UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2009 Tomáš Michek

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Program pro výuku a testování základů výrokové a predikátové logiky

Tomáš Michek

Bakalářská práce

2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Tomáš MICHEK**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Program pro výuku a testování základů výrokové a predikátové logiky**

Zásady pro vypracování:

Ô V úvodní části práce je nutné provést seznámení s významem výrokové a predikátové logiky a její nezastupitelnost v informatice
Ô Hlavní úkol spočívá ve vytvoření výukového programu. Ten bude sloužit studentů k domácímu opakování znalostí této části matematiky.
Ô Vytvořený výukový program bude obsahovat matematický výklad tématu a následný test. Dále popis instalace a ovládání programu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **LUKASOVÁ, Alena. Formální logika v umělé inteligenci. [s.l.] : [s.n.], 2003. 269 s. ISBN 80-251-0023-5.**
2. **ŠVEJDAR , Vítězslav. Logika neúplnost, složitost a nutnost. [s.l.] : Academia, 2002. 464 s**

Vedoucí bakalářské práce: **Ing. Soňa Neradová**
Katedra softwarových technologií

Datum zadání bakalářské práce: **15. ledna 2009**

Termín odevzdání bakalářské práce: **15. května 2009**

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan
vedoucí katedry

V Pardubicích dne 31. března 2009

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 12. 5. 2009

Tomáš Michek

SOUHRN

Ve své bakalářské práci se zabývám problematikou výrokové a predikátové logiky. První část práce by měla čtenáře seznámit s významem této části matematiky a s jejími základními pojmy. Druhá část práce se věnuje architektuře výukového programu, který obsahuje kromě samotné teorie také test a který bude sloužit studentům pro domácí opakování znalostí výrokové a predikátové logiky.

KLÍČOVÁ SLOVA

výroková logika, predikátová logika, programovací jazyky, výukové programy

TITTLE

A program for education and testing the basics of propositional and predicate logic

ABSTRACT

In my baccalaureate work deal with problems propositional and predicate logic. First volume of work would had reader acquaint with meaning those parts mathematics and with her basic notions. Alternative volume of work paies architecture tutorial, that contains except himself theory also test and will be of service students for native repetition knowledge dictum and predicate logic.

KEYWORDS

Propositional logic, predicate logic, programming languages, education programs

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Hlavním cílem mé bakalářské práce je vytvoření výukového programu zaměřeného na výrokovou a predikátovou logiku, který bude sloužit studentům jako pomůcka při opakování znalostí této problematiky. Kromě teoretického výkladu bude program obsahovat také procvičování a test.

Teoretická část práce má za úkol seznámit čtenáře nejen se základními pojmy této části matematiky, ale také s významem a nezastupitelností výrokové a predikátové logiky v umělé inteligenci.

Návrhu a vývoji aplikace je věnována druhá část této práce. Výukový program je vytvořen v programu Lazarus, což je vývojové prostředí, jehož ekvivalentem je komerční produkt Delphi od firmy Borland. Na rozdíl od Delphi je však Lazarus multiplatformní. Popis instalace, ovládání programu a seznámení s grafickým prostředím aplikace nalezne čtenář v uživatelské dokumentaci. Samotnou architekturu programu poté zachycuje dokumentace programátorská.

2. Význam formální logiky

Nezbytnou součástí teoretické i aplikované matematiky a informatiky je jazyk výrokové a predikátové logiky.

Teoretická informatika byla donedávna ztotožňována pouze s teorií algoritmu. Postupem času se především vlivem databázových technologií těžiště informatiky stále více přesunovalo směrem ke zpracování informací a k práci se znalostmi v jejich deklarativní podobě. Od informatiky se nyní stále více očekává její epistomologické zaměření, tj. zaměření na zkoumání a modelování postupů lidské duševní činnosti, spočívající v manipulaci se znalostmi a jejich využití k odvozování dalších znalostí. Protože jde o modelování lidského usuzování týkajícího se obecných věcí každodenního života inteligentních bytostí, hovoří se dnes o informatice už nejen jako o teorii algoritmů, ale též o umělé inteligenci. [1]

Pro úspěšné modelování usuzování je však přirozený jazyk až příliš bohatý a složitý a navíc často obsahuje vícevýznamové výrazy a podtextové významy. Formulace pravidel pro používání takových nástrojů komunikace by byla velmi obtížná. Z toho důvodu se vytvářejí zjednodušující modely a jejich jazyky.

Jedním z prostředků, pomocí nichž je možné úspěšně modelovat usuzování, je formální logika. Formální logika bývá také často nazývána jako logika matematická. V této práci je věnována pozornost dvěma typům formální logiky, které se liší svou složitostí a vyjadřovací silou. Ta jednodušší se nazývá výroková logika a ta složitější predikátová logika. V obou těchto případech se jedná o logiku, která je založena na dvouhodnotové interpretaci pojmu pravdivosti. Jinými slovy lze říci, že všechna tvrzení, jimiž se obě logiky zabývají, jsou buď pravdivá nebo nepravdivá, žádná jiná možnost přípustná není.

3. Výroková logika

Logika se zabývá zákonitostmi a formami správného myšlení či usuzování, její název pochází z řeckého slova „logos“ (myšlení, rozum, věda, slovo, pojem, nauka).

3.1. Výrok

Z hlediska logiky je výrokem každé tvrzení, o kterém lze jednoznačně rozhodnout, zda je pravdivé, či nepravdivé. Nejčastěji bývá zapsán jako jednoduchá oznamovací věta, ale je možné ho zapisovat také pomocí matematických symbolů a značek. Pravdivost a nepravdivost výroku se nazývá pravdivostní hodnotou. Pravdivému výroku je přiřazena pravdivostní hodnota *pravda*, nepravdivému výroku pravdivostní hodnota *nepravda*. Pravdivostní hodnota *pravda* se označuje číslicí 1, *nepravda* číslicí 0. Příkladem pravdivého výroku je výraz $1 + 1 = 2$, proto má pravdivostní hodnotu 1. Naproti tomu výraz $1 > 3$ je příkladem výroku nepravdivého, má tedy pravdivostní hodnotu 0.

3.2. Konjunkce

Používají se pro ni symboly AND, & nebo $\wedge$.

Konjunkce je výrok, který vznikne spojením libovolných výroků pomocí logické spojky *a*. Výsledný výrok má poté tvar *A a B*. Příkladem konjunkce je výrok *vešel do domu a rozsvítil*.

Pravdivá je konjunkce tehdy, jsou-li oba výroky, jež obsahuje, pravdivé. Ve všech ostatních případech je nepravdivá. To ukazuje také tabulka konjunkce.

A	**B**	**A $\wedge$ B**
1	1	1
1	0	0
0	1	0
0	0	0

Tabulka 1: Konjunkce, zdroj [autor]

3.3. Disjunkce

Používají se pro ni symboly OR nebo $\vee$.

Disjunkce je výrok, který vznikne spojením libovolných výroků pomocí logické spojky *nebo*. Výsledný výrok má poté tvar *A nebo B*. Příkladem konjunkce je výrok *x = 2 nebo x = 4*.

Pravdivá je disjunkce tehdy, je-li pravdivý alespoň jeden výrok, který se v ní vyskytuje. Nepravdivá je v případě, že jsou oba spojované výroky nepravdivé. To ukazuje také tabulka disjunkce.

A	**B**	**A $\vee$ B**
1	1	1
1	0	1
0	1	1
0	0	0

Tabulka 2: Disjunkce, zdroj [autor]

3.4. Negace

Používají se pro ni symboly NOT, $\neg$, případně se označuje pruhem nad proměnnou.

Negací je výrok, který vznikne přidáním logické spojky *ne-*. Výsledný výrok má poté tvar *ne-A*. Výrok *neprší*, je příkladem negace k výroku *prší*. U výrazů, na které nelze aplikovat spojku *ne-*, se používá obrat *není pravda, že*. Například pro negaci výroku *neprší* je nezbytné tento obrat použít, negací tedy bude výrok *není pravda, že neprší*.

Negace je pravdivá, je-li negovaný výrok nepravdivý. A naopak nepravdivá, je-li negovaný výrok pravdivý. To dokládá také tabulka negace.

A	**$\neg$ A**
1	0
0	1

Tabulka 3: Negace, zdroj [autor]

3.5. Implikace

Používají se pro ni symboly $\rightarrow$ nebo $\Rightarrow$.

Implikace je výrok, který vznikne dosazením libovolných výroků do schématu *jestliže ..., pak ....* Výsledný výrok má poté tvar *jestliže A, pak B*. První výrok v implikaci se nazývá antecedent a druhý konsekvent implikace. Příkladem implikace je výrok *jestliže je neděle, pak škola je zavřená*.

