МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Полоцкий государственный университет имени Евфросинии Полоцкой»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №1**

по дисциплине: «Функциональное программирование»

Вариант 3

|  |  |
| --- | --- |
| Выполнил | Студент гр. 23-ИТ-1  Страпко В.Ю. |
|  |  |
| Проверил | Старший преподаватель кафедры ТП  Забелендик О.Н. |

Полоцк 2025

**Цель:** приобрести навыки работы с интерпретатором языка Haskell. Получить представление об основных типах языка Haskell. Научиться определять простейшие функции.

1. **Задание**
2. Приведите пример нетривиального выражения, принадлежащего к следующему типу: ([Integer],[Double],[(Bool,Char)]). Требование нетривиальности в данном случае означает, что встречающиеся в выражениях списки должны содержать больше одного элемента.
3. Определите следующую функцию: Функция sort2, по двум целым возвращающая пару, в которой наименьшее из них стоит на первом месте, а наибольшее – на втором.
4. **Описание проделанной работы**

Тип переменной: exampleTuple имеет тип ([Integer], [Double], [(Bool, Char)]). Это кортеж из трех элементов:

1. Список целых чисел ([Integer]).

2. Список вещественных чисел ([Double]).

3. Список кортежей, где каждый кортеж содержит булево значение и символ ([(Bool, Char)]).

Значение переменной: exampleTuple содержит:

1. Список целых чисел: [1, 2, 3].

2. Список вещественных чисел: [3.14, 2.71, 1.618].

3. Список кортежей: [(True, 'a'), (False, 'b'), (True, 'c')]. Код представлен в листинге 1.

Листинг 1 – нетривиальное выражение типа ([Integer],[Double],[(Bool,Char)])

1. exampleTuple :: ([Integer], [Double], [(Bool, Char)])
2. exampleTuple = ([1, 2, 3], [3.14, 2.71, 1.618], [(True, 'a'), (False, 'b'), (True, 'c')])

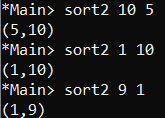
Тип функции: sort2 имеет тип Integer -> Integer -> (Integer, Integer). Это означает, что функция принимает два аргумента типа Integer и возвращает кортеж из двух Integer. Логика функции: функция принимает два целых числа x и y. С помощью условного выражения if проверяется, меньше ли x чем y. Если x < y, то возвращается кортеж (x, y) (то есть числа в порядке возрастания). Иначе возвращается кортеж (y, x) (то есть числа в порядке возрастания).Код функций приведен в листинге 2.

Листинг 2 - Функция sort2

1. sort2 :: Integer -> Integer -> (Integer, Integer)
2. sort2 x y = if x < y then (x, y) else (y, x)
3. **Результат выполнения работы**



**Рисунок 1 – первое задание**

****

**Рисунок 2 – второе задание**

**Вывод**

В ходе выполнения лабораторной работы были изучены основы работы с интерпретатором Haskell, типы, которые имеют значения в Haskell, кортежи и их особенности, списки, функции и условные выражения. С помощью полученных знаний были выполнены два задания по созданию нетривиального выражения и реализации функции с несколькими аргументами и условными выражениями.

1. **Контрольные вопросы**
2. В чем отличие команд интерпретатора от выражений языка Haskell?

Команды интерпретатора отличаются от выражений языка Haskell тем, что начинаются с символа двоеточия (:).

1. Основные типы языка Haskell.

Основными типами языка Haskell являются:

* Типы Integer и Int используется для представления целых чисел, причем значения типа Integer не ограничены по длине.
* Типы Float и Double используется для представления вещественных чисел.
* Тип Bool содержит два значения: True и False, и предназначен для представления результата логических выражений.
* Тип Char используется для представления символов.

1. Функции для работы с кортежами.

Для работы с парами в языке Haskell существуют стандартные функции fst и snd (работает только для кортежей, в которых всего 2 значения).

1. Функции для работы со списками.
   * Функция head возвращает первый элемент списка.
   * Функция tail возвращает список без первого элемента.
   * Функция length возвращает длину списка.
2. Допустимые имена переменных и функций.

Имена переменных и функций не могут начинаться с цифры. Нельзя использовать зарезервированные слова Haskell, такие как let, in, if, then, else, case, of, data, type, class, instance, и другие. Имена переменных и функций могут содержать буквы (латинские), цифры (но не начинаться с них), нижние подчёркивания (\_) и специальные символы, такие как !, @, #, $, %, ^, &, \*, -, +, =, ~, . и :.

1. Команды интерпретатора для работы с файлами программ.

* **:load или :l**: Загружает модуль из указанного файла.
* **:reload или :r**: Перезагружает текущий модуль.
* **:cd**: Изменяет текущую рабочую директорию.
* **:quit или :q**: Завершает сессию интерпретатора.
* **:type или :t**: Показывает тип выражения или функции.
* **:info или :i**: Предоставляет информацию о функции, типе или классе типов, включая их определения и экземпляры.
* **:!**: Выполняет команду оболочки (например, для просмотра содержимого файла).

1. Условные выражения в языке Haskell.

if условие then выражение else выражение. В условном выражении языка Haskell должны присутствовать и then-часть и else-часть. Эти выражения должны быть одного типа.

Условие в определении условного оператора представляет собой любое выражение типа Bool. Примером таких выражений могут служить сравнения При сравнении можно использовать следующие операторы:

* <, >, <=, >= – эти операторы имеют такой же смысл, как и в языке Си
* (меньше, больше, меньше или равно, больше или равно),
* == – оператор проверки на равенство,
* /= – оператор проверки на неравенство.

1. Определение функций в языке Haskell.

Определение функции add, принимающей два целых числа и возвращающей их сумму, выглядит следующим образом:

add :: Integer -> Integer -> Integer

add x y = x + y

Тип функции add может выглядеть несколько загадочно. В языке Haskell считается, что операция -> ассоциативна вправо. Таким образом, тип функции add может быть прочитан как Integer -> (Integer -> Integer), т.е. в соответствие с правилом каррирования, результатом применения функции add к одному аргументу будет функция, принимающая один параметр типа Integer.