Федеральное государственное бюджетное образовательное учреждение высшего образования
Казанский государственный энергетический университет
Кафедра информационных технологий и интеллектуальных систем

Вычисления, условия, циклы

Методические указания для выполнения лабораторных работ

Казань, 2024

Лабораторная работа. Вычисления, условия, циклы

Цель работы: приобрести навыки работы с итерационным оператором цикла, научиться программировать задачи с применением вложенных операторов и расчёты с итерационными рядами, генератором случайных чисел.

План лабораторной работы

1.1. Простейшая арифметика

1.2. Условный оператор, оператор выбора

1.3. Циклы foreach, while, for

Краткие теоретические сведения и программы с итерационным оператором в префиксной (`while`) и постфиксной (`do…while`) формах

Операторы итерационного цикла реализуют повторение одного и того же действия при *выполнении заданного условия*. Существует два вида операторов итерационного цикла.

Оператор цикла с предусловием вначале проверяет выполнимость условия и, в зависимости от этого выполняет или обходит операторы тела цикла. Формат цикла с предусловием следующий:

```
while (/*условие продолжения цикла*/)
{
/*один оператор или блок операторов*/;
/*управление условием*/;
}
```

Условие продолжения цикла должно быть истинно `true`, как только условие стало ложным, выполняется выход из цикла. Также как и в условных операторах выбора, фигурные скобки могут опускаться в том случае, если тело цикла — это один оператор. Но как правило в цикле выполняется несколько операторов, так как кроме выполнения полезного действия необходимо делать условие цикла `while` ложным, иначе цикл будет бесконечным, а это, в свою очередь, приведет к зависанию программы.

Пример 1. Автомобиль движется со скоростью 5 км/ч и начинает наращивать скорость с ускорением 10 км/ч^2 до тех пор пока не будет достигнута скорость 60 км/час. Определить, за какое время эта скорость будет достигнута.

```
#include<iostream>
using namespace std;
int main()
{int speed = 5, time = 0, count=0;
    while ( speed < 60 )
    {
        cout << count <<"-speed = " << speed << " time=
"<<time<<endl;
          speed += 10; // приращение скорости
          count++; // подсчёт повторений цикла
          time++;// наращивание времени
    }
     cout<< "final speed "<<speed<<" was reached in "<< time<< "
h"<<endl;
    system("pause");
    return 0;
}
```

В данном примере:

инициализация трёх переменных (скорости `speed`, времени `time` и счётчика цикла `count` реализуется до начала цикла;

условие **`speed < 60`** определяет возможность выполнения цикла, и пока скорость остаётся меньше 60, условие истинно и скорость будет нарщиваться;

управление условием реализуется оператором **`speed += 10;`**

тело цикла составляют операторы вывода на консоль и операторы приращения счётчика и времени.

Оператор цикла с постусловием `do … while` сначала выполняет блок тела цикла, а потом проверяется условие. Если условие истинно, то цикл будет выполнен еще раз, и так до тех пор, пока условие будет истинно. Поскольку условие проверяется после выполнения тела цикла, то блок цикла с постусловием всегда *будет выполнен хотя бы один раз*, независимо от истинности условия. Это может привести к ошибкам, поэтому использовать такой вариант цикла следует только тогда, когда это действительно упрощает алгоритм.

Синтаксис цикла с постусловием имеет следующий вид (обратите внимание на обязательную точку с запятой после условия):

```
do
{
/*один оператор или блок операторов*/;
/*управление условием*/;
```

```
}
while(/*условие повторения цикла*/);
```

Программирование задач с определением суммы бесконечного ряда – один из самых распространённых типов задач с применением оператора итерационного цикла.

Пример 2. Вычислить *сумму бесконечного ряда* с заданной точностью (суммировать до тех пор, пока модуль текущего элемента ряда не станет меньше заданной точности) двумя способами:

1) по общей формуле ряда;

2) по *рекуррентной формуле*.

