Лабораторная работа №4
Последовательные одномерные контейнеры

Цели и задачи работы: изучение алгоритмов формирования и обработки одномерных массивов и последовательных контейнеров, программирование и отладка программ формирования и обработки массивов.

Задание к работе:

Написать программу решения задачи в соответствии с индивидуальным вариантом.

Методика выполнения работы:

1. Разработать алгоритм решения задачи по индивидуальному заданию.
2. Написать и отладить программу решения задачи.
3. Протестировать работу программы на различных исходных данных.

При написании лабораторной работы 4 использование контейнеров строго обязательно как для C++, так и для других языков программирования. Все поэлементные циклы с постоянным шагом должны быть Range-based for loop (since C++11) https://en.cppreference.com/w/cpp/language/range-for

Индивидуальные задания
Первое задание

1. Постановка задачи

Пусть дано нелинейное уравнение $f(x) = 0,$ где $f(x)$ – функция, определена и непрерывна на некотором промежутке $[a,b]$. Требуется найти корни уравнения, числа x_1, x_2 и т.д. которые называются нулями функции.

Решения осуществляется в два этапа:

I. Находятся отрезки $[a_i,b_i]$, внутри которых содержится один простой или кратный корень. Этот этап называется процедурой отделения корней.

II. Уточняется до заданной точности одним из численных методов, в которых реализуются последовательные приближения.

Практически все приближенные методы нахождения корней уравнений относятся к классу итерационных методов.

Методом итерации назовем численный метод, который последовательно, шаг за шагом, уточняет первоначальное, грубое значение корня. Каждый шаг в методе называется итерацией. Важным свойством итерационных методов является сходимость метода.

2. Отделения корней

Будем говорить, что корень x^* отделен на $[a,b]$, если других корней на этом отрезке нет. Для отделения корней применяются два способа: графический и аналитический.

Основу аналитического способа составляют следующие теоремы.

Теорема 1. Если функция $f(x)$ определена и непрерывна на отрезке $[a_i, b_i]$, причем на концах отрезка $[a_i, b_i]$ принимает значения разных знаков, т.е. $f(a_i) \cdot f(b_i) < 0$, то на этом отрезке содержится, по крайней мере, один корень $x^* \in [a_i, b_i]$ для которого $f(x) = 0$.

Теорема 2. В условиях теоремы 1, если $f(x)$ непрерывная на $[a_i, b_i]$ и имеет конечную производную и ее первая производная сохраняет знак внутри отрезка $[a_i, b_i]\left(\underset{[a_i,b_i]}{sign f'(x) = const}\right)$, то на $[a_i, b_i]$ находится только один корень уравнения $f(x) = 0$.

Теорема Ролля. Пусть на $[a,b]$ определена функция $f(x)$, причем:

1) $f(x)$ непрерывна на $[a,b]$;

2) для любого x из $[a,b]$ существует конечная производная $f'(x)$;

3) $f(\mathrm{a}) = f(b)$.

Тогда найдется точка $c \in [a,b]$ такая что $f'(c) = 0$.

Теорема Ролля устанавливает, сколько всего различных корней может быть у уравнения.

Графический способ построение графика функции применяется наиболее часто, но не обладает большой точностью.

Часто бывает удобно заменить уравнение $f(x) = 0$ на равносильное $f_1(x) - f_2(x) = 0$, с формированием простых функций $f_1(x)$ и $f_2(x)$ и дальнейшим построением графиков этих функций. Корнями уравнения являются абсциссы точек пересечения графиков $y = f_1(x)$ и $y = f_2(x)$.

3. Метод половинного деления

Пусть дано нелинейное уравнение $f(x) = 0$ и отделен простой корень x^*, т.е. найден такой отрезок $[a_0, b_0]$, что $x^* \in [a_0, b_0]$, и на концах отрезка

функция имеет значения, противоположные по знаку ($f(a_0)*f(b_0)<0$). Отрезок $[a_0,b_0]$ называется начальным интервалом неопределенности, требуется уточнить местоположение корня уравнения с заданной точностью ε.

Процедура уточнения положения корня заключается в построении последовательности вложенных друг в друга отрезков, каждый из которых содержит корень уравнения. Для этого находится середина текущего интервала неопределенности $c_k=(a_k+b_k)/2$, $k=0,1,2,\ldots$ и в качестве следующего интервала неопределенности из двух возможных выбирается тот, на концах которого функция $f(x)$ имеет различные знаки.

Процесс завершается, когда длина текущего интервала неопределенности становится меньше заданной величины ε, задающей точность нахождения корня. В качестве приближенного значения корня берется середина последнего интервала неопределенности.

В основе метода половинного деления лежит теорема о вложенных отрезках. Последовательность отрезков

$$[a_1,b_1] \supset [a_2,b_2] \supset \ldots \supset [a_n,b_n] \supset \ldots$$

называется *вложенной*. При условии, что длины отрезков $|b_n - a_n| \underset{n\to\infty}{\to} 0$, эта последовательность называется стягивающейся.

Теорема Кантора. Для всякой стягивающейся последовательности вложенных отрезков существует единственная точка x, принадлежащая всем отрезкам этой последовательности.*

Замечание!

1. Метод имеет линейную, но безусловную сходимость, его погрешность за каждую итерацию уменьшается в 2 раза:

$$|b_k-a_k|=|b_0-a_0|\cdot 2^{-k},$$

Можно оценить число итераций ***k*** для достижения заданной точности ε:

$$k \geq log_2 \frac{(b_0-a_0)}{\varepsilon},$$

2. К достоинствам метода следует отнести то, что он позволяет найти простой корень для непрерывных функций при любых a_0, b_0, таких что $f(a_0)*f(b_0)<0$. Недостатки метода: он не обобщается на системы нелинейных уравнений и не может быть использован для нахождения корней четной кратности.