Pravdivá je implikace tehdy, jsou-li oba její výroky buďto pravdivé, nebo oba nepravdivé, nebo antecedent je nepravdivý a konsekvent pravdivý. Nepravdivá je pouze v případě, že antecedent je pravdivý a konsekvent nepravdivý. Jinými slovy implikace nepočítá s tím, že první výrok je pravdivý a druhý nepravdivý. To také dokládá tabulka implikace.

A	**B**	**A $\Rightarrow$ B**
1	1	1
1	0	0
0	1	1
0	0	1

Tabulka 4: Implikace, zdroj [autor]

3.6. Ekvivalence

Používá se pro ni symbol $\Leftrightarrow$.

Ekvivalencí je každý výrok, který vznikne dosazením libovolných výroků do schématu *..., když a jen když ....* Výsledný výrok má poté tvar *A, když a jen když B*. Příkladem ekvivalence je výrok *prší, když a jen když je neděle*. Místo obratu *když a jen když* lze používat také sousloví *tehdy a jen tehdy když* nebo *právě když*.

Pravdivá je ekvivalence tehdy, jsou-li oba její výroky buďto pravdivé, nebo oba nepravdivé. Nepravdivá je v případě, že je jeden z výroků pravdivý a druhý nepravdivý. To ukazuje také tabulka ekvivalence.

A	B	A $\Leftrightarrow$ B
1	1	1
1	0	0
0	1	0
0	0	1

Tabulka 5: Ekvivalence, zdroj [autor]

3.7. Výpočet pravdivostní hodnoty

Určit pravdivostní hodnotu složeného výroku je možné, jsou-li dány pravdivostní hodnoty jeho elementárních složek. Pravdivostní hodnoty jsou těmto složkám přiřazeny pomocí operace nazývané udělení hodnot. Udělením hodnoty *pravda* získá proměnná význam pravdivého výroku, udělením hodnoty *nepravda* naopak význam výroku nepravdivého.

Udělení hodnot přiřazuje hodnoty vždy všem proměnným ve výrazu. Například výraz $p \wedge q \Rightarrow p \vee r$, který obsahuje tři proměnné, může mít udělení hodnot 100. Toto udělení přiznává hodnoty všem třem proměnným výrazu podle jejich abecedního pořadí, abecedně prvnímu písmenu z výrazu (tj. písmenu p) přiznává hodnotu 1, zbylým písmenům (tj. písmenům q a r) hodnoty 0. Celkový počet možných udělení hodnot pro daný výraz závisí na počtu proměnných ve výrazu, pro n proměnných existuje 2^n různých udělení. Výše zmíněný příklad se třemi proměnnými tedy může mít osm různých udělení, konkrétně 111, 110, 101, 100, 011, 010, 001, 000.

Následující řešení popisuje výpočet pravdivostní hodnoty výrazu $p \wedge q \Rightarrow p \vee r$ pro udělení 101:

- udělení hodnot přiřazuje písmenu p hodnotu 1, písmenu q hodnotu 0 a písmenu r hodnotu 1,
- nejprve je třeba určit hodnotu $p \wedge q$, aby byla konjunkce pravdivá, musí být pravdivé oba výroky, výrok q je však nepravdivý, a proto bude mít $p \wedge q$ hodnotu 0,
- výraz lze tedy zjednodušit do tvaru $0 \Rightarrow p \vee r$,
- nyní je třeba určit hodnotu $0 \Rightarrow p$, protože antecedent je nepravdivý a konsekvent (výrok p) pravdivý, bude mít implikace $0 \Rightarrow p$ hodnotu 1,

- výraz lze tedy zjednodušit do tvaru $1 \vee r$,
- nakonec je třeba určit hodnotu $1 \vee r$, vzhledem k tomu, že je splněna podmínka, aby alespoň jeden z výroků disjunkce byl pravdivý, bude mít výraz $1 \vee r$ hodnotu 1,
- tato hodnota je zároveň i pravdivostní hodnotou výrazu $p \wedge q \Rightarrow p \vee r$.

Závěr tedy zní, že výraz $p \wedge q \Rightarrow p \vee r$ má pro udělení 101 pravdivostní hodnotu 1.

3.8. Pravdivostní tabulka

Určovat předchozím způsobem pravdivostní hodnoty pro všechna možná udělení hodnot danému výrazu by nebylo příliš efektivní. Pro takové případy je vhodnější použít konstrukci pravdivostní tabulky.

Řádky tabulky představují výpočty pro jednotlivá udělení hodnot proměnným výrazu. Poslední sloupec tabulky pak bývá sloupcem výsledným, to znamená, že obsahuje výsledné hodnoty výrazu pro jednotlivá udělení hodnot. Počet řádků tabulky odpovídá počtu možných udělení hodnot pro daný výraz.

Pravdivostní tabulka výrazu $p \wedge q \Leftrightarrow p \vee r$ vypadá následovně.

p	**q**	**r**	**p ∧ q**	**p ∧ q ⇔ p**	**p ∧ q ⇔ p ∨ r**
1	1	1	1	1	1
1	1	0	1	1	1
1	0	1	0	0	1
1	0	0	0	0	0
0	1	1	0	1	1
0	1	0	0	1	1
0	0	1	0	1	1
0	0	0	0	1	1

Tabulka 6: Pravdivostní tabulka, zdroj [autor]

Z pravdivostní tabulky lze vyčíst, že výraz $p \wedge q \Leftrightarrow p \vee r$ má pravdivostní hodnotu 0 pouze pro udělení 100, pro všechna ostatní udělení hodnot je jeho pravdivostní hodnota rovna 1.

3.9. Tautologie

Výrok je logicky platným (tautologií), je-li pravdivý pro každé udělení hodnot. Tautologií je například výraz $p \wedge q \Rightarrow p \vee r$, což lze dokázat pomocí pravdivostní tabulky.

p	**q**	**r**	**p ∧ q**	**p ∧ q ⇒ p**	**p ∧ q ⇒ p ∨ r**
1	1	1	1	1	1
1	1	0	1	1	1
1	0	1	0	1	1
1	0	0	0	1	1
0	1	1	0	1	1
0	1	0	0	1	1
0	0	1	0	1	1
0	0	0	0	1	1

Tabulka 7: Tautologie - pravdivostní tabulka, zdroj [autor]

Tabulka ukazuje, že výraz $p \wedge q \Rightarrow p \vee r$ má pro všechna udělení hodnotu 1, jedná se tedy skutečně o tautologii.

3.10. Kontradikce

Opakem tautologie je výrok nesplnitelný (kontradikce), který je pro každé udělení hodnot nepravdivý. Příkladem kontradikce je výraz $[p \wedge \neg q] \wedge [p \Rightarrow q]$, což lze opět dokázat pomocí pravdivostní tabulky.

p	**q**	**¬ q**	**p ∧ ¬ q**	**p ⇒ q**	**[p ∧ ¬ q] ∧ [p ⇒ q]**
1	1	0	0	1	0
1	0	1	1	0	0
0	1	0	0	1	0
0	0	1	0	1	0

Tabulka 8: Kontradikce - pravdivostní tabulka, zdroj [autor]

Z tabulky lze vypozorovat, že výraz $[p \wedge \neg q] \wedge [p \Rightarrow q]$ má pro všechna udělení hodnotu 0, což dokazuje, že se skutečně jedná o kontradikci.

3.11. Důsledkový vztah

V kapitole 3.9. bylo ukázáno, že tautologie platí vždy. Existují ovšem také výroky, jejichž platnost je předpokládána na základě premis (předpokladů), což jsou výroky, jejichž platnost je zaručena.

Vše demonstruje následující příklad, kde jsou dány premisy (pravdivé výroky) $p \vee \neg q$, $r \Rightarrow \neg q$, $p \Leftrightarrow r$ a úkolem je rozhodnout, zda je možné správným úsudkem odvodit závěry $q \Rightarrow r$ a $p \wedge r$. Řešení za pomoci pravdivostní tabulky demonstruje následující postup:

p	**q**	**r**	$\neg$ **q**	**p** $\vee$ $\neg$ **q**	**r** $\Rightarrow$ $\neg$ **q**	**p** $\Leftrightarrow$ **r**	**q** $\Rightarrow$ **r**	**p** $\wedge$ **r**
1	1	1	0	1	0	1		
1	1	0	0	1	1	0		
1	0	1	1	1	1	1	1	1
1	0	0	1	1	1	0		
0	1	1	0	0	0	0		
0	1	0	0	0	1	1		
0	0	1	1	1	1	0		
0	0	0	1	1	1	1	1	0

Tabulka 9: Důsledkový vztah - pravdivostní tabulka, zdroj [autor]

- nejprve je nutné zkonstruovat pravdivostní tabulku a určit hodnoty premis,
- podstatné jsou pouze ty řádky, v nichž jsou současně všechny premisy pravdivé, pro tyto řádky je následně třeba určit také pravdivostní hodnoty závěrů,
- z tabulky je vidět, že závěr $q \Rightarrow r$ je pravdivý ve všech případech, ve kterých platí premisy, tudíž závěr $q \Rightarrow r$ lze odvodit,
- naopak závěr $p \wedge r$ je v jednom případě pravdivý a ve druhém nepravdivý, proto ho na základě daných premis odvodit nelze.

