C++ Introduction

Namespaces

Contents

Namespaces demystified

Definition:

- Declarative region
- Provides scope to identifiers inside it

What are they used for?

- Organize code into logical groups
- Prevent name collisions (especially when working with multiple libraries)

Namespaces demystified

Namespaces demystified

Operator:

```
Namespace::identifier
```

→ Access identifier of namespace

Namespaces demystified

Operator:

```
Namespace::identifier
```

→ Access identifier of namespace

Using declaration:

```
using ns::id;
```

→ make identifier id of namespace ns available in this scope

Namespaces demystified

Operator:

```
Namespace::identifier
```

→ Access identifier of namespace

Using declaration:

```
using ns::id;
```

→ make identifier id of namespace ns available in this scope

Using directive:

```
using namespace ns;
```

→ make namespace called ns available

Example

```
#include <iostream>

namespace myLib {
  int x = 1;
  int myFunc(int n);
}

namespace otherLib {
  int x = 2;
  int myFunc(int n);
}

int main() {
  int x = 0;
  std::cout << x << myLib::x << otherLib::x << std::endl;
  return 0;
}
```

Example

```cpp
#include <string>

int main() {
  std::string str1; /* use full qualified name */
  using std::string; /* make string available */
  string str2; /* use new available name */
}
```

Example

```
#include <iostream>
#include <string>

using namespace std;

int main() {
  string str = "Hello World!";
  cout << str << endl;
}
/*--------------------------------------------*/
int main() {
  std::string str = "Hello World!";
  std::cout << str << std::endl;
}
```

Good practice

using only in implementation files

- Use them only in cpp files, NOT in header files
- Always use the full qualified identifier in header files to avoid name collision

using declarations > *using* directives

- If you only need a few identifiers from a namespace, declare them explicitly instead of importing the whole namespace
- using directives otherwise