VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

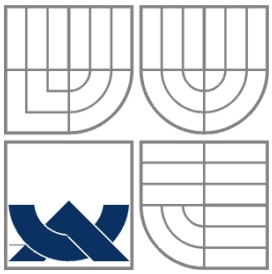
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULITMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GRAFICKÝ VÝUKOVÝ SYSTÉM

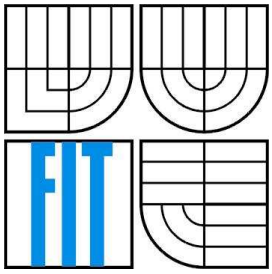
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PETER FARBIAK
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVEJ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GRAFICKÝ VÝUKOVÝ SYSTÉM

GRAPHICAL EDUCATIONAL SYSTEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — PETER FARBIAK
AUTHOR

VEDOUCÍ PRÁCE — ING.VÍT ŠTANCL
SUPERVISOR

BRNO 2009

Abstrakt

Cílem teoretické části této bakalářské práce je uvést čtenáře do základů psychologie učení, principů programování a algorimtizace, tvoření a stavby grafického užívatelského rozhraní. Praktická část se zaměřuje na vytvoření programu, který pomocí přidávání grafických objektů demonstruje principy základních algoritmických struktur.

Abstract

The goal of the theoretical part of this Bachelor's thesis is to introduce basics of learning psychology, principles of programming and algorithms and creating of graphical user interface to reader. Practical part is about creating program which demonstrates principles of basic algorithmic structures by adding graphical objects.

Klíčová slova

Grafický, základ, programování, algoritmus, učení, výuka, GUI, objekt.

Keywords

Graphical, basic, programming, algorithm, learning, tuition, GUI, object.

Citace

Farbiak Peter: Grafický výukový systém, bakalářská práce, Brno, FIT VUT v Brně, 2009

Grafický výukový systém

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Víta Štancla. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Farbiak Peter
18.5.2009

Poděkování

Týmto by som chcel poďakovať vedúcemu mojej bakalárskej práce Ing. Vítovi Štanclovi za jeho ochotu, trpezlivosť, a odbornú pomoc pri vypracovaní tejto práce.

Obsah

1 Úvod

Názov tejto bakalárskej práce je Grafický výukový systém. Od tohto názvu sa teda budú odvíjať aj jednotlivé kapitoly práce. Jej úlohou je vytvoriť program s kvalitným GUI, ktorý bude demonštrovať základné algoritmické štruktúry.

Práve preto musíme nahliadnuť do psychológie didaktických vied a pokúsiť sa nájsť najlepšiu cestu k výsledku, ktorý bude po stránke výuky čo najefektívnejší. Pre lepšie pochopenie akým spôsobom sa človek učí, ako má prebiehať výuka je vhodné si vysvetliť niektoré pojmy a zistenia z odvetvia psychológie konkrétne teda učenia a výuky. Toto je zahrnuté v kapitole Učenie a výuka (2). V jej podkapitolách potom budeme rozoberať druhy učenia (2.2), faktory, ktoré môžu proces učenia ovplyvňovať (2.3), rovnako podmienky, ktoré je pri učení potrebné splniť (2.4) a nakoniec tohto celku samotný proces učenia (2.5). Na konci tejto kapitoly by sme mali prísť k výsledku a rozhodnúť sa akým spôsobom budeme ďalej postupovať pri výbere koncepcie programovej časti tejto práce.

Práca má za úlohu demonštrovať a naučiť základným princípom programovacích štruktúr. Jej úloha je vysvetliť ako fungujú princípy programovania. Preto si v ďalšej kapitole (3) povieme o základoch programovania a algoritmizácii. V tejto časti tak budeme hovoriť práve o tom, čo má program za úlohu naučiť a teda pochopenie logiky programovania a algoritmov. Hovoriť sa tu bude o stratégiách riešení problémov (3.1), princípoch vyšších programovacích jazykov (3.3) a zoznámime sa so samotným riadením chodu programu (3.4), kde oboznámime čitateľa so základnými programovacími štruktúrami a pricípmi. V závere kapitoly sa budeme môcť rozhodnúť, ktoré programovacie štruktúry budú pre našu vec tie najpodstatnejšie. A práve vďaka tejto kapitole si určíme čo bude program demonštrovať a vyučovať.

Hneď prvé slovo názvu práce je slovo grafický a neoddeliteľnou súčasťou programovej časti tejto práce bude príjemné a jednoduché GUI. Z tohto dôvodu budeme v ďalšej kapitole (4) práce pojednávať o stavbe grafického užívateľského rozhrania. Povieme si ako na to, ktoré zásady je vhodné dodržiavať a predvedieme si krátku názornú ukážku tvorby jednoduchého programu s GUI. V tejto kapitole sa na začiatok dozvieme niečo o ďalších užívateľských rozhraniach (4.1) aby sme si pripomenuli akým prelomovým grafické rozhranie vôbec je a následne sa vrhneme na to podstatné a pojednáme o prvkoch, ktoré GUI využíva (4.2) a samozrejme o samotnej tvorbe GUI (4.3). Táto kapitola nás prevedie základmi stavby grafického rozhrania a vysvetlí ako môžeme dosiahnuť kvalitný výsledok, ktorý je pre túto prácu nepostrádateľný.

V praktickej časti (5) práca rozoberá samotný doprovodný program. Na úvod je zhrnutá charakteristika programu (5.1), tu sú zdôvodnené rozhodnutia pre implementáciu programu. Prečo program vyzerá tak ako vyzerá a obsahuje funckie, ktoré obsahuje sa dočítate v nasledujúcej kapitole (5.2). Po týchto informáciách prichádza popis samotnej implementácie (5.3). Tu sa práca zaoberá tým ako je program stavaný a aké prostriedky sú pre to použité. V nasledujúcej časti kapitoly o implementácii je uvedné akým spôsobom sú riešené dané funkcie programu a tiež tu je uvedené ako sú postavené jednotlivé komponenty. Na záver tejto kapitoly samozrejme nájdete niektoré ukážky vzhľadu aplikácie a popis jeho behu od spustenia až po ukončenie.

2 Učenie a výuka

Človek sa učí od narodenia až do smrti. V počiatkoch svojho života sa učí sedieť, chodiť, rozprávať, hygienickým návykom a podobne. Neskôr, po dovŕšení určitého veku začne navštevovať materskú školu, kde sa učí sociálnym vzťahom v kolektíve. Po tomto období prichádza do školy. Tu sa začína cielené získavanie vedomostí nevyhnutných pre život v spoločnosti. Písanie, čítanie, počítanie dávajú základ všetkým ostatným oborom a odvetviam do budúcnosti.

V širšom slova zmysle môžeme učenie chápať ako získavanie individuálnej skúsenosti v priebehu života jedinca. V užšom potom ako zámerné a systematické nadobúdanie vedomostí, zručností, návykov, spôsobov správania pod vedením v organizovaných podmienkach. [2]

2.1 Druhy učenia

V psychlogickej literatúre môžme nájsť viacero delení učenia. Jedno z nich, ktoré vypracovali v roku 1967 Linhrat a Maršálová delí druhy učenia na tieto [4]

- podmieňovanie
- percepčno – motorické
- verbálne
- pojmové
- učenie riešením problému

Učenie podmieňovaním
Základnou metódou učenia je utváranie podmienených reflexov. Vytvárajú sa dočasné spoje medzi asociáciami. Výsledkom tohto druhu učenia sú spevnené dočasné spoje. [2] [4]
Skinnerov výskum – Myš je uzavretá v priestore s pákou. Keď stlačí páku do misky sa jej prisunie potrava. Spočiatku krysa stláča páku náhodne. V ďalšom priebehu ju stláča cielene bez chaotických reakcií. [3]

Senzomotorické (percepčno-motorické)učenie
Prezentuje osvojovanie si pohybových operácií, rozličné manuálne zručnosti a návyky rozmanitých druhov spoločenskej činnosti. [4]

Verbálne učenie
Patrí k najčastejším druhom učenia. Osvojujú sa slovné odpovede určitého systému a spája sa s pamäťovým učením. [4]

Pojmové učenie
Súvisí s verbálnym učením, no odlišuje sa od verbálneho, pretože tu nejde len o vytváranie mechanických asociácií, ale aj logických operácií. Môžeme ho rozdeliť do dvoch fáz.

1. Tvorenie pojmov
2. Osvojovanie pojmov

Napomáha pri rozvoji abstraktného myslenia. [4]

Učenie riešením problému
Vyžaduje zvýšenú aktivitu subjektu, pozitívne ovplyvňuje efektívnosť procesu zámerného pôsobenia. [4]

2.2 Faktory ovplyvňujúce učenie

2.2.1 Motivácia

Motivácia je súhrn činiteľov, ktoré podnecujú, smerujú a udržujú správanie človeka. Motivácia je zákon a podmienka správneho učenia. Platí, že výkon v učení je daný nielen schopnosťami jedinca, ale tiež motiváciou. Pokiaľ chceme niečo dosiahnuť musíme sa dostatočne motivovať.
Formy motivácie [4]

- radosť a záujem
- odmena
- odmena primeranou samochválou

Vždy sa treba vyvarovať negatívnym motivačným stimulom ako je strach, stres, napomenutia.

2.2.2 Koncentrácia

Koncentrácia spočíva v prísnej voľbe záujmov a na obmedzení sa na podstatu veci. Ak sa chceme správne koncentrovať musíme vylúčiť vyrušovanie. Koncentrácia je dôležitá pre veľký prísun informácií. V prípade, že jedinec nie je schopný koncentrácie mal by ju získať cvičením. Je dôležité naučiť sa selektovať dôležité informácie od nepodstatných.

2.2.3 Pamäť

Pamäť je nevyhnutnou podmienkou učenia. Jej skvalitňovaniu musíme venovať pozornosť. Pamäť sa v období mladšieho dospelého veku zhoršuje a tak sa vytvára požiadavka k tréningu pamäti. Pamäť funguje súčasne v troch rovinách [4]

- ultrakrátkodobá pamäť – Premieňa zmyslové vnemy do energetického poľa a udržiava ich v rozpätí asi 30 sekúnd až 5 minút. Informácia sa preveruje ešte predtým ako je zatriedená do veľkého mozgu. Pokiaľ je zatriedená ako nepríjemná môže byť zdržaná a prerušená.
- Krátkodobá pamäť – Uskladňuje všetky informácie potrebné pre porozumenie zmyslu. Krátkodobá pamäť nám dovoľuje napríklad čítanie. Je veľmi poruchová a nové vnemy sa rýchlo prekrývajú staršími informáciami.
- Dlhodobá pamäť – Sú to informácie, ktoré prešli všetkými inštanciami, má neohraničenú schopnosť vnímania, no počet informácií, ktoré sa môžu uskladňovať k danému časovému bodu nie je veľký. Ak si chceme niečo zapamätať musíme to urobiť v prvých desiatich minútach aby sme zachovali informácie v dlhodobej pamäti.

