VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Podpora jazyka Monkey C v prostředí VS Code

Monkey C Language Support in VS Code

2021 Jakub Pšenčík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jakub Pšenčík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Podpora jazyka Monkey C v prostředí VS Code
Monkey C Language Support in VS Code

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit rozšíření pro Visual Studio Code, které do něj doplní podporu pro jazyk Monkey C. Základem rozšíření bude parser jazyka Monkey C, který student dostane dispozici. Rozšíření bude schopno nad výstupem tohoto parseru provádět sémantickou analýzu kódu. Výsledky této analýzy budou vizualizovány v rámci VS Code. Rozšíření bude dále umožňovat vygenerování nového projektu ze šablony, jeho konfiguraci a následné sestavení finální aplikace.

Hlavní body zadání:
1. Seznámení s problematikou vývoje rozšíření pro VS Code, sémantickou analýzou kódu a jazykem Monkey C.
2. Návrh a implementace rozšíření pro VS Code.
3. Testování výsledného řešení.

Seznam doporučené odborné literatury:

[1] CHERNY, Boris. Programming TypeScript: making your JavaScript applications scale. Sebastopol, CA: O'Reilly Media, 2019. ISBN 978-1492037651.
[2] PARR, Terence. The definitive ANTLR 4 reference. Dallas, Texas: The Pragmatic Bookshelf, [2012]. Pragmatic programmers. ISBN 978-1934356999.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

Datum zadání: 01.09.2020

Datum odevzdání: 30.04.2021

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. září 2020 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 1. září 2020 .....

Rád bych na tomto místě poděkoval svému vedoucímu práce Ing. Janu Janouškovi za odborné a metodické vedení, ochotu a pomoc v průběhu vypracovávání práce.

Abstrakt

V této bakalářské práci se budu zabývat vývojem rozšíření pro Visual Studio Code, jenž bude poskytovat plnou podporu jazyka Monkey C. Součástí práce bude bude představení prostředí Visual Studio Code a problematika vývoje rozšíření v tomto prostředí. Představen bude, mimo jiné, také jazyk Monkey C. V další části práce se budu zabývat nástrojem ANTLR, díky kterému jsme schopni generovat vlastní překladač jazyka pomocí bezkontextové gramatiky, parsováním kódu a jeho syntaktickou analýzou. Závěrem bude výsledné rozšíření testováno.

Klíčová slova: bakalářská práce, rozšíření, Monkey C, Typescript

Abstract

In this bachelor thesis I will deal with the development of extension for Visual Studio Code, which will provide full support for Monkey C language. The work will include an introduction to the Visual Studio Code environment and the development of extensions in this environment. Among other things, the Monkey C language will be introduced. In the next part of the work I will describe the ANTLR tool, used to generate our own language compiler using context-free grammar, code parsing and its syntactic analysis. Finally, the resulting extension will be tested.

Keywords: bachelor thesis, extension, Monkey C, Typescript

Obsah

Seznam použitých zkratek a symbolů

ANTLR	–	ANother Tool for Language Recognition

Seznam výpisů zdrojového kódu

1 Úvod

Tato bakalářská práce se věnuje tvorbě a vývoji rozšíření pro vývojové prostředí Visual Studio Code. Cílem práce je vytvořit takové rozšíření, která poskytne plnou podporu při vývoji aplikací v jazyce Monkey C. Jedná se o dynamicky postavený jazyk, stejně jako např. python, R, atd... Jazyk se používá k vývoji aplikací pro Garmin zařízení.
V další kapitolách budou detailně popsány klíčové komponenty k vytvoření finální aplikace, např. gramatika pro popis jazyka, ANTLR parser, atd...

2 Problematika vývoje rozšíření pro VS Code

2.1 Visual Studio Code

Visual Studio Code je editor zdrojového kódu vytvořený společností Microsoft pro Windows, Linux a MacOS. Nabízí spoustu užitečných funkcí, mezi které patří podpora ladění kodu, zvýrazňování syntaxe, automatické doplňování a našeptávání kódu, atd...

