Gymnázium Vysoké Mýto

(79-41-K/81 – Gymnázium, osmileté studium)

15 PUZZLE

Maturitní práce

Vypracoval: Filip Peška
Vedoucí práce: Mgr. Petr Štorek, Ph.D.

Vysoké Mýto 2019

Prohlašuji, že jsem maturitní práci vypracoval samostatně na základě literatury a pramenů uvedených v seznamu použité literatury.

Ve Vysokém Mýtě dne 9.4.2019 …….

Filip Peška

Anotace

Cílem této maturitní práce je vytvoření logické hry 15 Puzzle pro jednoho hráče v prostředí programovacího jazyka Pascal. Obsahem je programová dokumentace, jejíž součástí je popis použitých datových typů, proměnných a podprogramů. Dále se práce skládá z vlastního zhodnocení aplikace a uživatelského manuálu.

Obsah

1 Úvod

Jako svou maturitní práci z informatiky jsem si vybral naprogramování logické hry. Původně bylo mým cílem vytvořit populární hru s názvem 2048. Ta byla ovšem zpracována již v minulých letech, proto jsem musel svůj výběr pozměnit. Jelikož jsem měl část mé původně zamýšlené aplikace hotovou, vybíral jsem nové téma tak, abych mohl využít co nejvíce z původně vytvářeného programu. Takto jsem došel k logické hře 15 Puzzle.

Protože prostředí programovacího jazyka Pascal neumožňuje vytvoření moderně vyhlížející aplikace, rozhodl jsem se, že vzhledem práce napodobím první hry, které na počítačích vznikaly. Pro tyto účely jsem do „načítání hry“ a „menu“ přidal také elementární systémové zvuky, které může uživatel vypnout.

2 Programová dokumentace

V oddílu „Programová dokumentace“ podrobněji vysvětluji fungování programu z hlediska použitých proměnných, jednotek a podprogramů.

2.1 Typy proměnných

Ve svém programu jsem použil poměrně malé spektrum typů proměnných, a to jak globálních (proměnné pro celý program), tak lokálních (proměnné pouze pro podprogramy). Většinou jsem si vystačil s klasickým typem integer a char (podrobně popsáno níže). Na některých místech v mém programu by mohl být typ integer nahrazen také typem byte, shortint, nebo dokonce typem boolean. Tyto tři alternativní typy jsem popsal v podkapitole číslo 2.1.2.

2.1.1 Použité typy proměnných

V této podkapitole podrobněji popisuji pouze ty typy, které jsem ve svém programu použil.

Integer – Do proměnných tohoto typu mohou být ukládána pouze celá čísla od - 32 768 do 32 767. Všechny proměnné představující počítadlo, souřadnice nebo určitou volbu nesou v mém programu právě tento typ.

Char – Typ char pracuje se znaky, které mohou být jak tisknutelné (písmena, číslice, interpunkce), tak netisknutelné (mezerník, enter, backspace apod.). Systém může číst znaky přímo jako symboly (např. g, G, ‘9‘, !) nebo pomocí čísla ve tvaru desítkové soustavy z tabulky ASCII (např. #71, #278 apod.). Při zadávání jména ve hře čte program znaky kláves, které uživatel zadává (tudíž i enter či backspace). Každá další interakce uživatele s programem probíhá také právě pomocí tohoto typu – výběr v menu, resetování hry, potvrzení volby aj.

String – String neboli řetězec znaků slouží k ukládání jmen hráčů a k jejich následnému čtení a vypsání. String může vzniknout už jako původní řetězec nebo také jako součet jednotlivých znaků proměnných char. Při čtení z textového dokumentu se obsah čte právě po řetězcích.

Array of – Hru jako takovou hráč vidí jako pole čísel, které musí seřadit. Přesně tak tomu opravdu je i v samotném kódu, konkrétně se jedná o pole dvojrozměrné. Array of je použitý také za účelem ukládání a čtení tabulky výsledků. Na tuto akci si program díky vlastním typům record vystačí pouze s jedním rozměrem.

Text – Externí textový dokument, ve kterém jsou zapsána jména a výsledky jednotlivých hráčů. Slouží pro uložení dat, která se musí po zavření aplikace uchovat.

Const – Konstanty, které nelze nijak měnit. V mém podání použity pouze pro velikost čtvercových polí, která se vypisují okolo čísel, a pro počet vypisovaných hráčů.

Hrac4 – Vlastní typ, který jsem si za účelem snadného ukládání, čtení a vypisování nejlepších výsledků vytvořil. Jedná se o záznam (record) složený ze dvou proměnných – *jmeno* typu string a *skore* typu integer. Pole (Array of) těchto záznamů pak reprezentuje tabulku. Totožně fungují typy Hrac3 a Hrac5, jediným rozdílem jsou data, která čtou a ukládají, výběr konkrétního záznamového typu závisí na zvolené obtížnosti.