Методика решения задачи

1. Найти начальный интервал неопределенности $[a_0,b_0]$ одним из методов отделения корней, задать малое положительное число ε и присвоить k=0.
2. Найти середину текущего интервала неопределенности $c_k=(a_k+b_k)/2$.
3. Если $f(a_k)*f(c_k)<0$, то положить $a_{k+1}=a_k$, $b_{k+1}=c_k$ иначе $a_{k+1}=c_k$, $b_{k+1}=b_k$. В результате находится текущий интервал $[a_{k+1},b_{k+1}]$.
4. Если $|b_{k+1}-a_{k+1}|<=\varepsilon$, то процесс завершить: $x^*=(a_{k+1}+b_{k+1})/2$, иначе $k=k+1$ перейти к п.2.

4. Метод хорд

Этот метод в интервале неопределенности обеспечивает более быстрое нахождение корня, чем метод половинного деления. Для этого отрезок $[a,b]$ делится не пополам, а в отношении $|f(a)|:|f(b)|$.

Геометрически метод хорд эквивалентен замене кривой *y=f(x)* хордой, проходящей через точки *(a, f(a))* и *(b,f(b))* (рисунок 1).

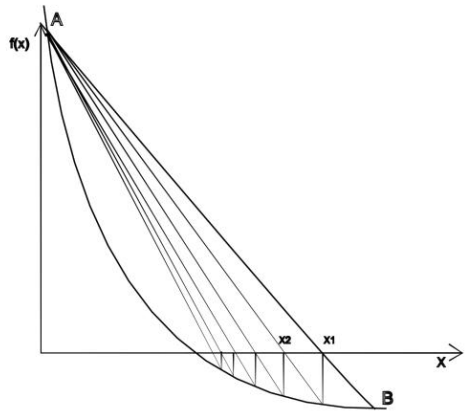

Рисунок 1 - Иллюстрация метода хорд

Уравнение хорды имеет вид:
(x-a)/(b-a)=(y-f(a))/(f(b)-f(a)).
Полагая $x=x^{(1)}$ *и* $y=0$, получаем

$$x^{(1)} = a - \frac{f(a)}{f(b) - f(a)}(b - a).$$

Проверяются знаки *F* и *F”* и фиксируется точка *A* или *B*.

В случае, когда знаки *F* и *F”* одинаковы *x=a* неподвижный конец хорды, то $x^{(0)}=b$:

$$x_{n+1} = x_n - \frac{F(x_n)}{F(x_n)-F(a)}(x_n - a). \quad (1)$$

Если неподвижный конец *x=b*, $x^{(0)}=a$ то:

$$x_{n+1} = x_n - \frac{F(x_n)}{F(b)-F(x_n)}(b - x_n). \quad (2)$$

Для выявления неподвижного конца используется условие $f''(x) \cdot f(t) > 0$, где *t=a* или *t=b*. Если неподвижен конец ***a***, то применяется (1), если конец ***b*** – (2).

Теорема позволяющая оценить погрешность метода.

Теорема. Пусть первая и вторая производные функции f из уравнения f(x)=0 непрерывны и сохраняют постоянный знак, а числа м и М такие, что $0 < m \le |f'(x)| \le M, x \in [a, b]$. *Тогда погрешность приближений к x*, найденных методом хорд, оценивается формулой:*

$$|x* - x_k| \le \frac{M-m}{m}|x_{k+1} - x_k|, \quad k = 1,2,\ldots,$$

где

$$m = \min_{x\in[a,b]} |f'(x)|, M = \max_{x\in[a,b]} |f'(x)|.$$

Методика решения задачи

1. Найти начальный интервал неопределенности $[a_0, b_0]$ одним из методов отделения корней, выбрать x_0 из этого интервала. Задать точность вычислений – малое положительное число ε и присвоить k=0.
2. Определить неподвижный конец отрезка.
3. Вычислить по рекуррентной формуле x_{k+1} через x_k.
4. Если $\frac{M-m}{m}|x_{k+1} - x_k| \le \varepsilon$, решением уравнения - будет $x^*=x_{k+1}$ процесс завершить, иначе *k=k+1* перейти к п.3.

5. Метод простых итераций

Рассмотрим численное уравнение: $y=f(x)$; f - нелинейная функция.

Уравнение необходимо привести к каноническому виду:

$x=\varphi(x)$

Геометрический метод представлен на рисунке 2.

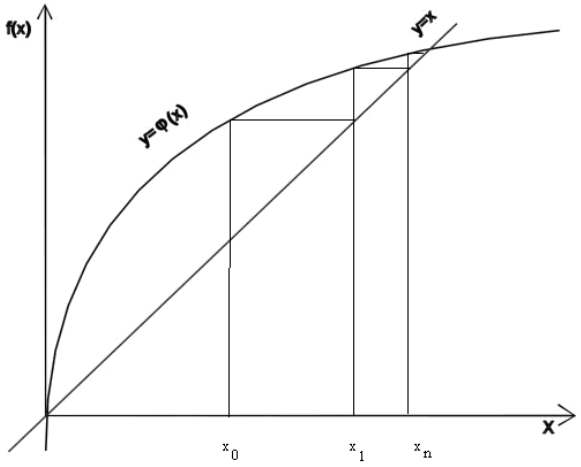

Рисунок 2

Для левой части $y=x$, для правой части $y=\varphi(x)$, там где они пересекаются – решение. На чертеже $y=x$ – биссектриса первого координатного угла.

Решением является точка пересечения биссектрисы и графика функции φ – точка x^*. Таких точек может быть несколько. Взять x_0 – начальное приближение, далее строится процесс одношаговой итерации:

$x_{n+1}=\varphi(x_n)$ $n=0,1,\ldots$

Для оценки погрешности $\varepsilon_n=x_n-x^*$ разложим в ряд Тейлора выражение для $x_{n+1}=\varphi(x_n)$. Таким образом, $x_n=x^*+\varepsilon_n$ и

$$x_{n+1}=x^*+\varepsilon_{n+1}=\varphi(x^*+\varepsilon_n)=\varphi(x^*)+\varepsilon_n\varphi'(x^*)+o(\varepsilon_n).$$

Т.к. $x^*=\varphi(x^*)$ и $o(\varepsilon_n)$ – можно пренебречь, получим:

$$\varepsilon_{n+1}=\varepsilon_n\phi'(x*)$$

1. Если $|\varphi'(x^*)|>1$, тогда сходимости нет.
2. Если $|\varphi'(x^*)|<1$, то $|\varepsilon_{n+1}|<|\varepsilon_n|$ и можно ожидать, что последовательность x_n будет сходиться к x^* со скоростью геометрической прогрессии со знаменателем $q=\varphi'(x^*)$.

a) При $\varphi'(x^*)>0$ ε_{n+1} и ε_n имеют одинаковые знаки, сходимость x_n к x^* монотонная.

b) При $\varphi'(x^*)<0$ знаки ε_{n+1} и ε_n различные, сходимость колебательного характера возле x^*.

c) При $\varphi'(x^*)=0$ ε_{n+1} – будет малой величиной высшего порядка малости в сравнении с ε_n.

