Lineární binární bezpečnostní kódy

Linear binary safety codes

Jakub Kupčík

Bakalářská práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub KUPČÍK**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Lineární binární bezpečnostní kódy**

Zásady pro vypracování:

1. Zpracujte literární rešerži na dané téma.
2. Seznamte se s existující elektronickou příručkou pro předmět Základy informatiky.
3. Doplňte elektronickou učební příručku o tematický okruh týkající se oblasti lineárních binárních bezpečnostních kódů. Zaměřte se na popis zabezpečení kódových slov pomocí různých lineárních binárních bezpečnostních kódů. Doplňte příručku o názorné řešené příklady z dané problematiky.
4. Na základě teorie naprogramujte algoritmus kódování a dekódování zejména Hammingových a Reed-Mullerových kódů. Tyto programy budou pracovat na rozhraní WWW.

Rozsah práce:

Rozsah přiloh:

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

Seznam odborné literatury:

Prokopová, Z.: Základy informatiky, Skriptum VUT Brno, Fakulta technologická, Brno, 1991.
FARANA, R.: Kapitoly ze základů informatiky. Skriptum VŠB-TU Ostrava, 2003. 108 s. ISBN 80-248-0265-1.
VLČEK, K.: Komprese a kódová zabezpečení v multimediálních komunikacích. Praha: BEN - technická literatura.
Kosek Jiří: PHP - Tvorba interaktivních internetových aplikací - podrobný průvodce. Praha, Grada, 1999.
Popelková Kateřina: Současná česká grafická úprava knih, možnosti propagace literatury, elektronická kniha. Bakalářská práce. FT UTB Zlín, 2001

Vedoucí bakalářské práce: **Ing. Bronislav Chramcov**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **13. února 2007**

Termín odevzdání bakalářské práce: **24. května 2007**

Ve Zlíně dne 13. února 2007

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Předkládaná bakalářská práce spadá tématicky do oblasti lineárních binárních bezpečnostních kódů a zároveň do multimediální podpory výuky. Konkrétně se práce věnuje zejména Hammingovým a Reed – Mullerovým kódům. Práce se nejprve zabývá popisem využití internetových pomůcek při výuce technických předmětů, dále pak obsahuje obecný popis lineárních binárních bezpečnostních kódů.

Podstatou práce je úprava a doplnění existující webové elektronické učební pomůcky. Hlavní přínos spočívá pak v návrhu a vytvoření programů v prostředí WebMathematica, které mohou studenti využívat on-line. Tyto programy kódují popřípadě dekódují zadaná kódová slova podle zvolené metody bezpečnostního kódu.

Klíčová slova: lineární binární kódy, WebMathematica, kódování, dekódování, internet ve výuce, Hammingovy kódy, Reed – Mullerovy kódy

ABSTRACT

The bachelor thesis, that was tabled, falls thematical to the linear binary safety codes area and at the same time to the multimedia espousal of teaching. Concretely primary the composition attends to Hamming and Reed – Muller codes. First the composition describes the using of the internet assistant during the studying technical subjects and then includes contains common describe of the linear binary safety codes.

The main part of this work is the modification and the completion of an exist internet teaching assistant. The main contribution bears in the design and formation of programs in the facies WebMathematica, which are for the students on-line. These programs code eventually decode desired code words after choice method of safety code.

Keywords: linear binary code, WebMathematica, coding, decoding, internet in the teaching, Hamming codes, Reed – Muller codes.

Děkuji panu ing. Bronislavu Chramcovovi, Ph.D. za vedení, pomoc při tvorbě této práce, cenné připomínky a za poskytnutí potřebných materiálů.

Prohlašuji, že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

V dnešní době jsou komunikace a přenos informace tak důležité činnosti, že už bychom si bez nich ani nedokázali představit svůj běžný život. Vždyť všechny informace vířící kolem nás, ať už v podobě počítačových, mobilních sítí, televizního signálu a prostě přenosu všeho druhu, musí mít tvar použitelný pro přenos nejrůznějšími informačními kanály, a právě k tomu slouží kódování. Na škodu také není použít takových kódů, které nám umožní detekovat, případně i opravovat možné chyby. Každá přijatá informace, která k nám doputuje přes informační kanál, je s určitou pravděpodobnostní zatížena chybou. Jednou z možností jak naše informace zakódovat je použití lineárních binárních bezpečnostních kódů. A právě o nich se dozvíte víc v následujících kapitolách.

Kromě toho zde také naleznete pár informací o využití internetových pomůcek při výuce technických předmětů. Důvodem proč práce obsahuje i tuto kapitolu je využití prostředí WebMathematica v praktické části. Dozvíte se zde o možnostech, které máme při tvorbě praktických ukázkových programů, pomocí nichž se student mnohdy naučí mnohem víc než při čtení několika stránek textu teorie. Zjistíte, že v tomto prostředí lze vytvářet aplikace, které si student může vyzkoušet bez speciálního softwaru on-line a především i bez zvláštních znalostí problematiky. Stačí mu v pohodlí domova navštívit stránky daného předmětu a nenásilnou formou se mnohé přiučit. Koho taková možnost výuky zajímá, najde v praktické části této práce ukázku takto vytvořených aplikací a jejich popis. Všechno se samozřejmě týká oblasti LINEÁRNÍCH BINÁRNÍCH BEZPEČNOSTNÍCH KÓDŮ.

I. TEORETICKÁ ČÁST

1 VYUŽITÍ INTERNETU VE VÝUCE

Internet je beze sporu fenoménem naší doby a postupem času a díky zdokonalovaní technologií se objevují stále nové a nové možnosti jeho využití. Ne jinak je tomu také ve školách. Internet se stal nezbytnou součástí studentů na všech vysokých školách a postupně se probojovává i do výuky a života studentů středoškolských. Z počátku se uplatňoval především při hledání informací, dnes už je jeho role podstatně někde jinde. Ať už jde o přihlašování na zkoušky nebo získávání aktualit ze školy, ale především se internet stává součástí výuky vlastní.

Typickým příkladem je výuka cizích jazyků, kdy student nemusí tahat do školy těžký slovník, protože využije některý z internetových, může zkoušet nejrůznější testy přímo na webu nebo třeba internetové výukové lekce. Internet ale nemusí sloužit ve výuce pouze studentovi, může také velmi zjednodušit práci učitele. Většina škol provozuje své vlastní webové stránky. Avšak také velká část učitelů dnes tvoří pro svůj předmět internetové stránky. Nejenže sem může vkládat cenné informace pro studenty, ale také odevzdávání úkolů pomocí webových formulářů je velkou výhodou a to jak pro studenty, tak pro učitele.

1.1 Počítačové algebraické systémy

Zaměřme se ale na využití internetu ve výuce technických předmětů, kterých se bude následující text především týkat. Díky tomu že dnes již internetové stránky nejsou pouze statické je možné stále větší využití pro praktické ukázky řešení nejrůznějších problémů. Například dnes není problém aby se student připojil přes web na přístroj v laboratoři a změřil pomocí něj hodnoty do svého protokolu. Velkou výhodou jsou však také Počítačové algebraické systémy (PAS), které umožňují velké usnadnění práce v technických a matematických oborech. Příkladem jsou AXIOM, MAPLE, MATHEMATICA, MATLAB, MATHCAD.. Problematika počítačových algebraických systému je složitá v tom, že je mnoho systémů a dají se velmi špatně mezi sebou porovnávat. Pro zvládnutí každého systému je potřeba dost času a zjistíte, že se systémem umíte toho čím dál více jen pokud s ním pravidelně a soustavně pracujete. Pokud zvládáte dobře jeden systém, není problém se naučit základy nějakého jiného, ale opravdu umět plně využít možnosti systémů jako je Mathematica a Matlab není jednoduché a možná ani v možnostech jediného člověka. Prakticky většinou školy využívají toho softwaru, na který získají nejvýhodněji licence.

Všechny tyto programy nacházejí využití především tam, kde je potřeba řešení složitějších matematických výpočtů a především jejich následné grafické zobrazení.

Obrovským přínosem těchto aplikací je především možnost propojit je právě s internetem. Student tím získá možnost vyzkoušet si na vlastní kůži příklady probírané v hodinách. Webmasteři ovládající technologie Java Script, PHP, Java a podobně by pravděpodobně dokázali většinu z výše jmenovaných problémů zvládnout i bez PAS. Ale třeba jen maticové násobení není v PHP nebo Java Scriptu příliš jednoduché. Pomocí dvourozměrných polí se sice k výsledku dostaneme, ale program bude mít hodně řádků a slabší programátor si s tím třeba neporadí vůbec. Při použití PAS není problém násobit matice na jednom řádku. Na internetové stránky se umístí formuláře, které odešlou data do speciálních forem výše zmíněných programů. Ty jsou zpracovány na serveru, kde je program nainstalován a výsledky jsou odeslány zpět na stránku, která se zobrazí studentovi. Příklady těchto forem PAS jsou:

- **MapleNet**, který umožňuje používat výpočetní možnosti Maple "na dálku" pouze pomocí webového rozhraní. **Maple T.A.** je webový nástroj pro vytváření testů a příkladů, které mohou sloužit nejen ke zkoušení a hodnocení, ale i domácí práci studentů.
- **MATLAB Web Server** je nástroj, který umožňuje vytváření aplikací prezentací

v Matlabu. Vstupní parametry simulace nebo regulační smyčky se zadávají prostřednictvím internetu pomocí HTML stránek, přičemž vlastní vypočet, resp. regulační proces, běží na vzdáleném počítači = serveru.

1.2 WebMathematica

Další z forem PAS. Jelikož byla WebMathematica použita v praktické části bakalářské práce, podívejme se na ni podrobněji. WebMathematica je jednoduchý způsob jak vytvářet interaktivní výpočty na webu. Tato technologie umožňuje vytvářet webové stránky, kde jsou využity výpočetní, vizualizační schopnosti Mathematicy v celé její šíři. Každému uživateli stačí standardní internetový prohlížeč (browser) pro zadávaní výpočtů i vizualizaci výsledků. WebMathematica a Mathematica jsou založené na stejné technologii, ale úplně jiném uživatelském rozhraní a jiné cílové skupině uživatelů. Webmathematica nabízí přístup do Mathematicy přes browser. Používá vše z Mathematicy.

WebMathematica dává známé webové prostředí, které potřebuje ale trochu tréninku, aby se dalo efektivně používat. Uživatelé nemusí znát prostředí Mathematici. Mathematica je jako vývojové prostředí pro webMathematicu. Například, analytici mohou v Mathematice vytvářet modely a inženýři si je mohou spouštět přes webMathematicu.

WebMathematica je založena na dvou standardních Java technologiích:

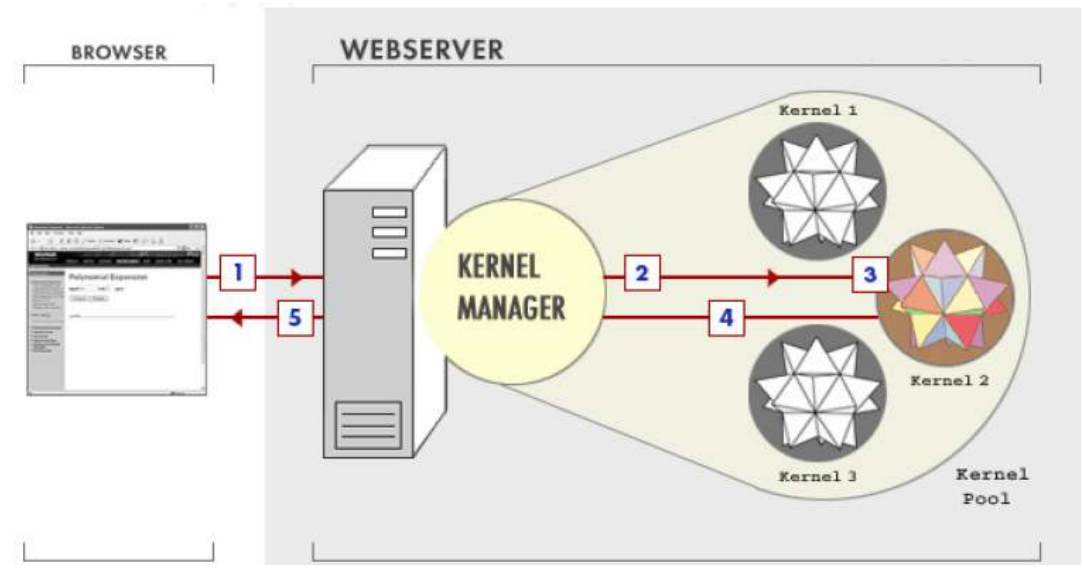
- Java Servlet

Servlety jsou speciální programy v Javě, které se spouštějí přes webový server, obvykle se nazývají"servlet container" někdy "servlet engine".

- Java Server Pages (JSP)

Při použití JSP se podobně jako v ASP nebo v PHP přímo do HTML kódu zapisují příkazy. Ty jsou nyní zapisovány v jazyce Java. Speciální servlet se stará o to, aby byla JSP stránka vždy po své modifikaci automaticky přeložena do byte-code.

Zpracování požadavku Webmathematicou:



Obr. 1. Postup zpracování požadavku WebMathematicou

1. webový prohlížeč posílá požadavek na webMathematica server
2. webMathematica server si rezervuje Mathematica kernel
3. Mathematica kernel je nastaven, provede kalkulace a vrací výsledky
4. webMathematica server vrací Mathematica kernel do „bazénu“
5. webMathematica server vrací výsledky webovému prohlížeči

1.2.1 Výroba vlastních MSP (Mathematica Server Pages)

- Statické stránky

1. výroba notebooku v Mathematice

a) nahrání balíčku

b) deklarace proměnných

c) funkce a hlavní kód

2. vytvořeníHTML stránky, která bude obsahovat potřebný text (bez Mathematica tagů)

3. uložit soubor jako JSP do adresáře kde je nainstalovaná webMathematica

4. přidání webMathematica tagů

5. přidaní kódu v Mathematice mezi webMathematica tagy

6. přidání specifických příkazů do kódu v Mathematice

7. otestování stránky v prohlížeči

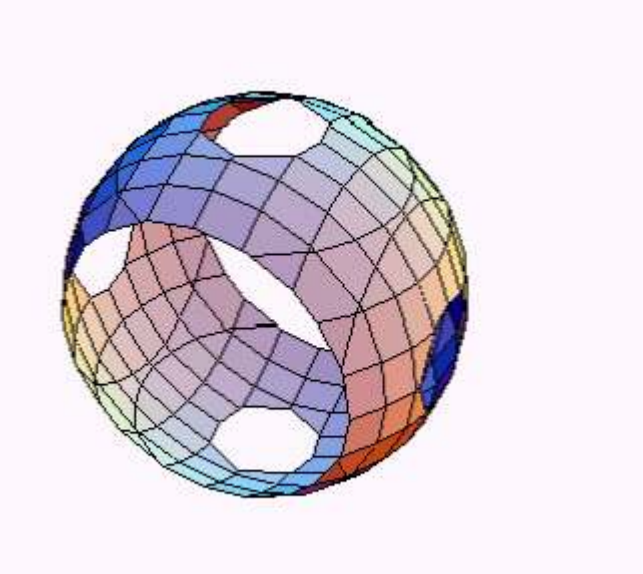
Příklad kódu:

```
<msp:evaluate>
    Get["Graphics`ContourPlot3D`"];
</msp:evaluate>

<msp:evaluate>
    xmin=-3; xmax=3;
    ymin=-2; ymax=2;
    zmin=-3; zmax=3;
</msp:evaluate>

<msp:evaluate>
    ContourPlot3D[Sin[Sqrt[x^2+y^2 + z^2]],
    {x, xmin, xmax}, {y, ymin, ymax}, {z, zmin, zmax},
```

Boxed-> False]

</msp:evaluate>

Výsledek:



Obr. 2. Výsledek statického kódu

[3]

- Dynamické stránky

Tvorba dynamické stránky probíhá obdobně jako tvorba stránky dynamické. Rozdíl je v tom, že uživatel může měnit hodnoty proměnných pomocí webových formulářů. Ty pak například ovlivní vlastnosti zobrazovaných grafů, proto tedy dynamické stránky. Jediným problém je správné předávání hodnot z formulářů do proměnných kódu Mathematicy.

