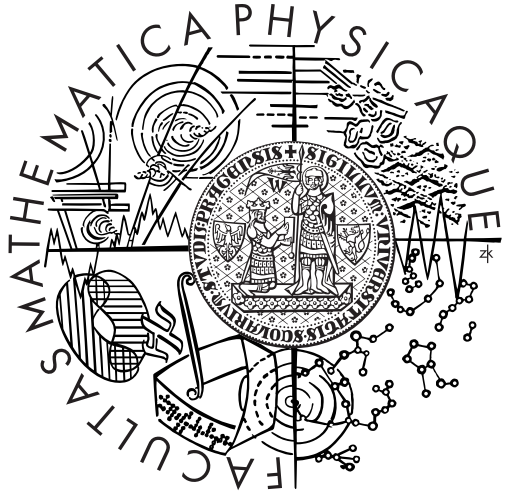
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁRSKA PRÁCA



Peter Šufliarsky

Interface pre programy v Delphi

Kabinet software a výuky informatiky

Vedúci bakalárskej práce: RNDr. Tomáš Holan, Ph.D.
Študijný program: informatika, obecná informatika

2009

Poďakovanie

Ďakujem môjmu vedúcemu za jeho podporu počas vývoja projektu a jeho cenné rady ohľadom programu i textu tejto práce.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a jej zverejňovaním.

V Prahe dňa 7.8.2009 Peter Šufliarsky

Obsah

Názov práce: Interface pre programy v Delphi

Autor: Peter Šufliarsky

Katedra (ústav): Kabinet software a výuky informatiky

Vedúci bakalárskej práce: RNDr. Tomáš Holan, Ph.D.

e-mail vedúceho: Tomas.Holan@mff.cuni.cz

Abstrakt:

Cieľom projektu je vytvoriť knižnicu pre prostredie Delphi, ktorá umožní externé čítanie, nastavovanie hodnôt vlastností jednotlivých ovládacích prvkov a simuláciu užívateľského vstupu v projektoch vytvorených pomocou prostredia Delphi. Súčasťou práce je ovládací program, ktorý demonštruje funkcionalitu uvedenej knižnice. Vykonávanie jednotlivých funkcií knižnice bude možné automatizovať pomocou skriptovacieho jazyka, ktorý je súčasťou práce a je interpretovaný ovládacím programom. V tomto jazyku bude možné popísat operácie užívateľského vstupu ako kliknutie myšou, vstup z klávesnice a sledovanie či zmenu stavu ovládacích prvkov programu s grafickým užívateľským prostredím. Vďaka tomuto bude možné dávkovo spúšťať takéto programy a následne testovať ich funkčnosť.

Kľúčové slová: Delphi, automatizácia, dávkové spúšťanie GUI, automatizované testovanie

Title: Interface for Delphi programs

Author: Peter Šufliarsky

Department: Department of Software and Computer Science Education

Supervisor: RNDr. Tomáš Holan, Ph.D.

Supervisor's e-mail address: Tomas.Holan@mff.cuni.cz

Abstract:

The aim of this project is to create a library for Delphi IDE, which enables external reading, changing values of component properties and simulation of user input in Delphi projects. A control software which demonstrates the functionality of that library is included. Execution of operations supported by the library will be automatizable with a scripting language, which is also included in the project and interpreted by the control software. With this scripting language user will be able to describe user input operations such mouse click, keyboard input, inspecting or setting property values for components in GUI programs. As a result, batch execution and automated testing of GUI programs will be possible.

Keywords: Delphi, automation, batch GUI execution, automated testing

Kapitola 1

Úvod

1.1 Preberaná problematika a prínos práce

Pri práci s počítačovým softvérom sa často stretávame s prípadmi, keď je potrebné viackrát vykonať nejakú operáciu alebo postupnosť operácií na rôznych údajoch. Príkladom môže byť úprava fotografií z dovolenky, organizácia veľkého množstva súborov ale i testovanie počítačových programov.

V prípade tých fotografií môžeme mať napríklad k dispozícii softvér, ktorý dokáže po niekoľkých kliknutiach zmeniť veľkosť jednej fotografie podľa požiadaviek užívateľa. Problém môže nastať, keď má užívateľ tých fotografií rádovo desiatky alebo dokonca stovky. Samostatná úprava každej z nich môže zabrať v konečnom dôsledku veľké množstvo času.

Ďalším príkladom môže byť problematika testovania softvéru. Počas vývoja softvéru je dôležité priebežne overovať jeho správne fungovanie. Toto sa robí väčšinou tak, že sa určí postupnosť operácií, ktoré tester na danom produkte vykoná a podľa výstupov zo softvéru dokáže určiť, či správne funguje. Počas vývoja sa softvér často mení, preto je nutné znovu a znovu overovať jeho funkčnosť a tak tester opakovane vykonáva zadané úkony.

Pri podobných problémoch sa naskytuje myšlienka, že spomínané úkony by bolo možné naprogramovať a namiesto užívateľa by ich vykonával počítač. Hlavnou výhodou tohoto riešenia je úspora času užívateľa alebo testera.

Cieľom tejto práce je navrhnúť a implementovať knižnicu pre prostredie Delphi, ktorá po začlenení do projektu umožní externému programu prečítať, v akom stave sú komponenty v okne aplikácie, samotné okno, podľa potreby tento stav zmeniť a tiež automatizáciu užívateľského vstupu. Súčasťou bude skriptovací jazyk, pomocou ktorého bude možné úkony bežne vykonávané užívateľom naprogramovať a vykonávať ich pomocou externého programu. Tento externý program komunikuje s programom, ktorý chceme automatizovane ovládať a jeho implementácia je súčasťou práce. Interface knižnice by mal byť jednoducho použiteľný, aby bolo možné vytvárať rôzne ovládacie programy a používať rôzne skriptovacie jazyky na automatizáciu prostredníctvom nej. Knižnica by mala byť jednoducho rozšíriteľná o podporu rôznych udalostí užívateľského vstupu.

1.2 Sprievodca po bakalárskej páci

Prvá kapitola

popisuje problematiku, ktorou sa práca zaoberá, jej ciele a možný prínos.

Druhá kapitola

sa zaoberá analýzou problému a existujúcich projektov podobného zamerania.

Tretia kapitola

popisuje implementáciu najdôležitejších častí projektu.

Štvrtá kapitola

je dokumentáciou pre užívateľa, popisuje použitie knižnice pre Delphi a externého ovládacieho programu. Popisuje tiež skriptovací jazyk, jeho syntax a možnosti.

Piata kapitola

dokumentuje projekt z hľadiska programátora a poskytuje informácie potrebné pre záujemcov o rozširovanie projektu.

Šiesta kapitola

obsahuje zhrnutie projektu, čo bolo implementované a čím by bolo vhodné v budúcnosti projekt rozšíriť.

Kapitola 2

Analýza a návrh

2.1 Analýza problému

Užívateľské prostredie alebo tiež GUI programu vytvoreného vo vývojovom prostredí Delphi pozostáva z jedného alebo viacerých okien. Spravidla v každom okne sa nachádza niekoľko ovládacích prvkov, komponentov, pomocou ktorých zadávame užívateľský vstup, alebo získavame výstup z programu. Príkladmi takýchto ovládacích prvkov sú tlačítka, hlavné menu alebo políčka pre zadávanie textu. Každé okno či ovládací prvok má definované vlastnosti, anglicky properties a udalosti, na ktoré môže reagovať, anglicky events.

Vlastnosti popisujú stav daného komponentu, jeho polohu, vzhľad a tiež údaje, ktoré v sebe má uložené. Príkladom môžes byť vlastnosť `Caption`, ktorá definuje nápis na tlačítku či v záhlaví okna. Ďalej to môže byť vlastnosť `Text`, v ktorej sa uchováva obsah textového políčka. Každá vlastnosť je nejakého údajového typu. Tento typ môže byť jednoduchý, ako napríklad celé číslo, znak alebo pravdivostná hodnota, no existujú tiež vlastnosti reťazcového typu, štruktúrovaného, výčtového, variantného alebo procedurálneho. V porovnaní s vlastnosťami jednoduchého údajového typu, bude potrebný iný prístup k vlastnostiam štruktúrovaného typu. Ako príklad poslúži vlastnosť `Font` objektu `TControl`, od ktorého sú odvodené užívateľské ovládacie prvky vrátane okien v prostredí Delphi. Pri pohľade do nástroja Object Inspector je vidieť, že nemá zmysel zisťovať hodnotu vlastnosti `Font`, ale zaujímavé sú až hodnoty zložiek vlastnosti `Font`. Týmito zložkami

sú: `Charset`, `Color`, `Height`, `Name`, `Pitch`, `Size` a `Style`. Nie všetky položky vlastnosti štruktúrovaného typu majú nutne jednoduchý údajový typ. Dôkazom toho je napríklad vlastnosť `TCustomForm.Menu`, ktorá má prvok `Images`. Preto pre poskytnutie funkcionality na úrovni nástroja Object Inspector bude nutné vytvoriť mechanizmus, ktorý dokáže rekurzívne prehľadávať vlastnosti ovládacích prvkov.

Ovládacie prvky v prostredí Delphi majú spravidla definovanú istú množinu udalostí. Udalosťou môže byť napríklad kliknutie myšou, stlačenie tlačítka na klávesnici alebo posúvanie či prekreslenie okna. Pokiaľ má program alebo ovládací prvok reagovať na nejakú udalosť vykonaním naprogramovaných operácií, tak sa vytvorí preň publikovaná metóda a jej adresa sa priradí ako hodnota adekvátnej udalosti definovanej pre ovládací prvok. V prípade, že bude možné za behu meniť hodnoty pre vlastnosti, bude zrejme možné tiež meniť adresy priradené udalostiam a tak im priraďovať publikované metódy pôvodne definované pre iné udalosti rovnakého typu. Príklad: Pre okno je definovaná publikovaná metóda v prípade dvojkliku myšou a pre tlačítko v tomto okne je definovaná publikovaná metóda pre prípad kliknutia myšou. Potom by malo byť možné po vhodnej zmene adresy pre udalosť `OnClick` tlačítka ako reakciu zavolať publikovanú metódu pôvodne definovanú pre udalosť `OnDblClick` okna.

Počas behu môže program zobrazovať modálne dialógové okná s hláseniami pre užívateľa a niekoľkými tlačítkami. Pokiaľ tieto dialógové okná vytvára program za behu po zavolaní procedúry `MessageBox` objektu `Application` alebo funkcie `MessageDlg`, tieto dialógové okná ani ich tlačítka nebudú dostupné v zozname okien a vládacích prvkov získanom cez navrhovanú knižnicu. Preto bude tiež potrebné vytvoriť mechanizmus, ktorý umožní poslať udalosť kliknutie myšou tlačítku modálneho dialógového okna.