Теорема 1*(о сходимости метода простых итераций и единственности получаемого решения)*

Пусть выполнены условия:

1. *Нелинейное уравнение $x=\varphi(x)$ имеет решение x^*, принадлежащее области G.*
2. *отображение $\varphi(x)$ является сжимающим в области G с коэффициентом q*

(<1):

|φ(x'')- φ(x')|<=q|x''-x'|, для любых x'',x' из G.

Тогда:

а) решение x является единственным решением в области G;*

б) последовательность x^k сходится к решению x со скоростью геометрической прогрессии. Т.е. при любом x^0 из условия* $|x^* - x^0| < R$, *где R>0 – некоторое малое число, справедливо неравенство:*

$$|x^* - x^k| \le q^k |x^* - x^0|, k = 0,1,\ldots$$

Теорема 2*(о достаточности условия сходимости метода простых итераций)*

Пусть выполнены условия:

1. *Функция φ(x) имеет производную для всех x из G;*
2. *|φ'(x)|<=q(<1) для всех x из G.*

Тогда φ(x) является сжимающим в g с коэффициентом q и последовательность x(0),…, x(к+1), … сходится к решению x.*

Док-во:

В силу условия 1 справедлива теорема Лагранжа о конечных приращениях:

φ(x'')- φ(x')<=(x''-x') φ'(δ), для всех x',x'' из G, δ из[x',x''].

Из условия 2 получим

|φ(x'')- φ(x')|<=q|x''-x'|, (из теоремы 1) последовательность $x^{(к)}$ сходится к x**, в силу непрерывности *φ(x) в G* имеем место равенство x*=x**.

Положим, что *x*** некоторый корень. Докажем, что *x**=x**. Положим, что корни разные

|φ(x')- φ(x")|≤q|*x'-x"*|,

тогда

*|x**-x*|=|φ(x**)-φ(x*)|*≤q|*x**-x**|,

т.к 0<q<1, то неравенство возможно, если *x**=x**.

Замечание.

1. Метод простых итераций представляет линейный итерационный процесс (метод первого порядка)
2. В силу неоднозначности подбора *(xₖ)* всегда можно подобрать таким образом, чтобы выполнялось условие сходимости. И погрешность ведет себя как члены геометрической прогрессии со знаменателем ***q*** .

Методика решения задачи

1. Уравнение f(x)=0 привести к каноническому виду x=ϕ(x). Для сходимости нужно обеспечить выполнение условия *|φ'(x*)|<=q<1 (q=const).* При этом задача сводится к нахождению абсциссы точки пересечения прямой y=x и кривой y= ϕ(x).
2. Задать начальное приближение $x^{(0)}$, ε – погрешность , к=0.
3. Вычислить следующее приближение:

   $x_{k+1}=\varphi(x_k)$
4. Если $|x^{(k+1)}-x^{(k)}|<=\varepsilon$, итерации завершаются и $x^*=x^{(к+1)}$,

   иначе *k=k+1* и перейти к 3.

   Вместо ε, если известно q можно использовать ·*(1-q)/q*.

6. Метод Ньютона

Метод позволяет свести решение нелинейных уравнений к решению последовательности линейных задач. Его называют также методом касательных.

Метод быстро сходится. Однако этот метод эффективен при весьма жестких ограничениях на характер функций *f(x)*:

1. существование второй производной функции *f(x)* на множестве *G:{a<=x<=b}*;
2. удовлетворения первой производной условию *f`(x)<>0* для всех x принадлежащих G;
3. знакопостоянство *f`(x),f``(x)* для всех *x*, принадлежащих G.

Геометрическая интерпретация метода Ньютона состоит в следующем (рисунок 3).

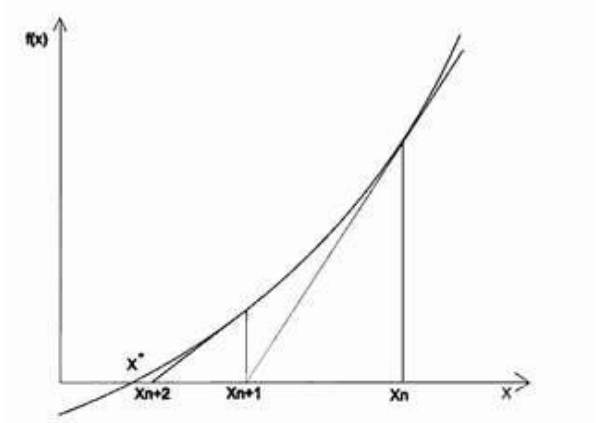

Рисунок 3 - Геометрическая интерпретация метода Ньютона

Задается начальное приближение $x^{(0)}$. Далее проводится касательная к кривой y=f(x) в точке $x^{(0)}$, т.е. кривая заменяется на прямую линию. В качестве следующего приближения выбирается точка пересечения этой касательной с осью абсцисс. Процесс построения касательных и нахождения точек пересечения с осью абсцисс повторяется до тех пор, пока приращение не станет меньше заданной величины ε.

Получим расчетную формулу Ньютона. Рассматриваем уравнение

f(x)=0, (3)

x - численная переменная, *f(x)* - дифференцируемая функция.

Если функция *f(x)* вещественная, то естественно считать начальное приближение x_i тоже вещественным.

Заменим уравнение (3) разложением в ряд Тейлора, ограничиваясь линейной частью:

$$f(x_0) + f\,'(x_0)(x_1 - x_0) = 0 \qquad (4)$$

Отсюда новое приближение:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Аналогично:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0,1,\ldots, \qquad (5)$$

Предполагается, что $f'(x_n) \neq 0$.

