UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

# ZVUKOVÝ PROGRAMOVACÍ JAZYK PRÍSTUPNÝ PRE NEVIDIACICH ŽIAKOV

**Diplomová práca**

**2021** **Bc. Ivana Nemsilajová**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

# ZVUKOVÝ PROGRAMOVACÍ JAZYK PRÍSTUPNÝ PRE NEVIDIACICH ŽIAKOV

**Diplomová práca**

| | |
|---|---|
| Študijný program: | Aplikovaná informatika |
| Študijný odbor: | 2511 Aplikovaná informatika |
| Školiace pracovisko: | Katedra didaktiky matematiky, fyziky a informatiky |
| Školiteľ: | doc. RNDr. Ľudmila Jašková, PhD. |

**2021** **Bc. Ivana Nemsilajová**


Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

## ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Bc. Ivana Nemsilajová
**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:** Zvukový programovací jazyk prístupný pre nevidiacich žiakov
*Blind friendly audio programming language*

**Anotácia:** Autorka vytvorí plne ozvučené detské programovacie prostredie s vlastným kompilátorom alebo interpreterom. Základné príkazy zabudovaného programovacieho jazyka budú slúžiť na prehratie zvukového súboru alebo vyslovenie zadaného textu. Tieto príkazy bude možné použiť aj v rámci komplikovanejších štruktúr, ako je cyklus, príkaz vetvenia, podprogram. Okrem toho bude možné pracovať s celočíselnými premennými (definovať ich, inicializovať, priradiť im náhodnú hodnotu, inkrementovať, dekrementovať a podobne). Editor kódu bude mať zabudovanú kontrolu syntaxe a funkciu ponuky príkazov.
Ako vývojový nástroj zvažujeme zvoliť prostredie používajúce jazyk Java.

**Cieľ:** Vytvoriť ozvučené programovacie prostredie na báze jazyka umožňujúceho programovať zvukové príbehy pozostávajúce zo sekvencie hovoreného slova a nahratých zvukov.

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

**Dátum zadania:** 08.10.2020

**Dátum schválenia:** 10.10.2020 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že som diplomovú prácu s názvom "Zvukový programovací jazyk prístupný pre nevidiacich žiakov"vypracovala samostatne pod vedením vedúcej diplomovej práce, s použitím uvedenej literatúry a zdrojov.

.....

Bratislava, 2021 Bc. Ivana Nemsilajová

**Poďakovanie:** .....

# Abstrakt

NEMSILAJOVÁ, Ivana: Zvukový programovací jazyk prístupný pre nevidiacich žiakov (Diplomová práca) – Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. – Školiteľ: doc. RNDr. Ľudmila Jašková, PhD.: FMFI UK, 2022, xy strán

Cieľom tejto diplomovej práce je vytvoriť zvukové programovacie prostredie pre nevidiacich žiakov, ...

**Kľúčové slová:** programovacie prostredie, nevidiaci žiaci, interpreter, zvukový programovací jazyk

# Abstract

NEMSILAJOVÁ, Ivana: Blind friendly audio programming language) – Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics. – Supervisor: doc. RNDr. Ľudmila Jašková, PhD.: FMFI UK, 2020, xy pages

The goal of this bachelor thesis is to develop...

**Keywords:**

# Obsah

**Úvod** **1**

**1 Východiská** **2**

1.1 Kategorizácia osôb so zrakovým postihnutím . . . . . 2

1.2 Tvorba softvéru pre žiakov so zrakovým postihnutím . . . . . 3

1.2.1 Asistenčné technológie . . . . . 4

1.3 Existujúce programovacie prostredia pre nevidiacich . . . . . 5

1.3.1 APL: Zvukový programovací jazyk pre nevidiacich . . . 5

1.3.2 CodeTalk . . . . . 6

1.3.3 Code Jumper . . . . . 7

1.3.4 Alan2 . . . . . 8

1.4 Použité technológie . . . . . 10

1.4.1 Vývojové nástroje . . . . . 10

1.4.2 Technológie pre nevidiacich . . . . . 11

1.5 Kompilátory a interpretre . . . . . 12

1.5.1 Kompilátor . . . . . 12

1.5.2 Interpreter . . . . . 13

**2 Špecifikácia** **15**

2.1 Používateľské rozhranie . . . . . 15

2.2 Vstup programu . . . . . 16

2.3 Výstup programu . . . . . 16

2.4 Špecifikácia jazyka . . . . . 17

**3 Návrh** **18**
3.1 Návrh používateľského rozhrania . . . . . 18
3.2 Návrh jazyka . . . . . 18

**4 Implementácia** **19**
4.1 Štruktúra kódu . . . . . 19
4.2 Analýza textu . . . . . 19
4.3 Vykonávanie príkazov . . . . . 19
4.4 Syntéza reči . . . . . 19

**5 Testovanie** **20**
5.1 Priebeh testovania . . . . . 20
5.1.1 Popis vzorky respondentov . . . . . 20
5.1.2 Popis metodiky pri overovaní . . . . . 20
5.2 Výsledky testovania . . . . . 21

