

Projekt z przedmiotu ZPR - dokumentacja wstępna

1. Skład zespołu

- Michał Andrzejewski - kontrola jakości, dokumentacja, programista, tester
- Aleksy Barcz - właściciel projektu, koordynator, programista, tester

2. Cel projektu

Rozbudowa biblioteki Boost.Python o zestaw szablonów pozwalający przekazywać kolekcje standardowe, uwzględniająca zarówno typy wbudowane (`int`, `double`, `std::string`) jak również typy zdefiniowane przez użytkownika. Efektem pracy będzie dostarczenie mechanizmu pozwalającego przekazywać podstawowe kontenery biblioteki standardowej z C++ do Pythona oraz odwrotnie, z zachowaniem ich podstawowych funkcjonalności. Realizacja projektu odbędzie się przy wykorzystaniu istniejących możliwości biblioteki Boost.Python w wersji 1.42.0.

3. Zakres projektu

Planowane jest udostępnienie w Pythonie następujących kontenerów STL:

- `std::vector` jako Pythonowy list, udostępniający podstawowe metody takie jak:
 - `append`
 - `insert`
 - `del`
 - `operator[]`
 - `len`
 - `remove`
- `std::list` jako Pythonowy list, udostępniający podstawowe metody takie jak:
 - `append`
 - `insert`
 - `del`
 - `remove`
 - `len`
 - operacje iteratorowe
- `std::set` jako Pythonowy set, udostępniający podstawowe metody takie jak:
 - `add`

- remove
- in
- len
- clear

4. Planowany sposób realizacji

Kontenery będą przekazywane do Pythona za pomocą wrapperów, opakowujących ich metody w metody zrozumiałe dla Pythona. Do operacji wymagających iterowania po kolekcji planowane jest wykorzystanie mechanizmów *boost/python/iterator*. Do przekazywania wrappowanych obiektów z powrotem do C++ wykorzystany zostanie mechanizm *extract*. Do zwracania obiektów będzie potrzebne wykorzystanie *Call-Policies*.

Wraz z kolekcją obiektów nie będą przekazywane żadne dane na temat ich typów (w szczególności - na temat typu użytkownika, jego pól i metod składowych). Użytkownik biblioteki powinien sam zadbać o kompatybilność pomiędzy implementacją klasy i jej metod w C++ i Pythonie.

Dzięki oparciu konstrukcji biblioteki o szablony będzie możliwe łatwe rozszerzenie jej funkcjonalności na inne kontenery STL.

5. Sposób testowania/prezentacji

Prezentacja działania utworzonych szablonów odbędzie się na kontenerach następujących typów:

- int
- double
- std::string
- klasa użytkownika Foo

Zautomatyzowane testy będą obejmować:

- kontrolę poprawności przekazania kolekcji standardowej języka C++ do Pythona
- kontrolę poprawności przekazania kolekcji języka Python do C++
- kontrolę zawartości kolekcji po przekazaniu jej do Pythona, wykonaniu operacji na kontenerze tego języka (np. `append()`) i zwróceniu z powrotem do C++
- kontrolę wykrywania i obsługi sytuacji wyjątkowych