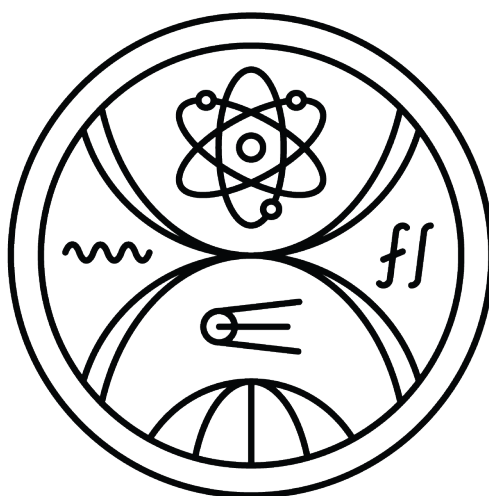


UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



EDUKAČNÉ PROSTREDIE NA PROGRAMOVANIE
HUDBY PRÍSTUPNÉ PRE NEVIDIACICH ŽIAKOV
SEKUNDÁRNEHO VZDELÁVANIA
DIPLOMOVÁ PRÁCA

2023
BC. JAKUB ŠVORC

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

EDUKAČNÉ PROSTREDIE NA PROGRAMOVANIE
HUDBY PRÍSTUPNÉ PRE NEVIDIACICH ŽIAKOV
SEKUNDÁRNEHO VZDELÁVANIA
DIPLOMOVÁ PRÁCA

Študijný program: Informatika
Študijný odbor: Aplikovaná informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: doc. RNDr. Ľudmila Jašková, PhD.

Bratislava, 2023
Bc. Jakub Švorc



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

- Meno a priezvisko študenta:** Bc. Jakub Švorc
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:** Edukačné prostredie na programovanie hudby prístupné pre nevidiacich žiakov sekundárneho vzdelávania
Educational environment for music programming accessible to secondary blind pupils
- Anotácia:** Autor vytvorí programovacie prostredie s vlastným kompilátorom alebo interpreterom. Základné príkazy zabudovaného programovacieho jazyka budú slúžiť na prehratie tónov zvoleným hudobným nástrojom. Okrem toho bude možné použiť aj komplikovanejšie štruktúry, ako je cyklus, príkaz vetvenia, podprogram, vlákno.
Editor kódu bude mať zabudovanú kontrolu syntaxe a funkciu prediktívnej ponuky príkazov.
Prostredie bude prístupné pre čítač obrazovky a bude plne ovládateľné pomocou klávesnice. Nevidiacim používateľom umožní okrem bežnej práce s textom aj jednoduchým spôsobom získať prehľad o štruktúre vytvoreného kódu.
Použitelnosť výslednej aplikácie pre cieľového používateľa bude zabezpečená vďaka výskumu vývojom (design based research), t.j. iteratívnym vývojom a overovaním s rôznymi typmi používateľov (nevidiaci programátor, učiteľ nevidiacich žiakov, nevidiaci žiak).
- Cieľ:** Vytvoriť programovacie prostredie umožňujúce programovať hudbu pozostávajúcu z viacerých paralelne znejúcich melódií. Dôraz bude kladený na zabezpečenie plnej prístupnosti a efektívnej práce s editorom kódu pre žiakov so zrakovým postihnutím.
- Literatúra:** S. Aaron, Code music with Sonic Pi, Retrieved from https://www.raspberrypi.org/magpi-issues/Essentials_Sonic_Pi-v1.pdf
C. C. De Oliveira, Designing educational programming tools for the blind: mitigating the inequality of coding in schools, 2017.
HADWEN-BENNETT, A. et al. Making Programming Accessible to Learners with Visual Impairments: A Literature Review, International Journal of Computer Science Education in Schools, April 2018, Vol. 2, No. 2, ISSN 2513-8359.
- Vedúci:** doc. RNDr. Ľudmila Jašková, PhD.
Katedra: FMFI.KDMFI - Katedra didaktiky matematiky, fyziky a informatiky
Vedúci katedry: prof. RNDr. Ivan Kalaš, PhD.

Pod'akovanie: Rád by som pod'akoval mojej školiteľke doc. RNDr. Ľudmile Jaškovej, PhD., za vedenie práce, cenné rady, trpezlivosť a motiváciu pri tvorbe Diplomovej práce.

Abstrakt

Táto diplomová práca opisuje vývoj programovacieho prostredia s vlastným kompilátorom a jazykom. Aplikácia je určená pre žiakov 2. stupňa základnej školy so zrakovým znevýhodnením. Základné príkazy zabudovaného programovacieho jazyka slúžia na prehratie tónov zvoleným hudobným nástrojom. Je možné použiť aj komplikovanejšie štruktúry, ako je cyklus, príkaz vetvenia, podprogram, vlákno. Editor kódu bude mať zabudovanú kontrolu syntaxe a funkciu prediktívne ponuky príkazov. Prostredie bude prístupné pre čítač obrazovky a bude plne ovládateľné pomocou klávesnice. Nevidiacim používateľom umožní okrem bežnej práce s textom aj jednoduchým spôsobom získať prehľad o štruktúre vytvoreného kódu. Použitelnosť výslednej aplikácie pre cieľového používateľa bude zabezpečená vďaka výskumu vývojom (design based research), t.j. iteratívnym vývojom a overovaním s rôznymi typmi používateľov (nevidiaci programátor, učiteľ nevidiacich žiakov, nevidiaci žiak).