2.3 Podmienky učenia

Učenie človeka sa uskutočňuje za určitých podmienok. Tie buď proces učenia zefektívňujú alebo naopak učeniu prekážajú. V psychológii môžeme nájsť viacero kategorizácií podmienok, napríklad na podmienky vnútorné a vonkajšie. [2]

Medzi vnútorné podmienky môžeme zaradiť [2]

- prítomný stav jedinca, jeho únavu a pozornosť
- biologické podmienky jedinca a jeho vývinu
- motiváciu k učeniu
- schopnosti a dosiaľ osvojené vedomosti a zručnosti
- procesy poznávania a učenia
- črty osobnosti vrátane morálneho charakteru

Vnútorné podmienky neoddeliteľne súvisia s podmienkami vonkajšími. Medzi tie patria napríklad [2]

- rozsah učiva
- prostredie v ktorom sa jedinec učí
- vybavenie
- spracovanie učiva

2.4 Proces učenia

Individuálne a skupinové učenie [2]
Pozitívny vplyv skupinového učenia je v tom, že v skupine si jednotlivec môže osvojiť úspešnú metódu učenia iných členov. Každý hneď dostáva spätnú informáciu o dosiahnutom výsledku istej časti učiva, čím motivuje do ďalšieho učenia.
Skupinové učenie môže prebiehať dvoma formami

A) formou súťaže – Tento spôsob je z pohľadu sociálnych vzťahov negatívny. Dôvodom je, že v kolektíve môžu vzniknúť hádky, neochota pomôcť, narušené priateľské vzťahy. Na druhej strane víťaz je motivovaný ďalej sa vzdelávať a snažiť sa ešte viac.
B) formou spolupráce – Spoločná úloha upevňuje vzťahy v kolektíve. Hrozí tu však negatívny vplyv toho, že slabší jedinci budú využívať silnejších k splneniu svojich úloh.

Psychohygienické podmienky učenia [2]
Najdôležitejšou podmienkou tejto kategórie je nesporne primerané zaťaženie jedinca. K preťažovaniu môže dôjsť zo strany blízkeho okolia alebo požiadavok, ktoré sú na jedinca kladené. Miera zaťaženia musí byť nastavená správne tak, aby sa rozvíjala funkčná zdatnosť psychiky jedinca a tým rozvoj jedinca.

Denný poriadok učenia [2]
Pri učení sa odporúča dodržovať vlastný rytmus. Výkon, ktorým jedinec nadobúda vedomosti môže byť zvýšený práve určitým rozvrhom učenia. Učenie bude mnohokrát efektívnejšie v prípade ak bude prebiehať denne v určitom čase ako keby prebiehalo rovnako dlhú dobu v chaoticky zvolených intervaloch. Toto súvisí so zvyknutím si organizmu na psychickú záťaž a jeho lepšou odozvou naň. Dôležitý pri tomto je samozrejme aktívny a pasívny oddych.

Hygienické potreby pre učenie [2]

- správne osvetlenie – Príliš jasné a silné svetlo neprimerane namáha oči, ktoré sa potom rýchlejšie unavia. Slabšie svetlo očiam neškodí ak sa nenamáhajú.
- Primeraná izbová teplota – Ak je teplota príliš vysoká človek stráca záujem o učenie, je ospalý a lenivý. Naopak ak je teplota príliš nízka človek sa sústredí na zimu.
- Správne prúdenie vzduchu – Pri učení nesmie chýbať čerstvý vzduch a kyslík. V dusnom prostredí sa organizmus neokysličí tak dobre ako pri čerstvom vzduchu. Mozog pri spávnom okysličení reaguje rýchlejšie.
- Odstraňovať hluk – Ruch odvádza pozornosť od učenia.
- Farby prostredia v ktorom sa učíme – Vplývajú na podvedomie jedinca.

2.5 Zhrnutie

Učenie je v živote jedinca nenahraditeľný proces. Je dôležité získať správne návyky a veľmi dôležité je získať motiváciu a chuť učiť sa. Jednou z najefektívnejších metód je najmä v mladom veku učenie hrou. Tento druh učenia poskytuje motiváciu v priebehu učenia a to konkrétne radosťou z hry a po ukončení radosťou z výhry. Toto veľmi napomáha k tomu, že jedinec (dieťa) sa k hre a tým pádom k učeniu bude vždy rád a znovu vracať. Pri hrách, ktoré sa venujú riešeniu určitých problémov alebo hrách, ktoré potrebujú aktívne uvažovanie o nasledujúcich krokoch sa tiež skĺbia dva druhy učenia. Učenie pojmové spoločne s učením riešením problému. Týmto sa rozvýja logické myslenie a zvyšuje sa efektívnosť práce jedinca.
Pre nás je hodnotný poznatok, že túto bakalársku prácu a konkrétne jej programovú časť bude vhodné implementovať zábavným, motivačným spôsobom. Tak aby výuka nespôsobovala psychické vyčerpanie a netlmila pozornosť jedinca, ale naopak aby štýlom, akým bude tvorená napomáhala k aktívnemu prístupu a tým efektívnejšiemu nadobúdaniu poznatkov. Najlepšou variantou sa podľa môjho názoru teda javí implementovať daný problém ako interaktívnu hru.

3 Základy programovania a algoritmizácia

Čo je to vlastne programovanie? Programovanie znamená navrhovanie, zápis, úpravy a testovanie programov. [1]

Programovanie zahrňuje nasledujúce etapy [1]

- Formulácia úlohy
- Analýza úlohy
- Návrh riešenia
- Zostavenie algoritmu riešenia
- Kódovanie programu
- Odladenie
- Optimalizácia

Formulácia úlohy
Je prvou etapu tvroby programu. Zaoberáme sa v nej požiadavkami zákazníka, získavame ich, analyzujeme a špecifikujeme. Snažíme sa zo zákazníkových nejasných predstáv vykresať čo najšpecifickejšie, štrukturované a konzistentné požiadavky.

Analýza úlohy
Podstata tejto úlohy spočíva v štúdii realizovateľnosti úlohy, či je úloha dostatočne špecifikovaná. Vytvárajú sa prvé riešenia a súčasťou tejto etapy je zároveň tvorba plánu akceptačných testov (testy na základe ktorých zákazník akceptuje produkt).

Návrh riešenia
V etape návrhu riešenia rozkladáme problém na mnoho jednoduchších podproblémov. Rovnako tu špecifikujeme komponenty pre ich riešenie a vytvárame návrh metód riešenia jednotlivých podproblémov.

Zostavenie algoritmu riešenia
Je to návrh datových štruktúr obsiahnutých v programe a zápis algoritmu. Algritmus je možné zapísať ako vývojový diagram alebo tiež pomocou programovacieho jazyka v kombinácii s prirodzeným jazykom.

Kódovanie programu
Kódovanie je zápis zdrojového textu úlohy v programovacom jazyku.

Odladenie
V tejto etape je program podrobený sade testov. Výsledky testov sú porovnávané so špecifikáciou a overuje sa tak správnosť programu.

Optimalizácia
Inými slovami je to vylepšovanie programu. Jedná sa o zlepšenie jeho vlastností (rýchlosť, pamäťová náročnosť...) bez zmien vstupu a výstupu.

V predchádzajúcom texte sme často mohli vidieť slovo algoritmus. Čo je algoritmus si vysvetlíme teraz.

Algoritmus je konečná usporiadaná množina úplne definovaných pravidiel pre vyriešenie nejakého problému. Je to presne definovaná postupnosť krokov, ktorých prevedením pre každé prípustné vstupné hodnoty získame po určitom počte krokov odpovedajúce výstupné hodnoty.[1] Inými slovami algoritmom rozumieme postup, ktorým sa dostaneme k vyriešeniu problému.

Algoritmus je teda zápis postupu resp. postupnosť krokov, ktorým sa dostaneme k riešeniu problému. V prvom rade však musíme riešenie nájsť.

Vlastnosti algoritmov [1]
Algoritmus má 4 základné vlastnosti.

- Rezultatívnosť (konečnosť)
- Hromadnosť (obecnosť)
- Determinovanosť
- Efektívnosť

Rezulatívnosť' hovorí, že algoritmus skončí po určitom počte krokov. Teda, že algoritmus je konečný. Táto vlastnosť zabráni cyklickému opakovaniu algoritmu tzv. nekonečnému cyklu.
Hromadnosť' znamená, že algorimus je všeobecne použiteľný pre celú triedu úlohu rovnakého druhu. To znamená, že program musí prebehnúť správne pre všetky zadané hodnoty, ktoré spĺňajú vstupné podmienky.
Determinovanosť' hovorí o tom, že algoritmus musí byť zadaný vo forme konečného počtu jednoznačných pravidiel. Znamená to, že pri každom kroku algoritmu musí byť presne definované ako má program pokračovať ďalej. Program nesmie mať žiadnu „slepú cestu". [1]
Efektívnosť' nemá na beh programu vplyv. Zabezpečuje, čo najlepšie využitie zdrojov v čo najkratšom čase. Zabezpečuje aby program nevykonával žiadne zbytočné operácie.

3.1 Stratégie riešenia problémov

V princípe môžme stratégie riešenia problémov rozdeliť na 3 spôsoby

- Dekompozícia
- Abstrakcia
- Modulárne programovanie (použitie dekompozície a abstrakcie)

Dekompozícia
Dekompozícia je rozloženie jedného väčšieho problému na radu menších a jednoduchších. Ďalšia časť dekompozície je nájdenie takého hierarchického usporiadania vďaka ktorému je možné zložité akcie zapísať radou jednoduchších a tie prípadne na ešte jednoduchšie. Takýmto spôsobom postupujeme až do momentu kedy máme dostupné prostriekdy na vykonanie všetkých operácií.

Abstrakcia
Je zjednodušenie problému dočasným zanedbaním niektorých detailov. Týmto je zložitý problém možné riešiť po častiach. Oddelí sa tak jadro problému od detailov.

3.2 Vyjadrovanie algoritmov

Slovný popis používa na vyjadrenie algoritmu súvislý text. Jeho čítanie vyžaduje veľkú pozornosť, čo odvádza od sústredenia sa na riešený problém. Z tohto dôvodu slovný popis nie je vhodný pre riešenie technických problémov.

