1. **Analýza struktury XML souboru**

**Hlavička:**

 Funkce:

* Specifikace verze XML

 XML:

* <?xml version="1.0"?>

**Blok flowgorithm:**

 Funkce:

* Definuje verzi Flowgorithmu a obaluje další bloky

 Informace:

* Uvnitř jsou dále zanořeny bloky ‚attributes‘ a ‚function‘

 XML:

* <flowgorithm fileversion="4.0"> [attributes] [function] … </flowgorithm>

**Blok attributes:**

 Funkce:

* Obsahuje atributy týkající se autora, uložení, vytvoření a editace

 Informace:

* Jednotlivé atributy mají formát:
	+ <attribute name="name" value=""/>
		- Parametr ‚name‘ představuje název atributu
		- Parametr ‚value‘ představuje hodnotu atributu
* Nacházejí se zde atributy, které se dají upravit
	+ <attribute name="name" value=""/>
	+ <attribute name="about" value=""/>.
* Nachází se zde atribut o autorovy (Windows User)
	+ <attribute name="authors" value="Zdenek"/>
* Nachází se zde atribut o datumu uložení
	+ <attribute name="saved" value="2024-03-05 03:57:09 odp."/>
* Nachází se zde atribut o vytvoření a editaci, který obsahuje jako hodnotu Hash, který je unikátní u každého autora/zařízení
	+ <attribute name="created" value="WmRlbmVrO1pERU5EQTsyMDI0LTAzLTAzOzA4OjE1OjI1IGRvcC47MjUzOA=="/>
	+ <attribute name="edited" value="WmRlbmVrO1pERU5EQTsyMDI0LTAzLTA1OzAzOjU3OjA5IG9kcC47MTg7MjcwNw=="/>

 XML:

* <attributes>

<attribute name="name" value=""/>

 <attribute name="authors" value="Zdenek"/>

 <attribute name="about" value=""/>

 <attribute name="saved" value="2024-03-05 03:57:09 odp."/>

<attribute name="created" value="WmRlbmVrO1pERU5EQTsyMDI0LTAzLTAzOzA4OjE1OjI1IGRvcC47MjUzOA=="/>

<attribute name="edited" value="WmRlbmVrO1pERU5EQTsyMDI0LTAzLTA1OzAzOjU3OjA5IG9kcC47MTg7MjcwNw=="/>

 </attributes>

**Blok function:**

 Funkce:

* Představuje definovanou funkci.

 Informace:

* Blok funkce obsahuje další dva bloky a to blok ‚parameters‘ a ‚body‘.
* Jednotlivé funkce mají parametry:
	+ <function name="Main" type="None" variable="">
		- Parametr ‚name‘ představuje název funkce
		- Parametr ‚type‘ představuje návratový datový typ funkce
		- Parametr ‚variable‘ představuje návratovou proměnnou funkce

 XML:

* <function name="Main" type="None" variable=""> [parameters] [body] </function>

**Blok parameters**

 Funkce:

* Zde jsou definovány předávané proměnné funkce.

 Informace:

* Jednotlivé parametry mají formát:
	+ <parameter name="A" type="Integer[]" />
		- Parametr ‚name‘ představuje název proměnné
		- Parametr ‚type‘ představuje datový typ proměnné

 XML:

* <parameters>

 <parameter name="A" type="Integer[]“/>

 <parameter name="B" type="Integer[]"/>

 </parameters>

**Blok body**

 Funkce:

* Představuje vykonávaný kód ve funkci.

 Informace:

* Tento blok obsahuje jednotlivé komponenty, které tvoří kód funkce.