Kľúčové slová: jedno, druhé, tretie

Abstract

This master's thesis describes the development of a programming environment with its own compiler and language. The application is intended for students of the 2nd level of primary school with visual impairments.

The basic commands of the built-in programming language are used to play tones with a chosen musical instrument. It is also possible to use more complex structures, such as loops, conditional statements, subprograms, and threads.

The code editor will have built-in syntax checking and a predictive command suggestion feature. The environment will be accessible for screen readers and fully controllable via the keyboard.

For visually impaired users, it will allow not only regular text work but also provide a simple way to understand the structure of the created code. The usability of the resulting application for the target user will be ensured through design-based research, i.e., iterative development and testing with various types of users (blind programmers, teachers of blind students, blind students).

Keywords:

Obsah

Úvod	1
0.1 Section	2
0.1.1 Subsection	2
0.2 Section	2
0.3 Section	2
1 Prehľad problematiky	3
1.1 Typy zrkového postihnutia a ich kategorizácia	3
1.1.1 Kategórie zrkového postihnutia	3
1.2 Analýza programov pre nevidiacich študentov	4
1.2.1 Quorum	4
1.2.2 Sonic Pi	5
1.2.3 MusicBlocks	6
1.2.4 TonIK2	7
1.2.5 Porovnanie riešení	8
1.3 Čítače obrazovky	9
1.3.1 NVDA - NonVisual Desktop Access	9
1.3.2 JAWS (Job Access With Speech)	9
1.4 Popis použitých technológií	10
1.4.1 Jazyk C#	10
1.4.2 .NET	10
1.5 Vývoj softvéru pre nevidiacich študentov	11
1.5.1 Požiadavky na softvér pre nevidiacich	11
1.5.2 Porovnanie a zhodnotenie jazykov	13
Záver	19
Príloha A	21
Príloha B	22

Zoznam obrázkov

1.1	SonicPi testovanie (pred/po): programovanie dojmy	5
1.2	SonicPi testovanie (pred/po): relevantnosť schopnosti programovať	6
1.3	TonIK2	7
1.4	Braillov displej	10
1.5	Priradenie hodnoty v Jave	13
1.6	Kód v programe TONIK 2	14
1.7	Zápis cyklu v jazyku Java	14
1.8	Zápis cyklu v jazyku TonIK2	15
1.9	Vetvenie programu v Jave	16
1.10	Vetvenie programu v TonIK2	16
1.11	Definícia a volanie podprogramu v Jave	17
1.12	Definícia a volanie podprogramu v Jave	17

Zoznam tabuliek

1.1	Aplikácie - zhrnutie	8
-----	--------------------------------	---

Úvod

Citácie:

[3] [8] [9]

Kuki

0.1 Section

Text...

0.1.1 Subsection

Text...

Subsubsection

Text...

Paragraph Text...

Subparagraph Text...

0.2 Section

Text...

0.3 Section

Text...

Definícia Zložením relácií $R \subseteq A \times B$ a $S \subseteq B \times C$ rozumieme binárnu reláciu $R \circ S \subseteq A \times C$ definovanú predpisom

$$R \circ S = \{(x, z) \in A \times C \mid \exists y \in B ((x, y) \in R \wedge (y, z) \in S)\}.$$

Kapitola 1

Prehľad problematiky

V tejto kapitole sa zoznámime s technológiami určenými pre zrakovo postihnutých používateľov. Porovnáme ich technické prevedenia, zhodnotíme ich použiteľnosť pre nevidiacich žiakov a popíšeme si znalosti a zručnosti, ktoré chceme rozvíjať v rámci vzdelávacieho procesu. Preskúmame možnosti týchto softvérov, ich širšej aplikácie a návaznosti na ďalšie učivo v oblasti informatiky. Zadefinujeme, čo potrebuje vedieť žiak základnej školy, aby čo najjednoduchšie a najlepšie pochopil základné programovacie konštrukcie a pojmy. Následne si predstavíme podobné existujúce implementácie.

1.1 Typy zrakového postihnutia a ich kategorizácia

Zrakové postihnutie sa prejavuje rôzne na základe čoho, ho vieme kategorizovať do niekoľkých typov [6]. V tejto podkapitole si ich stručne popíšeme, aby sme vedeli navrhnúť prístupný softvér pre každého. Pokúsime sa lepšie porozumieť potrebám jednotlivcov so zrakovým postihnutím a implementovať výsledky do nášho prostredia.

1.1.1 Kategórie zrakového postihnutia

Nevidiaci

Nevidiaci s úplnou stratou zraku sú závislí na alternatívnych spôsoboch získavania informácií, ako sú hlasové technológie, Braillovo písmo a hmatové pomôcky.

Čiastočne vidiaci a slabozrakí

wČiastočne vidiaci neprišli úplne o zrak, majú problémy najmä v priestorovej orientácii. Slabozrakí, podobne ako čiastočne vidiaci, majú ťažkosti s ostrosťou videnia, periférneho videnia, rýchlosti a presnosti zraku. Obe skupiny používateľov môžu do istej miery reagovať na vizuálne podnety. Dôležité je dbať na možnosti prispôsobenia ako sú farebné kontrasty, nastavenia veľkosti znakov, objektov a klávesové skratky.

Osoby s poruchami binokulárneho videnia

Poruchy binokulárneho videnia sú problémy so schopnosťou očí spolupracovať pri vnímaní priestoru a jeho hĺbky. Ide o široké spektrum zrakového postihnutia, teda tieto poruchy môžu mať rôzne príčiny a môžu sa prejaviť rôznymi spôsobmi. Môžu mať aj vplyv na spoluprácu očí pri sledovaní objektov.