Například:

Formulář: <input type = “text” name = “x”>

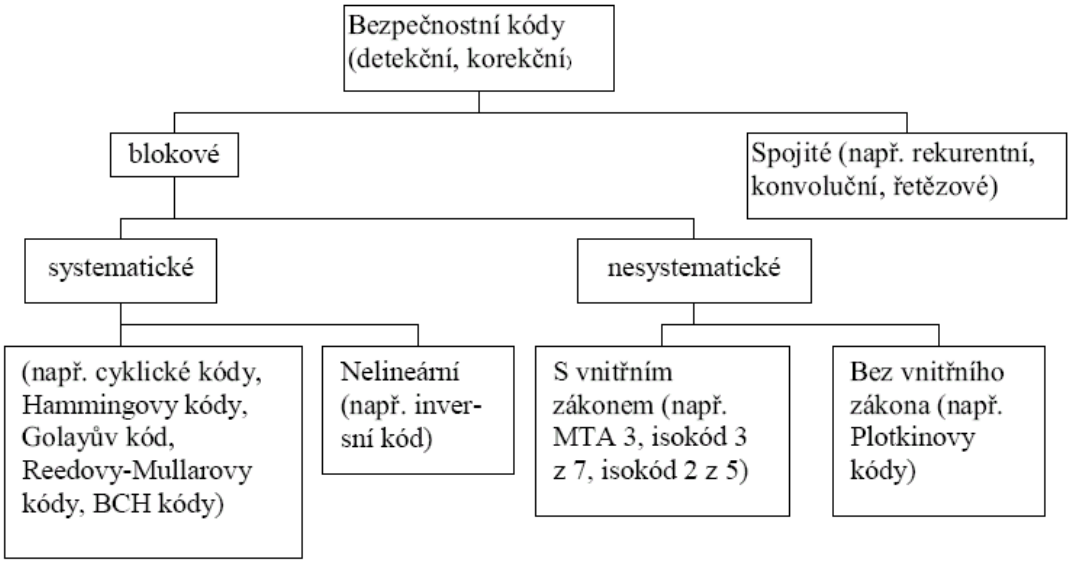
V Mathematice bude hodnota proměnné: $$x

2 KÓDOVÁNÍ

Kód je množina symbolů , kterými vyjadřujeme jednotlivé stavy systému. Množina symbolů může být vyjádřena buď pomocí tabulky nebo dohodnutým systémem pravidel (algoritmem). Mějme např. systém, který je určen polohou hrací kostky (krychle). Poloha kostky je nejčastěji vyjadřována pomocí množiny tečkových symbolů. Padne – li určitý počet teček, pak si jistě tento stav nepoznačíme v tečkovém kódu, ale pomocí arabských číslic 1 až 6, které tvoří jinou množinu symbolů pro vyjádření stavu systému (kostky). Jiným příkladem množiny symbolů jsou např. různé jednoduché obrázky, které umožní hrát dětem předškolního věku „Člověče, nezlob se“, aniž by uměli počítat, neboť mohou posouvat své figurky na odpovídající obrázky hrací desky. Kódování je činnost, během které převádíme jeden kód na druhý, a to buď pomocí tabulky nebo vhodným algoritmem. Kód lze libovolně převádět jeden na druhý, aniž by se změnila velikost informace. Při přechodu z jednoho kódu na druhý se mění množství znaků (nemusí být vždy stejné) a mění se i pravděpodobnost výskytu jednotlivých symbolů, přičemž dochází ke změně redundance. Lze najít takový kód, který bude mít největší entropii na symbol, tj. nejmenší počet symbolů na dané množství informace. Takový kód bude nejvýhodnější z hlediska ideálního přenosu (bez rušení), neboť bude vyžadovat nejmenší počet symbolů a nejkratší dobu přenosu. Protože reálné kanály jsou vždy s rušením, je nutné používat kódy, u kterých se zavádí záměrně redundance umožňující detekci, případně korekci chyb.

Rozdělení kódů:

Kódy dělíme podle různých hledisek. Např. kódy můžeme rozdělit na rovnoměrné a nerovnoměrné. Rovnoměrné kódy mají všechny znaky (symboly) tvořené stejným počtem prvků. Nerovnoměrné naopak různým. Význam v souvislosti s přenosem informace mají především kódy bezpečnostní, které se dělí na detekční a korekční (např. cyklický kód, Hammingův kód, ..).[1]



Obr. 3. Rozdělení bezpečnostních kódů

2.1 Lineární binární kódy

Patří mezi kódy systematické. Binární kód je lineární, jestliže výsledkem součtu dvou kódových slov je opět kódové slovo. Při použití systematického binárního blokového kódu je možné rozlišit, která část je původním informačním slovem (informační bity) a která část jsou přidané bity z důvodu kontroly nebo zabezpečení (kontrolní nebo zabezpečovací bity). Způsob vytváření slov je možné vyjádřit následovně: informační slovo o délce k bitů $u_1u_2...u_k$ je kódováním přeměněno na kódové slovo o délce $u_1u_2...u_ku_{k+1}u_{k+2}...u_n$.

Předpis pro kódování může být vždy popsán jako soustava rovnic, ve kterých jsou použity operace sčítání a násobení binárních čísel. Pro sčítání binárních čísel zde platí: 1 + 1 = 0. Lineární kódy se vyznačují také tím, že libovolná lineární kombinace kódových slov je opět kódovým slovem.

Generující matice kódu [*G*] je tvořena kódovými slovy tzv. báze kódu (jsou lineárně nezávislá). Chceme-li nalézt kódová slova lineárního binárního kódu, stačí když zjistíme jeho bázi. Ostatní slova lineárního kódu mohou být získána sčítáním libovolných dvou slov báze kódu. Matice G typu (k, n) je generující maticí lineárního kódu jestliže:

- každý její řádek je kódovým slovem
- každé kódové slovo je lineární kombinací řádků
- řádky jsou lineárně nezávislé

Báze je tvořena všemi kombinacemi kódových slov které jsou lineárně nezávislá. (Mohou být sečtena i 2 stejná slova, výsledek je však vždy nulové kódové slovo. To tedy vždy patří do lineárního kódu, ze stejného důvodu se však nemůže být obsaženo v generující matici.) Zabezpečující bitová slova vytváříme podle algoritmu daného kódování. Seřadíme li takto vytvořená kódová slova do řádků a uzavřeme je do matice, získáme tzv. *generující matici systematického kódu:*

$$G = [E|B] \tag{1}$$

Součinem informačního slova s generující maticí získáme kódové slovo. Tento postup se nazývá kódování nebo zakódování.

Opačný postup nazýváme dekódování. Dochází k němu po přijetí kódového slova přes informační kanál. K tomu využijeme **kontrolní matice**, kterou násobíme přijaté slovo. Pokud nedošlo při přenosu k chybám, měl by výsledek vypadat následovně:

$$H\begin{bmatrix} v_1 \\ v_2 \\ .. \\ .. \\ v_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ .. \\ .. \\ 0 \end{bmatrix} \tag{2}$$

Vztah mezi generující a kontrolní maticí:

$$H = \left[-B^T \middle| E'\right] \tag{3}$$

, kde E' je jednotková matice. [1]

2.1.1 Hammingovy kódy

Jeho kódová vzdálenost je d_{min} = 3. (Vznikají tedy výběrem jedné osminy slov z přirozeného binárního kódu.) Kód s minimální vzdáleností 3 detekuje dvojnásobnou chybu a opravuje jednonásobnou. Hammingovy kódy se snadno dekódují a jsou perfektní (mají nejmenší myslitelnou redundanci). Jestliže počet kontrolních bitů m roste po jedné, je celková délka Hammingových kódů

$$n = 2^m - 1 \tag{4}$$

, takže např. (3,2)-kód, (7,4)-kód, (15,11)-kód, ...

Tab. 1. Konfigurace Hammingových kódů

m	2	3	4	5	6	...
n	3	7	15	31	63	...
k	1	4	11	26	57	...

Pro kontrolní matici u Hammingova kódu platí:

- sloupce musí být nenulové (zaručuje detekci jednoduché chyby)
- součet dvou sloupců musí být nenulový (detekce dvojnásobné chyby)
- žádný sloupec se neopakuje

Příklad uspořádání pro (7,4)-kód:

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{5}$$

2.1.2 Rozšířený Hammingův kód

Jedná se o lineární binární kódy se schopností opravy chyb. Tyto bezpečnostní kódy již nemají nejmenší možnou redundanci = nejsou *perfektní*. **Hammingova vzdálenost je dána d_{min}= 4** (detekuje 3-násobnou chybu a opravuje jednu chybu)

Rozšířený Hammingův binární kód s ***m*** kontrolními znaky **(*m* = 3, 4, …)** má délku kódového slova 2^{m-1}. Takže vzniká kód ***K*(4,1), *K*(8,4), *K*(16,11)**, atd.

Tab. 2. Rozšířený Hammingův kód

m	3	4	4	6	7	...
n	4	8	16	32	64	...
k	1	4	11	26	57	...

Kontrolní matici tohoto kódu získáme z kontrolní matice Hammingova kódu tak, že ke každému řádku přidáme paritní bit sudé parity a do kontrolní matice doplníme řádek jedniček.

2.1.3 Reedovy – Mullerovy kódy

Byli objevené v roku 1954 a v porovnaní s BCH kódy mají slabší parametry . Jsou prakticky významné Například kód RM (1,5) použil kosmický koráb Mariner 9 při vysílání fotografií z Marsu. Jsou lineární to kódy, které mohou být dekódovány jednoduchou technikou majoritního rozhodování. Z těchto důvodů jsou Reedovy – Mullerovy kódy důležité i přesto, že někdy nesplňují nejmenší kódovou vzdálenost. Délka kódové kombinace je

$$m = 2^n \text{ [bit]}. \tag{6}$$

Počet nezabezpečených bitů které vstupují do kódovacího procesu je dán vztahem:

$$k = 1 + k_1 + k_2 + ... + k_z \tag{7}$$

, kde

$$k_1 = \binom{n}{1}, k_2 = \binom{n}{2}, ..., k_z = \binom{n}{z} \tag{8}$$

Minimální Hammingova vzdálenost RM kódu je

$$d_{min} = 2^{n-z}. \tag{9}$$

- $n \geq 3$ ... je libovolné celé kladné číslo

- z ... je tzv. ***řád kódu***. Platí nerovnost $z < n$

Kódování a dekódování těchto kódů bude ukázáno v praktické části, nyní si alespoň naznačme vytváření generující matice, protože RM kód je určen generující maticí $[G]_{R(z;\,n)}$. Například pro kódy s $n = 4$ bychom generující matici vytvářeli takto:

Tab. 3. Tvorba generující matice Reedova – Mullerova kódu

bod	Čísla sloupců ...	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	První řádek je tvořen $m = 2^n$ jedničkami	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	v_0
2	Následuje k_1= 4 řádků, kde do sloupců zapisujeme dvojkové vyjádření čísla sloupce - bráno odshora	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	v_1
		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	v_2
		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	v_3
		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	v_4
3	Následuje k_2 = 6 řádků, které vzniknou vynásobením všech možných dvojic řádků z bodu 2.	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	$v_1.v_2$
		0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	$v_1.v_3$
		0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	$v_1.v_4$
	$(a_1, a_2..., a_n) . (b_1, b_2,..., b_n) =$	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	$v_2.v_3$
	$= (a_1.b_1, a_2.b_2, ... ,a_n.b_n)$	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	$v_2.v_4$
		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	$v_3.v_4$
4	Následuje k = 4 řádků, které vzniknou vynásobením všech možných trojic z bodu 2.	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	$v_1.v_2.v_3$
		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	$v_1.v_2.v_4$
		0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	$v_1.v_3.v_4$
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	$v_2.v_3.v_4$

Bod 1, 2 tvoří matici kódu prvního řádu, tj. kód $R(1;4)$

Bod 1, 2, 3 tvoří matici kódu druhého řádu, tj. kód $R(2;4)$

Bod 1, 2, 3, 4 tvoří matici kódu třetího řádu, tj. kód $R(3;4)$.

[2]

2.1.4 Cyklické kódy

Cyklický kód můžeme definovat pomocí *generujícího polynomu* $g(z)$. Generující polynom cyklického kódu ***K(m,k)*** je polynom stupně

$$m - k = r \tag{10}$$

, který **musí dělit (být dělitelem) polynom (z^m-1).** Cyklickým posunem koeficientů generujícího polynomu vzniká **generující matice *G(z)***

$$G(z) = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{m-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{m-k-1} & g_{m-k} & 0 & \cdots & 0 \\ . & & & & & & & & \\ . & & & & & & & & \\ . & & & & & & & & \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & g_{m-k} \end{bmatrix} \tag{11}$$

Řádky tvoří polynomy:

g(z)

z.g(z)

.

.

$z^{k-1}.g(z)$

U cyklických kódů můžeme kontrolní matici nahradit kontrolním polynomem:

$$h(z) = \frac{z^m - 1}{g(z)} \tag{12}$$

Kontrolní matici cyklického kódu ***K(m,k)*** získáme cyklickými posuvy koeficientů kontrolního polynomu čteného od nejvyšší mocniny. Kontrolní matice má ***m-k*** řádků

$$H(z) = \begin{bmatrix} 0 & 0 & \dots & 0 & h_k & h_{k-1} & h_{k-2} & \dots & h_1 & h_0 \\ 0 & 0 & \dots & h_k & h_{k-1} & h_{k-2} & \dots & h_1 & h_0 & h \\ & & & & & & & & & \cdot \\ & & & & & & & & & \cdot \\ & & & & & & & & & \cdot \\ h_k & h_{k-1} & \dots & h_2 & h_1 & h_0 & \dots & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

2.1.5 BCH kódy

Jsou to obecné kódy s dobrými parametry. Je to nejdůležitější třída cyklických bezpečnostních kódů.

Mezi hlavní parametry patří:

- velká volitelnost parametrů
- jednoduchý vztah mezi počtem informačních znaků a počtem opravených chyb
- detailně vypracované dekódovací metody.