Od navrhovanej knižnice pre prostredie Delphi sa bude požadovať, aby dokázala z aplikácie, do ktorej je zaintegrovaná, získať zoznam okien a ovládacích prvkov v oknách. Pre každé okno aj ovládací prvok bude potrebné získať zoznam definovaných vlastností, ich aktuálne hodnoty a údajový typ. Pokiaľ to bude možné,

knižnica by mala dokázať rekurzívne prehľadávať vlastnosti štruktúrovaného údajového typu. Knižnica musí obsahovať mechanizmus umožňujúci nastavovanie hodnôt vlastností, zmenu publikovanej metódy volanej pri udalosti a umelé vyvolávanie udalostí, ktoré štandardne vyvoláva užívateľ svojím vstupom.

Knižnica bude vykonávať svoju funkciu na základe pokynov z externého programu. Na to je potrebné navrhnúť vhodný mechanizmus pre obojsmernú komunikáciu medzi knižnicou a externým riadiacim programom. Implementovaný komunikačný kanál musí byť schopný prenášať príkazy pre knižnicu z riadiaceho programu a tiež z knižnice odosielať údaje o oknách, ovládacích prvkoch, ich vlastnostiach a pre ne definovaných udalostiach do riadiaceho programu. Riadiaci program by mal poskytovať prostredie podobné nástroju Object Inspector v Delphi.

Súčasťou riadiaceho programu bude interpreter skriptovacieho jazyka a okrem grafickej verzie tohoto programu bude k dispozícii aj verzia pre príkazový riadok schopná spúšťať skripty. V grafickej verzii ovládacieho programu bude zabudovaný jednoduchý editor pre skripty s možnosťou skript spustiť, pozastaviť alebo úplne zastaviť. Užívateľ bude počas behu skriptu dostávať výpisy o priebehu a úspešnosti vykonávaných operácií.

Skriptovací jazyk bude obsahovať mechanizmy na prečítanie a nastavenie hodnoty vlastnosti nejakého okna alebo ovládacieho prvku. Bude umožňovať prácu s premennými, do ktorých si užívateľ bude môcť tieto hodnoty ukladať, vyhodnocovanie aritmetických a logických výrazov, podmienku `if` a cyklus `while`. V neposlednom rade musí podporovať príkazy, ktoré vyvolajú v ovládanom programe požadované udalosti. Pri nasadení v oblasti testovania softvéru je užitočné mať možnosť vytvoriť si screenshot okna, preto bude implementovaný aj príkaz pre túto operáciu. Ďalej bude jazyk obsahovať príkaz `Sleep` pre pozastavenie a príkaz `Exit` pre zastavenie vykonávania skriptu.

2.2 Projekty podobného zamerania

Na internete je možné nájsť viacero projektov, ktoré dokážu za behu programu čítať a meniť hodnoty vlastností okien a ovládacích prvkov. Spravidla sú implementované ako balíčky komponentov pre prostredie Delphi, ktoré sa integrujú do programov vytvorených pomocou tohoto prostredia.

Zaujímavý je napríklad *Greatis Object Inspector* [8], ktorý je implementovaný ako súbor komponentov pre prostredie Delphi. Komponent `TComponentInspector` dokáže za behu programu vykonávať prakticky tie isté funkcie ako nástroj Object Inspector v prostredí Delphi počas návrhu programu. Obsahuje tiež editory pre často používané vlastnosti štruktúrovaných typov a špeciálne obdoby nástroja Object Inspector pre prácu s databázovými komponentmi, objektom `TApplication`, systémovými vlastnosťami a konfiguračnými ini súbormi. Neumožňuje však odosielanie získaných údajov externému programu ani vyvolávanie udalostí.

Kapitola 3

Implementácia

3.1 Získanie zoznamu okien a ovládacích prvkov

Prvým problémom, ktorý bolo potrebné pri vývoji knižnice vyriešiť, bol spôsob, ako z bežiaceho programu získať zoznam jeho okien a ovládacích prvkov uložených v jeho oknách.

Po vyhľadávaní v dokumentácii *Windows SDK* pre Delphi [2], stránkach *Microsoft Developer Network* [3] a stránke *Torry's Delphi Pages* [8] sa ako jedna z možností ukázali funkcie Windows API `EnumWindows` a `EnumChildWindows`. Pri tomto riešení by nebolo potrebné integrovať do programu žiadnu knižnicu, ale stačilo by externému ovládaciemu programu zadať názov aplikácie, podľa neho pomocou funkcie Windows API `FindWindow` alebo `FindWindowEx` získať handle aplikácie a tento údaj odovzdať funkciám `EnumWindows` a `EnumChildWindows`. Problémom tohoto riešenia je, že by nám nevrátilo úplne všetky ovládacie prvky v oknách, ale len tie, ktoré sú v Delphi potomkami triedy `TWinControl`. Z toho dôvodu by sme napríklad nezískali komponenty ako `TLabel` alebo `TTimer`. Navyše tu neexistuje žiadny mechanizmus, ktorý by umožňoval pre získané okná a ovládacie prvky získať zoznam ich vlastností.

Po podrobnejšom štúdiu dokumentácie *Delphi Help* [1] sa naskytol elegantnejší spôsob riešenia uvedeného problému. V Delphi každá aplikácia s grafickým užívateľským prostredím obsahuje inštanciu triedy `TApplication`. Táto trieda

má okrem iných aj vlastnosti `ComponentCount` a `Components`. Prvá z nich vyjadruje počet okien v aplikácii a druhá je pole tých okien. Okná, s ktorými môže užívateľ priamo pracovať sú v poli `Application.Components` na indexoch `1` až `ComponentCount - 1`. Prvky tohoto poľa sú typu `TComponent` a majú tiež vlastnosti `ComponentCount` a `Components`. V tomto prípade vlastnosť `ComponentCount` vyjadruje počet ovládacích prvkov v okne a druhá je poľom týchto ovládacích prvkov. Ovládacie prvky sú v poli `Application.Components[i].Components` na indexoch `0` až `ComponentCount - 1`.

Pre každé okno je možné získať zoznam publikovaných metód. To sú procedúry, ktoré v Delphi definujeme počas návrhu ako reakcie na udalosti, napríklad kliknutie na tlačítko. Tento zoznam je možné prečítať z tabuľky virtuálnych metód definovanej pre triedu každého okna. Jej umiestnenie je na adrese, ktorú získame pripočítaním konštanty `vmtMethodTable` k adrese, na ktorej sa nachádza trieda okna. Konštanta `vmtMethodTable` je definovaná v knižnici `System`. V tejto tabuľke sa nachádzajú záznamy v tvare (dĺžka záznamu, adresa metódy, názov metódy).

3.2 Práca s vlastnosťami ovládacích prvkov

3.2.1 Získavanie zoznamu vlastností

Pri riešení tohoto problému bol veľmi užitočný článok Briana Longa, *Run-Time Type Information In Delphi - Can It Do Anything For You?* [7]. Podľa tohoto článku, Delphi počas prekladu generuje skupinu údajových štruktúr zvanú RTTI - Run-Time Type Information. Informácie o údajových typoch v programe sú bežne dostupné len počas návrhu. Účelom týchto údajových štruktúr je sprístupnenie istých informácií o údajových typoch i počas behu programu. Mechanizmus pre prácu s RTTI poskytuje knižnica `TypInfo` bežne dodávaná s prostredím Delphi, ale ako sa aj v uvedenom článku píše, je slabo zdokumentovaná.

Zoznam vlastností okna alebo ovládacieho prvku je možné získať pomocou funkcie `GetPropList` z knižnice `TypInfo`. Deklarácia tejto funkcie vyzerá v knižnici `TypInfo` nasledovne:

```
function GetPropList(TypeInfo: PTypeInfo; TypeKinds: TTypeKinds;
                     PropList: PPropList): Integer;
```

Prvým parametrom je ukazovateľ na tabuľku RTTI definovanú pre daný typ. Tento ukazovateľ je možné získať pomocou funkcie `TypeInfo`, pričom ako parameter jej zadáme prvok poľa `Application.Components` alebo `Application.Components[i].Components`. Druhou možnosťou pre získanie tohoto ukazovateľa je vlastnosť `ClassInfo`, ktorú majú prvky uvedených polí, pretože sú potomkami triedy `TObject`.

V druhom parametri definujeme, že vlastnosti akého typu nám má funkcia vrátiť. Parameter je výčtového typu `TTypeKinds`, kde podľa [7], prípustné hodnoty sú:

- `tkUnknown`: placeholder, nepoužíva sa
- `tkInteger`: pre ordinálne typy a typy podinterval
- `tkChar`: typy `Char` and `AnsiChar`
- `tkEnumeration`: všetky výčtové typy vrátane `Boolean`, `ByteBool`, `WordBool`, `LongBool` a `Bool`
- `tkFloat`: typy s plávajúcou desatinnou čiarkou okrem `Real`
- `tkString`: staršie reťazcové typy ako `String[12]` a `ShortString`
- `tkSet`: typ množina
- `tkClass`: triedy
- `tkMethod`: metódy objektu
- `tkWChar`: typ `WideChar`

- `tkLString`: dlhé reťazce pozostávajúce z prvkov typu `AnsiChar`
- `tkLWString`: reťazcový typ pridaný pri prípravách podpory pre Unicode
- `tkWString`: náhrada za `LWString` počnúc od Delphi 3
- `tkVariant`: typ `Variant`
- `tkArray`: typy pole
- `tkRecord`: záznamové typy
- `tkInterface`: typy interface
- `tkInt64`: 64-bitové celé čísla
- `tkDynArray`: dynamické polia
- `tkAny`: všetky typy

Tretí parameter je ukazovateľ na zoznam vlastností, získaný pomocou funkcie `GetPropList`. Zoznam vlastností je štruktúra typu `TPropInfo` definovaná podľa [7] nasledovne:

```
TPropInfo = packed record
   PropType: PTypeInfo;
   GetProc: Pointer;
   SetProc: Pointer;
   StoredProc: Pointer;
   Index: Integer;
   Default: Longint;
   NameIndex: Smallint;
   Name: ShortString;
end;
```

V nej je v našom prípade najdôležitejšia položka `PropType` definovaná podľa [7] nasledovne:

```
TTypeInfo = record
   Kind: TTypeKind;
   Name: string;
   {TypeData: TTypeData}
end;
```

Pomocou položky `Kind` vieme zistiť, akého typu spomedzi hore uvedených je daná vlastnosť a podľa toho prečítať alebo nastaviť jej hodnotu. Návratová hodnota funkcie `GetPropList` je počet vlastností, ktoré funkcia našla.

3.2.2 Čítanie hodnôt vlastností

Čítanie hodnoty vlastnosti je možné pomocou funkcie `GetPropValue` deklarovanej v knižnici `TypInfo` nasledovne:

```
function GetPropValue(Instance: TObject; const PropName: string;
                      PreferStrings: Boolean = True): Variant;
```

Prvým parametrom je prvok poľa `Application.Components` alebo `Application.Components[i].Components`.

Druhým parametrom je názov vlastnosti, ktorá je uložená v prvku `Name` záznamu typu `TPropInfo`.

Tretí parameter je možné pri volaní vynechať a v tomto projekte ho ani nevyužívam.

Funkcia vracia pri úspechu hodnotu zadanej vlastnosti, ktorú je možné pomocou funkcie `VarToStr` z knižnice `Variants` skonvertovať na typ `String`.