2.2 Typescript

TypeScript[1] je open-source programovací jazyk vyvinutý společností Microsoft. Jedná se o nádstavbu nad jazykem JavaScript, která jej rozšiřuje o statické typování a další atributy, které známe z objektově orientovaného programování jako jsou třídy, moduly a další. Samotný kód psaný v jazyce TypeScript se kompiluje do jazyka JavaScript. Jelikož je tento jazyk nádstavbou nad JavaScriptem, je každý JavaScript kód automaticky validním TypeScript kódem.

2.3 Komponenty potřebné pro vytvoření rozšíření

Předtím, než začneme pracovat na samotném rozšíření, je potřeba obstarat několik důležitých nastrojů a komponent, jenž jsou klíčové při vývoji.

2.3.1 Gramatika

Jako první je potřeba definovat popis jazyka MonkeyC. V našem případě je jazyk popsán prostřednictvím bezkontextové gramatiky, která formálně definuje syntax (pravidla) jazyka.

```
grammar MonkeyC;

options {
    superClass=MonkeyCBaseParser;
}
@header {import { MonkeyCBaseParser } from "../MonkeyCBaseParser";}


    DOT : '.';
    SEMI : ';';
    QUES : '?';
    COLON : ':';
    CLASS : 'class';
    FUNCTION : 'function';
    RETURN : 'return';
    NEW : 'new';
    VAR : 'var';
    CONST : 'const';
    MODULE : 'module';
    USING : 'using';
    AS : 'as';
    ENUM : 'enum';
    EXTENDS : 'extends';
```

Obr. 1 – ukázka hlavičky gramatiky MonkeyC.g4 pro popis jazyka

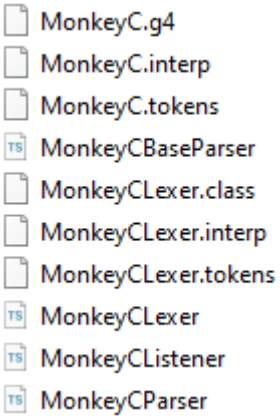
Na obrázku mužeme vidět prvních pár řádků gramatiky pro MonkeyC. Gramatika obsahuje spoustu známých klíčových slov, nebo-li tokenů, např. "CLASS", "FUNCTION", "USING", atd...

2.3.2 Java

Jelikož je ANTLR psán v Javě, je potřeba ji nainstalovat na naše zařízení, přičemž se požaduje verze 1.6 nebo vyšší. Poté již stačí stáhnout akualní ANTLR jar, což je momentálně "antlr-4.8-complete.jar".

