IPSA / N.DECKNER 2024

Cours 4 - Structure d'un programme

Notions : Bibliothèque utilisateur, bibliothèque standard, espace de noms.

Programme minimal

La fonction `main` est appelée au lancement du programme et retourne 0.

`prog.cpp`

```
int main() {

    return 0;
}
```

Programme avec un seul fichier

`prog.cpp`

```
int f(int x) {
    return x + 1;
}

int main() {
    y = f(1);

    return 0;
}
```

Programme avec plusieurs fichiers

Syntaxe pour inclure une bibliothèque utilisateur:

```
#include "lib.h"
```

Dans l'exemple le programme est écrit dans 3 fichiers: `lib.cpp`, `lib.h` et `prog.cpp`.

- `lib.cpp` : Définitions de fonctions
- `lib.h` : Déclarations de fonctions
- `prog.cpp` : Utilisation des fonctions

`prog.cpp` inclut les déclarations contenues dans `lib.h` avec l'instruction `#include "lib.h"`. On peut ainsi utiliser les fonctions définies dans `lib.cpp`, depuis `prog.cpp`. Ici `f` est appelée dans `prog.cpp` :

`prog.cpp`

```
#include "lib.h"

int main() {
    int result = f();
    return 0;
}
```

`lib.h` contient la déclaration de f. les instructions pour le préprocesseur `#ifndef LIB_H`, `#define LIB_H` et `#endif` empêche le compilateur de lire plusieurs fois le fichier `lib.h` si il est inclut dans plusieurs fichiers.

`lib.h`

```
#ifndef LIB_H
#define LIB_H

int f(int x);

#endif
```

`lib.cpp` inclut les déclarations des fonctions de `lib.h` avec `#include "lib.h"`

`lib.cpp`

```
#include "lib.h"

int f(int x) {
    return x + 1;
}
```

Espaces de noms

Les espace de noms servent à organiser le code. Il n'y a pas de conflit de noms entre espaces de noms.

Syntaxe:

`lib.cpp`

```
namespace namespace_name {
    int x = 1;

    int incr(int x) {
        return x + 1;
    }
}
```

`prog.cpp`

```cpp
#include "lib.h"

int y = namespace_name::x;
int z = namespace_name::incr(2);
```

Dans le fichier .h avec l'instruction `namespace`

`lib.h`

```cpp
#ifndef LIB_H
#define LIB_H

namespace namespace_1 {
    int x;

    int f(int x);
}

#endif  // LIB_H
```

Dans le fichier .cpp avec l'instruction `namespace`

`lib.cpp`

```cpp
#include "lib.h"

namespace namespace_1 {
    int x = 1;

    int f(int x) {
        return x + 1;
    }
}
```

Utilisation avec l'opérateur `::`

`prog.cpp`

```cpp
#include <iostream>
#include "lib.h"

int main() {
    // Utilisation de la variable x de namespace_1
    std::cout << namespace_1::x << std::endl;  // -> 1

    // Utilisation de la fonction f de namespace_1
    std::cout << namespace_1::f(1) << std::endl;  // -> 2

    return 0;
}
```

Utilisation avec l'instruction `using`

Avec l'instruction `using` il n'est plus nécessaire de spécifier l'espace de nom.

`prog.cpp`

```cpp
#include <iostream>
#include "lib.h"

using std;
using namespace_1;


int main() {
    cout << x << endl;

    cout << f(1) << endl;

    return 0;
}
```

Portée des variables

La portée d'une variable est la zone de code où on peut utiliser la variable. Cette zone dépend de l'endroit où la variable est déclarée. Une variable `x` peut être déclarée:

- dans une fonction: `x` est **locale** à la fonction et peut être utilisée uniquement depuis la fonction.
- dans une **classe** à l'extérieur d'une fonction: `x` est utilisable depuis tout le code de la classe.
- dans un **espace de noms** à l'extérieur d'une classe: `x` est utilisable dans tout l'espace de noms.
- en dehors d'un espace de noms et d'une classe: La variable est **globale** et est utilisable depuis tout le programme.

La bibliothèque standard

Lien: https://en.cppreference.com/w/cpp/standard_library

Inclure une bibliothèque de la bibliothèque standard

```
#include <iostream>
```

Utilisation

Les fonctions et types de la librairie standard sont dans l'espace de nom `std`. Pour les utiliser il faut inclure les fichiers d'entêtes .h nécessaires et écrire `std::` devant leur nom ou utiliser l'instruction `using std;`.

Exemples avec `cout` de la bibliothèque standard `<iostream>`:

```
#include <iostream>

std::cout << 1;
```

```
#include <iostream>

using std

cout << 1;
```

Exemples avec le type `vector` de la bibliothèque standard `<vector>`:

```
#include <vector>

using std

vector<int> vect = {1, 2, 3};
```

Entêtes de la bibliothèque standard parmis les plus utilisés

- `<iostream>` : Pour les opérations d'entrée/sortie, comme cout et cin, et la manipulation de fichiers.
- `<vector>` : Pour utiliser le conteneur dynamique vector pour stocker des collections d'éléments.
- `<map>` : Pour utiliser un conteneur qui stocke des paires clé-valeur de manière ordonnée.
- `<string>` : Pour travailler avec des chaînes de caractères.
- `<cmath>` : Pour effectuer des opérations mathématiques comm sin et cos.
- `<fstream>` : Pour travailler avec les fichiers.
- `<algorithm>` : Pour accéder à des algorithmes de traitement de données comme le tri.
- `<cstdlib>` : Pour des opérations de gestion de mémoire, comme malloc et free.
- `<ctime>` : Pour travailler avec le temps et la date.

Hors programme:

static, extern