BCH kódy opravují dvojnásobné a vícenásobné chyby. V porovnání s REED-MULLERovými kódy mají BCH kódy o něco náročnější dekódování, ale za to mají lepší parametry.

2.1.6 Reed – Salomonovy kódy

Jsou speciálním případem BCH kódů. Mají největší myslitelnou vzdálenost a dají se pomocí nich sestrojit dobré binární kódy.

2.1.7 Golayův kód

Je to jediný perfektní kód pro trojnásobné opravy.

II. PRAKTICKÁ ČÁST

3 EXISTUJÍCÍ WEBOVÁ POMŮCKA PRO PŘEDMĚT ZÁKLADY INFORMATIKY A JEJÍ ÚPRAVA

K předmětu Základy informatiky byla vytvořena internetová učební pomůcka, která má za úkol usnadnit studium tohoto předmětu. Jedná se o www stránky doplněné o řešené příklady a ukázkové programy týkající se probírané látky.

3.1 Struktura stránek

Hlavní části stránek jsou:

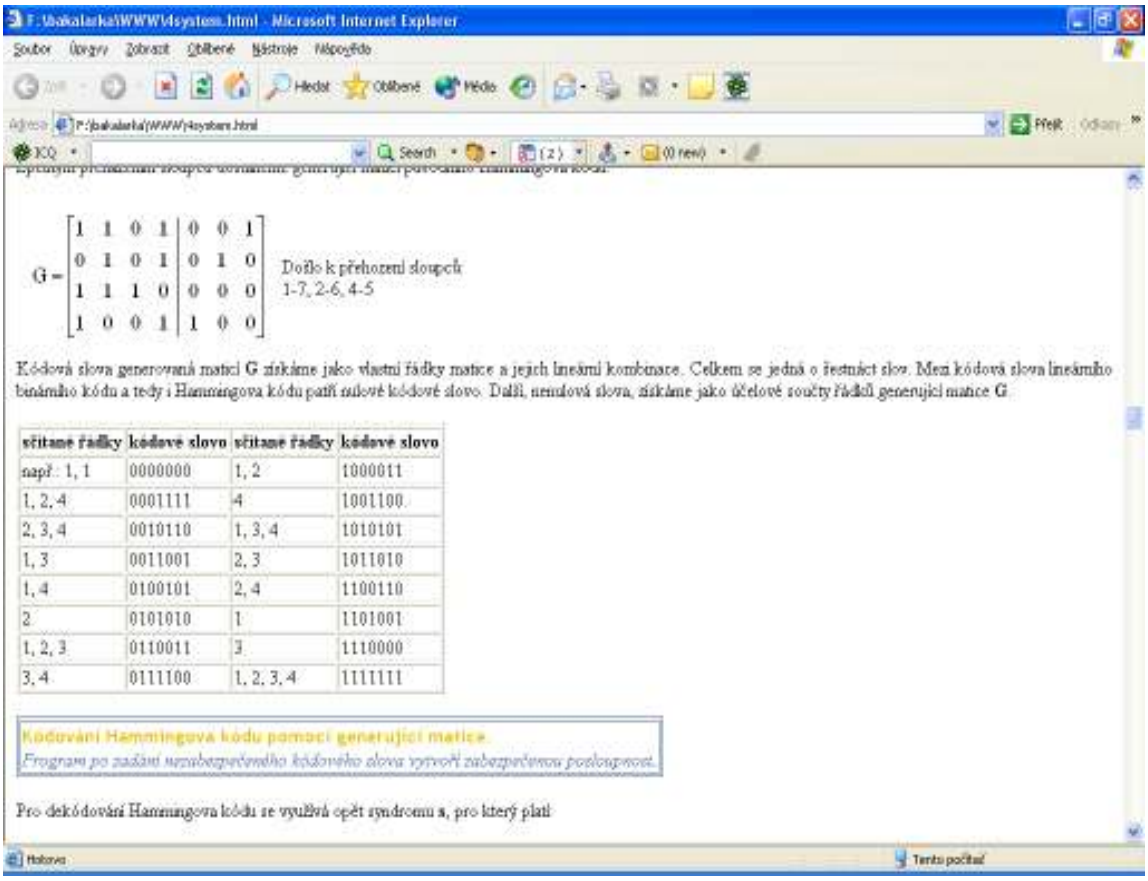
- Úvod
- Vznik a vývoj informace : vysvětlení pojmu informace, hledání její fyzikální kvantifikace, vznik teorie informace...
- Matematický aparát v teorii informace: základy pravděpodobnosti a teorie číselných soustav.
- Informace: základní pojmy, entropie, zdroje zpráv, přenos informace
- Kódování: elementární teorie kódování, rovnoměrné kódy, nerovnoměrné kódy, efektivní kódy
- Bezpečnostní kódy: systematické kódy (paritní, iterační, lineární, Hammingovy, rozšířené Hammingovy, Reedovy – Mullerovy, BCH, Reed – Solomonovy, nelineární, Golayův kód), nesystematické kódy
- Otázky ke zkoušce
- Přihlášení ke zkoušce: zde je odkaz na informační systém STAG, kde se studenti mohou přihlásit ke zkoušce
- Zadání seminární práce: studenti musí na konci semestru vypracovat seminární práci v prostředí Mathematica
- Přenášky: možnost stažení přednášek ve formátu Power Point
- Odkazy: odkazy na stránky s příbuznou tematikou

3.2 Prováděné úpravy

Mým úkolem bylo upravit kapitolu týkající se Lineárních binárních bezpečnostních kódů (konkrétně Hammingových a Reedových – Mullerových kódů) a především vytvoření programů pro kódování a dekódování zpráv prostřednictvím těchto kódů v prostředí WebMathematica.

Obr. 4. Základní vzhled webové pomůcky

V obsahu kapitol o Hammingových a rozšířených Hammingových kódech jsem prováděl pouze menší změny. Jednalo se zejména o doplnění vlastností těchto kódů, vytvoření tabulky s příklady kódových slov a celkovou kontrolu již vytvořeného textu. V kapitole o Hammingových kódech je nejprve obecný popis těchto kódů, dále je zde popsána tvorba kontrolní a generující matice a je zde prezentován na řešených příkladech postup kódování a dekódování. Tyto principy budou podrobněji popsány v textu o tvorbě vlastních programů. U rozšířeného Hammingova kódu je na příkladu naznačen rozdíl od předchozích kódů a taktéž popsán princip dekódování. Obě kapitoly obsahují odkazy na vytvořené programy ve WebMathematice.



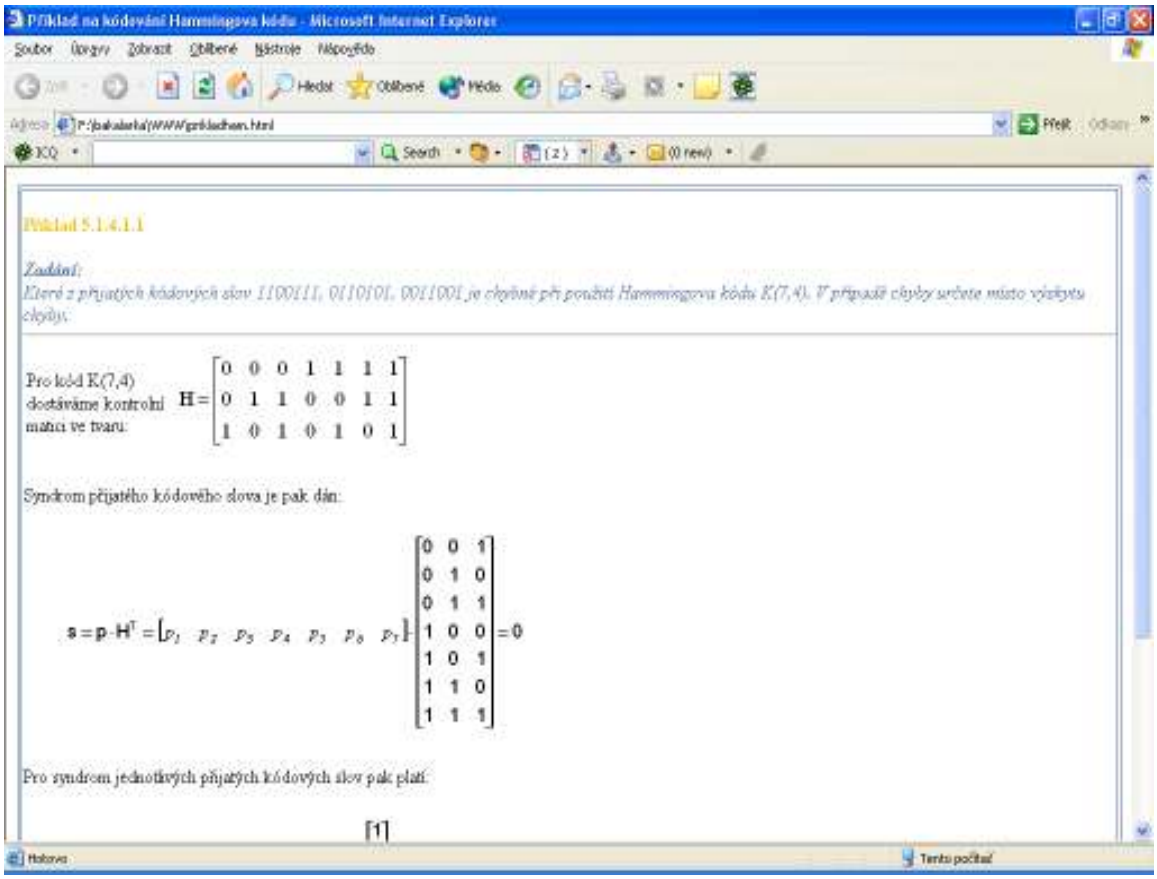
sčítané řádky	kódové slovo	sčítané řádky	kódové slovo
např. 1, 1	0000000	1, 2	1000011
1, 2, 4	0001111	4	1001100
2, 3, 4	0010110	1, 3, 4	1010101
1, 3	0011001	2, 3	1011010
1, 4	0100101	2, 4	1100110
2	0101010	1	1101001
1, 2, 3	0110011	3	1110000
3, 4	0111100	1, 2, 3, 4	1111111

Obr. 5. Ukázka změn v kapitole o Hammingových kódech

Kapitolu o Reedových – Mullerových kódech jsem vytvářel v podstatě celou znovu.

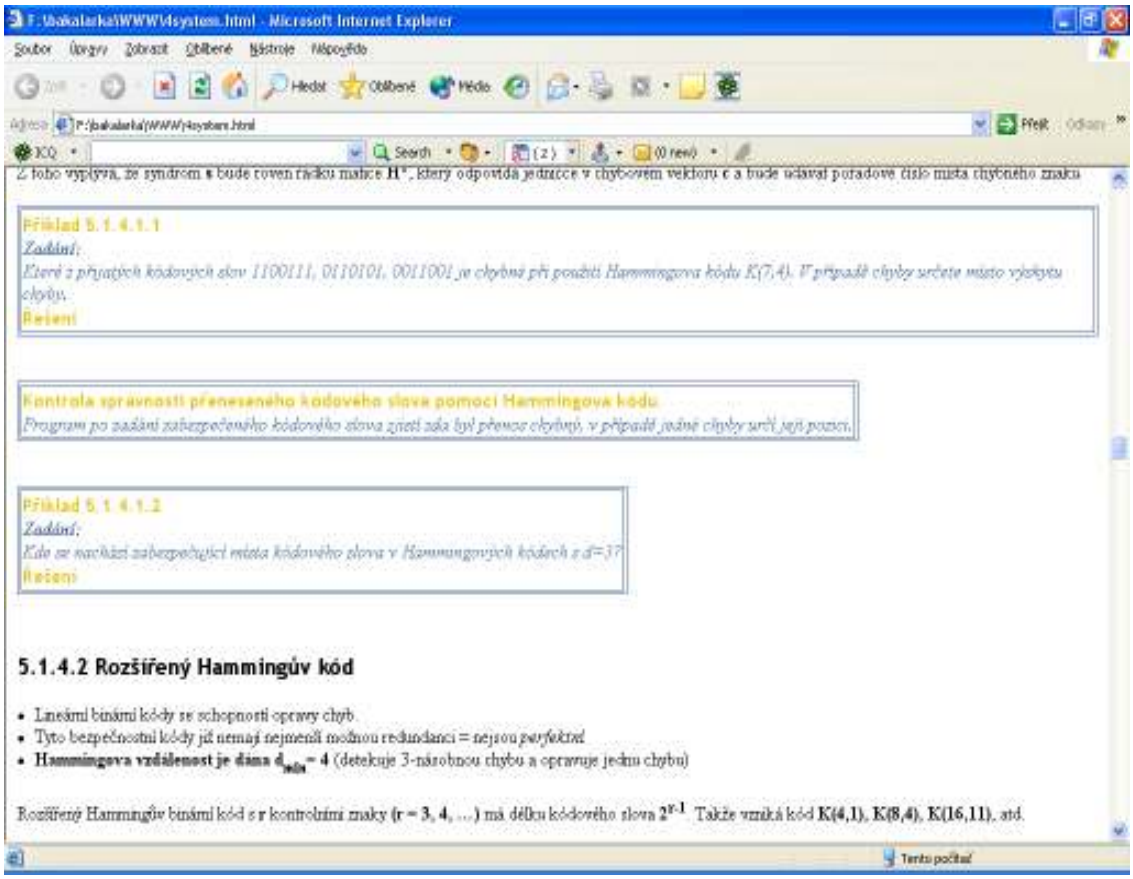
Na začátku jsou opět popsány základní vlastnosti tohoto kódu. Jelikož Reedových – Mullerových kódů je velké množství a volíme je v závislosti na požadavcích, je teorie kódování a dekódování vysvětlena na konkrétním příkladě, kde je naznačeno dekódování pomocí většinového rozhodování. U této kapitoly jsou opět odkazy na ukázkové on-line programy v prostředí WebMathematica.

Kromě obsahu těchto kapitol byl také upraven jejich vzhled kvůli zvýšení přehlednosti. Jedná se především o oddělení příkladů do samostatných stránek. V kapitole je pak odkaz zvýrazněn pomocí modrého rámečku a pro přehled je na tomto místě uvedeno i zadání příkladu napsané kurzívou a modrou barvou. Celkem jsem vytvořil pět nových stránek, které obsahují příklady. Také odkazy na programy ve WebMathematice jsou zvýrazněny modrým rámečkem a doplněny o stručný popis. Vzhled stránek samotných příkladů byl pak také vytvářen s ohledem na vzhled odkazů umístěných na hlavní stránce.



Obr. 6. Vzhled vytvořené stránky s příkladem

Vzhled a zvýraznění odkazů byl volen tak, aby příliš nenarušoval původní grafickou podobu stránek, ale na druhou stranu aby je student nepřehlédl při rychlejším procházení textu.



Obr. 7. Zvýraznění odkazů na příklady a vytvořené příklady

4 VYTVOŘENÍ PROGRAMŮ NA KÓDOVÁNÍ A DEKÓDOVÁNÍ V PROSTŘEDÍ WEBMATHEMATICA

Při práci s prostředím WebMathematica bylo nejproblematičtější najít správné postupy při spojování kódu programu Mathematica, webMathematica tagů a klasických html tagů. Objevovaly se různé problémy, například při práci se vstupy a výstupy. Díky jednoduchosti práce s maticemi v tomto prostředí nebyl větší problém naprogramovat samotné kódování a dekódování. Až na dekódování Reedova – Mullerova kódu šlo o práci se správnými generujícími a kontrolními maticemi. V následujícím textu jsou popsány principy a postupy jednotlivých problémů.