Теорема (о достаточном условии сходимости метода Ньютона)

Пусть выполняются условия:

1. *Функция $f(x)$ определена и дважды дифференцируема на [a,b];*
2. *Отрезку [a,b] принадлежит только один простой корень x*, так что $f(a)f(b)<0$;*
1. *Производные $f'(x), f''(x)$ на [a,b] сохраняют знак и $f'(x)<>0$;*
2. *начальное приближение $x^{(0)}$ удовлетворяет неравенству $f(x^{(0)})f''(x^{(0)})>0$(знаки функции и ее второй производной совпадают)*

Тогда с помощью метода Ньютона (5) можно вычислить корень уравнения $f(x)=0$ с любой точностью.

Пусть отрезок $[x^{(k)},x*]$ мал, т.е. итерации выполняются вблизи корня.

Полагая, что условия теоремы выполнены, разложим $f(x)$ в окрестности корня $x*$ до члена второго порядка.

$$f(x)\big|x = x^{(k)} = f(x^{(k)}) + (x* - x^{(k)})f'(x^{(k)}) + (x* - x^{(k)})^2\frac{f''(\xi)}{2} = 0$$

где $\xi \in (x^{(k)} - \delta, x^{(k)} + \delta)$. ($\delta$ – малая величина).

Далее:

$$x* = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} - \frac{(x* - x^{(k)})^2}{2f'(x^{(k)})}f''(\zeta)$$

Тогда

$$x* - x^{(k+1)} = -\frac{(x* - x^{(k)})^2}{2f'(x^{(k)})}f''(\xi)$$

Из последнего соотношения можно сделать оценку погрешности (к+1) приближения через погрешность к-го приближения:

$$\left|x* - x^{(k+1)}\right| \le \frac{M_2}{2m_1}\left|x* - x^{(k)}\right|^2, \qquad (6)$$

где $M_2 = max|f''(x)|,\ m_1 = min|f'(x)|$.

Оценка свидетельствует о квадратичной сходимости метода касательных вблизи корня. С вычислительной точки зрения это означает, что на каждом приближении количество верных цифр удваивается.

Очевидно, ошибка на каждом шаге убывает, если

$$\frac{1}{2}\frac{M_2}{m_1}|x_0 - x*| < 1$$

Полученное условие означает, что сходимость последующих приближений зависит от выбора начального приближения. Выполнения данного условия можно добиться за счет более аккуратной локализации корня.

Метод Ньютона является локально сходящимся, так как он сходится с определенной скоростью к истинному решению при условии, что стартует в достаточной близости от этого решения.

Методика решения задачи:

1. Задать начальные приближения $x^{(0)}$ так, чтобы выполнялось неравенство $f(x^{(0)})*f''(x^{(0)})>0$, а также малое положительное число ε. Положить k=0.
2. Вычислить $x^{(k+1)}$ по формуле: $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.
3. Если $|x_{k+1} - x_k| < \varepsilon$ процесс завершить и положить $x*=x^{(k+1)}$, иначе, $k=k+1$ и перейти к пункту 2.

7. Модифицированный упрощенный метод Ньютона

Здесь производная берется только в одной точке, вычислений меньше, но процесс сходится медленнее.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)} \qquad (7)$$

8. Комбинированный метод секущих – хорд

Данный метод гарантирует сходимость при выборе 2-х приближений x_0 и x_1. Можно вместо вычисления производной на каждом шаге заменить её приближенным значением:

$$f'(x_n) \approx \frac{f(x_n+\Delta x)-f(x_n)}{\Delta x} \quad (8)$$

Как и в модифицированном методе Ньютона, производная заменяется её приближением:

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{F(x_n)-F(x_{n-1})} F(x_n) \quad (9)$$

Проверка на завершение процесса:

$| x_{n+1} - x_n | <$ □

Метод секущих уступает методу Ньютона по скорости сходимости, однако не требует вычисления производной, которая вычисляется приближенно.

9. Метод Ньютона – Бройдена

Этот метод позволяет увеличить скорость сходимости последовательности приближений

$$x_{n+1} = x_n - c_n \frac{f(x_n)}{f'(x_n)}, \quad (9)$$

где c_n – выбирается на каждом шаге итерации, чтобы уменьшить значение по модулю $|f(x_{k+1})|$ по сравнению с $|f(x_k)|$. Таким образом, при $c_n=1$ получим метод Ньютона, В случае плохой сходимости выбираем $0<c_n<1$, при хорошей сходимости коэффициент $c_n>1$.

ЗАДАНИЕ

Отделите корни заданного уравнения, согласно варианту из табл.1, и уточните их одним из методов с точностью до ε=10^{-4}. Решить уравнения методом половинного деления, методом Ньютона и методом простых итераций. Либо другими методами, перечисленными в теоретической части к заданию 1.

Таблица 1

Уравнения по вариантам

1	$sin(x\ \ +1)-x\ \ -1=0$	9	$\boldsymbol{cos(x\ \ +0.5)-x=2}$
2	$2x+tg(x)=0$	10	$\sin x-2x=1$
3	$(x-6)^2+ln\,x=0$	11	$cos\,x-(x-1)^2=0$
4	$x\,ln(x+2)=2$	12	$x\,ln(x+1)=1$
5	$2\,ln\,x-0.5x+1=0$	13	$2x\ \ +cos(x\ \ )=0$
6	$x^2-3\,sin\,x=0$	14	$x^2+e^x=2$
7	$sin(x+1)=0.2x$	15	$5\,sin\,x=x$
8	$cos(x+0.3)=x^2$	16	$x\,ln(x+1)=1$

Порядок выполнения работы

Отделите графически все корни уравнения.

1. Составьте методику решения нелинейного уравнения для метода половинного деления, метода Ньютона, метода простых итераций.
2. Составьте программу уточнения корня методом половинного деления с точностью до ε, выводящую результаты в таблицу:

N	a_n	b_n	$b_n - a_n$
…	…	…	…

где a_n b_n – концы вложенных отрезков.

3. Составьте программу уточнения корня методом Ньютона и методом простых итераций с точностью до ε, выводящую результаты в таблицу:

N	x_n	x_{n+1}	$x_{n+1} - x_n$
…	…	…	…

4. Найдите все корни уравнения и выпишите их с верными знаками.
5. Сравнить методы решения нелинейных уравнений по скорости сходимости.