2.1.2 Alternativní typy proměnných

Typ integer je vyhovující, avšak poměrně velikostně rozsáhlý. Má zkrácené alternativy, které by bylo možné použít pro případné datové zjednodušení programu.

Boolean – Logický typ, který představuje pouze dva stavy – true a false. Můžeme si ho představit také jako číselný typ o dvou hodnotách.

Byte – Už samotné jméno naznačuje, že velikost jedné proměnné byte je 1 byte čili 8 bitů. Osahuje nezáporné hodnoty (0–255).

Shortint – Jedná se o byte, ve kterém existují záporná čísla (hodnoty -128–127).

Tabulka 1: Přehled velikostí číselných typů

Název	Velikost (bity)	Hodnoty
Integer	16	-32 768–32 767
Shortint	8	-128–127
Byte	8	0–255
Boolean	8*	true, false

*Ač typ boolean nabývá pouze dvou hodnot, nejmenší adresovatelná velikost je 8 bitů.

2.2 Použité jednotky

Pascal umí bez jakékoliv jednotky provádět jednoduché matematické operace, pracovat se znaky, čísly a řetězci a ovládat vstupy a výstupy z programu. Existují jednotky, které definují podprogramy usnadňující práci při programování.

Crt – Velmi důležitá přídavná jednotka, přidává do programu například následující funkce, které jsem při tvorbě hry použil: *GotoXY, Delay, Sound, TextColor, ClrScr, WhereX, WhereY, Window, ReadKey*.

Windows – Z této jednotky jsem použil pouze funkci *Beep*, jež generuje zvuky o zvolené frekvenci a délce.

2.3 Podprogramy

Podprogramy značně usnadňují práci s celým kódem. Tvorba univerzálního podprogramu může zabrat více času, ten se ovšem vrátí s každým dalším použitím téže procedury nebo funkce, neboť se nemusí vytvářet další, leč podobná. Procedura je typ podprogramu, který vykoná zadaný úkol a sám o sobě nevrací žádnou hodnotu. Výstupem funkce je naopak určitá hodnota.

Ve svém programu jsem používal hlavně procedury, jelikož jsem uživateli nepotřeboval sdělovat žádný výpočet, tudíž by mi funkce nic zásadně neusnadnila. Jediné mnou použité funkce byly předdefinovány systémem. V této kapitole nebudu popisovat všechny procedury, ale pouze ty, které jsou v programu něčím jedinečné nebo důležité a které jsem zároveň sám vytvářel.

Prectisito

Čtení ze souboru probíhá v této proceduře. Na základě dotazu na určitou tabulku výsledků otevře textový dokument (příkaz *reset*) pro danou obtížnost a naplní pole záznamy, které čte po řádcích (cyklus *for*). Na každém lichém řádku najde jméno (string), na každém sudém řádku skóre (integer). Tyto dva řádky spáruje do jednoho záznamu, díky čemuž je značně usnadněno vypisování.

Vypismito

Výpis probíhá v cyklech od jedné do konstantního počtu hráčů (10). Postupně se vypíše celé pole záznamů (cyklus *for*). Formátování je řešeno v rámci cyklu zjišťováním délky jména uživatele (funkce *length*) a následným vypsáním určitého počtu mezer, aby všechna skóre byla zarovnána doprava pod sebe.

Pridejsiho

V momentě, kdy uživatel úspěšně dokončí hru, se jeho jméno a skóre uloží na jedenácté místo v tabulce výsledků (příkaz *append*).

Seradsito

Záznamy se podle velikosti řadí cyklickým porovnáváním (cyklus *for*, podmínka *if*) sousedních hodnot skóre. Tato procedura probíhá pouze tehdy, je-li v tabulce 11 hodnot. Takto se seřadí pouze pole záznamů v programu, nikoli v textovém dokumentu, ten je až následně přepsán.

Smazsito

Mazání probíhá přepsáním textového dokumentu tak, že vypíše pouze seřazených prvních 10 záznamů. Jedenáctý tak není dále externě uchováván.

Dilek

Tato procedura vypisuje zvolený dílek na určitém místě. Například čtverec číslo 6 má pevně určenou pozici, kde se vždy vypisuje. Procedura slouží jak pro výpis všech políček, tak pro následné mazání políčka s nejvyšší hodnotou (jeví se jako prázdné místo, kam se hrací kameny posouvají). Stěžejní je právě větvení *case of*.

