VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

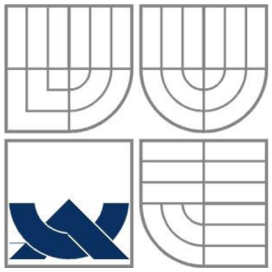
ABSTRAKTNÍ DATOVÉ TYPY PRO JAZYK C

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

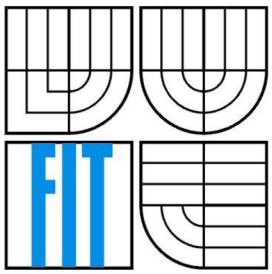
AUTOR PRÁCE
AUTHOR

TOMÁŠ DUDA

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ABSTRAKTNÍ DATOVÉ TYPY PRO JAZYK C

ABSTRACT DATA TYPES FOR C LANGUAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ DUDA

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. JAN M. HONZÍK, CSc.

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Duda Tomáš**
Obor: Informační technologie
Téma: **Abstraktní datové typy pro jazyk C**
Kategorie: Alg. a datové struktury

Pokyny:

1. Seznamte se detailně s textem kapitoly 3 o abstraktních datových typech ve studijní opoře pro předmět IAL.
2. Modifikujte text kapitoly zapsané pro pascalovský jazyk tak, aby zachoval co nejvíce i v detailech původní obsah, ale aby se vztahoval k zápisu příkladů a algoritmů v jazyce C.
3. Převeďte všechny příklady a algoritmy do jazyka C a odlaďte je v prostředí vytrvořeném pro tento účel.
4. Vytvořte program pro animovanou demonstraci operací nad jednosměrným, dvojsměrným a kruhovým seznamem s využitím nástrojů grafického zobrazení.
5. Navrhněte další vhodné kontrolní otázky a příklady.
6. Navrhněte příklady vhodné pro formulářově orientovanou písemnou zkoušku ze znalostí tohoto okruhu.

Literatura:

- Honzík, J: Algoritmy. Studijní opora. pro předmět Algoritmy. Elektronický text. FIT VUT v Brně

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Honzík Jan M., prof. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 15. června 2007
Datum odevzdání: 31. července 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Tomáš Duda**
Id studenta: 84419
Bytem: Padlých hrdinů 787/8b, 736 01 Havířov
Narozen: 03. 07. 1985, Havířov
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Abstraktní datové typy pro jazyk C
Vedoucí/školitel VŠKP: Honzík Jan M., prof. Ing., CSc.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 23.7.2007

....

Nabyvatel Autor

Abstrakt

Tato práce se zabývá abstraktními datovými typy v jazyce C. Jsou zde vysvětleny principy abstraktních datových typů a operace nad nimi. Hlavním úkolem je přepsání studentské opory, která je napsána v jazyce Pascal do jazyka C a vytvoření programu pro animovanou demonstraci. Tento program animuje operace nad lineárními seznamy a je implementován v prostředí FLASH.

Klíčová slova

Abstraktní datový typ, dynamické přidělování paměti, jednosměrně vázaný seznam, dvojsměrně vázaný seznam, zásobník, fronta, binární strom, jazyk C, jazyk Pascal, FLASH animace, ActionScript.

Abstract

This thesis deals with abstract data types in C language. There are explained abstract data types principles and operations around it. The main object of this thesis is a student scripts transcription, which is written in Pascal language, to the C language. Within all this belongs also an animated program for demonstration. This program animates operations around linear lists and is implemented in FLASH environment.

Keywords

Abstract data type, dynamic memory allocation, single-directional constrained list, bidirectional constrained list, stack, queue, binary tree, C language, Pascal language, FLASH animation, ActionScript.

Citace

Tomáš Duda: Abstraktní datové typy pro jazyk C, bakalářská práce, Brno, FIT VUT v Brně, 2007

Abstraktní datové typy pro jazyk C

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Prof. Ing. Jana Maxmiliána Honzíka, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Duda
23. 7. 2007

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu panu Prof. Ing. Janu Maxmiliánu Honzíkovi, CSc. za ochotný a přívětivý přístup k vedení mé bakalářské práce.

Obsah

1 Úvod

Tato bakalářská práce pojednává o modifikaci studijní opory k předmětu Algoritmy. Zadání této práce vzniklo z rozhodnutí usnadnit studentům, kterým je bližší jazyk C než pascalovský pseudokód porozumět a vytvářet studované algoritmy. Naše fakulta upustila v bakalářském stupni od výuky programovacího jazyka Pascal jako jediného jazyka a přešla k mnohem více používanému jazyku C. Cílem předmětu IAL je mimo jiné seznámení s pasivní znalostí pascalovských zápisů. Projekty a zadání domácích úkolů do předmětu Algoritmy jsou však zadávány v jazyce C, proto bylo užitečné doplnit oporu orientovanou na Pascal o verzi v jazyce C.

Textová část bakalářské práce se skládá z 6 kapitol. V kapitole 1 je čtenář obeznámen s obsahem textové části bakalářské práce a dozví se, o čem tato bakalářská práce pojednává. Kapitola 2 obsahuje popis vlastností jazyků Pascal a C. Detailní popis zadání projektu je v kapitole 3. V kapitole 4 je vysvětleno, jak pracují abstraktní datové typy (seznamy, zásobník, binární strom, aj.) a operace nad nimi. Kapitola 5 pojednává o vytváření programu pro animovanou demonstraci. Pro tento úkol jsem zvolil prostředí Macromedia FLASH MX 2004, které pracuje s programovacím jazykem ActionScript. Poslední kapitola patří souhrnu výsledků a zhodnocení významu tohoto projektu.

2 Historie a vlastnosti jazyků

V této bakalářské práci je řeč o třech programovacích jazycích. Pro uvození do tématu je dobré si o těchto jazycích něco říct. Jazyky Pascal a C jsou popsány v následujících kapitolách 2.1 a 2.2. Jazyku ActionScript je věnována samostatná kapitola.

2.1 Jazyk Pascal

Pascal byl prvotně navržen v roce 1969 profesorem Niklausem Wirthem. Název jazyka vznikl na počest francouzského matematika a fyzika Blaise Pascala, který už v 17. století sestrojil první kalkulátor.

Pascal je strukturovaný jazyk. Syntaxe programu je jednoduchá a zápis programu s lehkou znalostí angličtiny je velmi dobře čitelný. Pro ukázku je zde uveden zápis programu pro vypsání textu "Hello, World!" na obrazovku.

```
program HelloWorld(output);
begin
    writeln('Hello, World!');
end.
```

Výchozím jazykem pro Pascal měl sloužit programovací jazyk Algol 60. Pascal měl sloužit výhradně pro výuku programování. Toho mělo být dosaženo především omezeným počtem konstrukcí jazyka. Dalším požadavkem bylo, aby pomocí Pascalu byly vytvářeny efektivní a spolehlivě fungující programy na tehdejších počítačích. Tento úkol, jak se postupem času ukázalo, byl splněn.

Přestože byl Pascal vytvořen především pro výuku, můžeme se s ním setkat dnes i při vývoji komerčních programů.

2.2 Jazyk C

Tato kapitola je inspirována z [4]. Jazyk C je v současné době jeden z nejpoužívanějších programovacích jazyků. Je znám svou efektivností, ekonomií a přenositelností. Díky těmto schopnostem se uplatňuje ve všech oblastech programování. Primárně byl vytvořen pro systémové programování. Důkazem toho je fakt, že celé jádro operačního systému Unix je postaveno na tomto jazyce a má sloužit pro potřeby tohoto systému. Vyznačuje se také svou rychlostí. Dobře napsaný program v C je stejně rychlý jako program napsaný v assembleru a k tomu je čitelnější a srozumitelnější pro člověka.

Tento jazyk vznikl v Bellových laboratořích AT&T Denis Ritchie. Cílem bylo napsat jazyk pro snadnou a přenositelnou implementaci Unixu. Na tomto projektu se dále podíleli Brian Kernighan a Ken Thompson. Jméno "C" zvolili proto, neboť přebíral vlastnosti ze staršího jazyka nazvaného "B".

Stejně jako Pascal je C strukturovaný jazyk. Byl navržen jako poměrně malý jazyk kombinující efektivitu s výkonností. C neobsahuje v sobě funkce pro provádění vstupu a výstupu, alokaci paměti, práci s obrazovkou a řízení procesů. Pokud chceme tyto funkce využít, je třeba použít systémové knihovny. Díky tomu se můžeme rozhodnout, jestli použijeme funkce ze standardních knihoven nebo si vytvoříme funkce výhradně pro náš speciální případ. Tento návrh pomáhá oddělit vlastnosti jazyka od vlastností spojené s architekturou nebo konkrétním procesem. V C je tedy možno psát přenositelné programy.

Každý program v jazyce C musí obsahovat alespoň jednu funkci. Ta, která nesmí chybět v žádném programu je funkce `main`. Každá funkce se volá s různými parametry, které se uvádí za jménem funkce v kulatých závorkách. Pro porovnání s jazykem Pascal následuje identický program pro vypsání textu "Hello, World!".

```
#include <stdio.h>
int main(void)
{
    printf("Hello, World!\n");
    return 0;
}
```

V současné době se spíše používají objektově orientované verze jazyka C a to C++ nebo Objective-C, kvůli velkému rozmachu objektově orientovaného programování. Jazyk C a jeho příbuzné verze můžeme považovat za nejpoužívanější jazyky, neboť podle průzkumu na internetu je C a jazyk C# nejoblíbenějším jazykem.

3 Specifikace zadání

Tato kapitola přesně definuje zadání projektu.