2.3.3 Parser a Lexer

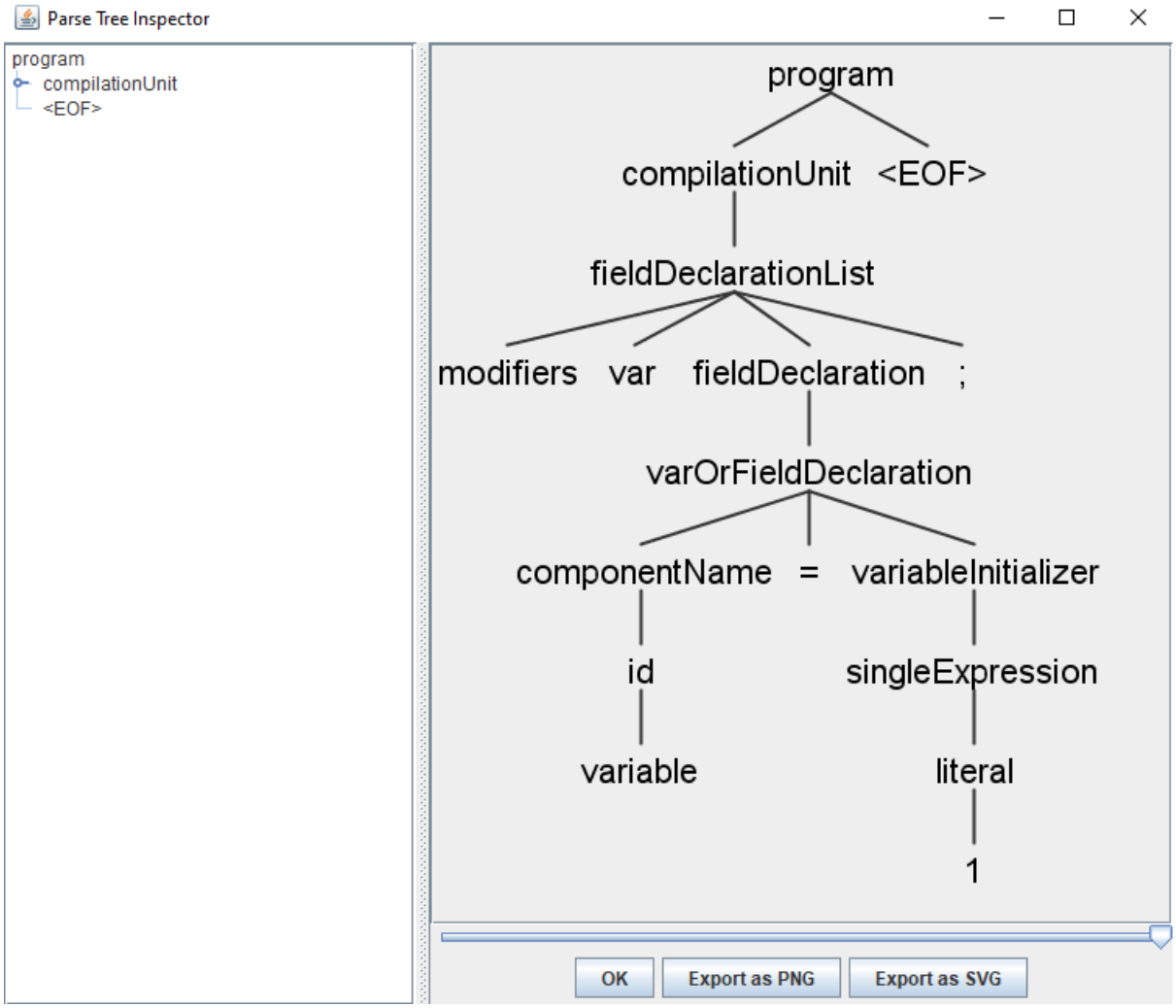
lexer - lexer (nebo také tokinizér) "rozdělí"text na vstupu (v našem případě MonkeyC kód) na tokeny.
parser - shora dolů prochází text na vstupu a porovnává jednotlivé řádky s pravidly obsažené v gramatice.
K vytvoření parseru a lexeru je potřeba spustit ANLTR nástroj, který s pomocí gramatiky tyto souboru vygeneruje.



Obr. 2 – potřebné soubory vygenerované nástrojem ANTLR

2.3.4 TestRig

ANTLR poskytuje flexibilní testovací nástroj umístněný v runtime knihovně s názvem TestRig. TestRig dokáže poskytnout spoustu informací o tom, jak "recognizéry"(parser a lexer) zpracovávají InputStream ze vstupního souboru.
Nejjednodušší způsob ověření, zda gramatika rozpoznává vstup správně, je zobrazit si vygenerovaný syntaktický strom vizuálně.

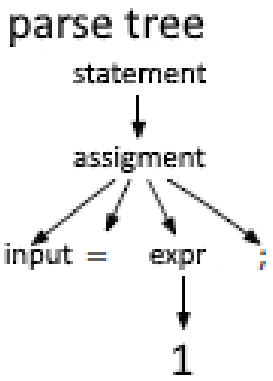


Obr. 3 – "ParseTreeInspector"- vizuální podoba syntaktického stromu

Na obrázku můžeme detailně vidět jak parser vytvořil syntaktický strom a jak jsou jednotlivé části propojené. Strom začíná kořenem, jenž je v naší gramatice pojmenová, jako "program". Takto je kořen pojmenován při každém parsování.

