #include "CEmploye_v3.h"

4.  using namespace std;

5.  int main( int argc, char* argv[] )
6.  {
7.      CPersonne personne1("Gerard");
8.      CEmploye employe1("Roger");
9.      CClient client1("Francois");
10.
11.     cout<<personne1.getName()<<endl;
12.     cout<<employe1.getName()<<endl;
13.     cout<<client1.getName()<<endl;
14.     cout<<" ****utilisation des pointeurs
    ****"<<std::endl;
15.     CPersonne *ppers1=&personne1;
16.     CPersonne *ppers2=&employe1;
17.
18.     cout<<ppers1->getName()<<endl;
19.     cout<<ppers2->getName()<<endl;

20.     cout<<" ****utilisation des references
    ****"<<std::endl;
21.     CPersonne &rpers1=personne1;
22.     CPersonne &rpers2=employe1;

23.     cout<<rpers1.getName()<<endl;
24.     cout<<rpers2.getName()<<endl;


25.     return 0;
26. }
```

Masquage de Noms

- *** nouvelle personne cree : [Gerard] ***
- *** nouvelle personne cree : [Roger] ***
- *** Personne : [Roger] devenue employe ***
- *** nouvelle personne cree : [Francois] ***
- *** constructeur de la classe CCompte par defaut ***
- *** Personne : [Francois] devenue client ***
- nom de la personne : Gerard
- nom de l'employe : Roger
- nom du client : Francois
- ****utilisation des pointeurs ****
- nom de la personne : Gerard
- nom de la personne : Roger
- ****utilisation des references ****
- nom de la personne : Gerard
- nom de la personne : Roger
- +++ Client : [Francois] disparue +++
- ++++ destructeur de la classe CCompte ++++
- +++ Personne : [Francois] disparue +++
- +++ Employe : [Roger] disparue +++
- +++ Personne : [Roger] disparue +++
- +++ Personne : [Gerard] disparue +++

Masquage de Noms

```
1.  class CPersonne
2.  {
3.  protected:
4.      int id;
5.      std::string name;

6.  public:
7.      CPersonne( std::string pers_name="JohnDoe",int pers_id=0);

8.      CPersonne( CPersonne const & autrepersonne);

9.      ~CPersonne( void);

10.     virtual std::string  getName( void ) const
11.     {   return "nom de la personne : "+name; };

12.     void  setName( std::string newname)
13.     {         name=newname;     };

14.     int  getID( void) const
15.     {    return id;  }
16. };
```

Masquage de Noms

- *** nouvelle personne cree : [Gerard] ***
- *** nouvelle personne cree : [Roger] ***
- *** Personne : [Roger] devenue employe ***
- *** nouvelle personne cree : [Francois] ***
- *** constructeur de la classe CCompte par defaut ***
- *** Personne : [Francois] devenue client ***
- nom de la personne : Gerard
- nom de l'employe : Roger
- nom du client : Francois
- ****utilisation des pointeurs ****
- nom de la personne : Gerard
- nom de l'employe : Roger
- ****utilisation des references ****
- nom de la personne : Gerard
- nom de l'employe : Roger
- +++ Client : [Francois] disparue +++
- ++++ destructeur de la classe CCompte ++++
- +++ Personne : [Francois] disparue +++
- +++ Employe : [Roger] disparue +++
- +++ Personne : [Roger] disparue +++
- +++ Personne : [Gerard] disparue +++

Tableaux dynamiques

```
1.  // bad_alloc.cpp
2.  #include<new> //pour la gestion des exceptions
3.  #include<iostream>
4.  using namespace std;

5.  int main() {
6.    int i;
7.    int * p=NULL;
8.    cout << "Combien d'elements? ";
9.    cin >> i;
10.    try {
11.       p = new int[i];
12.       for (int n=0; n<i; n++)
13.       {
14.           cout << "valeur: ";
15.           cin >> p[n];
16.        }
18.       cout << "vous avez tape: ";
19.        for (int n=0; n<i; n++)
20.            cout << p[n] << ", " ;
21.        delete[] p;
22.    }
23.    catch( bad_alloc &ba) {
24.       cout << ba.what( ) << endl;
25.   }
26.    return 0;
27. }
```