Второе задание
Вычисления в одномерных массивах

Таблица 2

Алгоритмы генерации по вариантам

Вариант	Алгоритм
1.	ranlux48
2.	knuth_b
3.	random_device
4.	minstd_rand
5.	minstd_rand0
6.	mt19937
7.	mt19937_64
8.	ranlux24_base
9.	ranlux48_base
10.	ranlux24

Вариант 1

1. Напишите программу, в которой определен массив из n чисел ($n \geq 10$) и инициализирован целыми случайными числами из диапазона [0, 100].
2. Поменяйте местами минимальный и второй максимальный элементы местами в массиве из пункта 1.
3. Напишите программу, модифицирующую массив, как в примере:

   [35, 39, 89, 37, 96] → [96, 35, 39, 89, 37].
4. Вывести на экран массив длиной $N \times 10$ таким образом, как указано на рисунках. Первый элемент массива равен 10. Необходимо выводить элементы строками по 10 элементов.

Вывод 1

N = 5

```
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
```

N = 7

```
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
```

Вывод 2

```
50 51 52 53 54 55 56 57 58 59
49 48 47 46 45 44 43 42 41 40
30 31 32 33 34 35 36 37 38 39
29 28 27 26 25 24 23 22 21 20
10 11 12 13 14 15 16 17 18 19
```

```
70 71 72 73 74 75 76 77 78 79
69 68 67 66 65 64 63 62 61 60
50 51 52 53 54 55 56 57 58 59
49 48 47 46 45 44 43 42 41 40
30 31 32 33 34 35 36 37 38 39
29 28 27 26 25 24 23 22 21 20
10 11 12 13 14 15 16 17 18 19
```

Вариант 2

1. Напишите программу, в которой определен массив $arr1$ из n чисел ($15 \leq n \leq 30$) и инициализирован целыми случайными числами из диапазона [10,

30]. Определите массив $arr2$ из 5 чисел и инициализируйте массив $arr2$ целыми случайными числами из диапазона [10, 30].

2. Выведите количество чётных элементов массива $arr1$ на нечётных местах, а также сами элементы.
3. Определите, какие числа из массива $arr2$ встречаются в массиве $arr1$. Выведите эти числа и сколько раз они встречаются в массиве $arr1$. Если ни одно число не встречается, то выведите соответствующее сообщение.
4. Напишите программу, модифицирующую массив случайных символов размера n , как в примере:

[v,!,N,R,(,3] → [!,N,R,(,3,v].

5. Определите и инициализируйте массив, состоящий из случайных целых чисел, входящих в диапазон [1000, 9000]. Вычислите сумму элементов массива. Напишите программу, меняющую местами цифры в элементах массива в порядке возрастания (например, 8315 → 1358). Отсортируйте массив в порядке возрастания. Вычислите сумму элементов массива и сравните с суммой элементов исходного массива.

Вариант 3

1. Напишите программу, в которой определен массив из n чисел ($n \geq 10$) и инициализирован случайными числами из диапазона [100, 200].
2. Найдите второй по величине элемент и сумму элементов между минимальным и вторым по величине элементами массива из пункта 1.
3. Определите два массива из n чисел ($n \geq 10$) и инициализируйте случайными числами из диапазона [-50, 50]. Заполните новый массив таким образом, чтобы значение каждого элемента нового массива равнялось сумме элементов на чётных местах и разнице элементов на нечётных местах соответствующих элементов заданных массивов.
4. Задан массив. Напишите программу, определяющую, сколько раз повторяется каждый из элементов массива.
5. Напишите программу, которая принимает диапазон лет, а возвращает массив, содержащий високосные года в заданном диапазоне.

Вариант 4

1. Напишите программу, в которой определен массив из n чисел ($n \geq 10$) и инициализирован случайными числами из диапазона [10, 100].
2. Найдите два наименьших элемента массива и вычислите сумму остатков получаемых от деления элементов массива из пункта 1 на минимальный элемент.
3. Определите массив. Найдите самую длинную возрастающую последовательность чисел в массиве. Определите и инициализируйте новый массив найденной последовательностью.
4. Вывести на экран массив длиной $N \times 8$ таким образом, как указано на рисунках. Первый элемент массива равен 2000. Необходимо выводить элементы строками по 8 элементов.

Вывод 1

N = 5

```
2000 2010 2020 2030 2040 2050 2060 2070
2080 2090 2100 2110 2120 2130 2140 2150
2160 2170 2180 2190 2200 2210 2220 2230
2240 2250 2260 2270 2280 2290 2300 2310
2320 2330 2340 2350 2360 2370 2380 2390
```

N = 10

```
2000 2010 2020 2030 2040 2050 2060 2070
2080 2090 2100 2110 2120 2130 2140 2150
2160 2170 2180 2190 2200 2210 2220 2230
2240 2250 2260 2270 2280 2290 2300 2310
2320 2330 2340 2350 2360 2370 2380 2390
2400 2410 2420 2430 2440 2450 2460 2470
2480 2490 2500 2510 2520 2530 2540 2550
2560 2570 2580 2590 2600 2610 2620 2630
2640 2650 2660 2670 2680 2690 2700 2710
2720 2730 2740 2750 2760 2770 2780 2790
```

Вывод 2

```
2390 2380 2370 2360 2350 2340 2330 2320
2240 2250 2260 2270 2280 2290 2300 2310
2230 2220 2210 2200 2190 2180 2170 2160
2080 2090 2100 2110 2120 2130 2140 2150
2070 2060 2050 2040 2030 2020 2010 2000
```

```
2720 2730 2740 2750 2760 2770 2780 2790
2710 2700 2690 2680 2670 2660 2650 2640
2560 2570 2580 2590 2600 2610 2620 2630
2550 2540 2530 2520 2510 2500 2490 2480
2400 2410 2420 2430 2440 2450 2460 2470
2390 2380 2370 2360 2350 2340 2330 2320
2240 2250 2260 2270 2280 2290 2300 2310
2230 2220 2210 2200 2190 2180 2170 2160
2080 2090 2100 2110 2120 2130 2140 2150
2070 2060 2050 2040 2030 2020 2010 2000
```

Вариант 5

1. Напишите программу, в которой определен массив из n чисел ($n \geq 10$) и инициализирован случайными вещественными числами из диапазона [0, 100].
2. Найдите среднее значение, как отношение суммы всех элементов на их количество, и вычислите сумму квадратов разностей между элементами массива из пункта 1 и полученным средним значением.
3. Поменяйте местами первый и последний разряды чётных элементов с чётным номером из пункта 1.
4. Напишите программу, модифицирующую массив, как в примере:

[25, 73, 100, 22, 82] → [73, 100, 22, 82, 25].