3 Syntaktická a sémantická analýza kódu

Abychom mohli implementovat náš jazyk (MonkeyC), je potřeba vytvořit aplikaci, která je schopna číst "věty"na vstupu a odpovídajicím způsobem rozpoznává klíčová slova či fráze naší gramatiky.(Obecně je jazyk složení platných vět, kdy věta se skládá z frází a fráze se skládá ze symbolů slovní zásoby).
Obecně lze říci, pokud nějaká aplikace vykonává (interpretuje) zápis jiného programu v jeho zdrojovém kódu ve zvoleném programovacím jazyce (v našem případě Typescript), že se jedná o tzv. interpret[2]. Jako příklady uvedu kalkulačku, aplikace pro čtení konfiguračních souborů, Python interprety, atd... Pokud, na druhou stranu, převádíme "věty"z jednoho jazyka do druhého (např. z Javy do 'C Sharp'), nazýváme takovou aplikaci překladačem.
Úkolem překladače či interpreta je tedy rozpoznat platné vstupy, tedy věty, fráze, subfráze, klíčová slova, atd... Přičemž rozpoznáním platného vstupu je myšleno identifikovat jednotlivé fráze a rozlišit je od jiných.
Jako příklad použijeme rozpoznání vstupu *input* = 1; s validní MonkeyC syntaxí. Víme tedy, že "input"je cíl přiřazení a "1"představuje hodnotum která se má přiřadit. Stejně jako jsme my schopni rozlišit v našem jazyce sloveso od podstatného jména, je naše aplikace schopna rozlišit toto přiřazení od např. importu knihovny pomocí klíčového slova "using".
Programy, které jsou schopny rozpoznávat konkrétní jazyk, se nazývají parsery nebo syntaktické analyzátory, přičemž syntax odkazuje na pravidla obsažená v popisu jazyka, čili gramatice.



Obr. 3 – přiřazení hodnoty 1 proměnné input

3.1 ANTLR - Nástroj pro generování překladače

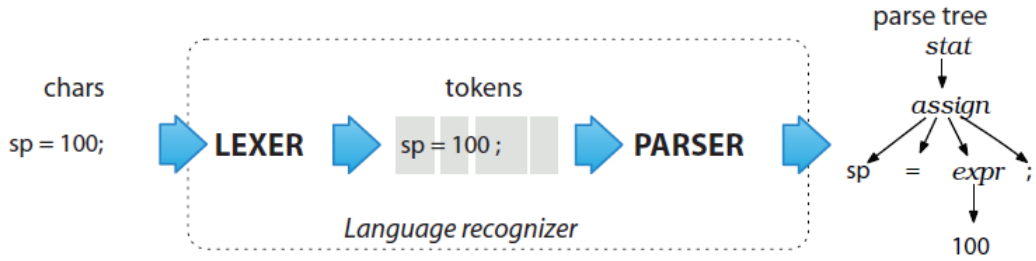
ANTLR nebo také ANother Tool for Language Recognition (jiný nástroj pro rozpoznávání jazyka)[2] je výkonný nástroj pro generování syntaktických analyzátorů, tzv. "parser". Tento parser je poté schopen číst, zpracovávat, spouštět nebo překládat strukturované textové či binární soubory. Používá se především k vytváření nových jazyků, nástrojů nebo "frameworků". Využívá bezkontextový jazyk typu LL.

3.1.1 LL syntaktický analyzátor

LL[3] je syntaktický analyzátor shora-dolů pro bezkontextové gramatiky. Analyzuje vstup zleva (Left) doprava a konstruuje nejlevější derivaci (Leftmost) věty. Gramatiky, které jsou takto analyzovatelné, se nazývají LL gramatiky.

3.2 Syntaktický strom

Syntaktický strom představuje datovou strukturu, kterou ANTLR vytvoří při parsování vstupního souboru. Struktura je složená z kořene (root) a uzlů, které mohou představovat buď další podstromy, které odpovídají pravidlům gramatiky, nebo listy stromu.
V naší aplikaci Syntaktický strom představuje třída AST (viz. kód níže), která uchovává všechna důležitá data pro korektní analýzu vstupního souboru, např. počet uzlů stromu, soubor vztahující se ke konkrétnímu stromu, kořen stromu, atd...