3.1 Přepis opory

Studijní opora byla napsána v roce 2005 panem Prof. Ing. Janem Maxmiliánem Honzíkem, CSc. V programovacím jazyce Pascal. Tento text slouží jako učební pomůcka pro předmět Algoritmy, který je vyučován na naší fakultě v 3. semestru bakalářského studia. Cílem projektu je přepsat kapitolu 3 této opory pojednávající o abstraktních datových typech tak, aby struktura a původní obsah byl co nejvíce zachován, ale aby učivo bylo vysvětleno na datových strukturách v jazyce C. Je nutno opravit pravopisné chyby a zavést jednotný formát písma (tím je myšleno např. stejná velikost nadpisů, jednotné odsazení).

3.2 Převod algoritmů

Příklady a algoritmy jsou napsány v jazyce Pascal. Dalším úkolem je převedení algoritmů do jazyka C. Je zapotřebí vytvořit ladící prostředí a příklady v tomto prostředí odzkoušet. Příklady musí odpovídat normě C99.

3.3 Demonstrační program

Zadání projektu obsahuje vytvoření programu pro animovanou demonstraci operací nad abstraktními datovými typy jednosměrný seznam, dvojsměrný seznam a kruhový seznam. Aplikace by měla vysvětlovat princip operací – jako je InsertFirst, PostInsert, DeleteInsert, atd. V aplikaci bude uživatel moci volit rychlost animace.

3.4 Návrh otázek a příkladů

Na konci každé kapitoly ve studentské opoře jsou uvedeny kontrolní otázky a úkoly. Mým úkolem je vytvořit další otázky, které by procvičily a zopakovaly znalosti studenta po prostudování příslušné kapitoly.

Dalším úkolem je navrhnout testové otázky na písemnou zkoušku, které by prověřily studentovy znalosti o abstraktních datových typech a práci s těmito strukturami.

4 Abstraktní datové typy

Celá studentská opora předmětu IAL pojednává o různých abstraktních datových typech (ADT) a operacích nad nimi. ADT umožňují abstrakci dat (zjednodušený pohled na data), což je nezbytné pro psaní větších programů, protože lidský mozek je schopen řešit úlohy jen do určité míry složitosti. Umožňuje skrýt implementaci jednotlivých operací nad daným datovým typem a způsob fyzického uložení informace v proměnné daného datového typu. Hlavním cílem je zjednodušit a zpřehlednit program, který provádí operace s daným datovým typem. Všechny ADT lze realizovat pomocí základních algoritmických operací (přiřazení, sčítání, násobení, podmíněný skok,…).

ADT je uživatelsky definovaný datový typ. Je tedy určen potřebami programátora a ne potřebami hardware jako "obyčejný" datový typ. Používá se pro lepší pochopení implementace složitých datových typů. Například v jazyce C k definování ADT slouží příkaz typedef. Jednoduchým příkladem abstrakce dat může být datový typ *bod*, který se skládá ze dvou čísel (souřadnic) a který může programátor používat při programování grafiky.

Potřeba ještě větší abstrakce ve velkých programech vedla ke vzniku objektově orientovaného programování, kde se k tomuto účelu používají třídy, které umožňují k datům připojit i metody, které tyto data zpracovávají.

Definice abstraktního datového typu podle studentské opory:

Abstraktní datový typ (ADT) je definován množinou hodnot, jichž mohou nabývat prvky tohoto typu a množinou operací definovaných nad tímto typem.

Při práci s ADT je třeba znát některé významné pojmy:

Konstruktor je operace, jejímiž vstupními parametry je výčet (všech) komponent struktury a výsledkem operace je datová struktura obsahující tyto komponenty. Konstruktor “vytvoří” datovou strukturu z jejich komponent. Příkladem konstruktoru v jazyce C je textový řetězec v příkazu str = "Textovy retezec" nebo množina (1,3,5,7,9).

Selektor je operace, která umožní přístup k jednotlivé komponentě datové struktury na základě uvedení jména struktury a zápisu přístupu (“reference”). Příklad selektoru nad textovým řetězcem z předcházejícího odstavce je zápis str[3], kde prvek řetězce má hodnotu ‘t’ (pole v jazyce C je číslované od 0). Přístup k prvku datové struktury se používá za účelem změny hodnoty prvku nebo k získání jeho hodnoty.

Iterátor je operace, která provede zadanou činnost nad všemi prvky homogenní datové struktury. Příklad: většina složitých grafických útvarů je realizována seznamem grafických elementů, z nichž je útvar sestaven. Operace, která jediným příkazem typu iterátor provede operaci "vykreslit" nad všemi elementy, zobrazí útvar jako celek.

Destruktor je operace nad dynamickými strukturami. Tato operace zruší dynamickou strukturu a vrátí prostor jí zaujímaný systému DPP. Citace z [3].

V studentské opoře se vyskytují ty nejznámější abstraktní datové typy. V následujících kapitolách je vysvětlen popis a práce s těmi nejdůležitějšími.

4.1 Seznamy

Seznam je dynamická struktura, kde každý prvek struktury má právě jednoho předchůdce a jednoho následníka. Výjimku tvoří první a poslední prvek. Předchůdce prvního prvku a následník posledního prvku neexistuje. V praxi to znamená, že ukazatel na tento prvek neukazuje nikam (v jazyce C tento stav reprezentuje konstanta NULL).

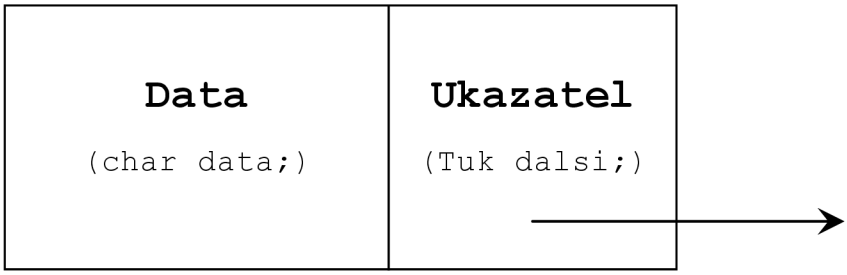
Prvkem seznamu může být libovolný jiný datový typ – také strukturovaný. Např. seznam seznamů. Seznam může být i prázdný, ukazatel na prvek pak obsahuje prázdnou hodnotu (v C opět konstanta NULL).

Výjimečná vlastnost seznamu je tzv. aktivita seznamu. Aktivita seznamu znamená, že seznam obsahuje aktivní prvek. Pokud je seznam neaktivní, aktivní prvek neobsahuje. Aktivní prvek je prvek seznamu, ke kterému se vztahují operace např. First, PostInsert, PostDelete, aj.

Přístup k prvnímu (okrajovému) prvku je přímý, přístup k ostatním prvkům seznamu je sekvenční ve směru průchodu. Seznam, který lze procházet jen jedním směrem, se nazývá "jednosměrně vázaný" zkráceně "jednosměrný" (kapitola 4.1.1). Seznam, u něhož lze procházet oběma směry je "dvousměrně vázaný" - "dvousměrný" (kapitola 4.1.2).

4.1.1 Jednosměrný seznam

Jednosměrně vázáný lineární seznam je abstraktní datová struktura obsahující prvky, které v sobě nesou hodnotu svého prvku a ukazatel na další prvek (obrázek 1). Na obrázku v závorkách je znázorněno, jak by mohla vypadat deklarace proměnných v uzlu seznamu (Tuk je typu ukazatel na prvek).

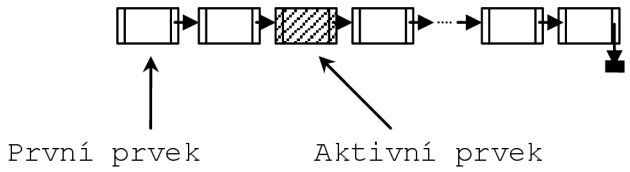


Obrázek 1: Prvek jednosměrného seznamu

Typ hodnoty prvku je libovolný, ale pro všechny prvky seznamu musí být stejný. Ukazatel na další

prvek je typu ukazatel na prvek. Pokud se tedy chceme dostat k jinému prvku než prvnímu, musíme projít celé pole prvků před tímto prvkem.

Celý jednosměrný seznam je definovaný prvním a aktivním prvkem, tzn., že máme přístup pouze k těmto dvěma prvkům. Všechny operace nad seznamem jsou pak vztaženy pouze k těmto dvěma prvkům.



Obrázek 2: Jednosměrný seznam

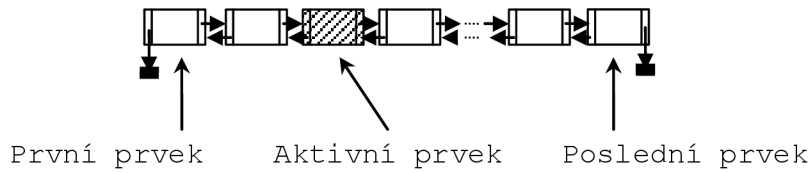
Jiná implementace jednosměrného seznamu je verze s "hlavičkou". Hlavička je neexistující prvek seznamu, který se vytvoří při vkládání prvku do seznamu a při ukončení vkládání se smaže. Hlavička má tu výhodu, že odstraňuje úvodní dialog kódu pro eventuální posun pole kvůli vkládání, který se musí vždy provést pro první prvek. Posun celého pole tak může proběhnout v jednom cyklu.

4.1.2 Dvojsměrný seznam

Dvojsměrně vázaný lineární seznam je abstraktní datová struktura obsahující prvky, které v sobě nesou hodnotu svého prvku a dva ukazatele – na předchozí a další prvek.

