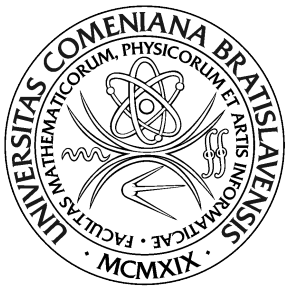
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

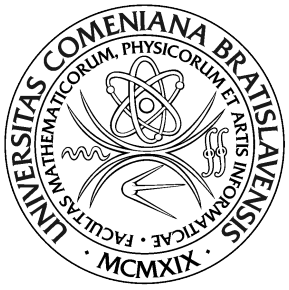


EASYLOGO PRE WEB

Diplomová práca

2021 Bc. Andrea Hajná

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



EASYLOGO PRE WEB

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra didaktiky matematiky, fyziky a informatiky
Školiteľ:	doc. RNDr. Ľubomír Salanci, PhD.

Bratislava, 2021 Bc. Andrea Hajná

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Andrea Hajná
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: EasyLogo pre web
Web EasyLogo

Anotácia: Cieľom práce je vytvoriť webovú verziu aplikácie EasyLogo, ktorá slúži na učenie sa základov programovania. Aplikácia bude naprogramovaná iba pomocou HTML a jazyka Javascprit tak, aby sa dala používať vo web prehliadači. Iné technológie nie sú žiadúce, aby bola aplikácia ľahko prenositeľná a fungovala aj bez pripojenia k internetu. Výsledná aplikácia bude určená pre žiakov aj učiteľov. Žiaci budú zostavovať programy ťahaním a budú môcť ukladať a načítavať hotové programy z textových súborov. Učitelia budú môcť pridávať svoje aktivity pomocou textových súborov alebo editora aktivít. Pre fungovanie aplikácie bude potrebné navrhnúť používateľské prostredie a vytvoriť kompilátor aj virtuálnu mašinu. Súčasťou práce bude výskum, nakoľko je výsledná aplikácia použiteľná v triede so žiakmi.

Vedúci: doc. RNDr. Ľubomír Salanci, PhD.
Katedra: FMFI.KDMFI - Katedra didaktiky matematiky, fyziky a informatiky
Vedúci katedry: prof. RNDr. Ivan Kalaš, PhD.

Dátum zadania: 01.12.2020

Dátum schválenia: 08.12.2020 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som vypracovala samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa.

. . . . . .

Bratislava, 2021 Bc. Andrea Hajná

Poďakovanie

...

Abstrakt

Cieľom práce je vytvoriť webovú verziu aplikácie EasyLogo, ktorá slúži na učenie sa základov programovania. Aplikácia bude naprogramovaná iba pomocou HTML a jazyka Javascprit tak, aby sa dala používať vo web prehliadači. Iné technológie nie sú žiadúce, aby bola aplikácia ľahko prenositeľná a fungovala aj bez pripojenia k internetu. Výsledná aplikácia bude určená pre žiakov aj učiteľov. Žiaci budú zostavovať programy ťahaním a budú môcť ukladať a načítavať hotové programy z textových súborov. Učitelia budú môcť pridávať svoje aktivity pomocou textových súborov alebo editora aktivít. Pre fungovanie aplikácie bude potrebné navrhnúť používateľské prostredie a vytvoriť kompilátor aj virtuálnu mašinu. Súčasťou práce bude výskum, nakoľko je výsledná aplikácia použiteľná v triede so žiakmi.

Kľúčové slová: EasyLogo, kompilátor, webová aplikácia, edukačný softvér

Abstract

The aim of the master thesis is create web version of EasyLogo application which is used for learning basics of programming. Application will be written using HTML and programming language Javascript to be used in web browser. Any other technologies are not eligible to be easy portable and to work without internet connection. The resultant web application will be destined for pupils and teachers. Pupils will be able to create programs by dragging and will be able to export and import complete programs from text files. Teachers will be able to add activities by using text files or activity editor. For functioning of application is needed to design user interface and create compiler also virtual machine. A part of the thesis will be research because the resultant app is applicable in class with pupils.

Keywords: EasyLogo, compiler, web application, educational software

Obsah

Kapitola 1

Úvod

V dnešnej dobe sú informačné technológie súčasťou nášho každodenného života. Niet teda divu, že sa práca s počítačom a technológiami stala školským predmetom, ktorý absolvujú už deti na prvom stupni základných škôl. Jedným z cieľov informatického predmetu je okrem práce s počítačom aj naučiť deti algoritmicky myslieť. Pri výučbe programovania sa preto používajú rôzne prostredia, ktoré sa snažia deťom výučbu spríjemniť a upriamiť ich pozornosť nie na technickú stránku programovania, ale naopak na samotné vytváranie algoritmov.

Cieľom práce je vytvoriť webovú verziu aplikácie EasyLogo, ktorá slúži na učenie sa základov programovania. Aplikácia bude naprogramovaná iba pomocou HTML a jazyka Javascript tak, aby sa dala používať vo web prehliadači. Iné technológie nie sú žiadúce, aby bola aplikácia ľahko prenositeľná a fungovala aj bez pripojenia k internetu. Výsledná aplikácia bude určená pre žiakov aj učiteľov. Žiaci budú zostavovať programy ťahaním a budú môcť ukladať a načítavať hotové programy z textových súborov. Učitelia budú môcť pridávať svoje aktivity pomocou textových súborov alebo editora aktivít. Pre fungovanie aplikácie bude potrebné navrhnúť používa-

teľské prostredie a vytvoriť kompilátor aj virtuálnu mašinu. Súčasťou práce bude výskum, nakoľko je výsledná aplikácia použiteľná v triede so žiakmi.