1.2 Analýza programov pre nevidiacich študentov

V súčasnosti už existuje širšia ponuka programovacích prostredí, jazykov a nástrojov, ktoré sú prispôbolené potrebám pre nevidiacich programátorov. V tejto podkapitole si urobíme prehľad súčasného stavu riešenej problematiky doma aj v zahraničí. Niekoľko z nižšie menovaných bolo vyvíjaných priamo na našej fakulte.

1.2.1 Quorum

Interpretovaný programovací jazyk Quorum prišiel spolu s integrovaným vývojovým prostredím (Integrated Development Environment - IDE) Quorum Studio v roku 2016. Jedná sa o programovací jazyk na tvorbu softvéru aj pre používateľov s rôznym fyzickým postihnutím. Na jeho tvorbe, finančnej podpore a údržbe sa podieľal niekoľko škôl a univerzít v USA.

Použitelnosť jazyka Quorum

Quorum je navrhnutý tak, aby čo najviac zjednodušil syntax a zároveň podporil plnú funkcionálnosť štandardných programovacích jazykov. Konkrétne poskytuje plnú funkcionálnosť a silu jazyka Java, nad ktorou bol postavený interpreter jazyka. Kompatibilita s JVM poskytuje rýchlosť, platformovú nezávislosť, pričom zjednodušuje syntax tak, aby dodržiavala štandardy, ktoré sú dôležité pre zrakovo postihnutých používateľov a spoluprácu s čítačom obrazovky - vynecháva zložité znaky, nie je potrebné dodržiavanie odsadenia, podpora debugovania, ovládateľnosť klávesovými skratkami.

Porovnanie s Javou

S Javou sa spájajú mnohé nepríjemnosti, ktoré sa týkajú jej syntaxe - dlhé názvy tried, ťažkopádna práca s textovými reťazcami (chýbajúca podpora formátovania stringov alebo iná manipulácia). Ďalej zložité názvy výnimiek, zložitá práca s objektovými premennými - na ich prácu je potrebné používať gettery a settery namiesto properties, množstvo vygenerovaného kódu (neobsahuje top-level statements ako C#, C++, C) a iné.

Použitelnosť nástroja Quorum je široká, nie je špecificky určený pre používateľov so zrakovým postihnutím. Kvôli tomu obchádza niektoré odporúčania a štandardy pre vývoj

softvéru pre zrakovo postihnutých. Pri použití starších verzií je potrebné mať už nainštalovanú Javu, ak ju inštalačný balíček nemá v sebe obsiahnutú.

1.2.2 Sonic Pi

Open-source aplikácia Sonic Pi je interaktívne programovacie prostredie s cieľom využitia na hudobnú tvorbu. Jeho jednoduchý jazyk a zameranie na vytváranie skladieb prostredníctvom programovania ho robia prístupným a pútavým pre študentov, ktorí majú záujem o hudobné aplikácie. Avšak práve jeho špecifickosť využitia môže byť obmedzením z hľadiska všeobecných vzdelávacích cieľov.

V článku [7] o testovaní SonicPi sledovali, ako študenti budú hodnotiť softvér na škálach zábava, relevantnosť a frustrácia v rôznych fázach tohto testovania (obr. 1.2). Testovacia skupina mala zo začiatku problém zorientovať sa a niektorí študenti nemali toľko skúseností z oblasti informatiky. Napriek tomu po tomto testovaní softvér a samotný proces programovania hodnotili pozitívne (obr. 1.1) aj napriek ťažkému začiatku (niekedy frustrujúcemu).

	T1	T2
Jacob	"boring"	"coding is really fun and interesting, I liked working with my friend on cool sounds"
Charlotte	"not sure sorry"	"I feel quite good, it was quite fun thanks!"
William	"OK"	"coding can be really cool, I feel really good"

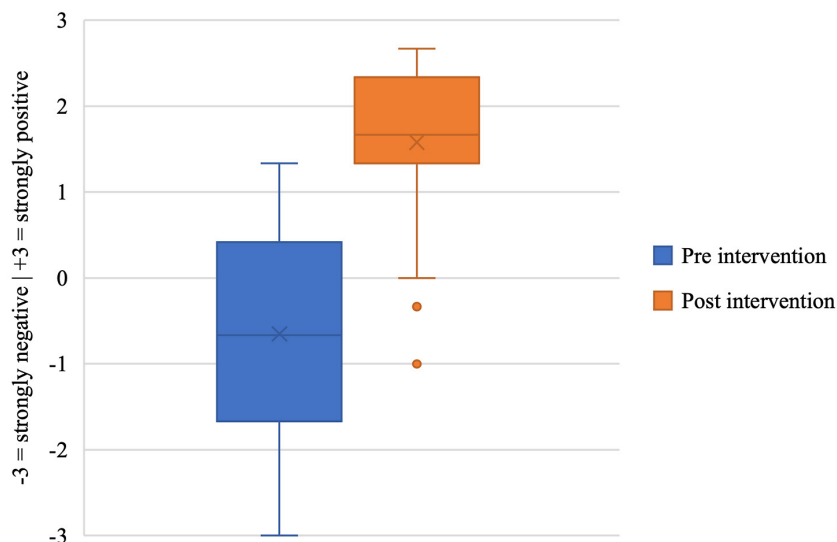
Obr. 1.1: SonicPi testovanie (pred/po): programovanie dojmy

Využitie Sonic Pi

Okrem širokej preddefinovanej sady zvukov umožňuje vytvárať svoje vlastné zvuky a rytmy. Používa sa pri vyučovaní ale aj profesionálne na tvorbu hudby rôznych zvukových ukážok do filmov, videohier alebo iných digitálnych diel.

