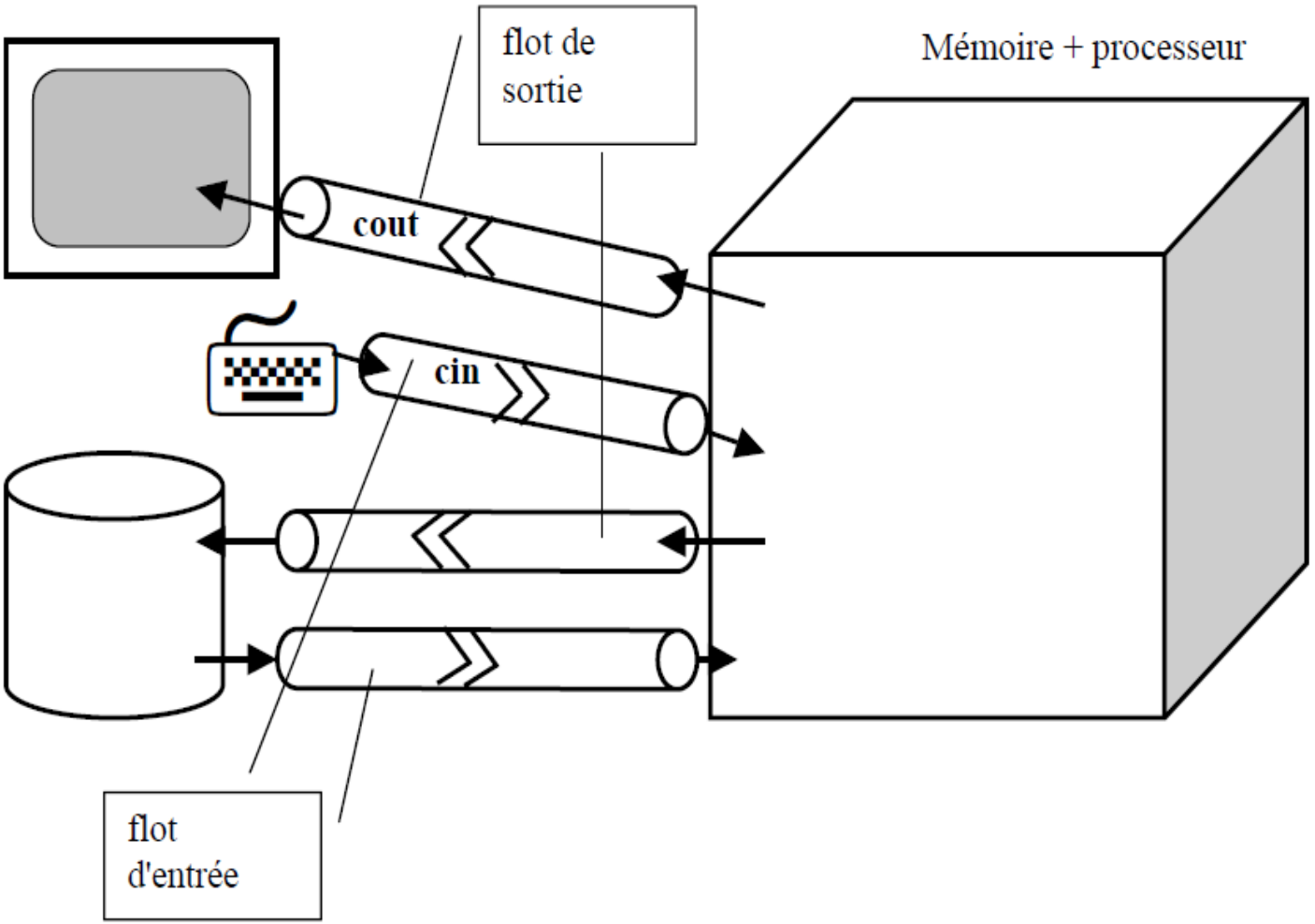
Séance 5

Gestion des entrées-sorties
Gestion de fichier

architecture



Rappel en C

Entête à inclure
#include <stdio.h> // <cstdio> en C++

Structure FILE * et variable stdin, stdout et stderr
FILE * stdin;
FILE * stdout;
FILE * stderr;

stdin : ce flot correspond au flux standard d'entrée de l'application. Par défaut, ce flux est associé au clavier(Quelques fonctions utilisent implicitement ce flux (scanf, par exemple).

stdout : c'est le flux standard de sortie de votre application. Par défaut, ce flux est associé à la console d'où l'application à été lancée. Quelques fonctions utilisent implicitement ce flux (printf, par exemple).

stderr : ce dernier flux est associé à la sortie standard d'erreur de votre application. Tout comme stdout, ce flux est normalement redirigé sur la console de l'application.