Diplomová práca v prvej časti analyzuje a popisuje vývojové prostredie EasyLoga, jeho programové konštrukcie a korytnačiu grafiku. V kapitole Analýza sa takisto zaoberám analýzou existujúcich riešení a prehľadom problematiky. Ďalej v diplomovej práci popisujem návrh samotného riešenia, teda návrh kompilátora, používateľského prostredia a technológie, ktoré som pri vývoji použila. V kapitole Implementácia popisujem konkrétnu implementáciu webového prostredia EasyLoga a a riešenie problémov, ktoré pri implementácii vznikli. V poslednej kapitole je popísaný priebeh a výsledky testovania.

Kapitola 2

Analýza

V tejto kapitole si podrobnejšie rozoberieme problematiku spracovania programu pomocou kompilátora. Takisto si priblížime črty a programové konštrukcie prostredia EasyLogo, zanalyzujeme požiadavky aplikácie a pozrieme sa na podobné existujúce riešenia.

2.1 EasyLogo

Nasledujúca podkapitola bola spracovaná podľa [10].

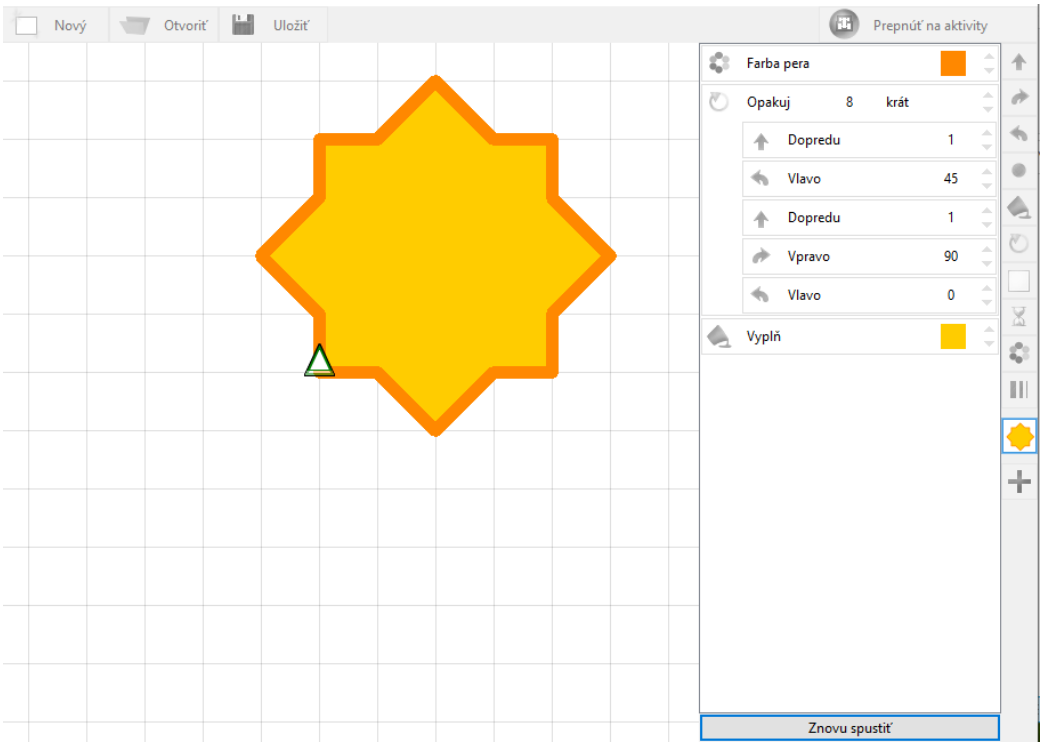
EasyLogo je vývojové prostredie vytvorené pre ľudí so základnými počítačovými znalosťami pre čo najjednoduchšiu výučbu základov programovania a hľadania riešenia logických problémov.

EasyLogo vychádza zo zjednodušenej verzie jazyka Logo, pričom vo svojich programových konštrukciách ponúka cykly, procedúry a rôzne kresliace príkazy.

2.1.1 Korytnačia grafika

Korytnačia grafika v prostredí EasyLogo pracuje s mriežkou, ktorá zjednodušuje kreslenie. Pri vytváraní prostredia sa kládol dôraz najmä na jednoduchosť používania a atraktívnosť pre používateľa. Prostredie má pomôcť začiatočníkom s pochopením základných programátorských konceptov, čo by zložité príkazy či matematické výpočty pri kreslení výrazne sťažovali. Počítanie krokov je tak omnoho jednoduchšie, ak sa tvary vykresľujú do mriežky.

Ďalšou obmenou oproti bežnej korytnačej grafike je, že pri otáčaní korytnačka rotuje o násobky 45°. Takéto rotácie sú jednoduché na výpočty pri kreslení a zároveň dostačujúce pre kreslenie rôznych tvarov.



Obr. 2.1: Ukážka prostredia EasyLogo - vykreslenie hviezdy

2.1.2 Program

Program je konštruovaný pomocou jednoduchých kartičiek umiestnených v pracovnom paneli prostredia, ktoré používateľ myšou usporadúva do bloku editora. Tlačidlá príkazov sú vytvorené z ikon, ktoré jednoznačne popisujú, čo príkaz vykonáva. Po nadídení myšou na tlačidlo príkazu sa zobrazí jeho krátky popis. Používateľ teda nezadáva príkazy, čím sa predíde syntaktickým chybám a nesprávnemu zápisu. Podobnú koncepciu využívajú aj jazyky ako Scratch a Baltie.

Program je vykonávaný automaticky, teda už počas toho ako používateľ robí v programe zmeny. Tie sa ihneď zobrazia aj pri kreslení korytnačky v mriežke.