Vývojové diagramy zobrazujú postupnosť operácií a obsahujú

- symboly pre vlastné operácie spracovania
- symboly spojenia, ktoré indikujú tok riadenia
- zvláštne symboly pre uľahčenie čítania a zápisu vývojového diagramu

Štruktúrogramy

Rozhodovacie tabuľky boli definované najmä pre algoritmizáciu úloh so zložitým rozhodovaním v oblasti spracovania hromadných dát. Sú nástrojom k vyjadreniu komplexnej logiky rozhodovania relatívne jednoduchým spôsobom. [1]

Programovacie jazyky sú dôsledne formalizované algoritmické jazyky, určené pre zápis algoritmu pre počítač. Zápis algoritmu v programovom jazyku nazývame program. [1]
Existujú stovky programovacích jazykov. Niektoré z nich sú univerzálne iné zamerané na určité typy problémov.
Syntax popisuje formálnu štruktúru jazyka. Definuje kľúčové slová, identifikátory, čísla a ďalšie programové entity a určuje spôsob ako ich je možné kombinovať.[1] Na základe pravidiel syntaxe sa dá určiť, či zápis je zápisom v danom programovacom jazyku.
Sémantika určuje logický význam jednotlivých výrazov jazyka. Z nej teda plynie aký má daná konštrukcia význam.[1]

3.3 Princípy vyšších programovacích jazykov

Všetky informácie, ktoré sú na počítači spracovávané je potrebné uložiť v podobe, ktorej počítač rozumie. Takejto forme sa hovorí dáta. Dáta sú opakovane interpretovateľné fomalizoavné podoby informácií.[1] Dáta sú ďalej spracované až na jednotlivé bity, čiže postupnosť jednotiek a núl. Písať programy v takejto forme by bolo krajne nepraktické a náročné, preto vznikol dátový typ. ***Dátový typ*** nám určuje akým spôsobom budeme konkrétnu sadu bitov používať a aké operácie je s nimi možné vykonať. [1]

3.3.1 Dátové štruktúry

Pre mnoho aplikácií je výber dátovej štruktúry najvýznamnejším rozhodnutím celej implementácie. Dátové štruktúry v spojení s operáciami, ktoré nad nimi môžeme vykonávať nazývame dátový typ. Programy spracovávajú informácie odvodené z reálneho sveta. Z toho sú

odvodené základné prvky, ktoré používame pre popis informácií. Sú to písmená a čísla. Rozsah číselných dátových typov a presnosť s ktorou je možné pracovať v prípade racionálnych čísel je daná počtom bitov, ktorými číslo reprezentujeme.

Datová štruktúra môže byť [1]

- homogénna – a to vtedy ak všetky komponenty štruktúry sú jedného typu
- heterogénna – ak komponenty sú rôznych typov
- statická – nemôže meniť v čase výpočtu počet komponent ani spôsob usporiadania
- dynamická môže meniť v čase výpočtu počet komponent a spôsob usporiadania

3.3.2 Dátové typy

Dátový typ je množina hodnôt a množina operácií nad týmito hodnotami.[1] Každý objekt vo vyšších programovacích jazykoch patrí práve jednému dátovému typu. Ten sa zavádza v definícii a deklarácii premennej.

Vlastnosti dátových typov

Datový typ je určený [1]

- názvom (int, char, float, meno...)
- množinou hodnôt (1, o, 3.14, Peter...)
- množinou operácií – každý operátor očakáva operandy stanoveného typu a vracia výsledok stanoveného typu
- množinou atribútov – ktoré vlastnosti dátového typu sú programovo dosiahnuteľné

Jednoduché dátové typy

Nazývajú sa jednoduché pretože patria k základným typom dát s ktorými je možné manipulovať. Zložitejšie dátové typy sú ich kombinácie. Predstavujú základ všetkých výpočtov. Patria sem znaky, čísla a logické hodnoty. Nad všetkými jednoduchými dátovými typmi sú definované operácie priradenia, ekvivalencie a usporiadania.[1]

Štandardné jednoduché dátové typy

Integer (int) prezentuje množinu celých čísel, ktorá je v danom programovacom jazyku k dispozícii ako štandardný dátový typ.
Real (float) reprezentuje reálne čísla. Narozdiel od typu **int** sú výsledky s operandami typu float len približné. Na reálnej osi ľubovoľne malý inerval obsahuje nekonečne mnoho reálnych čísel. V počítači reprezentuje **float** konečnú podmnožinu racionálnych čísel.
Znak (char) zahŕňa usporiadanú a konečnú množinu znakov.
Boolean (bool) zahŕňa logické hodnoty a operácie.

Premenné a deklarácie

Dátový objekt je obecné pomenovanie akéhokoľvek údaju uloženého v operačnej pamäti.
Deklarácia premennej je konštrukcia, ktorá pridelí premennej meno a typ,ale nevytvorí ju.
Definícia premennej okrem mena a typu priradí premennej aj pameťový priestor.

Operátory a výrazy
Výraz je konštrukcia jazyka, ktorá má hodnotu
Operand je časť výrazu ktorú je aplikovaný jeden z legálnych operátorov.
Operátor určuje akým spôsobom sa z operandov získa hodnota výrazu.

3.4 Riadenie chodu programu

Program sa všeobecne skladá z funkcií a premenných. Každý program musí obsahovať funkciu main. Main je prvá funkcia, ktorú program zavolá po spustení. Telo každej funkcie je zložené z množstva príkazov a definícií premenných. Princíp programovania spočíva vo vykonávaní príkazov, ktoré vedú k určitému zámeru. Samotné kódovanie programu spočíva v prepise algoritmu do vhodnej postupnosti príkazov.

3.4.1 Funkcie

Funckie sú základom programov. Umožňujú rozdeliť problém na menej jednoduchších podproblémov. (napr. Vytvoríte jednu funkciu pre čítanie zo súboru, druhú pre vykonanie akcie a tretiu pre zápis do súboru.)

Definícia funckie

```
NávratovýDátovýTyp MenoFunkcie ( ZoznamParametrov )
{
      TeloFunkcie ( postupnosť definícií premenných a príkazov )
}
```

```
Príklad 1.

int FunckiaVracajucaInteger (int parameter)
{
      parameter += 1;
      return parameter;
}
```

Úlohou funkcie je vykonať príkazy uvedené v jej tele a následne vrátiť výsledok návratového dátového typu uvedeného v definícii funkcie. Na vrátenie výsledku slúži príkaz return. Výnimkou je ak funckia vracia dátový typ ***void***, ktorý reprezentuje prázdny dátový typ. Vtedy sa nevracia nič.

Volanie funkcie

```
MenoFunkcie ( Parametre ) ;

Príklad 2.

FunkciaVracajucaInteger(parameter);
```

Ďalej si uvedieme niektoré programovacie konštrukcie, ktorými je možné riadiť chod programu. A to buď vynechaním niektorých príkazov, ich opakovaním a podobne. Súhrne tieto konštrukcie môžeme nazvať ***Príkazy***.

3.4.2 Príkazy

Výraz

Výrazom rozumieme aritmetický výraz, výskyt konštanty a premennej, priradenie a volanie funkcie.

Príklad výrazu je napríklad `a + b`, `a – 1` , ale tiež `a = 5` alebo `funkcia()`

Podmienené príkazy – príkaz if

Tieto príkazy umožňujú program vetviť, teda vykonať buď jeden blok príkazov alebo iný. Najpoužívanejší podmienený príkaz je určite príkaz if. Jeho konštrukcia je nasledovná:

```
if ( podmienka )
{
      BlokPríkazov
}
```

```
Príklad 3.

if ( a == 2 )
{
      printf(“Premenná a má hodnotu 2.“);
}
```

Jeho činnosť je nasledovná. V prvom rade sa vyhodnotí podmienka, ktorá je uvedená v zátvorkách za príkazom if. V prípade, že je podmienka splnená (vráti návratovú hodnotu 1) je vykonaný blok príkazov vnútri tela príkazu if. V prípade, že podmienka splnená nie je blok príkazov zostane nevykonaný.

Niekedy však potrebujeme v prípade, že podmienka splnená nie je vykonať iný blok príkazov. Na tento účel nám slúži príkaz if – else. Tento príkaz je rozšírenie príkazu if o druhú vetvu. Tá sa vykoná práve vtedy ak podmienka nie je splnená. Konštrukcia príkazu if – else je takáto :

```
if ( podmienka )
{
      BlokPríkazov
}
else
{
      BlokPríkazov2
}
```

```
Príklad 4.

if ( a == 2 )
{
      printf(“Premenná a má hodnotu 2.“);
}
else
{
      printf(“Premenná a nemá hodnotu 2.“);
}
```

Jeho činnosť bola vysvetlená vyššie. V krátkosti znamená, že v prípade ak je podmienka splená vykoná sa blok príkazov za if inak blok príkazov za else.

Príkazy cyklu – príkazy while a for

Ako môžeme vyrozumieť z názvu podkapitoly príkazy cyklu nám pomôžu v prípade ak potrebujeme určitý blok príkazov vykonávať opakovane. Podobne ako u príkazu if tu rozlišujeme podmienku a telo cyklu.

Základné 3 príkazy cyklu sú cykly while, do - while a for. Ich najväčší rozdiel je v tom, že príkazy while a for nemusia prebehnúť ani raz. Teda blok príkazov v tele cyklu while a for sa nemusí vykonať ani jeden krát. Naproti tomu príkaz do - while vždy aspoň raz prebehne.

Cyklus while

```
while ( podmienka )
{
      BlokPríkazov
}

Príklad 5.

int a = 0;
while ( a < 10 )
{
      printf(„Premenná a má hodnotu %d \n“, a);
      a++;
}
```

Cyklus while sa používa v prípade kedy nepoznáme presný počet opakovaní bloku príkazov v tele cyklu. Príkaz while vykonáva blok príkazov dovtedy kým má podmienka nenulovú hodnotu. V prípade, že sa blok príkazov začne vykonávať vykoná sa celý, neskončí v okamihu kedy by bola počiatočná podmienka splnená. Zrušiť jeho priebeh je možné umelo píkazom ***break.***

Cyklus do – while

```
do
{
      BlokPríkazov
} while ( podmienka );
```

```
Príklad 6.

int a = 1;
do
{
      printf(„Premenná a má hodnotu %d \n“, a);
      a++;
} while(a < 1)
```

Tento cyklus vykonáva blok príkazov vo svojom tele dovtedy kým je podmienka platná. Od cyklu while sa líši tým, že prebehne vždy aspon jedenkrát vzhľadom na to, že podmienka sa porovnáva až konci cyklu, po vykonaní jeho tela. Demonštruje to aj uvedený príklad. Keby sa podmienka vyhodnocovala na začiatku cyklu neprebehol by ani raz vzhľadom k tomu, že a má hodnotu 1 už po definícii. V cykle do – while však 1 krát prebehne a potom nasleduje príkaz za cyklom. Cyklus do – while je užitočný predovšetkým v prípadoch keď sa čaká na nejakú udalosť.

Cyklus for

```
for ( inicializácia; podmienka; inkrementácia )
{
      Príkaz alebo BlokPríkazov
}
```

```
Príklad 7.

for(int i = 0; i < 10; i++) {
      printf(„Premenná i má hodnotu %d \n“, i);
}
```

Cyklus for sa používa v prípadoch keď požadujeme presne daný počet opakovaní príkazu alebo bloku príkazov.
Príkaz for má 4 časti

- inicializáciu
- podmienku
- inkrementáciu
- píkaz alebo blok príkazov

Inicializácia zadáva počiatočnú hodnotu premennej, ktorá bude riadiť priebeh cyklu.
Podmienka testuje riadiacu premennú cyklu. V prípade, že podmienka je vyhodnotená ako pravdivá cyklus sa opakuje, v prípade, že nie cyklus končí a program pokračuje ďalej. Test podmienky sa vykonáva pred samotným vykonaním príkazov a blok príkazov tela cyklu for sa tým nemusí vykonať ani raz.
Inkrementácia sa deje po každom vykonaní cyklu. Upravuje sa tak riadiaca premenná.