 XML:

* <body>

[code]

 </body>

**Celá struktura XML souboru:**

<?xml version="1.0"?>

<flowgorithm fileversion="4.0">

 <attributes>

 <attribute name="name" value=""/>

 <attribute name="authors" value="Zdenek"/>

 <attribute name="about" value=""/>

 <attribute name="saved" value="2024-03-05 03:57:09 odp."/>

 <attribute name="created" value="WmRlbmVrO1pERU5EQTsyMDI0LTAzLTAzOzA4OjE1OjI1IGRvcC47MjUzOA=="/>

 <attribute name="edited" value="WmRlbmVrO1pERU5EQTsyMDI0LTAzLTA1OzAzOjU3OjA5IG9kcC47MTg7MjcwNw=="/>

 </attributes>

 <function name="Main" type="None" variable="">

 <parameters/>

 <body>

 [code]

 </body>

 </function>

 <function name="sum" type="Integer" variable="C">

 <parameters>

 <parameter name="A" type="Integer" array="False"/>

 <parameter name="B" type="Integer" array="False"/>

 </parameters>

 <body>

 [code]

 </body>

 </function>

</flowgorithm>

1. **Analýza komponent XML souboru**

**Input**

 Funkce komponenty:

* Input čte hodnotu z klávesnice a ukládá ji do proměnné

 Konfigurace komponenty:

* Textbox pro zadání jména proměnné, do které se vstup uloží

 Informace:

* Do proměnná typu String lze ukládat libovolný textový řetězec s písmeny a čísly
* Do proměnné typu Integer lze uložit pouze číselnou hodnotu (lze uložit i číslo s desetinným místem – nejspíš BUG, protože v přehledu proměnných je vidět desetinné číslo v proměnné Integer – správně by mělo brát pouze celé číslo – možná stará verze Flowgorithmu)
* Do proměnné typu Real lze vložit pouze číselnou hodnotu (buď celočíselnou nebo s desetinným místem)
* Do proměnné typu Boolean lze vložit pouze hodnoty „true, True, false, False“
* Lze také vkládat také na předem zvolení index v poli

 XML:

* <input variable="text"/>
	+ Parametr ‚variable‘ představuje název proměnné

**Output**

 Funkce komponenty:

* Output vyhodnotí výraz a výsledek zobrazí na obrazovce

 Konfigurace komponenty:

* Textbox pro zadání výrazu
* Checkbox pro vytvoření nového řádku za výrazem

 Informace:

* Lze vypsat jakoukoliv samostatnou proměnnou
* Nelze vypsat celé pole, ale pouze jenom prvek na konkrétním indexu
* Text je uvozený uvozovkami (viz: “Text“)
* Jednotlivé tokeny jsou rozděleny znakem: &
* Tokenem může být proměnná, text nebo expression

 XML:

* <output expression="&quot;Ahoj&quot;" newline="False"/>
	+ Parametr ‚expression‘ představuje výraz k vypsání
	+ Parametr ‚newline‘ přidává za text znak nové řádky

**Declare**

 Funkce komponenty:

* Declare se používá k vytvoření a pojmenování proměnných a polí.

 Konfigurace komponenty:

* TextBox s názvem proměnné
* RadioButtonGroup a výběrem datového typu
* TextBox pro zadání velikosti pole

 Informace:

* Nelze znovu vytvořit proměnnou se stejným názvem
* Podporované datové typy proměnných:
	+ Integer
	+ Real
	+ Char
	+ String
	+ Boolean
	+ Vlastní definované datové typy
* Podpora více rozměrných polí
* Deklarace pole:
	+ <declare name="int1" type="Integer[]"/>

 XML:

* <declare name="int1" type="Integer"/>
	+ Parametr ‚name‘ představuje název proměnné
	+ Parametr ‚type‘ datový typ

**Assign**

 Funkce komponenty:

* Assign vyhodnotí výraz a výsledek uloží do proměnné

 Konfigurace komponenty:

* TextBox s názvem proměnné
* TextBox s výrazem, který se uloží do proměnné

 Informace:

* Stejné jako u Input komponenty (jedná se v podstatě o to stejné obohacené o výrazy)

 XML:

* <assign variable="bool1" expression="true"/>
	+ Parametr ‚variable‘ obsahuje název proměnné
	+ Parametr ‚expression‘ obsahuje výraz
		- Výraz musí obsahovat i možnost vytvoření pole o určité délce (new Integer[4]
		- Nice to have: [1, 4, 5]