3.12. Normální formy

Normální formy formulí výrokové logiky jsou charakteristické tím, že si vystačí s trojicí logických spojek $\wedge$, $\vee$ a $\neg$. Před vysvětlením normálních forem je nutné seznámit se s *pojmy elementární konjunkce* a *elementární disjunkce*.

Elementární konjunkcí je výraz ve tvaru $a_1 \wedge a_2 \wedge \ldots \wedge a_n$, jestliže každé a je výroková proměnná nebo negovaná výroková proměnná, přičemž žádná z těchto proměnných se ve výrazu nevyskytuje vícekrát než jednou. Příkladem elementární konjunkce jsou výrazy $p \wedge q$ nebo $p \wedge q \wedge \neg r$, nikoli ale $p \wedge q \wedge p$ nebo $r \wedge p \wedge \neg r$. Stejná pravidla platí také pro elementární disjunkci ve tvaru $a_1 \vee a_2 \vee \ldots \vee a_n$. Příkladem elementární disjunkce jsou poté výrazy $p \vee q$ nebo $p \vee r \vee \neg q$, nikoli ale $p \vee q \vee p$ nebo $r \vee p \vee \neg r$.

Nyní je již možné přistoupit k definování normálních forem. V konjunktivní normální formě je výraz tvaru $c_1 \wedge c_2 \wedge \ldots \wedge c_n$, jestliže každé c je elementární disjunkce, přičemž žádná elementární disjunkce se v tomto výrazu nesmí vyskytovat vícekrát než jednou. V disjunktivní normální formě je výraz tvaru $c_1 \vee c_2 \vee \ldots \vee c_n$, jestliže každé c je elementární konjunkce, přičemž se opět žádná z elementárních konjunkcí nesmí v tomto výrazu objevit vícekrát než jednou. Příkladem výrazu v konjunktivní normální formě je $[p \vee \neg q \vee r] \wedge [\neg s \vee p \vee q]$, naopak příkladem výrazu v disjunktivní normální formě je $[p \wedge q \wedge r] \vee \neg s$. Oba tyto výrazy jsou dvoučlennými normálními formami. Přípustná je ale i normální forma jednočlenná, proto výraz $p \vee \neg q \vee r$ je jak ve tříčlenné disjunktivní normální formě, tak i v jednočlenné konjunktivní normální formě.

3.13. Úplné normální formy

Výraz, který obsahuje n výrokových proměnných, je v úplné konjunktivní normální formě, pokud je v konjunktivní normální formě a současně každý z členů této formy obsahuje vždy všech n výrokových proměnných. V úplné konjunktivní normální formě je na příklad výraz $[p \vee q \vee r] \wedge [p \vee \neg q \vee r] \wedge [p \vee \neg q \vee \neg r]$, nikoli ovšem $[p \vee q \vee r] \wedge [p \vee \neg r]$.

Pro výrazy v úplné disjunktivní normální formě platí naprosto stejná pravidla. V takové formě se nachází například výraz $[p \wedge q] \vee [p \wedge \neg q] \vee [\neg p \wedge \neg q]$, nikoli ovšem $[p \wedge q] \vee \neg p$.

Každý výraz výrokové logiky je možné vyjádřit ve tvaru úplné normální formy. Převod výrazu $p \Rightarrow [q \wedge r]$ do tvaru úplné konjunktivní normální formy s využitím pravdivostní tabulky ukazuje následující postup:

p	**q**	**r**	**q ∧ r**	**p ⇒ [q ∧ r]**
1	1	1	1	1
1	1	0	0	0
1	0	1	0	0
1	0	0	0	0
0	1	1	1	1
0	1	0	0	1
0	0	1	0	1
0	0	0	0	1

Tabulka 10: Úplné normální formy - pravdivostní tabulka, zdroj [autor]

- pro převod do úplné konjunktivní normální formy jsou důležité pouze řádky, které mají ve výsledném (posledním) sloupci hodnotu 0,
- jednotlivé členy úplné konjunktivní normální formy se skládají vždy ze všech proměnných vybraných řádků, přičemž platí, že pokud má proměnná v daném řádku udělenu hodnotu 1, je třeba ji znegovat,
- výsledkem je výraz $[\neg p \vee \neg q \vee r] \wedge [\neg p \vee q \vee \neg r] \wedge [\neg p \vee q \vee r]$.

Postup pro převod stejného výrazu do tvaru úplné disjunktivní normální formy je podobný:

- rozdíl je v tom, že při převodu do úplné disjunktivní normální formy jsou podstatné ty řádky pravdivostní tabulky, které mají ve výsledném sloupci hodnotu 1,
- druhým a zároveň posledním rozdílem je to, že je třeba znegovat ty proměnné, které mají v daném řádku hodnotu 0,
- výsledkem je výraz $[p \wedge q \wedge r] \vee [\neg p \wedge q \wedge r] \vee [\neg p \wedge q \wedge \neg r] \vee$ $\vee [\neg p \wedge \neg q \wedge r] \vee [\neg p \wedge \neg q \wedge \neg r]$.

4. Predikátová logika

Výroková logika nezahrnuje všechny úsudky, které lze považovat za správné. Například jsou dány následující tři výroky:

- Tomáš hraje na klavír,
- jestliže někdo hraje na klavír, pak má hudební sluch,
- Tomáš má hudební sluch.

Ve výrokové logice však správnost třetího výroku z předchozích dvou žádným způsobem nevyplývá. První věta totiž ve výrokové logice představuje elementární výrok p, druhá věta má tvar implikace dvou elementárních výroků $q \Rightarrow r$ a poslední věta je elementární výrok s. Ve výrokové logice výrok s z předpokladů p, $q \Rightarrow r$ neplyne.

Pro dokázání předchozího úsudku, je třeba zabývat se vnitřní strukturu jednotlivých vět. Že je úsudek správný, lze rozhodnout až na základě struktury těchto výroků. A právě tím se zabývá predikátová logika.

4.1. Základní pojmy

Mezi základní pojmy predikátové logiky patří *predikáty*, *jména*, *proměnné* a *kvantifikátory*. Pro pochopení těchto pojmů jsou dány následující elementární výroky:

- Tomáš je plnoletý,
- 2 + 1 = 3.

V těchto výrocích se vyskytují vlastní jména. Jsou jimi výrazy *Tomáš*, *1*, *2* a *3*. Jména jsou tedy výrazy, které něco označují nebo pojmenovávají. Věc, kterou jméno označuje, se nazývá *denotátem* jména. Například výraz *Tomáš* je jménem a osoba, kterou toto jméno označuje, je denotátem tohoto jména.

Jména lze dělit podle toho, kolik jim odpovídá denotátů. Jménu vlastnímu odpovídá vždy právě jeden denotát, jménu prázdnému žádný a jménu obecnému odpovídá hned několik denotátů. Jednoznačnost je v takovém případě určena souvislostí, ve které se obecné jméno vyskytuje.

Odstranění vlastních jmen z výše uvedených výrazů vzniknou predikáty. Jsou jimi výrazy $\ldots_1$ *je plnoletý*, $\ldots_1 + \ldots_2 = \ldots_3$, ale také třeba $\ldots_1 + 1 = \ldots_2$. Podle počtu volných míst ve výrazu rozlišujeme predikáty jednomístné, dvoumístné, trojmístné atd. Dále se predikáty dělí na unární a binární. Unární predikát slouží k vyjádření vlastnosti určitého objektu. Naopak predikát binární vyjadřuje vztah mezi dvěma objekty.

Dalším pojmem predikátové logiky jsou proměnné, které se vyskytují v následujících výrazech:

- x je plnoletý,
- $x + y = z$.

Z těchto výrazů je patrné, že se proměnné v mnohém podobají vlastním jménům. Nikdy ale nic fixně nepojmenovávají, pouze mohou pro danou úvahu převzít roli vlastního jména. Po významové stránce je proměnná určena definiční oborem označovaným písmenem D, který je dané proměnné přiřazen. Prvky tohoto oboru se nazývají hodnotami proměnné. Pomocí operace, jež se nazývá udělení hodnoty proměnné, lze s obsahem proměnné manipulovat, to znamená přiřadit proměnné některý z prvků jejího definičního oboru.

Ještě se zbývá zmínit o kvantifikátorech. Existuje kvantifikátor existenční $\exists$ s významem *pro některé* a kvantifikátor obecný $\forall$ s významem *pro každé*. Připojením kvantifikátoru $\exists$ před výraz vznikne výrok existenční, připojením kvantifikátoru $\forall$ výrok obecný. Podle postavení kvantifikátorů se proměnné dělí na vázané nebo volné. Je-li výskyt proměnné v dosahu některého z kvantifikátorů, jedná se o výskyt vázaný. V opačném případě jde o výskyt volný.