U cyklu for je možné každú z jej častí vynechať. Napriek tomu sa to neodporúča.

3.5 Zhrnutie

Funkcie, podmienené výrazy a cykly predstavujú spoločne s konceptom dátových typov prostriedky pre vytváranie štrukturovaných programov. Dôvodom pre vznik štrukturovaných jazykov bola potreba takých výrazových prosriedkov, ktoré umožnia vytvárať prehľadné, jednoducho pochopiteľné a znovupoužiteľné časti programov. [1]
Medzi základy programovania môžeme zaradiť najmä pochopenie jeho princípov. Konkrétne logickú prácu s premennými, výrazmi a najmä vyhodnocovanie podmienok, význam a použitie príkazov cyklu. Pochopenie týchto princípov dáva základ pre úspešné zvládnutie osvojenia si niektorého programovacie jazyka a samotného procesu programovania.
Z poznatkov z tej kapitoly sme si teda odniesli základy programovania a jeho štruktúr. V samotnej programovej časti bakalárskej práce teda využijeme najmä poznatky o cykloch a podmienkach, vzhľadom k tomu, že práve tieto sú najdôležitejšie základné programovacie štruktúry. Pokúsime sa demonštrovať ich logiku a význam pomocou jednoduchých úloh.

4 Stavba GUI

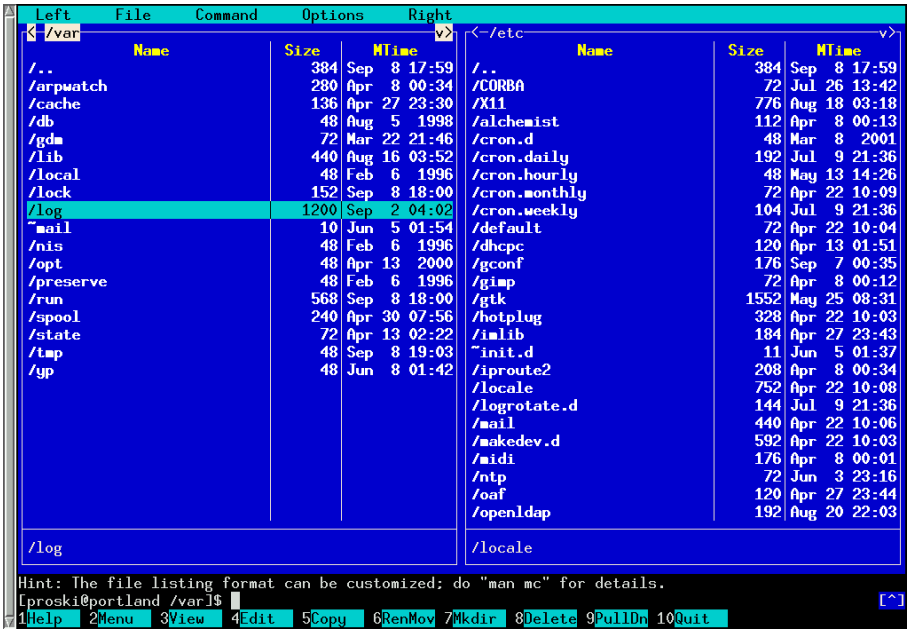
Čo je to vlastne skratka GUI? GUI je skratka z anglického Graphical User Interface, čo do slovenského jazyka môžeme preložiť ako grafické užívateľské rozhranie. Takéto rozhranie nám umožňuje ovládať počítač pomocou grafických ovládacích prvkov. Základom GUI sú okná na ktorých zobrazujú programy svoj výstup a to pomocou rôznych grafických ovládacích prvkov alebo komponent ako sú ikony, tlačítka, formuláre a podobne. Prvé grafické užívateľské rozhranie vyvinula spoločnosť Xerox v roku 1973. Ozajstný nástup a obľubu medzi užívateľmi však GUI zaznamenalo až v roku 1984 s nástupom počítačov Mac a následne s nástupom Microsoft Windows.

4.1 Ďalšie rozhrania

GUI nie je jediné používané rozhranie pre ovládanie počítačov. Pred príchodom grafiky a grafických rozhraní bolo možné ovládať počítač pomocou príkazov zadávaných do príkazového riadku.

4.1.1 Textové užívateľské rozhranie

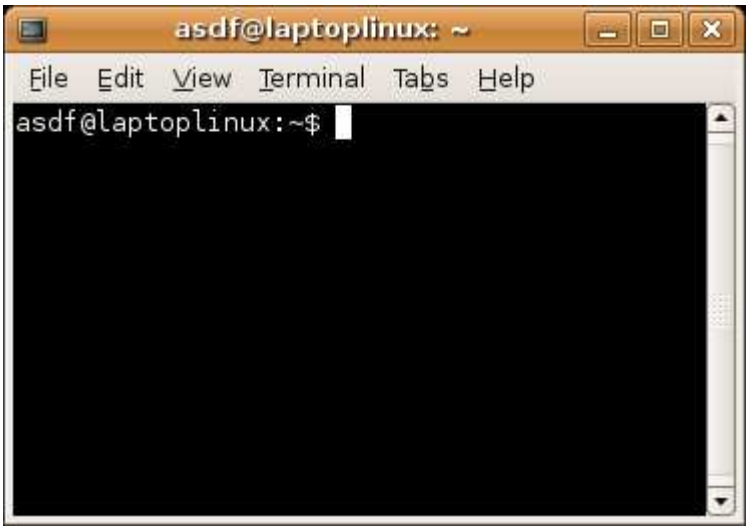
Toto rozhrnie tvorí akýsi medzistupeň medzí píkazovým riadkom a grafickým užívateľským rozhraním. Pracuje v textovom režime, kde obrazovka je pevne rozdelená na raster, pričom do každej pozície je možné zobraziť maximálne jeden znak.
Pomocou špeciálnych znakov (časti rámčekov, ukazateľ myši...) sú zostavené podobné ovládacie prvky ako u GUI, takže prostredie obsahuje okná, menu, tlačítka a ďalšie prvky obvyklé v GUI. [5]
Ako príklady tohto prostredia môžeme uviesť súborový manažér Norton Commander v operačnom systéme DOS alebo Midnight Commander používaný v unixových operačných systémoch.



Obrázok 1 – Ukážka textového užívateľského rozhrania

4.1.2 Príkazový riadok

Takáto práca s počítačom je pomerne náročná pre bežného užívateľa. Predpokladom pre zvládnutie práce s píkazovým riadkom je ovládanie príkazov, ktorým počítač rozumie. Príkazy sa zadávajú pomocou klávesnice a musia byť zadané presne.



Obrázok 2 – Ukážka príkazového riadku

4.2 Prvky GUI

Grafické užívateľské rozhranie je zložené z mnoha komponent a elementov. Tieto prvky sa na obrazovke zobrazujú v rôznych skupinách a poradiach a vyvolávajú rôzne akcie. Akciu je možné vyvolať stlačením klávesy, posunutím kurzoru myši nad komponentu, kliknutím na ňu a inými spôsobmi. Princím grafického rozhrania je vlastne reagovať na vyvolané udalosti.

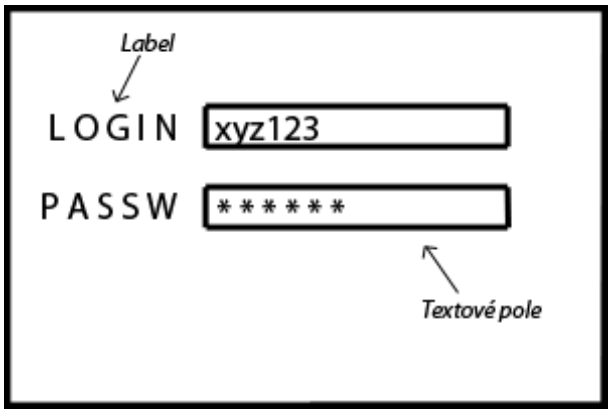
Okno je zvyčajne zobrazované ako obdĺžnik, prípadne štvorec a jeho obsah je zdanlivo nezávislý od zostatku obrazovky. Hlavnou črtou okien je schopnosť byť jednoducho a intuitívne používané aj neskúseným užívateľom. Okno podporuje základné operácie ako otvorenie, pohyb, zmena veľkosti, atď. Veľkosť okna môže dosahovať veľkosť obrazovky tzv. fullscreen mód. Okno je možné vybaviť ďalšími komponentami ako napríklad rámce, skrolovanie nástroje (scrollbars) a inými. [6]

Tlačítko. Jednoduchá koponenta, ktorá vyvoláva reakciu pri jej stlačení. Používané pri oznámení chybových alebo informačných správ musíte v okne stlačit tlačítko OK. To vyvolá akciu, ktorá uzavrie okno.

Obrázok 3 – Ukážka okna a tlačítka

Label. Z angličtiny môžeme slovo label preložiť ako štítok, menovka. Label je popisok pri komponente, je to prakticky text, ktorý informuje o zmysle komponenty. Napríklad label s textom „Login“ a label s textom „Password“ pri prihlasovaní do informačných systémov.

Textové pole. Do textového poľa je možné písať text. Väčšinou sa používa v prípadoch, keď je potrebná interakcia užívateľa zadaním mien, hesiel alebo príkazov. Napríklad už spomínané prihlasovanie do informačných systémov.



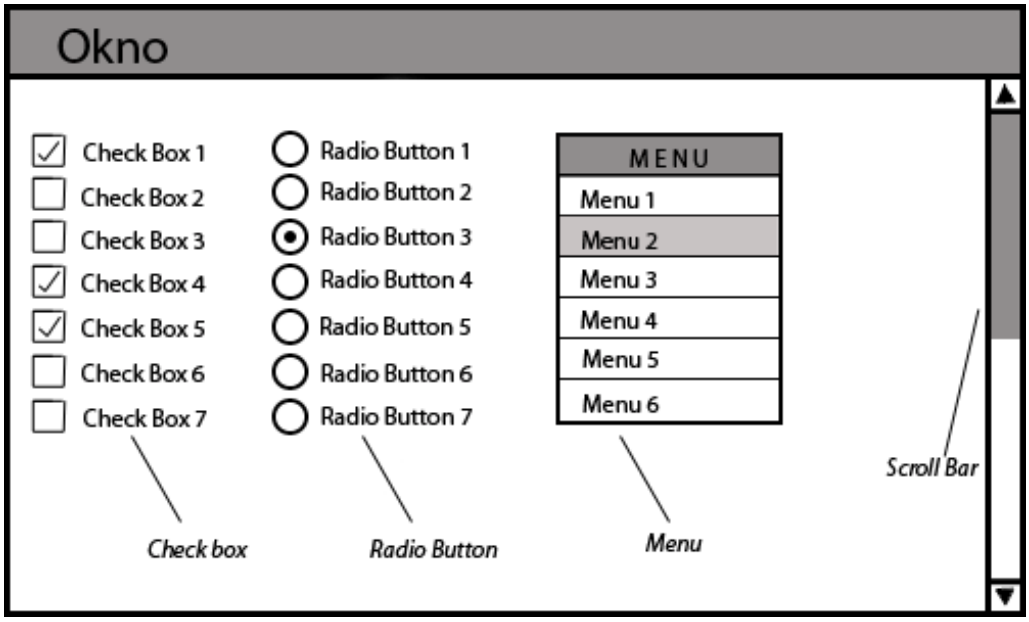
Obrázok 4 – Ukážka labelu a textového poľa

Menu. Zoznam zobrazený na počítačovej ploche. Zobrazuje funkcie dostupné pre voľbu užívateľa. [7] Menu poznáme najmä z internetových stránok, kde sa obvykle nachádza na ľavej strane obrazovky a predstavuje akýsi rozcestník po webovej stránke.