Obr. 4

```
export class AST {

    static nodeCount: number = 0;
    private parseTree : Node[];
    public documentName: string;
    public currentNode : Node;
    public root : Node;

    constructor(documentName: string) {

        this.parseTree = [];
        this.documentName = documentName;
        this.currentNode = new Node(undefined,undefined,undefined,undefined,
            undefined);
        this.root= new Node(undefined,undefined,undefined,undefined,undefined);
```

```
    }

    getParseTree() {
        return this.parseTree;
    }

    addNode(node : Node) : void {
        this.parseTree.push(node);
    }

    findNode(ruleNumber: number) : Node | undefined {

        for(let i = this.parseTree.length-1; i >= 0; i--) {
            if(this.parseTree[i] != null && this.parseTree[i].getContext()?.
                ruleIndex === ruleNumber) {
                return this.parseTree[i];
            }
        }
        return undefined;
    }
}
```

Výpis 1: třída AST v jazyce Typescript

Další, neméně důležitou, třídou v naší apikaci je třída Node, která představuje uzel stromu. Jedná se, ve své podstatě, o strukturu uchovávající následující atributy:
- kontext daného uzlu
- předka uzlu
- potomka uzlu
- hodnotu přestavující text v daném uzlu
- typ uzlu, který se odvíjí od pravidel v gramatice

```
export class Node {

    private id : number;
    private context : ParserRuleContext | undefined ;
    private parent : Node | undefined;
    private child : Node[] | undefined;
```

```
    private value: string | undefined;
    private _type : number | undefined;

    constructor(context: ParserRuleContext | undefined, parent : Node |
        undefined, child : Node[] | undefined, value : string | undefined,
                _type: number | undefined)
    {
        this.id = AST.nodeCount;
        this.context = context;
        this.parent = parent;
        this.child = child;
        this.value = value;
        this._type = _type;

        AST.nodeCount++;
    }

    getId() { return this.id; }
    getContext() { return this.context; }
    getParent() { return this.parent; }
    getChildren() { return this.child; }
    getValue() {return this.value; }
    getType() {return this._type; }
    addChild(child : Node) : void{ this.child?.push(child); }

}
```

Výpis 2: třída Node v jazyce Typescript

3.3 ANTLR Listener a callback funkce

TODO: popis listeneru a callbacků

3.4 Sémantický strom

Strom, jenž hraje zásadní roli v procesech, jako je našeptávání kódu, detekci chyb, kdy je použita třída z modulu, který není referencován, atd...

3.5 parser

Fusce nibh. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur? Etiam ligula pede, sagittis quis, interdum ultricies, scelerisque eu. Maecenas sollicitudin. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Integer vulputate sem a nibh rutrum consequat. Pellentesque sapien. Pellentesque arcu. Suspendisse nisl. Fusce consectetuer risus a nunc. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam quis nulla. Nulla non lectus sed nisl molestie malesuada. Duis viverra diam non justo. Sed ac dolor sit amet purus malesuada congue. Aenean id metus id velit ullamcorper pulvinar. Aliquam ornare wisi eu metus. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.

3.6 nedostatky rozšíření

TODO: error kdy to neumožňuje (tmp.format()).toString()

4 Představení jazyka Monkey C

Monkey C[4] je objektově orientovaný jazyk navržený pro snadný vývoj Connect IQ aplikací. Jedná se o dynamický programovací jazyk, podobně jako Java, PHP či Ruby. Cílem Monkey C je zjednodušit proces vytváření samotné aplikace a umožnit tak vývojářům více se soustředit na zákazníka a méně na omezení zdrojů. Využívá tzv. "reference counting"k automatickému čištění paměti, což vývojáře osvobozuje od manuální správy paměti (např. jako v jazyce C/C++).

```
using Toybox.Application as App;
using Toybox.System;

class MyProjectApp extends App.AppBase {

// onStart() is called on application start up
function onStart(state) {
}

// onStop() is called when your application is exiting
function onStop(state) {
}

// Return the initial view of your application here
function getInitialView() {
return [ new MyProjectView() ];
}
}
```