Po seznámení se základními pojmy by neměl být problém převést předchozí výroky do jazyka predikátové logiky. Například výrok *Tomáš je plnoletý* bude zapsán ve tvaru $P(t)$, kde t označuje *Tomáše* a P vlastnost (predikát) *být plnoletým*.

4.2. Logické funkce

Funkce je operace, která se aplikuje na argument funkce a jako výsledek dává hodnotu funkce pro daný argument. Pro představu existuje funkce f aplikovaná na argument x, kterému přiřazuje hodnotu y. Takováto funkce má zápis $y = f(x)$. Pro-

tože má tato funkce jediný argument, nazývá se jednoargumentová. Funkce aplikované na *n* argumentů se poté nazývají n-argumentové.

Za použití funkcí lze zformalizovat následující věty:

- Tomášův otec je hudebník,
- něčí otec je hudebník.

Pro první větu je potřeba konstanta pro *Tomáše* (t), predikát *být hudebníkem* (H) a funkce, která člověku přiřadí jeho *otce* (o). Výsledná formule bude mít tvar $H(o(t))$.

U druhé věty je navíc potřeba existenční kvantifikátor $\exists$ a místo *Tomáše* je nutné použít proměnnou *x*. Výsledná formule bude mít v tomto případě tvar $\exists x\, H(o(x))$.

4.3. Výpočet pravdivostní hodnoty

Výpočet pravdivostní hodnoty demonstrují následující příklady, pro které je dán definiční obor $D = \{a_0, a_1, a_2\}$. Definičním oborem predikátových proměnných prvního stupně je poté třída logických funkcí definovaná následující tabulkou.

x	λ_0	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
a_0	0	0	0	0	1	1	1	1
a_1	0	0	1	1	0	0	1	1
a_2	0	1	0	1	0	1	0	1

Tabulka 11: Třída logických funkcí, zdroj [autor]

Zadáním prvního příkladu je určit pravdivostí hodnotu výrazu $P(x) \Rightarrow Q(y)$ pro udělení hodnot x/a_0, y/a_2, P/λ_3, Q/λ_6. Samotný postup je následující:

- původní výraz $P(x) \Rightarrow Q(y)$ lze pro dané udělení hodnot přepsat do tvaru $\lambda_3(a_0) \Rightarrow \lambda_6(a_2)$,
- poté stačí vyhledat odpovídající buňky v tabulce, kde výrazu $\lambda_3(a_0)$ odpovídá hodnota 0 a výrazu $\lambda_6(a_2)$ také hodnota 0,
- vzhledem k tomu, že je antecedent i konsekvent nepravdivý, bude mít celá implikace hodnotu 1,

- tímto bylo dokázáno, že výraz $P(x) \Rightarrow Q(y)$ má pro udělení hodnot x/a_0, y/a_2, P/λ_3, Q/λ_6 pravdivostní hodnotu 1.

Zadáním druhého příkladu je určit pravdivostí hodnotu výrazu $\forall x\,[\neg Q(x) \Rightarrow P(x)]$ pro udělení hodnot P/λ_4, Q/λ_2. Postup vypadá následovně:

- vzhledem k tomu, že proměnná x je vázaná, protože je v dosahu obecného kvantifikátoru, je nutné vyhodnotit výraz $\forall x\,[\neg Q(x) \Rightarrow P(x)]$ pro všechna udělení hodnot proměnné x,
- konkrétně to jsou tyto tři výrazy $\neg\lambda_2(a_0) \Rightarrow \lambda_4(a_0)$, $\neg\lambda_2(a_1) \Rightarrow \lambda_4(a_1)$, $\neg\lambda_2(a_2) \Rightarrow \lambda_4(a_2)$,
- po vyhledání hodnot v příslušných buňkách tabulky lze zjistit hodnoty jednotlivých implikací, výrazy $\neg\lambda_2(a_0) \Rightarrow \lambda_4(a_0)$ a $\neg\lambda_2(a_1) \Rightarrow \lambda_4(a_1)$ mají hodnotu 1, výraz $\neg\lambda_2(a_2) \Rightarrow \lambda_4(a_2)$ hodnotu 0,
- vzhledem k tomu, že je před výrazem připojen obecný kvantifikátor s významem *pro každé*, musí být výraz pravdivý pro každé udělení,
- v tomto případě však tento požadavek splněn není, což znamená, že výraz $\forall x\,[\neg Q(x) \Rightarrow P(x)]$ má pro udělení P/λ_4, Q/λ_2 hodnotu 0.

5. Uživatelská dokumentace

5.1. O programu

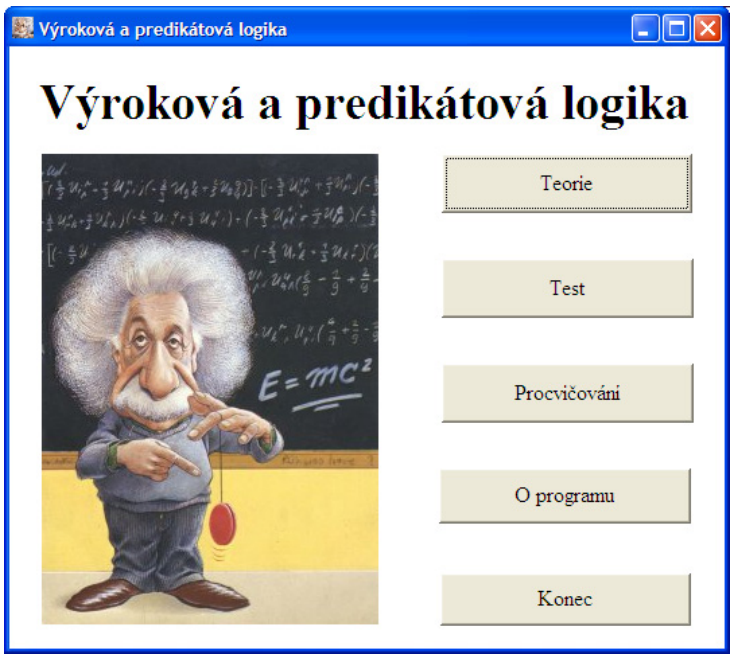
Program slouží studentům k opakování a procvičování znalostí ze základů výrokové a predikátové logiky. Obsahuje několik stránek teoretického výkladu, které může uživatel prolistovávat. Poté si student může ověřit své znalosti v testu, který se skládá z deseti náhodně vygenerovaných otázek, nebo zvolit pouze procvičování, ve kterém není počet otázek nijak omezen.

5.2. Instalace programu

Instalace programu je velmi jednoduchá. Po spuštění samorozbalovacího archivu *VyukovyProgram.exe* uživatel vybere složku, kam se má program nainstalovat. Zde se vytvoří adresář s názvem *Výukový program - výroková a predikátová logika*, který obsahuje složku s daty a také soubor *Spustit.exe*, jenž zajišťuje spuštění programu.