Výhody Sonic Pi

Zložitosť jazyka je prispôsobená širokej verejnosti a zvládli by ho aj žiaci základnej školy. Celkovo ide o pútavé interaktívne prostredie, kde výsledkom programovania je práve hudba.



Obr. 1.2: SonicPi testovanie (pred/po): relevantnosť schopnosti programovať

Nevýhody Sonic Pi

Nie je kompatibilné s čítačmi obrazovky, nie je plne ovládateľné klávesnicou a zahlcujúcim používateľským rozhraním. Jeho špecializovaný charakter môže byť obmedzením pre cieľovú skupinu užívateľov. Na základe toho, ho môžeme označiť za menej vhodný na vyučovacie účely.

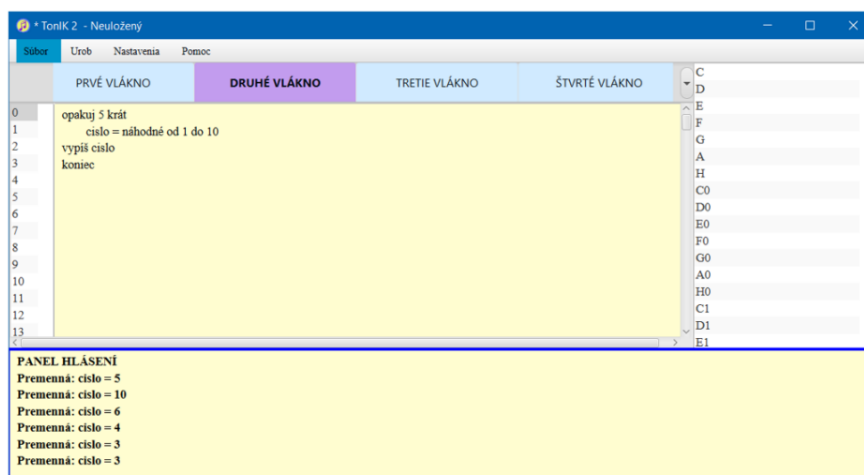
Napriek tomu, že jeho syntax je intuitívna a čitateľná pre bežných používateľov, obsahuje špeciálne znaky a bloky kódu sú v ňom automaticky formátované odsadením (ako v Pythone), čo by aj s použitím čítača obrazovky bolo náročné na orientáciu v kóde.

1.2.3 MusicBlocks

Music Blocks je navrhnutý pre učiteľov a študentov, aby mohli preskúmať základné koncepty hudby v prostredí vizuálneho programovania.

Music Blocks je inovatívny a prospešný pre hudobné vzdelávanie z niekoľkých dôvodov: na jednej strane predstavuje novú metódu porozumenia základným konceptom hudby, na druhej strane je nástrojom na učenie programovania a logických zručností. Integruje hudobné a STEM (veda, technika, inžinierstvo, matematika) základy spôsobom, ktorý je zábavný, škálovateľný a autentický. [3].

Music Blocks bol vyvíjaný na základe Turtle Blocks, ktoré bolo inšpirované Logom na kreslenie umelecky efektných obrázkov prostredníctvom programovania. Obe aplikácie sú určené pre internetový prehliadač. Music Blocks rozširuje možnosti Turtle Blocks tým, že obsahuje sadu audio funkcií súvisiacich s výškou tónu a rytmom.



Obr. 1.3: TonIK2

1.2.4 TonIK2

Aplikácia, ktorá vznikla ako diplomová práca [6]. Umožňuje programovať prostredníctvom hudby a zvukov, ktoré je možné modifikovať a ovládať pomocou príkazov (na obrázku 1.3).

Použitie TonIK2

Aplikácia umožňuje hudobné programovanie pre špecifickú skupinu používateľov, pre zrakovo postihnutých študentov základných škôl. Pomocou zvukov a hudobných prvkov sa snaží priblížiť a vysvetliť niektoré základné programovacie koncepty - cykly, vetvenia, používanie premenných a podprogramov, použitie vlákien, pre paralelný beh častí programu.

Tým, že je špecificky zameraná na skupinu zrakovo postihnutých žiakov, je dôležité, že dodržiava odporúčania a štandardy pre vývoj softvéru pre nevidiacich.

V nástroji sa programuje pomocou vlastného interpretovaného jazyka. Interpreter je napísaný v Jave a jazyk sa skladá z nasledovných typov príkazov:

- zvukové príkazy Tón, Postupnosť, Odzadu, Stupnica, Akord, Náhodný - tieto príkazy priamo prehrávajú zvuk,
- príkaz Zmeň nástroj - nastaví nástroj, ktorý ďalej prehrá zvukové príkazy,
- cyklus s pevným počtom opakovaní – umožňuje vytvárať jednoduché cykly so zadávaným počtom opakovaní,
- príkazy vetvenia - na základe vyhodnotenia logickej podmienky vykonajú jednu alebo druhú vetvu programu.

1.2.5 Porovnanie riešení

Cieľom našej práce je výskumnými metódami za pomoci testovania implementovať čo najlepšie riešenia pre náš edukačný softvér. Jeho prvý testovací návrh vytvoríme podľa preštudovaných vedeckých článkov a v tejto podkapitole spomínaných existujúcich riešení 1.1.