2.1.3 Programové konštrukcie

EasyLogo má nasledovné programové konštrukcie:

- forward - posúva korytnačku po mriežke o konštantný počet krokov
- right - otáča korytnačku o 45 stupňov doprava
- left - otáča korytnačku o 45 stupňov doľava
- repeat - opakuje príkazy v bloku konštantný počet krát
- dot - kreslí bodku
- fill - vypĺňa oblasti na mriežke konkrétnou farbou
- pen color - mení farbu pera
- pen width - mení šírku pera

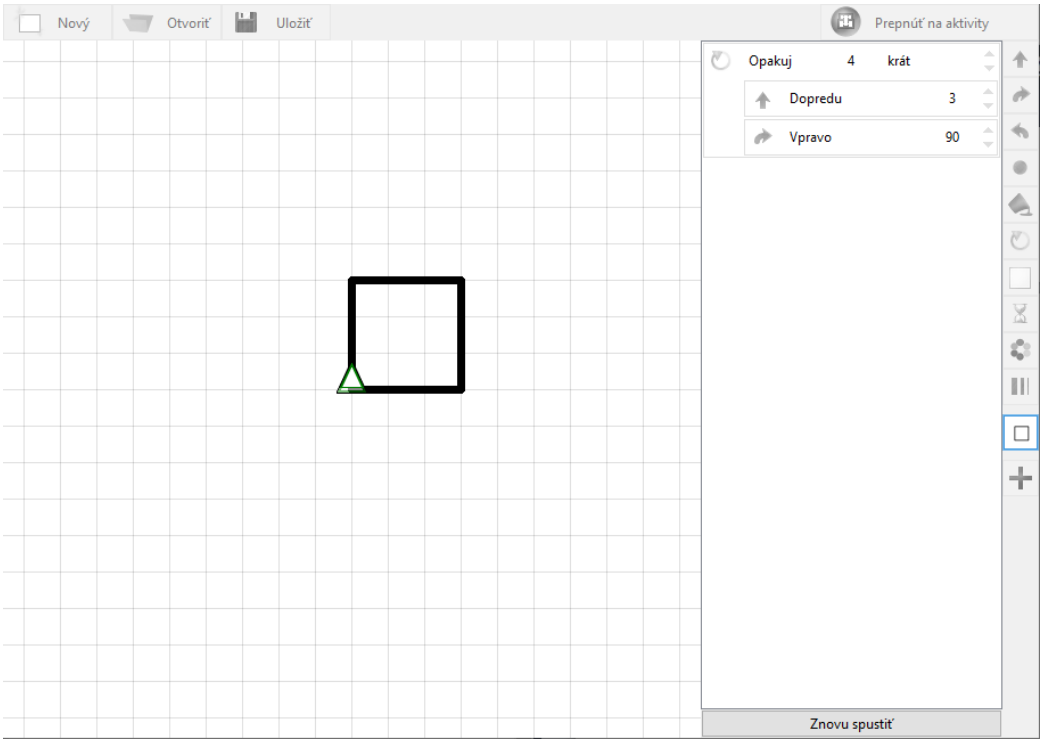
- move, draw - ummožňuje kresliť a nekresliť do mriežky pri pohybe korytnačky
- vytváranie procedúr
- vyvolávanie vytvorených procedúr

2.1.4 Režim

Prostredie umožňuje používateľovi prepínať sa medzi režimom voľnej tvorby a režimom aktivít.

Voľná tvorba

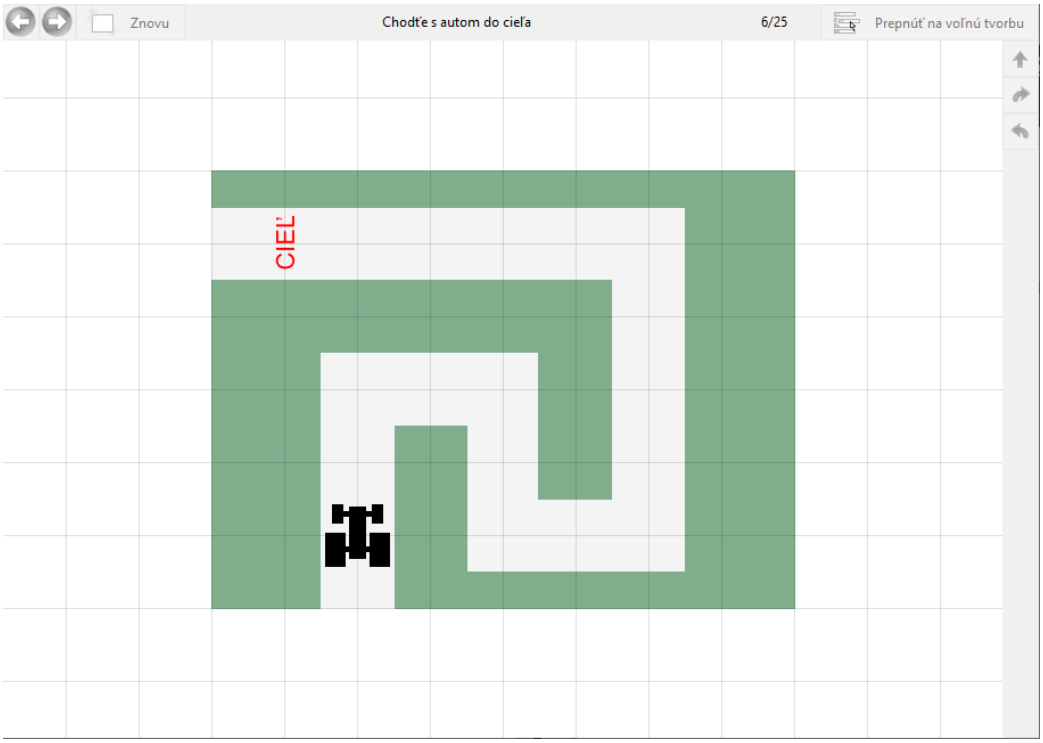
V tomto režime používateľ vytvára vlastné programy a procedúry. Pri zmene programu v editore sa zmeny prejavia aj vo vykresľovaní korytnačky, vďaka čomu používateľ ihneď vidí výsledok a môže program upravovať ďalej. Používateľ si vie svoje programy uložiť do súboru, prípadne neskôr načítať program zo súboru a ďalej ho upravovať.