**If**

 Funkce komponenty:

* Vyhodnocení zadaného logického výrazu

 Konfigurace komponenty:

* Textbox pro zadání logického výrazu

 Informace:

* Logické operátory (=, ==, >=, <=, <, >, !=, &&, ||).
* Porovnání čísel funguje s operátory (=, ==, >=, <=, <, >, !=).
* Porovnání textových řetězců funguje i s operátory (==, =, !=).
* Porovnání pravdivostních hodnot funguje pouze s operátory (=, ==, !=).
* Porovnání pravdivostních hodnot podporuje hodnoty ve formátu „true, True, false, False“.
* Pokud zadaná proměnná v podmínce neexistuje projeví se chyba až po spuštění programu.

 XML:

* **If s prázdným tělem**
	+ <if expression=" A = 1">

 <then/>

 <else/>

 </if>

* **If s kódem v True branch**
	+ <if expression=" A = 1">

 <then>

 [code]

 </then>

 <else/>

</if>

* **If s kódem ve False branch**
	+ <if expression=" A = 1">

 <then/>

 <else>

 [code]

 </else>

 </if>

* **If s kódem v True a False branch**
	+ <if expression=" A = 1">

 <then>

 [code]

 </then>

 <else>

 [code]

 </else>

</if>

* + Parametr ‚expression‘ představuje zadaný logický výraz

**Call**

 Funkce komponenty:

* Volání funkce

 Konfigurace komponenty:

* Textbox pro zadání funkce

 Informace:

* Funkce se do Textboxu uvádí např. sum(A, B)
* Pokud zadaná funkce neexistuje projeví se chyba až po spuštění programu.

 XML:

* <call expression="sum(A, B)"/>
	+ Parametr ‚expression‘ představuje zadanou volanou funkci

**While**

 Funkce komponenty:

* While cyklus

 Konfigurace komponenty:

* Textbox pro zadání podmínky

 Informace:

* Podporuje logické operátory (=, ==, != , <, >, <=, >=, &&, ||)
* Při porovnávání řetězců fungují i operátory (==, =, !=).

 XML:

* <while expression="D">

 [code]

 </while>

* + Parametr ‚expression‘ představuje podmínku cyklu

**For**

 Funkce komponenty:

* For cyklus

 Konfigurace komponenty:

* Textbox pro zadání řídící proměnné
* Textbox pro zadání počáteční hodnoty
* Textbox pro zadání konečné hodnoty
* Textbox pro zadání kroku
* Checkbox pro určení vzestupné čítání
* Checkbox pro určení sestupné čítání

 Informace:

* Řídící proměnná musí být předem deklarovaná
* V GUI je napsáno u komponenty, že je nastaveno sestupné čítání. Pro vzestupné čítání informace není uvedena
* Lze čítat pouze s proměnnými typu Integer a Real
* Problém s deklarací se projeví až po spuštění programu
* Když for cyklus čítá od 0 do 5 v znamená to **for(A = 0; A < 5; A++);** To samé platí pro sestupné čítání od 5 do 0 **for(A = 5; A > 0; A--);**

 XML:

* **For s vzestupným čítáním**
	+ <for variable="I" start="0" end="5" direction="inc" step="1">

 [code]

 </for>

* **For se sestupným čítáním**
	+ <for variable="I" start="5" end="0" direction="dec" step="1">

 [code]

</for>

* + - Parametr ‚variable‘ představuje řídící proměnnou
		- Parametr ‚start‘ představuje počáteční hodnotu proměnné
		- Parametr ‚end‘ představuje koncovou hodnotu proměnné
		- Parametr ‚direction‘ představuje podmínku cyklu
		- Parametr ‚step‘ představuje krok cyklu

**Do**

 Funkce komponenty:

* Do While

 Konfigurace komponenty:

* Textbox pro zadání podmínky

 Informace:

* Podporuje logické operátory (=, ==, != , <, >, <=, >=, &&, ||)
* Při porovnávání řetězců fungují i operátory (==, =, !=).

