Fichiers et répertoires

Introduction :

Un programme a souvent besoin d'échanger des informations, que ce soit pour recevoir des données d'une source ou pour envoyer des données vers un destinataire. On a aussi besoin de gérer les exceptions pour empêcher le programme de crasher suite à un pointeur « null » mais aussi pouvoir comprendre ce qui se passe dans notre programme. Ce document vous permettra d'avoir une idée sur la manipulation de ces derniers.

Fichiers et répertoires :

Pour facilité la tâche aux utilisateurs nous avons choisi de créer une classe file pour faciliter la manipulation des fichiers et répertoires. Le tableau ci-dessous présente les méthodes applicable aux instances de la classe file avec une petite description. Nous reviendrons sur chaque méthode pour expliquer comment ça marche.

Classe file		
Méthode	**Type de retour**	**Description**
file(path is **string**)	void	Constructeur.
is_file	boolean	Retourner true si l'instance de la classe file est un fichier.
is_directory	boolean	Retourner true si l'instance de la classe file est un dossier.
make_directory	boolean	Retourne true si le dossier est créée.
APPLICABLE AUX FICHIERS UNIQUEMENT		
open	boolean	Permet d'accéder au fichier.
is_empty	boolean	Retourne vrai si le fichier est vide
get_text	string	Retourne le contenue du fichier.
get_text(*begin is* ***string,*** *end is* ***string***)	string	Retourne le contenue du fichier entre deux chaînes de caractères.
get_text(pattern is regex)	vector of string	Retourne une liste des mots qui matchent avec l'expression régulière.

get_nbr_lines	integer	Retourne le nombre de lignes dans un fichier.
get_line(*line_number is* ***integer*****)**	vector of file	Retourne le contenue d'une ligne « line_number » du fichier.
get_line(*line_number is* ***integer***, *delimiter is* ***string***)	vector of 2 (or 1 if no delimiter found) string	Retourne un vector d'une ligne « line_number » coupé en 2 selon le délimiteurs.
write(*content is* ***string***)	boolean	Écrit le contenu du string dans le fichier et retourne true si c'est fait.
size	integer	Retourne la taille en octects du fichier.
close	boolean	Retourne true si la fermeture du fichier est bien faite.
APPLICABLE AUX DOSSIERS UNIQUEMENT		
sub_files	vector of file	Retourne la liste des fichiers dans un répertoire.
sub_directories	vector of file	Retourne la liste des répertoires dans un répertoire.

Exemples :

Code EZ	Code C++
AFFICHAGE DU CONTENU DU FICHIER	

```
program hello_file

procedure hello_file()
    // créer une instance de la classe file
    myfile is file("/path/to/file/myfile.txt")
    // ouvrir le fichier
    if myfile.open()
    then
            // afficher le contenu du fichier
            print myfile.get_text()
             myfile.close()
    else
            print "can't open this file, make sure
that your file is there"
    end
end
```

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {

  string line;
  // créer une instance de la classe file
  ifstream myfile ("/path/to/file/myfile.txt");
  // ouvrir le fichier
  if (myfile.is_open())
  {
    while ( getline(myfile,line) ) {
      // afficher le contenu du fichier
      cout << line <<endl;
    }
```

	myfile.close();
	}
	else cout << "can't open this file, make sure that your file is there" << endl;
	
	return 0;
	
	}

AFFICHAGE ET/OU ÉCRITURE DANS UN FICHIER	

program hello_file
 	
procedure hello_file()
// créer une instance de la classe file
myfile is file("/path/to/file/myfile.txt")
// ouvrir le fichier
if myfile.open()
// vérifier qu'il n'est pas vide
if !myfile.is_empty()
// itérer sur le nombre de lignes
for i in 1..myfile.get_nbr_lines()
// afficher ligne par ligne
print "ligne ", i," : " ,myfile.get_line(i)
end for
else
// écrire dans le fichier
myfile.write("this file is no longer empty")
end if
else
myfile.write("file created")
end // fin du if
end // fin de procedure
	#include <iostream>
	#include <fstream>
	#include <string>
	using namespace std;
	
	int main () {
	string line;
	int line_number=0;
	string file_name="/path/to/myfile.txt" ;
	ifstream myfile (file_name);
	if (myfile.is_open())
	{
	while (getline(myfile,line))
	{
	line_number++;
	cout << "ligne "<<line_number << " : "<< line <<endl;
	}
	// vérifier qu'il n'est pas vide
	if (line_number == 0) {
	ofstream myfile (file_name);
	if (myfile.is_open())
	{
	// écrire dans le fichier
	myfile << "this file is no longer empty";
	myfile.close();
	}
	}
	myfile.close();
	}
	else {
	fstream file;
	// créer le fichier
	file.open(file_name, std::ios::out);
	file << "file created";
	file.close();

	} return 0; }