4.1 Kódování Hammingova kódu

Pro kódování a dekódování Hammingova kódu je důležité nejdříve správně uspořádat kontrolní a tím také generující matici.

Je proto výhodné uspořádat sloupce kontrolní matice ***H*** tak, aby v *i*-tém sloupci bylo dvojkové vyjádření čísla *i*. Binární kód se nazývá Hammingův jestliže má kontrolní matici, jejíž sloupce jsou všechna nenulová slova dané délky a žádné z nich se neopakuje.(používá se binární rozvoj čísel 1, 2, 3, …, 2^r-1)

Pro $r = 2$ dostáváme kód $K(3,1)$ s kontrolní matici ve tvaru:

$$H = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \tag{14}$$

Taková matice je kontrolní maticí opakovacího kódu délky 3. Ten skutečně opravuje jednoduché chyby

Pro $r = 3$ dostáváme kód $K(7,4)$ s kontrolní matici ve tvaru:

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{15}$$

Abychom našli generující matici, **musíme přemístit sloupce do tvaru:**

$$H' = \left[R^T \middle| E_r \right] \tag{16}$$

Tzn.

$$H' = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \tag{17}$$

Došlo k přehození sloupců: 1-7, 2-6, 4-5

Generující matici ***G'*** dostaneme ze vzorce:

$$G' = \left[E_k | R\right] \tag{18}$$

$$G' = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right] \tag{19}$$

Zpětným přeházením sloupců dostaneme generující matici původního Hammingova kódu:

$$G = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right] \tag{20}$$

Došlo k přehození sloupců: 1-7, 2-6, 4-5

Zabezpečené kódové slovo nyní získáme prostým vynásobením nezabezpečeného kódového slova generující maticí.

Tohoto postupu kódování jsem využil při tvorbě programu na kódování Hammingova kódu ve WebMathematice.

4.1.1 Popis programu

Po kliknutí na odkaz v příslušné kapitole se studentovi objeví webový formulář na zadávání nezabezpečeného kódového slova. Pro kód (3,1) jednomístné, pro (7,4) čtyřmístné a pro (15,11) jedenáctimístné. Po odeslání tohoto slova se na obrazovku vypíše zadané kódové slovo, příslušná generující matice a především výsledné zakódované slovo.

4.1.2 Postup programu

- Nejdříve převede zadané slovo na textový řetězec z důvodu zabezpečení správného přijetí slova (při práci s čísly docházelo ke špatnému přečtení kódových slov začínajících nulou, např. 0111 bylo čteno jako 111)
- Tento řetězec je naformátován do tvaru použitelného pro maticové násobení v programu Mathematica (např. 0111 na {0,1,1,1}).
- Následuje samotné vynásobení generující maticí odpovídající délce vstupu .
- Dále úprava výsledku (sudá čísla nahrazena 0, lichá 1).
- Pokud nebyl zadán správný tvar kódového slova (špatná délka nebo znaky), je vypsána chybová hláška, v opačném případě se vypíšou výsledky.
- Zdrojový kód je k nahlédnutí v příloze P I

Obr. 8. Ukázka práce programu na kódování Hammingova kódu

4.2 Dekódování Hammingova kódu

Pro dekódování Hammingova kódu se využívá syndromu ***s***, pro který platí:

$$s = p \cdot H^T = (z \oplus c) \cdot H^T = z \cdot H^T + c \cdot H^T = 0 + c \cdot H^T = c \cdot H^T \qquad (21)$$

- při jednonásobné chybě ***vektor chyb - c*** pouze jednu jedničku.

Z toho vyplývá, že syndrom ***s*** bude roven řádku matice $\boldsymbol{H^T}$, který odpovídá jedničce v chybovém vektoru ***c*** a bude udávat pořadové číslo místa chybného znaku.

Příklad:

Které z přijatých kódových slov 1100111, 0110101, 0011001 je chybné při použití Hammingova kódu K(7,4). V případě chyby určete místo výskytu chyby.

Pro kód *K*(7,4) dostáváme kontrolní matici ve tvaru rovnice (15).

Syndrom přijatého kódového slova je pak dán:

$$s = p \cdot H^T = [p_1 p_2 p_3 p_4 p_5 p_6 p_7] \cdot \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = 0 \qquad (22)$$

Pro syndrom jednotlivých přijatých kódových slov pak platí:

$$s(1100111) = [1100111] \cdot H^T = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad (23)$$

= Chybný přenos (poslední místo)

$$s(0110101) = [0110101] \cdot H^T = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \qquad (24)$$

= Chybný přenos (třetí místo)

$$s(0011001) = [0011001] \cdot H^T = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{25}$$

= Bezchybný přenos

4.2.1 Popis programu

Do úvodního formuláře tohoto programu student zadává přijaté kódové slovo, které chce zkontrolovat. Pro kód (3,1) trojmístné, pro (7,4) sedmimístné a pro (15,11) patnáctimístné. Po odeslání vstupních informací je vypsána použitá kontrolní matice příslušného kódu, vypočítaný syndrom a v případě chyby místo v kódovém slově, které je chybné.

4.2.2 Postup programu:

- Nejdříve převede zadané slovo na textový řetězec z důvodu zabezpečení správného přijetí slova (při práci s čísly docházelo ke špatnému přečtení kódových slov začínajících nulou, např. 0111 bylo čteno jako 111)
- Tento řetězec je naformátován do tvaru použitelného pro maticové násobení v programu Mathematica (např. 0111 na {0,1,1,1}).
- Následuje vynásobení kontrolní maticí .
- Dále úprava výsledného syndromu (sudá čísla nahrazena 0, lichá 1).
- Zjištěný syndrom je porovnán se sloupci kontrolní matice.
- Pokud bylo přijaté slovo správně zadané, dochází k výpisu výsledků (pokud má správnou délku a tvar).
- Pokud byl syndrom nulový, je vypsána hláška že přijaté slovo neobsahuje chyby. Pokud se syndrom shoduje s některým sloupcem, vypíše se zpráva, že přijaté slovo obsahuje chybu na příslušném místě. Pokud syndrom není nulový a přesto se neshoduje s žádným sloupcem kontrolní matice, znamená to že přijaté kódové slovo obsahuje více chyb a není jej proto možné opravit.
- zdrojový kód je k nahlédnutí v příloze P II

Obr. 9. Program na dekódování Hammingova kódu

4.3 Kódování rozšířeného Hammingova kódu

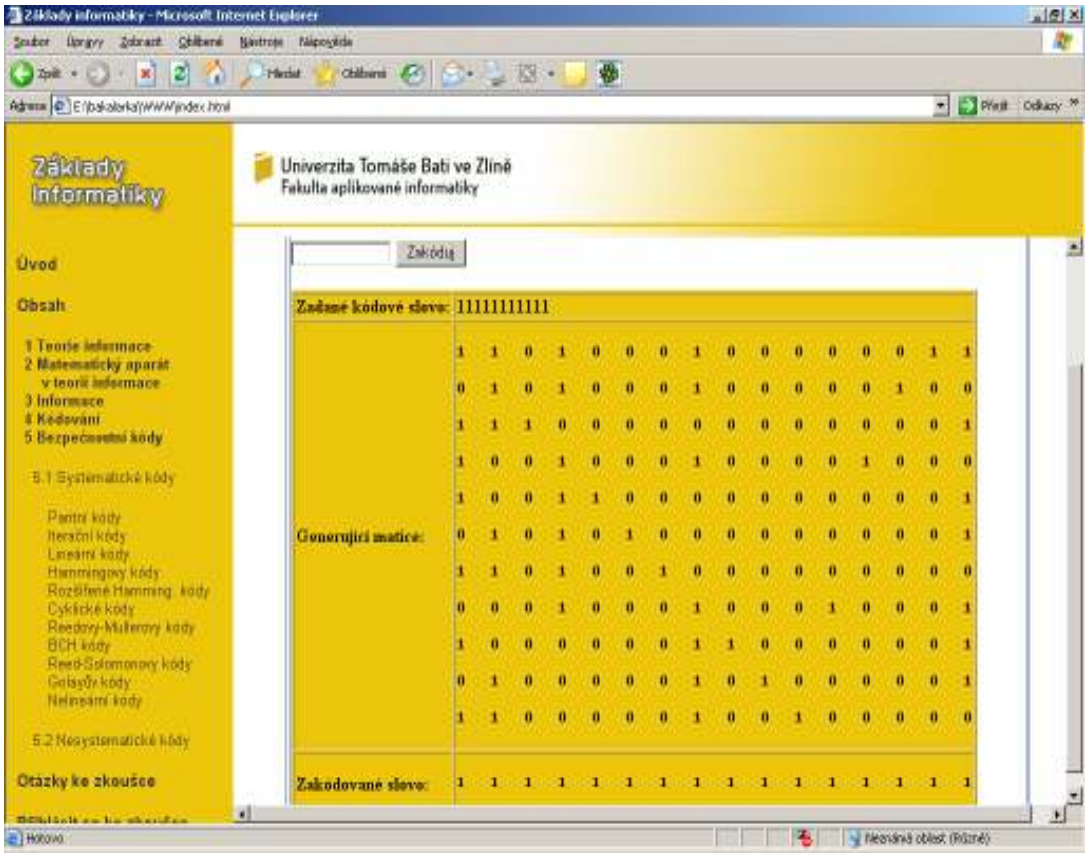
Kontrolní matici tohoto kódu získáme z kontrolní matice Hammingova kódu tak, že ke každému řádku přidáme paritní bit sudé parity a do kontrolní matice doplníme řádek jedniček. Pro generující matici je pak doplněn sloupec tak aby počet jedniček v řádku byl sudý (sudá parita). Zabezpečené kódové slovo získáme opět vynásobením nezabezpečené posloupnosti generující maticí.

4.3.1 Popis programu

Student stejně jako u Hammingova kódu zadává do formuláře nezabezpečené kódové slovo. Pro kód (4,1) jednomístné, pro (8,4) čtyřmístné a pro (16,11) jedenáctimístné! Po odeslání dat program vypíše zadané slovo, zabezpečené slovo a generující matici která byla použita pro příslušný kód. Od programu na kódování Hammingova kódu se tento liší především právě generujícími maticemi, které jsou upraveny podle výše uvedeného textu.

4.3.2 Postup programu:

- Nejdříve převede zadané slovo na textový řetězec z důvodu zabezpečení správného přijetí slova (při práci s čísly docházelo ke špatnému přečtení kódových slov začínajících nulou, např. 0111 bylo čteno jako 111)
- Tento řetězec je naformátován do tvaru použitelného pro maticové násobení v programu Mathematica (např. 0111 na {0,1,1,1}).
- Následuje samotné vynásobení generující maticí . V kódu jsou uloženy generující matice pro 3 příslušné rozšířené Hammingovy kódy. Program tedy vybírá jednu z těchto matic.
- Dále úprava výsledku (sudá čísla nahrazena 0, lichá 1).
- Pokud nebyl zadán správný tvar kódového slova (špatná délka nebo znaky), je vypsána chybová hláška, v opačném případě se vypíšou výsledky.
- Zdrojový kód obsahuje příloha P III



Obr. 10. Kódování rozšířeného Hammingova kódu

4.4 Dekódování rozšířeného Hammingova kódu

Dekódování rozšířeného Hammingova kódu se provádí stejným způsobem jako u Hammingova kódu pomocí syndromu ***s.*** Ten nás upozorní na případné chyby a jejich pozici. Postup je nejlépe vidět z příkladu.

Příklad:

Sestrojte kontrolní matici rozšířeného Hammingova kódu K(8,4). Které z přijatých kódových slov 11001011, 00000011, 00011110 je chybné? V případě jedné chyby určete místo výskytu chyby. Pokuste se sestrojit celý kód.

Pro kód $K(8,4)$ dostáváme kontrolní matici ve tvaru:

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{26}$$

Syndrom přijatého kódového slova je pak dán:

$$s = p \cdot H^T = \left[p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8\right] \cdot \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = 0 \tag{27}$$

Pro syndrom jednotlivých přijatých kódových slov pak platí:

$$s(11001011) = \left[11001011\right] \cdot H^T = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{28}$$

= Chybný přenos (první místo)

$$s(00000011) = \left[00000011\right] \cdot H^T = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{29}$$

= Chybný přenos s více jak jednou chybou

$$s(00011110) = \left[00011110\right] \cdot H^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{30}$$

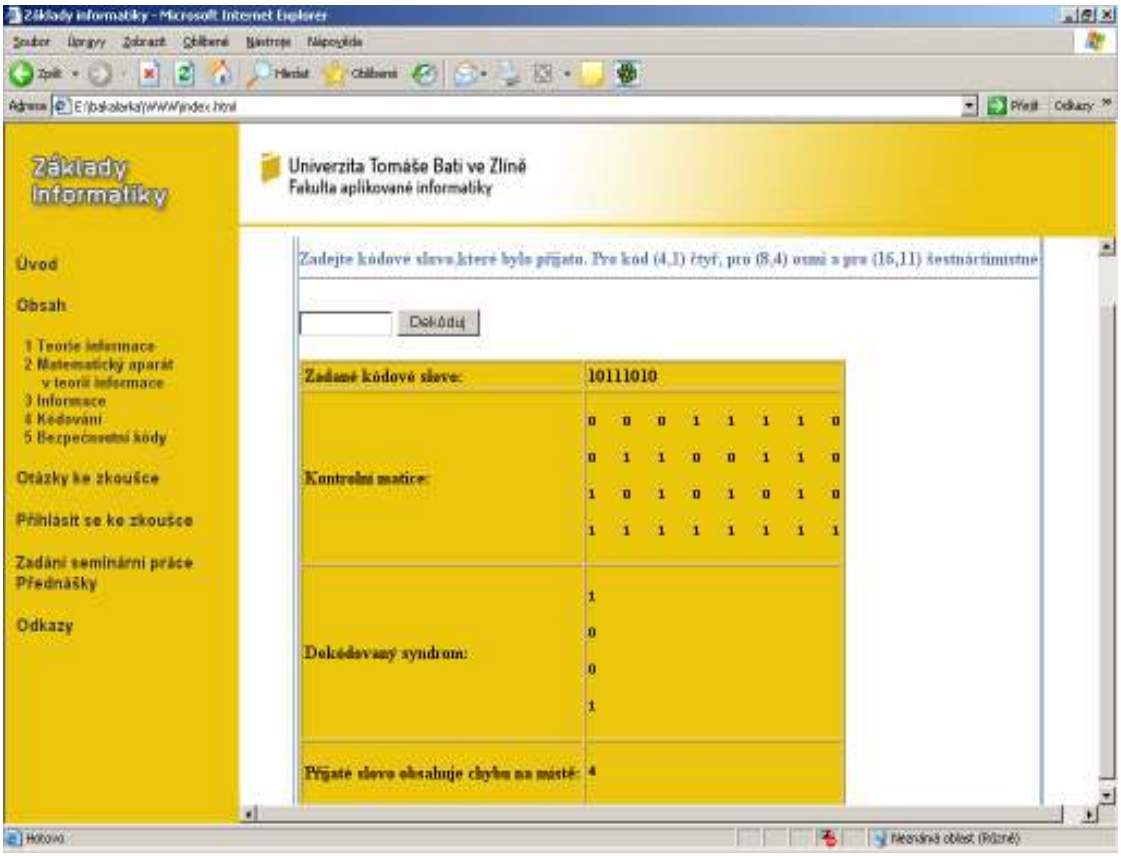
= Bezchybný přenos

4.4.1 Popis programu

Zadává se přijatá zabezpečená posloupnost u které chceme zkontrolovat správnost přenosu. Pro kód (4,1) slovo čtyřmístné, pro (8,4) osmimístné a pro (16,11) šestnáctimístné. Jako výsledek se zobrazí zadané slovo, použitá kontrolní matice, vypočítaný syndrom ***s*** a místo chyby, pokud slovo obsahovalo 1 chybu. Od programu na dekódování Hammingova kódu se tento liší především použitím jiných kontrolních matic.

