JAVA. Лабораторная работа №2
Тема: Алгоритмизация и использование управляющих структур в Java

В данной работе рассматриваются программы, использующие операторы ветвления. Ветвление используется для организации различных направлений вычислительного процесса.

Программа 1 - Вычисления числа Пи.

```
public class Pi {
static double pi;
static void leibnic(){
for(double i=1;i<100000000;i+=1){
if (i%2==0){
pi-=1/(2*i-1);
}else{
pi+=1/(2*i-1);}}
pi*=4;
System.out.print("\nЧисло Пи, подсчитанное по методу Лейбница равно: "+pi);}
static void vallis(){
double pi1=1, pi2=1;
for(double i=2;i<100000000;i+=2){
pi1*=i/(i+1);
pi2*=i/(i-1);}
pi=pi1*pi2*2;
System.out.print("\nЧисло Пи, подсчитанное по методу Валлиса равно: "+pi);}
public static void main(String[] args) {
leibnic();
vallis();}}
```

В данной программе описаны два метода нахождения числа Пи: метод Лейбница (ряд Лейбница) и метод Валлиса (формула Валлиса).

В данной программе описываются static переменная класса pi типа double. Расчеты методом Лейбница благодаря использованию имени метода leibnic(). Аналогично вызывается метод Валлиса vallis().

В методе Лейбница для вычисления числа Пи используется ряд:

$$\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}\ldots=\frac{\pi}{4}$$

Для вычисления данного ряда мы использовали цикл с условным оператором, где проверяется условие на четность i%2==0, если оно верно, то дробь имеет

положительный знак pi+=1/(2*i-1), если нет, то ставим знак минус. В конце, для получения приближенного значения числа Пи, переменная pi умножается на 4 и выводится на консоль.

Расчет по формуле Валлиса

$$\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\cdots=\frac{\pi}{2}$$

происходит немного по-другому. В методе объявляются две переменные pi1 и pi2 типа double, первая хранит произведение четных дробей, вторая - нечетных, затем их удвоенное произведение присваивается переменной pi и выводится на консоль.

Программа 2 - Нахождение решения уравнения $A^3+B^3+C^3=\overline{ABC}$

```
public class Sol {
public static void work(){
long x=0;
for(int i=1;i<10;i++){
for (int j=0;j<10;j++){
for (int k=0;k<10;k++){
x=i*100+j*10+k;
if (x==(Math.pow(i,3)+Math.pow(j, 3) + Math.pow(k,3))){
System.out.println(i+"^3+"+j+"^3+"+k+"^3+"+" = "+x);}}}}}
public static void main(String[] args) {
work();}}
```

Данная программа находит такие цифры A, B и C, сумма кубов которых равна числу, составленному из этих цифр.

В программе имеются три цикла, вложенные друг в друга, каждый цикл выполняется десять раз, таким образом тело цикла выполняется одну тысячу раз (10x10x10). Переменная x, составленная из кубов счетчиков циклов, где первый счетчик играет роль сотен числа, второй десятков, третий единиц. Если полученное x равно сумме кубов счетчиков, то на консоль выводится информация о полученных цифрах и числе.

Нахождение кубов чисел осуществляется с помощью метода pow() класса Math, в качестве первого параметра выступает само число, а вторым параметром служит показатель степени Math.pow(i,3). Можно использовать более знакомую и привычную запись i*i*i, j*j*j и k*k*k.

Рассматривается программа, в которой используется оператор множественного выбора, выполняющий задачу множественного ветвления.

Программа 3 - Перевод чисел из десятичной системы счисления в двоичную и шестнадцатеричную.

```
public class Trans {
public static String fromdectobin(int x){
String res="";
short q = 0;
while(x>0){
q=(short)(x%2);
x/=2;
res=q+res;}
return res;}
public static String fromdectohex(float x){
String res="";
short q = 0;
int c = 2;
while(c>0){
q=(short)(16*(x/16-Math.floor(x/16)));
c=(int) Math.floor(x/16);
x=c;
if(q<10){
res=q+res;
}else{
switch(q){
case 10:
res="A"+res;
break;
case 11:
res="B"+res;
break;
case 12:
res="C"+res;
break;
case 13:
res="D"+res;
break;
case 14:
res="E"+res;
break;
```

```
case 15:
res="F"+res;
break;}}}
return res;}
public static void main(String[] args) {
System.out.println(fromdectobin(90367));
System.out.println(fromdectohex(90367));
}}
```

В данной программе приведены два метода, один переводит десятичное число в двоичную систему счисления, второй — в шестнадцатеричную.

Для перевода десятичного числа в двоичное представление используется стандартный метод - деление исходного числа на 2 с сохранением остатка, в нашем примере остаток прибавляется к строке res с левой стороны. В данном методе используется цикл с предусловием while(), который каждый раз получает остаток от деления на 2, затем делит исходное число на 2 и заносит полученный в первом действии остаток слева в строку.