Nahodne_prirad

Podle zvolené obtížnosti se vypíše 9, 16 nebo 25 dílků. Každému z nich je přiřazena hodnota (cyklus *for*). Hodnot je právě tolik, kolik je dílků. Funkce *random* vybírá náhodné číslo od 1 do krajní meze (9/16/25). Vybrané číslo se vyjme z množiny čísel, nahradí se na původním místě nulou a přiřadí se určitému políčku. Pokud by systém vybral znovu stejné číslo, narazí v množině právě na onu nulu a proces náhodného výběru se opakuje (cyklus *repeat*). Postupně mají všechny dílky přiřazené číslo a množina, ze které se hodnoty vybíraly, je tvořena pouze nulami.

Vypiscisla

Čísla jsou náhodně rozlosována, tudíž je můžeme vypsat a ukázat uživateli. K tomuto jevu dochází právě v této proceduře. Hodnoty jsou vypisovány (cyklus *for*) do přibližného středu budoucích čtverců. Nejvyšší číslo z dané řady je vypsáno černě.

Vypisdilky

K výpisu dílků okolo čísel dochází zde. Od jedné do 9/16/25 se vypíší (cyklus *for*) čtverce z procedury *dilek*. Každý přesně na svou souřadnici (*GotoXY*).

Sixteen

Nejvyšší číslo se maže již v proceduře *vypiscisla*, nyní je konečně smazáno i jeho ohraničení. Mazání je pouze zdánlivé, neboť narazí-li program na mezní číslo, vypíše kolem něj černý čtverec. Používá podmínkový příkaz *if*.

Priradvypis

Procedura tvořená procedurami *vypisdilky*, *sixteen* a *vypiscisla*. Je samostatná, neboť se po celou hru po každém tahu opakuje. Neustále se musí zjišťovat, kde se nachází nejvyšší číslo, a jeho ohraničení i hodnotu je potřeba smazat. Aby se takto nesmazalo celé hrací pole, jsou na začátku procedury všechna ohraničení obnovena.

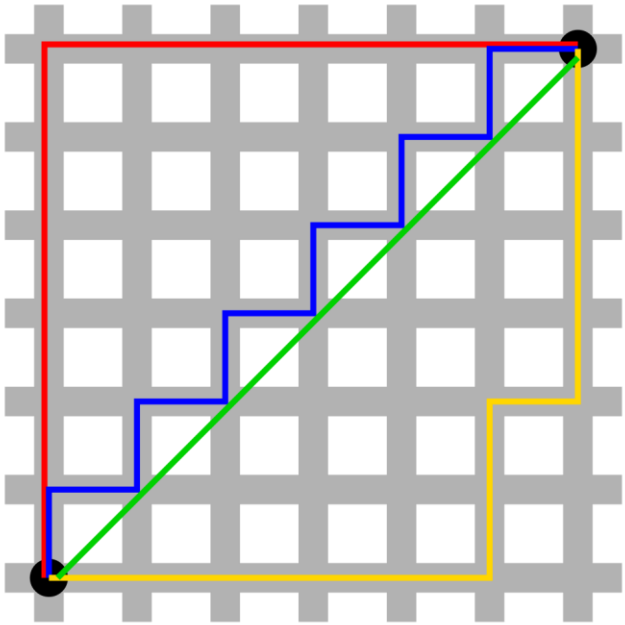
Pravidlo

Problém hry 15 Puzzle je ten, že možností, jak lze čísla náhodně do pole rozmístit například při obtížnosti 4, je 16!, to se pro představu rovná 20922789888000. Přesně polovina těchto permutací nevede ke zdárnému konci, proto se musí všechna tato vadná rozmístění eliminovat. Pravidlo uvádí, že je-li součet počtu inverzí v permutaci (oproti finální permutaci, kterou je posloupnost 1 až 16 s diferencí 1) a Manhattanské vzdálenosti prázdného políčka (nejvyšší číslo) od pravého dolního rohu sudý, poté má hlavolam řešení, pokud je součet lichý, řešení nenalezneme[1]. Analogicky funguje pravidlo i pro obtížnosti 3 a 5.

Procedura *pravidlo* zkoumá právě počet inverzí v permutaci. Program se postupně dotazuje, zda je číslo *a* na místě, kde má ve finální permutaci být. Pokud narazí na inverzi, prohodí hledané číslo *a* s číslem *b*, které se nachází na místě čísla *a*. Tím se zvýší počet inverzí o 1. Takto se seřadí celé hrací pole. Jedná se o kombinaci cyklů *for* a *repeat* společně s podmínkovým příkazem *if*.

ManhMe

Manhattanská vzdálenost je vzdálenost dvou bodů v rovině měřená po odvěsnách pravoúhlého trojúhelníka. Zažitější máme vzdálenost Euklidovskou, která je měřená po přeponě[2]. Na obrázku č. 1 je zeleně vyznačena Euklidovská vzdálenost dvou bodů, ostatní barvy vyjadřují vzdálenost Manhattanskou. Můžeme si všimnout, že při pohybu směrem k druhému bodu nezáleží na počtu změn směru.