Check box. Je to skupina kde viaceré možnosti môžu byť vybrané. Tieto komponenty sú používané v prípade ak je potrebné vybrať zo zoznamu žiadnu až všetky možnosti. [9]

Radio button. Je to skupina kde môže byť vybraná práve jedna možnosť. Tieto komponenty sú používané v prípade ak je potrebné vybrať zo zoznamu práve jednu z možností. [10]

Scroll bar. Komponenta používaná pre zobrazenie a kontrolu toho, ktorá časť dokumentu alebo okna je práve viditeľná. Poznáme dva druhy komponenty scroll bar a to horizontálnu a vertikálnu. Častejšie používaná je vertikálna. Nachádza sa na pravej strane okna a jej posúvaním nahor a nadol posúvame celý dokument, prípadne obsah okna. [11]



Obrázok 5 – Ukážka komponent check box, radio button, menu a scroll bar

4.3 Tvorba GUI

Tvorba grafického užívateľského prostredia závisí vo veľkej miere od použitej platformy. Ďalej závisí na tom ako budeme GUI vytvárať. Buď ho budeme programovat „ručne“ a teda budeme písať celý zdrojový kód sami alebo použijeme programy, ktoré nám umožnia vkladať prvky klikaním. Túto možnosť poskytuje napríklad vývojové prostredie MS Visual Studio a programovací jazyk .NET. Možnosti sú naozaj rôzne no princíp tvorby GUI je stále rovnaký alebo aspoň podobný. V tejto časti sa budeme snažiť zhrnúť niektoré základné informácie.

Ako základný podklad máme plochu, na ktorú budeme pridávať komponenty. Plocha je svojím spôsobom funkčná časť okna s ktorou môžeme aktívne pracovať. Našou úlohou je na túto plochu pridať grafické komponenty tak aby sme dosiahli požadovaný výsledok a funkciu. To ako už bolo uvedené môžeme docieliť buď písaním programového kódu alebo naklikaním komponent.
Samotné komponenty by nám však boli na nič bez akcií, ktorými majú tieto komponenty reagovať na určitú udalosť(väčšinou nazývame udalsosť event).Napríklad po kliknutí na tlačítko musíme určiť akciu, ktorá sa má vykonať. Alebo po vpísaní textu do Text boxu musíme tento text nejakým spôsobom prečítať a spracovať ho, inak by nemal žiadne opodstatnenie.

Ukážme si teda niečo z praxe. Konkrétne to bude vytvorenie okna v jazyku Java, následné pridanie labelu, textového poľa a tlačítka do tohto okna. Komponentám nastavíme akcie tak aby po vpísaní do textového poľa a stlačení tlačítka program vypísal text z textového poľa.

4.3.1 Praktická ukážka v jazyku JAVA

V prvom rade musíme vytvoriť okno a nastaviť mu základné parametre.

```
JFrame window = new JFrame("Our window");
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.setBounds(0, 0, 200, 200);
window.setVisible(true);
```

Prvý riadok kódu vytvorí okno s názvom „Our window“. Riadok druhý nám zaručí, že pri zatvorení okna sa zároveň ukončí aj program. V treťom riadku nastavíme pozíciu a rozmery okna. V našom prípade bude okno v bode [0,0], čo znamená v ľavom hornom rohu obrazovky a jeho rozmery budú 200x200 pixelov. Štvrý riadok zobrazí naše okno. Po spustení programu bude priebežne naše okno vyzerať takto.

Obrázok 6 – okno s názvom „Our window“

Ďalej chceme do nášho programu pridať label, textové pole a tlačítko a samozrejme im nastaviť vhodné pozície. To v prípade jazyka JAVA nie je až taká jednoduchá úloha ako uvidíme na nasledujúcom kóde. Nasledujúci kus programového kódu vložíme medzi tretí a štvrtý riadok.

```
Container pane = window.getContentPane();
JPanel panel = new JPanel();
```

- Tu sme vytvorili plochu, na ktorú budeme pridávať koponenty

```
JLabel label = new JLabel("Text");
label.setBounds(10, 30, 100, 30);
pane.add(label);
```

- Vytvorili sme label s textom „Text“ a nastavili jeho pozíciu na súradnice

[10, 30] vzhľadom k oknu, určili jeho veľkosť na 100x30 pixelov a vložili ho na plochu

```
JTextField field = new JTextField();
field.setBounds(50, 30, 100, 30);
label.setLabelFor(field);
pane.add(field);
```

- V tejto časti sme vytvorili textové pole podobne ako label a nastavili sme label pre textové pole.

```
JButton button = new JButton("OK");
button.setBounds(75, 100, 50, 30);
Insets insets = new Insets(3,3,3,3);
button.setMargin(insets);
pane.add(button);
```

- Vytvorili sme tlačítko a nastavili pozíciu textu „OK“ v tlačítku.

```
pane.add(panel);
```

- Plochu sme pridali do okna.

Týmto sme docielili finálny vzhľad nášho okna. To bude vypadať takto.

Obrázok 7 – okno z obrázku 6 obohatené o label, textové pole a tlačítko „OK“

Teraz máme hotové okno a jeho vzhľad. Avšak okno je nefunkčné. Pri stlačení tlačítka sa nič nestane, text, ktorý vpíšeme do poľa nebude nijakým spôsobom využitý. Aby okno malo význam a aby sme s ním mohli vykonať nejakú akciu musíme naprogramovať obsluhu tlačítka a textového poľa. Label zostane neaktívny, pretože tento prvok slúži len na informovanie o iných komponentách.

Nasledujúci kód pri stlačení tlačítka vypíše text, ktorý je vpísaný v textovom poli. Nasledujúci kód musíme pridať pred pridanie tlačítka na plochu.

```
button.addActionListener(
      new java.awt.event.ActionListener(){
      public void actionPerformed(java.awt.event.ActionEvent evt){
            akcia(field.getText());
      };
});
```

Toto nám zaručí, že pri stlačení tlačítka „OK“ sa zavolá funkcia s názvom akcia, ktorej ako parameter predáme text z textového poľa. Aby program mohol fungovať musíme samozrejme spomínanú funkciu vytvoriť. To urobíme nasledujúco.

```
private static void akcia(String text) {
        System.out.println(text);
}
```

Teraz je náše okno plne funkčné a pri stlačení tlačítka „OK“ sa vypíše text uvedený v textovom poli.

4.4 Zhrnutie

Grafické užívateľské rozhranie je veľmi praktický a užitočný systém. Jeho vytvorenie uľahčilo prácu s počítačmi mnohonásobne a priblížilo ich bežnému užívaniu v domácnostiach tak ako je tomu teraz. Tvorba rozhraní je jedna z najdôležitejších vecí v informačných technológiách vôbec vzhľadom k tomu, že práve GUI je to s čím sa užívateľ stretáva. Vytvára jeho prvý dojem a samozrejme mu umožní komunikáciu s programom. Preto má byť GUI čo najprívetivejšie a jednoduché pre pochopenie a prácu s ním. Je vhodné používať zaužívané postupy ako napríklad pri prihlasovaní do systému použiť dve textové polia, pričom prvé je login, druhé heslo a podobne.
Programovú časť tejto práce sa teda budeme snažit vytvárať v súlade so základmi, ktoré sme si uviedli. Grafické rozhranie bude jednoduché a intuitívne na ovládanie, ale pri tom by malo byť vzhľadom prívetivé. Základom bude okno, na ktoré budeme podľa potreby pridávať rôzne komponenty.

5 Grafický výukový systém

V tejto časti sú vysvetlené praktické časti tejto bakalárske práce. Bude tu pojednávané o programe, ktorý je súčasťou práce, o jeho implementácii, funkcii a podobne. Budú tu demonštrované ukážky v podobe screenshotov a častí kódu súborov.

5.1 Charakteristika programu

Zadanie bakalárskej práce hovorí, že program má mať grafické užívateľské rozhranie. Ďalej hovorí , že má demonštrovať výuku a vytváranie základných algoritmických štruktúr pomocou programu do, ktorého užívateľ vkladá grafické prvky (objekty). Úlohou tejto bakalárskej práce je teda vytvoriť program s prívetivým a jednoduchým grafickým rozhraním, ktorý sa bude jednoducho ovládať a pritom splní svoju úlohou, ktorou je výuka princípov vybraných programovacích štruktúr.
Z kapitoly o učení a výuke (2) som prišiel k názoru, že najlepším riešením ako program koncipovať bude interaktívna hra. Konkrétne taká, ktorú si užívateľ najprv sám vytvorí a potom ju môže sám používať. Tento koncept som zvolil z dvoch dôvodov:

1. motivácia – Predpokladám, že vlastná interakcia užívateľa do tvorby aplikácia a jej následné užívanie budú vplývať pozítívne na jeho motiváciu a zároveň zvýšia stupeň jeho koncentrácie a pozornosti keďže pôjde sčasti o jeho vlastnú prácu.
2. dva druhy učenia – Pri tomto štýle výuky užívateľ rozvíja svoje kreatívne myslenie pri tvorbe hry. Zároveň však rozvíja logické myslenie a učí sa princípom demonštrovaných programovacích štruktúr pri jej hraní.

Druhá dôležitá otázka je čo má program vyučovať. Odpoveď som našiel v kapitole 3. Po zvážení informácií z nej som sa rozhodol pre tieto programové štruktúry – príkazy cyklu while a for a podmienka if. Vybral som ich z jednoduchého dôvodu. Práve tieto 3 konštrukcie sú základnými stavebými prvkami pokročilých programov. Bez pochopenia logiky týchto štruktúr nie je možné vytvoriť takmer žiadnu náročnejšiu užívateľskú aplikáciu. Preto sa v programe budem snažiť demonštrovať práve tieto 3 štruktúry.
Posledná dôležitá otázka je ako danú aplikáciu naprogramovať po výzorovej ale aj logickej stránke. Na prvú otázke, teda vzhľad programu nájdeme základy v kapitole Stavba GUI (4). Z nej vieme, že základom bude okno. Tak to bude aj u našej aplikácie. Do okna budeme podľa potreby vkladať rôzne komponenty tak aby celok tvoril logickú aplikáciu prívetivo vyzerajúcu a pri tom jednoduchú s intuitívnym ovládaním.