Obr. 2.2: Ukážka prostredia EasyLogo - režim voľnej tvorby

Aktivity

V režime aktivít je vytvorených 25 úloh, z ktorých si môže používateľ vybrať. Pri každej úlohe sa zobrazuje používateľovi popis, čo je cieľom danej úlohy. Úlohy sú usporiadané podľa obtiažnosti od najľahších po najťažšie a postupne používateľa vedú od jednoduchých príkazov až po zložitejšie programátorské koncepcie. Najskôr teda používateľ iba posúva korytnačku po mriežke jednoduchými príkazmi, neskôr vytvára cykly a na konci dokáže pracovať s procedúrami.



Obr. 2.3: Ukážka prostredia EasyLogo - režim aktivít

V tomto režime môže byť prostredie okrem iného aj v troch módoch a to v móde *direct* , v ktorom používateľ má prístupné len príkazy ovládajúce pohyb a smer [obr 2.3]. V tomto móde používateľ nevytvára program ťahaním príkazov do blokov, ale klikaním na príslušné ikony riadi pohyb postavičky v grafickej ploche.

Ďalej v móde *simple* už používateľ vytvára program ťahaním a skladaním príkazových blokov, avšak nie je prístupné vytváranie procedúr.

V móde *procedural* je používateľovi prístupné aj tlačidlo na pridanie novej procedúry.

2.2 Špecifikácia

Pri vývoji softvéru je dôležité určiť ciele a požiadavky, ktoré má výsledný softvér spĺňať. V nasledujúcej podkapitole bližšie popíšem ciele, používateľov, funkčné a nefunkčné požiadavky softvéru, ktorý mám vytvoriť.

2.2.1 Ciele aplikácie

Aplikácia je určená pre programovanie v jazyku EasyLogo, pričom ide o vizuálny programovací jazyk, v ktorom používateľ programuje, tak že jednotlivé príkazy vo forme kartičiek skladá do blokov.

2.2.2 Používatelia

Aplikácia je určená na výučbu základov programovania, teda je určená najmä pre žiakov a učiteľov základných škôl. Avšak je vhodná pre akéhokoľvek používateľa so základnými počítačovými znalosťami.

Učiteľ

Učitelia môžu v aplikácii vytvárať rôzne aktivity a úlohy, ktoré potom v režime aktivít žiaci vykonávajú. Môže ísť o úlohy rôzneho typu, od vykreslenia objektu do kresliacej plochy až po opravu existujúceho programu.

Žiak

Žiakom je aplikácia prístupná v dvoch režimoch. V režime voľnej tvorby žiaci môžu vytvárať svoje vlastné programy a podprogramy. V režime aktivít žiaci prechádzajú súborom úloh a aktivít, ktoré plnia.

2.2.3 Funkčné a nefunkčné požiadavky

V tejto časti bližšie popíšem požiadavky na funkcionality a vlastnosti webovej aplikácie.

Funkčné požiadavky

1. **Vytvorenie programu**

   V aplikácií používateľ vytvára program, tak že príkazy myšou poťahuje a skladá do blokov v hlavnom programovom bloku. Rovnako potiahnutím von z programového bloku môže používateľ príkaz vyhodiť, prípadne ho premiestniť v programovom bloku.

2. **Spustenie programu**

   Program sa spúšťa automaticky pri zmene v hlavnom programovom bloku. Teda pri pridaní, vyhodení príkazu či pri jeho presunutí v programovom bloku. Každá zmena programu sa teda okamžite prejaví pri vykresľovaní v grafickej ploche. Program môže spustiť používateľ kliknutím na tlačidlo "Znova spustiť". Táto funkcionalita je potrebná napríklad pri vytváraní animácií, keď chce používateľ vidieť priebeh vykonávania programu znova.

3. **Uloženie programu**

   Aplikácia umožňuje uloženie aktuálne vytvoreného programu do súboru. Program je uložený v jazyku EasyLogo.

4. **Načítanie programu**

   V aplikácií môže používateľ zvoliť súbor, z ktorého chce program respektíve aktivitu načítať. Po vybratí sa súbor prečíta a príslušný prog-

ram sa vytvorí aj v hlavnom programovom bloku z jednotlivých kartičiek. Vstupný súbor musí obsahovať program v jazyku EasyLogo.

5. **Načítanie aktivít**

   Aplikácia pri prepnutí do režimu aktivít automaticky načíta zo súboru všetky aktivity. Medzi jednotlivými aktivitami sa vie používateľ posúvať.

Požiadavky na vlastnosti

Aplikácia musí fungovať bez pripojenia k internetu a musí byť ľahko prenositeľná a spustiteľná. Takisto aplikácia musí byť jednoduchá na používanie a intuitívna.

2.3 Prehľad problematiky

V nasledujúcej podkapitole rozoberieme porovnanie prostredí na výučbu programovania na slovenských školách a pozrieme sa na aspekty pri tvorbe edukačného softvéru, na ktoré treba dbať.

2.3.1 Edukačný softvér

Nasledujúca podkapitola je spracovaná podľa [Leh07].

Edukačný softvér je softvér určený na vzdelávanie a učenie sa, pri tvorbe ktorého je potrebné dbať na rôzne aspekty.

Z technických vlastností edukačného softvéru je dôležitá kompatibilita s operačným systémom, nízke požiadavky na hardvér počítača a jednoduchosť inštalácie softvéru.

Z pohľadu používateľa je dôležité, či je softvér intuitívny a jednoduchý na používanie, ale takisto či má pre používateľa atraktívny vzhľad a je odolný voči zlým vstupom. V niektorých prípadoch je dôležité, aj aby bol softvér lokalizovaný do materinského jazyka žiaka.