Exemple

```c
#include <stdio.h>
#include <stdlib.h>

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

    /* Cet exemple affiche le contenu d'un fichier sur la console */

    FILE * inputFile;

    argc--;   argv++;

    if ( argc == 0 ) {
        printf( "Usage: sample filename...\n" );
        exit( 0 );
    }

    inputFile = fopen( argv[0], "r" );
    if ( inputFile == NULL ) {
        printf( "Cannot open file %s\n", argv[0] );
        exit( 0 );
    }

    while ( ! feof( inputFile ) ) {
        fputc( fgetc( inputFile ), stdout );
    }

    fclose( inputFile );

    return 0;
}
```

Généralité sur les flots

- Les entrées et sorties en C++ se font toujours par l'intermédiaire de **flots (ou flux) :**
  - **cout** : flot de sortie standard, connecté par défaut à l'écran
  - **cin** : flot d'entrée standard, connecté par défaut au clavier
  - **cerr** : flot standard pour les erreurs, connecté par défaut à l'écran
- Les opérateurs :
  - **<< :** écriture sur un flot de sortie
  - **>> :** lecture sur un flot d'entrée

Entrée/sortie avec les classes

Fichier fraction.h

```
#include <iostream>

class Fraction {
    private :
        int numerateur = 0;
        int denominateur = 1;
    public :
        // ...
        friend ostream& operator<<(ostream& out, const Fraction & f);
        friend istream& operator>>(istream& in, Fraction & f);
};
```

Opérateur de sortie <<

Fichier main.cpp

```
ostream& operator<<(ostream& out, const Fraction & f) {
    out << f.numerateur  << "/" << f.denominateur  << endl;
    return out;
}
void main(){
Fraction f(5,6);
cout << f;                                  //  afficher 5/6
}
```

Opérateur d'entrée >>

```
Fraction f;
cout << "Entrez une fraction  :  " ;
cin >> f;                    //  format n/d
```

Opérateur d'entrée >>

- Vérifier le format (n/d) et gérer les erreurs :

```
istream& operator>>(istream& in, Fraction & f) {
        char c;
        if (!(in >> f.numerateur >> c >> f.denominateur) || (c!='/')) {
                in.setstate(ios::failbit);
        }
        return in;
}
```

Failbit : Logical error on i/o operation

Entrée/sortie avec les fichiers

- **Fichier physique:** collection d'octets sauvegardés sur un support physique.

- **Fichier logique:** variable liée au fichier physique, utilisée dans le programme (flux ou flot).

- Avant toute manipulation, un fichier doit être ouvert.
- A la fin des traitements du fichier, il doit être fermé.

Entrée/sortie avec les fichiers

#include <fstream>

Dans <fstream>, la librairie standard fournit :

- ofstream : pour écrire sur les fichiers
- ifstream : pour lire à partir de fichiers
- fstream : lire et écrire à partir de/vers des fichiers.

Ouverture et fermeture d'un fichier

- Ouvrir en mode lecture (constructeur ifstream)

  ifstream *nom_fichier_logique("chemin_fichier_physique");*

- Ouvrir en mode écriture (constructeur ofstream)

  ofstream *nom_fichier_logique("chemin_fichier_physique", [,**mode**]);*

- Fermer un flux (fichier)

  *nom_fichier_logique.*close();

Mode d'ouverture

ofstream *fichier_logique("fichier_physique", [,**mode**]);*

Les différents modes d'ouverture sont :

- ios::in ➔ permet la lecture (input) [1]
- ios::out ➔ permet l'écriture (output) [2]
- ios::app ➔ ajoute à la fin du fichier (append)
- ios::ate ➔ met le curseur à la fin du fichier (at end)
- ios::binary ➔ ouvre un fichier binaire (binary)
- ios::trunc ➔ vide le fichier à l'ouverture (truncate)

– Exemple

– ofstream <nom_fichier_logique> ("chemin_fichier_physique", **ios::in|ios::app**);

[1]. Mode par défaut de ifstream
[2]. Mode par défaut de ofstream

Ecriture dans un fichier

- L'écriture dans un fichier s'effectue comme l'affichage à l'écran avec l'opérateur <<
- Toute variable ou constante de type simple (bool, char, int, float, double, string, ...) peut être écrite.