Obrázek 3: Prvek dvojsměrného seznamu

Dvojsměrný seznam je identická struktura seznamu jednosměrného. Rozdíl oproti jednosměrnému seznamu je v obsahu prvku seznamu. Díky tomu, že prvek obsahuje ukazatel na předchozí prvek, je možné seznam procházet dvěma směry, od prvního k poslednímu a od posledního k prvnímu. Seznam je tedy definovaný jak prvním a aktivním prvkem, tak i posledním prvkem. Obecně se používá dvojsměrný seznam, pokud je však dvojsměrnost nadbytečná použije se seznam jednosměrný.



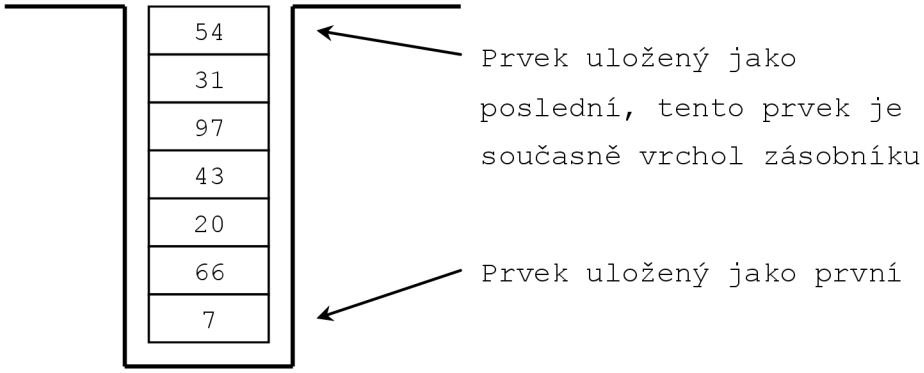
Obrázek 4: Dvojsměrný seznam

Poslední z řad seznamů je kruhový seznam. Tento seznam je implementován buď jako jednosměrný nebo jako dvojsměrný. Na rozdíl od těchto seznamů má kruhový seznam ustanoveno pravidlo, že následník posledního prvku je první prvek a předchůdce prvního prvku je prvek poslední (při implementaci jako dvojsměrný seznam).

4.2 Zásobník

Zásobník je homogenní, lineární, dynamická datová struktura, která je specifická svými operacemi přidávání a odebírání prvků. Tato struktura má v sobě specifické místo, které se nazývá „vrchol zásobníku“. Vrchol je jediné místo v zásobníku, kde máme přístup. Přidávání a odebírání prvků se děje pouze na tomto místě.

Tato struktura je též nazývána strukturou typu LIFO (Last In, First Out), tzn., že prvek vložený jako poslední bude jako první vybrán. Při přidávání prvků se vytvoří nový vrchol zásobníku, který má následující prvek starý vrchol zásobníku. Prvky se tak posunují směrem od vrcholu zásobníku. Pokud je prvek odebírán, odstraní se prvek ukazující vrchol a ukazatel vrcholu se posune na následující prvek.



Obrázek 5: Zásobník

Nejčastější implementace zásobníku je pomocí pole nebo lineárního seznamu. Pokud použijeme implementaci pomocí pole, zásobník ukládá své prvky do statického pole a vrchol

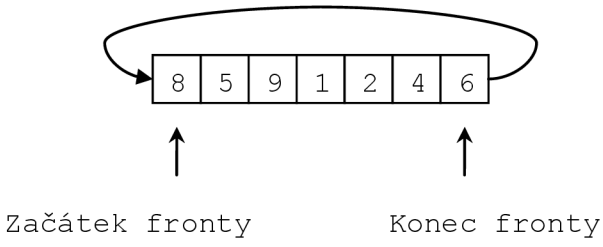
zásobníku je vždy poslední prvek v poli. Pro znázornění vrcholu je vytvořena proměnná celočíselného typu, která ukazuje, na jaké pozici v poli se vrchol nachází. Velikost zásobníku je limitována velikostí statického pole, kde ukládáme prvky. Pokud proměnná ukazující vrchol zásobníku se rovná velikosti pole, zásobník hlásí, že je plný. V tomto okamžiku není možné vložit na zásobník nový prvek. Stav plného zásobníku se zjišťuje speciální funkcí, která je součástí implementace zásobníku. Zásobník implementovaný lineárním seznamem používá ke svázání prvků ukazatele. Prvky musí obsahovat kromě hodnoty prvku také ukazatel na další prvek. Jak tento prvek může vypadat je znázorněno v kapitole 4.1.1. Je zapotřebí vytvořit proměnnou typu ukazatel, která bude znázorňovat vrchol zásobníku. Při inicializaci se do této proměnné vloží hodnota NULL. Při vkládání na zásobník se vloží nový prvek na začátek seznamu a tato proměnná se přepíše ukazatelem na tento prvek. Pokud porovnáme tyto implementace tak při implementaci polem je nutno hlídat přetečení zásobníku, tzn., abychom nepřidávali do plného pole, zatímco u implementace seznamem to není nutné, neboť seznam je limitován velikostí operační paměti.

4.3 Fronta

Fronta je identická datová struktura jako zásobník. Hlavním rozdílem je, že fronta vkládá a čte prvky na různých koncích struktury. Fronta pracuje stejným způsobem jako fronta v reálném světě, např. fronta lidí v samoobsluze.

Tato struktura se taky označuje jako datová struktura typu FIFO (First In First Out). V překladu to doslova znamená „první dovnitř, první ven“. V jazyku fronty to znamená, že ten prvek, který je vložen do fronty jako první, jako první taky bude přečten. První prvek se vloží na konec fronty, další prvek je pak vložen za první, čili se stane posledním prvkem. Prvek, který bude první přečten, je ten na úplném začátku.

Fronta je často využívána např. v logice správy sítí při vyřizování požadavků na server.



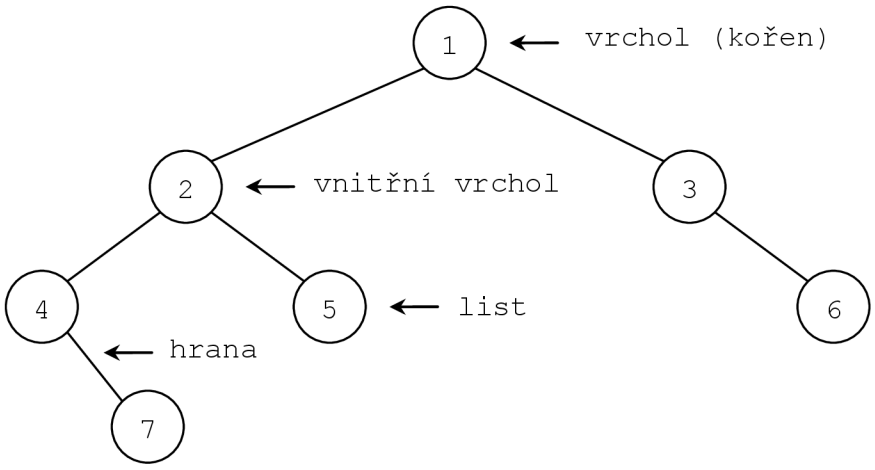
Obrázek 6: Fronta

Fronta se realizuje podobně jako zásobník. Jsou opět dva způsoby implementace – lineárním seznamem nebo statickým polem. Při implementaci polem se prvky vkládají na začátek pole a na konci se odebírají a čtou. Jsou vytvořeny dvě celočíselné proměnné pro ukazatele na začátek a konec fronty. Proměnná ukazující na začátek znázorňuje prvek, který bude první přečten. Proměnná

ukazující na konec znázorňuje místo, kde se bude vkládat další prvek. Fronta má podobu kruhového seznamu a neustálým čtením a vkládáním mění svojí podobu. Implementace pomocí lineárního seznamu fronty je hodně podobná implementaci seznamu zásobníku. Je zapotřebí však změnit orientace čtení a vkládání tak, aby odpovídalo smyslu fronty. Jak bylo zmíněno už v zásobníku, je zapotřebí rezervovat paměť pro ukazatele lineárního seznamu.

4.4 Binární strom

Při psaní této kapitoly jsem se inspiroval z [5]. Binární strom je poněkud odlišná datová struktura než předešlé struktury. Prvky nejsou řazeny jako posloupnost prvků, ale jsou hierarchicky uspořádány do tzv. "stromu". Každý prvek stromu se nazývá vrchol. Každý vrchol má maximálně dva své následníky spojené hranou, které jsou označeny jako levý a pravý. Žádný vrchol nemá dva potomky stejného typu. Tento útvar se nazývá binární strom. Následníkem vrcholu může být i další strom – ten je pak nazýván podstrom. Nejvyšší vrchol stromu se nazývá vrchol nebo kořen. Všechny ostatní vrcholy stromu jsou tedy "vnitřními vrcholy" stromu. Nejspodnější vrcholy stromu, které nemají následníky, jsou listy.



Obrázek 7: Binární strom

Implementace binárního stromu se provádí opět pomocí pole nebo lineárního seznamu. Při implementaci polem bude mít kořen index v poli 1, jeho levý následník bude mít index 2 a pravý pak index 3. Vzorce pro výpočet indexů dalších následníků by mohly vypadat takto:

- levý následník má index 2i
- pravý následník index 2i + 1
- předchůdce, pokud existuje, má index i/2 (celočíselně)

Tato implementace binárního stromu není efektivní řešení. Pro vytvoření je zapotřebí znát maximální velikost stromu, abychom mohli vytvořit dostatečně velké pole. Navíc pole musí obsahovat prvky pro pozice neobsazené vrcholy.

