Piotr Żelazko 277373 TKOM 2018L

RASPH - smart home scripting language

Wstęp

Celem projektu jest stworzenie języka skryptowego do zastosowań inteligentnego domu. Język ten powinien być zorientowany na łatwość i szybkość implementacji modułów związanych z zarządzaniem, wysterowaniem i zbieraniem danych z różnorodnych czujników, sterowników, silniczków. Język będzie silnie związany z platformą Raspberry Pi.

Język ten będzie nosił roboczą nazwę *rasph*.

Analiza wymagań

Wymagania funkcjonalne

- Zmienne/stałe globalne oraz lokalne;
- Definiowanie zdarzeń cyklicznych;
- Importowanie modułów;
- Instrukcje warunkowe;
- Operacje logiczne;
- Operacje arytmetyczne;
- Stałe tekstowe;
- Pętle;
- Kontenery;
- Definiowanie klas:
  - Definiowanie hierarchii dziedziczenia;
  - Definiowanie atrybutów;
  - Definiowanie metod;
  - Konstruktory oraz destruktory;
- Definiowanie interfejsów (dziedziczenie po klasie System.Interface)
- Definiowanie urządzeń (dziedziczenie po klasie System.Device);
- Możliwość łączenia z bibliotekami dynamicznymi;
- Procesy powinny móc komunikować się poprzez kolejki FIFO;
- Składnia będzie umożliwiała na wykonywanie synchroniczne asynchroniczne pewnych części programu;
- Wejście oraz wyjście:
  - Umożliwienie wypisania na `stdout, stderr`;
  - Umożliwienie wczytania z `stdin`;
  - Wypisanie do połączonego urządzenia poprzez zdefiniowany interfejs;
  - Odbiór danych z podłączonego urządzenie poprzez zdefiniowany interfejs;
- Mechanizm wyjątków.

Wymagania niefunkcjonalne

- Folder projektu powinien umożliwiać ustrukturyzowanie;
- rasph powinien być kompletny w sensie Turinga;
- Brak silnego typowania;
- rasph powinien dostarczać bibliotekę standardową;
- rasph powinien dostarczać mechanizm zarządzania pamięcią;
- Dodanie nowego urządzenia powinno wiązać się zaledwie z dodaniem pliku konfiguracyjnego oraz biblioteki dynamicznej umożliwiającej obsługę tego urządzenia;
- Nazwy zmiennych, klas, interfejsów itd. powinny rozpoczynać się od liter (a-z, A-Z).

Obsługa błędów

Informacje o błędach wykonania będą się przekierowane do `/dev/stderr` oraz/lub logowane do pliku logów. Nieobsłużony wyjątek będzie powodował zakończenie programu. Błąd na etapie parsowania spowoduje zakończenie programu z odpowiednim komunikatem.

Specyfikacja i składnia

Opis gramatyki oraz składni znajduje się w osobnych plikach.

Sposób realizacji

Projekt zrealizuję w języku C++17 z wykorzystaniem biblioteki boost. Z biblioteki boost wykorzystam moduły Boost.Test oraz Spirit.Lex do generacji leksera.

Moduły

Projekt podzielę na moduły i wystawię pomiędzy nimi interfejsy, tak aby każdy z modułów mógł być wymieniony bez ingerencji w pozostałe. Prawdopodobnie każdy z modułów będzie osobnym programem i procesy będą się komunikowały między sobą za pomocą mechanizmów komunikacji dostępnych w Unix'ie.

Proponuję podział na poniższe moduły :

- Analizator leksykalny - wyodrębnianie znaków i grupowanie ich w tokeny;
- Analizator składniowy - grupowanie tokenów w struktury;
- Analizator semantyczny - będzie pobierał struktury składniowe od analizatora składniowego i sprawdzał ich znaczenie w języku rasph;
- Zarządca tablicy symboli - będzie śledził nazwy użyte w programie i będzie odpowiedzialny za przechowywanie wartości zmiennych w trakcie działania programu, prawdopodobnie podmoduł interpretera;
- Interpreter - interpreter będzie interfejsem pomiędzy użytkownikiem a ww. modułami.

Tokeny

Tokeny zdefiniowane zostały w pliku `tokens` oraz `keywords`.

Uruchamianie i testowanie

Sposoby uruchamianie

Projekty napisane w języku rasph będą mogły zostać uruchomione na dwa sposoby.

Pierwszym ze sposobów będzie uruchomienie interpretera komend. Drugi natomiast będzie umożliwiał uruchomienie projektu jako daemon'a i komunikację z nim poprzez mechanizm kolejek FIFO.

Testowanie

Testowanie projektu będzie się odbywało poprzez uruchomienie testów jednostkowych napisanych we framework'u Boost.Test. Do testowania funkcjonalności użyję Raspberry Pi B2, ESP8266, termometru DS18B20 oraz diody LED.

Przykłady

Przykłady użycia znajdują się w osobnych plikach i tworzą jeden spójny projekt.

Celem przykładowego projektu jest przedstawienie zasad użycia języka rasph; na podstawie przykładowego projektu tworzona będzie biblioteka standardowa. W etapie końcowym projekt ten będzie przykładem działającej aplikacji.

Zaznaczyć jednak trzeba, że przykładowy projekt może ulec zmianie w trakcie implementacji projektu semestralnego na TKOM.