- Syntaxe :

```
fichier_logique << variable;
```

Exemple

Fichier test.cpp

```
Fraction f1(1,2);
Fraction f2(2,3);
ofstream ofs ( "Fractions.txt",  ios::out);  //mode ecriture
ofs << f1;
ofs << f2;
// ...
ofs.close();
```

```
$> cat Fractions. txt
1/2
2/3
```

Lecture d'un fichier

- *ifstream fichier_logique("fichier_physique");*

- La lecture dans un fichier s'effectue avec l'opérateur >>
- Toute variable ou constante de type simple (bool, char, int, float, double, string, ...) peut être lue.

- Syntaxe

fichier_logique >> variable;

- La lecture s'arrête au premier espace ou au premier caractère qui ne peut pas faire partie de la représentation ASCII du type lu.

Lecture d'un fichier

- La fonction qui permet de lire un fichier texte ligne par ligne : getline()

- Syntaxe :

bool getline (ifstream &, string & line)

  - **true** : il reste encore des données
  - **false** : c'est la fin du fichier

Manipulation d’un fichier

- Il est conseillé avant toute opération sur le flux de tester les indicateurs d'état en appelant une de ces méthodes :
  - bad() : renvoie true si une opération de lecture ou d'écriture échoue (exemple : écrire dans un fichier qui n'est pas ouvert en écriture)
  - fail() : retourne true dans les mêmes cas que bad(), mais aussi dans le cas où une erreur de format se produit (exemple : lecture/écriture d’un caractère au lieu d’un entier)
  - eof() : renvoie true si un fichier ouvert en lecture a atteint la fin.
  - good() : c'est le drapeau d'état le plus générique. Il retourne false lorsque les méthodes précédentes retourneraient true.

Exemple

```
ifstream ifs("Fractions.txt");
if (ifs) {
        Fraction f;
        while (!ifs.eof()) {
            ifs >> f;
            if (!ifs.fail()) {
                cout << f;
            }
        }
        ifs.close();
}
else {
    cerr << "Impossible d'ouvrir le fichier " << endl;
}
```

Position dans le fichier

- Pour connaître la position courante:
  - S'il est ouvert avec ifstream : tellg().
  - S'il est ouvert avec ofstream: tellp().

➔ Ces deux méthodes renvoient la position courante dans le fichier (le numéro de l'octet courant depuis le début du fichier).

Position dans le fichier

- Pour se déplacer à une position précise dans le fichier :
  - S'il est ouvert avec ifstream : seekg(pos,mode)
  - S'il est ouvert avec ofstream : seekp(pos,mode)

➔ pos: le numéro d'octet où se positionner

➔ mode:

- ios::beg (octet indiqué depuis le début du fichier)
- ios::cur (octet indiqué depuis la position courante dans le fichier)
- ios::end (octet indiqué depuis la fin du fichier)

Exemple (2 programmes)

P1, saisit et mémorise dans un fichier nommé répertoire, le nom et le numéro de téléphone de 10 personnes

```
# include <fstream> //ne pas inclure iostream !!
#include <iostream>
void main ( )
{ // déclaration de variables communication entre le fichier et la mémoire centrale
   string nom, tel;
   ofstream fichrep ("repertoire.txt"); // ouverture du répertoire en écriture
   for(int i=0; i<10; i++)    {
       cout << "\npersonne "<< i+1 << ":\n"
       cout << "nom? ";
       cin >> nom;
       fichrep <<nom << " ";
       cout << "\ntelephone?";
       cin >> tel;
       fichrep << tel << "\n";
    }
   fichrep.close();
}
```

Exemple (2 programmes)

P2 permet d'afficher à l'écran le contenu du fichier répertoire, en sautant une ligne entre chaque personnes

```
# include <fstream>
#include <iostream>
# include <conio.h>
# include <cstring>
void main( ) {
   string nom;
   string tel;
   ifstream fichrep ("repertoire.txt"); // ouverture du fichier en lecture
   fichrep >> nom >> tel;
   while (! fichrep.eof( ) ) // tant qu'on est pas arrivé à la fin du fichier…
  {
      cout << nom << " \t"<< tel << "\n"; // …on affiche
      fichrep >> nom >> tel; // et on lit les informations suivantes
   }
   fichrep.close( );
   getch( );
}
```