Obr. 3 – ukázka jednoduchého fragmentu kódu v MonkeyC

5 Návrh a implementace rozšíření pro VS Code

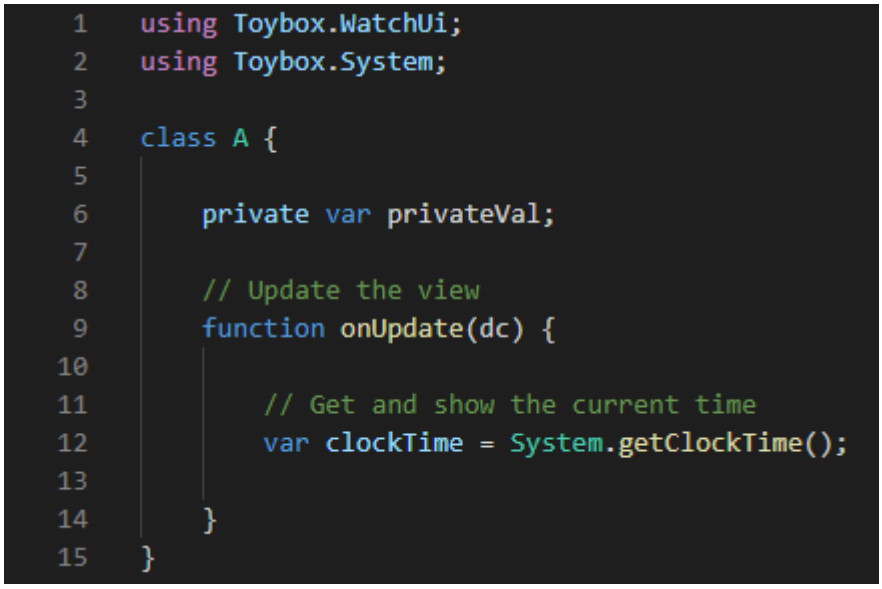
Rozšíření bude, jak již bylo nastíněno výše, implementováno v jazyce Typescript. K vytvoření rozšíření samotného je potřeba Node.js a Git. Poté je vyžadována instalace programu "Yeoman""VS Code Extension Generator".

5.1 zvýraznění kódu

Zvýraznění, nebo-li obarvení klíčových slov kódu je realizováno pomocí JSON souboru, jenž pomocí regulárních výrazů v textu vyhledává příslušná slova a ty poté obarvuje.

```
1   using Toybox.WatchUi;
2   using Toybox.System;
3
4   class A {
5
6       private var privateVal;
7
8       // Update the view
9       function onUpdate(dc) {
10
11          // Get and show the current time
12          var clockTime = System.getClockTime();
13
14      }
15  }
16
```

Obr. 4 – ukázka neobarveného kódu v MonkeyC



Obr. 5 – ukázka obarvení kódu v MonkeyC

Hned na první pohled je zřejmé, že obarvení poskytuje uživateli větší přehled a orientaci v kódu, jak je vidět na obrázku [5]

5.2 Automatické doplňování a našeptávání kódu

Hlavním aktérem při našeptávání vhodných částí kódu je "antlr4-c3 The ANTLR4 Code Completion Core".