Pokiaľ je nejaká vlastnosť štruktúrovaného typu ako napríklad `tkClass`, je možné na ňu zavolať funkciu `GetPropList` a tak získať zoznam položiek v štruktúre. Pretože nie každá štruktúrovaná vlastnosť musí mať typ, ktorý je odvodený

z typu TObject, nie je v tomto prípade možné použiť pri volaní GetPropList vlastnosť ClassInfo alebo funkciu TypeInfo. Preto v tomto prípade používam funkciu GetTypeData z knižnice TypInfo, ktorej zadám ako parameter ukazovateľ PropType zo štruktúry PropInfo. Funkcia vráti ukazovateľ na štruktúru typu TTypeData, ktorej položka ClassType má už vlastnosť ClassInfo.

TTypeData je variantný záznam s rôznymi položkami, ktoré sa využívajú v závislosti od hodnoty TTypeInfo.Kind. Podľa [7] nasledovne vyzerá nasledovne:

```
TTypeData = packed record
   case TTypeKind of
      tkUnknown, tkLString, tkWString, tkVariant: ();
      tkInteger, tkChar, tkEnumeration, tkSet, tkWChar (...):
      tkFloat: (FloatType: TFloatType);
      tkString: (MaxLength: Byte);
      tkClass: (...);
      tkMethod: (...);
      tkInterface: (...);
      tkInt64: (...);
end;
```

3.2.3 Nastavovanie hodnôt vlastností

Knižnica TypInfo obsahuje rôzne procedúry pre rôzne druhy údajových typov k nastavovaniu ich hodnôt. Týmito procedúrami sú:

- SetOrdProp
- SetEnumProp
- SetSetProp
- SetObjectProp
- SetStrProp
- SetWideStrProp
- SetFloatProp

- `SetVariantProp`
- `SetMethodProp`
- `SetInt64Prop`
- `SetInterfaceProp`

Parametre sú pre každú procedúru rovnaké. Ako prvý zadáme prvok z poľa `Application.Components` alebo `Application.Components[i].Components`. Druhým parametrom je názov vlastnosti a tretím hodnota, ktorú jej chceme nastaviť.

3.3 Komunikácia medzi knižnicou a ovládacím programom

Pre komunikáciu medzi implementovanou knižnicou *AutoLib* a externým ovládacím programom projekt využíva správy systému Windows a pomenovanú rúru, anglicky named pipe.

Pred začatím práce musí ovládací program zistiť, či v systéme bežia programy s integrovanou knižnicou *AutoLib*. To dosiahne tak, že pomocou funkcie Windows API `PostMessage` s prvým parametrom nastaveným na hodnotu `HWND_BROADCAST` rozošle všetkým oknám na pracovnej ploche správu `MSG_AUTOLIB_QUERY`. Túto správu má v sebe zadefinovanú knižnica *AutoLib* aj ovládací program. Pokiaľ aplikácia so zaintegrovanou knižnicou *AutoLib* dostane túto správu, zavolá obsluhu tejto správy definovanú v *AutoLib*. Na túto správu knižnica odpovedá správou `MSG_AUTOLIB_INFO` definovanou v knižnici *AutoLib* aj ovládacom programe. Knižnica *AutoLib* obsahuje procedúru `AppMessageHandler`, ktorá sa pri inicializácii knižnice nastaví ako implicitná obsluha pre udalosť, keď aplikácia dostane správu. Pokiaľ procedúra zistí, že aplikácia dostala jednu zo správ definovaných v knižnici *AutoLib*, zavolá príslušnú obsluhu. V opačnom prípade zavolá pôvodnú obsluhu správ definovanú v aplikácii, ktorej adresu si knižnica pri inicializácii odpamätá.

Pretože súčasťou projektu je aj program pre spúšťanie skriptov z príkazového riadku, knižnica musí byť schopná komunikovať pomocou správ aj s ovládacím programom bez grafického prostredia a tiež musí vedieť, či mu správu `MSG_AUTOLIB_QUERY` poslal program s grafickým prostredím, alebo program pre príkazový riadok. Programu pre príkazový riadok, ktorý neobsahuje okná ani inštanciu objektu `TApplication` totiž nie je možné poslať správu pomocou funkcií `SendMessage` alebo `PostMessage`.

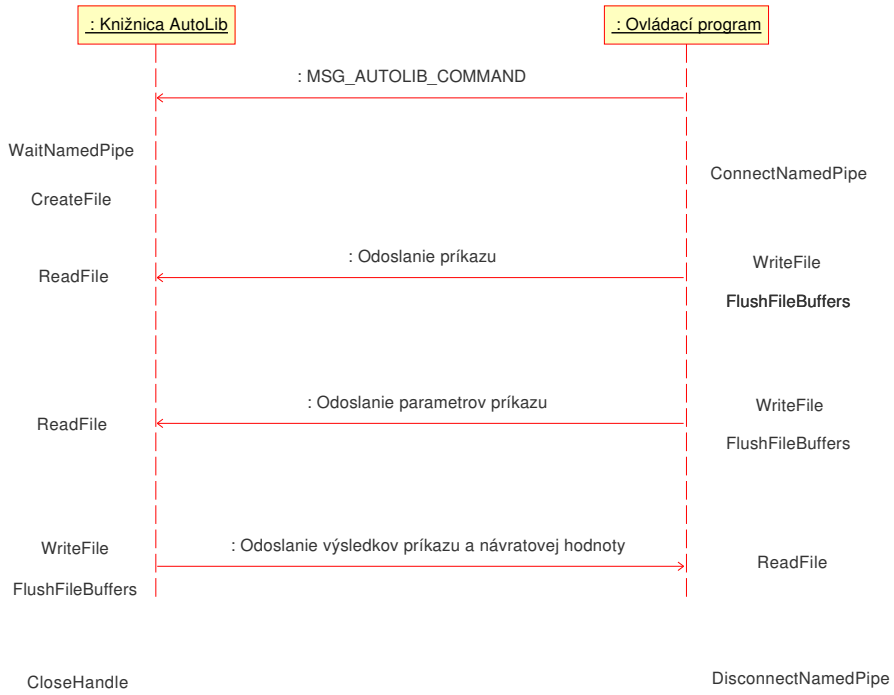
V prípade, že ovládací program má grafické prostredie, prvým parametrom tejto správy je handle hlavného okna ovládacieho programu. Potom knižnica odosiela správu `MSG_AUTOLIB_INFO` pomocou funkcie `PostMessage` hlavnému oknu ovládacieho programu.

Ak je ovládací program určený pre spúšťanie z príkazového riadku, prvý parameter správy `MSG_AUTOLIB_QUERY` je nula a druhý obsahuje ID hlavného vlákna ovládacieho programu. Potom knižnica *AutoLib* odosiela správu `MSG_AUTOLIB_INFO` pomocou funkcie Windows API `PostThreadMessage` hlavnému vláknu ovládacieho programu. Textový ovládací program po odoslaní správy `MSG_AUTOLIB_QUERY` opakovane volá funkciu Windows API `PeekMessage` až kým prijíma správy `MSG_AUTOLIB_INFO` od bežiacich aplikácií s integrovanou *AutoLib*.

Prvým parametrom správy `MSG_AUTOLIB_INFO` je handle aplikácie s integrovanou knižnicou *AutoLib* a druhý parameter je handle hlavného okna tejto aplikácie. Po zvolení programu, ktorý chceme riadiť z ovládacieho programu, handle aplikácie ovládací program používa na posielanie ďalších riadiacich správ. Knižnica si odpamätá podľa typu ovládacieho programu obsah prvého alebo druhého parametra správy `MSG_AUTOLIB_QUERY` pre ďalšiu komunikáciu a tiež si pamätá, či sa jedná o handle okna alebo id vlákna.

Prenos údajov o oknách, ovládacích prvkoch, vlastnostiach a prenos riadiacich príkazov pre knižnicu prebieha cez obojsmernú pomenovanú rúru. Ovládací program ako pipe server po spustení vytvorí pomenovanú rúru `\\.\pipe\AutoPipe` pomocou funkcie Windows API `CreateNamedPipe`. Názov pre rúru je uložený ako konštanta v knižnici *AutoLib* aj ovládacom programe. Ďalšia komunikácia medzi ovládacím programom a knižnicou *AutoLib* prebieha tak, že ovládací program cez rúru odošle požiadavku s potrebnými údajmi, knižnica *AutoLib* si ich prečíta,

vykoná požadované operácie a cez rúru odošle ovládaciemu programu informácie o výsledku operácie plus údaje, ak ovládací program o nejaké požiadal. Komunikáciu cez rúru iniciuje vždy ovládací program. Ak však chce cez rúru odoslať knižnici *AutoLib* nejaké udaje, musí jej dať vedieť o svojom zámere, aby knižnica vedela, kedy sa má k rúre pripojiť a prebrať údaje. To sa rieši pomocou správy `MSG_AUTOLIB_COMMAND`, ktorú vždy pred zápisom do rúry odošle ovládací program aplikácii s integrovanou knižnicou *AutoLib*. Správa nepoužíva žiadne parametre, slúži len na upozornenie knižnice *AutoLib*, že ovládací program jej posiela požiadavku. V prípade, že by sme nepoužívali ovládací program pre príkazový riadok, bolo by možné správu `MSG_AUTOLIB_COMMAND` nahradiť udalosťami pomocou triedy `TEvent` v Delphi. Tieto udalosti sa však dajú používať len v programoch obsahujúcich okná.



Obrázok 1: Schéma komunikácie ovládacieho programu a knižnice *AutoLib* cez rúru

Pri odosielaní správy `MSG_AUTOLIB_COMMAND` je dôležité použiť funkciu `PostMessage` a nie `SendMessage`. Pri `SendMessage` môžu nastať problémy so synchronizáciou. Ovládací program môže zavolať funkciu `ConnectNamedPipe` ešte pred tým, ako knižnica zavolá funkciu `CreateFile`.

3.4 Skriptovací jazyk

Skriptovací jazyk bol vytvorený pomocou balíčku nástrojov *Delphi Yacc & Lex* [6]. Zdrojové kódy jazyka pozostávajú zo zdrojového súboru `lexer.l` pre nástroj Lex a zdrojového súboru `parser.y` pre nástroj Yacc. Pomocou programov `dlex.exe` a `dyacc.exe` dodávaných s balíčkom *Delphi Yacc & Lex*, sa vytvoria pre Delphi knižnice `lexer.pas` a `parser.pas`. Do časti uses v ovládacom programe je potrebné zadať knižnice `parser`, `lexlib`, `yacclib` a `dlib`. Potom je potrebné v ovládacom programe vyvoriť inštancie tried `TLexer` a `TParser`, zavolať procedúru `yyinit` a za tým metódu `parse` triedy `TParser` na skript, aby sa spustilo jeho vykonávanie.