 XML:

* <do expression="A = 1">

 [code]

 </do>

* + Parametr ‚expression‘ představuje podmínku cyklu
1. **Návrh XML souboru**

**Break**

 Funkce komponenty:

* Break ukončí cyklus a začne vykonávat kód za cyklem

 XML:

* <break/>

**Continue**

 Funkce komponenty:

* Continue ukončí kolo cyklu a pokračuje dalším kolem (popřípadě kódem za cyklem, pokud už není další kolo cyklu)

 XML:

* <continue/

**Návrat pole z funkce**

 Funkce komponenty:

* Umožnit vrátit pole jako návratovou hodnotu z funkce

 Informace:

* Podpora více rozměrných polí

 XML návrh:

* <function name="Test" type="Integer[4]" variable="t">

**Vlastní datové typy**

 Funkce komponenty:

* Umožnit uživateli vytvořit vlastní datové typy
* Dokázat je držet v poli, vrátit z funkce

 Informace:

* Na úrovni bloků *functions* a *attributes* vytvořit blok *struct*
* V bloku *struct* další 2 bloky *parameters* a *function*
	+ Blok *struct* by měl parametr *name* s názvem datového typu
* V bloku *parameters* by dále byly bloky s konkrétními proměnnými (stejná struktura jako v declare)
* Přístup k parametrům a funkcím např. Clovek.vek, Clovek.bezi().
* Bude se jednat o referenční typ

 XML:

* <struct name=“Nazev“>

<parameters>

[code]

</parameters>

<function>

[code]

</function>

 </struct>

1. **Návrhový dokument**

Interpreter bude rozdělen celkem do tří balíků: Function, ComplexLogic a DataType. Na úrovni balíků bude soubor Interpreter.cs, který bude obsahovat funkci main.



Interpreter.cs

 *Atributy*

* stack – představuje zásobník pro ukládání návratových hodnot funkcí
* heap – představuje haldu celého programu
* functionStack – představuje zásobník pro uložení instancí jednotlivých funkcí

 *Funkce*

* main – pomocí funkce openFile funkce otevře XML soubor a následně v cyklu volá funkci step nad funkcemi v zásobníku
* openFile – funkce otevře soubor XML soubor a zjistí jeho existenci

**Balík Function**

****

Interface Stepable

 *Funkce*

* step – funkce provede jeden příkaz

Abstract Commands

 *Atributy*

* variables – zásobník pro uložení lokálních proměnných

 *Funkce*

* processCommand – funkce zjistí, o jaký příkaz se jedná a zavolá příslušnou funkci
* declare, assign... – všechny funkce popisující v dokumentu

Function

 *Atributy*

* code – fronta obsahující jednotlivé příkazy
* complexLogic – zásobník pro uložení rozpracovaných komplexních logik

**Balík ComplexLogic**

Interface ComplexLogic

 *Funkce*

* getCommand – funkce vrátí jeden příkaz k provedení z komplexní logiky (for, while, dowhile)

For

 *Atributy*

* code – fronta obsahující jednotlivé příkazy
* Counter – counter počítající kola for cyklu

While

 *Atributy*

* code – fronta obsahující jednotlivé příkazy
* condition – podmínka pro pokračování while cyklu

DoWhile

 *Atributy*

* code – fronta obsahující jednotlivé příkazy
* condition – podmínka pro pokračování dowhile cyklu

**Balík DataType**



StackObj

 *Atributy*

* name – název proměnné pro přístup
* value – pokud se jedná o proměnnou s primitivním datovým typem, bude zde konkrétní datový typ, array a structure bude v tomto případě null
* array, structure – stejný princip jako u value, vždy pouze jedno z těchto tří není null

Array

 *Atributy*

* items – list představující pole, ideální zvolit při imlementaci ArrayList

Structure

 *Atributy*

* atributes – mapa obsahující vždy jméno atributu a jeho datový typ
* functions – mapa obsahující název funkce a její kód