$\sum_{k=0}^{\infty}(-1)^k\frac{(k+1)!}{(2k+1)!}$; точность $\varepsilon = 10^{-3} = 0{,}001$.

Вопрос о сходимости данного ряда мы не решаем, т.к. условие завершения накопления суммы дано в постановке задачи. Хотя и так очевидно, что данный ряд сходится, поскольку факториал в знаменателе растёт быстрее чем в числителе.

Первый способ. Поскольку функции факториал не имеется в библиотеке транслятора, то факториал решаем стандартным приёмом через накопление произведения.

Входные параметры:

eps – точность вычислений ε

R – значение первого элемента ряда при начальном значении *k* – номере 1-го элемента ряда

Промежуточные параметры

P1 – интегратор для накопления произведения (*k*+1)!

P2 – интегратор для накопления произведения (2**k*+1)!

k – номер итерации, совпадает с номером элемента ряда *R* – текущий (k-ый) элемент ряда.

Выходные параметры

S – сумма элементов ряда

Определение начальных значений переменных

Для переменной k начальное значение дано: $k = 0$ Подставим начальное значение k в формулу ряда $R_k = (-1)^k\frac{(k+1)!}{(2k+1)!}$, получим начальное значение переменной R: $R_0 = (-1)^0\frac{(0+1)!}{(2\cdot 0+1)!} = 1$ Принимаем начальное значение сумматора $S = 0$, начальные значения интеграторов $P1 = 1$, $P2 = 1$.

Контрольный пример: пусть ε=0,1:

$k = 1$; $P1 = 1\cdot 2$; $P2 = 1\cdot 2\cdot 3$; $R = -0.333 > \varepsilon$; $S = 0.667$;

$k = 2$; $P1 = 1\cdot 2\cdot 3$; $P2 = 1\cdot 2\cdot 3\cdot 4\cdot 5$; $R = +0.02 < \varepsilon$; $S = 0.687$; Выход из цикла.

Ответ: S = 0.687

Текст программы (с распечаткой для сверки с контрольным примером):

```
#include<iostream>
```

```
#include<cmath>
using namespace std;
int main()
{int k=0,P1=1,P2=1;
float R=1, s=0, eps;
cout<<"\nInput accuracy eps:"; cin>>eps;
cout<<"accuracy = "<<eps;
do
{s+=R;
k=k+1;
P1*=(k+1);
P2*=(2*k)*(2*k+1);
R=pow(-1.,k)*float(P1)/float(P2);
cout<<'\n'<<k<<' ' <<R<<' '<<P1<<' '<<P2<<endl;
}while(abs(R)>eps);
cout << "Sum = "<<s<< " number of iterations = "<<k<<endl;
system("pause");
return 0;
}
```

Решение по данной программе:

```
Input accuracy eps:0.1
accuracy = 0.1
1 -0.333333 2 6

2 0.05 6 120
Sum = 0.666667 number of iterations = 2
Для продолжения нажмите любую клавишу . .
```

```
Input accuracy eps:0.001
accuracy = 0.001
Sum = 0.711905 number of iterations = 4
Для продолжения нажмите любую клавишу . .
```

Анализ решения показывает, что программа работает верно, т.к. результаты программы совпадаютс контрольным примером. Для проверки полноты программы, получено решение и для другого значения точности.

Анализ программы позволяет выяснить её неэффективность:

1. Возведение в степень -1 неоправданно при большом количестве итераций, т.к. результат всегда будет или +1 или -1.
2. Накопление произведения для вычисления двух факториалов тоже неэффективно, поскольку при большом количестве итераций будет приводить к недопустимо большим числам в числителе и знаменателе.

При вычислении *R* используются операторы приведения типа (от целого к вещественному) для того, чтобы, во-первых, избежать целочисленного деления, и, во-вторых, чтобы не было лишних предупреждений от транслятора.