Z edukačného pohľadu sa na softvér pozeráme ako na nástroj pre výučbu a učenie sa. Pozeráme sa napríklad na to, či náročnosť softvéru je primeraná pre cieľových používateľov aplikácie, ale aj či je po pedagogickej stránke softvér korektný, teda používa správnu terminológiu, má nestranný obsah.

Ďalším podstatným aspektom je aj či poskytuje softvér možnosti na riadenie výučby a hodnotenia. To znamená, či softvér žiakovi poskytuje možnosť uložiť prácu a neskôr v nej pokračovať, či učiteľovi umožňuje import a export žiackych záznamov alebo vyhodnocovanie štatistík a výsledkov.

2.3.2 Prostredie na výučbu programovania

V súčasnosti existuje na trhu množstvo prostredí na výučbu základov programovania, či už licencovaných alebo voľne dostupných na internete.

Tieto prostredia sa líšia spôsobom, akým je vytváraný program, programátorskými konštrukciami a konceptami, ktoré poskytujú, ale aj spôsobom ako zabraňujú vzniku syntaktických chýb, respektíve ako na ne používateľa upozorňujú.

Gujberová a Tomcsányi [DDS13] vykonali analýzu prostredí na výučbu programovania na slovenských školách, v ktorej porovnali desať prostredí, medzi ktorými je aj EasyLogo.

Pri porovnávaní prostredí sa pozerali napríklad na to či je prostredie špecializované. To znamená, že prostredie zahŕňa úlohy a aktivity, ktoré si môže používateľ aj sám vytvárať.

Ďalej sa zaoberali typom programovania, teda či ide o procedurálne prog-

ramovanie ako v prípade EasyLoga alebo objektovo orientované či udalosťami riadené programovanie.

Ďalšími dôležitými aspektami, na ktoré sa pozreli, sú programové konštrukcie, ktoré jazyk ponúka(podmienky, cykly, procedúry, premenné), spôsob konštruovania programu (písanie, ťahanie, vyberanie) a forma reprezentácie kódu (textová podoba, kartičky, skladačky, ikony). Autori analýzy spomínajú, že niektoré prostredia, vrátane EasyLoga, využívajú až dve reprezentácie. V prípade EasyLoga používateľ najskôr potiahne príkaz vo forme ikony do programu, kde sa zmení do formy kartičky s názvom príkazu.

Autori porovnávali aj ako prostredie používateľa upozorňuje na syntaktické chyby v prípade programovania v textovej forme (možnosť výberu zo správnych možností, možnosť vpisovať len správny typ údajov). V prípade programovania vo vizuálnych programovacích jazykoch ako navádza používateľa k správnemu vytváraniu programu (prispôsobené, do seba zapadajúce tvary kartičiek).

Pri výučbe programovania najmä u detí na základných školách je veľmi dôležitá lokalizácia prostredia do materinského jazyka. Pridanou hodnotou preto je, ak je prostredie lokalizované vo viacerých jazykoch. Autori teda pri prostrediach porovnávali aj počet možných lokalizácií.

2.3.3 Kompilátor

Jedným z cieľov práce je vytvoriť kompilátor a virtuálnu mašinu.

Kompilátor prekladá program napísaný vo vyššom programovacom jazyku čitateľnom pre človeka do nižšieho programovacieho jazyka, respektíve strojového kódu čitateľného pre procesor počítača.

Virtuálna mašina je program, virtuálny stroj (procesor) ktorý umožňuje vykonávať inštrukcie medzikódu (bajtkódu) preloženého z vyššieho progra-

movacieho jazyka kompilátorom.

Fázy kompilácie

Kompilačný proces prebieha všeobecne viacerými fázami. Poradie či prerozdelenie týchto fáz sa však môže v kompilátoroch líšiť. Každá fáza berie výsledok predchádzajúcej fázy a posiela výsledok spracovania ďalšej fáze. V nasledujúcej podkapitole si priblížime fázy kompilácie, ktorými bude prechádzať aj kompilátor webovej verzie EasyLoga.

Lexikálna analýza

Prvou fázou kompilácie je lexikálna analýza. Vstupom v tejto fáze je samotný zdrojový kód. Úlohou lexikálneho analyzátora je zo vstupu odstrániť medzery, komentáre či nové riadky a z jednotlivých znakov poskladať elementárne jazykové prvky – lexémy.

Lexémy sú postupnosti znakov, ktoré delíme na slová, čísla a iné symboly. Reťazec s rozpoznanou lexémou v lexikálnom analyzátore nazývame token. Tokeny, ktoré sú podľa vopred stanovených pravidiel jazyka validné sa ďalej posunú na spracovanie syntaktickému analyzátoru.

Syntaktická analýza

Druhou fázou kompilačného procesu je syntaktická analýza. Úlohou syntaktického analyzátora je spracovávať lexémy, ktoré dostane na vstupe od lexikálneho analyzátora, pričom sa snaží rozpoznať jednotlivé programové konštrukcie.

Výsledkom syntaktickej analýzy môže byť dátová štruktúra reprezentujúca program, prípadne vyhodnotenie správnosti syntaxe programu, ale takisto aj samotná interpretácia a vykonanie programu.

Sémantická analýza

Ďalšou fázou kompilácie je sémantická analýza, ktorej úlohou je odhaliť sémantické chyby v zdrojovom kóde. V tejto fáze sa kontrolujú typy premenných, operánd v operátoroch či argumentov funkcií. Takisto sú v tejto fáze odhalené aj chyby ako nedeklarované premenné, viacnásobná deklarácia premenných a funkcií.

Generovanie kódu

Poslednou fázou kompilácie je generovanie samotného kódu. Vstupom v tejto fáze je optimalizovaný medzikód a jej výsledkom je zbiehateľný cieľový kód ako napríklad kód virtuálnej mašiny, assemblerový kód či samotný strojový kód. V tejto fáze sa priraďuje miesto v pamäti pre jednotlivé premenné a prekladajú sa inštrukcie medzikódu do postupností inštrukcií cieľového jazyka.