Z nasledujúcich výsledkov pozorovania sa pokúsime vyvodit', aké znaky by malo mať naše prostredie.

Aplikácia	Jazyk	Platforma	Tematika	Komplexnosť
Quorum	Založený na jazyku Java	Multiplatformový	Všeobecná	Mierne pokročilí
Sonic Pi	Ruby	Multiplatformový	Hudobná	Začiatočníci, pokročilí
MusicBlocks	Vizuálny programovací jazyk	Webová aplikácia	Hudobná	Začiatočníci
TonIK2	Inšpirovaný jazykom Logo	Multiplatformový	Hudobná	Začiatočníci

Tabuľka 1.1: Aplikácie - zhrnutie

Všetky aplikácie sú navrhnuté pre edukačné účely. Rozvíjajú predstavivosť a oboznamujú žiakov a študentov s princípmi programovania zaujímavou formou. Quorum je najbližší k rozšíreným programovacím jazykom ako sú Java, C#, Python, ale je obohatený o audio funkcie a prvky.

Všetky jazyky sú navrhnuté tak, aby obsahovali čo najmenej nealfanumerický znakov, ako sú zátvorky, bodkočiarky a podobne. Pracujú perfektne s čítačmi obrazovky. Prostredia sú jednoduché a intuitívne. Naplno využívajú variabilitu svojich nástrojov, a tak aj s menšou ponukou môže používateľ riešiť komplexné úlohy. Ak aplikácie obsahujú grafické prvky, tak sú veľké, kontrastné. Celé menu je obsluhovateľné klávesovými skratkami a do jednotlivých okien sa môže používateľ dostať príslušnou klávesou alebo kombináciou kláves.

Žiaci majú k dispozícii:

- premenné
- cykly
- vetvenia - podmienky
- jednoduché podprogramy
- spätnú väzbu v podobe audio výstupu (napr. prečítajú sa kontrolné výpisy)
- terminál - kontrola kódu a hlásenie chýb

1.3 Čítače obrazovky

Spomenuli sme programy, ktoré boli vyvíjané s ohľadom na špecifické potreby programátorov so zrakovým postihnutím. V tejto časti predstavíme nástroje, ktoré úzko súvisia s vyššie uvedenými aplikáciami, či už sú na nich závislé alebo sú v nejakej forme ich súčasťou.

Čítače obrazovky sú preto kľúčovým nástrojom v digitálnom svete. Ich vývoj sa v priebehu času výrazne zdokonalil vzhľadom na narastajúce potreby používateľov a nové technológie.

Používatelia sa nevyhnutne stretnú aj s nedostatkami čítačov, najmä pri komplexných webových stránkach či ne-optimalizovaných aplikáciách. Niektoré grafické prvky môže byť obtiažne interpretovať, čím používateľ prichádza o informácie. Navyše sú limitovaní skutočnosťou, že nie všetky aplikácie (či už webové alebo desktopové) dodržiavajú normy pre spoluprácu s čítačmi obrazovky.

1.3.1 NVDA - NonVisual Desktop Access

Je to otvorený a bezplatný softvér [2]. Poskytuje plnú funkcionálnu podporu na viacerých platformách. Pravidelne aktualizovaný so živou komunitou používateľov a otvoreným zdrojovým kódom. NVDA podporuje až 55 jazykov vrátane slovenského. Vývojármi tohto softvéru sú dvaja nevidiaci programátori, ktorí ťažiac z vlastných skúseností, dokázali navrhnuť softvér, ktorý sa následne začal používať vo viac ako 175 krajinách. Nie je adresovaný len programátorom, ale širokej verejnosti a pomáha používateľom s každodennými úlohami ako je pohyb na internete, písanie dokumentov, financie, online komunikácia a mnoho ďalších. Je plne kompatibilný s aplikáciami Word, Excel, PowerPoint, Google Chrome, Mozilla, Microsoft Edge, WordPad, NotePad, a keďže ho preferujú aj žiaci, tak aj s naším prostredím.

1.3.2 JAWS (Job Access With Speech)

Patrí medzi najpopulárnejšie čítače obrazovky na svete. Poskytuje hlasový a Braillový výstup pre počítačové aplikácie. Obsahuje [1] :

- ovládače pre populárne Braillové displeje
- funkciu OCR (optické rozoznávanie znakov) pre obrazové súbory alebo neprístupné PDF dokumenty
- službu Picture Smart na rozpoznávanie obrazov
- hlasového asistenta pre často používané príkazy

Pracuje s Microsoft Office, Google Docs, Chrome, Edge, Firefox a mnohými ďalšími. Podporuje Windows® 11, Windows 10, Windows Server® 2019 a Windows Server 2016.



Obr. 1.4: Braillov displej

Braillov displej

Braillov displej je zariadenie na obrázku 1.4, ktoré môže užívateľ s čítačom obrazovky synchronizovať s počítačom, tabletom (napríklad iPadom) alebo telefónom. Displej zobrazuje hmatateľné braillové znaky textu z obrazovky. Vďaka tomu ho môžu používať aj nepočujúci, nie len nevidiaci ľudia. Tieto zariadenia, na rozdiel od čítačov obrazovky, nie sú tak finančne dostupné. [4]

1.4 Popis použitých technológií

Pri vývoji programovacieho prostredia sme sa rozhodli využiť technológie, ktoré nám zabezpečia čo najlepšie riešenia pre prístupný softvér pre nevidiacich programátorov.