5.3. Program - menu

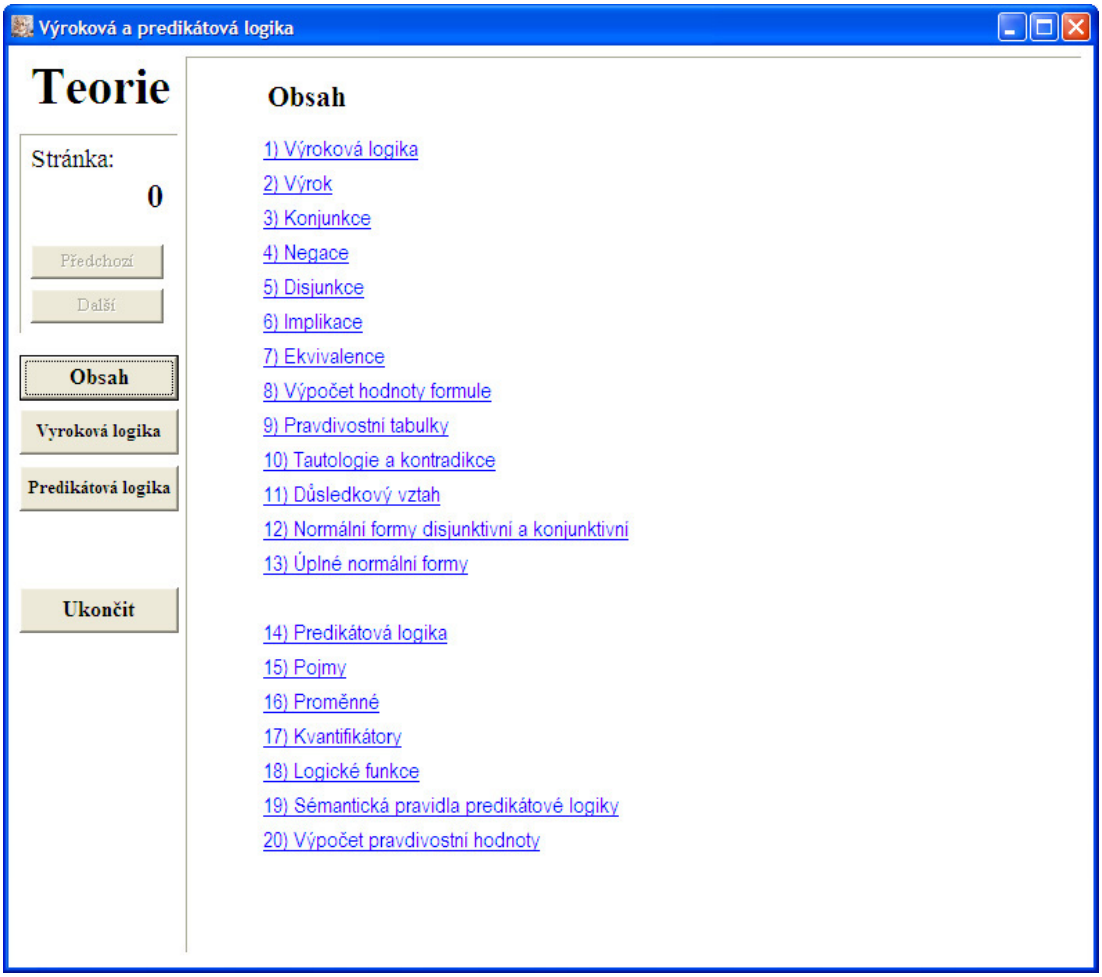
Po spuštění programu se objeví úvodní obrazovka, která je zobrazena na obrázku 1.



Obrázek 1: Výukový program - menu, zdroj [autor]

5.4. Program – teorie

Po stisknutí tlačítka *Teorie* v hlavním menu se zobrazí okno, které je zachyceno na obrázku 2.

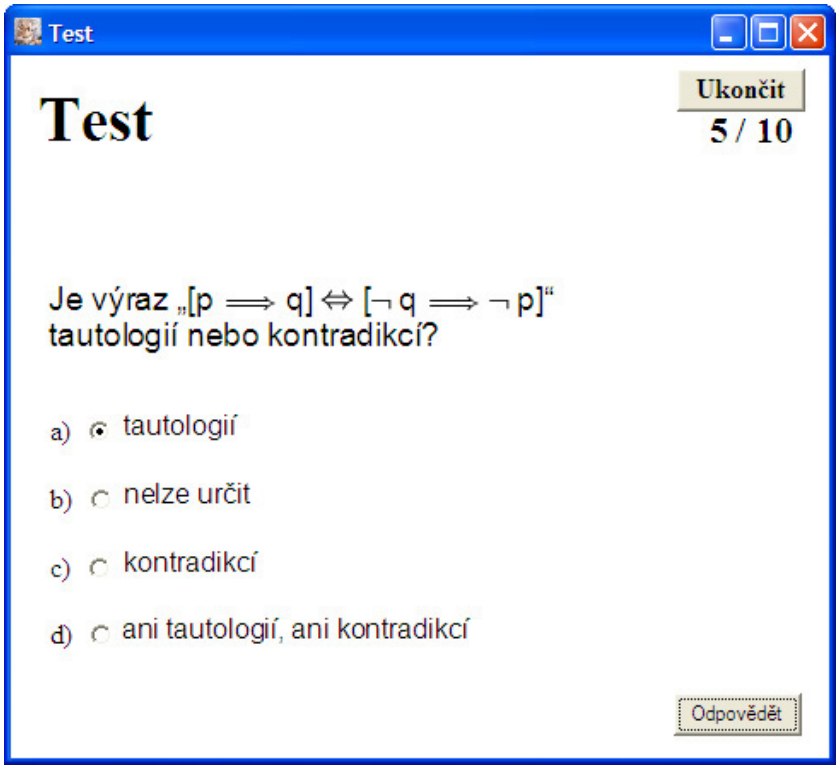


Obrázek 2: Výukový program - teorie, zdroj [autor]

Uživatel si může z obsahu vybrat libovolnou kapitolu, kterou si zobrazí kliknutí na její název. Zvolená kapitola se zobrazí místo obsahu a zároveň se v levém horním rohu zaktualizuje informace o pořadovém čísle prohlížené stránky. Vrátit se poté zpět na obsah je možné pomocí tlačítka *Obsah*. Pro rychlejší procházení může uživatel využít tlačítek *Výroková logika* a *Predikátová logika*, která zobrazí vždy úvodní stránku zvolené kapitoly. Listování po jednotlivých stránkách lze provádět pomocí tlačítek *Předchozí* nebo *Další*. Návrat do hlavního menu programu je možné uskutečnit pomocí tlačítka *Ukončit*.

5.5. Program – test

Volba *Test* v hlavním menu programu vyvolá okno, ve kterém uživatel musí vyplnit své jméno a příjmení. Následně si spustí test pomocí tlačítka *Spustit test*. Test se skládá z deseti otázek, které jsou po spuštění testu náhodně vygenerovány. Prvních šest je zaměřeno na výrokovou logiku a zbývající čtyři se věnují logice predikátové. Jak samotný test vypadá, ukazuje obrázek 3.



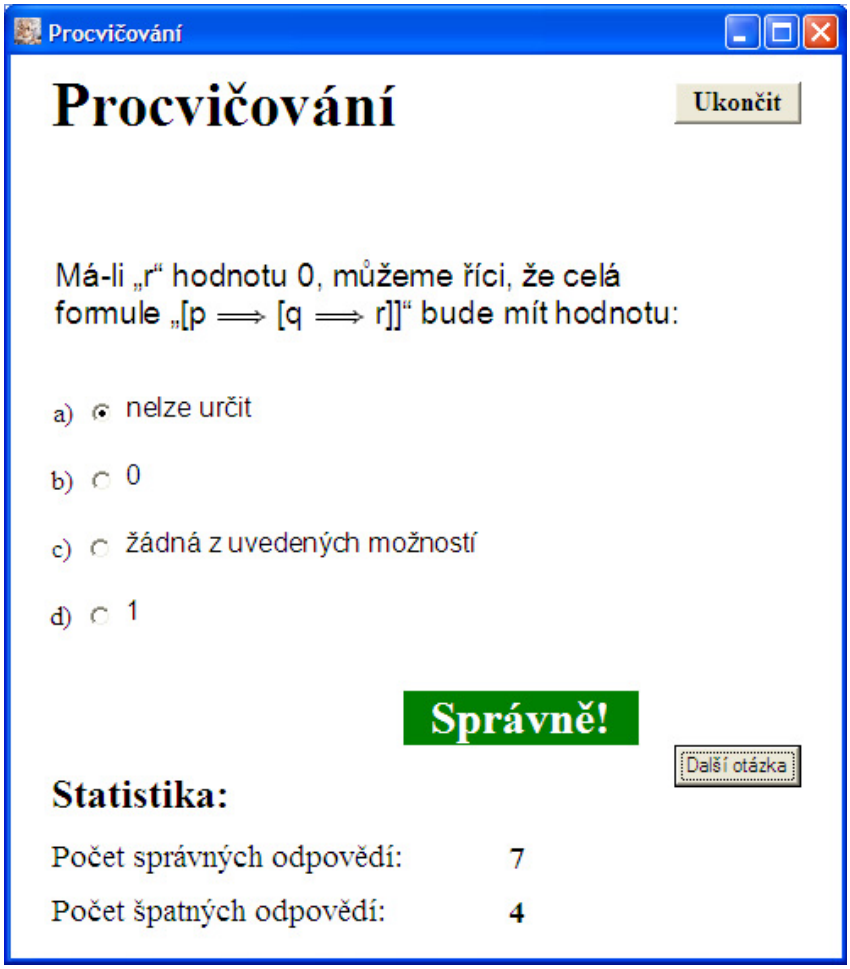
Obrázek 3: Výukový program - test, zdroj [autor]

U každé otázky jsou k dispozici čtyři odpovědi, vždy je správná pouze jediná z nich. Uživatel označí odpověď, která je podle jeho mínění správná, a stiskne tlačítko *Odpovědět*. Po zodpovězení se již není možné k otázce vrátit. Pro lepší přehled o průběhu testu se v pravém horním rohu se zobrazuje informace s pořadovým číslem otázky, na kterou uživatel právě odpovídá. Po zodpovězení poslední otázky testu se kompletní test odešle pomocí FTP, aby měl výsledky k dispozici také vyučující. Uživateli se následně zobrazí pouze statistika, která obsahuje počet správných a špatných odpovědí a procentuální úspěšnost.

