***Казиахмедов Туфик Багаутдинович***
канд. пед. наук, доцент, заведующий кафедрой
***Мосягина Татьяна Васильевна***
преподаватель
ФГБОУ ВО «Нижневартовский государственный университет»
г. Нижневартовск, ХМАО – Югра

# ФОРМИРОВАНИЕ АЛГОРИТМИЧЕСКОГО СТИЛЯ МЫШЛЕНИЯ БУДУЩИХ БАКАЛАВРОВ ИВТ

***Аннотация:*** *в данной статье рассмотрена концепция формирования алгоритмического мышления будущих бакалавров ИВТ. Выявлена необходимость применения четких подходов формирования алгоритмического мышления студентов и обучения их представлению алгоритмов в разных стилях, таких как функции, предикаты, прямые алгоритмы, компоненты методов классов при разделении языков на алгоритмические, функциональные, логические, объектно-ориентированные.*

***Ключевые слова:*** *стили программирования, парадигмы программирования, предикаты, функции, комплексные задания.*

Вопрос предметной подготовки бакалавров ИВТ в условиях длительных реформ в образовании необходимо тщательно переосмыслить с учетом текущего состояния развития информационных технологий и инструментария разработки [3].

Сегодня мы наблюдаем бурный рост инструментов разработки программного обеспечения, парадигм программирования. Основной проблемой остается кроссплатформенность программ. Развитие Java машины, Framework.Net позволяет в какой-то степени разрешить эту проблему. Но тем не менее, мы очень часто применяем инструменты, предназначенные для разработки под конкретную платформу (Android Studio, MS Visual Studio).

Для какой бы платформы не разрабатывалась программа, прежде всего нужен анализ задачи для формализации. Поэтому алгоритмическое мышление является необходимым атрибутом, которым должны владеть будущие бакалавры ИВТ. Отдельное изучение стилей программирования и соответственно сред программирования (функциональных, логических, алгоритмических) не всегда дает понять студентам, что не зависимо от инструментария, алгоритмы одни и те же, но стиль их представления отличается. В систему подготовки бакалавров мы ввели такой курс как «Методы разработки и оценки алгоритмов». Этот курс изучается после изучения таких дисциплин как «Программирование», «Дискретная математики», «Структуры и алгоритмы обработки данных», «Web программирование». Приведем некоторые лабораторные работы по данному курсу. Все лабораторные выполняются 3 стилями: логический, функциональный, объектно-ориентированный. В скобках указаны среды или языки разработки

*Лабораторная №1. Алгоритмы. Функции. Предикаты*

1. (Пролог). Создайте предикат, проверяющий являются ли два человека
- сестрами;
- братьями;
- дедушкой и внуком (внучкой);
- дядей и племянником (племянницей);
- супругами;
- родственниками.

2. (C++). Создайте программу, проверяющую являются ли два человека
- сестрами;
- братьями;
- дедушкой и внуком (внучкой);
- дядей и племянником (племянницей);
- супругами;
- родственниками.

3. (Питон, Лисп). Создайте функции, определяющие являются ли два человека

– сестрами;
– братьями;
– дедушкой и внуком (внучкой);
– дядей и племянником (племянницей);
– супругами;
– родственниками.

*Лабораторная №2. Рекурсии*

1. (ПРОЛОГ). Создайте предикаты, вычисляющий

– неотрицательную степень целого числа;
– по натуральному числу N сумму чисел от 1 до N.

2. (C++). Создайте функции, вычисляющие

– неотрицательную степень целого числа;
– по натуральному числу N сумму чисел от 1 до N.

3. (Питон или Лисп)

Определите функции, вычисляющие

– неотрицательную степень целого числа;
– по натуральному числу N сумму чисел от 1 до N.

*Лабораторная №3. Системы счисления*

1. (ПРОЛОГ)Создайте предикат, переводящий число из десятичной системы счисления в двоичную.

2. (C++) Создайте функцию, переводящий число из десятичной системы счисления в р-ичную.

3. (Питон или Лисп). Создайте функцию, переводящий число из десятичной системы счисления в р-ичную.

*Лабораторная №4. Работа с массивами*

(*Пролог*).