2.4 Existujúce riešenia

Existuje mnoho online dostupných aplikácií, ktoré sú zamerané na výučbu základov programovania. V nasledujúcej podkapitole popíšem bližšie tri z nich a to konkrétne programové prostredia Scratch, Baltie a Robot Karel, nakoľko sú tieto aplikácie voľne dostupné na internete.

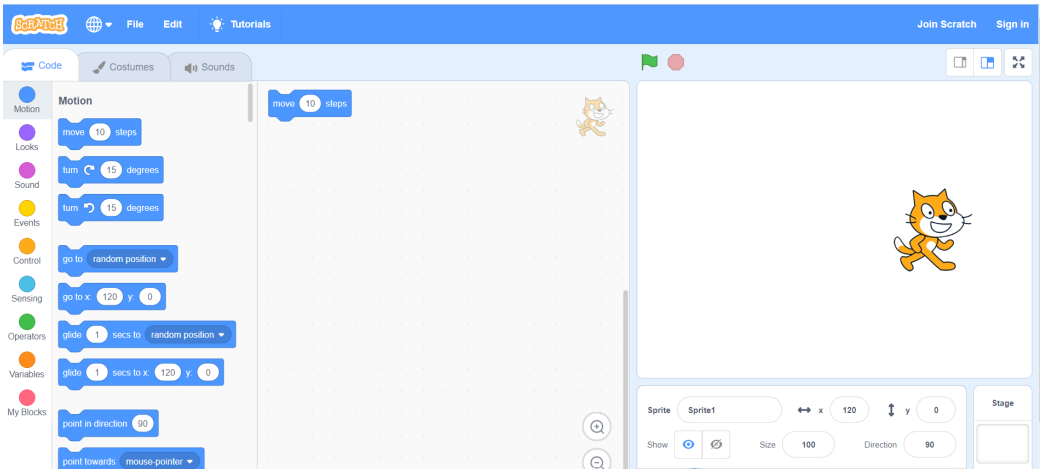
2.4.1 Scratch

Scratch je vyšší programovací jazyk, ktorý umožňuje jednoduché vytváranie interaktívnych príbehov, hier a animácii.

Scratch navrhli v roku 2003 Mitchel Resnick, Yasmin Kafai a John Maeda. Neskôr bolo vyvinuté prostredie pre tento programovací jazyk skupinou

Lifelong Kindergarten Group z MIT Media Lab, ktorú viedol Mitchel Resnick.

Jazyk Scratch je vizuálny programovací jazyk, v ktorom užívateľ programuje, tak že jednotlivé príkazy vo forme skladačky spája a vytvára bloky. Scratch obsahuje príkazy na ovládanie pohybu postavičky, cykly, premenné, operátory, podmienky, prácu s myšou, zvukom a obrazom, ale takisto príkazy na vytváranie vlastných procedúr a funkcií.



Obr. 2.4: Ukážka programovacieho prostredia jazyka Scratch (prevzaté z [scr])

Používateľské rozhranie aplikácie Scratch je rozdelené na 3 hlavné bloky. V prvom bloku naľavo sú ikony príkazov, ktoré sa dajú ťahať do hlavného bloku (v strede). Scratch ponúka veľké množstvo príkazov, preto sú rozdelené do charakteristických skupín, aby bolo jednoduché medzi príkazmi vyhľadávať a pracovať s nimi. V strede je blok, ktorý obsahuje aktuálne vytváraný program a na pravo je blok s grafickou plochou kde vidíme vykonávanie programu. Pri vykresľovaní môžeme nastavovať rôzne parametre ako pozadie, veľkosť grafickej plochy ale aj konkrétnu postavičku s ktorou sa po grafickej ploche hýbeme, jej počiatočné súradnice a smer.

2.4.2 Baltie

Baltie predstavuje programovací a kresliaci nástroj najmä pre deti na výučbu programovania. Ide o programovací jazyk, ktorý vychádza zo syntaxe jazyka C. V programovom prostredí Baltie sa program sa vytvára poťahovaním ikon do správneho poradia.

Pomocou tohto nástroja je možné vytvárať rôzne animácie, prezentácie, hry či programy.

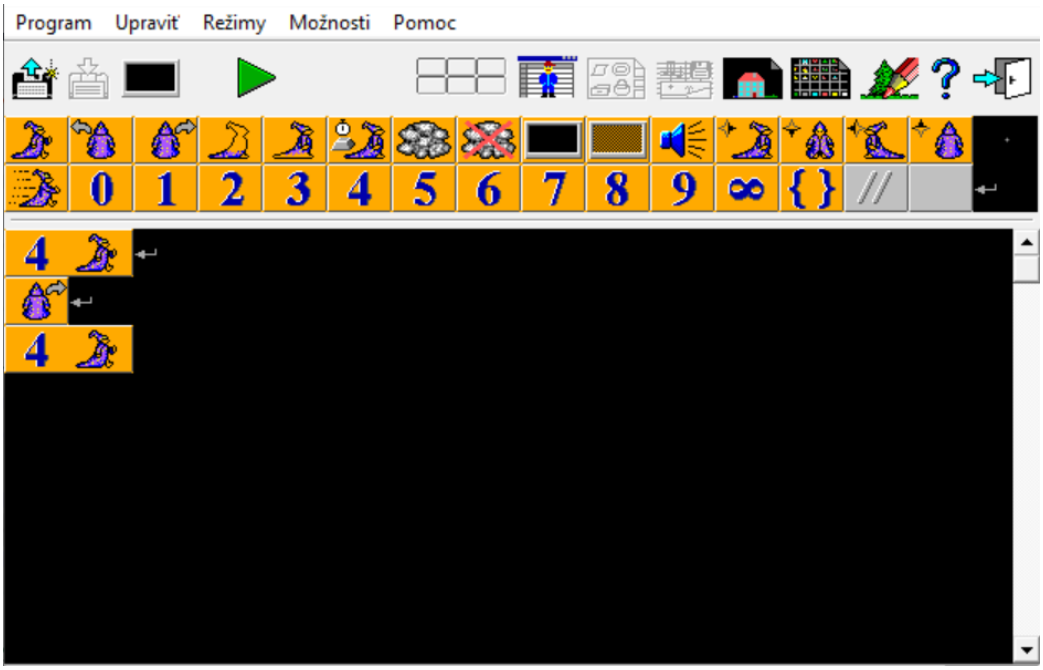
Baltie obsahuje štandardné programovacie príkazy ako podmienky, cykly, logické operátory, bitové operátory, premenné, polia, procedúry, rekurziu ale takisto rôzne príkazy pre animáciu a prácu s dátami. Pre ladenie programu je v tomto programovom prostredí k dispozícií aj krokovanie a zobrazenie premenných.