Častěji se používá implementace seznamem. Prvek obsahuje hodnotu uzlu a dva následníky (levý a pravý) typu ukazatel na ukazatel. U této implementace je složitější průchod stromem, např. pro výpis prvků stromu. Pokud to byl obyčejný seznam, postupovali jsme od prvního prvku k poslednímu pomocí ukazatele na další prvek. U stromu jsou definovány 3 typy průchodů:

- Přímý průchod (preorder) - navštívíme vrchol a potom levý a pravý podstrom.
- Vnitřní průchod (inorder) - navštívíme levý podstrom, vrchol a pravý podstrom.
- Zpětný průchod (postorder) - navštívíme levý a pravý podstrom a potom vrchol.

Pro náš ilustrační strom budou průchody vypadat pak takto:

- Preorder – 1, 2, 4, 7, 5, 3, 6
- Inorder – 4, 7, 2, 5, 1, 3, 6
- Postorder – 7, 4, 5, 2, 6, 3, 1

V binárním stromu byly klíče (čísla podle, kterých se vkládá a vyhledává v stromu) samotné hodnoty prvku. Implementace, kde klíče a hodnoty uzlu jsou dvě proměnné, se nazývá "binární vyhledávací strom". Vkládání do binárního vyhledávácího stromu podle rekurzivní definice se děje takto: *Je-li strom prázdný, nový prvek se stane jeho kořenem. Jinak je-li klíč nového prvku menší než klíč kořene, prvek vložíme do levého podstromu. Je-li klíč nového prvku větší než klíč kořene, vložíme ho do pravého podstromu.*

Vkládání do binárního stromu může vést i ke značné neefektivitě. Prvky mohou být ukládány neustále do pravého potomka vrcholu a tím vznikne strom s výškou (počet hran vedoucí k nejvíce zanořenému listu) rovnající se počtem prvků ve stromě – 1. V tomto případě strom připomíná spíš lineární seznam a binární strom zde ztrácí smysl. Tento problém řeší AVL strom. Je to výškově (pro každý vrchol platí, že výška jeho podstromů je stejná nebo se liší o 1) vyvážený binární vyhledávácí strom, takže AVL bude mít vždycky tvar stromu.

5 Program pro animační demonstraci

Pro vytvoření programu pro demonstrační animaci jsem zvolil prostředí Macromedia FLASH MX 2004. Tato kapitola obsahuje popis vývojového prostředí FLASH a vývoj programu pro animační demonstraci. Tento program i jeho zdrojový soubor je součástí příloh.

5.1 Seznámení se s FLASHem

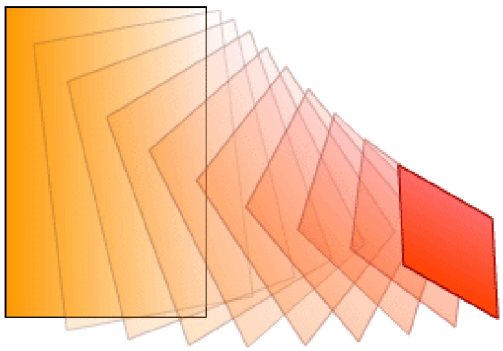
Tato kapitola je inspirována z [7]. V dnešní době, pokud se rozhodne nějaká firma prezentovat na internetu, klade velký důraz na design svých webových stránek. Stránky mohou být statické, ale musí být pěkné pro oko, čili nějaké animace, bannery popř. nějaké rozbalovací menu. K animaci se dosud používaly animované gify, Java aplety nebo DHTML. Animovaný gif je několik obrázků uložených do jednoho souboru. Po načtení stránky prohlížeč cyklicky zobrazuje obrázky v souboru animovaného gifu. Každý pohyb v takovém gifu je separátní obrázek, proto mohou být animované gify velmi velké a i malé animace načítají pomalu. Animované gify nejsou interaktivní a opakují stejné obrázky.

K tomu, aby se daly animace ovládat, pozastavit nebo vyčkat na reakci uživatele je výborný nástroj FLASH. FLASH v sobě zahrnuje jak editor grafiky, tak prostředí pro tvorbu animací. Díky své jednoduchosti zaujímá v této oblasti tvorby animace velmi pevnou pozici.

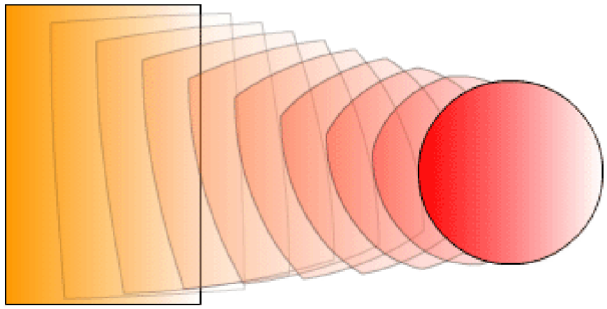
FLASH je editor vektorové grafiky. Vektorová grafika je definována pouze pomocí čar a výplní na rozdíl od grafiky rastrové, která pracuje s pouze bitmapou (pixely). Objekt vektorové grafiky je tvořen pouze vektorem. Tento způsob vykreslování je mnohem náročnější na procesor, neboť procesor musí vypočítat, kde se má objekt vykreslit. Výhoda je však, že cílový soubor s grafikou je mnohem menší a objekty nejsou rozměrově limitovány (mohou se libovolně zvětšovat a zmenšovat bez ztráty kvality). Navíc výplně a čáry mohou být průsvitné, nebo vybarvené přechodovou výplní s minimálním zvětšením souboru.

Vytvoření animace ve FLASHi lze provést pomocí skupiny obrázků, které půjdou za sebou. Tento způsob však není příliš efektivní. Ve FLASHi existují dva způsoby jak automaticky vykreslit pohyb nějakého objektu. První z nich je tzv. MotionTween (viz. obrázek 8a: MotionTween). Je to pohyb, kdy objekt může během animace měnit svou velikost, barvu, pozici, průhlednost. Tato animace se skládá ze dvou klíčových snímků. Tyto snímky však musí obsahovat instance (kopie objektů z knihovny) stejných symbolů. Pokud tedy vložíme do prvního snímku instance objektů A a B, do koncového snímku musíme dát opět instance A a B. Druhým způsobem jak vytvořit automatický pohyb je ShapeTween. Na rozdíl od MotionTween, v tomto druhu animace klíčové snímky musí obsahovat pouze čistou grafiku a ne instance stejných symbolů. U pohybu ShapeTween

může grafika měnit také svůj tvar, čili z kolečka může vzniknout např. čtvereček (viz. obrázek 8b: ShapeTween).



Obrázek 8a: MotionTween



Obrázek 8b: ShapeTween

Po vytvoření FLASH animace v editoru se tato animace exportuje do souboru formátu SWF. Nevýhodou je, že tento export není vratný, čili animaci už nemůžeme upravovat, pokud nevlastníme zdrojový soubor. Přehrávání animací se děje ve webovém prohlížeči (musí mít nainstalován FLASH plugin) nebo pomocí zvláštního přehrávače. Většina uživatelů takový prohlížeč nemá, proto se k FLASH animaci přibaluje přehrávač a vznikne tzv. projektor (EXE soubor), který se dá spustit na kterémkoli počítači. Vytvoření projektoru se používá zejména u FLASHových prezentací nebo her, neboť velikost je cca o 500KB větší než u běžného souboru.

5.1.1 Historie FLASHe

Tento text je citován z [6].

Počátky FLASHe spadají někdy do roku 1994. Tehdy vlastně ještě nešlo o FLASH, jak jej známe dnes. Jmenoval se SmartSketch a byl založen na Javě. Od tohoto směru se však ustoupilo. Java jako programovací jazyk totiž nevyhovoval nárokům na rychlost a spolehlivost. Když se někdy v roce 1995 objevily prohlížeče podporující zásuvné moduly typu PLUG-IN, byl SmartSketch přejmenován na FutureSplash Animator a byla zcela změněna jeho podoba. Macromedia v této době pracovala na svém projektu s názvem Shockwave.

V roce 1996 Macromedia kupuje FutureSplash Animator a vzniká tak Macromedia FLASH 1.0. Tato verze sice ještě neobsahuje skriptování ActrionScript, ale nastiňuje nadějný směr vývoje webových animací.

Následuje verze 2, která dovoluje základní skriptovou manipulaci s přehráváním animace. Objevují se prvky jako tlačítko a grafika, vzniká proměnná.

Verze 3 přináší ozvučení animací a s tím spojené příkazy, dále pak příkazy typu fscommand a možnost skriptově načítat SWF animace.

Verze 4 je doslova revoluční. Celý ActionScript je přepracován, vzniká velké množství příkazů. Vznikají funkce, příkazy pro movieclip, podmínky a smyčky, nové operátory, načítání proměnných ze souboru a spoustu dalších.

Verze 5 je opět přelomová. Vznikají objekty se svými metodami a vlastnostmi. Je možno vytvářet vlastní funkce. Vznikají komponenty. Většina příkazů je přeorientována na objekty. Vzniká přehlednější dot syntaxe a podporována je komunikace se serverem pomocí XML Socket.

Verze 6 (MX) přináší podstatné rozšíření objektů a metod. Vznikají UI komponenty, vzniká spolupráce s videem. FLASH player 6 podporuje obousměrný streamovaný přenos zvuku a videa pomocí kamer a mikrofonů. Je vyvinut nový komunikační protokol RTMP a Server-side ActionScript pro komunikaci se serverovými službami a serverový balík FLASH Communication Server MX.