– создайте предикат, заменяющий в исходном списке первое вхождение заданного значения другим;

– создайте предикат, заменяющий в исходном списке все вхождения заданного значения другим;

– создайте предикат, порождающий по заданному натуральному числу N список, состоящий из натуральных чисел от 1 до N (по возрастанию).

(C++).

– создайте функцию, заменяющую в исходном массиве первое вхождение заданного значения другим;

– создайте функцию, заменяющую в исходном массиве все вхождения заданного значения другим;

– создайте функцию, порождающую по заданному натуральному числу N массив(динамический), состоящий из натуральных чисел от 1 до N (по возрастанию).

(Лисп или Питон).

– создайте функцию, заменяющую в исходном массиве первое вхождение заданного значения другим;

– создайте функцию, заменяющую в исходном массиве все вхождения заданного значения другим;

– Создайте функцию, порождающую по заданному натуральному числу N массив(динамический), состоящий из натуральных чисел от 1 до N (по возрастанию).

*Лабораторная №5. Простейшие алгоритмы*

(*Пролог*)

– создайте предикат, вычисляющий сумму и произведение элементов списка;

– создайте предикат сортировки списка выборочным методом;

– реализуйте предикат min_list для нахождения минимального элемента массива.

(*C++*)

Создайте класс, в котором определен массив целых чисел и 4 функции:

– функция вычисляющую сумму элементов массива;

– функцию вычисляющую произведение элементов массива;

– функцию сортировки массива выборочным методом;

– функцию для нахождения минимального элемента массива.

(*Лисп или Питон*)

Реализуйте 4 функции:

– функцию вычисляющую сумму элементов массива;

– функцию вычисляющую произведение элементов массива;

– функцию сортировки массива выборочным методом;

– функцию для нахождения минимального элемента массива.

*Лабораторная №6. Деревья*

(*Пролог*)

– создайте предикат, проверяющий, что дерево является двоичным справочником;

– создайте предикат, переписывающий дерево в двоичный справочник;

– создайте предикат, который будет находить среднеарифметическое значение чисел, находящихся в вершинах дерева.

(*C++*)

Создайте класс Bin_tree, реализующий методы бинарного дерева:

– добавление вершины;

– удаление вершины;

– обходы (прямой, обратный, симметричный).

(*Питон или Лисп*)

Создайте функции для добавления, удаления вершин бинарного дерева.

Причем, задания можно выполнить любым инструментом. Главное требование, чтобы использовались все стили. Рассмотрим пример выполнения лабораторной работы. Здесь C++ заменен Java Script.

Создайте функции, определяющие являются ли два человека:

– сестрами;

– братьями;

– дедушкой и внуком (внучкой);

– дядей и племянником (племянницей);

– супругами;

– родственниками.