Второй способ. Для начала *выведем рекуррентную формулу*, т.е. соотношение, которое позволяет получать любой *k*-й член через (*k*-1)-й: $R_k = Q \cdot R_{k-1}$. Вычислим значение коэффициента *Q* для нашего ряда:

$$Q = \frac{R_k}{R_{k-1}} = \frac{(-1)^k}{(-1)^{k-1}} \cdot \frac{(k+1)!}{k!} \cdot \frac{(2k-1)!}{(2k+1)!} = -\frac{k!(k+1)}{k!} \cdot \frac{(2k-1)!}{(2k-1)!\cdot 2k \cdot (2k+1)} =$$
$$= -\frac{k+1}{2k \cdot (2k+1)}.$$

Текст программы для этого варианта и соответствующее решение представлены ниже:

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{int k=0;
float Q, R=1, s=0, eps;
cout<<"\nInput accuracy
eps:"; cin>>eps;
cout<<"accuracy = "<<eps;
do
{s+=R;
k=k+1;
Q=-float(k+1)/float((2*k)*(2*k+1));
R*=Q;
cout<<"\n k="<<k<<"   " <<Q<< "   "<<R<<endl;
}while(abs(R)>eps);
cout << "\nSum = "<<s<< " number of iterations = "<<k<<endl;
system("pause");
return 0;
}
```

```
Input accuracy eps:0.1
accuracy = 0.1
 k=1   -0.333333   -0.333333

 k=2   -0.15   0.05

Sum = 0.666667 number of iterations = 2
Для продолжения нажмите любую клавишу . . .
```

Преимущества этого метода очевидны: нет необходимости в неоправданно больших умножениях, бессмысленном возведении в степень (-1), этот алгоритм требует меньше процессорного времени.

Случайные числа в языке программирования C++ (Табл. 5б Приложения) могут быть сгенерированы функцией `rand()` из стандартной библиотеки C++. Функция `rand()` генерирует числа в диапазоне от 0 до `RAND_MAX`. `RAND_MAX` — это константа, определённая в библиотеке <cstdlib>. Для Microsoft Visual Studio `RAND_MAX` = 32767, но оно может быть и больше, в зависимости от компилятора. Чтобы функция `rand()` всегда возвращала разные числа, её нужно использовать в паре с функцией `srand(unsigned int arg)`. Аргумент `arg` задаёт то стартовое число, на базе которого и создаётся база случайных чисел.

Чаще всего в качестве передаваемой величины в функцию `srand()` используют *системное время в секундах,* т.к. это число будет всегда разным, а соответственно, мы будем получать на выходе из `rand()` разные случайные числа.

Чтобы передать в функцию `srand()` текущее системное время, можно использовать библиотечную функцию `time()`, описанную в библиотеке <ctime>. Для того, чтобы эта функция возвращала текущее время в секундах (секунды отсчитываются от 00:00:00), нужно вызывать ее с параметром `NULL`: `srand(time(NULL));` или `srand(time(0));`. Если нет необходимости в формировании *всегда разных* случайных чисел, то функцию `srand` можно задать с любой константой или вовсе не включать её в программу.

Пример 3. В очередном тираже лотереи было продано $n < 100$ билетов по p рублей каждый. По очереди вынимают 5 номеров. На первый номер выигрыш составил 25% от общей суммы, на второй -15%, на третий – 5%, на четвертый - 3% и на пятый - 2%. Вы купили m билетов. Составить программу, определяющую, выигрыш Ваших билетов.

Словесный алгоритм этой задачи состоит из следующих шагов:

1. Вводим число проданных билетов n, стоимость билет p, число купленных Вами билетов m.
2. Подсчитываем общий доход лотери $St = n \cdot p$.
3. Используя генератор случайных чисел, инициализируем выйгрывшие номера в диапазоне от 1 до 100 и записываем их в массив *Nwin*(5).
4. Организуем проверку купленных билетов:
   4.1 ввести номер *NN* Вашего билета с консоли,
   4.2 если он совпадает с одним из выигрывших билетов, определяем величину приза *Sn*, если не совпадает, то выдать сообщениен, что билет ничего не выиграл.
   4.3 повторять шаги 4.1 – 4.2 до тех пор пока не будут проверены все m билетов.

Текст программы, соответствующий описанной модели, следующий:

```
#include<iostream>
#include<ctime>
#include<iomanip>
using namespace std;
int main()
{unsigned short int n,m,NN;
float p,St,Sn;
cout <<"n:";cin>>n;
cout <<"p:";cin>>p;
St=n*p;
unsigned short int Nwin[5];

srand( time( 0 ) ); // инициализация базы случайных чисел
for(int i=0;i<5;i++)
{Nwin[i] = 1 + rand() % 100; // выйгрышные номера
cout<<' '<<Nwin[i];
```

```
}
cout << "\n How many tickets do you buy? " ;cin>>m;
while (m>0)
{cout <<"Input your number: ";cin >>NN;
// Проверка совпадения Вашего билета с выйгрывшими номерами
for(int i=0;i<n;i++)
   if (Nwin[0]==NN) Sn=St*0.25;
  else if(Nwin[1]==NN) Sn=St*0.15;
      else if(Nwin[2]==NN) Sn=St*0.05;
     else if(Nwin[3]==NN) Sn=St*0.03;
     else if(Nwin[4]==NN) Sn=St*0.02;
     else Sn = 0.;

if (Sn>0) cout <<left<<setw(9)<<"You won"<<Sn<<" rub."<<endl;
else cout <<left<<"You did not win" <<endl;
m--;}
system("pause");
return 0;
}
```

Анализ программы.

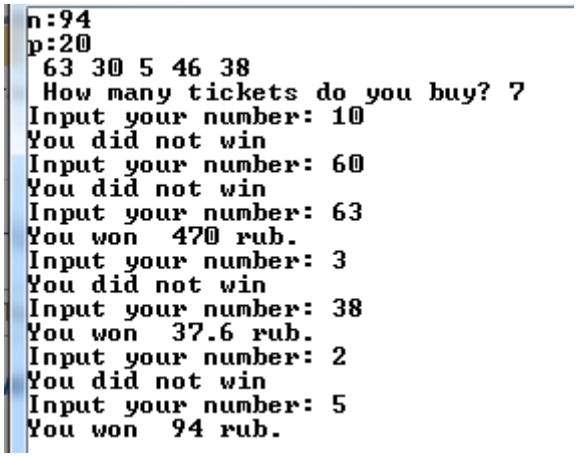
Для того, чтобы использоать функции генератора случайных чисел, в заголов программы подключена библиотека `ctime`. Поскольку параметром `srand` задан `time(0)`, случайные значения всегда будут разными при разных запусках программы.

Для того, чтобы при выводе на консоль использовать манипуляторы формата `setw`, `left` (Табл. 6б Приложения), обеспечивающие вывод текста на консоль с левой границы и отступ между выводимыми значениями.

Для проверки совпадения Вашего билета с выйгрывшими номерами в теле цикла `while` вложен счётный цикл `for`, в теле которого спомощью оператора `if`, реализуется сравнение. Можно, как в *Примере 5* Лабораторной работы №2 данного пособия, задать выигрывшие номера константами и использовать оператор `switch`.

Оператор `while` заканчивает свою работу при условии, что все купленные билеты проверены, т.е. переменная *m*, уменьшающаяся при каждой итерции, станет равна нулю.

Распечатка выигрывших номеров приведена лишь для проверки работоспособности программы, результаты которой следующие:



Данное решение проверяет и полноту программы: есть реакция программы на совпадение/несовпадение номеров с выигрывшими.

Приведённый пример демонстрирует возможность *вложения циклов* друг в друга. В данном случае в теле итерационного цикла `while` используется счётный оператор `for`.

Пример 4. Определить количество цифр в числе *N*, заданным случайным образом.

Есть несколько способов решения этой задачи. Рассмотрим два самых, на наш взгляд, простых решения: цикл с делением и десятичный логарифм и округление.