Kapitola 4

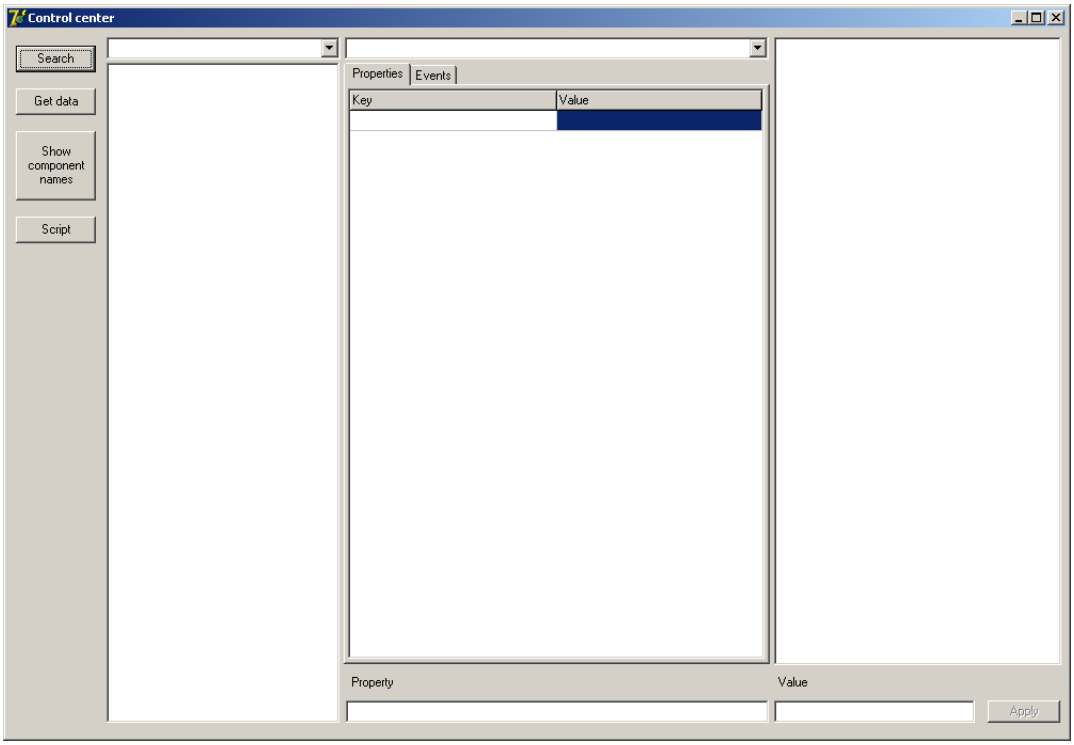
Užívateľská dokumentácia

4.1 Integrácia knižnice AutoLib do projektu v Delphi

Súbor `autolib.pas` skopírujeme do adresára, v ktorom sa nachádza projekt. V prostredí Delphi si v hlavnom menu zvolíme Project/View source. Zobrazí sa nám zdrojový kód hlavného súboru v projekte. Do časti `uses` v tomto súbore pridáme `autolib`, potom projekt preložíme a spustíme. Po tomto úkone je možné vykonávať na projekte všetky úkony podporované knižnicou *AutoLib*.

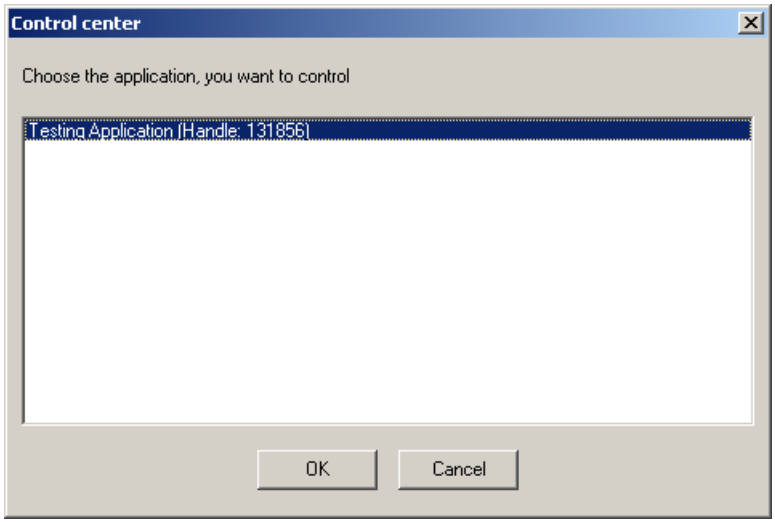
4.2 Sprievodca grafickým prostredím programu Control center

Program *Control center* sa dodáva vo forme jedného súboru s názvom `controller.exe`. Po spustení tohoto programu sa objaví hlavné okno tak, ako to vidieť na obrázku 1:



Obrázok 1: Hlavné okno programu *Control center*

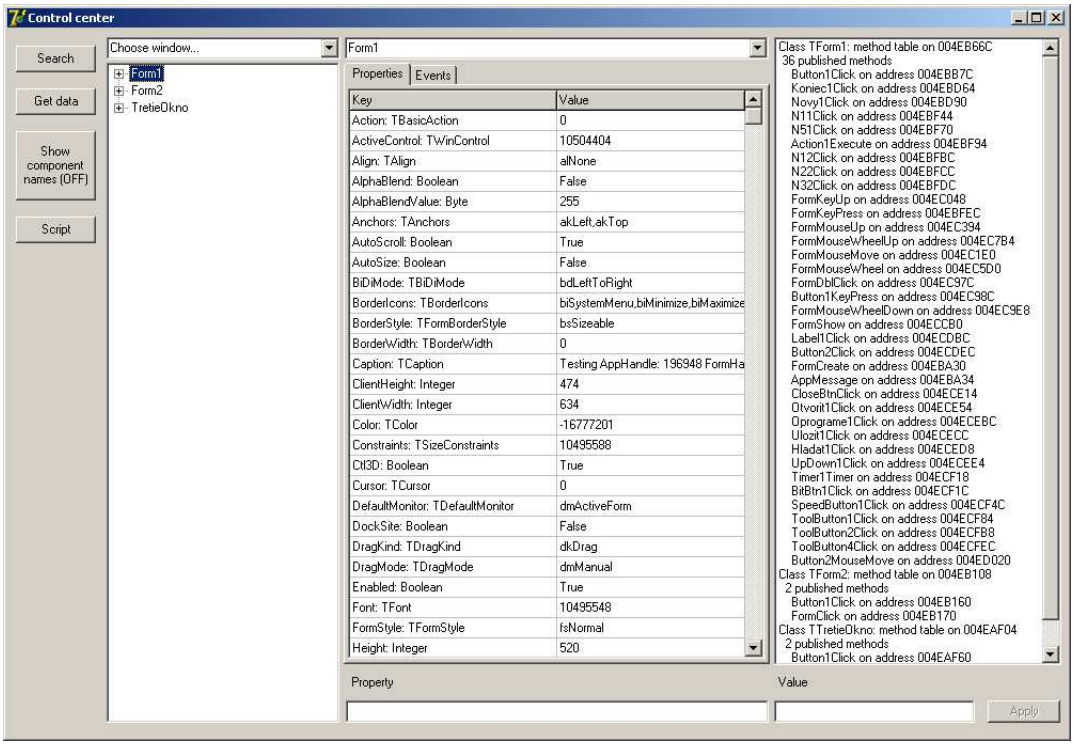
Hlavné okno programu pozostáva z tlačítok pre prácu s hlavnými funkciami programu a ovládacích prvkov, ktoré zobrazujú informácie o aplikácii, ktorú chceme riadiť. Práca s programom spravidla začína kliknutím na tlačítko *Search*. Po ňom sa zobrazí okno, ktorého screenshot vidieť na obrázku 2:



Obrázok 2: Okno pre výber aplikácie, ktorú chceme ovládať

Pokiaľ v danej chvíli bežia nejaké programy, ktoré majú zaintegrovanú knižnicu *AutoLib*, ich zoznam sa zobrazí v tomto okne. Pre každý program sa vypíše nápis

v záhlaví hlavného okna a tiež hodnota handle hlavného okna programu. Zobrazovanie handle môže byť užitočné v prípade, že je spustených viacero inštancií toho istého programu so zaintegrovanou knižnicou *AutoLib*. Po zvolení programu zo zoznamu, kliknití na tlačítko *OK* program *Control center* nadviaže spojenie so zvolenou aplikáciou a nechá si cez knižnicu *AutoLib* odoslať zoznam okien, ovládacích prvkov, ich vlastností a aktuálnych hodnôt týchto vlastností. Získané údaje sa zobrazia v hlavnom okne programu *Control center* podobne, ako to vidieť na obrázku 3:



Obrázok 3: Hlavné okno programu *Control center* po načítaní údajov o oknách, komponentoch a vlastnostiach

V poradí sprava, prvým z hlavných ovládacích prvkov je `TreeView`, ktorý zobrazuje zoznam okien v aplikácii. Každú položku reprezentujúcu okno je možné rozbaliť a zobrazí sa zoznam ovládacích prvkov v danom okne. Nad `TreeView` je umiestnený `ComboBox`, ktorý umožňuje zvoliť konkrétne okno a v `TreeView` sa potom budú zobrazovať len ovládacie prvky rozmietnenené v zvolenom okne.

Vpravo od `TreeView` sú dve záložky, pričom v prvej je možné nastavovať hodnoty pre vlastnosti ovládacích prvkov a v druhej priraďovať udalostiam ob-