**Záver** **22**

**Príloha** **25**

# Zoznam obrázkov

1.1 Braillov displej . . . . . 5
1.2 Prostredie Code Jumper . . . . . 8
1.3 Prostredie Alan2 . . . . . 9

# Zoznam tabuliek

# Úvod

Cieľom tejto práce je ... Programovacie prostredia, ktoré sú vývojárom dobre známe sú zamerané na vidiacich programátov. Človek so zrakovým postihnutím môže mať s týmito softvérmi problém aj napriek využitiu asistenčných technológií. Skúsený nedidiaci programátor sa s týmito prekážkami dokáže popasovať, ale pre začiatočníkov je to veľká bariéra. Väčšina softvérov vytvorených na výučbu programovania pre začiatočníkov sú zamrané na vizuálny výstup, ktorý je prakticky nemožné plnohodnotne interpretovať nevidiacim. Z toho dôvodu veľa nádejných nevidiacich programátorov ani nemôže nazrieť do informatického sveta a prebudiť v sebe vývojára, ktorý by priniesol nové inovácie a pokroky.

# Kapitola 1

# Východiská

V prvej kapitole si prejdeme prehľad informácií z oblasti tvorby softvéru pre žiakov so zrakovým postihnutím. Kapitola je rozdelená do piatich častí.
V prvej časti si kategorizujeme druhy zrakového postihnutia a ozrejmíme, na ktorú kategóriu žiakov sa zameriame.
Druhá časť nám poskytne prehľad pomôcok, ktoré používajú nevidiaci pri práci s počítačom.
Tretia časť je venovaná prehľadu už existujúcich programovacích prostredí pre nevidiacich, pričom sa budeme venovať rozdielne zameraným programom, od softvéru pre deti až po doplnky pre pokročilých vývojárov.
V štvrtej časti bližšie popíšeme technológie, ktoré využijeme pri vývoji nášho prostredia, aby čo najlepšie pokrylo požiadavky používateľov.
V poslednej časti si vysvetlíme rozdiel medzi kompilátorom a interpretrom a vyberieme, ktorý spôsob spracovania kódu budeme používať v našej práci.

## 1.1 Kategorizácia osôb so zrakovým postihnutím

V literatúre možno nájsť rôzne kategorizácie osôb so zrakovým postihnutím. Keďže vytvárame sofotvér pre žiakov základnej školy, dôležitý je aspekt prijímania informácií vo vzdelávaní. Preto uvádzame kategorizáciu pochádzajucu z oblasti špeciálnej pedagogiky.

V špeciálnej pedagogike delíme žiakov so zrakovým postihnutím do štyroch skupín [7]:

- **Nevidiaci -** nie sú schopní vnímať okolitý svet zrakom, vytvárať vizuálne vnemy a predstavy, čo spôsobuje horšiu priestorovú orientáciu a mobilitu.
- **Čiastočne vidiaci -** vidia v obmedzenej miere a nie sú schopní orientovať sa podľa zraku.
- **Slabozrakí -** majú zníženú zrakovú ostrosť a zníženú schopnosť zrakového vnímania, čo sa prejavuje znížením rýchlosti a presnosti zrakvého vnímania a rýchlejšou únavou pri zrakovej práci.
- **Žiaci s poruchami binokulárneho videnia -** poruchy videnia za vzájomnej spolupráce oboch očí.

Programovacie prostredie, ktoré sme vytvorili je určené pre nevidiacich žiakov a je prispôsobené ich špecifickým potrebám pre prácu s počítačom.

## 1.2 Tvorba softvéru pre žiakov so zrakovým postihnutím

Aby bol náš softvér prístupný pre nevidiacich žiakov, musíme pri jeho tvorbe myslieť na niekoľko hlavných princípov: [5]:

- Vizuálne informácie, ako sú grafika, farby či rozloženie elementov, používateľ nie je schopný vnímať.
- Namiesto myši budú používateľia na navigáciu a ovládanie softvéru používať klávesnicu.
- Technológie musia byť kompatibilné s asistenčnými technológiami.

### 1.2.1 Asistenčné technológie

Používatelia elektornických zariadení dostávajú výstup zväčša vo vizuálnej podobe, preto je potrebné, aby mali ľudia so zrakovým postihnutím na interakciu so zariadeniami rôzne pomôcky. V tejto časti sa pozrieme na najpoužívanejšie technológie, ktoré pomôžu nevidiacim pri práci s počítačom.