Následuje 7. verze, pojmenovaná MX 2004. Poprvé se produkt rozděluje na dvě podoby: MX 2004 a MX 2004 Pro. Obě verze nabízí vylepšené rozhraní, nové efekty (zejména pro práci s textem - podpora CSS), vylepšenou časovou osu, rozšíření kompatibility importovaných objektů (PDF, EPS, …), konečně také panel „historie“, šablony pro vytvoření složitých Motion a Shape Tweenů, vylepšené trasování bitmap a ActionScript 2.0, který se vyznačuje větší přímočarostí a robustností.

Verze Professional navíc nabízí podporu externího ActionScriptu, lepší zpracování video souborů, podporu zvuků ve formátu MIDI pro mobilní zařízení, vylepšené formuláře a jednoduché vytváření slidů a prezentací ve speciálním módu.

3. prosince 2005 společnost Adobe Inc. odkupuje Macromedii s celým portfoliem produktů, následuje vydání FLASHe 8. Ten se vyznačuje pokročilou prací s videem včetně alpha kanálu, blend filtry, možností cacheovat vektory jako rastry, ClearType antialiasingem písma, vytvářením multiplatformních animací a spoustou dalších vylepšení v ActionScript 2.0

Adobe Systems Inc. je v současnosti jednou z největších softwareových společností s bezmála 6 000 zaměstnanci a ročním výnosem kolem 2 bilionů dolarů. Produkty pro webdesign jsou nyní soustředěny do balíku Studio.

5.1.2 Prostředí Macromedia FLASH MX 2004

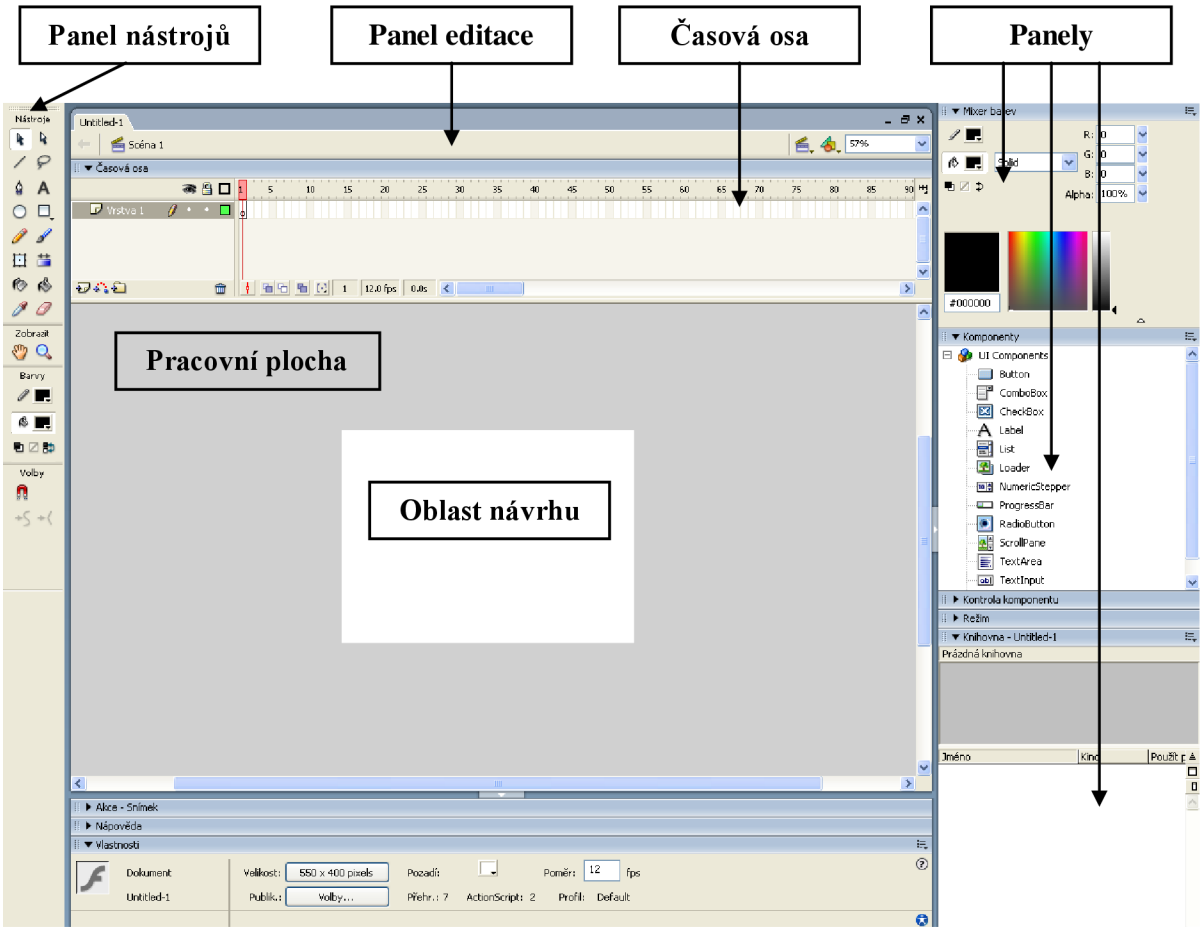
Při otevření nového dokumentu se objeví obrazovka rozdělena do 6 částí.

Panel nástrojů – tento panel slouží k editaci grafiky. Jsou zde uvedeny všechny funkce, které je možno s grafikou provést.

Panel editace – na tomto panelu je zobrazováno vaše aktuální umístění v rámci projektu jako např. jméno scény, počet tlačítek, které umožňují editaci scény a symbolů, zoom.

Časová osa – jedna z nejdůležitějších funkcí ve FLASHi. Umožňuje uživateli, aby se pohyboval v různých časových úsecích animace a měnil tak objekty ve scéně v tomto čase.

Panely – mezi tyto panely patří např. knihovna (zde se uchovávají objekty, které používáme ve scéně), komponenty (předdefinované objekty – tlačítko, rozbalovací seznam, …) a také panel akce, do kterého se píše kód ActionScriptu.



Obrázek 9: Základní obrazovka prostředí

Pracovní plocha – světle šedivá plocha okolo oblasti návrhu se nazývá pracovní plocha. Konečný uživatel pří přehrávání videa nebude nic vidět z toho, co se objevuje na pracovní ploše. Pokud chcete, aby některé objekty nebyly vidět, můžete je umístit na pracovní plochu. Pokud např. chcete vytvořit animaci přilétávajícího ptáka, můžete umístit jeho obrázek mimo oblast návrhu, na pracovní plochu.
Oblast návrhu – vše co je v této oblasti se zobrazuje v samotné animaci. Velikost oblasti, barvu, měřítko návrhu lze měnit.

5.1.3 Jazyk ActionScript

Text této kapitoly je inspirován z [8]. ActionScript (dále jen AS) je vnitřní jazyk, který používá FLASH MX 2004. Jeho formát je podobný (nikoliv identický) JavaScriptu. Díky přidání ActionScript do projektu je možné vytvořit plně interaktivní animaci nebo prezentaci. Animaci můžete nastavit tak,

že uživatelovy události, jako je kliknutí na tlačítko a stisknutí klávesy spustí skripty, které říkají animaci, jakou akci má provést. Umožňuje také tvorbu formulářů pro vkládání a odesílání údajů. Pomocí příkazů AS můžete volat příkazy jiných jazyků jako např. PHP.

Psaní kódu se provádí do panelu Akce (Actions), kde můžete využít jednoduchého průvodce při zadávání příkazů, který vloží funkci do kódu s předepsanou syntaxí (mód Normal) nebo psát kód ručně (mód Expert).

ActionScript může být připojen k instanci tlačítka nebo instanci videoklipu nebo klíčovému snímku na časové ose. AS však nemůže být připojen na instanci grafického symbolu.

ActionScript má v drtivé většině stejnou syntaxi příkazů jako Javascript. Jedny z nejzákladnějších příkazů AS, které jsou využity v projektu:

- Goto – přechod na určitý snímek nebo na určitou scénu, popř. ve spojení „GotoAndPlay" přechod na snímek a jeho spuštění, v parametru funkce lze zadat číslo snímku nebo jeho název
- Play, Stop – slouží k spuštění animace nebo její zastavení
- attachMovie – vytvoří instanci (kopii) objektu z knihovny a vloží jej do scény

Všechny příkazy (Goto, Play) jsou adresovány té časové ose nebo tomu objektu, kde se zrovna nacházíme. Pokud však chceme adresovat jiné objekty ve scéně nebo jiné časové osy, musíme použít tzv. tečkovou syntaxi. Budu-li chtít z hlavní časové osy definovat proměnnou "cislo" v objektu nazvaném "objekt", bude příkaz vypadat takto:

```
objekt.cislo = 5;
```

Toto je relativní zápis. Další možnost jak definovat cestu je zadání absolutní. Je vždy nutné použít alias "_root".

```
_root.objekt.cislo = 5;
```

Takto definovaná cesta bude fungovat z kteréhokoli místa v dokumentu.