5.2 Aké funckie má program

Základom k tomu aké funkcie program obsahuje je otázka na čo je program určený. Má ísť o jednoduchý grafický výukový program, ktorý bude koncipovaný do hry. Užívateľ si sám bude tvoriť hru a to pomocou pridávania grafických objektov. Z toho je samozrejmé, že potrebujeme objekty, ktoré budeme pridávať a miesto, na ktoré bude tieto objekty možné pridať. Základom programu je teda plocha, na ktorú je možné vkladať spomínané grafické objekty. Ďalšia súčasť sú celkom logicky samotné vkladané objekty. Program obsahuje objekty pevne vložené, ktoré sú

jeho súčasťou. Sú dva a demonštrujú príkazy cyklov while a for. Keby však program obsahoval len určité množstvo možností rýchlo by sa stal nudným a neatraktívnym. Vytratila by sa tak požadovaná motivácia a koncentrácia. Bolo by vhodné aby sa obejkty mohli meniť alebo pridávať. Preto je do programu implementovaná možnosť dynamického vkladania objektov. Každý takýto objekt demonštruje použitie a princíp podmienky if. Všetky tieto pridané objekty držia statickz svoju polohu. Na plochu je po začatí hry pridaný ešte jeden (hlavný) objekt, ktorým je možné po ploche hýbať. Princípom hry je rozloženie objektov po ploche a následné reakcie hlavného objektu na objekty ostatné. Cieľom hry je dostať sa s hlavným objektom zo štartu do cieľa. Takáto je teda základná funckia hry. Okrem nej program samozrejme umožňuje ukladanie a načítanie rozohratých hier a umožňuje tak bez komplikácií a zbytočného nastavovania novej plochy jednoducho a bez problémov spustiť hru nanovo.

5.3 Implementácia

Program je implementovaný v jazyku JAVA. Tento jazyk som zo zadaných možností vybral preto, že práve JAVA poskytuje pokročilé grafické nástroje, ktoré sú jednoducho implementovateľné. Je to objektovo orientovaný jazyk, ktorý poskytuje všetky potrebné nástroje pre úspešné splnenie zadania tejto práce.

Samotný program je zložený z trinástich tried a používa štandarné knižnice jazyka JAVA, ktoré sú súčasťou základného balíku. Jediná importovaná knižnica je dom4j-1.6.1.jar, ktorá zabezpečuje prácu s XML súbormi.

5.3.1 Popis tried

Main.java – Vytvára okno triedy bar. V okne sa nachádza 100 komponent typu button v poli 10x10.
bar.java – Trieda, ktorá definuje okno. Táto trieda dedí od JFrame a oknu pridáva lištu. V lište sa nachádzajú možnosti ovládania programu (štart, uloženie, načítanie, pridanie objektu a ukončenie programu).
button.java – Definuje tlačítko, pričom dedí z triedy JButton. Okrem klasického tlačítka obsahuje ikonu a dve hodnoty typu int index a level. Tlačítku je pridaný MouseListener pre detekciu stlačenia.
add_object_window.java – Táto trieda vytvára okno, ktoré slúži pre pridanie nového objektu. Dedí z triedy JFrame a obsahuje komponent typu JCheckBox a 2 typu JTextField.
hero.java – Trieda definujúca objekt hero. Objekt hero obsahuje pole 12 hodnôt typu int.
hero_win.java – Vytvára okno, ktoré sa zobrazí po spustení programu a užívateľ v ňom nastaví 12 atribútov objektu hero.
object_window.java – Vytvorí okno, ktoré sa zobrazí pri pridaní objektu na hraciu plochu. Zoznam objektov je načítaný z XML súboru, kde sú tieto informácie uložené.
object_atributes – Trieda, ktorá vytvára okno pri pridávaní objektu na hraciu plochu a získava od užívateľa level tohto objektu.
started_game.java – Zisťuje či je hra v móde vytvárania alebo v móde hrania.
logic.java – Táto trieda zabezpečuje celú logiku hry. Určuje či je pohyb po ploche povolený, vyhodnocuje možnosť vstúpiť na pozíciu alebo naopak nevstúpiť.

save_load.java – Trieda, ktorá zabezpečuje ukladanie a načítanie hry. Hry sa ukladajú v XML súbore.
xml_reader.java – Vykonáva čítanie z XML súboru. Obsahuje funkcie, ktoré vracajú výsledok podľa typu požiadavku.
xml_writer.java – Zapisuje do XML súboru XML dokument.

5.3.2 Rozdelenie objektov

Program, ktorý som vytvoril k tejto práci má veľmi jednoduché a inuitívne grafické rozhranie. Pri jeho tvorení som vychádzal zo základov, ktoré sme si uviedli vo štvrtej kapitole. Základom je okno, ktoré obsahuje pole 100 tlačítok, ktoré sú rozložené v mriežke 10x10. Na každom tlačítku sa vždy nachádza jeden objekt. Tieto objekty pridáva na plochu samotný užívateľ a vytvára tak hraciu plochu. Implicitne sú všetky tlačítka nastavené rovnako. Je na nich znázornená ikona základného objektu Main. Poďme sa zoznámiť s druhmi objektov, ktorými je možné nahradiť objekt Main.

Jednoduchý objekt – Je každý objekt, ktorý do programu pridá užívateľ. Pri jeho pridaní je potrebné zadať cestu k ikone objektu a zároveň vybrať z 12 ponúknutých atribútov tie, na ktoré bude objekt reagovať. Pri pridaní obejtku na plochu sa zadáva level, ktorý značí aký náročný bude prechod na pozíciu tohto objektu. Level je možné voliť od 1 do 10.
Tento typ objektu demonštruje jednu z dôležitých algoritmických štruktúr, ktoré sme si uviedli v kapitole 3.4.2 a to podmienku *if*. Princíp znázornenia tejto podmienky spočíva v jednoduchom porovnaní levelu objektu a vyrátaného levelu hrdinu pre tento objekt. Ak je podmienka splnená hrdina sa posunie na dané políčko (vykoná sa tak telo podmienky) ak však podmienka splnená nie je na toto políčko nie je možné vstúpiť (telo podmienky zostane nevykonané).
Objekt Hero – Tento objekt je postavou, ktorou sa hráč pohybuje po ploche. V ďalšom texte budeme objektu Hero hovoriť hrdina. Hrdina má 12 atribútov, rovnakých aké vyberáme pri pridávaní objektu. Pri spustení hry si užívateľ tieto atribúty nastaví, pričom každý môže nadobúdať hodnotu od 1 do 10.
Zložité objekty – Zložité objekty sú také, ktoré demonštrujú určitú algoritmickú štruktúru.

***Objekt Hádaj* –** Práve tento objekt je jeden z nosných objektov celej práce pretože zobrazuje logický princíp jednej z algoritmických štruktúr, čo je poslaním celého programu. Konkrétne to je príkaz cyklu *while,* ktorý sme rozoberali v kapitole 3.4.2. Jeho úlohou je vykonávať operáciu až pokiaľ podmienka pred cyklom nie je splnená. Program pomocou tohto objektu znázorňuje cyklus while tak, že v prípade vstupu na pozíciu s týmto objektom program vygeneruje číslo od 1 do 100. Úlohou hráča je toto číslo uhádnuť a pokiaľ sa mu to nepodarí musí zostať na tejto pozícii. Program mu pomôže tým, že po každom nesprávnom pokuse vypíše či ide o číslo menšie alebo väčšie. Princíp znázornenia je teda v tom, že pokiaľ užívateľ nesplní podmienku, že vygenerované číslo sa rovná tomu, ktoré háda musí opakovať hádanie. Porovnanie čísla hádaného s číslom vygenerovaným programom teda môžeme chápať ako podmienku cyklu a následné hádanie a zobrazenie informácie, či je číslo, ktoré užívateľ tipoval správne,menšie alebo väčšie ako telo cyklu.

***Objekt Klikaj* –** Toto je ďalší z nosných objektov. Objekt Klikaj demonštruje príkaz cyklu *for*, ktorý sme si uvádzali taktiež v kapitole 3.4.2. Ukážka tohto príkazu cyklu spočíva v tom, že v momente vstupu na toto políčko je otvorených 1 až 20 okien, podľa toho aké

číslo vygeneruje program, ktoré musí hráč pozatvárať. Užívateľ teda musí pre 1. až posledné okno, ktoré sa mu zobrazí vykonať operáciu zatvorenia, čo môžeme pochopiť ako telo cyklu.

Objekty s iným významom – Toto sú objekty, ktoré slúžia pre orientáciu v hre. Sú to objekty Start, Finish a Main (základná plocha). Tieto objekty majú level 0 a vstup na ne je možný za každých okolností.

5.3.3 Pridávanie objektov do programu

Jedna z dôležitých vecí na tomto programe je ako sme si uviedli dynamické pridávanie objektov do programu. Týmto je taktiež zabezpečená väčšia interakcia užívateľa s tvorbou samotnej hry. Rozširujú sa tak jeho tvorivé schopnosti a môže si prispôsobiť aplikáciu podľa svojich potrieb, čo má kladný vplyv na pojmové učenie a taktiež zvyšuje koncentráciu, keďže ide o kreatívnu činnosť. O tomto sme hovorili v kapitole 2.

Princíp dynamického vkladania je v tom že každému objektu je pri jeho pridaní do programu nastavené na, ktoré z 12 atribútov, ktoré má aj samotný hrdina objekt bude reagovať a na ktoré nie. To dáva programu možnosť rozmanitého rozširovania.

Objekty je možné pridávať kedykoľvek a to pomocou položky „Add object“ v Menu „Object“ vrchnej lišty. Po zvolení tejto položky je zobrazené okno, kde sa nachádzajú 2 textové polia (JTextField) a 12 komponent triedy JCheckBox, o ktorých sme hovorili v kapitole 4.2. Do prvého textového poľa vpíšete meno objektu, do druhého cestu k obrázku, ktorý bude slúžiť ako ikona na tlačítko. V check boxoch nastavíte, na ktoré atribúty objekt bude reagovať, na ktoré nie. Po potvrdení pridania objektu ho môžete začať používať.