4.4.2 Postup programu:

- Nejdříve převede zadané slovo na textový řetězec z důvodu zabezpečení správného přijetí slova (při práci s čísly docházelo ke špatnému přečtení kódových slov začínajících nulou, např. 0111 bylo čteno jako 111)
- Tento řetězec je naformátován do tvaru použitelného pro maticové násobení v programu Mathematica (např. 0111 na {0,1,1,1}).
- Následuje vynásobení kontrolní maticí . Pro příslušné kódy jsou v programu uloženy kontrolní matice. Program vybere tu správnou pro výpočet syndromu.
- Dále úprava výsledného syndromu (sudá čísla nahrazena 0, lichá 1).
- Zjištěný syndrom je porovnán se sloupci kontrolní matice.
- Pokud bylo přijaté slovo správně zadané, dochází k výpisu výsledků (pokud má správnou délku a tvar).
- Pokud byl syndrom nulový, je vypsána hláška že přijaté slovo neobsahuje chyby. Jestliže se syndrom shoduje s některým sloupcem, vypíše se zpráva, že přijaté slovo obsahuje chybu na příslušném místě. Pokud syndrom není nulový a přesto se neshoduje s žádným sloupcem kontrolní matice, znamená to že přijaté kódové slovo obsahuje více chyb a není jej proto možné opravit.
- Zdrojový kód je opět k dispozici v příloze P IV.



Obr. 11. Dekódování rozšířeného Hammingova kódu

4.5 Kódování Reedova – Mullerova kódu

V teoretické části byl popsán postup tvorby generující matice. Bity zabezpečené posloupnosti pak můžeme získat maticovým násobením nezabezpečeného kódového slova příslušnou generující maticí nebo pomocí vytvářecích rovnic které lze z generující matice odvodit.

Příklad:

Vytvořte soustavu rovnic pro výpočet bitů posloupnosti [F] kódu R(2;4).

Generující matice má tvar:

$$G=\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \rightarrow & p_0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \rightarrow & p_1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \rightarrow & p_2 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \rightarrow & p_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \rightarrow & p_4 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \rightarrow & p_5 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \rightarrow & p_6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \rightarrow & p_7 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \rightarrow & p_8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \rightarrow & p_9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \rightarrow & p_{10} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \\
f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} & f_{11} & f_{12} & f_{13} & f_{14} & f_{15} & &
\end{bmatrix} \tag{31}$$

Z ní získáváme rovnice (*f* odpovídá součtu těch *p*, pro které je v daném sloupci 1):

$$f_0 = p_0 \tag{32}$$

$$f_1 = p_0 \oplus p_1 \tag{33}$$

$$f_2 = p_0 \oplus p_2 \tag{34}$$

$$f_3 = p_0 \oplus p_1 \oplus p_2 \oplus p_5 \tag{35}$$

$$f_4 = p_0 \oplus p_3 \tag{36}$$

$$f_5 = p_0 \oplus p_1 \oplus p_3 \oplus p_6 \tag{37}$$

$$f_6 = p_0 \oplus p_2 \oplus p_3 \oplus p_8 \tag{38}$$

$$f_7 = p_0 \oplus p_1 \oplus p_2 \oplus p_3 \oplus p_6 \oplus p_8 \tag{39}$$

$$f_8 = p_0 \oplus p_4 \tag{40}$$

$$f_9 = p_0 \oplus p_1 \oplus p_4 \oplus p_7 \tag{41}$$

$$f_{10} = p_0 \oplus p_1 \oplus p_2 \oplus p_5 \tag{42}$$

$$f_{11} = p_0 \oplus p_1 \oplus p_2 \oplus p_4 \oplus p_5 \oplus p_7 \oplus p_9 \tag{43}$$

$$f_{12} = p_0 \oplus p_3 \oplus p_4 \oplus p_{10} \tag{44}$$

$$f_{13} = p_0 \oplus p_1 \oplus p_3 \oplus p_4 \oplus p_6 \oplus p_7 \oplus p_{10} \tag{45}$$

$$f_{14} = p_0 \oplus p_2 \oplus p_3 \oplus p_4 \oplus p_{10} \tag{46}$$

$$f_{15} = p_0 \oplus p_1 \oplus p_2 \oplus p_3 \oplus p_4 \oplus p_5 \oplus p_6 \oplus p_7 \oplus p_8 \oplus p_9 \oplus p_{10} \tag{47}$$

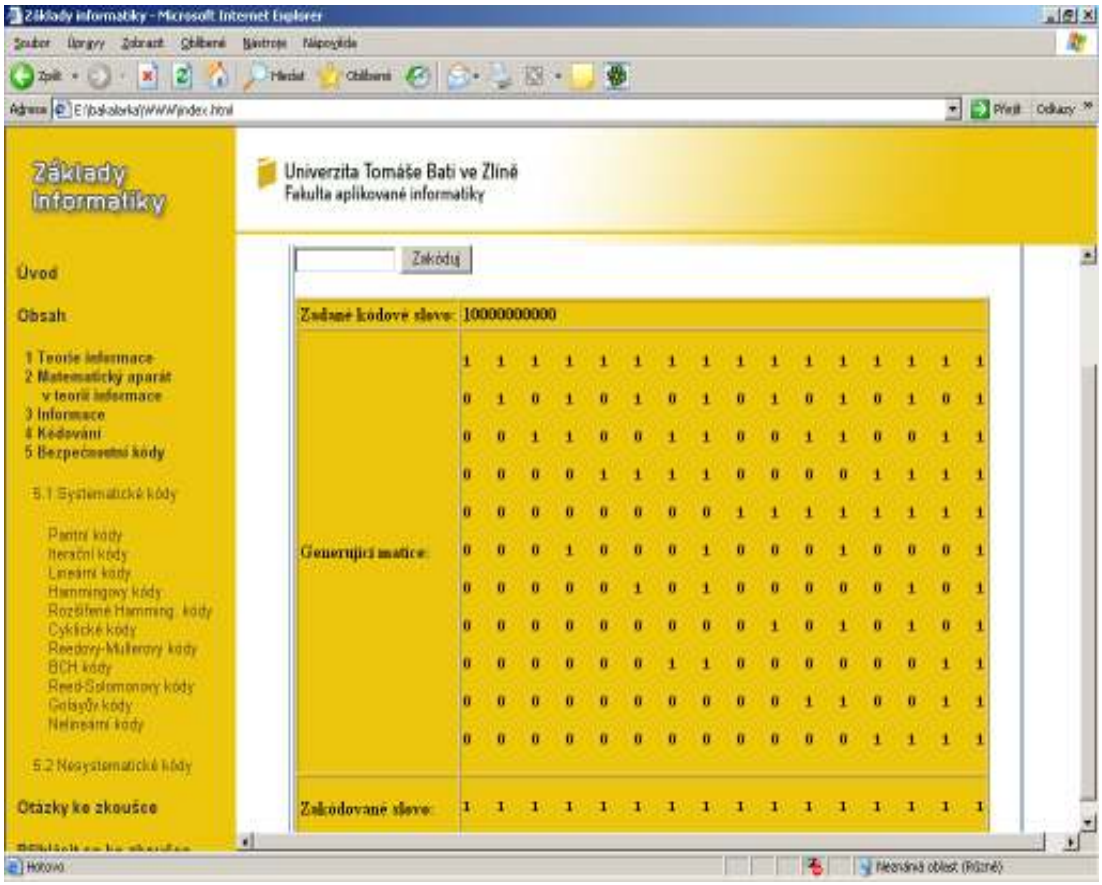
Pomocí nich dostaneme zabezpečenou posloupnost *F*.

4.5.1 Popis programu

Jelikož teorie je na webových stránkách vysvětlována na kódech s $n = 4$ (především $R(2;4)$), je i program vytvořen pro praktickou ukázku těchto kódů. Student zadává nezabezpečenou posloupnost, pro kód $R(1;4)$ pětimístnou a pro $R(2;4)$ jedenáctimístnou. Po zakódování pomocí příslušné matice se vypíše zadané kódové slovo, vybraná generující matice a především zabezpečená posloupnost.

4.5.2 Postup programu

- Nejdříve převede zadané slovo získané z webového formuláře na textový řetězec z důvodu zabezpečení správného přijetí slova (při práci s čísly docházelo ke špatnému přečtení kódových slov začínajících nulou, např. 0111 bylo čteno jako 111)
- Tento řetězec je naformátován do tvaru použitelného pro maticové násobení v programu Mathematica (např. 0111 na {0,1,1,1}).
- Následuje výběr správné generující matice a samotné vynásobení.
- Dále úprava výsledku (sudá čísla nahrazena 0, lichá 1).
- Provádí se kontrola správného zadání na vstupu a pokud bylo zadané kódové slovo ve správném tvaru, vypíšou se výsled. V opačném případě se objeví chybová hláška upozorňující na problém.
- Zdrojový kód je k nahlédnutí v příloze P V.



Obr. 12. Kódování Reedova – Mullerova kódu

4.6 Dekódování Reedova – Mullerova kódu

Reedovy – Mullerovy kódy patří mezi kódy lineární. Mohli bychom tedy pro dekódování podobně jako u Hammingových kódů použít kontrolní matice a syndromu ***s***. U těchto kódů se ale používá spíše majoritního rozhodování při dekódování. Nezískáme tak konkrétní chybnou pozici v přijatém slově, ale získáme hledané nezabezpečené slovo bez chyb i přes špatný přenos. Princip tohoto způsobu je nejlépe pochopitelný u konkrétního příkladu který je použit také pro program.

Nejmenší Hammingova vzdálenost pro kód $R(2;4)$ je $d_{min} = 2^{4-2} = 4$. Pro každý nezabezpečený bit potřebujeme tedy nalézt čtyři nezávislé rovnice.

Jsou to:

$$p_{10} = f_0 \oplus f_4 \oplus f_8 \oplus f_{12} = f_1 \oplus f_5 \oplus f_9 \oplus f_{13} = \\ = f_2 \oplus f_6 \oplus f_{10} \oplus f_{14} = f_3 \oplus f_7 \oplus f_{11} \oplus f_{15} \tag{48}$$

$$p_9 = f_0 \oplus f_2 \oplus f_8 \oplus f_{10} = f_1 \oplus f_3 \oplus f_9 \oplus f_{11} = \\ = f_4 \oplus f_6 \oplus f_{12} \oplus f_{14} = f_5 \oplus f_7 \oplus f_{13} \oplus f_{15} \tag{49}$$

$$p_8 = f_0 \oplus f_2 \oplus f_4 \oplus f_6 = f_1 \oplus f_3 \oplus f_5 \oplus f_7 = \\ = f_8 \oplus f_{10} \oplus f_{12} \oplus f_{14} = f_9 \oplus f_{11} \oplus f_{13} \oplus f_{15} \tag{50}$$

$$p_7 = f_0 \oplus f_1 \oplus f_8 \oplus f_9 = f_2 \oplus f_3 \oplus f_{10} \oplus f_{11} = \\ = f_4 \oplus f_5 \oplus f_{12} \oplus f_{13} = f_6 \oplus f_7 \oplus f_{14} \oplus f_{15} \tag{51}$$

$$p_6 = f_0 \oplus f_1 \oplus f_4 \oplus f_5 = f_2 \oplus f_3 \oplus f_6 \oplus f_7 = \\ = f_8 \oplus f_9 \oplus f_{12} \oplus f_{13} = f_{10} \oplus f_{11} \oplus f_{14} \oplus f_{15} \tag{52}$$

$$p_5 = f_0 \oplus f_1 \oplus f_2 \oplus f_3 = f_4 \oplus f_5 \oplus f_6 \oplus f_7 = \\ f_8 \oplus f_9 \oplus f_{10} \oplus f_{11} = f_{12} \oplus f_{13} \oplus f_{14} \oplus f_{15} \tag{53}$$

$$p_4 = f_7 \oplus f_{15} = f_6 \oplus f_{14} = f_5 \oplus f_{13} = f_4 \oplus f_{12} \tag{54}$$

$$p_3 = f_0 \oplus f_4 = f_1 \oplus f_5 = f_2 \oplus f_6 = f_3 \oplus f_7 \tag{55}$$

$$p_2 = f_0 \oplus f_2 = f_1 \oplus f_3 = f_4 \oplus f_6 = f_5 \oplus f_7 \tag{56}$$

$$p_1 = f_0 \oplus f_1 = f_2 \oplus f_3 = f_4 \oplus f_5 = f_6 \oplus f_7 \tag{57}$$

O bitu p_0 se rozhodne jinak. Obsahuje-li přijatá kódová kombinace $s > d_{min}$ jedniček, bude tento bit 1. Obsahuje-li přijatá kódová kombinace $s < d_{min}$ jedniček, bude 0.

Bez problémů jsou realizovatelné součty pro bity p_{10} až p_5 , tedy ty, které odpovídají řádkům vyššího řádu (vznikly součinem řádků x_i . x_j nižšího řádu), protože ty vzájemně zapojují všechny přenesené bity f_i do kontrolních operací a chyba jednoho či dvou se v některé kontrolní operaci projeví (pro každou jedničku v řádku hledáme takový součet 4 sloupců, který ve výsledku dá jedničku pouze na její pozici). To neplatí pro bity p_4 až p_0 . Problém se řeší převedením přijaté posloupnosti do polohy "slučitelné" se zabezpečením kódem prvního řádu (poté hledáme podobným způsobem součet 2 sloupců pro zbylé bity). Dekódování tedy probíhá ve dvou etapách. Napřed se opraví část, která obsahuje bity p_{10} až p_5. Tato část je pak použita k přepočítání přijaté posloupnosti tak, aby v druhé etapě dekódování byly opravovány pouze bity p_4 až p_1 . Pro opravu p_0 se použije počet jedniček v přijaté posloupnosti tak, jak je naznačeno v předcházejícím textu.

Příklad:

Naznačte dekódování přijaté posloupnosti kódu R(2;4):

Tab. 4. Přijatá posloupnost

f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
1	1	1	1	1	0	0	0	0	1	1	0	1	1	0	0

Bit f_6 byl přijat chybně a my si na tomto příkladu ukážeme jak dojde k opravě při dekódování.