5. Определите и инициализируйте массив из n чисел ($n \geq 50$) случайными целыми числами из диапазона [-10, 10]. Определите, какие числа сколько раз встречаются в массиве. Удалите из массива повторяющиеся элементы.

Вариант 6

1. Напишите программу, в которой определен массив из n чисел ($n \geq 10$) и инициализирован случайными вещественными числами из диапазона [-100, 100].
2. Найдите количество положительных элементов массива из пункта 1 и вычислите сумму элементов, расположенных после последнего элемента, равного нулю.
3. Найдите самую длинную возрастающую последовательность чисел. Определите и инициализируйте новый массив найденной последовательностью.
4. Вывести на экран массив длиной $N \times 6$ таким образом, как указано на рисунках. Первый элемент массива равен 100. Необходимо выводить элементы строками по 6 элементов.

	Вывод 1	*Вывод 2*
N = 4	100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123	105 104 103 102 101 100 106 107 108 109 110 111 117 116 115 114 113 112 118 119 120 121 122 123
N = 8	100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147	105 104 103 102 101 100 106 107 108 109 110 111 117 116 115 114 113 112 118 119 120 121 122 123 129 128 127 126 125 124 130 131 132 133 134 135 141 140 139 138 137 136 142 143 144 145 146 147

Вариант 7

1. Напишите программу, в которой определен массив из n чисел ($n \geq 10$) и инициализирован случайными вещественными числами из диапазона [-100, 100].
2. Вычислите произведение отрицательных элементов и сумму положительных элементов массива из пункта 1, расположенных до максимального элемента.

3. Напишите программу, которая заменяет регистр каждого символа строки на противоположный.
4. Напишите программу, определяющую количество элементов массива, значение которых больше соседних элементов.
5. Определите и инициализируйте массив, состоящий из случайных целых чисел. Вычислите сумму элементов массива. Напишите программу, меняющую местами цифры в элементах массива в случайном порядке. Отсортируйте массив в порядке возрастания. Вычислите сумму элементов массива и сравните с суммой элементов исходного массива.

Вариант 8

1. Напишите программу, в которой определен массив из n целых чисел ($n \geq 10$). Инициализируйте массив случайными числами из диапазона [150, 300].
2. Найдите и выведите самую длинную последовательность чисел массива из пункта 1, упорядоченную по убыванию.
3. Определите и инициализируйте новый массив, состоящий из чисел, меньших среднеарифметического значения массива из пункта 1.
4. Напишите программу, модифицирующую массив случайных символов размера n ($n \geq 5$), как в примере:

   [0,I,D,Q,I,A] → [D,Q,I,A,0,I].
5. Определите и инициализируйте массив, состоящий из случайных целых чисел, входящих в диапазон [100, 900]. Отсортируйте новый массив по убыванию. Определите, сколько раз каждая цифра встречается в элементах массива.

Вариант 9

1. Напишите программу, в которой определен массив из n целых чисел ($n \geq 10$). Инициализируйте массив случайными целыми числами из диапазона [-100, 100].
2. Найдите среднее арифметическое отрицательных и положительных элементов массива из пункта 1. Поменяйте местами максимальный и минимальный элементы массива.

3. Найдите и выведите убывающую последовательность элементов массива из пункта 1.
4. Вывести на экран массив длиной $N \times 5$ таким образом, как указано на рисунках. Первый элемент массива равен 1000. Необходимо выводить элементы строками по 5 элементов.

Вывод

N = 4

```
1410 1306 1203 1101 1000
1945 1836 1728 1621 1515
2505 2391 2278 2166 2055
3090 2971 2853 2736 2620
```

N = 8

```
1410 1306 1203 1101 1000
1945 1836 1728 1621 1515
2505 2391 2278 2166 2055
3090 2971 2853 2736 2620
3700 3576 3453 3331 3210
4335 4206 4078 3951 3825
4995 4861 4728 4596 4465
5680 5541 5403 5266 5130
```

Вариант 10

1. Напишите программу, в которой определен массив из n чисел ($n \geq 20$) и инициализирован случайными вещественными числами из диапазона [-250, 250].
2. Найдите количество отрицательных элементов массива из пункта 1 и сумму модулей элементов, расположенных после минимального по модулю элемента.
3. Найдите самую длинную убывающую последовательность чисел в массиве массива из пункта 1. Выведите последовательность, индексы начала и конца.
4. Определите и инициализируйте новый массив самой длинной возрастающей последовательностью из предыдущего пункта, записанной наоборот.
5. Напишите программу, в которой определены два массива и инициализированы случайными числами. Модифицируйте массивы путём циклического сдвига на один элемент вправо синхронно в обоих массивах (например, [1, 2, 3, 4, 5] и [1, 2, 3] → [3, 1, 2, 3, 4] и [5, 1, 2]).

Вариант 11

1. Напишите программу, в которой определен массив из n чисел ($n \geq 10$) и инициализирован случайными вещественными числами из диапазона [100, 900].
2. Введите некоторое число A и найдите количество элементов массива из пункта 1, больших A. Вычислите произведение элементов, расположенных после максимального по модулю элемента.
3. Поменяйте в массиве из пункта 1 местами попарно соседние элементы, если в элементе с чётным номером количество десятков больше, чем количество сотен. Если количество элементов нечётное, последний элемент не рассматривается.
4. Определите и инициализируйте массив из n чисел ($n \geq 15$) случайными целыми числами из диапазона [10, 20]. Определите, какой из элементов повторяется наибольшее количество раз.
5. Определите и инициализируйте массив случайными числами. Вычислите сумму элементов массива. Напишите программу, меняющую местами цифры в элементах массива в порядке убывания (например, 9164 → 9641). Отсортируйте массив в порядке убывания. Вычислите сумму элементов массива и сравните с суммой элементов исходного массива.