5.3.4 Ako je to implementované

Základom GUI ako sme hovorili v kapitole 4 je okno. Inak tomu nie je ani v tomto programe. Základ je teda okno triedy bar, ktorá rozširuje klasické okno triedy JFrame o vrchnú lištu (JMenuBar). Lišta obsahuji dve rolovateľné menu. Prvé menu „Game“ ktoré obsahuje 4 položky. Sú to „Start Game“, „Save Game“, „Load Game“ a „Exit game“. A duhé „Object“ obsahujúce položku „Add object“. Ich funkciu si vysvetlíme neskôr. Do funkčnej plochy okna je vložených 100 tlačítok triedy button. Objekt tejto triedy je klasické tlačítko (JButton) avšak obsahuje naviac ikonu (ImageIcon), ktorá je zobrazená na tomto tlačítku a ďalej dve premenné dátového typu integer index a level. Premenná index určuje, pozíciu na ktorej sa tlačítko nachádza, premenná level samozrejme určuje level objektu, ktoré sa na tomto políčku nachádza. Každý button má priradený MouseListener, ktorý reaguje na kliknutie naň. V prípade, že je na tlačítko kliknuté volá sa funkcia, ktorá zistí, či je hra v móde vytvárania alebo v móde hrania. Podľa toho určí ďalšiu akciu. V prípade, že sa nachádzame v móde vytvárania zobrazí sa nám okno triedy JFrame, ktoré obsahuje zoznam (JList) objektov, ktoré je možne pridať na plochu. Zoznam objektov je uchovávaný v XML súbore objects.xml o ktorom budeme hovoriť neskôr. Zo zobrazeného zoznamu je potrebné vybrať jeden z objektov, ktorý na danom políčku požadujeme. V prípade, že vyberieme jednoduchý objekt je mu ešte potrebné nastaviť level, v prípade, že ide o objekt špeciálny, štart, cieľ alebo základný objekt Main je tento úkon vynechaný, pretože tieto objekty majú implicitne level nastavený na hodnotu 0, čo zaručí vstup na ne v každom prípade. Týmto spôsobom pripravíme plochu k hre. Po spustení hry pomocou položky „Start Game“ v Menu vo vrchnej lište sa hra spustí. Pred samotným hraním je však potrebné nastaviť atribúty

hrndinu. Preto sa zobrazí okno (JFrame), ktoré obsahuje 12 komponent triedy JSlider, ktoré umožnia pohybom posuvníka po úsečke nastaviť atribút hrdinu v rozmedzí 1 až 10. Po potvrdení vlastností hrdinu sa hra prepne do módu hrania. V tomto momente po kliknutí na tlačítko sa nezobrazí možnosť pridania objektu na plochu, ale považuje sa to za pohyb na dané miesto. V prípade nepovoleného pohybu program oznámi chybu prostredníctvom oznamovacieho okna JOptionPane. Pri pohybe na políčko, kde je možné vstúpiť vzhľadom na porovnanie atribútov hrdinu a levelu objektu sa posunie hrdina na pozíciu daného objektu. Inak sa však reaguje na objekty zložité.

Reakcia na zložité objekty

Objekt Hádaj – Pri vstupe na tento objekt je vygenerované číslo od 1 do 100 pomocou funkcie Random(). Užívateľovi sa zobrazí jednoduché okno, ktoré je podobné tomu v praktickej ukážke z kapitoly 4. Do tohto okna vpíše užívateľ svoj typ. Následne sa zobrazí odpoveď vo forme dialógového okna (JOptionPane) o tom, či bol typ správny alebo nesprávny. Tento proces sa vykonáva až do momentu kým hráč neuhádne správne číslo a hrdina nie je posunutý na pozíciu, kde sa nachádzal objekt Hádaj.

Objekt Klikaj – Ak hráč vstúpi na tento objekt počítač ako v vpípade objektu Hádaj generuje pseudonáhodné číslo pomocou funkcie Random() avšak teraz v rozmedzí 1 až 20. Po vygenerovaní tohto čísla sa po celej obrazovke vytvorí práve toľko okien triedy JDialog, aké číslo bolo vygenerované. Rozmiestnenie po celej obrazovke je zabezpečené generovaním súradníc okna podľa veľkosti plochy obrazovky. Pre vytvorenie okien je použitá trieda JDialog, ktorá zabezpečí, že program nebude pracovať dovtedy, kým všetky okná nebudú spracované, teda uzatvorené.

Cieľom hry je pomocou pohybu po ploche a prekonávaním objektov dostať hrdinu do cieľa.

5.3.5 Uchovávanie objektov a uložených hier

Aby bolo možné dynamicky pridávať objekty je potrebné ich aj niekde uchovávať. Na uchovávanie informácií o programoch slúžia väčšinou databázy. Pre náš program však postačí XML súbor, ktorý vhodnou štruktúrou databázu nahradí. Na ukladanie používam 2 súbory. Zvlášť sú uložené informácie pre objekty a to v súbore „objects.xml“ a zvlášť informácie o uložených hrách, súbor „save.xml“. Prácu s týmito súbormi zabezpečujú triedy read_xml.java a write_xml.java. Využívajú importovanú knižnicu dom4j-1.6.1.jar. Trieda read_xml.java obsahuje metódy pre čítanie z xml súborov, ktoré majú rôznorodé vstupné parametre a výstupné dátové typy. Trieda write_xml.java obsahuje metódy prezápis dokumentu do súboru. Štruktúru, ktorú som zvolil pre ukladanie do dokumentov si ukážeme následne.

objects.xml

```
<?xml version =”1.0” encoding=”UTF-8” ?>
<objects>
      <object name=”Main”>
            <pic cesta = “img/main.png” />
      </object>
      <object name=”Start”>
```

```
            <pic cesta = “img/start.png” />
      </object>
      <object name=”Finish”>
            <pic cesta = “img/finish.png” />
      </object>
      <object name=”Hadaj“
            <pic cesta = “img/hadaj.png“ />
      </object>
      <object name=”Klikaj“
            <pic cesta = “img/klikaj.png” />
      </object>
      <object name=”Enemy” stre=”y” agil=”y” inte=”y” spee=”y” swim=”n”
      figh=”y” jump=”y” math=”n” logi=”y” beau=”n” abil=”y” danc=”n”>
            <pic cesta = “img/enemy.png” />
      </object>
</objects>
```

Koreňový uzol dokumentu má názov objects. Ten obsahuje uzly object. Každý takýto uzol má atribút meno, pod ktorým ho vidí užívateľ v programe a obsahuje element pic s atribútom cesta, ktorý obsahuje cestu k ikone. Prvých 5 objektov sú špeciálne objekty, ktoré sú súčasťou programu. Uzol s menom Enemy je objekt pridaný užívateľom a okrem atribútu name obsahuje ďalších 12 atribútov, ktoré znázorňujú vlastnosti na ktoré v prípade, že atribút má hodnotu „y“ reaguje a v prípade, že má hodnotu „n“ nereaguje hrdina.