Vypočítají se hodnoty všech čtyř kontrolních rovnic, které byly získány z generující matice a hodnota, která se vyskytuje víckrát, je většinová.

První etapa:

Z rovnice (48):

$$\begin{aligned} p_{10} &= 1 \oplus 1 \oplus 0 \oplus 1 = 1 \\ &= 1 \oplus 0 \oplus 1 \oplus 1 = 1 \\ &= 1 \oplus 0 \oplus 1 \oplus 0 = 0 \\ &= 1 \oplus 0 \oplus 0 \oplus 0 = 1 \end{aligned}$$

většina je 1

Z rovnice (49):

$$\begin{aligned} p_{9} &= 1 \oplus 1 \oplus 0 \oplus 1 = 1 \\ &= 1 \oplus 1 \oplus 1 \oplus 0 = 1 \\ &= 1 \oplus 0 \oplus 1 \oplus 0 = 0 \\ &= 0 \oplus 0 \oplus 1 \oplus 0 = 1 \end{aligned}$$

většina je 1

Z rovnice (50):

$$\begin{aligned} p_{8} &= 1 \oplus 1 \oplus 1 \oplus 0 = 1 \\ &= 1 \oplus 1 \oplus 0 \oplus 0 = 0 \\ &= 0 \oplus 1 \oplus 1 \oplus 0 = 0 \\ &= 1 \oplus 0 \oplus 1 \oplus 0 = 0 \end{aligned}$$

většina je 0

Stejně postupujeme u bitů $p_7 - p_5$

Výsledkem první etapy je dekódování prvků:

Tab. 5. Dekódované bity p_5 - p_{10}

p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
					0	1	1	0	1	1

Druhá etapa:

Tab. 6. Přepočítání posloupnosti pro druhou etapu

	1	1	1	1	1	0	1	0	0	1	1	0	1	1	0	0	... přijatá posloupnost
⊕	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$p_5.(v_1.v_2) = 0$ ... protože $p_5 = 0$
⊕	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	$p_6.(v_1.v_3)$
⊕	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	$p_7.(v_1.v_4)$
⊕	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$p_8.(v_2.v_3) = 0$... protože $p_8 = 0$
⊕	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	$p_9.(v_2.v_4)$
⊕	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	$p_{10}.(v_3.v_4)$
	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	... přepočítaná posloupnost

Přepočítanou posloupnost nyní použijeme do rovnic pro p_4 ... p_1. Z rovnic vypočteme většinové výsledky stejně jako u horních šesti bitů. Konečný výsledek tedy má tvar:

Tab. 7. Výsledek dekódování

p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
1	0	0	0	1	0	1	1	0	1	1

Došlo tedy k získání správného výsledku i přesto že přijatý posloupnost byla chybná. Kód *R(2;4)* má $d_{min} = 4$, je tedy schopný opravit 1 chybu. Kdyby obsahovala chyby 2, výsledky

dekódovacích rovnic by nebyl jednoznačný a nebylo by tedy možné jednoznačně rozhodnout o výsledku.

Popis programu:

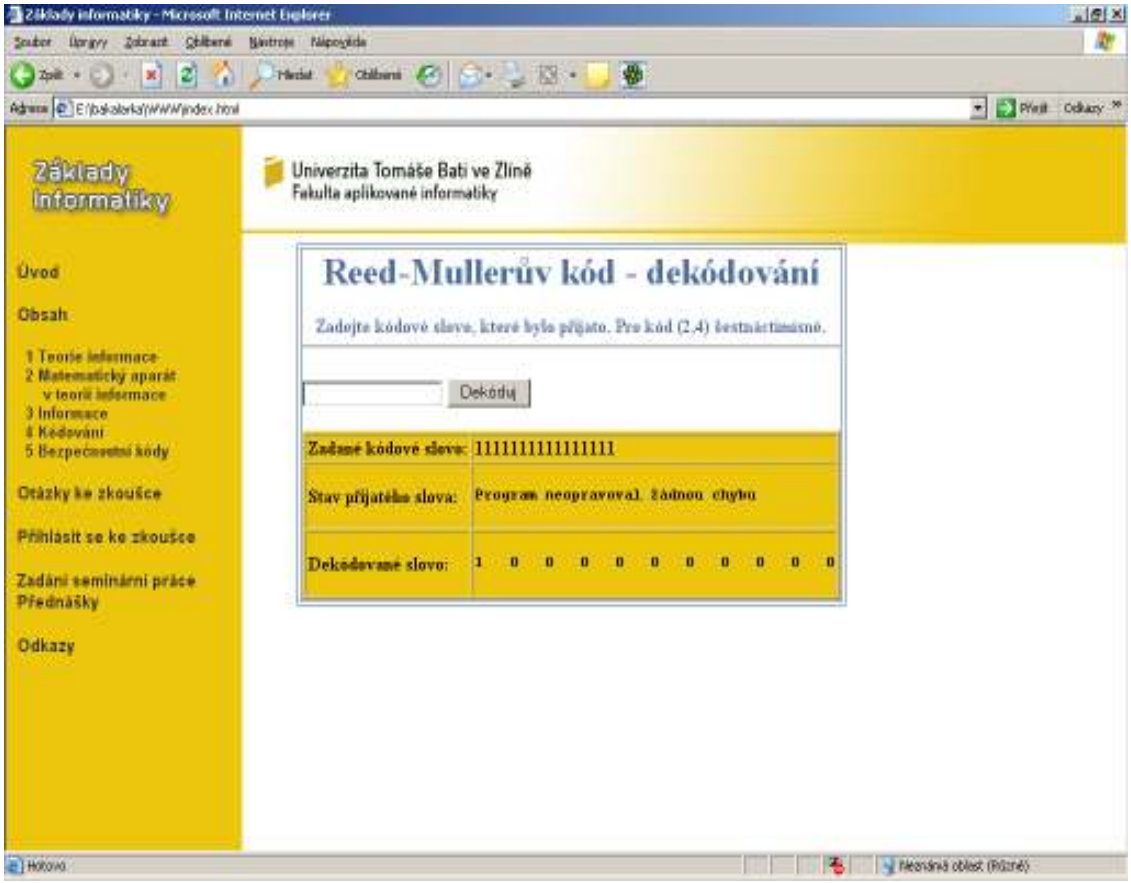
Oba předchozí programy na dekódování byly v podstatě podobné a lišili se především použitím kontrolních matic. Program na dekódování Reedova – Mullerova kódu využívá algoritmu majoritního dekódování, je tedy od základu jiný. Vzhledově je podobný. Student zadává šestnáctimístnou zabezpečenou posloupnost kódu *R(2;4)*. Program byl vytvořen pouze k tomuto kódu, protože na webových stránkách je teorie vysvětlena především na něm a algoritmus opravuje každý bit posloupnosti zvlášť, proto by bylo složité tvořit program pro víc kódů zároveň. Po odeslání formuláře se vypíše zadané slovo, dále hlášení jestli program prováděl opravu chyby a především výsledné kódové slovo. Pokud by přijatá posloupnost obsahovala 2 chyby, bylo by vypsáno, že jej nelze jednoznačně dekódovat.

Ve zdrojovém kódu lze najít na první pohled nesmyslné řádky. Jsou tam většinou kvůli prvnímu spuštění programu. Bez nich docházelo k nesmyslným výpisům.

Postup programu:

- Nejdříve převede zadané slovo získané z webového formuláře na textový řetězec z důvodu zabezpečení správného přijetí slova (při práci s čísly docházelo ke špatnému přečtení kódových slov začínajících nulou, např. 0111 bylo čteno jako 111). V programu je navíc ošetřen první výpočet po spuštění programu nastavením proměnných na počáteční hodnoty. Jinak program nepracoval správně.
- Tento řetězec je naformátován do tvaru použitelného pro práci s listy a maticemi v programu Mathematica (např. 0111 na {0,1,1,1}).
- Nejdříve se dekódují od bity $p_{10} - p_5$ od posledního. Pro každý bit se používají 4 odvozené dekódovací rovnice. Na základě nich (= většinové rozhodnutí) je určena výsledná hodnota bitu (výsledky rovnic se sečtou a pokud je výsledek menší než 2, bit bude 0, pokud je rovný 2 nelze rozhodnout a jsou zjištěny 2 chyby a pokud větší než 2 je bude bit 1. Takto rozhodneme o všech šesti bitech.
- Na základě získaných výsledků je přepočítána posloupnost pro kontrolu zbývajících bitů. Jedná se maticový součet a sudé výsledky jsou označeny za 0, liché za 1.

- Opět použijeme odvozené rovnice pro zbylé bity a postup je stejný jako u šesti horních bitů
- Nakonec rozhodneme o bitu p_0. Sečteme jedničky v přijaté posloupnosti a pokud je výsledek větší jak d_{min} , označíme bit za 1, jinak za 0.
- Dále program kontroluje zda byla posloupnost zadána ve správné délce a tvaru. Pokud ano, vypíše výsledek. Jestliže přijatá posloupnost obsahovala 1 chybu, program při dekódování chybu odstraní. Pokud byli chyby 2 program nemůže o bitech rozhodnou a vypíše chybovou hlášku. U třech chyb by program sice chybu detekoval a nahlásil pokus o opravu, nemohl by však slovo opravit správně (d_{min} = 4).
- Zdrojový kód je k nahlédnutí v příloze P VI.



Obr. 13. Dekódování Reedova – Mullerova kódu

ZÁVĚR

Podstatou předkládané bakalářské práce, která spadá tematicky do oblasti lineárních binárních bezpečnostních kódů a zároveň do oblasti multimediální podpory výuky, byla úprava a doplnění existující webové elektronické učební pomůcky.

Jedním z dílčích cílů bylo upravit a zejména doplnit kapitoly o Hammingových a Reedových – Mullerových kódech. Grafický vzhled programů, kapitol i ukázkových příkladů byl volen tak, aby se příliš nelišil od původního návrhu stránek. Podrobnější popis úprav, které byli provedeny, je náplní kapitoly 3.

Hlavním cílem práce bylo doplnit příručku o programy v prostředí WebMathematica. Během řešení bakalářské práce bylo třeba nastudovat základy využití prostředí WebMathematica, které je na Univerzitě Tomáše Bati ve Zlíně přístupné pro studenty.

Poté bylo vytvořeno celkem šest programů z oblasti kódování a dekódování zadaného kódového slova podle zvolené metody bezpečnostního kódu. Tyto programy by měly přispět ke zpřehlednění a lepšímu pochopení daného problému. Popis jednotlivých programů lze najít v kapitole 4.

Je nutné zdůraznit, že prostředí WebMathematica je velmi vhodným nástrojem pro vytváření praktických příkladů, se kterými mohou studenti komunikovat on-line. Mezi nevýhody tohoto systému lze zařadit například obtížné hledání chyby ve zdrojovém kódu, jelikož příkazy se vepisují do zdrojového kódu html stránky a není proto možné zobrazovat chybové hlášky jako u samotného programu Mathematica.

Celá práce je doplněna o teorii lineárních binárních bezpečnostních kódů, která společně s příklady ukazuje principy programů na kódování a dekódování. Věřím že výsledky mé práce alespoň trochu pomohou studentům při skládání zkoušek.

ZÁVĚR V ANGLIČTINĚ

The essence of the tabled composition, which falls thematical to the linear binary safety codes area and at the same time to the multimedia espousal of teaching, was the modification and the completion of an exist internet teaching assistant.

One of the partial goals was to modify and first of all to complete chapters about Hamming and Reed – Muller codes. The graphics wiev of programs, chapters and examples was chosen same as the wiev of original pages. More detailed description of adaptations, wich were made, contains the chapter 3.

The main goal of the work was to complete the internet teaching assistant by programs in facies WebMathematica. During the making this bachelor thesis it has been the occasion to learn basic applications of facies WebMathematica., which is avaliable for the students at the Thomas Bata University in Zlín.

After it were made six programs of the coding and decoding desired code words area after choice method of safety code. These programs should have been assistant to make easier and to better understanding the problem. Description of these programs contains the chapter 4.

It is necessary to accent that the using of the systems like the WebMathematica is an big advantage to make the practical examples, which are on-line for the students. A disavantage is a difficult searching errors in the source code, because commands are written in the source code of html pages and it is not possible to show errors like in the program Mathematica.

The whole work is completed by the theory of linear binary safety codes, which together with examples shows principles of programs for coding and decoding. I hope that results of my work will help other students during passing the exams.

SEZNAM POUŽITÉ LITERATURY

[1] VLČEK, Karel. *Komprese a kódová zabezpečení v multimediálních komunikacích*. 2. vyd. Praha : BEN - TECHNICKÁ LITERATURA, 2004. 258 s. ISBN 80-7300-134-9.

[2] NĚMEC, Karel. *Majoritní dekódování blokových kódů* [online]. 21.7.2004 [cit. 2007-04-19]. Dostupný z WWW: < http://147.229.144.23/clanky/04041/>.

[3] PRSKAVEC, Ladislav. *WebMathematica 2* [online]. 19.1.2005 [cit. 2007-02-10]. Dostupný z WWW: < http://k315.feld.cvut.cz/vyuka/webmathematica/>.

[4] PROKOPOVÁ, Z. *Zaklady informatiky*. Skriptum VUT Brno. Brno: FAKULTA TECHNOLOGICKÁ, 1991.

[5] POPELKOVÁ, Kateřina. *Současná česká grafická úprava knih, možnosti propagace literatury.* Elektronická kniha. Bakalářská práce. FT UTB Zlín, 2001.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PAS Počítačové algebraické systémy.