```
#include<iostream>
#include<cmath>
#include<ctime>
using namespace std;
int main()
{int Number,M,N, count;
srand(10);
cout<< " What is the maximal value of the Number?";
cin>>M;
N=rand()%M;
Number=N;
// Метод - цикл с делением
count = (Number == 0) ? 1 : 0;
        while (Number != 0)
          {
            count++;
            Number /= 10;
        }
cout<<"\nCounts of digits in the number "<<N<<" is equal
"<<count<<endl;

// Метод - десятичный логарифм и округление
// хорош для очень больших чисел.
N=rand()%M;
```

```
Number=N;
count=(Number == 0) ? 1 : (int) ceil(log10(abs(Number) + 0.5));
cout<<"\nCounts of digits in the number "<<N<<" is equal
"<<count<<endl;
system("pause");
return 0;
}
```

Решение задачи демонстрирует: 1) при втором вызовке фуекции `rand()` сформировано другое случайное число, 2) оба метода правильно подсчитывают количество цифр в числе:

```
 What is the maximal value of the Number?10000

Counts of digits in the number 71 is equal 2

Counts of digits in the number 6899 is equal 4
Для продолжения нажмите любую клавишу . . .
```

2. Методика выполнения самостоятельной работы

1. Ознакомиться с условием задачи и примерами решения аналогичных задач из раздела 1 данной лабораторной работы.
2. Для программирования задач с итерационными циклами обязательно составить контрольные примеры для небольших значений параметров, определяющих длительность итерации. При вычислении бесконечных рядов контроль осуществляется самой постановкой задачи, тем не менее хотя бы для одного значения аргумента провести расчёт на калькуляторе. Для задач повышенной сложности обязательно составить словесно-числовую модель и получить контрольные значения.
3. Проверить полноту задачи: рассмотреть все возможные исходы решения в зависимости от исходных данных, предусмотреть случаи возможного зависания, зацикливания программы и запрограммировать корректную реакцию программы на эти ситуации.
4. Записать словесный алгоритм или составить блок-схему алгоритма.
5. Записать код программы на C^{++}.
6. Запустить программу, провести синтаксическую отладку.
7. Проверить работоспособность программы путём сравнения результатов с контрольным примером на все возможные случаи исходных данных.
8. Завершить работу составлением Отчёта, где будут описаны все этапы выполнения самостоятельного задания и приведены распечатки консольного вывода. Образец отчета приведен в Приложении.

3. Задания для самостоятельной работы

Варианты задания 1. Составить программу для решения поставленных задач с применением операторов итерационного цикла.

1. Напечатайте все точные квадраты натуральных чисел, не превосходящие данного числа n. (например, при вводе 50 программа должна вывести 1 4 9 16 25 36 49).
2. Дано натуральное число n. Определите, является ли оно степенью числа 2, и выведите слово YES, если является и слово NO, если не является.
3. Для данного натурального числа n определите такое наименьшее целое k, что $2k \geq n$. Например, при вводе числа 7 программа должна вывести 4.
4. В первый день спортсмен пробежал x километров, а затем он каждый день увеличивал пробег на 10% от предыдущего значения. По данному числу y определите номер дня, на который пробег спортсмена составит не менее y километров. Например, при вводе 10 и 20 программа должна вывести 9. x и y – действительные числа, ответ – целое число.
5. Дано натуральное число n. Напишите программу, вычисляющую сумму цифр числа n.
6. Дано натуральное число n. Составьте программу, определяющую количество нулей среди всех цифр числа n. Выведите результат.
7. Дано натуральное число n. Составьте программу, определяющую наименьшую и наибольшую цифры данного числа. Выведите наименьшую и наибольшую цифры данного числа.
8. Вводится последовательность целых чисел до тех пор, пока не будет введено число 0. После ввода числа 0 программа должна завершить свою работу и вывести сумму введенных чисел.