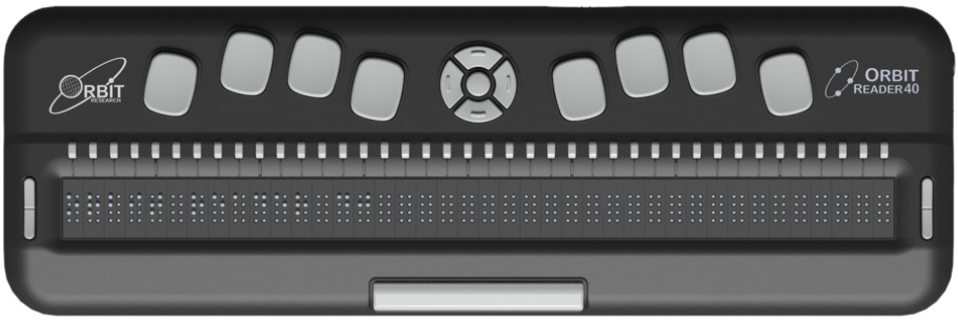
#### Čítač obrazovky

Čítač obrazovky [5] je základnou asistenčnou technológiou pre nevidiacich a slabozrakých používateľov pri práci s počítačom. Je to softvér, ktorý transformuje textový a obrazový obsah aplikácie alebo webovej stránky na syntetizovanú reč. Hoci čítač obrazovky nedokáže plnohodnotne replikovať vizuálny obsah, pretože nedokáže analyzovať obrázky, ako vývojari môžme tomuto nedostatku pomôcť tým, že pridáme obrázkom alternatívny text.

Väčšina čítačov transformuje obsah obrazovky do zvukovej podoby, ale existujú aj čítače, ktoré zobrazujú údaje prostredníctvom Braillovho displeja.

#### Braillov displej

Obnoviteľný Braillov displej [12] (obrázok 1.1) je zariadenie zobrazujúce text v podobe Braillovho písma. Používateľ číta text hmatom. Výhodou tohto zariadenia je, že narozdiel od syntetizéra reči ho môžu používať nepočujúci aj nevidiaci ľudia.
Žiaľ, narozdiel od softvéru na čítanie obrazovky sú tieto zariadenia veľmi drahé a nie každý si ich môže finančne dovoliť.




Obr. 1.1: Braillov displej

## 1.3 Existujúce programovacie prostredia pre nevidiacich

Ako sme už spomínali v úvode, vhodných programovacích prostredí pre nevidiacich žiakov je nedostatok.
V tejto časti sa pozrieme na niekoľko existujúcich nástrojov, ktoré pomáhajú začiatočníkom získať prvé skúsenosti, ale aj pokročilejšie zručnosti v oblasti programovania.

### 1.3.1 APL: Zvukový programovací jazyk pre nevidiacich

APL (Audio Programming Language for Blind Leaners) [11] je zvukový programovací jazyk navrhnutý špeciálne pre nevidiacich študentov. Je založený na zvukovom rozhraní, čím podporuje začínajucich nevidiacich študentov, aby mohli rozvíjať a precvičovať svoje zručnosti v riešení algoritmických problémov. APL sa teda nepovažuje za alternatívu ku konvenčným programovacím jazykom, ale len za nástroj, ktorý pomáha pri lepšej integrácii nevidiacih programátorov. Bol vyvinutý v Jave a používa FreeTTS Java speech synthesizer.

Autori testovali použiteľnosť tohto jazyka počas implementácie aj po jej skončení a výsledky ukázali, že APL motivoval nevidiacich študentov strednej školy k písaniu programov a ďalšiemu rozvoju zručností.
Naše prostredie sa bude od APL líšiť v tom, že príkazy jazyka budú po slovensky, tým pádom budú pre deti intuitívnejšie a ľahšie pochopia ich funkcionalitu.

### 1.3.2 CodeTalk

CodeTalk [10] je doplnok pre nevidiacich implementovaný v programovacom prostredí Visual Studio. Tvorcovia sa zamerali na výzvy, ktorým nevidiaci čelia v programovacích prostrediach založených na grafickom rozhraní. Extrahuje informácie súvisiace s kontextom a umožní vývojarovi prístup so zníženým úsilím.
Funkcie, ktoré sú v CodeTalk implementované:

- Po otvorení súboru sa používateľovi spraví zhrnutie kódu - informácie o triedach a funkciách v súbore zhrnie do prístupného stromovo usporiadaného prehľadu. Vývojár používajúci čítač obrazovky si ho môže prejsť a pochopiť tak štruktúru kódu.
- V prípade potreby vie čítač povedať kontext aktuálneho riadku, triedy a menný priestor, ku ktorému riadok patrí.
- Informácie o chybe sa programátor dozvie namiesto červeno vyznačeného riadku pomocou proaktívnych tónov chýb.
- Umožňuje zvukové debuggovanie.

Nevidiaci programátori sa vyjadrili, že netušili koľko informácií vidiaci programátori dostávajú, pretože im ich nikdy čítač obrazovky toľko neposkytol. Celkové hodnotenie tohto doplnku bolo v priemere 8,8 z 10, čo je vysoká úspešnosť a zjednodušenie života mnohým zrakovo postihnutým používateľom Visual Studia.