1.4.1 Jazyk C#

C# (C Sharp) je objektovo orientovaný programovací jazyk od spoločnosti Microsoft. Vhodný pre rôznorodé aplikácie a kompatibilný s .NET, vďaka čomu môžeme pristupovať ku množstvu knižníc. Medzi jeho výhody patrí bezpečnosť s automatickým čistením pamäti, objektovú orientáciu pre modularitu kódu a čitateľnosť. Je obľúbený pre vývoj desktopových, webových, mobilných aplikácií a hier, vďaka čomu je široko využívaným jazykom.

1.4.2 .NET

.NET je open-source platforma od Microsoftu. Ide o univerzálny nástroj na vývoj aplikácií, ktorý je kombinovateľný s viacerými programovacími jazykmi ako sú C# alebo C++/CLI. Okrem Windows, Linux a macOS ním môžeme vyvíjať aplikácie aj pre Android a iOS.

1.5 Vývoj softvéru pre nevidiacich študentov

rozne úrovne postihnutia -> poskytnut rozne nastavenia pre jednotlivé špecifické požiadavky. v tejto podkapitole zhrnieme ake sú odporúčania.

Ako sme už spomenuli v sekcii 1.1.1, rôzne postihnutia majú rôzne požiadavky na funkčnosť.

Spoločným problémom je, že väčšina jazykov sa snaží byť na všeobecné použitie a sú zamerané na čísla a symboly. Preto je potrebné dodržiavať praktiky a odporúčania, ktoré tento problém riešia.

1.5.1 Požiadavky na softvér pre nevidiacich

V tejto časti si popíšeme odporúčania [8, 5] na tvorbu jazyka, prostredia a softvéru, ktorý je určený pre používateľov so zrakovým postihnutím a porovnáme si požiadavky pre rôzne úrovne postihnutia.

- Nevidiaci

Pretože nevidiaci sú závislí od čítačov obrazovky, nie je potrebné aby bol softvér vizuálne príťažlivý. Musíme však dbať na to, aby bol softvér ovládateľný klávesnicou a klávesovými skratkami.

Spolu s tým je dôležité aby boli prístupné ovládacie prvky (tlačidlá, menu, ponuky, okná a pod.), ktoré sú pre používateľa dôležité, aby sme tak zabránili neočakávanej manipulácii s elementami.

Okrem dostupnosti ovládacích prvkov je potrebné aj ich správne, výstižné a konkrétne pomenovanie. Napríklad popis prvku "my textbox" používateľovi nepovie čo sa do daného textového poľa píše. Ak ho popíšeme "textové pole s kódom" je jasnejšie načo je určené.

Taktiež treba brať do úvahy škálu nastavení a prispôbení, ktoré pre používateľov sprístupníme. Napríklad nemá zmysel pre nevidiaceho používateľa poskytovať nastavenie farebnej schémy aplikácie alebo veľkosť písma.

Pretože syntax jazyka bude čítaná čítačom obrazovky, je potrebné minimalizovať špeciálne symboly (bodky, zložené zátvorky a pod.), nestavať bloky kódu pythonovou syntaxou (blok kódu je tvorený odsadením). Možnosť používať skrátené slová namiesto celých klúčových slov jazyka (napr. "repeat" skrátit' na "rpt"), ušetrí používateľom čas najmä v prípade, kedy sa opakovane číta celý text kódu.

- Čiastočne vidiaci a slabozrakí

Slabozraký oproti nevidiacemu má čiastočný pohľad na našu aplikáciu. S tým súvisí aj jej prispôbitelnosť.

Slabozrakí sú schopní čítať sami pomocou zväčšovacích nástrojov. Je preto potrebné v softvéroch dodržiavať kontrast písma od pozadia.

Pri niektorých úrovniach postihnutia s čiastočným videním nemusí byť ani potrebné používať zväčšovacie nástroje, stačí keď je možné zväčšovať písmo textu.

Okrem veľkosti, je dôležitý aj samotný font. Je dôležité myslieť na to, že v niektorých fontoch sú niektoré znaky ľahko zameniteľné. Najčastejším príkladom je písmeno "l" číslo "1". Odporúča sa používať bezpätkový font, ktorý neobsahuje vizuálne dekorácie a je čitateľnejší.

Je dobré zvážiť aj rôzne farebné schémy aplikácie, ktoré prispôbujú kontrast písma voči pozadiu a zameriavajú sa na problém s farbosleposťou. V spojení s farbami je pri dôležitých súčiastiach aplikácie namiesto, keď kritické akcie označíme rôznymi farbami, ktoré evokujú o akú akciu sa jedná (napríklad zelené tlačidlo "potvrď", červené tlačidlo "zruš").

Odporúča sa vyhýbať dekoráciám v aplikácií. Vyskakovacie polo-priesvitné okná nie sú vhodné, nakoľko je ťažké ich vidieť keď je obrazovka zväčšená alebo zameraná na časť, kde sa toto okno nenachádza.

Animované alebo blikajúce prvky môžu pôsobiť príjemne a pútavo, avšak pri ich nadmernom použití môžu byť zahlcujúce a rušivé, zvlášť pre slabozrakého používateľa, ktorému môžu odvádzať pozornosť.

- Osoby s poruchami binokulárneho videnia

Používatelia s touto poruchou sú schopní plného vnímania vizuálnej stránky softvéru. Vnímajú farby, prvky používateľského rozhrania (okná, políčka, tlačidlá a podobne). Na základe rozsahu postihnutia môžu byť schopní sledovať softvér aj bez zväčšovacích pomôcok a softvéru.