save.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<saves>
      <save1>
            <pol0 picture="img/main.png" level="0"/>
            <pol1 picture="img/lake.png" level="3"/>
            <pol2 picture="img/enemy.png" level="6"/>
            <pol3 picture="img/main.png" level="0"/>
            <pol4 picture="img/mountain.png" level="4"/>
                  .
                  .
                  .
            <pol95 picture="img/main.png" level="0"/>
            <pol96 picture="img/question_mark" level="0"/>
            <pol97 picture="img/main.png" level="0"/>
            <pol98 picture="img/lake.png" level="8"/>
            <pol99 picture="img/main.png" level="0"/>
      </save1>
</saves>
```

Kořenový uzol dokumentu je uzol saves, ten obsahuje potomkov, pričom názov uzlu, ktorý je potomkom je zároveň meno uloženej hry. Každý takýto uzol obsahuje 100 elemntov pol0 až

pol99, ktoré obsahujú ako atribúty picture, kde je uložená cesta k ikone a level, kde je uložený level objektu.

5.3.6 Po spustení aplikácie

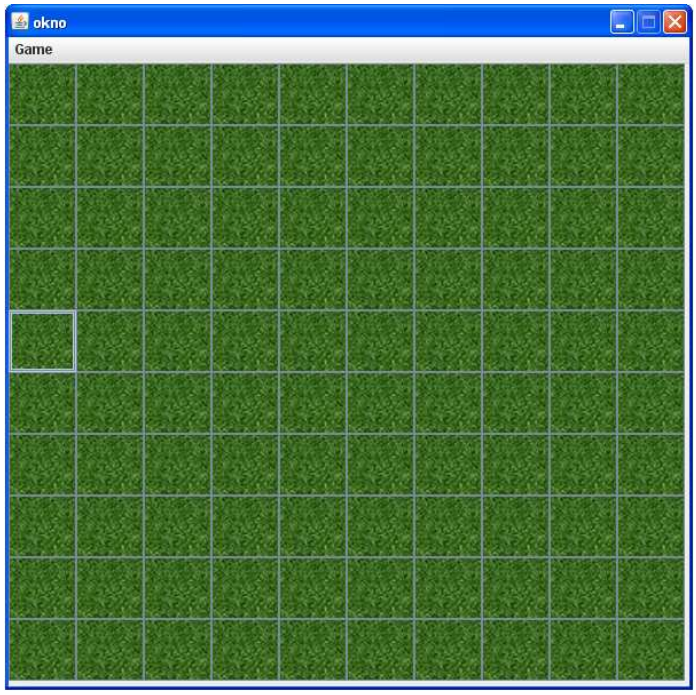
Po spustení aplikácie sa zobrazí základné okno. Pred samotným začatím hry si užívateľ sám navolí kde na ploche chce pridať aké objekty. Jednoduchým objektom pri ich pridaní na plochu priradí level, nastaví štart a cieľ. Plochu nastavíme klikaním na tlačítka. Pri stisku tlačítka sa vytvorí okno so zoznamom objektov, ktoré je možné pridať na plochu.Po nastavení plochy hráč spustí hru voľbou v menu. Plochu je možné kedykoľvek uložiť a taktiež načítať uloženú hru.

Po spustení hry sa program prepne do módu hrania. Avšak skôr ako začneme hrať je potrebné nastaviť atribúty nášho hrdinu.Preto je zobrazené okno s voľbou nastavenia týchto atribútov. Po tomto úkone je všetko pripravené pre hru. V tomto momente sa už po kliknutí na ikonu nezobrazí možnosť pridania objektu na plochu, ale bude sa ním rozumieť, že hráč sa chce na túto pozíciu posunúť s hrdinom.

Princíp hry spočíva v pohybe s hrdinom po ploche. Cieľom je dostať sa od štartu k cieľu. Hrdina sa môže pohybovať v štyroch základných smeroch (nahor, nadol, doľava a doprava) vždy o jedno políčko. Pri vstupe na jednoduchý objekt algoritmus vyhodnotí, či je tento ťah možný. To porovnaním relevantných atribútov hrdinu, ktoré zlúči do jedného čísla s levelom, ktorý bol priradený danému objektu. V prípade, že hrdinove atribúty sú väčšie ako level presun pozície je povolený, v prípade, že tomu tak nie je posunutie nie je možné. Hra končí v momente keď sa hrdina dostane na objekt Finish. Vtedy si môžete vytvoriť novú hru, otvoriť niektorú z uložených hier alebo program ukončiť.

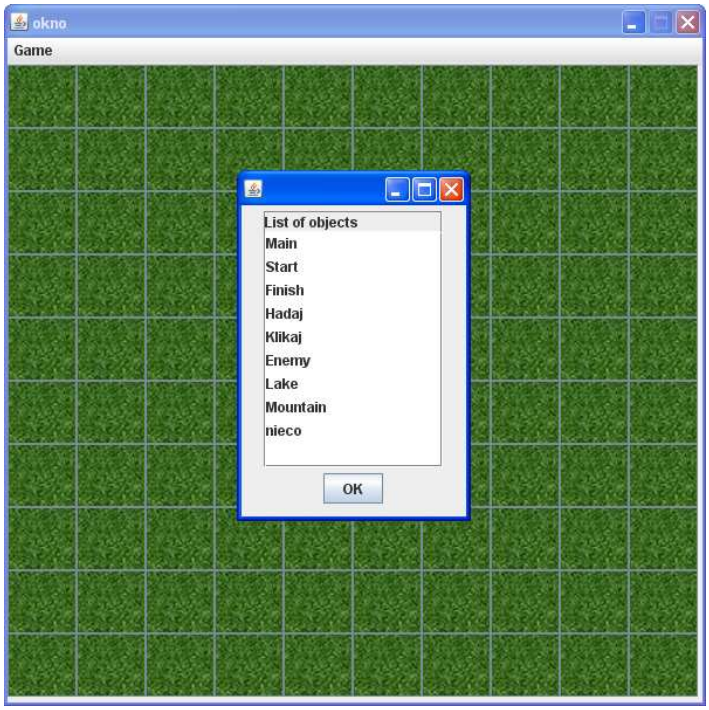
5.3.7 Ukážka grafického rozhrania

V tejto časti bude zobrazených niekoľko obrázkov, ktoré zobrazujú grafické rozhranie programu. Prvý obrázok zobrazuje program po spustení. V tomto momente je plocha vyplnená základnými objektami main. A je možné na ňu pridávať objekty.



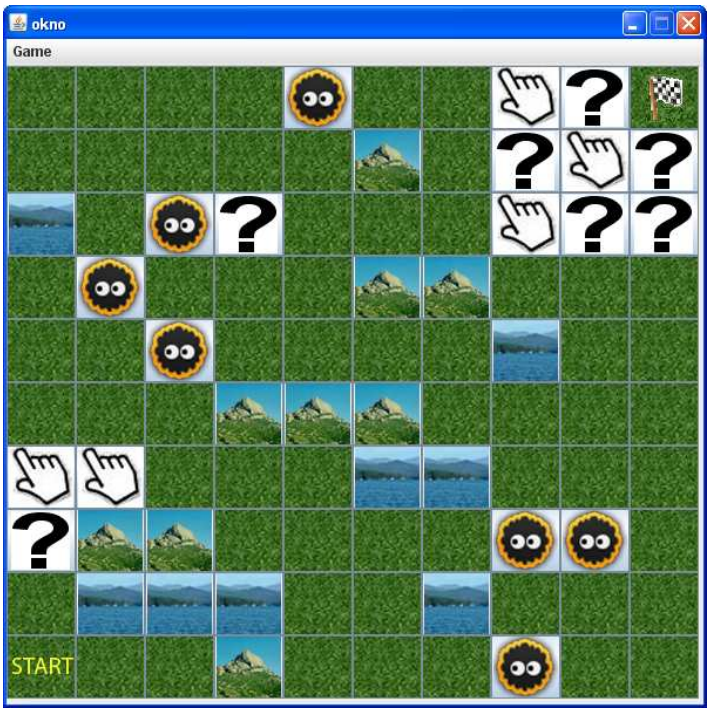
Obrázok 8 – Ukážka hracej plochy po spustení

Na ďalšom obrázku vidíme okno, ktoré je zobrazené po kliknutí na ikonu v móde tvorenia hry. V okne je zobrazený zoznam objektov, ktoré môže užívateľ pridať.



Obrázok 9 – Ukážka okna pre výber objektu

Nasledujúca plocha je pripravená na stpustenie. Sú na nej pridané objekty a tiež políčko štart a cieľ. Po odštartovaní sa zobrazí okno s nastavením atribútov hrdinu a program sa dostáva do módu hrania.



Obrázok 10 – ukážka hracej plochy pripravenej pre štart

Toto sú základné okná pre prácu s programom. Zostávajúce okná sú väčšinou jednoduché a ich obsluha je intuitívna. Sú to napríklad okno pre načítanie hry, kde vyberiete zo zoznamu jednu hru a potvrdíte tlačítkom OK alebo okno pre zadanie levelu, ktoré obsahuje textové pole do ktorého zadáte číslicu od 1 do 10 a potvrdíte tlačítkom OK.

5.4 Zhrnutie

Program, ktorý je pridaný ako príloha tejto bakalárskej práce má za úlohu štýlom hry priblížiť základné logické a algoritmické štruktúry programovania. Jeho základom je jednoduché GUI, ktoré je postavené na základoch tak ako sú uvedené v kapitole 4. Pomocou pridávania grafických objektov a jednoduchej logiky program demonštruje príkazy cyklov for a while a podmienky if, ktoré sú rozoberané v kapitole 3. Využitie tohto programu je predpokladané najmä na základncýh školách pre výuku už spomenutých konštrukcií. Program je písaný v programovacom jazyku JAVA a je spustiteľný na platforme Windows a zároveň tak aj na platforme UNIXovej. Jeho výhodou je jednoduchosť a intuitívnosť ovládania. Nevýhodou môže byť statické pole pohybu o rozmeroch 10x10 políčok.

6 Záver

Bakalárska práca sa týka grafického výukového systému. Jej cieľom je zoznámiť čitateľa so základmi spomenutých oblastí, preto je práca písaná tak aby priniesla čo najviac nových a základných informácií pre človeka neznalého problému, vysvetlila a opodstatnila prečo som postupoval pri tvorbe programu tak ako som postupoval. Každá kapitola predstavuje jedno rozhodnutie. Na konci každej kapitoly sme si na jej základe mohli utvoriť záver ako bude daná sféra programu stavaná a prečo.

V prvej kapitole (vynímajúc úvod) sme si hovorili o didaktických vedách, o spôsoboch a druhoch učenia, o tom, čo je pre učenie a výuku dôležité. Vďaka tejto kapitole sme sa dozvedeli, že veľmi podstatná je motivácia a vlastná zaangažovanosť, ktoré udržia pozornosť užívateľa a prinútia ho sa aktívne venovať aplikácii a tým učeniu toho, čo demonštruje.

Následne sme si vysvetlili základy programovania. Hovorili sme o základných štruktúrach a logických princípoch. Výsledkom tejto kapitoly bolo rozhodnutie o tom čo náš program bude vyučovať, ktoré štruktúry sú najdôležitejšie pre pochopenie programovania ako celku.

Vo štvrtej kapitole sme rozoberali stavbu GUI. Na jej základe sme potom stavali vlastné grafické rozhranie pre náš program. Ukázali sme si tu ako na to, vysvetlili niektoré základné pojmy a práve táto kapitola dala základ tomu ako náš program v konečnom dôsledku vyzerá po vonkajšej, grafickej stránke.

Piata kapitola pojednala o samotnej implementácii programovej časti tejto práce. Uviedli sme si v nej čo je úlohou programu, prečo a ako vznikal. Opísali sme si použité konštrukcie a vysvetlili princípy ako program funguje a aké funkcie poskytuje. Program má za účel štýlom hry demonštrovať základné algoritmické štruktúry, ktoré boli načrtnuté v kapitole 3 a napomáhať tak k rozvoju logického myslenia a získania základov programovacej logiky. Pri vytváraní programu je použitý jazyk JAVA a jeho štandardné knižnice doplnené o knižnicu dom4j-1.6.1.jar, ktorá pracuje so súbormi XML.

Táto bakalárska práca môže mať využitie v oblasti výuky najmä u mladších ľudí, predovšetkým na základných prípadne stredných školách. Snažil som sa ju vytvoriť tak aby jej užívanie bolo čo najefektívnejšie. Zvolil som koncepciu hry, ktorá udrží užívateľa pozorného a snažil som sa čo najlepšie demonštrovať zábavnou formou úloh vybrané programové a algoritmické štruktúry.

Literatúra

[1] KRESÍKOVÁ, J., MARTINEK, D.; *Základy programování : IZP*. 2006. 235 s.

[2] HUČEKOVÁ, D. Psychológia učenia. 2008 [cit. 2009-05-15], s. 1-5.
URL <http://www.radsvetla.inky.sk/novic/resources/Psychologia%20ucenia%20sa.pdf>.

[3] ĎURIČ, L., et al. Pedagogická psychológia. *Jaspis*. 1991.

[4] DROTÁROVÁ, E. Pedagogická psychológia. 2009 [cit. 2009-05-15].
URL <http://www.x-sat.cz/DTI/data/2_semestr/prednasky/Pedagog_psych_prepis_MarianP.doc>.

[5] *User Interface* [online]. [cit. 2009-05-15].
URL <http://en.wikipedia.org/wiki/User_interface>.

[6] *Window Definition* [online]. 2004 [cit. 2009-05-15]. The Linux Information Project.
URL <http://www.linfo.org/window.html>.

[7] *GUI Definition* [online]. 2004 [cit. 2009-05-15]. The Linux Information Project.
URL <http://www.linfo.org/gui.html>.

[8] *Your dictionary : Menu definition* [online]. 1996-2009 [cit. 2009-05-15].
URL <http://www.yourdictionary.com/menu>.

[9] *Dictionary : Check box* [online]. 2009 [cit. 2009-05-15].
URL <http://dictionary.reference.com/browse/check%20box>.

[10] *Dictionary : Radio Button* [online]. 2009 [cit. 2009-05-15].
URL <http://dictionary.reference.com/browse/radio%20button>.

[11] *Dictionary : Scroll Bar* [online]. 2009 [cit. 2009-05-15].
URL <http://dictionary.reference.com/browse/scroll%20bar>.

Dodatok A – prevzaté obrázky

Obrázky použité v technickém správe

Obrázok 1 : URL <http://www.gnu.org/software/mc/images/mc-panels.png>.

Obrázok 2 : URL <http://www.everyjoe.com/newlinuxuser/files/2007/03/terminal1.png>.

Obrázky prevzaté do programovej časti

lake.png : URL <http://www.seasonsoflakeburton.com/img/20-Lake-Burton-07_b.jpg>.

question_mark.png :
URL <http://www.wpclipart.com/small_icons/misc_5/.cache/question_mark.png>.

click.png : URL <https://www1.dsidata.sk/upload/image/iptv/hand.png>.

main.png : URL <http://images.funadvice.com/photo/image/83829/tiny/grass.jpg>.

finish.png : URL
<http://www.novell.com/documentation/extend52/Docs/help/Director/books/PFImages/FinishActivity.gif>.

Ostatné obrázky použité v technickej správe a rovnako tak v programovej časti sú dielom autora bakalárskej práce.

Dodatok B – prílohy bakalárskej práce

Ako príloha bakalárskej práce je 1 DVD s programom, voľne dostupnou knižnicou programovacieho jazyka JAVA dom4j.jar a priečinok s obrázkami použitými v programe.