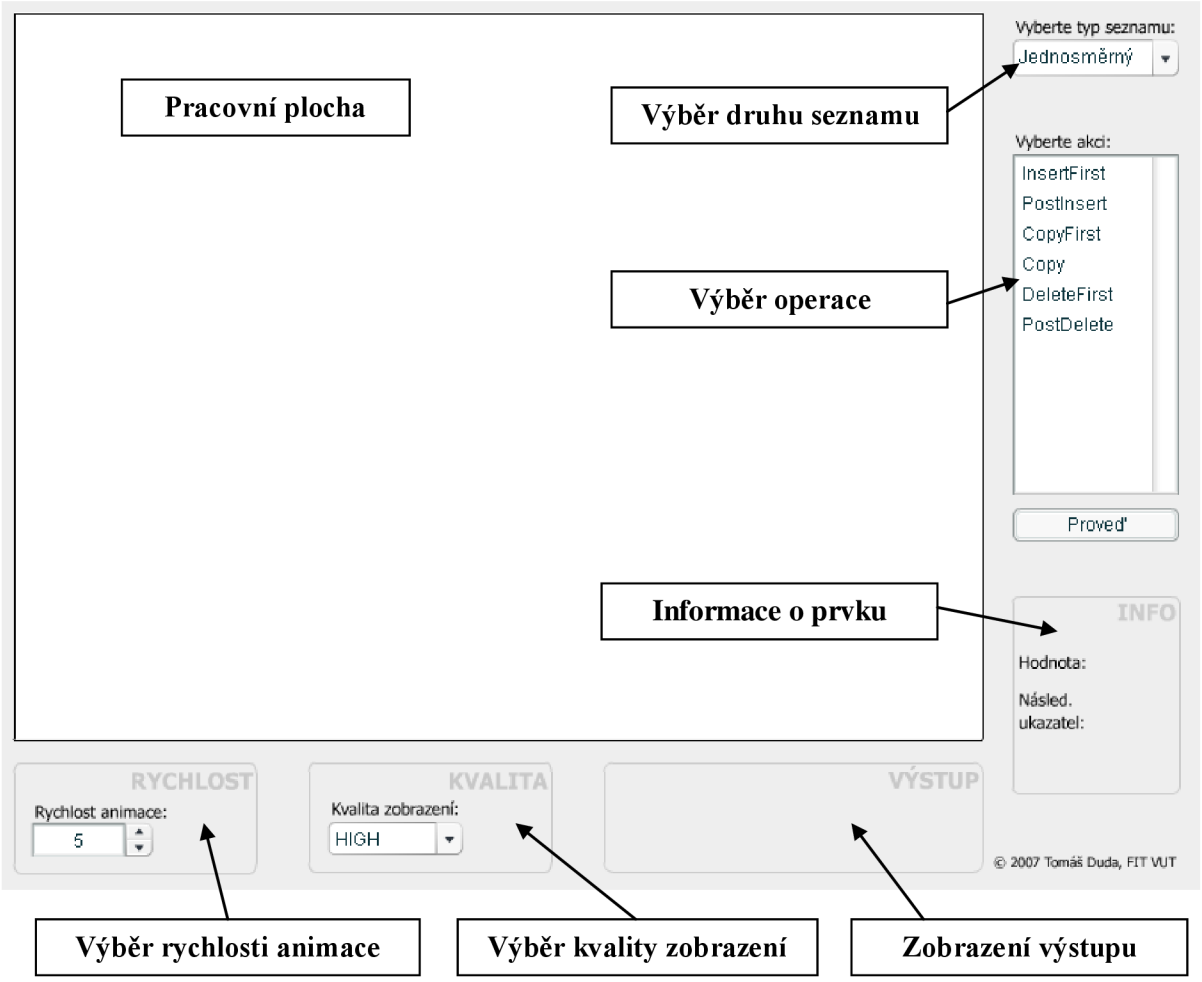
Objekty ve scéně se ukládají do tzv. vrstev. Každý objekt, který vkládáme na scénu, by měl být vložen do samostatné vrstvy. Díky vrstvám lze objekty překrývat a nastavovat, který prvek bude překrytý a který ne.

Práce s ActionScripty je jedním z technicky nejvíce náročných aspektů práce v prostředí FLASH. Všechny akce a kreslení pomocí panelů nástrojů na scéně lze provést pomocí AS, proto je nejdůležitějším nástrojem v prostředí FLASH.

5.2 Vývoj programu

V této kapitole je popsán vývoj samotné animace. Kapitola je rozdělena na podkapitoly implementace 5.2.2 a uživatelské rozhraní 5.2.1, kde je ukázáno, jak aplikace vypadá a funguje.

5.2.1 Uživatelské rozhraní



Obrázek 10: Úvodní obrazovka aplikace

Po spuštění aplikace se objeví úvodní obrazovka (viz obrázek 10). Na obrázku vidíme pracovní plochu a ovládací prvky aplikace. Na pracovní plochu jsou vkládány prvky seznamu. Podle toho, jaký seznam zvolí uživatel, se vkládají prvky na plochu. Jsou rozlišeny různými barvami, aby bylo rozeznat, o jaký seznam jde. Kruhový seznam má stejnou barvu jako jednosměrný, neboť je stejně implementován.

Seznam "Výběr operace" neobsahuje operace s aktivitou seznamu. Pokud chceme udělat nějaký prvek aktivní, klikneme na něj a prvek se zbarví do červena.

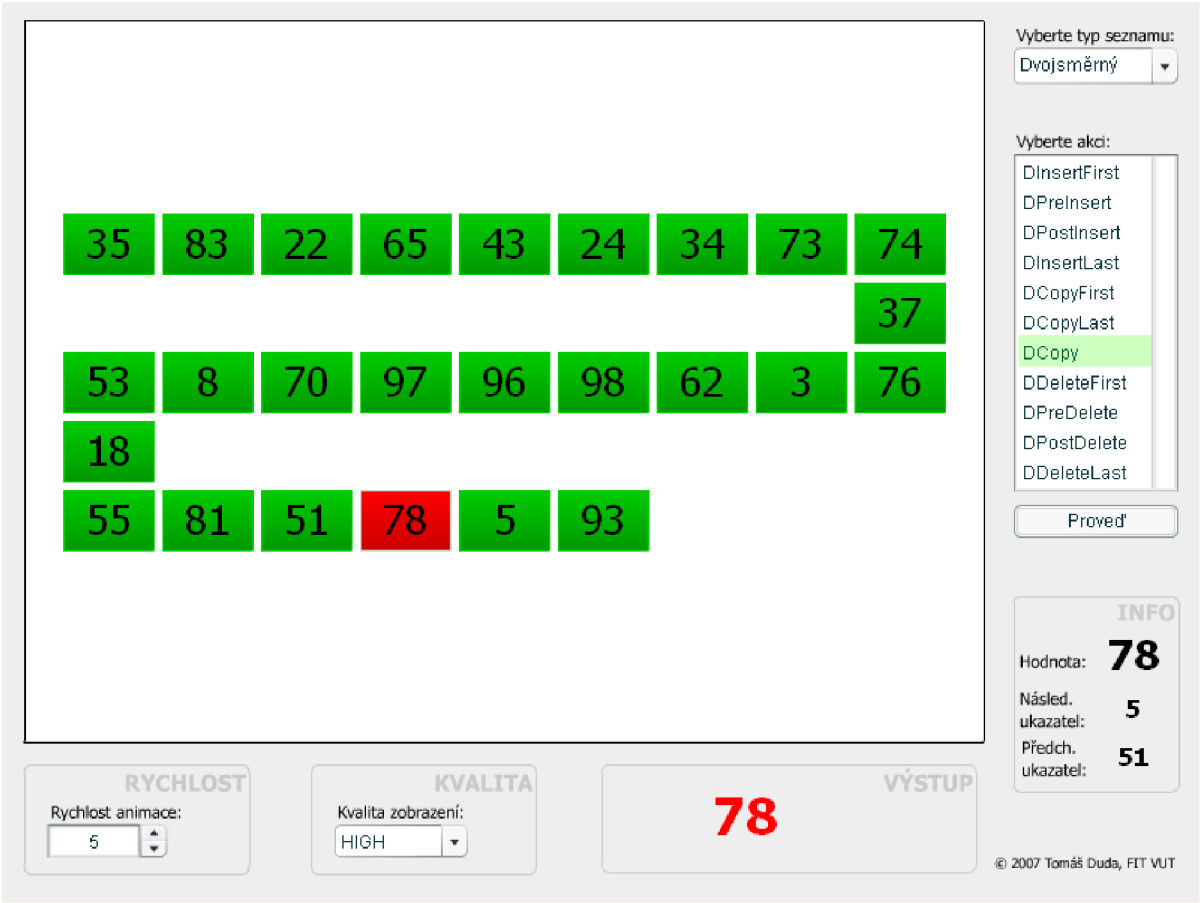
Prvky se postupně vykreslují zleva doprava po pravý okraj pracovní plochy. Po dosažení tohoto okraje se vykreslí 2 prvky dolů a následně se budou prvky vykreslovat zprava doleva. Seznam pak vytváří iluzi "hada". Uvnitř prvku se vytváří pseudonáhodné číslo, které je v rozmezí od 0 do 100. Toto číslo slouží jen k identifikaci prvků.

Informace o prvku, nad kterým se nachází myš, zobrazuje panel Info. Tento panel zobrazuje hodnotu prvku neboli číslo, které je v něm zobrazeno a ukazatel na následující prvek. U dvojsměrného seznamu se zde objeví ještě položka ukazatel na předchozí prvek.

V panelu Výstup se zobrazují výsledky čtení funkcí Copy a CopyFirst a také chybové hlášky, které nastanou při špatném zavolání funkce nad seznamem, např. při čtení hodnoty aktivního prvku nad neaktivním seznamem.

Kvalita zobrazení se nastavuje pomocí panelu Kvalita. Musíme počítat s tím, že aplikace bude spouštěna i na starších a méně výkonnějších počítačích. Pokud snížíme kvalitu zobrazení animace, aplikace poběží plynuleji i na starších počítačích.