9. Температура масла в закрытой колбе возрастает с M градусов на t градусов каждую минуту. При достижении значения N градусов возможен разрыв стенок колбы. Определить предельно допустимое время нагрева масла, чтобы не допустить разрыва колбы при любых параметрах M, N, t.
10. По данному числу n найти последнее число Фибоначчи, не превышающее n. Указание: числа Фибоначчи образуют следующую последовательность: 0, 1, 1, 2, 3, 5, 8, …, т.е., начиная с третьего, каждое число равно сумме двух предыдущих.
11. Напишите программу, которая переставляет цифры любого натурального числа в обратном порядке, например, из 179 получится 971.
12. Назовем число палиндромом, если оно не меняется при перестановке его цифр в обратном порядке. Напишите программу, проверяющую по данному числу n, является ли оно палиндромом. Напишите программу, которая по заданному числу K выводит количество натуральных палиндромов, не

превосходящих *K*. Например, при вводе 1 программа выводит 1, а при вводе 100 программа выводит 18. Строковые функции не использовать!

Варианты задания 2. Составить программу для вычисления суммы *бесконечного* ряда. ***Формулой общего члена ряда и возведением (-1) в степень не пользоваться!*** Обязательно использовать рекуррентные соотношения. Для проверки корректности программы сравнить полученное решение со значением соответствующей функции, взятой из библиотеки математических функций `cmath`.

Для проверки работоспособности программы и её полноты обязательно составить контрольный пример для 3-4 членов ряда.

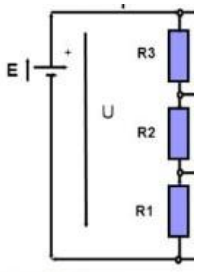
1. $ln\,x = 2\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}} = 2\,(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5} + \cdots),\quad x > 0$
2. $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots = \sum_{n=1}^{\infty}\frac{x^n}{n!},\quad |x| < \infty$
3. $e^{-x} = \sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \cdots,\quad |x| < \infty$
4. $ln(x+1) = \sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} - \cdots,\quad -1 < x \le 1$
5. $ln\frac{1+x}{1-x} = 2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1} = 2\,(x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots),\quad |x| < 1$
6. $arcctgx = \frac{\pi}{2} + \sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} - \cdots,\quad |x| \le 1$
7. $ln(1-x) = -\sum_{n=1}^{\infty}\frac{x^n}{n} = -\,(x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \cdots),\quad -1 \le x < 1$
8. $arctgx = \frac{\pi}{2} + \sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5}\cdots,\quad x > 1$
9. $arctgx = \sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots,\quad |x| \le 1$
10. $arctg\ x = -\frac{\pi}{2} + \sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \cdots,\quad x < -1$
11. $Arth\ x = \sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \cdots,\quad |x| < 1^*$
12. $Arcth\ x = \sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}} = \frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \cdots,\quad |x| > 1^*$

*Гиперболические арктангенс (ареатангенс) и арккотангенс могут быть вычислены в своей области определения через элементарные функции по формуле:

$$Arthx = \frac{1}{2}ln\,(\frac{1+x}{1-x}),\ |x| < 1;\ arcth\,x = ln\frac{\sqrt{x^2-1}}{x-1} = \frac{1}{2}ln\frac{x+1}{x-1};\ |x| > 1$$

Варианты задания 3. Задачи с неявно заданными алгоритмами развивают умения формализовать алгоритм, составить его словесное описание. Для эффективного решения всегда надо составлять контрольный пример. Для решения потребуются умения программировать вложенные циклы и работать с массивами.

1. В фонде денежной игры имеется *M* рублей, которые случайным образом раскидываются по 10 шкатулкам. В последнюю шкатулку закладываются все оставшиеся деньги, если они остались. Если все деньги израсходованы раньше, то оставшиеся шкатулки остаются пустыми, т.е. в них 0 рублей. Игрок имеет право выбирать любые 3 шкатулки, пока он не встретит пустую или пока его выигрыш составляет меньше 50% от исходной суммы *M*. Смоделировать данную игру.