### 1.3.3 Code Jumper

Code Jumper [9] (obázok 1.2) je fyzický programovací jazyk určený najmä pre deti vo veku 7-11 rokov. Prostrdie bolo vytvorené pre deti so zrakovým postihnutí, bez ohľadu na úroveň zraku. Prostredie je zložené z hardvérovej a softvérovej časti.
Hardvérová časť sa skladá z fyzických blokov, reprezentujúcich príkazy na prehratie hudby, príkaz pauzy, cyklu a podmienky. Každý blok má niekoľko konektorov a káblov, čo umožňuje spájanie blokov, ich prepojenie a definovanie štruktúry programu. Bloky sa zapájajú do konektorov na hlavnej jednotke(Hube), ktorá predstavuje logický východiskový bod. Bloky príkazov boli navrhnuté tak, aby boli hmatovo aj vizuálne odlišné. Majú svoje otáčacie tlačidlá na nastavenie parametrov. Taktiež sa dajú do nich vložiť zásuvné bloky reprezentujúce konštanty, náhodné čísla, nekonečno, pripočítanie a zníženie počítadla a premenné.
Prostredie bolo navrhnuté tak, aby bolo vhodné aj pre mladšie deti (7 rokov), ktoré začnú s jednoduchými programami a naučia sa čo je postupnosť príkazov. Zároveň prostredie poskytuje dostatočný rozsah na rozvíjanie znalostí a konštruovanie vnorených cyklov a podmienok aj pre staršie deti (11 rokov).
Vytvorený program sa zapisuje do textovej podoby(kódu) cez softvérovú časť. Textovú podobu si môžu žiaci vypočuť cez tlačidlo prehrávania na hlavnej jednotke alebo pomocou asistenčných technológií v aplikácii.
Nakoľko je možné, aby žiaci pracovali iba s fyzickými blokmi prostredia, môžu sa vďaka nemu učiť programovacie konštrukcie aj žiaci, ktorí nevedia pracovať s počítačom. Na druhej strane môžme povedať, že hoci budú mať žiaci základné znalosti z programovania, ale nevedia pracovať s počítačom, bude sa im ťažko prechádzať do iných programovacích jazykov. V našom prostredí síce budú musieť ovládať základnú prácu s počítačom alebo sa ju naučiť, ale keď sa ju raz naučia, budú oveľa flexibilnejší a bude pre nich jednoduchšie prejsť na iné programovacie jazyky.

Obr. 1.2: Prostredie Code Jumper

### 1.3.4 Alan2

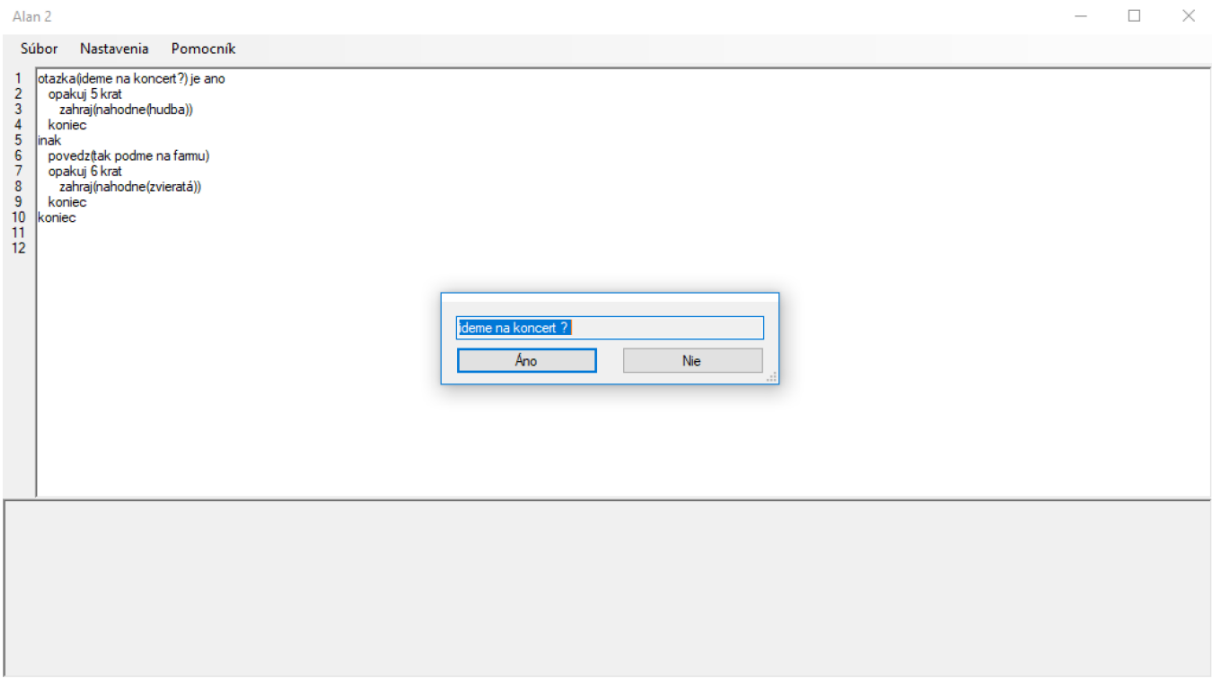
Alan2 [6] je zvukové programovacie prostredie pre nevidiacich žiakov nižšieho sekundárneho vzdelávania, ktoré ako svoju diplomovú prácu vytvoril Matúš Kováč. Prostredie má slúžiť pri výučbe informatiky na rozvoj algoritmického myslenia. Výsledná aplikácia (obrázok 1.3) je jednoduchá a ľahko sa v nej zorientujeme. Je možné ju ovládať iba pomocou klávesnice. Hlavnou časťou je textové pole, do ktorého píšu používatelia program a ten vedia následne spustiť.