Panel Rychlost nastavuje, s jakou rychlostí budou zobrazovány animace při posunu seznamu. Rychlost animace má 10 úrovní.



Obrázek 11: Ukázka běhu aplikace

5.2.2 Implementace

Implementaci aplikace lze rozdělit na dvě části. První část bylo vložení ovládacích prvků na pracovní plochu a jejich umístění. V druhé části bylo zapotřebí přiřadit objektům scény různé akce a napsat k nim ActionScripty.

5.2.2.1 Inicializace proměnných

Po spuštění se nejdříve nastaví příkazem *fscommand* globální proměnné předdefinované FLASHem. Jsou to proměnné fullscreen (určuje, jestli se animace bude moci roztáhnout na celou obrazovku), allowscale (určuje, jestli animace bude moci měnit svou velikost při chycení ukazatelem za okraj animace), showmenu (určuje, jestli se při kliku pravým tlačítkem na animaci objeví FLASH menu) a trapallkeys (určuje, jestli budou funkční FLASHovské klávesové zkratky).

Při inicializaci se vytvoří pomocná struktura *Tpolozka*, definují se proměnné *rychlost* (v ní je uložena rychlost animace) a *seznam* (určuje, se kterým seznamem se zrovna pracuje), pomocné struktury *aktivni* (uchování informací o aktuálním prvku) a *polozka*, pole struktur s typem *Tpolozka* s názvem *polozkySeznamu* a další pomocné proměnné.

Dále jsou vytvořeny pomocné funkce aktualizaceAktivni, smazJednSeznam, smazDvojSeznam, smazKruhSeznam a smazJednPlochu. Funkce aktualizaceAktivni se zavolá, pokud se změní aktivní prvek a v ní se vyhledá podle hodnoty v poli *polozkySeznamu* prvek a jeho informace se uloží do pomocné struktury *aktivni*. Funkce smazJednSeznam, smazDvojSeznam a smazKruhSeznam mají za úkol smazat všechny položky na ploše. Podle toho jaký seznam je na ploše, zavolá se příslušná funkce. Funkce v cyklu *for* maže prvky seznamu adresované od indexu, který se rovná délce seznamu minus 1, do indexu 0. Přesný popis adresace prvků je popsán v kapitole 5.2.2.5.

5.2.2.2 Umístění ovládacích prvků na plochu

Prostředí FLASH zahrnuje v sobě předdefinované ovládací prvky. Z těchto předdefinovaných prvků jsem použil 2 rozbalovací seznamy (první na výběr typu seznamu a druhý k výběru kvality animace), číselný krokovač (k nastavení změny rychlosti animace), seznam (k zobrazení operací, které je možné provádět s daným typem seznamu). Tyto ovládací prvky jsem rozmístil kolem pracovní plochy tak, aby se vešly do rozměrů animace 800x600.

5.2.2.3 Uchovávání údajů o vytvořených prvcích

Pro ukládání prvků, které jsou přidány na plochu, byla vytvořena struktura *Tpolozka*. Tato struktura je tvořena 5 položkami, které představují vlastnosti prvku. Jedná se tedy o:

x …… x souřadnice umístění prvku

y …… y souřadnice umístění prvku

hodnota … proměnná udávající náhodné číslo, které je zobrazeno uvnitř položky

poradi …… proměnná udávající v jakém pořadí se aktuální prvek nachází od začátku seznamu

vrstva …… proměnná udávající v jaké vrstvě se prvek nachází

X-ová souřadnice se nachází v intervalu od 72 do 600 pixelů. Y-ová souřadnice je v intervalu od 70 do 438 pixelů. Hodnota je pseudonáhodné číslo zobrazené uvnitř prvku a nachází se v intervalu od 0 do 100. Každý prvek se nachází v samostatné vrstvě. Pokud by se totiž nacházely ve stejné vrstvě, později vložený prvek by překryl ten dřívější.

Tyto prvky, které se vloží na plochu, se vzápětí uloží do pole vytvořeného pro účel ukládání informací o prvcích nacházejících se na ploše. Pole se nazývá *polozkySeznamu*. Pole není omezeno žádným limitem prvků, jsme tedy omezeni pouze velikostí operační paměti.

5.2.2.4 Vkládání prvků na plochu

Prvky se vkládají ihned po vybrání jakékoli operace související s insert a stisknutí tlačítka *Proveď*. Po tomto úkonu se na ploše objeví nový prvek. Prvky se začínají objevovat uprostřed pracovní plochy, přesněji na pozicích x=72, y=254. Tato pozice je uložena v pomocných proměnných, které jsou vybírány, kdykoli se vkládá na první pozici kromě výjimek, kdy je seznam posunut.

Popis animování vkládání prvku na scénu je popsán v kapitole 5.2.2.6.

Prvních 9 prvků se vkládá horizontálně směrem doprava se vzdáleností 66 pixelů. Až se dosáhne pravého okraje, desátý a jedenáctý prvek se vloží vertikálně pod prvek devátý. Jejich posun od sebe je 46 pixelů. Další prvky se vkládají opět horizontálně s odstupem 66 pixelů, nyní se však prvky vkládají směrem k levému okraji. Po dosažení levého okraje se opět dva prvky budou vkládat horizontálně pod sebe s posunem 46 pixelů. Tento postup se bude provádět tak dlouho, dokud na pracovní ploše nebude 49 prvků. Tato hodnota je dána velikostí pracovní plochy 640x480 pixelů a velikostí vkládajících prvků 60x40 pixelů. Navíc při každém přidání prvku směrem dolů se celý již vykreslený seznam posune o 23 pixelů směrem nahoru. Tímto se dosáhne toho, že seznam je vždy uprostřed pracovní plochy.

Při každém přidání prvku se aktualizují příslušné položky seznamu. Pokud se přidává prvek doprostřed seznamu, předešlé nebo následující prvky se neposunují, ale pouze se aktualizuje proměnná *poradi* (samozřejmě se aktualizují i souřadnice, pokud došlo ke změně) a prvek zůstává na stejném místě.

5.2.2.5 Adresace prvků

Jména prvků na pracovní ploše jsou tvořena ze jména a čísla vrstvy, ve které se nacházejí. Pro každý prvek je vytvořeno speciální jméno. Zde je popis, jak jsou vytvářena jména:

- Jednosměrný seznam – polozka_jedn_(číslo vrstvy)
- Dvojsměrný seznam – polozka_dvoj_(číslo vrstvy)
- Kruhový seznam – polozka_jedn_(číslo vrstvy)

Kruhový seznam má v názvu *jedn* proto, neboť kruhový seznam je implementován pomocí jednosměrného seznamu a tudíž je tvořen z prvků seznamu jednosměrného.

Pokud je adresován konkrétní prvek na pracovní ploše, je nutné ho adresovat speciálním výrazem. Ve FLASHi se volají proměnné nebo funkce s předponou, která reprezentuje umístění, kde se prvky nacházejí. Položky seznamu se nacházejí přímo v hlavní scéně, proto jejich výraz bude mít následující podobu: _root.polozka_jedn_(číslo vrstvy). Předpona „_root“ značí hlavní scénu, kde jsou umístěny položky seznamu. Pokud je adresováno např. textové pole s názvem textove_pole v nějaké

položce jednosměrného seznamu, která leží ve vrstvě 5, tak příkaz pro adresaci takového textového pole je _root.polozka_jedn_5.textove_pole. Pro konkatenaci jména a vrstvy do jednoho výrazu je použit příkaz eval, který se používá ve tvaru: eval("_root.polozka_jedn_"+5). Tímto se spojí adresa prvku se jménem vrstvy.

Pro vyhledání příslušných prvků v poli *polozkySeznamu* v programu je využit cyklus *for*, který prochází celé pole *polozkySeznamu*. Při procházení je využito toho, že každý prvek má jedinečnou hodnotu (číslo, které je zobrazeno uprostřed prvku) a jedinečné číslo vrstvy. Při hledání se porovnává hledaná hodnota nebo hledaná vrstva s hodnotami nebo vrstvami v poli *polozkySeznamu*. Podle toho, která položka je známa, se pracuje s hodnotou nebo vrstvou.

5.2.2.6 Animace prvků

Nejdříve než lze psát příkazy pro animaci, je důležité importovat knihovny s funkcemi pro animování. Pro import těchto knihoven se použije příkaz: import mx.transitions.Tween. V FLASHi existuje způsob, jak ovládat charakter animací, aby neměly jen konstantní průběh. Tento způsob se jmenuje Easing a pokud ho chceme využít, je nutné importovat jeho knihovny ve tvaru: import mx.transitions.easing.*. Easing umožňuje měnit rychlost animací na začátku, na konci nebo uprostřed. Po provedení těchto importů lze psát příkazy pro animaci prvků.

V tomto programu se využívá tzv. Motion Tween (vysvětlení pojmu v kapitole 5.1). Je využit při vkládání a pohybu prvků. Při vkládání je použit pohyb, kdy položka seznamu se plynule zvětšuje ze zmenšené položky, která má 30% z výsledné velikosti, do položky výsledných rozměrů 60x40 pixelů. Další animace je pohyb položek. Tato animace se provede, pokud se vkládá položka seznamu někam doprostřed seznamu. Všechny následující prvky se posunou na pozici svého následujícího prvku.