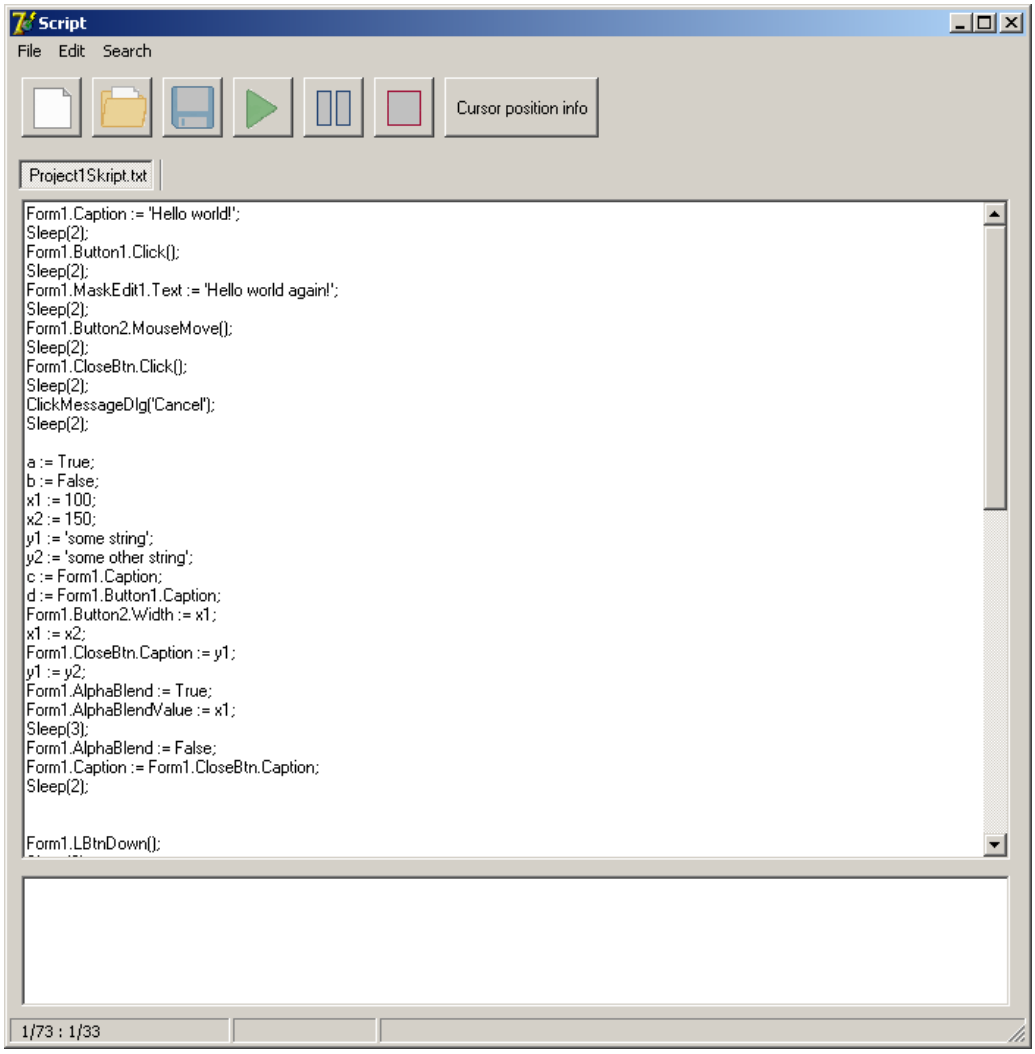
služné publikované metódy. Po kliknutí na nejaké okno alebo ovládací prvok v `TreeView` sa tomto editore zobrazí zoznam vlastností definovaných pre tento objekt, ich typy a hodnoty. Priamo v tomto editore je možné nastavovať hodnoty vlastností podobne, ako počas návrhu programu pomocou nástroja Object inspector v Delphi. Stačí v pravej časti editora vyplniť novú hodnotu a stlačiť na klávesnici *Enter* alebo kliknúť myšou mimo editovaného políčka. Nad editorom sa nachádza `ComboBox`, pomocou ktorého je možné zvoliť ovládací prvok, ktorého vlastnosti budú zobrazené v editore. Editor nepodporuje prácu s vlastnosťami štruktúrovaného typu, ktoré sa delia na viacero položiek.

Vpravo od editora vlastností a udalostí sa nachádza ovládací prvok `ListBox`, v ktorom sa zobrazuje zoznam publikovaných metód rozdelený podľa okien, pre ktoré sú tieto metódy definované. Podľa tohoto zoznamu, vieme zistiť, pre aké udalosti a ktoré ovládacie prvky je definovaná v programe nejaká reakcia.

Pod editorom a zoznamom publikovaných metód sú dve textové políčka, pomocou ktorých je možné nastavovať hodnoty pre vlastnosti. V prvom z nich sa definuje vlastnosť a to vo formáte `Okno.Vlastnosť` alebo `Okno.Komponent.Vlastnosť`. V druhom sa definuje pre túto vlastnosť nová hodnota. Po jej vyplnení a kliknutí na tlačítko `Apply` sa pre definovanú vlastnosť nastaví zadaná hodnota.

Okrem spomínaných ovládacích prvkov sa v pravej časti hlavného okna nachádzajú tlačítka *Get data*, *Show component names* a *Script*. Pomocou tlačítka *Get data* je možné znovu načítať cez knižnicu *AutoLib* z programu zoznam okien, ovládacích prvkov a ich vlastností. Tlačítko *Show component names* môže byť užitočné, keď užívateľ potrebuje poznať identifikátor, alebo inak povedané meno nejakého okna či komponentu. Po spustení programu *Control center* a načítaní údajov cez knižnicu *AutoLib* je toto tlačítko v stave *OFF*. Po kliknutí naň sa zmení stav na *ON* a v tej chvíli sa pre okná a komponenty ovládanej aplikácie začnú zobrazovať ich názvy pomocou vlastnosti `Hint`. Stačí umiestniť kurzor myši na okno, tlačítko, či iný ovládací prvok a zobrazí sa jeho názov, ktorý zodpovedá hodnote vlastnosti `Name` pre daný komponent. Opakovaným kliknutím na tlačítko *Show component names* sa toto zobrazovanie názvov zruší. Po kliknutí na tlačítko

Script sa zobrazí okno, ktoré slúži na písanie a spúšťanie skriptov. Screenshot tohoto okna vidieť na obrázku 4:



Obrázok 4: Skriptovacie okno programu *Control center* s otvoreným skriptom

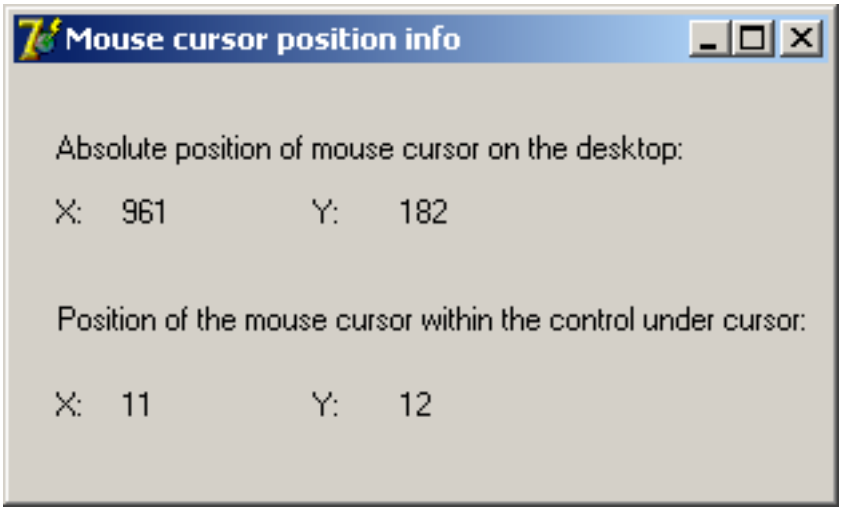
Skriptovacie okno obsahuje hlavné menu v ktorom ponuka *File* slúži na prácu so súbormi skriptov, ponuka *Edit* na prácu s textom a ponuka *Search* na vyhľadávanie a nahradzovanie podreťazcov v skriptoch.

Ďalej sú tu tlačítka na prácu so súbormi a spúšťanie skriptov. Prvé tlačítko slúži na vytvorenie nového prázdneho súboru pre skript, druhé na otvorenie súboru so skriptom a tretie na uloženie súboru so skriptom.

Štvrté tlačítko slúži na spustenie aktuálne zobrazeného skriptu. Piate tlačítko slúži na pozastavenie vykonávania skriptu. Po opätovnom kliknutí na toto tlačítko,

alebo kliknutí na štvrté tlačítko vykonávanie skriptu pokračuje ďalej. Šieste tlačítko slúži na úplné zastavenie vykonávania skriptu.

Posledné siedme tlačítko otvára okno s jednoduchým nastrojom, ktorý zobrazuje polohu myši. Pri simulácii vstupu z myši je niekedy vhodné zadať aj súradnice, na ktoré sa má kurzor myši premiestniť alebo kam chce užívateľ kliknúť. Tento nástroj zobrazuje okrem absolútnej polohy kurzora myši na pracovnej ploche aj polohu kurzora vzhľadom na ovládací prvok, nad ktorým sa práve nachádza. Screenshot tohoto nástroja je vidieť na obrázku 5:



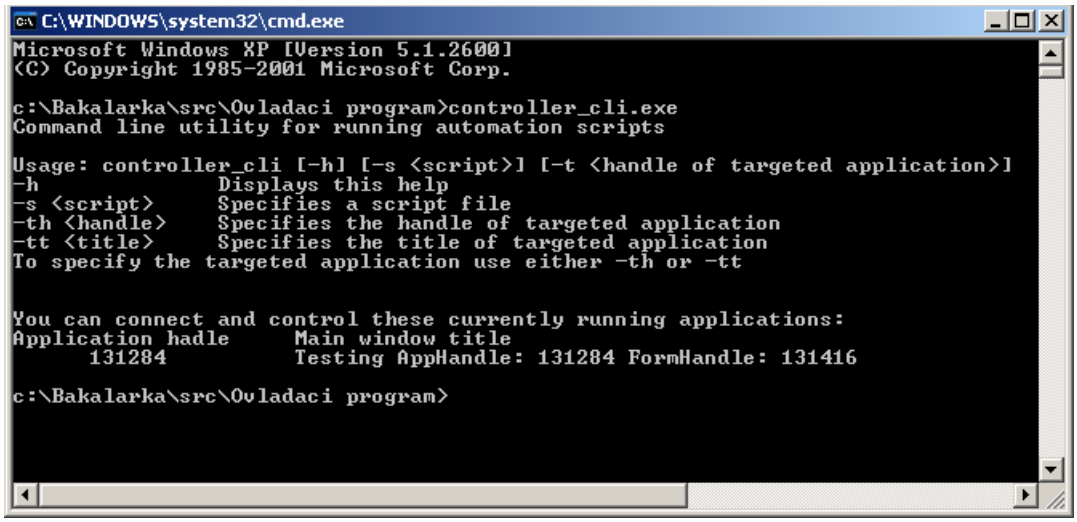
Obrázok 5: Nástroj pre zobrazovanie súradníc, na ktorých je kurzor myši

Pod tlačítkami sa v skriptovacom okne nachádzajú záložky s textovými poľami pre zobrazovanie a úpravu skriptov. Je možné mať naraz otvorených viacero skriptov, pričom každý sa zobrazí v samostatnej záložke. Popis skriptovacieho jazyka je v kapitole 4.4. Pod týmto textovým poľom je ďalšie textové pole, ktoré vypisuje počas behu skriptu informácie o príkazoch, ktoré už prebehli. Zobrazujú sa tu tiež chybové hlásenia.

4.3 Spúšťanie skriptov z príkazového riadku

Program pre spúšťanie skriptov z príkazového riadku sa dodáva vo forme jedného súboru s názvom `controller_cli.exe`. Po spustení tohoto programu bez parametrov alebo s parametrom `-h` sa vypíše stručný návod na použitie pro-

gramu a pod ním zoznam bežiacich aplikácií, ktoré majú v sebe zaintegrovanú knižnicu *AutoLib*. Screenshot z tohoto programu vidieť na obrázku 6:



Obrázok 6: Program pre spúšťanie skriptov z príkazového riadku po spustení bez parametrov

Pri spúšťaní skriptu týmto programom je potrebné špecifikovať skript a aplikáciu, na ktorej chceme operácie popísané skriptom vykonať. Skript sa špecifikuje pomocou parametra `-s`, za ktorým nasleduje cesta k súboru so skriptom. Aplikáciu na ktorej chceme operácie v skripte vykonať, je možné definovať dvomi spôsobmi:

Prvý spôsob je pomocou parametra `-th` a handle aplikácie, ktorého hodnotu zistíme z výpisu po spustení programu bez parametrov.
Príklad: `controller_cli.exe -s scriptsProject1Skript.txt -th 123456`

Druhý spôsob je pomocou parametra `-tt` a názvu aplikácie. Názov aplikácie nie je nutne zhodný s nápisom v záhlaví hlavného okna, ale je to text, ktorý sa zobrazuje v paneli úloh, keď je applikácia spustená.
Príklad: `controller_cli.exe -s scriptsProject1Skript.txt -tt Testing`

Výstupom z programu je záznam o priebehu vykonávania jednotlivých príkazov v skripte a prípadné chybové hlásenia.