Jazyk prostredia je navrhnutý po slovensky s intuitívnymi príkazmi. Dva základné príkazy jazyka sú prehratie zvuku a povedanie textu. Ďalej sú v jazyku príkazy cyklu, vetvenia, podmienok, podprogramu aj priradenia. Výstupom programu je postupnosť zvukov, vzniknutá zo vstupného kódu.

Nedostatkom aplikácie je, že kontextová ponuka predikovaných príkazov nespolupracuje s čítačom obrazovky. Takže hoci sa kontextová ponuka zobrazí,

nevidiaci žiak sa o nej nikdy nedozvie, pretože ju čítač neprečíta. Vhodným vylepšením by bolo, keby pri implementácii príkazu povedz() čítal text čítač obrazovky a nie syntetizér textu eSpeak, pretože používa iný, menej prirodzený a ťažšie zrozumiteľný hlas ako čítač, z čoho môžu byť žiaci zmätení. Ďalším nedostatkom je, že eSpeak nečíta správne niektoré písmená s interpunkciou.

Týmto nedostatkom sa v našom prostredí pokúsime vyhnúť a syntax nášho jazyka bude o čosi jednoduchšia. Výrazným rozdielom bude, že eliminujeme zátvorky, aby sme sa vyhli zbytočným chybám v konštrukcii príkazov, ktoré by mohli nevidiaci kvôli zátvorkám spraviť.




Obr. 1.3: Prostredie Alan2

## 1.4 Použité technológie

V tejto časti sa pozrieme na technológie, ktoré sme zvolili na vývoj našej aplikácie a taktiež na tie, ktoré sme museli použiť z dôvodu vývoja softvéru pre nevidiacich.

### 1.4.1 Vývojové nástroje

#### Java

Java je objektovo orientovaný programovací jazyk. Zdrojové kódy sa nekompilujú do strojového kódu, ale do byte-code, ktorý nie je závislý od konkrétnej platformy. Byte-code spracováva interpreter - Java Virtual Machine. Java je jeden z najpopulánejších programovacích jazykov, pretože funguje na rôznych platformách(Windows, Mac, Linux,...).
Použitie Javy pre našu aplikáciu sme zvolili z dôvodu dobrej kompatibility s čítačom obrazovky.

#### JavaFX

JavaFX je knižnica pre programovací jazyk Java na vytváranie bohatých desktopových a webových aplikácií, ktoré môžu bežať konzistentne na viacerých platformách. Obsahuje množstvo grafických a mediálnych API. Aplikácie vyvinuté pomocou JavaFX môžu bežať na rôznych zariadeniach, ako sú počítače, mobilné telefóny, tablet, televízory atď. JavaFX obsahuje všetky funkcie, na ktoré potrebovali predtým programátori rôzne knižnice a má plnú podporu Accessibility API.

#### IntelliJ IDEA

IntelliJ IDEA [1] je integrované vývojové prostredie od firmy JetBrains pre jazyky JVM, ako sú Java, Kotlin, Scala či Groovy, navrhnuté tak, aby maximalizovalo produktivitu vývojárov. Je to multiplatformové IDE, ktoré poskytuje konzistentné skúsenosti v systémoch Windows, MacOS a Linux. Prostredie

poskytuje šikovné dokončovanie kódu, statickú analýzu kódu a refraktorovanie.

### 1.4.2 Technológie pre nevidiacich

#### Čítač obrazovky NVDA

Čítač obrazovky NVDA(NonVisual Desktop Access) [4] je voľne dostupný čítač obrazovky pre operačný systém Windows. Má otvorený zdrojový kód, čo umožňuje prekladateľom a vývojárom na celom svete neustále prispievať k jeho rozširovaniu a zlepšovať tento softvér. NVDA je preložený do viac ako 55 jazykov a používajú ho ľudia vo vyše 175 krajinách.
Tento softvér vytvorili dvaja nevidiaci programátori, aby sprístupnili prácu s počítačom miliónom nevidiacich ľudí, pre ktorých bol Braillov displej finančne neprístupný. Týmto ľudom ulahčili prístup k vzdelaniu a zamestnaniu, nehovoriac o každodenných funkciách, ako je online bankovníctvo, nakupovaie či posielanie správ.
Okrem všeobecných funkcií vo Windowse, NVDA pracuje so softvérmi Microsoft, WordPad, NotePad, Windows Media Player, s prehliadačmi Google Chrome, Mozilla Firefox, Internet Explorer a Microsoft Edge. Taktiež podporuje Outlook a väčšinu funkcií Microsoft Word, Excel a PowerPoint. Niektoré ďalšie kancelárske balíky sú podporované prostredníctvom Java Access Bridge.