Вариант 12

1. Напишите программу, в которой определены массивы $arr1$ и $arr2$, состоящие из n элементов ($n \geq 10$, n – чётное число), и инициализированы случайными целыми числами из диапазона [10, 100].
2. Выведите элементы массивов массива из пункта 1, которые являются простыми числами из диапазона [0, 10].
3. Отсортируйте первую половину массива $arr1$ по возрастанию и вторую половину массива $arr2$ по убыванию. Модифицируйте массивы таким образом, чтобы в одном из массивов оказались только отсортированные значения, а во втором – несортированные. То есть необходимо скопировать, например, первую половину массива $arr1$ в несортированную часть массива $arr2$, а несортированную часть массива $arr2$ в сортированную $arr1$.

4. Определите и инициализируйте массив, состоящий из случайных символов. Напишите программу, заменяющую в массиве строчные буквы на заглавные, а заглавные на строчные.
5. Определите и инициализируйте массив, состоящий из случайных целых чисел, входящих в диапазон [1000, 2000]. Отсортируйте массив по убыванию. Создайте новый массив, содержащий элементы исходного массива, в которых цифры повторяются.

Вариант 13

1. Напишите программу, в которой определен массив $arr1$ из n чисел ($n \geq 10$) и инициализирован целыми случайными числами из диапазона [0, 100].
2. Найдите сумму элементов массива $arr1$ между наибольшим и наименьшим элементами.
3. Определите и инициализируйте новый массив $arr2$ элементами, расположенными между наибольшим и наименьшим элементом массива $arr1$, массив $arr2$ должен иметь такую же размерность, что и $arr1$, поэтому дополните свободные места случайными элементами из массива $arr1$.
4. Определите и инициализируйте массив, состоящий из случайных символов. Напишите программу, сортирующую в порядке возрастания элементы на чётных местах.
5. Определите и инициализируйте массив, состоящий из случайных целых чисел, входящих в диапазон [100, 900]. Отсортируйте массив по возрастанию. Определите новый массив. Поменяйте цифры в элементах исходного массива в случайном порядке. Если число окажется больше исходного, то добавьте его в новый массив, если нет – вместо модифицированного числа запишите в новый массив 0.

Вариант 14

1. Напишите программу, в которой определен массив 1 из n чисел (n чётное число), инициализированный числами из диапазона [100, 900], и массив 2 из n / 2 чисел, инициализированный числами, принадлежащими диапазону [2, 50].
2. Выведите чётные элементы массива 1 на нечётных местах и нечётные элементы массива 2 на чётных местах.

3. Определите и инициализируйте массив из k чисел ($k \geq 10$), являющимися результатом целочисленного деления случайного элемента из массива 1 на случайный элемент из массива 2.
4. Определите и инициализируйте массив, состоящий из случайных символов. Напишите программу, удаляющую из массива заглавные буквы.
5. Определите и инициализируйте массив, состоящий из случайных целых чисел, входящих в диапазон [0, 300]. Отсортируйте массив по возрастанию. Создайте новый массив, содержащий элементы исходного массива, в которых цифры повторяются дважды.

Вариант 15

1. Напишите программу, в которой определен массив из n целых чисел ($n \geq 10$). Инициализируйте массив случайными целыми числами из диапазонов [-300, -150] и [150, 300].
2. Вычислите среднее арифметическое элементов массива из пункта 1, наибольший и наименьшей элементы.
3. Переставьте элементы массива из пункта 1 наоборот (дополнительный массив не используйте). Отсортируйте в порядке возрастания первую половину массива из пункта 1, а вторую – в порядке убывания.
4. Определите и инициализируйте массив, состоящий из случайных символов. Напишите программу, удаляющую из массива цифры и строчные буквы.
5. Напишите программу, в которой определены два массива и инициализированы случайными числами. Модифицируйте массивы путём циклического сдвига на один элемент влево синхронно в обоих массивах (например, даны два массива [1, 2, 3] и [4, 5, 6], после преобразования получится [2, 3, 4] и [5, 6, 1].

Третье задание
Генерация псевдослучайных чисел

Задание выполняется на языке C++. Для генерации чисел использовать **алгоритмы (таблица 4):**

Таблица 4

Алгоритмы по вариантам

Вариант	Алгоритм
1	RC4
2	Xorshift
3	Генератор Фибоначчи с запаздыванием
4	Генератор псевдослучайных чисел на основе алгоритма BBS
5	Линейный конгруэнтный генератор

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1. Линейный конгруэнтный генератор.

Последовательность чисел $X_1, X_2, \ldots$, определена как

$$X_i = (A*X_{i-1}+B) \bmod C,$$

где C - диапазон значений, при этом C > 0;

A - множитель (0 <= A <= C);

B - инкрементирующее значение (0 <= B <= C).

Также задается значение seed:

X_0 - начальное значение (0 <= X_0 < C).

Так как числа последовательности образуются с использованием деления по модулю, в какой-то момент последовательность начнет повторяться. Необходимо вывести сгенерированную последовательность и найти индекс, с которого начинается повторение последовательности.

Пример 1

X_0 = 2, A = 3, B = 5, C = 10

Вывод:

2 1 8 9 2 1 8 9 2 1

5

Пример 2

X_0 = 5, A = 3, B = 3, C = 7

Вывод:

5 4 1 6 0 3 5 4 1 6

7

2. Генератор псевдослучайных чисел на основе алгоритма BBS

Алгоритм генерации псевдослучайных чисел, называемый алгоритмом BBS или генератором с квадратичным остатком, был предложен для целей криптографии в 1986 году.

Вычисляется стартовое число генератора $x_0 = x^2 \bmod M$,
где M = p * q (целое число Блюма);

p и q – два больших простых числа, сравнимые с 3 по модулю 4;

x – случайное целое число, взаимно простое с M.

Последовательность чисел $x_1, x_2, \ldots$, определена как

$x_{n+1} = x_n^2 \bmod M$.

