Université Ibn Tofail
جامعة ابن طفيل
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵉⴱⵏ ⵟⴼⵢⵍ
Ibn Tofail University

POO sous C++
Exceptions

Med. AMNAI

Filière SMI - S5

Département d'Informatique

19 janvier 2021

Plan

Plan

Plan

Introduction

De nombreux problèmes peuvent survenir pendant l'exécution d'un programme :

- L'insuffisance de mémoire, la perte d'un fichier, la saisie non valide d'une valeur ;
- Le rôle d'un programme consiste à prévoir et gérer ces erreurs ;
- Informer les utilisateurs et éventuellement à mettre en oeuvre des solutions de reprise ;
- Mettre en oeuvre des correction de ces erreurs d'exécution.

Introduction (1)

- Une **exception est un problème** imprévu qui survient lors de l'exécution.
- Ne pas gérer **une exception peut causer un arrêt** de votre programme

Gestion des erreurs en utilisant des fonctions

```cpp
#include<iostream>
#include<string.h>
#include <stdlib.h>

using namespace std;

//--------------------except0.cpp--------------------------
int positive(int v){
    if(v<0) return 0;
        cout << " Valeur positive " << endl;
    return 1 ;
}
//---------------------------------------------------------
int inf(int v,int max){
    if(v>max) return 0;
        cout << " Valeur inferieure a : " << max << endl;
    return 1;
}
//---------------------------------------------------------
int sup(int v,int min){
    if(v<min) return 0;
        cout << " Valeur superieure a : " << min << endl;
    return 1;
}
```

```cpp
int main(){
    int minimum=10;
    int maximum=100;
    int n,retour;

    cout<<"Saisir une valeur entiere : "; cin >> n;
    retour=positive(n);

    if(retour==1){
        retour=inf(n,maximum);
            if(retour==1){
                retour=sup(n,minimum);
                    if(retour==1){
                        cout << " Valeur correcte ! " << endl;
                        return 1;
                    }else{
                        cout<<" Valeur inferieure a : " << minimum;
                        return 0;
                    }
            }else{
                    cout << " Valeur superieure a : " << maximum << end
                    return 0;
                }
    }else{
        cout << " Valeur negative ! " << endl;
        return 0;
    }
}
```

Mise en oeuvre des exceptions

Pour mettre en oeuvre des exceptions sans se baser sur les valeurs de retour des fonctions, on doit :

1. Définir une classe d'exception.
2. Lancer l'exception (**throw**).
3. Intercepter (**catch**) l'exception.

Lancer une exception (1)

```
//1- Définir une classe d'exception
class erreur{ };

//-----------------------

void positive(int v){
    if(v<0){
        erreur er;
        throw er; // 2- Lancer l'exception
    }
        cout<<"Valeur positive " << endl;
}

//La fonction précédente peut être simplifiée :

void positive(int v){
    if(v<0) throw erreur(); // 2- Lancer l'exception
        cout<<"Valeur positive " << endl;
}
```

Intercepter l'exception

```
try{
    //Appels de fonctions pouvant générer des erreurs
}
catch (classe d exception){ //3- Intercepter l'exception
    //Ce bloc s'exécute pour une exception de type
    //'classe d'exception'
}
catch(...){
    //Ce bloc s'exécute pour toutes les exceptions
    //en dehors de la classe d'exception
}
```

Intercepter l'exception (1)

Si une exception est envoyée par une de ces fonctions appelées dans le bloc "**try**", le mécanisme d'exception entraînera les étapes suivantes :

- Tous les objets créés dans le bloc "**try**" sont détruits ;
- Le programme sort du bloc "**try**" après la fonction qui a entraîné l'exception et n'exécute pas les instructions situées après cette fonction.
- C++ exécute dans l'ordre soit le bloc **catch** correspondant à l'exception interceptée si elle existe, soit le bloc **catch(...)**.

Exemple

```
//1.                    except1.cpp
class erreur { };
//2.
int positive(int v){
    if(v<0) throw erreur();
        cout << " Valeur positive " << endl;
}
//-------------------------------------------------------
int inf(int v,int max){
    if(v>max) throw erreur();
        cout << " Valeur inferieure a : " << max << endl;
    return 1;
}
//-------------------------------------------------------
int sup(int v,int min){
    if(v<min) throw erreur();
        cout << " Valeur superieure a : " << min << endl;
    return 1;
}
```

```
//-------------------------------------------------------
int main(){
    int minimum=10; int maximum=100;
    //3.
    try
    {
        int n ;
        cout << " Saisir une valeur entiere : "; cin >> n;
        positive(n);
        inf(n,maximum);
        sup(n,minimum);
        cout << " Valeur correcte ! " << endl;

    }catch(erreur er){

            cout<< " valeur incorrecte! " << endl;
    }
    catch(...){
            cout<<"Erreur inconnue !\n";
    }
}
```

Exercice

Toutes les fonctions de l'exemple précédent utilisent la même classe d'exception **erreur**. Proposer plusieurs classes d'exception chacune responsable de l'affichage d'un message d'erreur.