#### Java Accessibility API

Java Accessibility API [3] je rozhranie, ktoré umožňuje vytvárať aplikácie písané v jazyku Java, ktoré sú prístupné osobám so zdravotným postihnutím. Aplikácie sú kompatibilné s asistenčnými technológiami, ako sú čítačky obrazovky, zväčšovacie obrazovky, systémy rozpoznávania reči a obnoviteľné Braillovo písmo. Java Accessibility API sprístupňuje informácie o komponentoch grafického používateľského rozhrania asistenčným technológiám a poskytuje používateľom alternatívnu prezentáciu a ovládanie aplikácií v Jave.

#### Java Access Bridge

Java Access Bridge [2] je technológia, ktorá umožňuje aplikáciám Java a appletom, ktoré implementujú rozhrania Java Accessibility API, zviditeľniť asistenčné technológie v systémoch Microsoft Windows. API je súčasťou Java Accessibility Utilities, čo je sada pomocných tried ktoré pomáhajú asistenčným technológiám poskytovať prístup k súpravám nástrojov používateľského rozhrania, ktoré implementujú Java Accessibility API.
Po spustení v systéme Microsoft Windows komunikuje aplikácia s knižnicami DLL Java Accsess Bridge, ktoré komunikujú prostredníctvom knižníc Java Access Bridge s Java Virtual Machine. Tieto knižnice komunikujú s Java Accessibility Utilities, ktoré zhromažďujú informácie o tom, čo sa deje v aplikácii a preposiela ich čítačke obrazovky cez Java Access Bridge.

## 1.5 Kompilátory a interpretre

Keďže vytvárame prostredie s vlastným programovacím jazykom, dôležitou otázkou je, aký spôsob spracovania vstupného kódu v našom jazyku si zvolíme.
Najskôr sa pozrieme, na spôsoby, ktoré môžme zvoliť a potom si vyberieme, akým spôsobom budeme postupovať v našom programe.

### 1.5.1 Kompilátor

Programátori píšu kód vo vysokoúrovňovom jazyku, pretože je bližší ľudskej reči, ľahšie sa v ňom uvažuje o problémoch a jeho zdrojový kód je kratší. Kompilátor [8] prekladá program napísaný vo vysokoúrovňovom programovacom jazyku, do nízkoúrovňového strojového kódu. Počas tohto procesu zároveň odhaľuje a nahlasuje chyby v zdrojovom kóde.
Keďže je vývoj kompilátora zložitá úloha, je dobré si prácu štruktúrovať a kompiláciu rozdeliť na niekoľko fáz. Fázy majú isté usporiadanie, ale v niektorých kompilátoroch sa môže ich poradie líšiť, môžu byť kombinované alebo rozdelené medzi iné fázy, alebo môže byť pridaná ďalšia fáza.

Základné fázy kompilátora a ich poradie:

- **Lexikálna analýza -** lexikálny analyzátor dostane ako vstup postupnosť znakov (zdrojový kód) a odfiltrovaním medzier, nových riadkov atď. rozdelí reťazec na tokeny. Tokeny sú názvy premenných, čísla, kľúčové slová a pod. Cieľom lexikálnej analýzy je uľahčiť prácu syntaktickej analýze.
- **Syntaktická analýza -** ako vstup dostane zoznam tokenov, vytvorený lexikálnou analýzou a usporiada ich do syntaktického stromu, ktorý zobrazuje štruktúru programu.
- **Kontrola typu -** v tejto fáze sa analyzuje syntaktický strom. Zisťuje sa, či program neporušuje syntax jazyka. Napríklad, či bola použitá premenná deklarovaná a či je použitá v správnom kontexte.
- **Generovanie prechodného kódu -** genrátor preloží program do jednoduchého a strojovo nezávislého prechodného kódu. Výhodou je, že ak kompilátor potrebuje vygenerovať kód pre niekoľko strojových architektúr, je potrebný iba jeden preklad do prechodného kódu.
- **Pridelenie registra -** symbolické názvy premenných v prechodnom kóde sa preložia na čísla, z ktorých každé zodpovedá registru v cieľovom strojovom kóde.
- **Generovanie strojového kódu -** generátor preloží prechodný kód do strojového kódu pre špecifickú architektúru daného zariadenia.
- **Skladanie -** strojový kód je preložený do binárnej reprezentácie.

### 1.5.2 Interpreter

Interpreter [8] je ďalší spôsob implementácie programovacieho jazyka, ktorý zdieľa mnoho aspektov s kompilátorom. Lexikálna analýza a kontrola typov sú rovnaké ako pri kompilovaní, ale miesto generovania kódu zo syntaktického stromu sa pracuje priamo na vyhodnotení výrazov a vykonávaní príkazov.