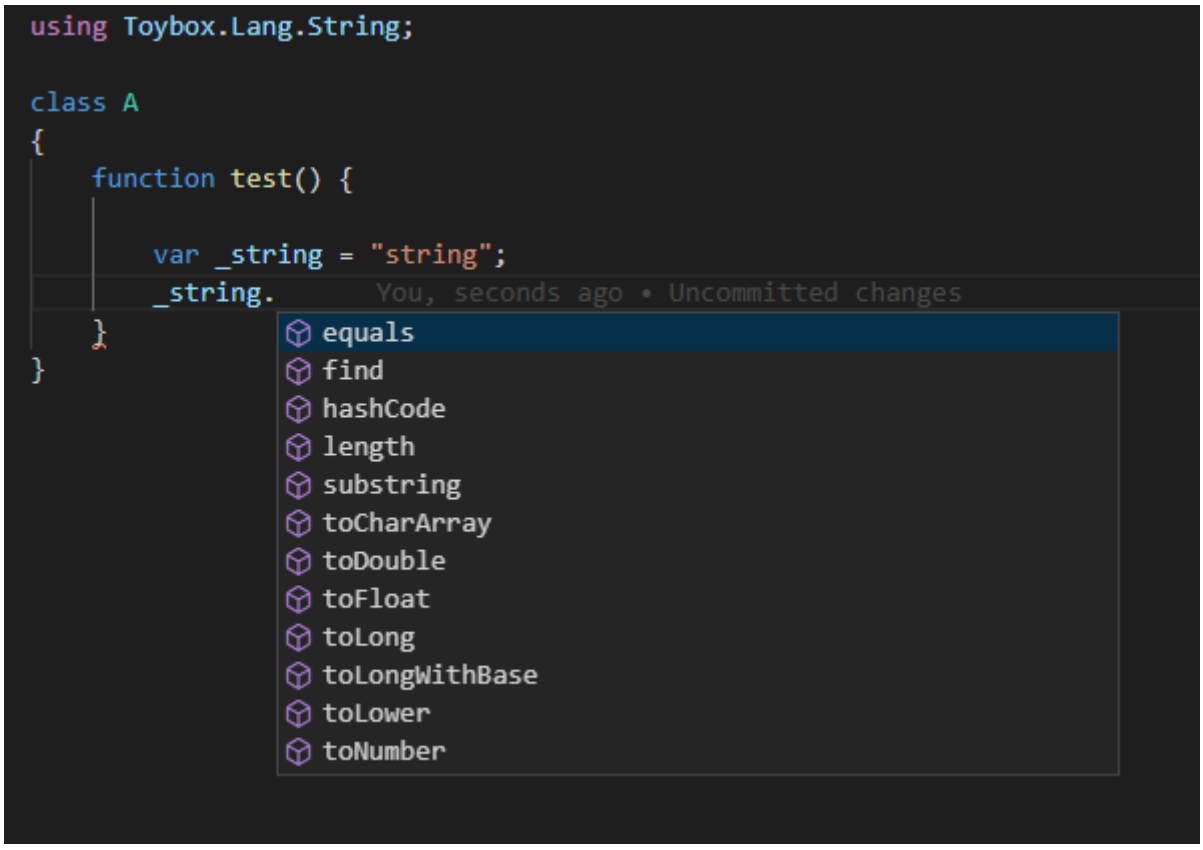
Jedná se o "stroj"sloužící k dokončování gramatického agnostického kódu pro analyzátory založené na ANTLR4. Engine c3 je schopen poskytnout kandidáty na doplnění kódu, kteří jsou užiteční pro editory s analyzátory generovanými ANTLR, nezávisle na skutečném jazyku / gramatice použité pro generování.

Původní implementace je poskytována jako "node module"a je psána v jazyce Typescript.

Pro zobrazení možných symbolů ve zdrojovém kódu zjevně potřebujete zdroj pro všechny dostupné symboly na dané pozici. Jejich poskytnutí je obvykle úkolem tabulky symbolů. Jeho obsah lze odvodit z vašeho aktuálního zdrojového kódu (pomocí analyzátoru + analyzátoru Listeneru). Statičtější části (například runtime funkce) lze načíst z disku nebo poskytnout pevně zakódovaný seznam atd. Tabulka symbolů pak může odpovědět na vaši otázku ohledně všech symbolů daného typu, které jsou viditelné z dané pozice. Pozice obvykle odpovídá konkrétnímu symbolu v tabulce symbolů a struktura pak umožňuje snadno získat viditelné symboly. Engine c3 je dodáván s malou implementací tabulky symbolů, která však není pro použití knihovny povinná, ale poskytuje snadný start, pokud již nemáte vlastní třídu tabulek symbolů.