Exercice (Sol)

```
//1.        exceptExo.cpp
class erreur_positive{ };
class erreur_inf{ };
class erreur_sup{ };
//-------------------------------------------------------
//2.
int positive(int v){
    if(v<0) throw erreur_positive();
        cout << " Valeur positive " << endl;
}
int inf(int v,int max){
    if(v>max) throw erreur_inf();
        cout << " Valeur inferieure a : " << max << endl;
}
int sup(int v,int min){
    if(v<min) throw erreur_sup();
        cout << " Valeur superieure a : " << min << endl;
}
```

```
//-------------------------------------------------------
int main(){
    int minimum=10; int maximum=100;
    //3.
    try
    {
        int n ;
        cout << " Saisir une valeur entiere : "; cin >> n;
        positive(n);
        inf(n,maximum);
        sup(n,minimum);
        cout << " Valeur correcte ! " << endl;

    }catch(erreur_positive er){
        cout<< " Erreur ! valeur negative ! "<<endl;
    }
    catch(erreur_inf er){
        cout<< " Erreur ! valeur superieure A :"<<maximum<<endl;
    }
    catch(erreur_sup er){
        cout<< " Erreur ! valeur inferieure A :"<<minimum<<endl;
    }
    catch(...){
        cout<< " Erreur inconnue ! " << endl;
    }
}
```

Afficher les messages l'aide des fonctions membres

```
//1. -------ExceptFctM.cpp ----------------------------------
class erreur_positive{
public :
    void affiche(){
        cout<< " Erreur ! valeur negative ! "<<endl;
    }
};
class erreur_inf{
    int maxi;
    public:
        erreur_inf(int ma){maxi=ma;}
        void affiche(){
            cout<< " Erreur ! valeur superieure A :"<<maxi<<endl;
        }
};
class erreur_sup{
    int mini;
    public:
        erreur_sup(int mi){mini=mi;}
        void affiche(){
            cout<< " Erreur ! valeur inferieure A :"<<mini<<endl;
        }
};
```

```
//2- ------------------------------------------------------
int positive(int v){
    if(v<0){
        erreur_positive er; throw er;
    }
        cout << " Valeur positive " << endl;
}
int inf(int v,int max){
    if(v>max){
        erreur_inf er(max); throw er;
    }
        cout << " Valeur inferieure a : " << max << endl;
}
int sup(int v,int min){
    if(v<min){
        erreur_sup er(min); throw er;
    }
        cout << " Valeur superieure a : " << min << endl;
}
```

Afficher les messages l'aide des fonctions membres (1)

```
int main(){
    int minimum=10; int maximum=100;
    //3.
    try
    {
        int n ;
        cout << " Saisir une valeur entiere : "; cin >> n;
        positive(n);
        inf(n,maximum);
        sup(n,minimum);
        cout << " Valeur correcte ! " << endl;

    }catch(erreur_positive e){
        e.affiche();
    }
    catch(erreur_inf e){
        e.affiche();
    }
    catch(erreur_sup e){
        e.affiche();
    }
    catch(...){
        cout<< " Erreur inconnue ! " << endl;
    }
}
```

Hiérarchie des classes d'exception (1)

```
//1. HerClsExcp.cpp ----------------------------------------
class erreur{
    public:
        virtual void affiche(){ cout << " Erreur !"<<endl;}
};
class erreur_positive:public erreur{
    public :
        void affiche(){
            cout << " Erreur ! valeur negative !"<<endl;}
};
class erreur_inf:public erreur{
    int maxi;
    public:
        erreur_inf(int ma){ maxi = ma;}
        void affiche(){
            cout<< " Erreur ! valeur superieure A :"<<maxi<<endl;}
};
class erreur_sup:public erreur{
    int mini;
    public:
        erreur_sup(int mi){ mini = mi; }
        void affiche(){
            cout<< " Erreur ! valeur inferieure A :"<<mini<<endl;}
};
```

Hiérarchie des classes d'exception (2)

```
//2. ------------------------------------------------------------
int positive(int v){
    if(v<0){
        erreur_positive *er; er=new erreur_positive ; throw er;
    }
        cout << " Valeur positive " << endl;
}
int inf(int v,int max){
    if(v>max){
        erreur_inf *er; er=new erreur_inf(max); throw er;
    }
        cout << " Valeur inferieure a : " << max << endl;
}
int sup(int v,int min){
    if(v<min){
        erreur_sup *er; er=new erreur_sup(min); throw er;
    }
        cout << " Valeur superieure a : " << min << endl;
}
```

Hiérarchie des classes d'exception (3)

```
//-------------------------------------------------------------------
int main(){
    int minimum=10; int maximum=100;
    //3.
    try
    {
        int n ;

        cout << " Saisir une valeur entiere : "; cin >> n;
        positive(n);
        inf(n,maximum);
        sup(n,minimum);
        cout << " Valeur correcte ! " << endl;

    }catch(erreur *e){
        e->affiche(); delete e ;
    }
    catch(...){
        cout<< " Erreur inconnue ! " << endl;
    }
}
```