Interpreter môže v niektorých prípadoch spracovať časť syntaktického stromu viac krát (napr. telo cyklu), preto je poväčšine interpretácia v porovnaní s kompiláciou pomalšia. Výhodou je, že programovanie interpretra je jednoduchšie ako programovanie kompilátora, preto sa v používa v aplikáciách, kde nie je podstatná rýchlosť častejšie.

Kompilátor a interpreter sa taktiež môžu pri vývoji aplikácií kombinovať. Kompiláor vytvorí prechodný kód a namiesto generovania strojového kódu sa prechodný kód interpretuje. Existujú aj programy, v ktorých sa niektoré časti kompilujú do strojového kódu, iné časti do prechodného kódu, ktorý sa následne interpretuje a zvyšné časti syntaktického stromu sa priamo interpretujú.

V našej aplikácii bude vzhľadom na požiadavky jazyka postačujúce spracovanie zdrojového kódu interpretrom.

# Kapitola 2

# Špecifikácia

Ako sme si ozrejmili v predchádzajucej kapitole, nevidiaci používateľia väčšinou nemajú možnosť využívať funkcionalitu aplikácií plnohodnotne. Pretože je naša aplikácia určená práve týmto ľudom, musíme ju ich požiadavkám prispôsobiť.
V tejto časti si prejdeme konkrétne body, na ktoré sa pri vývoji programu musíme sústrediť, aby sme nevidiacim zabezpečili bezproblémovú interakciu s našou aplikáciou.

## 2.1 Používateľské rozhranie

Aplikácie pre nevidiacich a slabozrakých by mali mať čo najjednoduchšie používateľské rozhranie, bez zbytočných prvkov, s ktorými môže mať čítač obrazovky problém. Celá funkcionalita musí byť ovládateľná klávesnicou. Používateľ musí byť schopný zorientovať sa v ňom a nájsť hladané prvky rovako jednoducho ako vidiaci človek.
Najpodstatnejším a najvyužívanejším prvkom v používateľskom rozhraní našej aplikácie bude editačné viacriadkové pole, do ktorého bude používateľ písať príkazy. Na spodu obrazovky bude menšie okno, v ktorom sa budú zobrazovať chyby v programe.
Na vrchu aplikácie bude navigácia so záložkami Súbor, Nastavenia a Pomocník.

- V záložke Súbor si vie používateľ načítať existujúci súbor, uložiť si aktuálne rozpísaný kód alebo vytvoriť nový súbor. Všetky tieto akcie sa budú dať vyvolať aj pomocou klávesových skratiek.
- V Nastaveniach bude možnosť zapnutia riadkovania, ktoré bude žiakom nápomocné hlavne pri debuggovaní. Taktiež bude možné nastaviť, či chceme aby nám aplikácia ponúkala kontextovú ponuku na dopĺňanie príkazov a možnosť nastavenia veľkosti písma.
- Záložka Pomocník poskytne používateľovi rýchly prehľad príkazov a zvukov, ktoré môže použiť.

## 2.2 Vstup programu

Vstupom programu bude zdrojový kód, teda text, ktorý žiak napísal do editačného viacriadkového poľa vo forme príkazov, premenných, čísel atď.
Vstupný kód bude možné uložiť do súboru. Z uloženého súboru môže žiak svoj kód načítať a pokračovať v jeho editovaní alebo ho znovu spustiť. Súbory budú uložené s príponou .txt.

## 2.3 Výstup programu

Výstupom programu bude postupnosť zvukových efektov a syntetizovaného textu, ktorú žiak naprogramoval pomocou základných a rozšírených príkazov.
Ak bola počas interpretácie v kóde nájdená syntaktická chyba, ako výstup programu sa v textovom okne zobrazí informácia o chybe a konkrétnom riadku, kde chyba nastala. Tento výstup bude taktiež zvukovo oznámený čítačom obrazovky.

## 2.4 Špecifikácia jazyka

Vzhľadom na to, že prostredie je určené pre nevidiacich žiakov, aj syntax jazyka im musíme prispôsobiť.
Príkazy budú v slovenskom jazyku, podobné prirodzenej reči a budú oddelené medzerou. Interpreter bude ignorovať prebytočné medzery. Telo príkazov nebude potrebné zapisovať do zátvoriek. V prípade príkazov, ktoré vo svojom tele obsahujú viacero príkazov, ako napríklad cyklus, bude telo ukončené príkazom koniec.

# Kapitola 3

# Návrh

## 3.1 Návrh používateľského rozhrania

## 3.2 Návrh jazyka

# Kapitola 4

# Implementácia

## 4.1 Štruktúra kódu

## 4.2 Analýza textu

## 4.3 Vykonávanie príkazov

## 4.4 Syntéza reči

# Kapitola 5

# Testovanie

## 5.1 Priebeh testovania

V tejto časti si popíšeme metodiku výskumu pri overovaní funkčnosti nášho programovacieho prostredia.

