Tekst projektnog zadatka

Napisati biblioteku za rad sa regularnim jezicima koji se reprezentuju konačnim automatima (DKA i ε-NKA) i/ili regularnim izrazima. Realizovati sljedeće funkcionalnosti za rad sa regularnim jezicima: Izvršavanje konačnih automata (DKA i ε-NKA) formiranih od strane korisnika biblioteke. Konstrukcije reprezentacija unije, presjeka, razlike ispajanja jezika, komplementa jezika irezultata primjene Kleenove zvijezde nad jezikom, uz podršku za ulančavanje operacija. Određivanje dužina najkraće i najduže riječi u jeziku, kao i ispitivanje konačnosti jezika. Minimizacija broja stanja za determinističke konačne automate.Transformacija ε-NKA u DKA, kao i transformacija regularnog izraza u konačni automat. Poređenje reprezentacija regularnih jezika, uključujući regularne izraze, po jednakosti jezika. Napisati aplikaciju koja učitava specifikaciju reprezentacije regularnog jezika (podržane reprezentacije su DKA, ε-NKA i regularni izraz) i koja ispituje pripadnost specificiranih stringova reprezentovanom jeziku. Izvršiti leksičku analizu specifikacije reprezentacije regularnog jezika i u slučaju leksičkih nepravilnosti evidentirati broj relevantnih linija specifikacije koje sadrže nepravilnosti. Napisati aplikaciju za generisanje programskog koda za simulaciju mašine stanja na osnovu specifikacije determinističkog konačnog automata. Omogućiti da, za svako formirano stanje, korisnici generisnog koda mogu da specificiraju reakciju na događaje ulaska u stanje i izlaska iz stanja. Omogućiti da korisnici generisanog koda za svaki simbol alfabeta automata mogu da specificiraju reakciju na izvršavanje prelaza za taj simbol, a koja može da zavisi od stanja iz kojeg se vrši prelaz. Treba biti omogućeno nadovezivanje reakcija, tako da se efektivno može formirati lanac reakcija na neki događaj. Pridržavati se principa OOP, principa pisanja čistog koda, DRY principa, principa ponovne upotrebe koda, SOLID principa i konvencija za korišteni programski jezik. Pravilno dokumentovati kod upotrebom komentara. Obratiti pažnju na performanse po pitanju vremena izvršavanja i zauzeća memorije pri pisanju i odabiru algoritama i struktura podataka, te dokumentovati asimptotsku kompleksnost istih. Pravilno pokriti funkcionalnosti iz ove specifikacije sa jediničnim testovima. Dozvoljeno je korištenje standardne biblioteke, ali ne i nestandardnih biblioteka, za odabrani programski jezik.

Text of the project task

Write a library for working with regular languages represented by finite automata (DKA and ε-NKA) and/or regular expressions. Realize the following functionalities for working with regular languages: Execution of finite automata (DKA and ε-NKA) formed by users of the library. Constructions of representations of union, intersection, disjunction of languages, complements of languages, and the result of applying Kleen's star over a language, with support for chaining operations. Determining the lengths of the shortest and longest words in the language, as well as testing the finality of the language. Minimization of the number of states for deterministic finite automata. Transformation of ε-NKA into DKA, as well as transformation of a regular expression into a finite automaton. Comparison of representations of regular languages, including regular expressions, by language equality. Write an application that loads the specification of the representation of a regular language (supported representations are DKA, ε-NKA and regular expression) and that examines whether the specified strings belong to the represented language. Perform a lexical analysis of the specification of the representation of the regular language and, in case of lexical irregularities, record the number of relevant lines of the specification that contain irregularities. Write an application to generate program code to simulate a state machine based on the specification of a deterministic finite state machine. Enable that, for each formed state, users of the generic code can specify the reaction to state entry and exit events. Enable users of the generated code for each symbol of the automaton alphabet to specify the reaction to the execution of the transition for that symbol, which may depend on the state from which the transition is made. It should be possible to link reactions, so that a chain of reactions to an event can be effectively formed. Adhere to OOP principles, clean code writing principles, DRY principles, code reuse principles, SOLID principles and conventions for the used programming language. Document code properly using comments. Pay attention to performance in terms of execution time and memory usage when writing and selecting algorithms and data structures, and document their asymptotic complexity. Properly cover the functionality of this specification with unit tests. It is allowed to use the standard library, but not the non-standard libraries, for the selected programming language.

Text der Projektaufgabe

Schreiben Sie eine Bibliothek für die Arbeit mit regulären Sprachen, die durch endliche Automaten (DKA und ε-NKA) und/oder reguläre Ausdrücke repräsentiert werden. Realisieren Sie die folgenden Funktionalitäten für die Arbeit mit regulären Sprachen: Ausführung endlicher Automaten (DKA und ε-NKA), die von Benutzern der Bibliothek gebildet werden. Konstruktionen von Darstellungen von Vereinigung, Schnittmenge, Disjunktion von Sprachen, Komplementen von Sprachen und das Ergebnis der Anwendung von Kleens Stern auf eine Sprache, mit Unterstützung für Verkettungsoperationen. Bestimmen der Länge der kürzesten und längsten Wörter in der Sprache sowie Testen der Endgültigkeit der Sprache. Minimierung der Zustandszahl für deterministische endliche Automaten Transformation von ε-NKA in DKA sowie Transformation eines regulären Ausdrucks in einen endlichen Automaten. Vergleich der Repräsentationen regulärer Sprachen, einschließlich regulärer Ausdrücke, nach Sprachgleichheit. Schreiben Sie eine Anwendung, die die Spezifikation der Repräsentation einer regulären Sprache lädt (unterstützte Repräsentationen sind DKA, ε-NKA und reguläre Ausdrücke) und die überprüft, ob die angegebenen Zeichenfolgen zu der repräsentierten Sprache gehören. Führen Sie eine lexikalische Analyse der Spezifikation der Repräsentation der regulären Sprache durch und notieren Sie im Falle von lexikalischen Unregelmäßigkeiten die Anzahl der relevanten Zeilen der Spezifikation, die Unregelmäßigkeiten enthalten. Schreiben Sie eine Anwendung zum Generieren von Programmcode zum Simulieren eines Zustandsautomaten basierend auf der Spezifikation eines deterministischen Zustandsautomaten. Aktivieren Sie, dass Benutzer des generischen Codes für jeden gebildeten Zustand die Reaktion auf Zustandseintritts- und Austrittsereignisse angeben können. Ermöglichen Sie Benutzern des generierten Codes für jedes Symbol des Automatenalphabets, die Reaktion auf die Ausführung des Übergangs für dieses Symbol zu spezifizieren, was von dem Zustand abhängen kann, von dem aus der Übergang erfolgt. Es soll möglich sein, Reaktionen zu verknüpfen, so dass eine Kette von Reaktionen auf ein Ereignis effektiv gebildet werden kann. Halten Sie sich an OOP-Prinzipien, Clean Code Writing-Prinzipien, DRY-Prinzipien, Code-Wiederverwendungsprinzipien, SOLID-Prinzipien und Konventionen für die verwendete Programmiersprache. Dokumentieren Sie Code richtig mit Kommentaren. Achten Sie beim Schreiben und Auswählen von Algorithmen und Datenstrukturen auf die Performance in Bezug auf Ausführungszeit und Speicherverbrauch und dokumentieren Sie deren asymptotische Komplexität. Decken Sie die Funktionalität dieser Spezifikation ordnungsgemäß mit Komponententests ab. Für die gewählte Programmiersprache darf die Standardbibliothek verwendet werden, nicht jedoch die Nicht-Standardbibliotheken.