Zatímco tabulka symbolů poskytuje symboly daného typu, musíme zjistit, který typ je ve skutečnosti vyžadován. To je úkol enginu c3. V nejjednodušším nastavení vrátí pouze klíčová slova (a další symboly lexerů), která jsou povolena gramatikou pro danou pozici (což je samozřejmě stejná pozice, která se používá k nalezení kontextu pro vyhledávání symbolů ve vaší

tabulce symbolů). Klíčová slova jsou pevná sada slov (nebo sekvencí slov), která se obvykle nenachází v tabulce symbolů. Skutečné textové řetězce můžete získat přímo ze slovníku analyzátoru. C3 za ně vrací pouze lexerové tokeny.



Obr. 6 – ukázka automatického našeptávání kódu na proměnné typy string

5.3 komentáře

TODO: popsat vytahování návratových typů a parametrů z komentářů nad funkcemi v Toyboxu

6 Testování výsledného řešení

Testování probíhá napříč mnoha fragmenty kódu.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aliquam erat volutpat. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. Vivamus porttitor turpis ac leo. Sed ac dolor sit amet purus malesuada congue. Maecenas aliquet accumsan leo. Etiam posuere lacus quis dolor. Curabitur sagittis hendrerit ante. Duis condimentum augue id magna semper rutrum. Mauris dolor felis, sagittis at, luctus sed, aliquam non, tellus. Integer vulputate sem a nibh rutrum consequat. Duis pulvinar. Duis viverra diam non justo. Aenean vel massa quis mauris vehicula lacinia.

Fusce nibh. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur? Etiam ligula pede, sagittis quis, interdum ultricies, scelerisque eu. Maecenas sollicitudin. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Integer vulputate sem a nibh rutrum consequat. Pellentesque sapien. Pellentesque arcu. Suspendisse nisl. Fusce consectetuer risus a nunc. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam quis nulla. Nulla non lectus sed nisl molestie malesuada. Duis viverra diam non justo. Sed ac dolor sit amet purus malesuada congue. Aenean id metus id velit ullamcorper pulvinar. Aliquam ornare wisi eu metus. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.

7 Závěr

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aliquam erat volutpat. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. Vivamus porttitor turpis ac leo. Sed ac dolor sit amet purus malesuada congue. Maecenas aliquet accumsan leo. Etiam posuere lacus quis dolor. Curabitur sagittis hendrerit ante. Duis condimentum augue id magna semper rutrum. Mauris dolor felis, sagittis at, luctus sed, aliquam non, tellus. Integer vulputate sem a nibh rutrum consequat. Duis pulvinar. Duis viverra diam non justo. Aenean vel massa quis mauris vehicula lacinia.

Fusce nibh. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur? Etiam ligula pede, sagittis quis, interdum ultricies, scelerisque eu. Maecenas sollicitudin. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Integer vulputate sem a nibh rutrum consequat. Pellentesque sapien. Pellentesque arcu. Suspendisse nisl. Fusce consectetuer risus a nunc. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam quis nulla. Nulla non lectus sed nisl molestie malesuada. Duis viverra diam non justo. Sed ac dolor sit amet purus malesuada congue. Aenean id metus id velit ullamcorper pulvinar. Aliquam ornare wisi eu metus. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.

8 Literatura

1. TypeScript [online] [cit. 2021-02-18]. Dostupné z: *https* : *//cs.wikipedia.org/wiki/TypeScript*.

2.Interpret (software) [online] [cit. 2021-03-08]. Dostupné z: *https* : *//cs.wikipedia.org/wiki/Interpret*$_{(softw}$

3. ANTLR [online] [cit. 2021-03-07]. Dostupné z: *https* : *//www.antlr.org/*.

4. LL [online] [cit. 2021-03-07]. Dostupné z: *https* : *//cs.wikipedia.org/wiki/LL*$_s$*yntaktický*$_a$*nalyzátor/*.

5. MonkeyC [online]. Dostupné z: *https* : *//developer.garmin.com/connect* − *iq/monkey* − *c/*