V případě, že se nepodaří výsledky odeslat, může uživatel ukončit test bez odeslání výsledků, nebo se pokusit výsledky poslat znova.

5.6. Program – procvičování

Po zvolení možnosti *Procvičování* v hlavním menu dojde ke zobrazení formuláře, kde má uživatel možnost vyzkoušet si náhodně generované otázky. Jejich počet není žádným způsobem omezen, tudíž se ihned po zodpovězení vygeneruje další otázka. Okna s procvičováním je zobrazeno na obrázku 4.

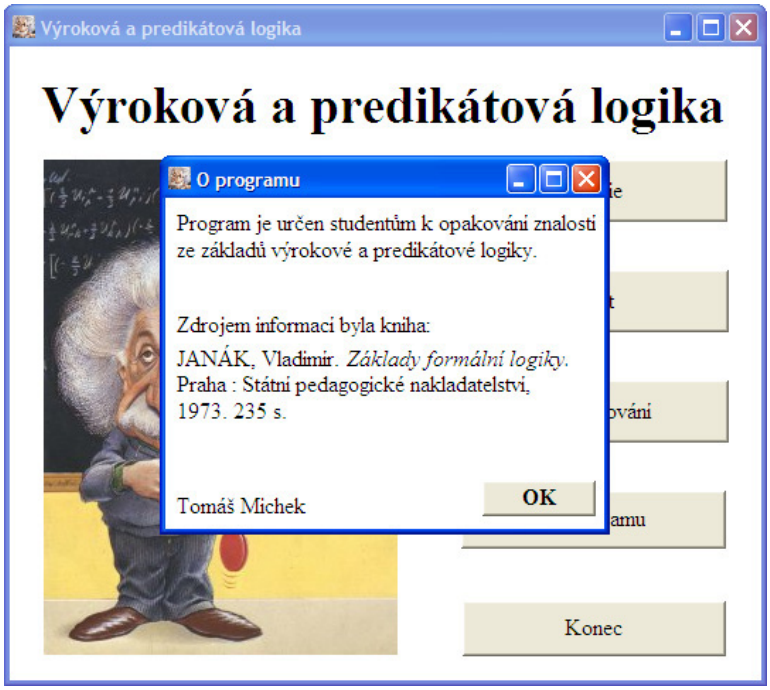


Obrázek 4: Výukový program - procvičování, zdroj [autor]

Zodpovězení otázky je stejné jako u testu. Rozdíl je v tom, jak již bylo řečeno, že počet otázek není nijak omezen. Otázky se tedy generují až do doby, než uživatel stiskne tlačítko *Ukončit*. Další odlišností od testu je to, že se uživatel ihned po každé otázce dozví, zda odpověděl správně nebo špatně. Kromě toho se také průběžně po celou dobu procvičování počítá statistika s počtem správných a špatných odpovědí.

5.7. Program – další volby

Poslední dvě možnosti v hlavním menu, které zbývá popsat, se skrývají pod tlačítky *O programu* a *Konec*. Druhé tlačítko, jak název napovídá, zajišťuje ukončení výukového programu a po stisknutí tlačítka *O programu* se uživateli pouze zobrazí informace týkající se výukového programu, jak je vidět na obrázku 5.



Obrázek 5: Výukový program - o programu, zdroj [autor]

6. Programátorská dokumentace

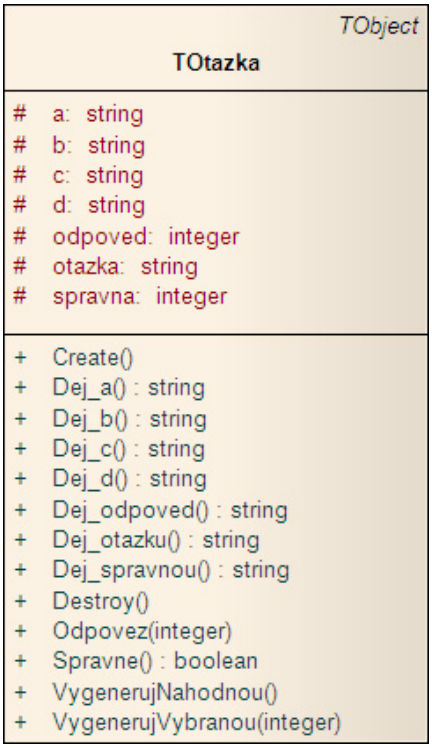
6.1. Členění programu

Výukový program byl vytvořen ve vývojovém prostředí Lazarus, které je velmi podobné komerčnímu produktu Borland Delphi. Na rozdíl od Delphi je však Lazarus multiplatformní, lze spustit na Windows, na Linuxu, v BSD a na mnoha dalších platformách.

Celý program je členěn do několika modulů a to modul *otazka.pas* (obsahuje třídu TOtazka), modul *generovanytest.pas* (obsahuje třídu TTest), *kniha.pas* (obsahuje třídu Kniha). Moduly *menu.pas*, *teorie.pas*, *procvicovani.pas*, *test.pas* a *about.pas* jsou moduly formulářů. Pro odeslání výsledků přes FTP je použit modul *ftpsend.pas* (obsahuje třídu TFTPSend), který využívá ještě moduly *blcksock.pas*, *jclappinst.pas*, *jclbase.pas*, *jclresources.pas*, *sswin32.pas*, *synacode.pas*, *synafpc.pas*, *synaip.pas*, *synautil.pas* a *synsock.pas*. Všechny tyto moduly jsou součástí Synapse, což je volně šiřitelná knihovna tříd.

6.2. Třída TOtazka

6.2.1. Diagram třídy TOtazka



Obrázek 6: Diagram třídy TOtazka, zdroj [autor]

6.2.2. Popis třídy TOtazka

Vzhledem k tomu, že se ve výrokové a predikátové logice vyskytují znaky, jako jsou kvantifikátory a další matematické symboly, bylo by obtížné zobrazovat otázky klasicky v textové podobě. Z toho důvodu jsou všechny otázky i odpovědi uloženy v PNG souborech. Program pracuje s celkem 150 otázkami, přičemž každá se skládá z pěti obrázkových souborů (zadání otázky a čtyři možné odpovědi). Názvy PNG souborů jsou číslovány takovým způsobem, aby první bylo vždy zadání otázky, druhá správná odpověď a poté tři špatné možnosti. Třída *TOtazka* si pořadí odpovědí náhodně přehází.

6.2.3. Proměnné a metody třídy TOtazka

Proměnné třídy TOtazka:

- *otazka* – řetězec obsahující cestu k PNG souboru se zadáním otázky,

- *a* – řetězec obsahující cestu k PNG souboru s první možnou odpovědí,
- *b* – řetězec obsahující cestu k PNG souboru s druhou možnou odpovědí,
- *c* – řetězec obsahující cestu k PNG souboru s třetí možnou odpovědí,
- *d* – řetězec obsahující cestu k PNG souboru se čtvrtou možnou odpovědí,
- *spravna* – číslo správné odpovědi,
- *odpoved* – číslo odpovědi, kterou označil uživatel.

Metody třídy TOtazka:

- *Create()* – konstruktor třídy *TOtazka*,
- *Destroy()* – destruktor třídy *TOtazka*,
- *VygenerujNahodnou()* – procedura, která vygeneruje jednu ze 150 otázek a náhodně změní pořadí možných odpovědí, na základě vygenerovaných údajů naplní hodnoty všech proměnných (do proměnné *odpoved* vloží vždy nulovou hodnotu), tato procedura se využívá při procvičování,
- *VygenerujVybranou(integer)* – obdoba předchozí procedury s tím rozdílem, že nevybírá otázku náhodně, ale podle číselné hodnoty v parametru, využívá se při testu,
- *Dej_otazku()* – funkce, která jako návratovou hodnotu předá řetězec s cestou k PNG souboru se zadáním otázky,
- *Dej_a()* – funkce, která jako návratovou hodnotu předá řetězec s cestou k PNG souboru s první možnou odpovědí,
- *Dej_b()* – funkce, která jako návratovou hodnotu předá řetězec s cestou k PNG souboru s druhou možnou odpovědí,
- *Dej_c()* – funkce, která jako návratovou hodnotu předá řetězec s cestou k PNG souboru s třetí možnou odpovědí,
- *Dej_d()* – funkce, která jako návratovou hodnotu předá řetězec s cestou k PNG souboru se čtvrtou možnou odpovědí,
- *Dej_odpoved()* – funkce, která jako návratovou hodnotu předává řetězec označující odpověď, kterou zvolil uživatel,
- *Odpovez(integer)* – procedura, která nastaví hodnotu proměnné *odpoved* na hodnotu předanou parametrem,

- *Dej_spravnou()* – funkce, která jako návratovou hodnotu předává řetězec označující správnou odpověď, tento řetězec se určuje na základě hodnoty v proměnné *spravna*,
- *Spravne()* – funkce vracející booleovskou hodnotu, pokud uživatel zodpověděl danou otázku správně, vrací hodnotu *true*, v opačném případě *false*.