4.4 Popis skriptovacieho jazyka

4.4.1 Syntax

Skriptovací jazyk obsahuje premenné, príkazy, podmienku `if`, cyklus `while`, operátor priradenia, aritmetické operátory, logické operátory a identifikátory pre okná, ovládacie prvky, za ktorými ešte môže nasledovať identifikátor udalosti. Každý príkaz alebo výraz okrem konštrukcií `if` a `while` sa nachádza v samostatnom riadku a je zakončený znakom `;` Pri zadávaní príkazov, definícii okien, komponentov alebo vlastností je potrebné dodržiavať písanie veľkých a malých písmen podľa popisu príkazov a údajov zobrazovaných v programe Control center. Je možné pridávať aj komentáre a to tak, že pred komentár užívateľ zapíše `//` a potom text až po koniec riadku je považovaný za komentár.

Okno, ovládací prvok alebo vlastnosť sa definuje nasledovne:

- `Okno`
- `Okno.Vlastnosť`
- `Okno.Komponent`
- `Okno.Komponent.Vlastnosť`

Pokiaľ chceme simulovať nejakú udalosť užívateľského vstupu, definuje sa nasledovne:

- `Okno.Udalosť(parametre)`
- `Okno.Komponent.Udalosť(parametre)`

Okrúhle zátvorky je nutné zapísať i vtedy, keď pre udalosť nešpecifikujeme žiadne parametre.

Premenné

Identifikátor premennej, je textový reťazec začínajúci veľkým alebo malým písmenom anglickej abecedy a obsahuje písmená anglickej abecedy alebo číslice. Premenné nie je potrebné deklarovať ani im zadávať dátový typ. Vytvárajú sa pri prvom priradení hodnoty do nich.

Výrazy

Výrazy sa v skriptovacom jazyku delia na tri typy: číselné, reťazcové a pravdivostné.

Číselný výraz môže obsahovať celočíselné konštanty, premenné, definície vlastností, binárne operátory `+`, `-`, `*`, `/` a okrúhle zátvorky.

Príklad: `a * (b + 5)`

Reťazcový výraz môže obsahovať reťazcové konštanty, premenné, definície vlastností alebo operátor zreťazenia `+`.

Príklad: `a + 'Ahoj'`

Pravdivostný výraz môže obsahovať konštanty `True` a `False`, premenné, definície vlastností, operátory `and`, `or` a `not`, porovnávanie dvoch číselných výrazov pomocou `>`, `<`, `>=`, `<=`, `=`, `<>`, porovnávanie dvoch textových reťazcov pomocou `=`, `<>`, porovnávanie dvoch pravdivostných výrazov pomocou `=`, `<>` a okrúhle zátvorky.

Príklad: `(a > b) and Form1.AlphaBlend`

Príkaz Sleep

Slúži na pozastavenie vykonávania skriptu na počet sekúnd daný parametrom `čas`. Parameter `čas` je celé číslo.

Príklad volania: `Sleep(5);`

Príkaz ClickMessageDlg

Slúži na simuláciu kliknutia myšou na tlačítko modálneho dialógového okna. Dialógové okná, ktoré program v Delphi zobrazuje po zavolaní procedúr `ShowMessage`, `MessageBox` alebo `MessageDlg`, ani ich tlačítka nie sú zaradené medzi komponenty aplikácie podobne ako hlavné okno a ostatné okná v aplikácii. Parameter tlačítko môže byť textový reťazec uzavretý v apostrofoch alebo premenná obsahujúca textový reťazec. Tento textový reťazec obsahuje nápis na tlačítku, na ktoré chceme simulovať kliknutie myšou.

Príklad volania: `ClickMessageDlg('OK');`

Príkaz Exit

Slúži na okamžité zastavenie vykonávania skriptu.

Volanie: `Exit();`

Operátor priradenia

Slúži na ukladanie hodnôt do premenných a tiež nastavovanie hodnôt vlastností okien alebo komponentov. Na pravej strane môže byť premenná alebo definícia vlastnosti, na ľavej strane môže byť výraz číselného, reťazcového alebo pravdivostného typu.

Syntax: `premenná/vlastnosť := výraz;`

Podmienka if, else

Pokiaľ sa pravdivostný výraz za `if` vyhodnotí ako pravdivý, vykonajú sa príkazy v bloku medzi `begin` a `end` pod výrazom `if` podmienka `then`. V opačnom prípade pokiaľ je definovaná aj časť `else`, vykonajú sa príkazy v bloku medzi `begin` a `end` pod kľúčovým slovom `else`. Výraz `if` podmienka `then` musí byť v jednom riadku. V nasledujúcom riadku musí byť kľúčové slovo `begin`, v ďalších riadkoch príkazy a za nimi v novom riadku kľúčové slovo `end`. Zodpovedajúce si `begin` a `end` musia byť rovnako vzdialené od počiatku riadku. Za `else` nasleduje tiež `begin` a `end`. Za posledným `end` nasleduje znak `;`

```
if x > 5 then
begin
   Form1.Caption := 'Hello world!';
end;
if x > 5 then
begin
   Form1.Caption := 'Hello world!';
end
else
begin
   Form1.Caption := 'Ahoj svet!';
end;
```

Cyklus while

Kým sa pravdivostný výraz za `while` vyhodnocuje ako pravdivý, vykonávajú sa príkazy v bloku medzi `begin` a `end` pod výrazom `while` podmienka `do`. Výraz `while` podmienka `do` musí byť v jednom riadku. V nasledujúcom riadku musí byť kľúčové slovo `begin`, v ďalších riadkoch príkazy a za nimi v novom riadku kľúčové slovo `end`. Zodpovedajúce si `begin` a `end` musia byť rovnako vzdialené od počiatku riadku. Za `end` nasleduje znak `;`

Príklad:

```
while x > 5 do
begin
   Form1.Memo1.TypeText('Hello world!');
   x := x - 1;
end;
```

4.4.2 Podporované udalosti a ich volanie

Udalosti podporované knižnicou *AutoLib* a skriptovacím jazykom môžeme rozdeliť do troch hlavných skupín:

- udalosti pre simuláciu vstupu z myši
- udalosti pre simuláciu vstupu z klávesnice
- udalosti pre simuláciu práce s oknom

V prvej skupine majú všetky udalosti okrem dvoch udalostí pre simuláciu kolieska myši rovnaké parametre, pričom každý parameter je celočíselný. Preto ich popíšeme hneď na začiatku. Udalosti okrem spomínaných dvoch môžeme zavolať bez parametrov, s dvomi alebo štyrmi parametrami. V prípade volania s dvomi parametrami prvý vyjadruje x-ovú a druhý y-ovú súradnicu polohy kurzora myši vzhľadom k oknu, alebo ovládaciemu prvku, na ktorom udalosť simulujeme. V prípade volania so štyrmi parametrami, prvé dva parametre sú súradnice. Tretí parameter pokiaľ má hodnotu 1, znamená to, že pri udalosti je stlačené na klávesnici tlačítko Ctrl. Pokiaľ má štvrtý parameter hodnotu 1, znamená to, že pri udalosti je stlačené na klávesnici tlačítko Shift. Pokiaľ vyvolávame uvedené

udalosti bez tretieho a štvrtého, pripadne bez všetkých parametrov, udalosť sa zavolá akoby mali všetky parametre hodnotu 0. Týmito udalosťami sú:

- **MouseMove**: posunutie kurzora myši
- **LBtnDown**: zatlačenie ľavého tlačítka myši
- **LBtnUp**: pustenie ľavého tlačítka myši
- **MBtnDown**: zatlačenie stredného tlačítka myši
- **MBtnUp**: pustenie stredného tlačítka myši
- **RBtnDown**: zatlačenie pravého tlačítka myši
- **RBtnUp**: pustenie pravého tlačítka myši
- **LBtnClick**: kliknutie ľavým tlačítkom myši
- **Click**: synonymum pre udalosť LBtnClick
- **MBtnClick**: kliknutie stredným tlačítkom myši
- **RBtnClick**: kliknutie pravým tlačítkom myši
- **DblClick**: dvojklik ľavým tlačítkom myši

Volanie nasledujúcich dvoch udalostí sa od predchádzajúcich líši tým, že vyžadujú aspoň jeden parameter a to je vzdialenosť o ktorú sa posunulo koliesko myši. Pokiaľ sa volajú s tromi parametrami, tretí a štvrtý parameter zodpovedajú súradniciam polohy kurzora myši. Pokiaľ sa volajú s piatimi parametrami, potom štvrtý a piaty parameter zodpovedajú stavu tlačítok Ctrl a Shift na klávesnici.

- **WheelUp**: otočenie kolieska myši smerom hore
- **WheelDown**: otočenie kolieska myši smerom dole

V druhej skupine sa všetky udalosti volajú s jedným parametrom. U prvých troch to je identifikátor tlačítka klávesnice, ktoré chceme simulovať, uzavretý

do apostrofov a pri udalosti `TypeText` je to textový reťazec, ktorého zadanie cez klávesnicu chceme simulovať, uzavretý do apostrofov. Pokiaľ chceme simulovať stlačenie nejakého klávesu so znakom, potom ako parameter zadávame práve tento znak. Pre ďalšie klávesy použijeme nasledujúce identifikátory:

- **Backspace** pre kláves Backspace
- **Tab** pre kláves Tab
- **Return** alebo **Enter** pre kláves Enter
- **Shift** pre kláves Shift
- **Ctrl** pre kláves Ctrl
- **Alt** pre kláves Alt
- **CapsLock** pre kláves Caps Lock
- **Esc** pre kláves Esc
- **Space** pre kláves Space
- **PageUp** pre kláves Page Up
- **PageDown** pre kláves Page Down
- **End** pre kláves End
- **Home** pre kláves Home
- **Left** pre šípku vľavo
- **Up** pre šípku hore
- **Right** pre šípku vpravo
- **Down** pre šípku dole
- **Insert** pre kláves Insert
- **Delete** pre kláves Delete
- **Win** pre ľavý kláves Windows

- **Num0** až **Num9** pre čísla numerickej časti klávesnice
- **Multiply** pre kláves * na numerickej časti klávesnice
- **Add** pre kláves + na numerickej časti klávesnice
- **Subtract** pre kláves - na numerickej časti klávesnice
- **Decimal** desatinnú bodku na numerickej časti klávesnice
- **Divide** pre kláves na numerickej časti klávesnice
- **F1** až **F12** pre klávesy F1 až F12
- **NumLock** pre kláves Num Lock
- **ScrollLock** pre kláves Scroll Lock

Operácie pre simuláciu vstupu z klávesnice:

- **KeyDown**: simuluje zatlačenie zadaného klávesu na klávesnici
- **KeyUp**: simuluje pustenie zadaného klávesu na klávesnici
- **KeyPress**: simuluje stlačenie zadaného klávesu na klávesnici
- **TypeText**: jediným a povinným parametrom je textový reťazec. Použitie je možné napríklad, keď chceme zapísať do textového poľa nejakú postupnosť znakov.