2. В фонде денежной игры имеется *M* рублей, которые случайным образом раскидываются по 10 шкатулкам, при этом объем денег в шкатулке не может быть больше *M/10*. В последнюю шкатулку закладываются все оставшиеся деньги. Игрок об этом не знает. Он выбирает вначале число n, которое случайным образом получает значение от 1 до трёх, затем он имеет право открыть *n* шкатулок и забрать выигрыш. Пока в шкатулках остаются деньги, в игру вступает следующий игрок, и так до тех пор, пока не закончились шкатулки с деньгами. Смоделировать эту игру, определяющую количество игроков и выигрыш каждого из них.
3. В фонде денежной игры имеется *M* рублей, которые случайным образом раскидываются по *10* шкатулкам, при этом объем денег в шкатулке не может быть больше *M/10*. В последнюю шкатулку закладываются все оставшиеся деньги, если они остались. Игрок об этом не знает. Он выбирает любую шкатулку и, если в ней пусто, открывает другую шкатулку, в противном случае, забирает выигрыш и выходит из игры, передавая право игры другому игроку, пока в шкатулках ещё остались деньги Смоделировать эту игру, определяющую количество игроков и выигрыш каждого из них.
4. Дана электрическая схема, состоящая из источника питания с напряжением U = 220в и 10 различных сопротивлений R_i, которые в данную схему могут подключаться последовательно и имеют случайные значения от 10 до 25 ом. Определить, сколько сопротивлений из 10 можно подключить в схему, чтобы ток в цепи был не ниже заданной величины I_0 = 1,5а. Определить номер сопротивления, имеющего максимальное значение.



5. Дана электрическая цепь, состоящая из источника питания с напряжением U и 10 различных сопротивлений R_i, соединённых

последовательно. Определить, сколько сопротивлений из 10 можно последовательно удалить из цепи, чтобы ток в цепи не превысил заданного значения I_0. Определить значение максимального сопротивления, удалённого из цепи.

6. Смоделировать процесс закупки N товаров стоимости от q до r рублей при условии, что на закупку выделена сумма M рублей. Оценить, какое количество из N товаров возможно купить на выделенную сумму. Определить неизрасходованную сумму и стоимость самого дешёвого и дорого из купленных товаров.

*** * ***

4. Контрольные вопросы

1. Чем отличаются циклы с предусловием и постусловием?
2. Возможен ли итерационный цикл без операторов тела цикла? Возможен ли только один оператор в теле цикла?
3. Что такое рекуррентная формула для членов ряда? Как её получить, если известна формула общего члена ряда?
4. В каком диапазоне формируются случайные числа на C++?
5. Как сделать так, чтобы при разных вызовах программы формировались одни и те же | разные случайные числа?

5. Домашнее задание

1. Даны положительные числа a, x, ε. В последовательности y_1, y_2,.., образованной по закону

$$y_0 = a;$$
$$y_i = \frac{1}{2}\left(y_{i-1} + \frac{x}{y_{i-1}} \right);$$

Найти первый член y_n, для которого выполняется условие: $\left| y_n^2 - y_{n-1}^2 \right| < \varepsilon$;

Не использовать массивы.

2. Дано действительное число x. Вычислить:

$$\frac{(x-2)(x-4)(x-8)\cdots(x-64)(x-128)\cdots(x-?)}{(x-1)(x-3)(x-7)\cdots(x-63)(x-127)\cdots(x-??)};$$

Здесь **?** и **??** – числа, удовлетворяющие условию **?** < M, где M >> 1000.

3. Мачеха, чтобы не пустить Золушку на бал, рассыпала в сенях 5 345 453 567 зерен, чтобы та сложила их в сусеки. Тётя Фея посылала ей на помощь всех птиц. Первая партия – воробышков - собрала 3452 зерна за 5 минут. Каждая новая партия птиц - синички, вороны, и пр. - собирала на 5% больше предыдущей. За какое время (дни, часы, минуты) птицы очистят сени и заполнят сусеки рассыпанным зерном?