Obrázek č. 1 – Manhattanská x Euklidovská metrika

Procedura *ManhMe* zjišťuje právě vzdálenost prázdného pole od pravého dolního rohu Manhattanskou metrikou. Vzdálenost je udávána v dílcích. Program využívá převážně cyklus *for* a příkaz *if*.

Zkouska

Pomocí příkazu *if* kontroluje, zda je součet počtu inverzí v permutaci a Manhattanské vzdálenosti prázdného pole od pravého dolního rohu sudý nebo lichý. Parita součtu je zkoumána funkcí *mod* jako zbytek po dělení dvěma (je-li výsledek 1, číslo je liché). Zároveň je díky příkazu *if* ošetřena situace, kdy se hráči náhodně vygeneruje již seřazené pole. Pravděpodobnost tohoto jevu u střední obtížnosti je pro představu asi $9{,}559*10^{-14}$. Ač je tato pravděpodobnost prakticky nulová, může daná situace nastat.

Prirad_spravne

Pokud je součet počtu inverzí v permutaci a Manhattanské vzdálenosti lichý, provádí *prirad_spravne* procedury *nahodne_prirad* a *zkouska* tak dlouho, dokud není zajištěna a prokázána sudost onoho součtu. Pokud by byl součet nula, proces je taktéž opakován. Využívá cyklus *repeat*.

Loading

Načítání je znázorněno cyklicky (cyklus *for*) se měnícími řetězci symbolů ‘O o o‘, ‘o O o‘ a ‘o o O‘ v pravidelném intervalu. Každá změna je díky jednotkové proceduře *beep* provázena elementárním zvukem.

Jmenouzi

Po načtení hry je uživatel vyzván k zadání svého jména či přezdívky. Program čeká na klávesu, kterou uživatel stiskne. Může se jednat o číslici, velké/malé písmeno, backspace nebo enter. Stiskne-li hráč backspace jako první znak, ozve se zvuk systémové chyby, stiskne-li jako první enter, program nijak nezareaguje.

Po stisknutí klávesy enter, pokud se kurzor nachází na jiné než původní pozici, se všechny zadané tisknutelné znaky sečtou a uloží jako řetězec znaků (typ string). Počet znaků je omezen na 16.

V této proceduře jsou používány hojně cykly *for*, podmínky *if*, systémové procedury *GotoXY* a větvící příkaz *goto*.

Tutorial

Program vede hráče k dohrání hry „na zkoušku“. Zkoumá, zda stiskl požadované klávesy pro validní tah, a seznamuje hráče s aplikací. Rozmístění čísel není náhodné, ale je pevně zařazeno do procedury. Opět převážně cyklus *repeat* společně s funkcí *ReadKey*.

Hlmenu

Nejdelší a nejrozvětvenější procedura, která by mohla sloužit po minimální úpravě jako hlavní program. Při programování se mnohdy stane, že je potřeba použít jednu proceduru ve druhé a naopak. Řešením může být například deklarace procedury do procedury, jako jsem to musel udělat v tomto podprogramu.

Podprocedura *koliktahu* udává počet tahů, které hráč učinil. Při překročení meze se spustí další podprocedura procedury *hlmenu,* a to *nap_gameover*, který vypíše velký nápis „KONEC HRY“.

V proceduře samotné se vypíše hlavní menu, v němž se uživatel pohybuje pomocí šipek či kláves W a S. Pohyby v menu i podmenu pracují převážně na základě větvení *case of* a funkce *ReadKey*. Aktuální volbu reprezentují čísla, která se mění v závislosti na stisknuté klávese. Díky větvení *case of* se při určité volbě vypíše určitá položka v menu žlutou barvou. Potvrzení volby klávesou enter buď spustí hru, ukončí program nebo vymaže obrazovku pomocí příkazu *ClrScr* a vypíše dané podmenu.

Validita tahů ve hře jako takové se zkoumá pomocí souřadnic, na které se může prázdné pole pohnout. Při snaze o invalidní tah program nezareaguje. Stisknutí pohybové klávesy zároveň zvyšuje počet tahů v proceduře *koliktahu* o 1. Program cyklicky kontroluje, zda není pole již seřazené. Pokud se číslo nachází na svém místě, systém to zaznamená a zvýší hodnotu proměnné *g* o 1. Jakmile se hodnota *g* rovná dané mezi (9/16/25), je pole seřazeno. Uživatelův výsledek je porovnáván s ostatními výsledky a případně zařazen do textového dokumentu a „síně slávy“.

V podmenu položky „NASTAVENÍ A TUTORIÁL“ se nachází možnost vypnout/zapnout zvuky. Všechny nesystémové zvuky funkce *beep* mají určitou proměnnou jako svou frekvenci, která je při vypnutí zvuků vynulována. Tento stav se zkoumá díky podmínkovému příkazu *if*. V případě zapnutí zvuků je navrácena původní frekvenční hodnota.