```
Combien d'elements? 4
valeur: 75
valeur : 436
valeur : 1067
valeur : 8
Vous avez tape: 75, 436, 1067, 8,
```

Tableaux via vectors

```
1.  // bad_alloc.cpp
2.  #include<new> //pour la gestion des exceptions
3.  #include<iostream>
4.  #include<vector>
5.  using namespace std;

6.  int main() {
7.    int i;
8.    cout << "Combien d'elements? ";
9.    cin >> i;
10.   vector<int> p(i);
11.      for (int n=0; n<i; n++)
12.      {
13.         cout << "valeur: ";
14.         cin >> p[n];
15.       }
18.      cout << "vous avez tape: ";
19.       for (int n=0; n<i; n++)
20.           cout << p[n] << ", " ;
21.
22.   return 0;
23. }
```

```
Combien d'elements? 4
valeur: 75
valeur : 436
valeur : 1067
valeur : 8
Vous avez tape: 75, 436, 1067, 8,
```

Tableaux

```
1.  // POO_banque_tableaux.cpp
2.  #include <iostream>
3.  #include "CCompte_vf.h"
4.  #include <vector>

5.  int main(int argc, char* argv[]) {
6.      CCompte compte1[2]; //version statique
7.      CCompte compte1b[2]={CCompte(10),CCompte(11)}; //version statique

8.      std::cout<<"le compte 1[0] a un solde de " << compte1[0].recup_solde() <<std::endl;
9.      std::cout<<"le compte 1[1] a un solde de " << compte1[1].recup_solde() <<std::endl;
10.     std::cout<<"le compte 1b[0] a un solde de " << compte1b[0].recup_solde() <<std::endl;
11.     std::cout<<"le compte 1b[1] a un solde de " << compte1b[1].recup_solde() <<std::endl;

12.     CCompte *compte2=NULL; //version dynamique
13.     compte2=new CCompte[8];

14.     std::cout<<"le compte 2[0] a un solde de " << compte2[0].recup_solde()<<std::endl;

15.     delete []compte2;

16.     std::vector<CCompte> v1(10,-2); //version dynamique via STL
17.     std::cout<<"le compte v1[0] a un solde de " << v1[0].recup_solde()<<std::endl;
18.     std::cout<<"le compte v1[1] a un solde de " << v1.at(1).recup_solde()<<std::endl;
19.     return 0;
20. }
```

Tableaux

1. *** constructeur de la classe CCompte par defaut ***
2. *** constructeur de la classe CCompte par defaut ***
3. *** constructeur de la classe CCompte avec argument ***
4. *** constructeur de la classe CCompte avec argument ***
5. **le compte 1[0] a un solde de 0**
6. **le compte 1[1] a un solde de 0**
7. **le compte 1b[0] a un solde de 10**
8. **le compte 1b[1] a un solde de 11**
9. *** constructeur de la classe CCompte par defaut ***
10. *** constructeur de la classe CCompte par defaut ***
11. *** constructeur de la classe CCompte par defaut ***
12. *** constructeur de la classe CCompte par defaut ***
13. **le compte 2[0] a un solde de 0**
14. ++++ destructeur de la classe CCompte ++++
15. ++++ destructeur de la classe CCompte ++++
16. ++++ destructeur de la classe CCompte ++++
17. ++++ destructeur de la classe CCompte ++++
18. *** constructeur de la classe CCompte avec argument ***
19. *** constructeur de la classe CCompte par copie ***
20. *** constructeur de la classe CCompte par copie ***
21. *** constructeur de la classe CCompte par copie ***
22. *** constructeur de la classe CCompte par copie ***
23. ++++ destructeur de la classe CCompte ++++
24. **le compte v1[0] a un solde de -2**
25. **le compte v1[1] a un solde de -2**
26. ++++ destructeur de la classe CCompte ++++
27. ++++ destructeur de la classe CCompte ++++
28. ++++ destructeur de la classe CCompte ++++
29. ++++ destructeur de la classe CCompte ++++
30. ++++ destructeur de la classe CCompte ++++
31. ++++ destructeur de la classe CCompte ++++
32. ++++ destructeur de la classe CCompte ++++
33. ++++ destructeur de la classe CCompte ++++