Operácie pre prácu s oknami:

- **Minimize**: volá sa bez parametrov a vyvolá minimalizáciu daného okna.
- **Maximize**: volá sa bez parametrov a vyvolá maximalizáciu daného okna.
- **Restore**: volá sa bez parametrov a vyvolá obnovenie veľkosti daného okna do pôvodného stavu.
- **Resize**: volá sa s dvomi parametrami a nastavuje veľkosť daného okna. Prvým parametrom je šírka okna a druhým jeho výška.

- **Move**: volá sa s dvomi parametrami a nastavuje polohu daného okna. Prvým parametrom je x-ová súradnica ľavého horného rohu okna a druhým y-ová súradnica ľavého horného rohu okna.
- **Close**: volá sa bez parametrov a vyvolá zatvorenie daného okna.
- **Screenshot**: volá sa s jedným parametrom a ukladá do súboru screenshot klientskej časti daného okna vo formáte jpeg. Parameter obsahuje cestu k súboru, do ktorého sa obrázok má uložiť.

Kapitola 5

Programátorská dokumentácia

5.1 Knižnica Autolib

Knižnica obsahuje jedinú triedu `TAutoControl`. V triede je definovaná verejná procedúra `AppMessageHandler`, ktorá slúži na obsluhu správ pre knižnicu. Pri inicializácii knižnice sa obsluha správ pre aplikáciu nahradí touto procedúrou a adresa pôvodnej procedúry na obsluhu správ sa uloží do premennej `oldAppMessageHandler`. Vo chvíli keď aplikácia dostane správu, zavolá sa procedúra `AppMessageHandler`, ktorá zistí, či sa nejedná o niektorú zo správ pre obsluhu knižnice. Ak áno, zavolá príslušné procedúry a funkcie, v opačnom prípade zavolá pôvodnú procedúru na obsluhu správ a odovzdá jej správu.

Ako už bolo popísané v oddieli 3.3, knižnica má definované tri správy systému Windows a to:

- `MSG_AUTOLIB_QUERY`
- `MSG_AUTOLIB_INFO`
- `MSG_AUTOLIB_COMMAND`

Po zachytení správy `MSG_AUTOLIB_COMMAND` čaká knižnica na názov operácie, ktorú má vykonať. Tento údaj jej ovládací program posiela cez rúru. O identifikáciu požadovanej operácie a zavolanie príslušnej procedúry sa stará procedúra `GetAndExecuteCommand`. Podporované operácie sú:

CMD_TOGGLEHINTS:

Pokiaľ je premenná `showHints` nastavená na `0`, potom pre každé okno a ovládací prvok, ktorý má vlastnosti `ShowHint` a `Hint` nastaví `ShowHint` na `True` a do vlastnosti `Hint` na začiatok pridá hodnotu vlastnosti `Name` daného komponentu vloženú medzi okrúhle zátvorky. Vďaka tomuto užívateľ posunutím kurzora myši nad príslušný ovládací prvok zistí jeho názov. Poznať názvy jednotlivých komponentov je nutné, pokiaľ chce užívateľ čítať alebo nastavovať ich vlastnosti. Potom premennú `showHints` nastaví na `1`. Pokiaľ má premenná `showHints` hodnotu `1`, nastaví ju na `0` a z vlastnosti `Hint` vymaže názvy komponentov spolu so zátvorkami. Pokiaľ po vymazaní tohoto údaju zostane reťazec `Hint` prázdny, nastaví hodnotu vlastnosti `ShowHint` pre daný komponent na `False`.

CMD_GETWINSCOMPSANDPROPS:

Žiadosť o odoslanie zoznamu okien, ovládacích prvkov a ich vlastností ovládaciemu programu. Údaje sa odosielajú v poradí:

```
Počet okien ako celé číslo
i-krát, kde i je počet okien:
   Názov okna ako reťazec znakov
   Počet ovládacích prvkov v okne ako celé číslo
   j-krát, kde j je počet ovládacích prvkov:
      Názov ovládacieho prvku ako reťazec znakov
      Počet vlastností ovládacieho prvku ako celé číslo
      k-krát, kde k je počet vlastností:
         Druh typu podľa TTypeKind ako reťazec znakov
         Údajový typ vlastnosti ako reťazec znakov
         Názov vlastnosti ako reťazec znakov
         Hodnota vlastnosti ako reťazec znakov
         Pokiaľ sa jedná o výčtový typ, potom sa ako doplňujúca
          informácia odošle v reťazci zoznam prípustných hodnôt
         Pokiaľ sa jedná o udalosť, potom sa ako doplňujúca
          informácia odošle reťazec zložený z údajov
          TMethod.Code, TMethod.Data
```

```
Počet vlastností okna ako celé číslo
Informácie o každej vlastnosti rovnako ako u ovládacích prvkov
Zoznam publikovaných metód pre každé okno vo formáte:
   i published methods
   Pre každú metódu reťazec vo formáte x on address y, kde
    x je názov metódy a y je adresa, na ktorej je uložená
```

CMD_GETPROPVAL:
Získanie hodnoty vlastnosti. Z rúry očakáva číslo, ktoré vyjadruje index okna v poli `Application.Components`, číslo, ktoré vyjadruje index ovládacieho prvku v poli `Application.Components[index okna].Components` a reťazec znakov vyjadrujúci názov vlastnosti. Pokiaľ chceme zistiť hodnotu vlastnosti okna, potom ako index ovládacieho prvku pošleme číslo `-1`. Obslužná procedúra odosiela cez rúru reťazec znakov, v ktorom je uložená hodnota danej vlastnosti a návratovú hodnotu, ktoá je v prípade úspechu `0`, inak `-1`.

CMD_SETPROPVAL:
Nastavenie hodnoty vlastnosti. Z rúry očakáva číslo, ktoré vyjadruje index okna v poli `Application.Components`, číslo, ktoré vyjadruje index ovládacieho prvku v poli `Application.Components[index okna].Components`, reťazec znakov vyjadrujúci názov vlastnosti a podľa typu vlastnosti reťazec alebo číslo vyjadrujúce novú hodnotu vlastnosti. Pokiaľ chceme nastaviť hodnotu vlastnosti okna, potom ako index ovládacieho prvku pošleme číslo `-1`. Obslužná procedúra odosiela cez rúru jediné číslo ako návratovú hodnotu. V prípade úspechu je návratová hodnota `0`, v prípade neúspechu `-1`. Nakoniec knižnica odosiela ovládaciemu programu správu `MSG_AUTOLIB_INFO`, pretože po zmene niektorých vlastností hlavne u okien sa stáva, že sa zmení handle okna.

CMD_CLICKMSGDLGBTN:
Simulácia kliknutia na tlačítko modálneho dialógového okna, pričom cez rúru očakáva reťazec znakov obsahujúci nápis na danom tlačítku. Obslužná procedúra najprv získa handle dialógového okna pomocou funkcie Windows API `FindWindow`,

ktorej ako triedu hľadaného okna zadá ukazovateľ na textový reťazec obsahujúci text TMessageForm. Ak sa hodnota handle nerovná nule, procedúra získa handle tlačítka pomocou funkcie Windows API `FindWindowEx`, ktorá ako parametre dostane handle dialógového okna, ukazovateľ na textový reťazec obsahujúci text TButton a ukazovateľ na textový reťazec obsahujúci nápis na tlačítku, na ktoré chceme kliknúť. Po získaní handle tlačítka je možné poslať mu správu systému Windows `BM_CLICK` pomocou funkcie `SendMessage` alebo `PostMessage`. [4]

CMD_SENDWINMESSAGE:
Odoslanie správy systému Windows oknu alebo ovládaciemu prvku. Z rúry očakáva 5 čísel a to index okna, index ovládacieho prvku (môže byť zase `-1`), id správy, `wParam` správy, `lParam` správy

Nasledujúcich 11 operácií pre simuláciu vstupu z myši očakáva z rúry 6 čísel a to: index okna, index ovládacieho prvku (môže byť zase `-1`), x-ová súradnica polohy kurzora vzhľadom na ľavý horný roh okna alebo ovládacieho prvku, y-ová súradnica polohy kurzora vzhľadom na ľavý horný roh okna alebo ovládacieho prvku, `1` ak simulujeme operáciu pri stlačenom klávese Shift, inak `0`, `1` ak simulujeme operáciu pri stlačenom klávese Ctrl, inak `0`.

- **CMD_MOUSEMOVE**: odoslanie správy `WM_MOUSEMOVE`
- **CMD_LBTNDOWN**: odoslanie správy `WM_LBUTTONDOWN`
- **CMD_LBTNUP**: odoslanie správy `WM_LBUTTONUP`
- **CMD_MBTNDOWN**: odoslanie správy `WM_MBUTTONDOWN`
- **CMD_MBTNUP**: odoslanie správy `WM_MBUTTONUP`
- **CMD_RBTNDOWN**: odoslanie správy `WM_RBUTTONDOWN`
- **CMD_RBTNUP**: odoslanie správy `WM_RBUTTONUP`
- **CMD_LBTNCLICK**: odoslanie správ `WM_LBUTTONDOWN` a `WM_LBUTTONUP`
- **CMD_MBTNCLICK**: odoslanie správ `WM_MBUTTONDOWN` a `WM_MBUTTONUP`

- **CMD_RBTNCLICK**: odoslanie správ `WM_RBUTTONDOWN` a `WM_RBUTTONUP`
- **CMD_DBLCLICK**: odoslanie správy `WM_LBUTTONDBLCLK`

Nasledujúce 2 operácie pre simuláciu vstupu z kolieska myši očakávajú z rúry 7 čísel a to: index okna, index ovládacieho prvku (môže byť zase `-1`), x-ová súradnica polohy kurzora vzhľadom na ľavý horný roh okna alebo ovládacieho prvku, y-ová súradnica polohy kurzora vzhľadom na ľavý horný roh okna alebo ovládacieho prvku, počet posunov kolieska, `1` ak simulujeme operáciu pri stlačenom klávese Shift, inak `0`, `1` ak simulujeme operáciu pri stlačenom klávese Ctrl, inak 0.

- **CMD_WHEELDOWN**: odoslanie správy `WM_MOUSEWHEEL`
- **CMD_WHEELUP**: odoslanie správy `WM_MOUSEWHEEL`

Nasledujúce štyri operácie slúžia pre simuáciu vstupu z klávesnice. Z rúry očakávajú tri čísla a to index okna, index ovládacieho prvku (môže byť zase `-1`) a okrem poslednej ešte virtuálny kód klávesu, ktorý chceme simulovať. Operácia `CMD_TYPETEXT` očakáva namiesto kódu klávesu reťazec znakov, ktorých písanie na klávesnici má simulovať.

