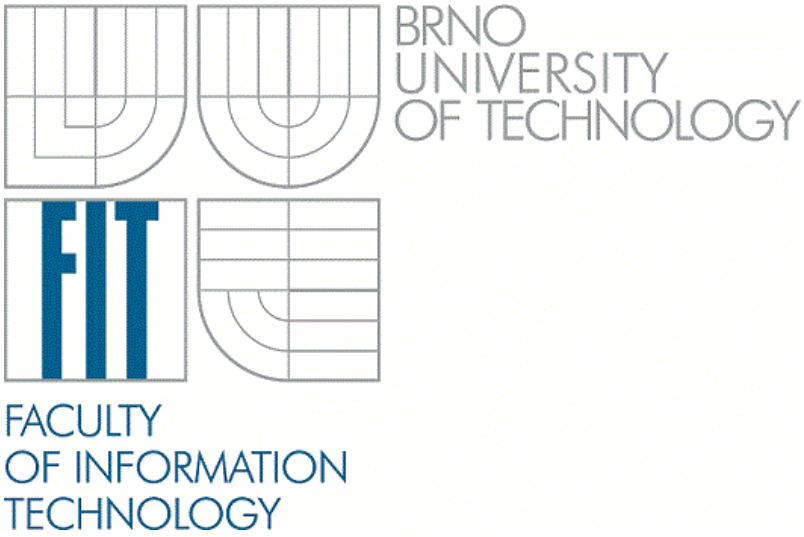

Project to VYPe course

Lexical and syntactical structures in programming language Miranda

Abstract

Vojtěch Beneš, xbenes04
Jakub Horčička, xhorci01

October 2011

This document deals with the content of the presentation which topic is Lexical and syntactical structures in programming language Miranda. Step by step, main characteristics of this language will be described there. They will be supported by suitable examples during the presentation.

1. Basics

The origin of the name Miranda comes from the William Shakespeare's character of the same name that in the play The Tempest utters the words: *"O Brave New World"*. The idea is that the Miranda language is an introduction to the Brave New World of functional programming.

Miranda is a non-strict purely functional programming language with lazy evaluation (call-by-need). It was designed by David Turner in 1985 as a fast interpreter in C for Unix-like operation systems. It is a predecessor of functional programming language Haskell.

The program (script) in Miranda is a set of equations (the order is irrelevant) that define various mathematical functions and abstract data types.

2. Data types and structures

There are three basic data types in Miranda: `char`, `num`, and `bool`. Types can be polymorphic which is indicated by using the symbols * ** *** etc. The user may introduce new data types, for example binary tree, enumerations, or unions. The definition of a new type is introduced by "::==" and consists of its name and one or more constructors. In addition, Miranda permits the definition of abstract data types.

The most commonly used data structure in Miranda is *list*, which is written with square brackets and commas. The list is a homogeneous data type (all elements have the same type). There are several operators used with lists: append "++", subtraction "--", length "#", infix ":" that prefixes element to the front of the list, and infix "!" for subscripting. Miranda permits nested lists and thanks to lazy evaluation infinite lists too. A rather general definition of the list is provided by list comprehension that has similar notation as it is used in the set theory.

On the other hand, *tuples* are sequences of elements with potentially mixed types. They are written using parentheses instead of square brackets. Tuples cannot be subscripted – elements are extracted by the pattern matching.

3. Functions, currying, and high order functions

Functions in Miranda can be passed as parameters and returned as results (high order function). Their application is left associative. Functions with two or more parameters may be partially parameterized or curried. *Currying*[1] provides transforming function with multiple parameters in such way that it can be called as a chain of functions each with a single argument. Miranda has also a quite large library of standard functions.

4. Guarded equations, block structure, and modules

An equation can have several alternative right hand sides distinguished by "guards". *Guards* are written on the right following a comma. The last guard indicating the default case can be written as "otherwise".

[1] http://freaknet.org/martin/libri/Miranda/Overview.html#Curry

Miranda also permits a local definition on the right hand side which is introduced by “where” clause. Miranda allows to organize programs with a nested block structure using *where* clauses. Since used by the parser, the indentation of inner blocks is compulsory.

Miranda provides a basic mechanism to separate the compilation and linking. A script can contain one or more directives `%include`, `%export`, or `%free`.