Baltie ponúka 3 režimy:

- Režim Skladať scénu - v tomto režime si môže používateľ vytvárať rôzne obrázky a scény pomocou predmetov umiestnených v bankách či prípadne si dotvoriť vlastné predmety.

- Režim Čarovať scénu - v tomto režime si môže používateľ pomocou príkazov ovládajúcich pohyb a čarovanie Baltíka natrénovať vytváranie správnych postupností príkazov.

- Režim Programovať - tento režim obsahuje konkrétne príkazy, premenné, procedúry, vkladanie zvukových i obrázkových záznamov či príkazy na vytváranie animácií.



Obr. 2.5: Ukážka programovacieho prostredia Baltie - režim Programovať

Prostredie je v režime *Programovať* horizontálne rozdelené na 3 bloky. V prvom sa nachádzajú tlačidlá na uloženie, načítanie a spúšťanie programu. V druhom sa nachádzajú samotné príkazy vo forme kartičiek s ikonami, ktoré používateľ ťahá do tretieho bloku, kde vytvára samotný program.

Režim *Programovať* je možné prepnúť do režimu *Začiatočník* alebo režimu *Pokročilý*. V režime *Začiatočník* sa nachádzajú v paneli s príkazmi len základné príkazy ako môžme vidieť na obrázku (2.5). V režime *Pokročilý* sú používateľovi prístupné všetky príkazy a programové konštrukcie jazyka. Po spustení programu sa editor vypne a vykreslí sa grafická plocha (2.6).

Obr. 2.6: Ukážka programovacieho prostredia Baltie - beh programu

Verzia programového prostredia, s ktorou som pracovala je demo verzia Baltík 3 a ide o staršiu verziu, ktorej používateľské prostredie nie je úplne intuitívne. Na webovej stránke SGP Systems [bal] sú však voľne dostupné edukačné videá pre prácu a programovanie v prostredí Baltík.

2.4.3 Robot Karel

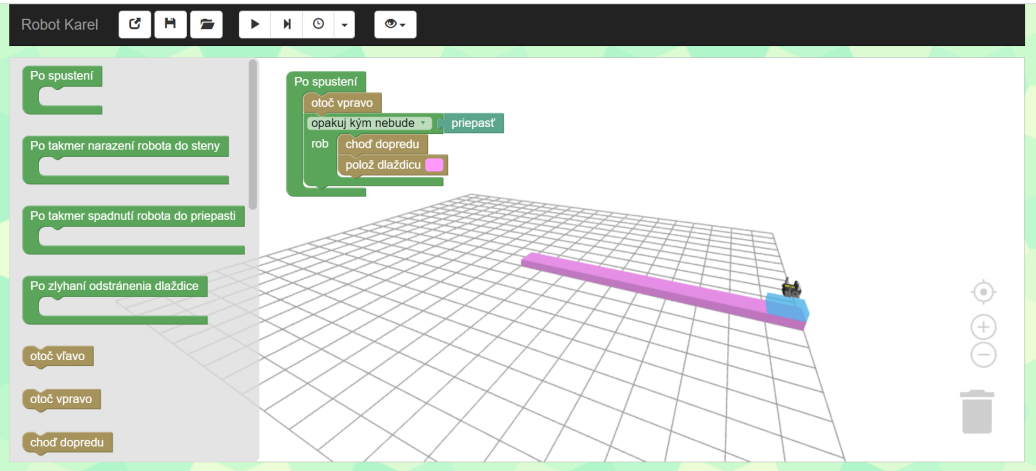
Robot Karel je programovací jazyk určený najmä pre začiatočníkov, ktorý navrhol Richard Eric Pattis v roku 1981. Jazyk dostal pomenovanie podľa českého autora Karla Čapka, ktorý ako prvý v scenári divadelnej hry R.U.R. použil slovo robot pre stroj podobajúci sa človeku.

Používateľ mohol ovládať robota pohybujúceho sa po miestnosti, kde sa nachádzali steny, tehly a ďalšie objekty.

Jazyk okrem piatich základných príkazov: *move* (choď dopredu), *turn-*

left (otoč sa vľavo), *putbeeper* (polož), *pickbeeper* (zober) a *turnoff* (vypni sa) obsahoval aj základné programové konštrukcie jazyka Pascal ako cykly, podmienky a procedúry pre riadenie činnosti programu.

V súčasnosti existuje mnoho verzií, rozšírení a podôb tohto programovacieho jazyka, pre porovnanie som si vybrala verziu, ktorú v roku 2016 naprogramoval ako diplomovú prácu Jakub Jantošík, nakoľko v tejto verzii používateľ vytvára program skladaním príkazov do blokov.