V podmenu položky „OBTÍŽNOST“ volíme obtížnost celé hry. Pokud před spuštěním hry jako takové obtížnost nenastavíme, spustí se automaticky klasická verze 15 Puzzle (4x4 polí). Při potvrzení volby jiné obtížnosti se přiřadí určité proměnné číslo 3/4/5 a indikátor výběru se rozsvítí zeleně u zvolené položky. Tyto výběry a indikátor pracují díky příkazu *case of*.

Hlavní program

V hlavním programu se procedurou *assign* přiřadí určité textové dokumenty proměnným typu text, ohraničí se prostor aplikace a proběhne spuštění procedury *hlmenu*, ve které se odehrává naprostý zbytek akcí.

3 Závěr

Vytvoření hry trvalo o poznání déle, než jsem přepokládal. Při volbě tématu jsem například nevěděl, že celých 50 % rozmístění čísel nepovede k řešení. Jednalo se pravděpodobně o největší překážku, na kterou jsem během tvorby narazil.

Původně jsem chtěl pro hru vytvořit v Pascalu i soundtrack, to ovšem nebylo možné, jelikož Pascal neumí provádět dvě samostatné procedury zároveň (hrát soundtrack a zároveň kontrolovat tahy hráče atd.).

Až na problikávání, kterému se kvůli neustálému opakování vypisování nešlo vyhnout (hlavně u obtížnosti 5), jsem s výsledkem svého snažení spokojen. Myslím, že se mi v programu povedlo zkombinovat funkčnost a zároveň určitou estetiku.

Na celé aplikaci jsem pracoval od prosince do března. Postupně jsem si uvědomoval, že některá má předchozí řešení byla možná příliš zdlouhavá nebo složitá, záměrně jsem je ovšem nijak nepřepisoval. Na celém programu je díky tomu vidět, jak s přibývajícím časem rostl nejen počet řádků programu, ale i mé schopnosti a zkušenosti. Jsem si vědom, že bych program mohl zjednodušit například zúžením datových typů (integer → byte) nebo využitím méně proměnných, které se mohly přepisovat.

Čeho bych chtěl do budoucna dosáhnout, je napsání algoritmu, který by k dané permutaci čísel vypsal, za jaký minimální počet tahů lze hru při maximální efektivitě hráčova počínání dohrát.

Mnoho informací o systémových jednotkách a předdefinovaných podprogramech jsem si musel samozřejmě vyhledat, ale celkově jsem se snažil využít maximum svých znalostí z výuky jako takové, a to jak z informatiky a programování, tak z matematiky. Čas strávený nad programováním hodnotím jako dobře využitý. Řešení problémů a překonávání překážek mne bavilo.

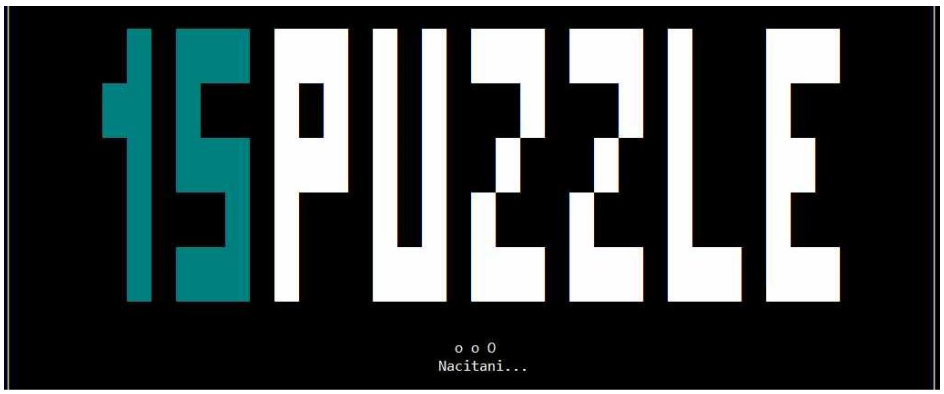
4 Uživatelský manuál

Oddíl „Uživatelský manuál“ slouží primárně k seznámení uživatele s výslednou podobou aplikace a pomáhá uživateli orientovat se v jejích částech.