6.3. Třída TTest

6.3.1. Diagram třídy TTest

Obrázek 7: Diagram třídy TTest, zdroj [autor]

6.3.2. Popis třídy TTest

Třída *TTest* zajišťuje poskládání testu z otázek, jejichž čísla jsou náhodně vygenerována. Jak již bylo zmíněno, celkem má program k dispozici 150 otázek. Ty jsou podle náročnosti rozděleny do deseti kategorií po patnácti otázkách. Aby se nemohlo stát, že některý test bude obsahovat pouze lehčí a jiný naopak pouze těžší otázky, vybírá se právě jedna otázka z každé kategorie. Teoreticky tedy může být sestaveno 15^{10} (zhruba 577 miliard) různých testů.

Po dokončení testu se výsledky uloží do HTML souboru, pomocí kterého jsou odeslány ke kontrole vyučujícímu.

6.3.3. Proměnné a metody třídy TTest

Proměnné třídy TTest:

- *jmeno* – řetězec obsahující jméno uživatele vyplňujícího test,
- *prijmeni* – řetězec obsahující příjmení uživatele vyplňujícího test,
- *pocet_otazek* – číslo obsahující informaci o počtu otázek v testu (uživatel nemá možnost počet otázek měnit, každý test má tedy deset otázek),
- *otazky* – dynamické pole obsahující prvky typu *TOtazka*.

Metody třídy TTest:

- *Create(integer, string, string)* – konstruktor třídy *TTest*, jeho první parametr udává počet otázek v testu (podle něho se nastaví hodnota proměnné *pocet_otazek* a také délka dynamického pole *otazky*), zbylé parametry obsahují řetězce se jménem a příjmením uživatele (konstruktor podle nich nastaví proměnné *jmeno* a *prijmeni*),
- *Destroy()* – destruktor třídy *TTest* zajišťující uvolnění paměti, která byla přidělena jednotlivým otázkám v testu,
- *VygenerujTest()* – procedura zajišťující vygenerování testu, náhodně vybere z každé kategorie jednu otázku a objekt typu *TOtazka* vloží do pole *otazky*,
- *Dej_otazku(integer)* – funkce, která vrací jako návratovou hodnotu prvek typu *TOtazka*, předán je ten prvek, jehož pozice v poli *otazky* odpovídá hodnotě celočíselného parametru funkce,
- *PocetSpravnych()* – funkce vracející celočíselnou hodnotu, která odpovídá počtu správně zodpovězených otázek v testu,
- *Uspenost()* – funkce, která jako návratovou hodnotu předává reálné číslo odpovídající procentuální úspěšnosti dosažené v testu,
- *NazevSouboru()* – funkce vracející řetězec, který je sestaven z hodnot proměnných *jmeno* a *prijmeni* a také z hodnoty času, ve kterém byl test dokončen, tento řetězec slouží jako název HTML souboru, do něhož bude celý test uložen,
- *ObsahSouboru()* – funkce předávající jako návratovou hodnotu objekt typu *TStringList*, což je víceřádkový text, který představuje obsah HTML souboru,

jedná se tedy o HTML kód, jenž popisuje všechny otázky z testu (barevně jsou odlišeny správné a špatné odpovědi),

- *Uloz()* – funkce, která využívá obě předchozí metody, do umístění *data/testy/* uloží HTML soubor s názvem získaným od metody *NazevSouboru()* a obsahem získaným od metody *ObsahSouboru()*, jako návratovou hodnotu předává řetězec s cestou k uloženému souboru.

6.4. Třída TKniha

6.4.1. Diagram třídy TKniha

Obrázek 8: Diagram třídy TKniha, zdroj [autor]

6.4.2. Popis třídy TKniha

Třída *TKniha* slouží ke zobrazení teorie ve výukovém programu a plní podobné funkce jako běžná kniha v reálném světě. Umožňuje nejen zobrazit stránku se zvoleným číslem, ale také postupné listování. Z důvodu, že se v textu vyskytují kvantifikátory a další matematické symboly, jsou jednotlivé stránky uloženy ve formátu PNG (stejně jako testové otázky).

6.4.3. Proměnné a metody třídy TKniha

Proměnné třídy TKniha:

- *stranky* – pole, jehož prvky obsahují řetězce s cestami k PNG souborům, které odpovídají jednotlivým stránkám,
- *aktualni_stranka* – pořadové číslo právě zobrazené stránky,

- *celkem_stranek* – číslo odpovídající celkovému počtu stránek s teorií.

Metody třídy TKniha:

- *Create()* – konstruktor třídy *TKniha*, který naplní pole *stranky* řetězci s cestami k jednotlivým PNG souborům,
- *Destroy()* – destruktor třídy *TKniha*,
- *DejVybranou(integer)* – funkce, která vrací řetězec s cestou k PNG souboru, tento řetězec je vybrán z pole *otazky* z pozice odpovídající předanému parametru,
- *DejDalsi()* – funkce předávající jako návratovou hodnotu řetězec z pole *stranky*, který je umístěn na pozici s hodnotou *aktualni_stranka + 1* a následně zvýší o jedničku hodnotu proměnné *aktualni_stranka* (předtím samozřejmě kontroluje, zda již není poslední stránka zobrazena, v takovém případě vrací prázdný řetězec),
- *DejPredchozi()* – obdoba předchozí funkce s tím rozdílem, že neposkytuje následující prvek pole *stranky*, ale předchozí,
- *DejCisloAktualni()* – funkce, která vrací celočíselnou hodnotu odpovídající hodnotě v proměnné *aktualni_stranka*.

6.5. Třídy formulářů

6.5.1. Třída TForm_menu

Třída *TForm_menu* definuje úvodní obrazovku programu (viz obrázek 1). Metodami této třídy jsou:

- *FormCreate(TObject)* – procedura, která je volána při vytvoření formuláře, provede příkaz *randomize*, čímž zajistí správnost náhodného generování otázek v testu a v procvičování,
- *button_teorieClick(TObject)* – procedura, která se vykoná po kliknutí na tlačítko *Teorie*, zajišťuje zobrazení okna s teorií,
- *button_testClick(TObject)* – procedura, která se vykoná po stisknutí tlačítka *Test*, zajišťuje zobrazení okna s testem,

- *button_procvicovaniClick(TObject)* – procedura, která se vykoná po kliknutí na tlačítko *Procvicovani*, zajišťuje zobrazení okna s procvičováním,
- *button_aboutClick(TObject)* – procedura, která se vykoná po stisknutí tlačítka *O programu*, zobrazí modální okno obsahující informace o výukovém programu,
- *button_konecClick(TObject)* – procedura, která se vykoná po kliknutí na tlačítko *Konec* a zajistí ukončení programu.

6.5.2. Třída TForm_teorie

Tato třída definuje okno s teorií (viz obrázek 2). Metodami této třídy jsou:

- *FormCreate(TObject)* – procedura, která je volána při vytvoření formuláře, zajistí zobrazení obsahu a vytvoří objekt typu *TKniha*, který přiřadí do globální proměnné *knizka*,
- *label1Click(TObject)* až *label20Click(TObject)* – procedury, které reagují na kliknutí na názvy jednotlivých kapitol v obsahu, zajistí skrytí obsahu a pomocí objektu *knizka* a jeho metody *DejVybranou* zobrazí příslušnou kapitolu teoretického výkladu,
- *button_dalsiClick(TObject)* – procedura reagující na stisknutí tlačítka *Další*, pomocí objektu *knizka* a jeho metody *DejDalsi* zobrazí následující stránku,
- *button_predchoziClick(TObject)* – obdoba předchozí procedury s tím rozdílem, že využívá metodu *DejPredchozi* objektu *knizka* a zajistí tak zobrazení stránky předešlé,
- *button_vyrokClick(TObject)* – procedura vyvolaná stisknutím tlačítka *Výroková logika*, pomocí objektu *knizka* a jeho metody *DejVybranou* zobrazí první stránku z výrokové logiky,
- *button_predikatClick(TObject)* – obdobná procedura jako předchozí, rozdíl je pouze v tom, že reaguje na stisknutí tlačítka *Predikátová logika* a zobrazuje první stránku z této kapitoly,
- *button_obsahClick(TObject)* – procedura reagující na stisknutí tlačítka *Obsah*, zajišťuje opětovné zobrazení obsahu,
- *button_konecClick(TObject)* – procedura vyvolaná kliknutím na tlačítko *Ukončit*, zajistí zavření okna s teorií a zobrazení hlavního menu programu.