### 5.1.1 Popis vzorky respondentov

Pred tým, než našu aplikáciu predstávíme žiakom a tí ju prvý krát otestujú, je potrebné aby jej funkčnosť otestovali skúsenejší používatelia.
Funkčnosť aplikácie budú postupne vo viacerých fázach testovať rôzne typy používateľov. Potrebujeme, aby ju otestovli autori, žiaci aj učitelia, ktorí ju budú používať vo svojom vyučovacom procese. V nasledujúcej časti si presnejšie predstavíme v ktorých fázach ju rôzne typy používateľov odskúšajú.

### 5.1.2 Popis metodiky pri overovaní

Naša predstava o realizácii testov je nasledovná:

- V prvej fáze prebehne testovanie autorkou, školiteľkou a ideálne aj spolužiakmi alebo nezaiteresovanými ľudmi, ktorí posúdia prácu v aplikácii a intuitívnosť prostredia. Títo používatelia môžu objaviť základné nedostatky a vyjadriť svoje pripomienky k základnej funkcionalite ap-

likácie. Súčasťou prvej fázy búdú aj autorkou navrhnuté testy a ich výstupy.

- Ak autorka so školiteľkou usúdia, že aplikácia je dostatočne odladená a spĺňa navrhovanú funkcionalitu, posunieme ju učiteľom informatiky nevidiacich žiakov. Tí sa stretávajú s nevidiacimi žiakmi denne a tak najlepšie poznajú ich požiadavky a vedia, v ktorých oblastiach môže nastať problém a dôsledne danú oblasť otestovať. Od učiteľov očakávame pripomienky v oblasti funkčnosti aplikácie s čítačom obrazovky a korektnosťou výstupného zvukového reťazca.

- V poslednej fáze predstavíme programovacie prostredie tým, pre ktorých bolo vyvýjané, teda samotným nevidiacim žiakom. Predpokladáme stretnutie so stredoškolskými žiakmi, vrámci ich vyučovania. Vzhľadom na to, že máme v aplikácii zložitejšie konštrukcie, na jednom stretnutí by sme ju nestihli prejsť celú. Aby sme vedeli otestovať jej kompletnú funkčnosť by bolo ideálne absolvovať so žiakmi aspoň 5 vyučovacích hodín. Na týchto hodnách by sme postupne prešli sekvenciu príkazov, príkazy opakovania, vetvenie, podprogramy a premenné.

Po každej fáze sa zanalyzujú získané dáta a zapracujeme relevantné pripomienky.

## 5.2 Výsledky testovania

# Záver

# Literatúra


[1] Intellij idea overview. https://www.jetbrains.com/help/idea/discover-intellij-idea.html (visited: 2021-11-06).

[2] Java Access Bridge. https://www.oracle.com/java/technologies/javase/javase-tech-access-bridge.html (visited: 2021-11-09).

[3] Java Accessability API. https://docs.oracle.com/en/java/javase/11/access/java-accessibility-overview.html#GUID-17F9FD40-E191-41CE-BCF9-D956F1EF5111 (visited: 2021-11-09).

[4] NVDA. https://www.nvaccess.org/about-nv-access/ (visited: 2021-11-09).

[5] Institute for Disability Research, Policy, and Practice. Visual disabilities. https://webaim.org/articles/visual/blind (visited: 2021-10-31).

[6] Matúš Kováč. Programovacie prostredie prístupné pre nevidiacich žiakov nižieho sekundárneho vzdelávania, diplomová práca, FMFI UK, Bratislava 2018.

[7] JANA Lopúchová. Základy pedagogiky zrakovo postihnutých. *Kol. autorov: Základy špeciálnej pedagogiky pre prácu so študentmi stredných a vysokých škôl. Bratislava: Univerzita Komenského*, pages 73–91, 2007.

[8] Torben Egidius Mogensen. *Basics of Compiler Design*. 2010.

[9] Cecily Morrison, Nicolas Villar, Alex Hadwen-Bennett, Tim Regan, Daniel Cletheroe, Anja Thieme, and Sue Sentance. Physical programming

for blind and low vision children at scale. *Human–Computer Interaction*, 36(5-6):535–569, 2021.

[10] Venkatesh Potluri, Priyan Vaithilingam, Suresh Iyengar, Y Vidhya, Manohar Swaminathan, and Gopal Srinivasa. Codetalk: Improving programming environment accessibility for visually impaired developers. 2018.

[11] Jaime Sánchez and Fernando Aguayo. Apl: Audio programming language for blind learners.

[12] Techopedia. Refreshable braille display. `https://www.techopedia.com/definition/15186/refreshable-braille-display` (visited: 2021-10-31).

# Príloha : obsah elektronickej prílohy

V elektronickej prílohe priloženej k práci sa nachádza zdrojový kód programu a súbory s výsledkami experimentov. Zdrojový kód je zverejnený aj na stránke `http://mojadresa.com/`.