4.1 Instalace a spuštění hry

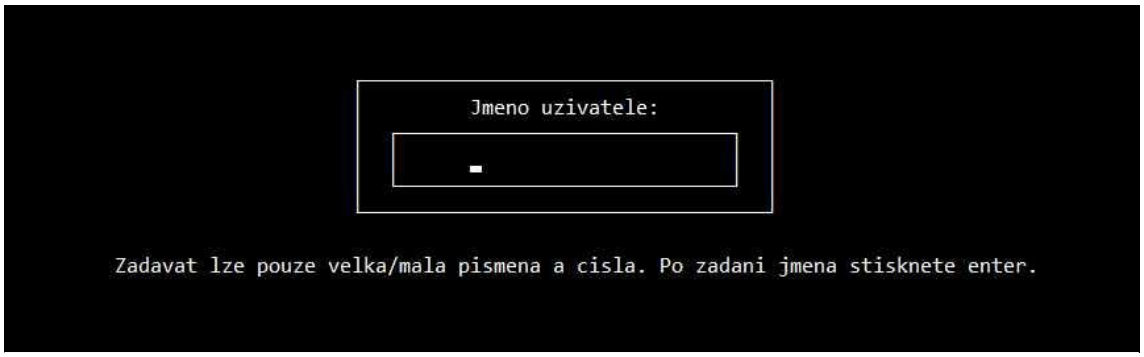
Soubor s aplikací by měl být před spuštěním z externího datového disku vyjmut nebo zkopírován na místní disk počítače (C:). Hru doporučuji hrát výlučně v takzvaném „full screenu“ – otevření na celou obrazovku. Na aplikaci by neměla za žádných okolností být aplikována funkce „split screen“ – rozdělení obrazovky na dvě části. Textové dokumenty nejsou určené pro přímou interakci s uživatelem.

První, co uživatel při spuštění aplikace PUZZLE15 uvidí, je načítací obrazovka. Postupně se vypíše název celé hry, pod ním probíhá načítání.



Obrázek č. 2 – Načítací obrazovka

Načítání je dokončené v momentě, kdy se zobrazí hlášení „Stisknete mezernik.“. Následkem stisknutí požadované klávesy je zobrazení tabulky s výzvou pro napsání jména či přezdívky hráče. (obrázek č. 3) Stiskem enteru se uživatel přesune do hlavního menu.



Obrázek č. 3 – Zadání uživatelova jména

4.2 Hlavní menu

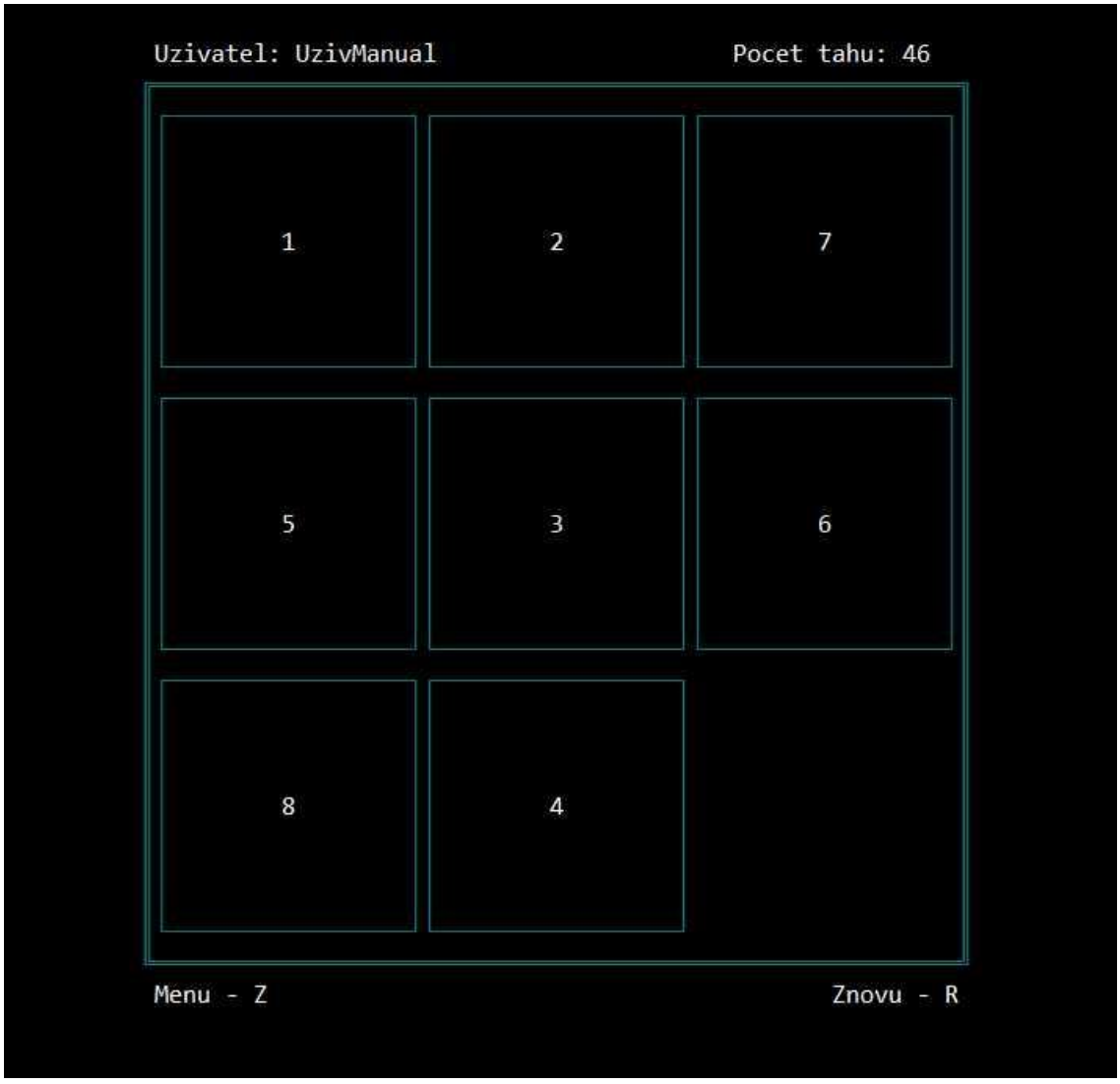
V hlavním menu (obrázek č. 4) se uživatel pohybuje pomocí šipek nahoru (alternativou je klávesa W) a dolů (klávesa S). Žlutou barvou se zvýrazní ta položka, která bude po stisku klávesy enter vybrána.