Animace se provede vytvořením Motion třídy. Např. pro vytvoření pohybu položky v x-ové souřadnici bude příkaz pro vytvoření Motion třídy vypadat takto:

myTween = new Tween(_root.polozka_jedn_5, "_x", None.easeOut, 72, 138, _root.rychlost, false);

Proměnná *myTween* znamená název animace, výraz *new Tween* vytváří novou Motion třídu podle svých parametrů. První parametr udává jméno položky, se kterou se animace bude provádět. Druhý parametr udává, jaká vlastnost se při animaci bude měnit. Třetí parametr znamená, jestli se využije metoda Easing a pokud ano, tak jaký druh. Následující dva parametry ukazují počáteční hodnotu vlastnosti před animací a konečnou hodnotu vlastnosti po animaci. Předposlední parametr udává rychlost animace. Poslední parametr udává, jestli předchozí parametr se udává v počtech snímků za sekundu nebo velikost časového odstupu mezi jednotlivými snímky (pokud je false, rychlost je udávána ve velikosti časových odstupů mezi jednotlivými snímky). Vysvětlení proměnné _root.rychlost je v kapitole 5.2.2.7.

5.2.2.7 Ovládací prvky

Programátor může každému ovládacímu prvku přidělit základní metody, jako např. On(change), která vykoná obsah svého těla při změně stavu ovládacího prvku. Tato konkrétní metoda byla využita u většiny ovládacích prvků.

Číselný krokovač v levém spodním rohu má za úkol měnit rychlost animace. Obsahuje metodu On(change) nesoucí v sobě switch, který nastavuje globální proměnnou rychlost adresovatelnou příkazem *_root.rychlost*. Tato proměnná obsahuje velikost časových odstupů mezi jednotlivými snímky. Proměnná se pohybuje v rozmezí od 30 do 2, čím menší, tím je odstup menší.

Rozbalovací seznam pro změnu kvality celé animace obsahuje v sobě metodu On(change), která hlídá hodnotu tohoto prvku. Pokud se změní hodnota, je přepsána globální proměnná *_quality* předdefinována FLASHem, která nastavuje kvalitu animace. Seznam má tři hodnoty HIGH, MEDIUM a LOW.

Pole výstup v sobě má dvě textové pole *info_vystup* a *info_chyba*. Tato pole jsou adresována, pokud je nutné vypsat hodnotu prvky při funkcích *copy* (*info_vystup*) nebo vypsat chybovou hlášku (*info_chyba*).

Animace má obsahovat výběr typu seznamu – jednosměrný, dvojsměrný, kruhový. K tomuto účelu slouží rozbalovací seznam "Výběr druhu seznamu". Nastavuje, jaké operace budou zobrazeny v seznamu "Výběr operace", které definují, jaké operace se dají provádět se seznamem na pracovní ploše. Přitom nastavuje i globální proměnnou *seznam* znázorňující, který seznam je vybrán. Pokud je na ploše vykreslený nějaký seznam, seznam je smazán metodami smazJednSeznam, smazDvojSeznam nebo smazKruhSeznam.

Seznam "Výběr operace" zobrazuje, jaké operace je možné dělat s aktuálním seznamem. Operace se vybere kliknutím na příslušnou operaci.

Tlačítku Proveď je věnována kapitola 5.2.2.8.

Panel *Info* obsahuje 3 textové pole, které zobrazují informace o prvku, na kterém se zrovna nachází ukazatel myši. Každá položka seznamu v sobě nese metodu On(rollOver), která při přejetí ukazatele přes danou položku přepisuje hodnoty těchto tří polí.

5.2.2.8 Struktura programu

V této podkapitole je popsáno hlavní chování animace.

Většina úkonů celé aplikace se děje v metodě On(press) tlačítka `Proveď`. Metoda On(press) obsahuje funkci `switch` rozskakující se podle aktuální hodnoty seznamu "Výběr operace". Před tím, než proběhne samotné vložení nebo odstranění, musí být zjištěno, kam se prvek vkládá, popř. odstraňuje a jestli se nemají už vytvořené prvky seznamu posunout. Pokud ano, seznam se posune a uloží se nové parametry prvků do pole *polozkySeznamu*.

Vložení prvku seznamu zajišťuje předdefinovaná funkce attachMovie. Tato funkce je volána s aliasem _root, neboť tento prvek umisťujeme do hlavní scény. Podle proměnné ukazující typ

seznamu, který je vybrán (jednosměrný, dvojsměrný, kruhový) se vytvoří instance prvku seznamu z knihovny. Po vytvoření nového prvku se v těle metody onClipEvent(load) prvku vygeneruje náhodné číslo, které bude zobrazeno ve vytvořeném prvku. Po vložení prvku na plochu se uloží parametry prvku (pozice prvku, číslo zobrazené v prvku, číslo vrstvy a pořadí prvku v seznamu) do pole *polozkySeznamu*. Zápis do pole struktur ve FLASHi musí obsahovat všechny položky struktury, které se v struktuře objevují. Pokud tedy chceme přepsat pouze jednu položku, zápis musí obsahovat všechny položky struktury. Je nutné tedy vyjmout staré hodnoty a zpátky je s novýma vložit. Takový zápis může vypadat takto:

```
_root.polozkySeznamu[i] = ( {x:200, y:200, hodnota:5, poradi:10, vrstva:8} );
```

Při mazání se využívá funkce removeMovieClip, která se volá ve tvaru (jméno mazaného prvku).removeMovieClip(). Po provedení tohoto vymazání se opět aktualizují všechny parametry zbývajících prvků (změna polohy a pořadí).

Aktivita prvku je vyřešena tak, že pokud uživatel klikne na jakýkoli prvek na ploše, změní se jeho pozadí z původní barvy na červenou. Tento problém by se dal jednoduše řešit s využití funkce GoAndStop, kdy při kliknutí na prvek by se provedla tato funkce a přešlo by se na další snímek, kde by byl prvek s červeným pozadím. Pomocí této funkce jsem ovšem nebyl schopný tento přechod naimplementovat. Musel jsem tedy použít „drastičtější“ metodu. Při kliknutí na prvek si program zapamatuje, na který prvek bylo kliknuto, funkcí smazJednPlochu smaže celou pracovní plochu, vykreslí nový aktivní prvek s červeným pozadím a pomocí informací uložených v poli *polozkySeznamu* je schopný vykreslit zbývající prvky seznamu.

Závěr

Tímto přepisem studentské opory je usnadněno studentům porozumět problematice abstraktních datových typů a algoritmů vyučovaných v předmětu Algoritmy tak, jak jsou definovány pomocí jazyka C. Pomocí programu pro demonstrační animaci studenti pochopí, jak probíhají operace nad seznamy, jak se mění ukazatele na další a předchozí prvky.

Opora je přepsaná i v textové části tak, aby odpovídala zásadám jazyka C. Nejobtížnější prací na tomto projektu bylo vytvořit animaci pro demonstraci lineárních seznamů. Pro vytvoření animace jsem zvolil prostředí Macromedia FLASH MX 2004. Výsledné FLASH animace jsou dnes hojně využívány na internetových stránkách, proto jsem se chtěl naučit v tomto prostředí pracovat. Bylo nutné nastudovat základy jazyka ActionScript, neboť tento jazyk není vyučován na naší fakultě. Zdroje informací jsem čerpal z internetových tutorialů a diskusních fór.

Práce na tomto projektu začala přibližně ve 4. semestru. Od této doby začaly konzultace s vedoucím projektu a zjišťovaly se podrobné požadavky, jak má být opora přepsána. Intenzivnější práce začala v 5. semestru, kdy jsem postupně začal přepisovat oporu. V této době bylo už vytvořeno přesné zadání projektu. Na konci tohoto semestru bylo specifikováno ze strany pana vedoucího, jak by měla vypadat aplikace pro animaci seznamů. Od této chvíle jsem postupně začal přepisovat příklady a algoritmy do jazyka C. Zhruba v polovině 6. semestru jsem začal pracovat na aplikaci. Tato práce trvala až do samotného odevzdání celé bakalářské práce.

Pro rozšíření tohoto projektu a pro lepší pochopení všech datových struktur, které se vyskytují v opoře, bych navrhl vytvořit animační program pro každou z těchto struktur. Opora by rovněž mohla být v budoucnosti multi-jazyčná a další generace studentů by mohly vytvořit variantu textu opory např. pro současně hodně oblíbený Javascript.

Literatura

[1] Fotr, J.: *Naprogramujte si vlastní hru v Macromedia Flash MX 2004*. Computer Press, Praha 2005, ISBN: 80-251-0576-8.

[2] Weiss, N.: *Flash MX professional 2004 pro vývojáře webových aplikací*. Zoner Press, Praha 2004, ISBN: 80-86815-12-9.

[3] Honzík, J.: *Algoritmy, Studijní opora*. FIT VUT, Brno 2005.

[4] Wikipedie, C (programovací jazyk) [online], 2007, [cit. 4.5.2007]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/C_(programovací_jazyk)>.

[5] Javaaloritmy, ADT Strom [online], 2007, [cit. 5.5.2007]. Dostupný z WWW: <http://javaalgoritmy.wz.cz/strom.htm>.

[6] Hozík, M., Trocha historie [online], 2004, [cit. 7.5.2007]. Dostupný z WWW: <http://flash.jakpsatweb.cz/index.php?page=seznameni>.

[7] Berka, R., Chapter 14. Macromedia Flash MX 2004 [online], 2004, [cit. 8.5.2007]. Dostupný z WWW: < http://www.cgg.cvut.cz/~apg/apg-tutorials03/ch14.html>.

[8] Berka, R., Action Script – FLASH, 3. Publikování na webu [online], 2004, [cit. 8.5.2007]. Dostupný z WWW: <http://www.cgg.cvut.cz/~apg/apg-tutorials03/ch03.html>.

Seznam příloh

Příloha 1. CD