Pretože používateľ je schopný sledovať aplikáciu, je dôležité dodržiavať nie len zásady spomínané vyššie pre čiastočne vidiacich, ale aj zvážiť možnosť prispôbenia grafického používateľského rozhrania. Podobne ako bolo spomínané vyššie, je dôležité dodržiavať kontrast textu a dôležitých výstupov voči pozadiu, prípadne ponúknuť možnosť prispôbiť si farebné schémy na jednotlivé elementy.

Pretože používateľ môže pracovať aj s väčšou zobrazovacou jednotkou (monitor, televízia a iné), je veľmi nápomocné, keď si používateľ môže prispôbiť grafické rozhranie podrobnejšie, ako napríklad preusporiadaním prvkov, zmenou ich veľkosti a rozlíšenia a nezávislé nastavenie každého grafického komponentu.

1.5.2 Porovnanie a zhodnotenie jazykov

Viacero zo spomínaných nástrojov je vytvorených technológiou Java. Na začiatok si preto porovnáme jazyk Java s jazykom z nástroja TonIK2.

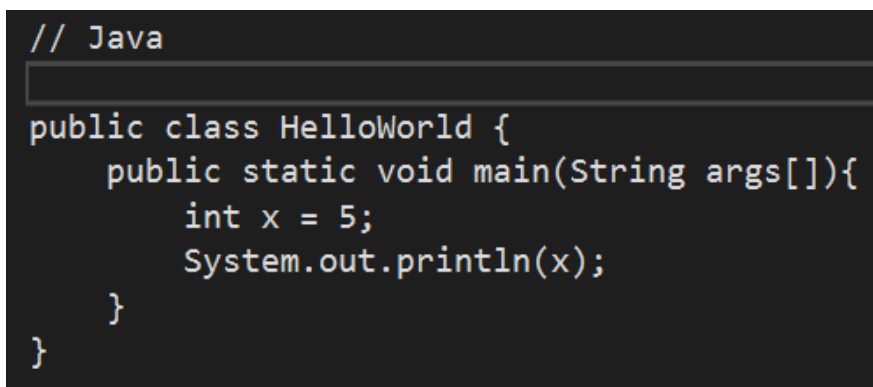
Pre porovnanie si ukážeme a porovnáme základné príkazy ako priradenie (vytvorenie premennej), cyklus, vetvenie a podprogram.

Príkaz priradenia

Premenná slúži na ukladanie hodnôt. Môžu to byť výsledky výpočtov, vstup od používateľa, konštanty, nami zadané hodnoty alebo objekty.

V niektorých jazykoch môže byť potrebné premenné najprv zadeklarovať, teda akýmsi spôsobom oznámiť, že tej-ktorej premennej chceme meno neskôr používať (prípadne jej vyhradiť miesto v pamäti). Tým v nej však nevznikne hodnota. Tú musíme neskôr priradiť.

Iné jazyky umožňujú vytvoriť premennú kedykoľvek počas behu programu. Tá sa teda deklaruje automaticky počas priradenia.



```
// Java

public class HelloWorld {
    public static void main(String args[]){
        int x = 5;
        System.out.println(x);
    }
}
```

Obr. 1.5: Priradenie hodnoty v Jave

Ako môžeme vidieť na obrázku 1.5 je potrebné napísať väčšie množstvo kódu ako v iných jazykoch (tzv. boiler-plate code). Taktiež obsahuje viacero špeciálnych znakov (zložené a hranaté zátvorky) a veľa bodkovej notácie.

```
// Tonik 2  
  
x = 5  
vypíš x
```

Obr. 1.6: Kód v programe TONIK 2

V tomto jazyku si môžeme všimnúť, že premenná vznikne v momente, keď jej priradíme hodnotu. Podobne ako v jazyku Python. Výhodou je, že nemusíme dopredu premennú deklarovať, ako napríklad v spomínanej Jave, ale vieme ju vytvoriť v momente jej potreby.

Cyklus

Cyklus je blok kódu, ktorý sa opakovane vykonáva. Koľko krát sa tento blok bude vykonávať určite iteračná premenná (často sa nazýva *i*).

Na základe vyhodnotenia logickej podmienky sa telo cyklu vykonáva, alebo sa jeho vykonávanie preskočí (prípadne ukončí).

Cyklus má viacero podôb - s pevným počtom opakovaní, s podmienkou na začiatku a s podmienkou na konci.

Cyklus s pevným počtom opakovaní je konceptuálne rovnaký ako cyklus s podmienkou na začiatku - kontroluje sa stav premennej voči nejakej hodnote. Cyklus s podmienkou však môže okrem číselných porovnaní mať aj zložitejšie podmienky alebo volania podprogramov na mieste, kde sa kontrolujú podmienky vykonania cyklu.

```
// Java  
  
public class HelloWorld {  
    public static void main(String args[]){  
        int x = 5;  
        for (int i = 0; i < x; i++){  
            // telo cyklu  
        }  
    }  
}
```

Obr. 1.7: Zápis cyklu v jazyku Java

Na zápis cyklického vykonávania príkazov potrebujeme okrem už spomínaného boilerplatekódu deklarovať počítaciu premennú, ktorá drží informáciu o počte, koľko krát sa má cyklus vykonať. K tomu potrebujeme deklarovať ďalšiu premennú a ďalšie špeciálne znaky (zátvorky a bodkočiarky).

```
// Tonik 2
[
x = 5
opakuj x krát
// telo cyklu
koniec vypíš x
```

Obr. 1.8: Zápis cyklu v jazyku TonIK2

V jazyku TonIK2 špeciálne znaky opäť vynechávame a označujeme iba koniec bloku, ktorý predstavuje telo cyklu.