6.5.3. Třída TForm_test

Třída TForm_test definuje okno s testem, které je vidět na obrázku 3. Metodami této třídy jsou:

- *ZakladniObrazovka()* – pomocná procedura, která zajistí zobrazení úvodní obrazovky testu s políčky pro zadání příjmení a jména uživatele a tlačítkem pro spuštění testu,
- *Vysledek()* – pomocná procedura, která je využívána po dokončení testu, pokusí se pomocí FTP odeslat výsledky uložené v HTML souboru, pokud je odeslání úspěšné, zobrazí uživateli informace o počtu správných odpovědí a procentuální úspěšnost testu, v případě neúspěšného odeslání jsou uživateli zobrazena tlačítka pro další pokus o odeslání nebo naopak pro ukončení testu bez odeslání výsledků,
- *FormCreate(TObject)* – procedura, která je prováděna při vytvoření formuláře, zavolá pomocnou proceduru *ZakladniObrazovka*,
- *button_spustitClick(TObject)* – procedura volaná při kliknutí na tlačítko *Spustit test*, zkontroluje, zda uživatel skutečně vyplnil jméno i příjmení, vytvoří objekt typu *TTest* (jako parametry předává konstruktoru tohoto objektu jméno a příjmení uživatele a počet otázek v testu, který je pevně nastaven na hodnotu 10) a přiřadí ho do globální proměnné *vygenerovany_test*, tento objekt následně pomocí své metody *VygenerujTest* vygeneruje deset testových otázek, první z nich je poté zobrazena uživateli,
- *button_odpovedetClick(TObject)* – procedura vyvolaná stisknutím tlačítka *Odpovědět*, informace o tom, jakou odpověď uživatel označil, je předána objektu typu *TOtazka* pomocí parametru metody *Odpovez,* jestliže uživatel odpovídal na poslední otázku v testu, uloží objekt *vygenerovany_test* pomocí své metody *Uloz* kompletní obsah testu do HTML souboru a zavolá se pomocná procedura *Vysledek*, v opačném případě se zobrazí další otázky testu,
- *button_odeslatClick(TObject)* – procedura, reagující na stisknutí tlačítka *Odeslat* (toto tlačítko je zobrazeno pouze v případě, že selhal pokus o odeslání výsledků na FTP server), zajistí opětovné zavolání pomocné procedury *Vysledek*,

- *button_neodesilatClick(TObject)* – procedura reagující na tlačítko *Neodesílat* (toto tlačítko je rovněž zobrazeno jen v případě, že selhal pokus o odeslání výsledků na FTP server), na rozdíl od předchozí metody se nepokusí výsledky odeslat znovu, ale pouze uživateli zobrazí počet správných odpovědí a procentuální úspěšnost testu,
- *button_konecClick(TObject)* – procedura vyvolaná kliknutím na tlačítko *Ukončit*, zavolá destruktor objektu *vygenerovany_test*, zajistí zavření okna s testem (před zavřením zavolá pomocnou proceduru *ZakladniObrazovka* pro případ opětovného spuštění testu) a zobrazení hlavního menu programu.

6.5.4. Třída TForm_procvicovani

Tato třída definuje okno s procvičováním (viz obrázek 4). Metodami této třídy jsou:

- *FormCreate(TObject)* – procedura, která je volána při vytvoření formuláře, vytvoří objekt typu *TOtazka*, který přiřadí do globální proměnné *vygenerovana_otazka*, a pomocí jeho metody *VygenerujNahodnou* mu nastaví hodnoty atributů, následně otázku zobrazí uživateli,
- *button_odpovedetClick(TObject)* – procedura vyvolaná po stisknutí tlačítka *Odpovědět*, pomocí parametru metody *Odpovez* je objektu *vygenerovana_otazka* předána informace o tom, jakou odpověď uživatel označil, poté je odpověď vyhodnocena a uživateli se zobrazí hláška, zda odpověděl správně nebo špatně, zaktualizují se také informace o počtu správných a špatných odpovědí, nakonec se místo tlačítka pro odpověď zobrazí tlačítko pro přejití k další otázce,
- *button_dalsiClick(TObject)* – procedura reagující na kliknutí na tlačítko *Další otázka*, objektu *vygenerovana_otazka* jsou pomocí jeho metody *VygenerujNahodnou* nastaveny nové hodnoty atributů, které obsahují informace o další otázce, která je následně uživateli zobrazena, místo tlačítka pro další otázku se samozřejmě opět objeví tlačítko pro odpověď,
- *button_konecClick(TObject)* – procedura, která se vykoná po kliknutí na tlačítko *Ukončit*, zajistí vynulování informací o počtu správných a špatných odpovědí, zavře okno s procvičováním a zobrazí hlavní menu programu.

6.5.5. Třída TForm_about

Třída *TForm_about* definuje okno s informacemi o programu, které je vidět na obrázku 5. Jedná se o modální okno, které má jedinou metodu:

- *button_konecClick(TObject)* – procedura prováděná po stisknutí tlačítka *OK*, zajišťuje pouze zavření modálního okna.

7. Závěr

Věřím, že se mi v teoretické části této bakalářské práce podařilo čtenáři stručně a jasně vysvětlit základní pojmy a znalosti týkající se výrokové a predikátové logiky, která má v posledních letech v oblasti informatiky stále větší význam.

Hlavním cílem práce bylo vytvoření programu pro výuku a testování základů výrokové a predikátové logiky. Tento program byl vytvořen ve vývojovém prostředí Lazarus, které je možné volně stáhnout z webových stránek tohoto projektu. Uživatel výukového programu má k dispozici teoretický výklad této části matematiky, následně si své znalosti může ověřit pomocí procvičování nebo formou testu, jehož výsledky jsou k dispozici také vyučujícímu.

Před uvedením do provozu by bylo možné doplnit výukový program o volbu počtu otázek v testu a také o časový limit, kterým by byl uživatel při jeho řešení omezen. U možnosti volby počtu otázek v testu spočívá hlavní problém v tom, ze kterých kategorií by se měly otázky vybírat. Pokud by se vybíraly úplně náhodně, bez ohledu na kategorie, nebylo by zaručeno, že všichni uživatelé budou mít stejně náročné testy. Proto by bylo nutné pro jednotlivé počty otázek v testu přesně definovat, kolik otázek z které kategorie se má vybrat. Co se týče časového limitu, myslím si, že vzhledem k tomu, že program bude sloužit převážně pro domácí opakovaní, není časový limit nezbytně nutný.

Celkem lze aplikaci považovat za vhodný nástroj k procvičování znalostí výrokové a predikátové logiky.

8. Použitá literatura a ostatní zdroje

[1] LUKASOVÁ, Alena. Formální logiky v umělé inteligenci. 1. vyd. Brno : Computer Press, 2003. 269 s. ISBN 80-251-0023-5.

[2] JANÁK, Vladimír. *Základy formální logiky*. 1. vyd. Praha : Státní pedagogické nakladatelství, 1973. 235 s.

[3] WIKIPEDIE. *Logika* [online]. 2009 , 4. 4. 2009 [cit. 2009-04-23]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Logika>.

[4] WIKIPEDIE. *Výroková logika* [online]. 2009 , 4. 4. 2009 [cit. 2009-04-23]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Výroková_logika>.

[5] WIKIPEDIE. *Predikátová logika* [online]. 2009 , 3. 1. 2009 [cit. 2009-04-23]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Predikátová_logika>.

[6] WIKIPEDIE. *Konjunkce (matematika)* [online]. 2008 , 26. 3. 2009 [cit. 2009-04 23]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Konjunkce_(matematika)>.

[7] WIKIPEDIE. *Disjunkce* [online]. 2009 , 10. 4. 2009 [cit. 2009-04-23]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Disjunkce>.

[8] WIKIPEDIE. *Negace* [online]. 2009 , 7. 4. 2009 [cit. 2009-04-23]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Negace>.

[9] WIKIPEDIE. *Implikace* [online]. 2009 , 3. 4. 2009 [cit. 2009-04-23]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Implikace>.

[10] WIKIPEDIE. *Ekvivalence* [online]. 2009 , 3. 2. 2009 [cit. 2009-04-23]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Ekvivalent>.

[11] RACLAVSKÝ, Jiří. *Predikátová logika* [online]. 1999-2004 , 9. 3. 2006 [cit. 2009-04-23]. Dostupný z WWW: <http://www.phil.muni.cz/fil/logika/pl.php>.

[12] RACLAVSKÝ, Jiří. *Výroková logika* [online]. 1999-2004 , 9. 3. 2006 [cit. 2009-04-23]. Dostupný z WWW: < http://www.phil.muni.cz/fil/logika/vl.php>.

[13] RNDr. Ruličová, Iva. *Relační algebra.* (přednáška) Pardubice: UNIVERZITA PARDUBICE, 8. 1. 2009.

9. Přiložené CD

Složka *test* obsahuje soubor *VyukovyProgram.exe*, který slouží pro instalaci výukového programu.

Složka *src* obsahuje vytvořenou aplikaci včetně zdrojových kódů.

Složka *doc* obsahuje textovou část práce ve formátech *doc* a *pdf*.