Obrázek č. 4 – hlavní menu

4.2.1 „Nova hra“

Při výběru položky „Nova hra“ dochází ke spuštění samotné hry dle zvolené obtížnosti. Nenavštíví-li uživatel jako první záložku „Obtížnost“, kde danou volbu provede, automaticky se spouští hra s 16 hracími kameny. Hráč pohybuje políčky pomocí šipek nebo kláves WSAD. Při stlačení jedné z těchto kláves dochází k přemístění toho políčka, které se daným směrem může pohnout. Pokud takové neexistuje, program nereaguje.

Cílem hry je seřadit hrací kameny vzestupně od 1 do nejvyššího čísla (9/16/25). Hráč má možnost návratu do hlavního menu stisknutím klávesy Z. Pokud chce hru resetovat nebo změnit náhodné uspořádání čísel, může zmáčknout klávesu R. V levém horním rohu je napsáno hráčovo jméno, v pravém horním rohu naopak počet tahů, které už učinil. Hráč nesmí přesáhnout 9999 tahů u žádné z obtížností. Pokud se tak stane, bude mu zamezeno v hraní pokračovat a oznámeno, že nesplnil úkol.



Obrázek č. 5 – lehká obtížnost (3)

4.2.2 „Obtiznost“

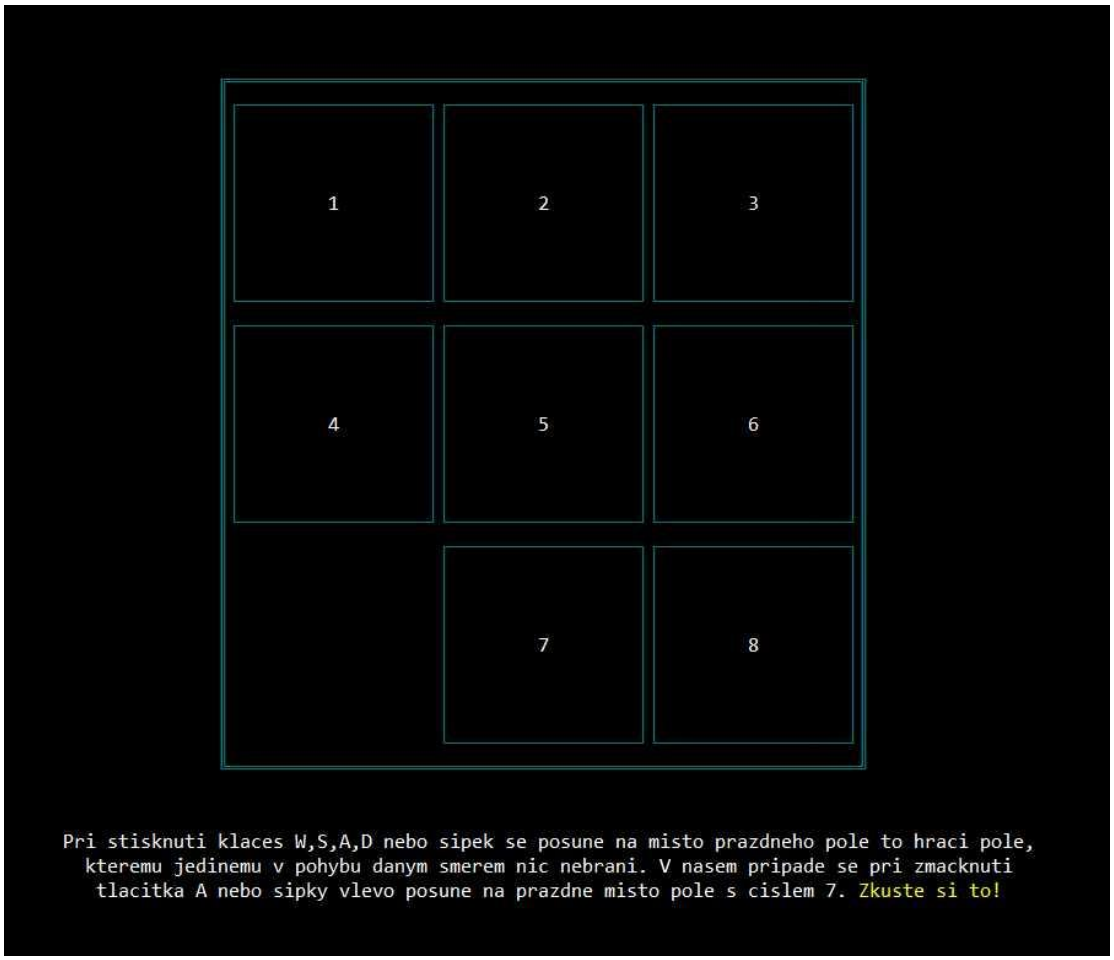
Touto volbou se dostane uživatel k podmenu, které mu dává na výběr ze tří možností obtížnosti – lehká (obrázek č. 5), střední a těžká. Zeleně svítící indikátor značí, která obtížnost je aktuálně vybrána. Potvrzením jiné volby se indikátor přemístí. Vybráním položky „zpet“ se hráč přesune zpět do hlavního menu.