Syntax jazyka je podobná, ako veta v slovenskom jazyku, takže namiesto uvažovania o tom, ako žiak napíše cyklus mu stačí si to približne zapamätať v slovenčine.

Vetvenie programu

Vetvenie programov je koncept, ktorý vytvára v programe vetvy (možnosti), ktorými sa vykonávanie programu môže vydať. Je riadené podmienkami, ktoré môžu byť vyhodnotené ako pravda alebo nepravda (používajú sa aj termíny ako splnená alebo nesplnená podmienka, 0 alebo 1, true alebo false).


```
// Java

public class HelloWorld {
    public static void main(String args[]){
        int x = 5;

        if (x == 5) {
            // príkazy ak je podmienka splnená
        }

        else {
            // príkazy ak nie je podmienka splnená
        }
    }
}
```

Obr. 1.9: Vetvenie programu v Java

Vetvenie v Java okrem štandardných aritmetických a logických porovnaní môže vyvolávať podprogramy alebo obsahovať zložitejšie logické výroky (logickú konjunkciu, logickú disjunkciu, logickú negáciu).

Nie je však možné viazať tieto operácie za sebou bez použitia logických AND alebo OR, teda nemôžeme napísať podmienku ako v napríklad v Pythone " $0 < x < 5$ ";

```
// Tonik 2

x = 5
Ak x == 5
// príkazy keď podmienka je splnená
Inak
// príkazy keď podmienka nie je splnená
Koniec
```

Obr. 1.10: Vetvenie programu v TonIK2

V jazyku TonIK2 je možné reťaziť logické porovnania, nie je však možné v kontexte vetvenia vykonávať logické AND ani OR, volať podprogramy, ktoré môžu robiť kontrolu alebo výpočty a na základe výsledku vyhodnotiť podmienku.

Podprogramy

Podprogramy sú kusy kódu, ktoré vieme zavolať. Ich účelom je zjednodíť často písaný kód alebo spoločnú funkcionálnu programovú časť do jednoduchého volania. Tým sa zlepšuje čitateľnosť.

ľonost' a zjednoduší vývoj aplikácií.

Volanie podprogramov môže vracať výsledok, ktorý môžeme neskôr použiť'. Existujú však aj podprogramy, ktoré nevracajú žiadnu hodnotu. Taktiež do podprogramov môžeme predávať parametre, pomocou ktorých sa môže vypočítať výsledok alebo nemusíme predávať žiadne parametre.

```
// Java

public class HelloWorld
{
    public void podprogram1()
    {
        // telo podprogramu
    }

    public static void main(String args[])
    {
        podprogram1();
    }
}
```

Obr. 1.11: Definícia a volanie podprogramu v Java

```
// Tonik 2

urob podprogram1
    // telo podprogramu
koniec

podprogram1
```

Obr. 1.12: Definícia a volanie podprogramu v Java

Zhrnutie

Jazyky z "céčkovej"rodiny sú dobré na orientáciu a čítanie pre bežných používateľ'ov. Vieme jasne odlíšiť hranice bloku kódu, vieme rýchlo získať potrebné informácie z pohľ'adu na signatúry funkcií, určovať a čítať program-flow (alternatívy vetvenia) a celkovo vieme mať dobrý prehľ'ad o programe.

Toto však neplatí pre nevidiacich alebo slabozrakých používateľov. Tieto jazyky obsahujú veľa špeciálnych znakov. Sú určené na všeobecné účely, teda nie sú špecificky navrhnuté tak, aby spĺňali štandardy jazyka pre nevidiacich používateľov.

Jazyk TonIK2 je špecificky navrhnutý pre túto skupinu používateľov, budeme z neho brať inšpiráciu pri definovaní našej syntaxe.

Záver

Literatúra

- [1] Jaws. <https://www.freedomscientific.com/products/software/jaws/>, Accessed: 17.11.2023.
- [2] Nvda. <https://www.nvaccess.org/about-nv-access/>, Accessed: 16.11.2023.
- [3] Walter Bender, Devin Ulibarri, Ymca Malden, and Yash Khandelwal. Music blocks: A musical microworld. 2015.
- [4] Texas School for the Blind and Visually Impaired. Braille display devices. <https://www.tsbvi.edu/statewide-resources/services/braille/display>, Accessed: 17.11.2023.
- [5] Alex Hadwen-Bennett, Sue Sentance, and Cecily Morrison. Making programming accessible to learners with visual impairments: A literature review. *International Journal of Computer Science Education in Schools*, 2, 05 2018.
- [6] Iveta Krempaská. Výučbové prostredie na programovanie melódií prístupné pre nevidiacich žiakov zŠ. diplomová práca, FMFI UK, 2022.
- [7] Christopher Petrie. Programming music with sonic pi promotes positive attitudes for beginners. *Computers Education*, 179:104409, 2022.
- [8] Jaime Sánchez and Fernando Aguayo. Apl: Audio programming language for blind learners. In Klaus Miesenberger, Joachim Klaus, Wolfgang L. Zagler, and Arthur I. Karshmer, editors, *Computers Helping People with Special Needs*, pages 1334–1341, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [9] Andreas Stefik, Christopher Hundhausen, and Derrick Smith. On the design of an educational infrastructure for the blind and visually impaired in computer science. *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 05 2011.

Príloha A: obsah elektronickej prílohy

Príloha B: Používateľská príručka