*Код программы*:

```
<script type="text/javascript">
    // - База и стандартные предикаты ----------------------------
    var mans = ["Иван", "Артём", "Саша", "Степан", "Кирилл"];
    var woms = ["Наташа","Маша","Ира"];
    var mothers = {
                        "Ира" : ["Маша","Наташа","Степан"],
                        "Наташа" : ["Артём","Иван"]
                        };
    var fathers = {
                        "Саша" : ["Иван","Артём"],
                        "Кирилл" : ["Ира"]
                    }
    function man(value) {
        if(mans.includes(value)) return true;
        return false;
    }

    function wom(value) {
        if(woms.includes(value)) return true;
        return false;
    }
```

```
function isSinglePoint(arr1, arr2) {
        for(var i = 0; i<arr1.length; i++) {
                for(var j = 0; j<arr2.length; j++)
                {
                        if(arr1[i] == arr2[j])
                        {
                                return true;
                        }
                }
        }
        return false;
}
function isFather(ft, ch) {
        if(fathers[ft]) {
                var childs = fathers[ft];
                return childs.inlcudes(ch);
        }
        return false;
}
function isMother(mt, ch) {
        if(mothers[mt]) {
                var childs = mothers[mt];
                return childs.inlcudes(ch);
        }
        return false;
}
function getMotherOf(ch) {
        for (var k in mothers){
                if (mothers.hasOwnProperty(k)) {
```

```
                var arr = mothers[k];
                if(arr.includes(ch)) {
                    return k;
                }
            }
        }
        return "";
}
function getFatherOf(ch) {
        for (var k in fathers){
            if (fathers.hasOwnProperty(k)) {
                var arr = fathers[k];
                if(arr.includes(ch)) {
                    return k;
                }
            }
        }
        return "";
}
// - Дополнительные предикаты------------------------------
function isSister(v1,v2) {
        return wom(v1) && wom(v2) && (getMotherOf(v1) == getMotherOf(v2));
}
function isBrother(v1,v2) {
return man(v1) && man(v2) && (getMotherOf(v1) == getMotherOf(v2));
}
function isMarried(v1, v2) {
return man(v1) && wom(v2) && isSinglePoint(fathers[v1],mothers[v2]);
}
```

```
function isRodstv(v1, v2) {
 if(isSister(v1,v2))
        return true;
    else
    {
        if(isBrother(v1,v2))
            return true;
        else {
            var m1 = getMotherOf(v1);
            var m2 = getMotherOf(v2);
            if(m1 != "" && m2!= "") {
                return m1==m2;
            }
        }
    }
return false;
}
function isDyadya(v1, v2) {
z = getMotherOf(v2);
if(isRodstv(v1,z))
    {
    return true;
    }
    return false;
}
function isDedushka(v1, v2) {
    var z = fathers[v1];
    var z2 = mothers[getMotherOf(v2)];
    if(!z)return false;
```

```
        if(!z2)return false;
  if(z.includes(getMotherOf(v2)))
            return true;
            return false;
}
 </script>
```

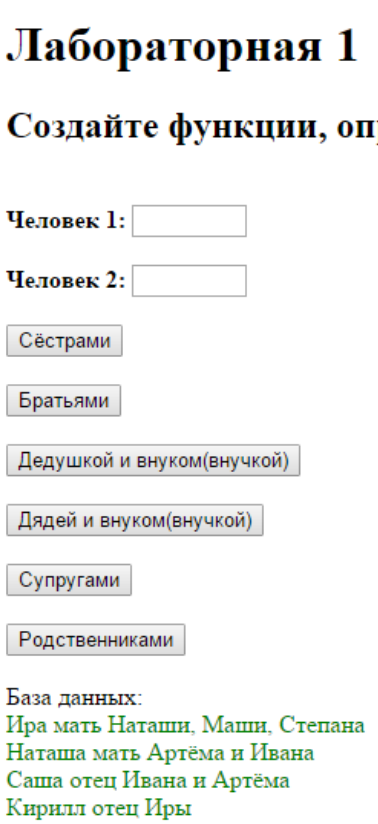


Рис. 1. Результаты выполнения приложения

В этом курсе так же используются алгоритмы для простейших роботов, машинного перевода, алгоритмы распознавания объектов и смысла текстов.

Алгоритмическое мышление – это та основа, на которую опираются дисциплины, связанные с изучением методов организации и управления базами данных, инструментария разработки и тестирования программных комплексов и информационных систем, робототехники и мехатроники, интеллектуального ана-

лиза данных, объектно-ориентрованного анализа и проектирования программных систем и технологий. С программой и содержанием курса «Образовательная робототехника» можно ознакомиться в [1–2].

***Список литературы***

1. Ваграменко Я.А. Методическое обеспечение подготовки учителей образовательной робототехники. Педагогико-технологический аспект / Я.А. Ваграменко, Т.Б. Казиахмедов, Г.Ю. Яламов // Педагогическая информатика. – №1. – 2016. – С. 43–50.

2. Ваграменко Я.А. Методическое обеспечение подготовки учителей образовательной робототехники. Методический аспект / Я.А. Ваграменко, Т.Б. Казиахмедов, Г.Ю. Яламов // Педагогическая информатика. – №2. – 2016. – С. 30–44.

3. Казиахмедов Т.Б. Опережающее обучение в области индустрии информационных технологий в условиях развивающейся экономики и перманентных реформ высшего образования / Т.Б. Казиахмедов // Педагогическая информатика. – №4. – 2014. – С. 62–72.