4.2.3 „Nastaveni a tutorial“

Pod touto záložkou se skrývá také podmenu. Konkrétně možnosti „tutorial“, „změna uzivatele“, „vypnout/zapnout zvuky“ a „zpet“. Volba zpět funguje analogicky u všech dalších podmenu, které se v programu vyskytují.

Tutoriál (obrázek č. 6) hráče nechá hrát krátkou, systémem řízenou a komentovanou hru při nejnižší obtížnosti. Uživatel by se měl seznámit se základními mechanikami hry.

Změna uživatele hráče přesměruje na obrazovku z obrázku č. 3. Zde má možnost vymyslet si úplně jiné jméno, případně zadat znovu stejné. Návrat ze zadávací obrazovky je do hlavního menu. Při pohybu v menu slyší uživatel zvuky. Ty může vypnout (zapnout) potvrzením v pořadí třetí volby v tomto podmenu.



Obrázek č. 6 - tutoriál

4.2.4 „Sin slavy“

Do síně slávy jsou ukládány výsledky deseti nejlepších hráčů z vybrané obtížnosti. (obrázek č. 7) Aby zde přibyl nový záznam, musí být hra dokončena v menším počtu tahů, než za který to zvládl hráč na desáté pozici (v našem případě dohrát hru za méně než 2445 tahů, za které to zvládl Lukas). Každá z položek v podmenu odkazuje na síň slávy jedné z obtížností.

Obrázek č. 7 – síň slávy

4.2.5 „Konec hry“

Vybráním položky „konec hry“ se program ukončí. Buďte opatrní, aplikace se nedotazuje, zda ji chcete opravdu vypnout.

5 Seznam použité literatury

Citované zdroje:

1. ENG Wikipedia. *Wikipedia* [online]. [cit. 2019-04-09]. Dostupné z: https://en.wikipedia.org/wiki/15_puzzle

2. Shluková analýza. *Karlin.mff.cuni* [online]. [cit. 2019-04-09]. Dostupné z: http://www.karlin.mff.cuni.cz/~zichova/PRFUK/Kapitola7.doc

Obrázky:

Obr. *Manhattanská x Euklidovská metrika* [online]. [cit. 2019-04-09]. Dostupné z: https://en.wikipedia.org/wiki/Taxicab_geometry

Informace:

CZ Wikipedie. *Wikipedia* [online]. [cit. 2019-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/Patn%C3%A1ctka

Soubory. *jhejhal.wz* [online]. [cit. 2019-04-09]. Dostupné z: http://jhejhal.wz.cz/prg/pas_soubory.htm

Manhattanská vzdálenost. *Wikipedia* [online]. [cit. 2019-04-09]. Dostupné z: https://en.wikipedia.org/wiki/Taxicab_geometry

Wiki FreePascal. *Freepascal* [online]. [cit. 2019-04-09]. Dostupné z: http://wiki.freepascal.org/

ASCII. *Asciitable* [online]. [cit. 2019-04-09]. Dostupné z: http://www.asciitable.com/

FreePascal. *Freepascal* [online]. [cit. 2019-04-09]. Dostupné z: https://www.freepascal.org/