JSP Java server pages.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZDROJOVÝ KÓD 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta HTTP-EQUIV="Content-type" Content="text/html; charset=Windows(CP 1250)">
            <title>Hammingovy kódy - kódování</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td>
<font color="#5070a0">
<h1><center>Hammingovy kódy - kódování</center></h1>
<center><h4>Zadejte kódové slovo, které chcete zakódovat. Pro kód (3,1) jednomístné, pro (7,4) čtyřmístné a pro
(15,11) jedenáctimístné!
</font><hr></center>
<form method="post" action="hamming2.jsp">
<input type="text" size="10" name="vstup">
<input type="submit" value="Zakóduj">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadané kódové slovo:</b>
 </td>
 <td>
<msp:evaluate>
vstup=ToString[$$vstup]
</msp:evaluate>
 </td></tr>
<msp:evaluate>

a = "";
aa = "";
For[i=0,i<StringLength[vstup],i++;
a=a<>StringTake[vstup,{i}]<>",";];
For[i=0,i<StringLength[a]-1,i++;aa=aa<>StringTake[a,{i}];];
aa = "{" <> aa <> "}";
a=ToExpression[aa,TraditionalForm];
If[StringLength[vstup] == 1, g = {{1, 1, 1}};];
If[StringLength[vstup] == 4,
    g = {{1, 1, 0, 1, 0, 0, 1}, {0, 1, 0, 1, 0, 1, 0}, {1, 1, 1, 0, 0, 0,
          0}, {1, 0, 0, 1, 1, 0, 0}};];
If[StringLength[vstup]==11,
g={{1,1,0,1,0,0,0,1,0,0,0,0,0,0,1},{0,1,0,1,0,0,0,1,0,0,0,0,0,1,0},{1,1,1,0,0,0,0,0,0,0,0,0,0,0,0},{1,0,0,1,0,0,0,1,0,0,0,0,1
,0,0},{1,0,0,1,1,0,0,0,0,0,0,0,0,0,0},{0,1,0,1,0,1,0,0,0,0,0,0,0,0,0},{1,1,0,1,0,0,1,0,0,0,0,0,0,0,0},{0,0,0,1,0,0,0,1,0,0,0,1
,0,0,0},{1,0,0,0,0,0,0,1,1,0,0,0,0,0,0},{0,1,0,0,0,0,0,1,0,1,0,0,0,0,0},{1,1,0,0,0,0,0,1,0,0,1,0,0,0,0}};
];
x=a.g;
x=ToExpression[StringReplace[ToString[x],{"2"->"0","3"->"1","4"->"0","5"->"1","6"->"0","7"-
>"1"}],TraditionalForm];
</msp:evaluate>
<tr>
  <td>
   <b>Generující matice:</b>
  </td>
  <td>
<msp:evaluate>
```

```
MSPFormat[MatrixForm[g]]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Zakódované slovo:</b>
  </td>
  <td>
<msp:evaluate>
rozhod=1;
For[i=0,i<StringLength[vstup],i++;
If[StringTake[vstup,{i}]=="0" \[Or] StringTake[vstup,{i}]=="1", rozhod2=1, rozhod=0];];
If[rozhod==1,
If[StringLength[vstup]== 1 \[Or] StringLength[vstup]==4 \[Or] StringLength[vstup]==11,
MSPFormat[MatrixForm[{x}]], MSPFormat[MatrixForm["Špatná délka zadaného
slova"]]],MSPFormat[MatrixForm["Zadané slovo obsahuje nepovolené znaky"]]]
</msp:evaluate>
  </td></tr></table></b>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```

PŘÍLOHA P II: ZDROJOVÝ KÓD 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
        <title> Hammingovy kódy - dekódování</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td>
<font color="#5070a0">
<h1><center>Hammingovy kódy - dekódování</center></h1>
<center><h4>Zadejte kódové slovo, bylo přijato. Pro kód (3,1) trojmístné, pro (7,4) sedmimístné a pro (15,11)
patnáctimístné
</font>
<hr>
</center>
<form method="post" action="hamdek.jsp">
<input type="text" size="20" name="vstup">
<input type="submit" value="Dekóduj">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadané kódové slovo:</b>
 </td>
 <td>
<msp:evaluate>
vstup=ToString[$$vstup]
</msp:evaluate>
 </td></tr>
<msp:evaluate>

v = "";
vv = "";
For[i=0,i<StringLength[vstup],i++;
v=v<>StringTake[vstup,{i}]<>",";];
For[i=0,i<StringLength[v]-1,i++;vv=vv<>StringTake[v,{i}];];
vv = "{" <> vv <> "}";
v=ToExpression[vv,TraditionalForm];
If[StringLength[vstup] == 3, h = {{0, 1, 1},{1,0,1}};];
If[StringLength[vstup] == 7,
h = {{0, 0, 0, 1, 1, 1, 1}, {0, 1, 1, 0, 0, 1, 1}, {1, 0, 1, 0, 1, 0, 1}};
    ];
If[StringLength[vstup]==15,
h={{0,0,0,0,0,0,0,1,1,1,1,1,1,1,1},{0,0,0,1,1,1,1,0,0,0,0,1,1,1,1},{0,1,1,0,0,1,1,0,0,1,1,0,0,1,1},{1,0,1,0,1,0,1,0,1,0,1,0,1
,0,1}};
];
s=h.v;
s=ToExpression[StringReplace[ToString[s],{"2"->"0","3"->"1","4"->"0","5"->"1","6"->"0","7"->"1","8"->"0","9"-
>"1","10"->"0","11"->"1"}],TraditionalForm];
</msp:evaluate>
<tr>
  <td>
   <b>Kontrolní matice:</b>
  </td>
  <td>
```

```
<msp:evaluate>
MSPFormat[MatrixForm[h]]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Dekódovaný syndrom:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[MatrixForm[s]]
</msp:evaluate>
  </td></tr>

<tr>
<td>
<b>Přijaté slovo obsahuje chybu na místě:</b>
</td>
<td>
<msp:evaluate>
a=0;
rozhod=1;
For[
     i=0,i<StringLength[vstup],i++;
  If
     [StringTake[vstup,{i}]=="0" \[Or] StringTake[vstup,{i}]=="1", rozhod2=1, rozhod=0
     ];
    ];
If
   [rozhod==1,


If[StringLength[vstup]== 3 \[Or] StringLength[vstup]==7 \[Or] StringLength[vstup]==15,
             If
      [s=={0,0} \[Or] s=={0,0,0} \[Or] s=={0,0,0,0}, MSPFormat["Přijaté slovo neobsahuje chyby"],
           For
             [i = 0, i < Length[Part[h, 1]], i++; x = h[[All, i]];
                                      If[x == s, a = i]
             ];
           If[a==0,MSPFormat[Přijaté slovo obsahuje více chyb],MSPFormat[a]]

             ],MSPFormat[MatrixForm["Zadané slovo má špatnou délku"]]],
MSPFormat[MatrixForm["Zadané slovo obsahuje nepovolené znaky"]]
    ]

</msp:evaluate>
</td>

</tr></table></b>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```

PŘÍLOHA P III: ZDROJOVÝ KÓD 3

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
        <title>Rozšířené Hammingovy kódy - kódování</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td>
<font color="#5070a0">
<h1><center>Rozšířené Hammingovy kódy - kódování</center></h1>
<center><h4>Zadejte kódové slovo, které chcete zakódovat. Pro kód (4,1) jednomístné, pro (8,4) ctyřmístné a pro
(16,11) jedenáctimístné!
</font>
<hr></center>
<form method="post" action="hammingroz.jsp">
<input type="text" size="10" name="vstup">
<input type="submit" value="Zakóduj">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadané kódové slovo:</b>
 </td>
 <td>
<msp:evaluate>
vstup=ToString[$$vstup]
</msp:evaluate>
 </td></tr>
<msp:evaluate>

a = "";
aa = "";
For[i=0,i<StringLength[vstup],i++;
a=a<>StringTake[vstup,{i}]<>",";];
For[i=0,i<StringLength[a]-1,i++;aa=aa<>StringTake[a,{i}];];
aa = "{" <> aa <> "}";
a=ToExpression[aa,TraditionalForm];
If[StringLength[vstup] == 1, g = {{1, 1, 1,1}};];
If[StringLength[vstup] == 4,
g = {{1, 0, 0, 0, 0, 1, 1, 1}, {0, 1, 0, 0, 1, 0, 1, 1}, {0, 0, 1, 0, 1, 1, 0,
       1}, {0, 0, 0, 1, 1, 1, 1, 0}};
   ];
If[StringLength[vstup]==11,
g={{1,1,0,1,0,0,0,1,0,0,0,0,0,0,1,1},{0,1,0,1,0,0,0,1,0,0,0,0,0,1,0,0},{1,1,1,0,0,0,0,0,0,0,0,0,0,0,1},{1,0,0,1,0,0,0,1,0,0
,0,0,1,0,0,0},{1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1},{0,1,0,1,0,1,0,0,0,0,0,0,0,0,1},{1,1,0,1,0,0,1,0,0,0,0,0,0,0,0},{0,0,0,1,0
,0,0,1,0,0,0,1,0,0,0,1},{1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1},{0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,1},{1,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0}}
;
];
x=a.g;
x=ToExpression[StringReplace[ToString[x],{"2"->"0","3"->"1","4"->"0","5"->"1","6"->"0","7"-
>"1"}],TraditionalForm];
</msp:evaluate>
<tr>
  <td>
   <b>Generující matice:</b>
```

```
  </td>
  <td>
<msp:evaluate>
MSPFormat[MatrixForm[g]]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Zakódované slovo:</b>
  </td>
  <td>
<msp:evaluate>
rozhod=1;
For[i=0,i<StringLength[vstup],i++;
If[StringTake[vstup,{i}]=="0" \[Or] StringTake[vstup,{i}]=="1", rozhod2=1, rozhod=0];];
If[rozhod==1,
If[StringLength[vstup]== 1 \[Or] StringLength[vstup]==4 \[Or] StringLength[vstup]==11,
MSPFormat[MatrixForm[{x}]], MSPFormat[MatrixForm["Špatná délka zadaného
slova"]]],MSPFormat[MatrixForm["Zadané slovo obsahuje nepovolené znaky"]]]
</msp:evaluate>
  </td></tr></table></b>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```

PŘÍLOHA P IV: ZDROJOVÝ KÓD 4

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
        <title> Hammingovy rozšířené kódy - dekódování</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td>
<font color="#5070a0">
<h1><center>Hammingovy rozšířené kódy - dekódování</center></h1>
<center><h4>Zadejte kódové slovo,které bylo přijato. Pro kód (4,1) čtyř, pro (8,4) osmi a pro (16,11) šestnáctimístné
</font><hr></center>
<form method="post" action="hamdekroz.jsp">
<input type="text" size="10" name="vstup">
<input type="submit" value="Dekóduj">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadané kódové slovo:</b>
 </td>
 <td>
<msp:evaluate>
vstup=ToString[$$vstup]
</msp:evaluate>
 </td></tr>
<msp:evaluate>
v = "";
vv = "";
For[i=0,i<StringLength[vstup],i++;
v=v<>StringTake[vstup,{i}]<>",";];
For[i=0,i<StringLength[v]-1,i++;vv=vv<>StringTake[v,{i}];];
vv = "{" <> vv <> "}";
v=ToExpression[vv,TraditionalForm];
If[StringLength[vstup] == 4, h = {{1,1,0,0},{1,0,1,0},{1,0,0,1}};];
If[StringLength[vstup] == 8,
h = {{0, 0, 0, 1, 1, 1, 1, 0}, {0, 1, 1, 0, 0, 1, 1, 0}, {1, 0, 1, 0, 1, 0, 1, 0}, {1,1,1,1,1,1,1,1}};
    ];
If[StringLength[vstup]==16,
h={{0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0},{0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0},{0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0},{1,0,1,0,1,0,1,0,1,0
,1,0,1,0,1,0},{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}};
];
s=h.v;
s=ToExpression[StringReplace[ToString[s],{"2"->"0","3"->"1","4"->"0","5"->"1","6"->"0","7"->"1","8"->"0","9"-
>"1","10"->"0","11"->"1"}],TraditionalForm];
</msp:evaluate>
<tr>
  <td>
   <b>Kontrolní matice:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[MatrixForm[h]]
</msp:evaluate>
  </td></tr>
```

```
<tr>
  <td>
   <b>Dekódovaný syndrom:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[MatrixForm[s]]
</msp:evaluate>
  </td></tr>
<tr>
<td>
<b>Přijaté slovo obsahuje chybu na místě:</b>
</td>

<td>
<msp:evaluate>
a=0;
rozhod=1;
For[
    i=0,i<StringLength[vstup],i++;
  If
    [StringTake[vstup,{i}]=="0" \[Or] StringTake[vstup,{i}]=="1", rozhod2=1, rozhod=0
    ];
   ];
If
  [rozhod==1,
If[StringLength[vstup]== 4 \[Or] StringLength[vstup]==8 \[Or] StringLength[vstup]==16,
            If
     [s=={0,0,0} \[Or] s=={0,0,0,0} \[Or] s=={0,0,0,0,0}, MSPFormat["Přijaté slovo neobsahuje chyby"],
          For
            [i = 0, i < Length[Part[h, 1]], i++; x = h[[All, i]];
                                    If[x == s, a = i]
            ];
          If[a==0,MSPFormat["Přijaté slovo obsahuje dvě chyby, nelze určit místo"],MSPFormat[a]]
            ],MSPFormat[MatrixForm["Zadané slovo má špatnou délku"]]],

MSPFormat[MatrixForm["Zadané slovo obsahuje nepovolené znaky"]]
   ]
</msp:evaluate>
</td>
</tr></table></b>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```

PŘÍLOHA P V: ZDROJOVÝ KÓD 5

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
        <title>Reed-Mullerovy kódy - kódování</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td>
<font color="#5070a0">
<h1><center>Reed-Mullerovy kódy - kódování</center></h1>
<center><h4>Zadejte kódové slovo, které chcete zakódovat. Pro kód R(1,4) pětimístné a pro R(2,4) jedenáctimístné!
</font><hr></center>
<form method="post" action="reedmuller.jsp">
<input type="text" size="10" name="vstup">
<input type="submit" value="Zakóduj">
</form>
<p>
<b>
<p>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadané kódové slovo:</b>
 </td>
 <td>
<msp:evaluate>
vstup=ToString[$$vstup]
</msp:evaluate>
 </td></tr>
<msp:evaluate>

a = "";
aa = "";
For[i=0,i<StringLength[vstup],i++;
a=a<>StringTake[vstup,{i}]<>",";];
For[i=0,i<StringLength[a]-1,i++;aa=aa<>StringTake[a,{i}];];
aa = "{" <> aa <> "}";
a=ToExpression[aa,TraditionalForm];
If[StringLength[vstup] == 5,
g = {{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {0, 1, 0, 1, 0, 1, 0,
     1, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
     1, 1}, {0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1}, {0, 0, 0, 0, 0,
      0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1}};
   ];
If[StringLength[vstup]==11,
g = {{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {0, 1, 0, 1, 0, 1, 0,
     1, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
     1, 1}, {0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1}, {0, 0, 0, 0, 0,
      0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1}, {0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
     0, 0, 0, 1}, {0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1}, {0, 0, 0,
      0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
     0, 0, 0, 0, 1, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1}, {0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1}};
];
x=a.g;
x=ToExpression[StringReplace[ToString[x],{"2"->"0","3"->"1","4"->"0","5"->"1","6"->"0","7"->"1","8"->"0","9"-
>"1","10"->"0","11"->"1"}],TraditionalForm];
```

```
</msp:evaluate>
<tr>
  <td>
   <b>Generující matice:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[MatrixForm[g]]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Zakódované slovo:</b>
  </td>
  <td>
<msp:evaluate>
rozhod=1;
For[i=0,i<StringLength[vstup],i++;
If[StringTake[vstup,{i}]=="0" \[Or] StringTake[vstup,{i}]=="1", rozhod2=1, rozhod=0];];
If[rozhod==1,
If[StringLength[vstup]== 5 \[Or] StringLength[vstup]==11,
MSPFormat[MatrixForm[{x}]], MSPFormat[MatrixForm["Špatná délka zadaného
slova"]]],MSPFormat[MatrixForm["Zadané slovo obsahuje nepovolené znaky"]]]
</msp:evaluate>
  </td></tr></table></b>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```