Способ перевода из десятичного в шестнадцатеричное представление использует аналогичный прием, производя последовательно деление на 16, за одним исключением - остаток в нашей программе вычисляется с помощью функции floor(), которая возвращает целую часть дробного числа. В результате получим целочисленный остаток от деления. Процесс записи остатков в строку более сложный, чем в случае двоичной системы счисления, так как в шестнадцатеричной для представления числа недостаточно базового набора цифр от 0 до 9 и необходимо использовать буквы A-F. Для того, чтобы в строку результата добавлять буквы, в программе используется оператор множественного выбора switch(), в котором производится проверка остатка. В зависимости от величины остатка, в строку результата добавляется необходимая буква.

Методы fromdectobin, fromdectohex возвращают значения типа String, которые можно непосредственно использовать в качестве аргумента метода print() для вывода на консоль. В качестве аргументов этих методов берутся числа, которые требуется перевести в заданную систему счисления.

Работа с массивами: поиск минимального элемента и сортировка массива, а также простейшие приемы работы со строками.

Программа 4 - Работа с массивами

```
public class Arrays {
public static short minX(){
short x=0;
short[] array = new short[10];
System.out.print("Исходный массив: ");
for (int i=0;i<10;i++){
array[i]=(short)Math.round(50*Math.random());
System.out.print(array[i]+", ");
}
x=array[0];
for(int i=1;i<10;i++){
if(x>array[i]){
x=array[i]; }}
return x;
}
public static void sort(){
short temp;
short[] array = new short[10];
System.out.print("\nИсходный массив: ");
for (int i=0;i<10;i++){
array[i]=(short)Math.round(50*Math.random());
System.out.print(array[i]+", ");
}
System.out.print("\nМассив отсортированный: ");
for(int i=0;i<9;i++){
for(int j=9;j>i;j--){
if(array[j-1]>array[j]){
temp=array[j];
array[j]=array[j-1];
array[j-1]=temp;
}}
System.out.print(array[i]+", ");
}}
public static void main(String[] args) {
System.out.print("\nМинимальный элемент: "+minX());
sort(); }}
```

В данной программе используются два метода — minX() и sort(). В каждом из представленных методов описывается по одному массиву, выделяя память на десять элементов для каждого массива. Затем происходит заполнение массивов случайными числами с помощью метода Math.random(). Явным преобразованием (short) приводим полученные значения типа double к значениям типа short.

После того, как массивы готовы, можно производить над ними различные действия. Метод minX() ищет минимальный элемент в полученном массиве.

Второй метод производит сортировку полученного массива методом пузырька. Затем выводит значения его элементов.

В методе main() вызываем minX() и sort(). Поскольку метод minX() возвращает значение строкового типа, его можно использовать в качестве аргумента конструкции System.out.print().

Программа 5 - Работа со строками

```
public class Strings {
public static String compare(String s1, String s2){
String s3="";
if (s1.equals(s2)){
s3="Строки \""+s1+"\" и \""+s2+"\" равны";
} else {
s3="Строки \""+s1+"\" и \""+s2+"\" не равны"; }
return s3;
}
public static String add(String s1, String s2){
System.out.print("\nРезультат сложения строк \""+s1+"\" и "+"\""+s2+"\": ");
s1+=" "+s2;
return s1;
}
public static void main(String[] args) {
System.out.println(compare("АБВГ","АБВ"));
System.out.print(compare("АБВ","АБВ"));
System.out.print(add("Hello","World"));
}}
```

В приложении имеется два метода, которые сравнивают и склеивают две строки. Первый метод compare() производит сравнение двух строк и выводит результат на консоль. Сравнение производится с помощью функции equals() в строке. Если строки совпадают, данная функция возвращает значение true.

Во втором методе происходит простое склеивание строк с помощью операции «+».

Задания к работе:

1. Вычислить выражение $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+...+\frac{1}{9999}-\frac{1}{10000}$, используя оператор условия. Ответ 0,69.

2. Для произвольной цифры от 0 до 9 вывести на консоль ее значение прописью. Например, для цифры 9 на консоли должна быть напечатана строка «Девять».

3. Задание обратное первому: при вводе слов, обозначающих цифры от 0 до 9 - отображать их числовое значение. Например, при вводе слова "девять" - отображать в консоле цифру 9. (Информацию о работе со строками вы найдете на следующих страницах).

4. Дан массив из целых чисел A(n), где n=1,25. Необходимо поменять местами его максимальный и минимальный элемент и вывести отдельно максимальный элемент (с помощью описания нового метода, например maxX()).

5. Дан массив из целых чисел B(n), где n=1,25. Необходимо упорядочить массив по возрастанию.

6. Дан массив из целых чисел C(n), где n=1,20. Необходимо найти среднее значение и вывести его на консоль.

7. Дан массив из целых чисел D(n), где n=1,30. Посчитайте сумму четных и нечетных элементов массива.

8. Напишите программу, выводящую на консоль таблицу 3x5 случайных элементов (a(i,j)< 10).

9. Даны 5 строк s1, s2, s3, s4 и s5, на основе условия: если строка s4 равна строке s5, нужно сложить строки s1 и s2, иначе нужно сложить строки s1 и s3.