- **CMD_KEYDOWN**: odoslanie správy `WM_KEYDOWN`
- **CMD_KEYUP**: odoslanie správy `WM_KEYUP`
- **CMD_KEYPRESS**: odoslanie správ `WM_KEYDOWN`, `WM_CHAR` a `WM_KEYUP`
- **CMD_TYPETEXT**: Simulácia písania na klávesnici. Z rúry očakávajú dve čísla a to index okna, index ovládacieho prvku (môže byť zase `-1`) a reťazec znakov. Pre každý znak v reťazci sa odošlú správy `WM_KEYDOWN`, `WM_CHAR` a `WM_KEYUP`.

Nasledujúce operácie slúžia pre simuáciu manipulácie s oknom a vytvorenie screenshotu okna:

- **CMD_RESIZEWIN**: Nastavenie veľkosti okna. Z rúry očakáva 3 čísla a to index okna, šírku a výšku

- **CMD_MOVEWIN**: Nastavenie pozície ľavého horného rohu okna. Z rúry očakáva 3 čísla a to index okna, x-ovú a y-ovú súradnicu nového umiestnenia ľavého horného rohu okna.
- **CMD_MINIMIZEWIN**: Maximalizácia okna. Z rúry očakáva index okna.
- **CMD_MAXIMIZEWIN**: Minimalizácia okna. Z rúry očakáva index okna.
- **CMD_RESTOREWIN**: Obnovenie okna do pôvodnej veľkosti. Z rúry očakáva index okna.
- **CMD_CLOSEWIN**: Zatvorenie okna. Z rúry očakáva index okna.
- **CMD_MAKESCREENSHOT**: Vytvorenie screenshotu okna a uloženie do súboru. Z rúry očakáva index okna a cestu k súboru. Ako návratovú hodnotu posiela 0 v prípade úspechu, -1 ak okno nie je viditeľné a -2 ak zadaný súbor už existuje.

5.2 Ovládací program

Ovládací program musí mať definované správy

- `MSG_AUTOLIB_QUERY`
- `MSG_AUTOLIB_INFO`
- `MSG_AUTOLIB_COMMAND`

Správu `MSG_AUTOLIB_INFO` musú byť schopný prijať cez svoj objekt `Application`, niektoré okno alebo vlákno. Pokiaľ príjíma túto správu cez objekt `Application` alebo okno, potom pri odosielaní správy `MSG_AUTOLIB_QUERY` do parametru `wParam` pridáva handle objektu `Application` alebo handle okna a parameter `lParam` necháva nastavený na 0. V prípade, že správu prijíma cez vlákno, parameter `wParam` je `0` a do `lParam` vloží handle daného vlákna. Rozhranie cez ktoré sa komunikuje s knižnicou *AutoLib* je popísané v kapitole 5.1 a schéma komunikácie medzi ovládacím programom a knižnicou je popísaná v oddieli 3.3. Práca so skriptovacím jazykom je popísaná v oddieli 3.4 a popis skriptovacieho jazyka sa nachádza v nasledujúcom oddieli 5.3.

5.3 Skriptovací jazyk

Lexikálna analýza je naprogramovaná v súbore `lexer.l`. Na začiatku sú definované kľúčové slová a názvy definovaných príkazov. Nasledujú regulárne výrazy pre rozoznávanie definície ackií, vlastností, premenných a čísel reťazcov a nakoniec operátory.

Parser je naprogramovaný v súbore `parser.y`. V prípade, že parser narazí na postupnosť znakov v tvare `identifikátor.identifikátor...`, zavolá na ňu funkciu `WinCompProp` z knižnice `operations.pas`. Táto funkcia prejde zadaný výraz a zistí, ktorý identifikátor definuje aké okno, ovládací prvok, vlastnosť alebo udalosť. V závislosti od toho vráti v premenných `win` a `comp` indexy okna a ovládacieho prvku, potom v premenných `prop` a `act` názov vlastnosti a názov udalosti. Príslušný kód v parseri porovná obsah premennej `act` so známymi názvami udalostí a podľa toho zavolá príslušnú procedúru z knižnice `operations.pas`, aby poslala knižnici *AutoLib* potrebné inštrukcie.

O parsovanie konštrukcií `if` a `while` sa stará funkcia `ParseLines` v knižnici `operations.pas`. V prípade, že parser narazí na riadok obsahujúci `if` podmienka `then` alebo `while` podmienka `do`, vyhodnotí podmienku, nastaví na `True` premennú `found_if` resp. `found_while` a podľa výsledku vyhodnotenia podmienky nastaví hodnotu pre premennú `if_expr` resp. `while_expr`. Funkcia `ParseLines` po odoslaní každého riadku parseru kontroluje obsah týchto premenných a podľa neho skontroluje syntaktickú správnosť nájdenej konštrukcie a parseru pošle príkazy zo zodpovedajúceho bloku `begin end`

Pokiaľ parser narazí na príkaz `Exit`, nastaví hodnotu premennej `found_exit` na `True`. Po tom, čo funkcia `ParseLines` zistí, že táto premenná je nastavená na `True`, ukončí vykonávanie skriptu.

Pre lepšie pochopenie skriptovacieho jazyka odporúčam pozrieť sa do zdrojových súborov pre lexer, parser a tiež do funkcie `ParseLines` v knižnici `operations.pas`.

Kapitola 6

Záver

Projekt umožňuje v ľubovoľnom programe naprogramovanom v prostredí Delphi čítanie a nastavovanie väčšiny vlastností okien a ovládacích prvkov za behu cez externý ovládací program. Umožňuje tiež zmenu obslužnej metódy pre udalosti. Využitie týchto funkcionalít je porovnateľne jednoduché a intuitívne ako práca s nástrojom Object inspector v prostredí Delphi. Neposkytuje síce tak širokú podporu údajových typov ako projekt *Greatis Object Inspector* [5], nepodporuje prácu s vlastnosťami, ktoré majú štruktúrovaný typ, výhodou oproti spomínanému projektu však nesporne je možnosť čítania a nastavovania vlastností cez externý program. Vďaka tomu do cieľovej aplikácie nie je potrebné vkladať žiadne ďalšie ovládacie prvky, cez ktoré by sa práca s vlastnosťami vykonávala a jedinou nutnou úpravou je pridanie knižnice *AutoLib* do časti `uses` v projekte.

Okrem toho knižnica *AutoLib* umožňuje posielanie udalostí oknám aj ovládacím prvkom a tým simulovať vstup z myši alebo klávesnice. Všetká funkcionalita je jednoducho automatizovatelná pomocou priloženého skriptovacieho jazyka. Jazyk sa svojou syntaxou líši len minimálne od objektového Pascalu, preto by jeho využitie nemalo byť pre vývojárov pracujúcich s prostredím Delphi problémom. Skripty je možné pomocou na to určeného programu spúšťať cez príkazový riadok alebo v rámci dávkových súborov. Tým je projekt pripravený pre použitie všade tam, kde sa pracuje na vývoji softvéru pomocou prostredia Delphi a užívatelia chcú ušetriť čas, ktorý by venovali opakovanému vykonávaniu rovnakých operácii, napríklad počas testovania. Rozhranie knižnice umožňuje vývojárom vyrábať

alternatívne ovládacie programy, ktoré by mohli pracovať aj s inými skriptovacími jazykmi ako je ten, ktorý je súčasťou projektu. Vďaka návrhu skriptovacieho jazyka je relatívne jednoduché rozšíriť ho o ďalšiu funkcionalitu.

Existuje niekoľko námetov na rozšírenie projektu. Pokiaľ ide o manipuláciu s vlastnosťami, bolo by užitočné pridať podporu pre ďalšie údajové typy, hlavne tie štruktúrované. Po pridaní tejto funkcionality by boli prínosom tiež editory pre niektoré vlastnosti štruktúrovaného typu, ako napríklad `TControl.Font`, `TTreeView.Items`, `TImage.Picture`, `TMainMenu.Items` a podobne. Zaujímavým rozšírením by bola tiež možnosť nahrávať akési makrá tak, že užívateľ by vykonal ručne nejaké operácie na programe, knižnica `AutoLib` by ich zaznamenala, poslala ovládaciemu programu a ten by postupnosť operácií zaznamenal ako skript. Pomenovaná rúra umožňuje tiež komunikáciu po sieti, preto by bolo v budúcnosti možné upraviť komunikačný protokol, aby sa dalo programy riadiť aj ovládacím programom bežiacim na inom počítači. Ďalšou možnosťou je lokalizácia ovládacieho programu do iných jazykov, ako je angličtina, prípadne umiestnenie všetkých textov tohoto programu do samostatného súboru pre každý jazyk. Odtiaľ by sa načítavali podľa jazykového nastavenia. Nakoniec by bolo vhodné skriptovací jazyk rozšíriť o cykly for, podprogramy a samozrejme podporu ďalších udalostí.

Dodatok A

Obsah priloženého CD

Knižnica *AutoLib* sa nachádza v súbore `autolib.pas`

V adresári `ControlCenter` sa nachádza program *Control center* distribuovaný ako spustiteľný exe súbor s názvom `controller.exe`. Tento adresár ďalej obsahuje program pre spúšťanie skriptov z príkazového riadku distribuovaný ako súbor `controller_cli.exe`. V tomto adresári je podadresár `source`, kde sa nachádzajú zdrojové kódy k programu *Control center* aj programu pre spúšťanie skriptov z príkazového riadku. V adresári `source` sa nachádza podadresár `scriptlang`, kde sú uložené zdrojové súbory pre interpreter skriptovacieho jazyka.

V adresári `Sample` je uložený v podobe spustiteľného exe súboru program *Sample*, ktorý je vytvorený v prostredí Delphi a je možné na ňom skúšať funkcionalitu knižnice *AutoLib* a programu *Control Center*. V tom istom adresári je uložený súbor `samplescript.txt`, ktorý obsahuje ukážkový skript pre dávkové spustenie niektorých operácií v programe *Sample*. Zdrojové kódy programu *Sample* sú v podadresári `source`.

Súbor `bakalarska_praca.pdf` je elektronickou verziou tejto bakalárskej práce.

Literatúra

[1] *Dokumentácia Delphi Help k Borland Delphi 7.*

[2] *Dokumentácia Windows SDK k Borland Delphi 7.*

[3] *Microsoft Developer Network.* `http://msdn.microsoft.com/en-us/library/default.aspx`.

[4] *Invisible man that push a button.* Experts Exchange, `http://www.experts-exchange.com/Programming/Languages/Pascal/Delphi/Q_10246766.html`, 1999.

[5] *Greatis Object Inspector 1.57.* Greatis Software, `http://www.greatis.com/delphicb/objinsp`, 2008.

[6] Albert Graëf et al. *Delphi Yacc & Lex 1.4.* `http://www.grendelproject.nl/dyacclex`, 2005.

[7] Brian Long. *Run-Time Type Information In Delphi - Can It Do Anything For You?* Brian Long's Consultancy & Training Services, `http://www.blong.com/Conferences/BorConUK98/DelphiRTTI/CB140.htm`, 2005.

[8] Thomas Stutz. *How can I enumerate windows and child windows?* Torry's Delphi Pages, `http://www.swissdelphicenter.ch/torry/showcode.php?id=410`, 2004.