PŘÍLOHA P VI: ZDROJOVÝ KÓD 6

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
        <title> Reed-Mullerův kód - dekódování</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td>
<font color="#5070a0">
<h1><center>Reed-Mullerův kód - dekódování</center></h1>
<center><h4>Zadejte kódové slovo, které bylo přijato. Pro kód (2,4) šestnáctimísné.
</font><hr></center>
<form method="post" action="reeddek.jsp">
<input type="text" size="17" name="vstup">
<input type="submit" value="Dekóduj">
</form>
<p>
<b>
<p>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadané kódové slovo:</b>
 </td>
 <td>
<msp:evaluate>
vstup=ToString[$$vstup]
</msp:evaluate>
 </td></tr>
<msp:evaluate>
If[vstup=="$$vstup",f={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
f = "";
chyba = 0;
oprava = 0;
vv = "";
For[i=0,i<StringLength[vstup],i++;
f=f<>StringTake[vstup,{i}]<>",";];
For[i=0,i<StringLength[f]-1,i++;vv=vv<>StringTake[f,{i}];];
vv = "{" <> vv <> "}";
f=ToExpression[vv,TraditionalForm];];
p = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
g = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
h = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
m = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
j = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
k = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
l = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
pdeset = {0, 0, 0, 0};
pdevitka = {0, 0, 0, 0};
posm = {0, 0, 0, 0};
psedm = {0, 0, 0, 0};
psest = {0, 0, 0, 0};
ppet = {0, 0, 0, 0};
pctyri = {0, 0, 0, 0};
ptri = {0, 0, 0, 0};
pdva = {0, 0, 0, 0};
pjedna = {0, 0, 0, 0};
```

```
pnula = 0;

pdeset[[1]] = f[[1]] + f[[5]] + f[[9]] + f[[13]];
If[pdeset[[1]] == 3, pdeset[[1]] = 1];
If[pdeset[[1]] == 2 \[Or] pdeset[[1]] == 4, pdeset[[1]] = 0];
pdeset[[2]] = f[[2]] + f[[6]] + f[[10]] + f[[14]];
If[pdeset[[2]] == 3, pdeset[[2]] = 1];
If[pdeset[[2]] == 2 \[Or] pdeset[[2]] == 4, pdeset[[2]] = 0];
pdeset[[3]] = f[[3]] + f[[7]] + f[[11]] + f[[15]];
If[pdeset[[3]] == 3, pdeset[[3]] = 1];
If[pdeset[[3]] == 2 \[Or] pdeset[[3]] == 4, pdeset[[3]] = 0];
pdeset[[4]] = f[[4]] + f[[8]] + f[[12]] + f[[16]];
If[pdeset[[4]] == 3, pdeset[[4]] = 1];
If[pdeset[[4]] == 2 \[Or] pdeset[[4]] == 4, pdeset[[4]] = 0];
For[i = 0, i < 4, i++; p[[11]] = p[[11]] + pdeset[[i]];];
If[p[[11]] == 1 \[Or] p[[11]]==2 \[Or] p[[11]] == 3, oprava = 1];
If[p[[11]] == 2, chyba = 1,
  If[p[[11]] < 2, p[[11]] = 0, p[[11]] = 1];];

pdevitka[[1]] = f[[1]] + f[[3]] + f[[9]] + f[[11]];
If[pdevitka[[1]] == 3, pdevitka[[1]] = 1];
If[pdevitka[[1]] == 2 \[Or] pdevitka[[1]] == 4, pdevitka[[1]] = 0];
pdevitka[[2]] = f[[2]] + f[[4]] + f[[10]] + f[[12]];
If[pdevitka[[2]] == 3, pdevitka[[2]] = 1];
If[pdevitka[[2]] == 2 \[Or] pdevitka[[2]] == 4, pdevitka[[2]] = 0];
pdevitka[[3]] = f[[5]] + f[[7]] + f[[13]] + f[[15]];
If[pdevitka[[3]] == 3, pdevitka[[3]] = 1];
If[pdevitka[[3]] == 2 \[Or] pdevitka[[3]] == 4, pdevitka[[3]] = 0];
pdevitka[[4]] = f[[6]] + f[[8]] + f[[14]] + f[[16]];
If[pdevitka[[4]] == 3, pdevitka[[4]] = 1];
If[pdevitka[[4]] == 2 \[Or] pdevitka[[4]] == 4, pdevitka[[4]] = 0];
For[i = 0, i < 4, i++; p[[10]] = p[[10]] + pdevitka[[i]];];
If[p[[10]] == 1 \[Or] p[[10]] == 3, oprava = 1];
If[p[[10]] == 2, chyba = 1,
  If[p[[10]] < 2, p[[10]] = 0, p[[10]] = 1];];

posm[[1]] = f[[1]] + f[[3]] + f[[5]] + f[[7]];
If[posm[[1]] == 3, posm[[1]] = 1];
If[posm[[1]] == 2 \[Or] posm[[1]] == 4, posm[[1]] = 0];
posm[[2]] = f[[2]] + f[[4]] + f[[6]] + f[[8]];
If[posm[[2]] == 3, posm[[2]] = 1];
If[posm[[2]] == 2 \[Or] posm[[2]] == 4, posm[[2]] = 0];
posm[[3]] = f[[9]] + f[[11]] + f[[13]] + f[[15]];
If[posm[[3]] == 3, posm[[3]] = 1];
If[posm[[3]] == 2 \[Or] posm[[3]] == 4, posm[[3]] = 0];
posm[[4]] = f[[10]] + f[[12]] + f[[14]] + f[[16]];
If[posm[[4]] == 3, posm[[4]] = 1];
If[posm[[4]] == 2 \[Or] posm[[4]] == 4, posm[[4]] = 0];
For[i = 0, i < 4, i++; p[[9]] = p[[9]] + posm[[i]];];
If[p[[9]] == 1 \[Or] p[[9]] == 3, oprava = 1];
If[p[[9]] == 2, chyba = 1,
  If[p[[9]] < 2, p[[9]] = 0, p[[9]] = 1];];


psedm[[1]] = f[[1]] + f[[2]] + f[[9]] + f[[10]];
If[psedm[[1]] == 3, psedm[[1]] = 1];
If[psedm[[1]] == 2 \[Or] psedm[[1]] == 4, psedm[[1]] = 0];
psedm[[2]] = f[[3]] + f[[4]] + f[[11]] + f[[12]];
If[psedm[[2]] == 3, psedm[[2]] = 1];
If[psedm[[2]] == 2 \[Or] psedm[[2]] == 4, psedm[[2]] = 0];
psedm[[3]] = f[[5]] + f[[6]] + f[[13]] + f[[14]];
If[psedm[[3]] == 3, psedm[[3]] = 1];
If[psedm[[3]] == 2 \[Or] psedm[[3]] == 4, psedm[[3]] = 0];
psedm[[4]] = f[[7]] + f[[8]] + f[[15]] + f[[16]];
If[psedm[[4]] == 3, psedm[[4]] = 1];
```

```
If[psedm[[4]] == 2 \[Or] psedm[[4]] == 4, psedm[[4]] = 0];
For[i = 0, i < 4, i++; p[[8]] = p[[8]] + psedm[[i]];];
If[p[[8]] == 1 \[Or] p[[8]] == 3, oprava = 1];
If[p[[8]] == 2, chyba = 1,
  If[p[[8]] < 2, p[[8]] = 0, p[[8]] = 1];];

psest[[1]] = f[[1]] + f[[2]] + f[[5]] + f[[6]];
If[psest[[1]] == 3, psest[[1]] = 1];
If[psest[[1]] == 2 \[Or] psest[[1]] == 4, psest[[1]] = 0];
psest[[2]] = f[[3]] + f[[4]] + f[[7]] + f[[8]];
If[psest[[2]] == 3, psest[[2]] = 1];
If[psest[[2]] == 2 \[Or] psest[[2]] == 4, psest[[2]] = 0];
psest[[3]] = f[[9]] + f[[10]] + f[[13]] + f[[14]];
If[psest[[3]] == 3, psest[[3]] = 1];
If[psest[[3]] == 2 \[Or] psest[[3]] == 4, psest[[3]] = 0];
psest[[4]] = f[[11]] + f[[12]] + f[[15]] + f[[16]];
If[psest[[4]] == 3, psest[[4]] = 1];
If[psest[[4]] == 2 \[Or] psest[[4]] == 4, psest[[4]] = 0];
For[i = 0, i < 4, i++; p[[7]] = p[[7]] + psest[[i]];];
If[p[[7]] == 1 \[Or] p[[7]] == 3, oprava = 1];
If[p[[7]] == 2, chyba = 1,
  If[p[[7]] < 2, p[[7]] = 0, p[[7]] = 1];];

ppet[[1]] = f[[1]] + f[[2]] + f[[3]] + f[[4]];
If[ppet[[1]] == 3, ppet[[1]] = 1];
If[ppet[[1]] == 2 \[Or] ppet[[1]] == 4, ppet[[1]] = 0];

ppet[[2]] = f[[5]] + f[[6]] + f[[7]] + f[[8]];
If[ppet[[2]] == 3, ppet[[2]] = 1];
If[ppet[[2]] == 2 \[Or] ppet[[2]] == 4, ppet[[2]] = 0];
ppet[[3]] = f[[9]] + f[[10]] + f[[11]] + f[[12]];
If[ppet[[3]] == 3, ppet[[3]] = 1];
If[ppet[[3]] == 2 \[Or] ppet[[3]] == 4, ppet[[3]] = 0];
ppet[[4]] = f[[13]] + f[[14]] + f[[15]] + f[[16]];
If[ppet[[4]] == 3, ppet[[4]] = 1];
If[ppet[[4]] == 2 \[Or] ppet[[4]] == 4, ppet[[4]] = 0];
For[i = 0, i < 4, i++; p[[6]] = p[[6]] + ppet[[i]];];
If[p[[6]] == 1 \[Or] p[[6]] == 3, oprava = 1];
If[p[[6]] == 2, chyba = 1,
  If[p[[6]] < 2, p[[6]] = 0, p[[6]] = 1];];

If[p[[6]] == 0, g = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   g = {0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1}];
If[p[[7]] == 0, h = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   h = {0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1}];
If[p[[8]] == 0, m = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   m = {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1}];
If[p[[9]] == 0, j = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   j = {0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1}];
If[p[[10]] == 0, k = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   k = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1}];
If[p[[11]] == 0, l = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   l = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1}];
ff = f + g + h + m + j + k + l;
ff = ToExpression[StringReplace[ToString[ff], {"2" -> "0", "3" -> "1", "4" -> "0", "5" -> "1","6" -> "0", "7" ->
"1"}],TraditionalForm];

pctyri[[1]] = ff[[8]] + ff[[16]];
If[pctyri[[1]] == 2 , pctyri[[1]] = 0];
pctyri[[2]] = ff[[7]] + ff[[15]];
If[pctyri[[2]] == 2 , pctyri[[2]] = 0];
pctyri[[3]] = ff[[6]] + ff[[14]];
If[pctyri[[3]] == 2 , pctyri[[3]] = 0];
pctyri[[4]] = ff[[5]] + ff[[13]];
If[pctyri[[4]] == 2 , pctyri[[4]] = 0];
```

```
For[i = 0, i < 4, i++; p[[5]] = p[[5]] + pctyri[[i]];];
If[p[[5]] == 1 \[Or] p[[5]] == 3, oprava = 1];
If[p[[5]] == 2, chyba = 1,
  If[p[[5]] < 2, p[[5]] = 0, p[[5]] = 1];];

ptri[[1]] = ff[[1]] + ff[[5]];
If[ptri[[1]] == 2 , ptri[[1]] = 0];
ptri[[2]] = ff[[2]] + ff[[6]];
If[ptri[[2]] == 2 , ptri[[2]] = 0];
ptri[[3]] = ff[[3]] + ff[[7]];
If[ptri[[3]] == 2 , ptri[[3]] = 0];
ptri[[4]] = ff[[4]] + ff[[8]];
If[ptri[[4]] == 2 , ptri[[4]] = 0];
For[i = 0, i < 4, i++; p[[4]] = p[[4]] + ptri[[i]];];
If[p[[4]] == 1 \[Or] p[[4]] == 3, oprava = 1];
If[p[[4]] == 2, chyba = 1,
  If[p[[4]] < 2, p[[4]] = 0, p[[4]] = 1];];

pdva[[1]] = ff[[1]] + ff[[3]];
If[pdva[[1]] == 2, pdva[[1]] = 0];
pdva[[2]] = ff[[2]] + ff[[4]];
If[pdva[[2]] == 2, pdva[[2]] = 0];
pdva[[3]] = ff[[5]] + ff[[7]];
If[pdva[[3]] == 2, pdva[[3]] = 0];
pdva[[4]] = ff[[6]] + ff[[8]];
If[pdva[[4]] == 2, pdva[[4]] = 0];
For[i = 0, i < 4, i++; p[[3]] = p[[3]] + pdva[[i]];];
If[p[[3]] == 1 \[Or] p[[3]] == 3, oprava = 1];
If[p[[3]] == 2, chyba = 1,
  If[p[[3]] < 2, p[[3]] = 0, p[[3]] = 1];];

pjedna[[1]] = ff[[1]] + ff[[2]];
If[pjedna[[1]] == 2, pjedna[[1]] = 0];
pjedna[[2]] = ff[[3]] + ff[[4]];
If[pjedna[[2]] == 2, pjedna[[2]] = 0];
pjedna[[3]] = ff[[5]] + ff[[6]];
If[pjedna[[3]] == 2, pjedna[[3]] = 0];
pjedna[[4]] = ff[[7]] + ff[[8]];
If[pjedna[[4]] == 2, pjedna[[4]] = 0];
For[i = 0, i < 4, i++; p[[2]] = p[[2]] + pjedna[[i]];];
If[p[[2]] == 1 \[Or] p[[2]]==2 \[Or] p[[2]] == 3, oprava = 1];
If[p[[2]] == 2, chyba = 1,
  If[p[[2]] < 2, p[[2]] = 0, p[[2]] = 1];];

x = 0;
For[i = 0, i < 16, i++; If[f[[i]] == 1, x = x + 1;]];
If[x > 5, p[[1]] = 1;, p[[1]] = 0;];

</msp:evaluate>
<tr>
<td>
Stav přijatého slova:
</td>
<td>
<msp:evaluate>
If[vstup=="$$vstup",MSPFormat[MatrixForm[{p}]],
If[oprava==1, MSPFormat[MatrixForm["Přijaté slovo obsahovalo chyby a program se je pokusil opravit při
dekódování"]], MSPFormat[MatrixForm["Program neopravoval žádnou chybu"]]]
]
</msp:evaluate>
</td>
</tr>
<tr>
<td>
<b>Dekódované slovo:</b>
```

```
</td>
<td>
<msp:evaluate>
a=0;
rozhod=1;
If[vstup=="$$vstup",MSPFormat[MatrixForm[{p}]],
If[chyba==1,MSPFormat[MatrixForm["Přijaté slovo obsahuje dvě chyby a nelze jej proto dekódovat"]],
For[
    i=0,i<StringLength[vstup],i++;
 If
    [StringTake[vstup,{i}]=="0" \[Or] StringTake[vstup,{i}]=="1", rozhod2=1, rozhod=0
    ];
   ];
If
  [rozhod==1,
If[StringLength[vstup]==16,MSPFormat[MatrixForm[{p}]]
            ,MSPFormat[MatrixForm["Zadané slovo má špatnou délku"]]],
MSPFormat[MatrixForm["Zadané slovo obsahuje nepovolené znaky"]]
   ]
]
]
</msp:evaluate>
</td>
</tr></table>
</b>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```