Obr. 2.7: Ukážka programovacieho prostredia Robot Karel

Prostredie [2.7] obsahuje grafickú plochu rozdelenú na 2 hlavné časti a horný panel, ktorý slúži na ukladanie, spúšťanie, krokovanie programu a otáčanie kamery. V prvej časti grafickej plochy sa nachádzajú príkazy, z ktorých ťahaním a spájaním používateľ vytvára program. Druhou časťou je priestor, v ktorom sa pohybuje robot. Prostredie umožňuje používateľovi pozerať sa na robota ortogonálne a perspektívne z rôznych uhlov.

Prostredie je intuitívne a jednoduché na používanie, preto je vhodné aj pre deti či ľudí so základnými počítačovými znalosťami.

Kapitola 3

Návrh

V tejto kapitole sa budem venovať návrhu webovej verzie pre aplikáciu EasyLogo.

3.1 Technológie

Vzhľadom k požiadavkám na vlastnosti webovej verzie aplikácie EasyLogo ako sú ľahká prenositeľnosť, jednoduché spustenie vo web prehliadači, nezávislosť od webového prehliadača a pripojenia k internetu som sa rozhodla pre implementáciu použiť okrem HTML, CSS a programovacieho jazyka Javascript takisto aj knižnice týchto technológií, ktoré sú vhodné pre použitie bez internetového pripojenia a nutnosti servera.

Vue.js je javascriptový framework pre vytváranie používateľského rozhrania a single page aplikácií. Framework Vue využíva šablóny a komponenty, ktoré sú založené na HTML syntaxi, takže ich prehliadač dokáže čítať.

Bootstrap je CSS framework pre vytváranie responzívnych webových aplikácií

cií. Obsahuje rôzne šablóny pre tlačidlá, formuláre, navigáciu, texty a ďalšie komponenty.

3.2 Návrh používateľského rozhrania

Pri návrhu používateľského rozhrania som vychádzala najmä z existujúcej desktopovej verzie aplikácie EasyLogo.

Základné rozloženie a dizajn jednotlivých častí aplikácie ako sú grafická plocha, blok na ťahanie príkazov a horná navigácia je rovnaké ako v desktopovej verzii.

3.3 Návrh kompilátora

Pri návrhu kompilátora

3.4 Návrh virtuálnej mašiny

Kapitola 4

Implementácia

V tejto kapitole

Kapitola 5

Výskum

Nakoľko výsledná aplikácia je použiteľná v triede so žiakmi, súčasťou práce je aj výskum, ktorý spočíva v testovaní vytvorenej aplikácie so žiakmi a učiteľmi.

V nasledujúcej kapitole bližšie popíšem jednotlivé testovacie scenáre, priebeh a výsledky testovania.

5.1 Ciele testovania

Cieľom testovania webovej verzie prostredia EasyLogo je otestovať či aplikácia spĺňa všetky funkčné a nefunkčné požiadavky, takisto či je aplikácia jednoduchá a intuitívna na používanie.

Ďalej chceme zistiť, či tester bude pri zadanej úlohe postupovať, tak ako očakávame a či počas testovania narazí na nejaký problém, prípadne spraví krok, ktorý sme nečakali. Naším cieľom bude aj sledovať jeho výraz tváre a správanie, aby sme zistili, aké pocity v ňom aplikácia počas vykonávania úlohy vzbudzuje, t.j. či veľmi dlho premýšľa kam kliknúť, prezrádza výraz jeho tváre, že nevie ako ďalej v aplikácii postupovať, prípadne sa pýta kon-

trolné otázky.

Ďalším cieľom nášho testovania je aj objaviť prípadné chyby aplikácie či získať názor testera, či by niečo v aplikácii zlepšil, zmenil prípadne čo sa mu počas testovania v aplikácii nezdalo jasné a intuitívne.

5.2 Testeri

5.3 Testovacie scenáre

5.3.1 Testovací scenár č.1

Literatúra

[ALSU07] A.V. Aho, M.S. Lam, R. Sethi, and J.D. Ullman. *Compilers: Principles, Techniques, & Tools*. Pearson/Addison Wesley, 2007.

[bal] Baltie. `https://www.sgpsys.com/cz/`. Navštívené: január 2022.

[com] Compiler Design. `https://www.tutorialspoint.com/compiler_design/`. Navštívené: 30. apríl 2021.

[DDS13] Ira Diethelm, Malte Dünnebier, and Jörn Syrbe. *Informatics in Schools: Local Proceedings of the 6th International Conference ISSEP 2013 - Selected Papers*. 01 2013.

[FHSS18] Martina Forster, Urs Hauser, Giovanni Serafini, and Jacqueline Staub. *Autonomous Recovery from Programming Errors Made by Primary School Children: 11th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2018, St. Petersburg, Russia, October 10-12, 2018, Proceedings*, pages 17–29. 10 2018.

[HSS17] Juraj Hromkovic, Giovanni Serafini, and Jacqueline Staub. Xlogoonline: A single-page, browser-based programming envi-

ronment for schools aiming at reducing cognitive load on pupils. pages 219–231, 11 2017.

[Jan16] J. Jantošík. Robot karel. Diplomová práca, Katedra základov a vyučovania informatiky, Fakulta matematiky, fyziky a informatiky, Univerzita Komenského v Bratislave, 2016.

[Leh07] Daniela Lehotská. Edukačný softvér. In *Matematika Informatika Fyzika*, 2007.

[MuDoCS09] T.Æ. Mogensen and Københavns universitet. Department of Computer Science. *Basics of Compiler Design*. Torben Ægidius Mogensen, 2009.

[rob] Robot Karel. `http://karel.oldium.net/`. Navštívené: november 2021.

[scr] Scratch. `https://scratch.mit.edu/`. Navštívené: november 2021.

[tyn] Tynker. `https://www.tynker.com`. Navštívené: november 2021.

[10] Ľubomír Salanci. Easylogo - discovering basic programming concepts in a constructive manner. In *Constructionism, Paris*, 2010.

Zoznam obrázkov