Интересным является то, что x_n можно получить по формуле

$$x_n = x_0^{2^n \bmod ((p-1)(q-1))} \bmod M$$

Пример

p = 19, q = 23, M = 19*23 = 437, x = 7.

Результат

49 216 334 121 220

3. Генератор Фибоначчи с запаздыванием

Известны разные схемы использования метода Фибоначчи с запаздыванием. Будем использовать следующую:

$$S_n = S_{n-j} \$ S_{n-k} \pmod m,\ 0 < j < k.$$

где S – вещественное число;

j, k – целые положительные числа, параметры генератора;

$ - операция сложения/вычитания/умножения/XOR.

m - обычно имеет степень 2.

Пример:

Вычислить последовательность чисел, генерируемую аддитивным методом Фибоначчи с запаздыванием начиная с k2 при следующих исходных данных:

j = 2, k = 1, k0 = 1; k1 = 4, m = 16.

Результат

5 9 14 7 5 12

Варианты выполнения задания. Допускается реализация четырех операций (+,-,*,xor). Либо возможно придумать свою схему, например, вида (или любую другую):

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & если\ k_{i-a} \geq k_{i-b} \\ k_{i-a} - k_{i-b} + 1, & если\ k_{i-a} < k_{i-b} \end{cases}.$$

4. Xorshift

Генераторы случайных чисел Xorshift, также называемые генераторами сдвигового регистра, представляют собой подмножество регистров сдвига с линейной обратной связью (LFSR), которые обеспечивают особенно эффективную реализацию в программном обеспечении без чрезмерного использования разреженных полиномов. Они генерируют следующее число в своей последовательности, многократно принимая исключающее или числа с битовой версией самого себя.

5. RC4

RC4 – фактически класс алгоритмов, определяемых размером его блока или слова – параметром n. Два счетчика назовем i и j. Все вычисления проводятся по модулю 2n.

Алгоритм RC4 состоит из двух этапов. На первом, подготовительном этапе производится инициализация таблицы замен S. На втором, основном этапе вычисляются псевдослучайные числа.

Инициализация таблиц

```
for i from 0 to 2^n-1
    S[i] := i
endfor
j := 0
for i from 0 to 2^n-1
```

```
    j := (j + S[i] + K[i]) mod 2ⁿ
    swap values of S[i] and S[j]
endfor
Алгоритм псевдослучайной генерации
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 2ⁿ
    j := (j + S[i]) mod 2ⁿ
    swap values of S[i] and S[j]
    t := (S[i] + S[j]) mod 2ⁿ
    K := S[t]
    output K
endwhile
```

https://intuit.ru/studies/courses/691/547/lecture/12385?page=2

Четвертое задание
Теория голосований

На выборах участвуют **n** кандидатов и **k** избирателей. Каждый избиратель формирует полное ранжирование (цепочку) кандидатов от самого предпочтительного до наименее предпочтительного. Например, если кандидаты – Петя, Вася и Оля, то голос вида «Вася – Оля – Петя» означает, что для данного избирателя кандидат Вася имеет наивысший приоритет, а Петя – наименьший.

Необходимо определить победителя выборов, используя два алгоритма: метод Борда и метод Кондорсе.

Метод Борда

Каждому кандидату начисляются баллы в зависимости от позиции в списке каждого избирателя. Например, если всего **n** кандидатов, то кандидат, стоящий на первом месте, получает **(n – 1)** баллов, на втором – **(n – 2)** и так далее до **0** баллов за последнее место. Побеждает кандидат, набравший суммарно наибольшее количество баллов.

Метод Кондорсе

Для каждого кандидата проводится парное сравнение с каждым другим кандидатом. Если кандидат A выигрывает против кандидата B в большинстве прямых сравнений (то есть оказывается выше в списках большинства избирателей), то A считается «победителем» в парном сравнении. Победителем по методу Кондорсе является кандидат, который выигрывает в парных встречах со всеми остальными кандидатами (если такой кандидат существует).

Программа должна вывести победителя согласно каждому методу. Если результаты разнятся – можно добавить комментарий о том, что разные алгоритмы могут давать разные результаты в зависимости от специфики голосования.

Протестируйте программу на различных наборах данных, в том числе на крайних случаях (например, когда все избиратели отдают одинаковые предпочтения или когда не существует кандидата, удовлетворяющего условию Кондорсе). В заключительной части лабораторной работы опишите, в каких случаях методы могут давать разные результаты и какие недостатки имеет каждый метод.

Подробное описание алгоритмов можно найти в статье:

- Материалы по экономике и голосованию (раздел 3)

Пятое задание*
Дилемма заключенного

Требуется реализовать игру «Предать или сотрудничать» и реализовать 3 алгоритма поведения в игре.

Игра состоит из случайного кол-ва раундов от 100 до 200 (итоговое кол-во раундов при каждом запуске игры генерируется случайно). На протяжении игровой сессии сражаются 2 алгоритма. В каждом раунде каждый алгоритм выбирает, либо сотрудничество, либо предательство. Если алгоритм А выбирает предательство и алгоритм Б выбирает предательство они получают по 4 очка. Если алгоритм А выбирает сотрудничество, а алгоритм Б выбирает предательство - алгоритм А получает 0 очков, а алгоритм Б получает 20 очков. Если оба алгоритма выбирают сотрудничество оба получают 24 очка. Каждому алгоритму в каждом раунде известны результаты всех предыдуще раундов текущей игровой сессии, на основе этих данных алгоритм может выбирать будет он сотрудничать или предаст.

Каждый алгоритм должен представлять из себя функцию с сигнатурой:

boolean func(int32 round_number, array[boolean] self_choices, array[boolean] enemy_choices)

round_number – номер текущего раунда

self_choices – массив булевых значений, содержит информацию о собственных выборов (предать или сотрудничать) за все предыдущие раунды

enemy_choices - массив булевых значений, содержит информацию о выборов (предать или сотрудничать) противника за все предыдущие раунды

true – сотрудннчество

false – предательство

Каждому студенту нужно реализовать 3 алгоритма и запустить игру 3 раза чтобы выявить лучший алгоритм. Лучший алгоритм будет сражаться в финальной битве между всеми студентами группы на очном занятии.